SIAM J. COMPUT.
Vol. 21, No. 4, pp. 671-696, August 1992

004

AN OPTIMAL ALGORITHM FOR INTERSECTING THREE-DIMENSIONAL CONVEX POLYHEDRA*

BERNARD CHAZELLE†

Abstract. This paper describes a linear-time algorithm for computing the intersection of two convex polyhedra in 3-space. Applications of this result to computing intersections, convex hulls, and Voronoi diagrams are also given.

Key words. computational geometry, convex polyhedra

AMS(MOS) subject classifications. 68Q25, 68H05

1. Introduction. Given two convex polyhedra in 3-space, how fast can we compute their intersection? Over a decade ago, Muller and Preparata [22] gave the first efficient solution to this problem by reducing it to a combination of intersection detection and convex hull computation. Another route was followed in 1984 by Hertel et al., who solved the problem by using space sweep [16]. In both cases, a running time of $\Theta(n \log n)$ was achieved, where n is the combined number of vertices in the two polyhedra. Resolving the true complexity of the problem, however, remained elusive.

The different but related problem of *detecting* whether two convex polyhedra intersect, by using preprocessing, was studied by Chazelle and Dobkin [5], Dobkin and Munro [9], and Dobkin and Kirkpatrick [6]. More germane to our concerns here is the off-line version of the detection problem. Dobkin and Kirkpatrick [7] have shown that detecting whether two convex polyhedra intersect can be done in a linear number of operations. By stating the problem as a linear program over three variables, other linear-time algorithms originate in the works of Megiddo [19] and Dyer [10]. Previous results also include an efficient algorithm for intersecting two polyhedra, one of which is convex (Mehlhorn and Simon [21]). Optimal solutions for intersecting convex polygons are given in Shamos and Hoey [27] and O'Rourke et al. [23]. For additional background material on polyhedral intersections, the reader should consult Edelsbrunner [11], Mehlhorn [20], and Preparata and Shamos [25].

Our main result is an algorithm for constructing the intersection between two convex polyhedra in linear time. The algorithm does not use any complicated data structure and seems a good candidate for practical implementation. As is customary, our result assumes that the input conforms with any one of the standard (linear-time equivalent) polyhedral representations given in the literature [3], [15], [22]. This is not a minor point, because nonstandard representations can easily make the problem more difficult. (For example, think how much more difficult the problem would be if we were given only the vertices without any other facial information.) From our algorithm for pairwise intersections we immediately derive an efficient method for intersecting k convex polyhedra. The complexity of the algorithm is $O(n \log k)$, where n is the total number of vertices, which is provably optimal. Other applications include merging Voronoi diagrams in two dimensions and computing convex hulls in 3-space.

2. Polyhedra and shields. At the heart of Dobkin and Kirkpatrick's detection [6] and separation algorithms [7] is an ingenious hierarchical representation of a convex

* Received by the editors February 15, 1989; accepted for publication (in revised form) July 16, 1991. The author wishes to acknowledge the National Science Foundation for supporting this research in part under grant CCR-8700917.

† Department of Computer Science, Princeton University, Princeton, New Jersey 08544.

polyhedron. Further applications of that versatile data structure have been given in [12], [21]. The representation can be seen as a specialization of Kirkpatrick's point location structure [18]. A convex polyhedron P of n vertices is made the first element of a descending chain of $O(\log n)$ nested convex polyhedra, such that the last one has a constant number of vertices and the others differ from their immediate predecessors by shelling off small, disjoint polyhedral cones. We need to go further and modify this hierarchy of polyhedra in several ways.

First, we represent the set of nested polyhedra as a single geometric object, namely, a simplicial cell complex, so we can walk freely from one to the next. In this context, walking means being able to trace a polygonal curve in 3-space within the hierarchy in time proportional to the size of the curve (i.e., its number of vertices) and the number of cells (counting multiplicities) crossed by the curve. Thus, if a curve lies inside P and connects two points on the boundary, we can go from one endpoint to the other while keeping track of where we are within the hierarchy, all of this in time proportional to the size of the curve and the number of cells crossed. If the curve is a straight-line segment, then because of convexity the time becomes $O(\log n)$.

This data structure is still insufficient for our purposes, because sometimes we will need to follow a curve that leaves P and later re-enters the polyhedron from the outside. To trace the curve after we leave P we need a hierarchy for the "outside" of P as well. Unfortunately, the outside of a convex polyhedron is not convex, so we cannot apply the Dobkin-Kirkpatrick construction verbatim. Instead, we switch to dual space because the set of planes that avoid P gets mapped to a convex polyhedron. So, we now have two nested sequences of $O(\log n)$ polyhedra, one fitting inside P and the other fitting inside its dual. The resulting data structure is called the *shield* of P: It consists of a primal part, which allows us to navigate inside P, and a dual part, which, we hope, allows us excursions outside. The latter is true, but in an indirect way. Indeed, to trace a curve that connects two points on the boundary of P and lies outside the polyhedron is still not very easy. But, instead, consider a finite sequence of planes, all of which lie outside P, except for the first and last ones, which are tangent to P. We can visualize this sequence mechanically by starting with the first plane and pivoting along the appropriate line to get to the second plane, and so on, until we reach the position of the last plane. Dually, this motion corresponds to the traversal of a polygonal curve inside the dual of P that connects two points on the boundary (namely, the duals of the first and last planes in the sequence). Because of the dual hierarchy we are thus able to trace this curve, which, in primal space, means "tracing" the corresponding sequence of planes. These operations will allow us to navigate inside and outside P and discover the points where we leave and re-enter the polyhedron. The navigation inside P follows polygonal curves, while the one outside P follows sequences of pivoting planes.

As it turns out, we cannot afford to keep the full contents of the hierachies: Only their outermost layers can be used. Thus, we discard all but a constant number, say k, of the nested polyhedra. The result is a geometric structure that we call the *k-shield* of P. Intuitively, we cannot afford to traverse either hierarchy all the way across because we would pay a factor of $\log n$ time in doing so, and this overhead would result in an $O(n \log n)$-time intersection algorithm. Of course, we are now missing so much information that navigating in a k-shield is rather difficult. However, we can use a well-tuned form of recursion to get around this problem.

Informally, the linear algorithm works as follows: We check that both input polyhedra P and Q have a point in common and we compute their k-shields, for some appropriate constant k. Then we begin to traverse the edges of one of the polyhedra,

say P, and while doing so we keep track of where we are in the primal part of the k-shield of Q (assuming that we start somewhere inside Q). This is called *broadcasting* from P. As long as we navigate inside Q we can use the primal part of its k-shield to guide us. When we reach portions of the boundary of P that lie outside Q, however, we must switch to dual space and use the dual part of the shield to guide the navigation. A transition from primal to dual space is called a *mutation*: It involves changing the mode of navigation from one that traces a polygonal curve within a shield to one that follows a sequence of pivoting planes or, equivalently, one that traces a polygonal curve in the dual part of the shield. Unfortunately, a mutation cannot be carried out instantaneously and requires a little bit of geometric work.

A yet more serious difficulty is what to do when we reach the last layer L in either one of the hierarchies of Q, say the primal one, and we need to go deeper to carry on the navigation. Recall that most of the inner layers of the shield have been removed and, thus, tracing a polygonal curve all across the hierarchy is not possible. When this happens we call upon the intersection algorithm recursively with P and L as input and thus discover, in this indirect manner, the tracing pattern along L. In other words, we use recursion to palliate the lack of inner layers. What makes this idea work is that as we do so we also switch from a broadcasting from P to a broadcasting from L. This switching trick is actually the key to breaking the $n \log n$ barrier. Indeed, this leads to a recurrence on the running time $T(n)$ of the form $T(n) = 4T(n/5) + O(n)$, which solves to linear.

An interesting side effect is that because we operate in both primal and dual spaces, the algorithm ends up computing the intersection, as well as the convex hull, of the two input polyhedra. Actually, we keep switching between these two tasks in a co-routine-like fashion. Although the algorithm is not particularly complicated, proving its correctness requires a certain amount of thoroughness in investigating the topology of several convex polyhedra. The fact that polyhedral boundaries are not smooth manifolds further complicates the analysis but also makes it more interesting.

A. BACKGROUND. We begin with some geometric terminology. Given $X \subseteq \mathbb{R}^d$, the closure (respectively, interior) of X is denoted cl X (respectively, int X). The frontier of X is defined as cl $X \cap$ cl $(\mathbb{R}^d \backslash X)$, or as cl $X \cap$ cl $(A \backslash X)$ if the relative topology of some $A \supseteq X$ is understood. The unit d-sphere and the open unit d-ball are denoted S^d and D^d, respectively. A disjoint union of k-faces (subsets of $\mathbb{R}^d$ homeomorphic to D^k), for $k = 0, \cdots, d$, is called a d-dimensional *cell complex* if, given any two faces f and g, the intersection of cl f and cl g is either a union of faces or the empty set. A cell complex is called *simplicial* (or triangulation) if each face is the interior of a simplex (in the relative topology of its affine closure).

We take a rather general view of a polyhedron as any subset of 3-space that is locally a cone with a compact base [26]. Given a point $p \in \mathbb{R}^3$ and a subset $C \subset \mathbb{R}^3$, we say that the points $\alpha p + (1-\alpha)q$, for all $q \in C$ and $0 \leqq \alpha \leqq 1$, form a *cone* pC if for each point of pC distinct from p, the choice of q is unique. A subset P of $\mathbb{R}^3$ is called a *polyhedron* if each point $p \in P$ has a cone neighborhood pC, whose *base* C is compact. We use the term *boundary* to refer to the frontier of a polyhedron P and denote it ∂P. It is not hard to see that a polyhedron is a locally finite union of simplices and, hence, is piecewise linear. It need not be a manifold, however. An example of a valid polyhedron is shown in Fig. 2.1. An open halfplane π is a polyhedron but, because of the compactness condition, ceases to be one as soon as we include one point on its frontier. Adding the entire frontier is fine, however.

We need this level of generality because we will sometimes be dealing with rather convoluted shapes. As it turns out, however, most of our time will be spent with convex

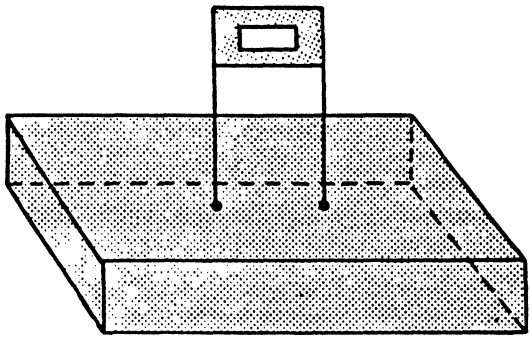

FIG. 2.1. *Example of a valid polyhedron.*

polyhedra, for which a more global (but slightly restrictive) definition is preferable [11]. A *convex polyhedron* is a nonempty intersection of a finite number of closed halfspaces. It is called a *convex polytope* if it is bounded. For technical convenience, we will restrict our discussion to convex polytopes, but it is easy to generalize it to the unbounded case. We assume that the boundary of a convex polytope P is facially structured as a two-dimensional cell complex with a minimal set of vertices. This last requirement means that a vertex (0-face) must be the intersection of three or more bounding planes (i.e., planes that delimit the defining halfspaces), but an edge (1-face) need not lie in the intersection of two distinct bounding planes. This assumption allows us to triangulate ∂P and still have a convex polytope; on the other hand, it forbids the facets (2-faces) incident upon any given vertex from being all coplanar. For storing two- and three-dimensional cell complexes we shall assume the representations of Baumgart [3], Muller and Preparata [22], or Guibas and Stolfi [15] and of Dobkin and Laszlo [8], respectively, or any other data structures that allow us to navigate at ease between adjacent cells. Such representations will be called *standard.*

To conclude this laundry list of assumptions and definitions, we introduce a well-known duality between points and planes, namely, the polarity δ, which maps any point $p = (\alpha, \beta, \gamma)$ distinct from the origin O to the plane of equation $\alpha x + \beta y + \gamma z = 1$: $\delta(p)$ is the plane normal to Op that lies at a distance $1/|Op|$ from the origin on the same side as p. Given a convex polytope P, whose interior contains the origin, the dual of P is the set of planes whose dual points lie in P. Forming the union of all these planes and taking the closure of the complement defines a convex polytope, which is called the *dual polytope* of P and is denoted P^δ. If the polytope P has no coplanar facets (e.g., no triangulation has been forced upon its boundary), then each vertex, edge, and facet of P corresponds respectively to a unique facet, edge, and vertex of its dual polytope, and the correspondence is involutory. Given a standard representation of a convex polytope P, it is elementary to compute a standard representation of P^δ in linear time, supplemented with pointers between each k-face of P and its dual $(2-k)$-face. Note that if the origin is not interior to P, then instead of a single polytope we obtain one or two unbounded polyhedra in 3-space.

Central to the Muller-Preparata method [22] is the use of the fact that convex hulls and intersections play dual roles. Indeed, if the origin lies in the interior of two convex polytopes P and Q, then the convex hull of P^δ and Q^δ is the dual polytope of $P \cap Q$. In other words, identifying the binary operation *intersection* in primal space with the binary operation *convex hull* in dual space yields the following commutative

diagram:

$$\begin{array}{ccc} P, Q & \xrightarrow{\delta} & P^{\delta}, Q^{\delta} \\ \downarrow & & \downarrow \\ P \cap Q & \xrightarrow{\delta} & (P \cap Q)^{\delta} = \text{Hull}\,(P^{\delta} \cup Q^{\delta}) \end{array}$$

Unlike the Muller-Preparata approach, which commutes through the diagram only once, our algorithm will spend most of its time traveling back and forth between primal and dual spaces.

B. SHIELDING A CONVEX POLYTOPE. We open this discussion with a variant of the Dobkin-Kirkpatrick construction. Let P be a convex polytope of n vertices with nonempty interior. We assume that its boundary has been triangulated, which is easy to ensure in linear time. Recall that the *degree* of a vertex refers to its number of incident edges. We select a maximal independent set of constant-degree vertices: (i) Pick any vertex of degree at most 8, and mark it along with all its adjacent vertices; (ii) iterate on this process, always making sure to pick unmarked vertices. Termination occurs when we run out of unmarked vertices of degree at most 8. Because the number of edges is at most $3n-6$, we find that the sum of all vertex degrees does not exceed $6n-12$. Since every vertex has degree at least 3, the number m of vertices of degree at most 8 satisfies $9(n-m) \leqq 6n-12-3m$, and hence $m \geqq n/2$. Therefore, at least $n/18$ vertices will be selected in this process. As shown in Edelsbrunner [11], we can actually do better by considering vertices in order of nondecreasing degree. This allows us to find an independent set of at least $n/7$ vertices of degree at most 12. In both cases, the time for selecting the desired vertices is linear.

Around each selected vertex v, we perform some local surgery by removing v and its "umbrella" of incident faces and recomputing the convex hull of P (Fig. 2.2). Since v has degree at most 12, this shelling operation can be done in constant time. Note that because of the independence of the selected set of vertices, the order in which vertices are "popped out" does not matter. In $O(n)$ time we thus will have (i) removed all selected vertices and their incident faces, (ii) computed the new convex hull P_1, and (iii) triangulated its boundary. We easily verify that P_1 is a valid convex polytope; in particular, each of its vertices lies on at least three distinct bounding planes. We

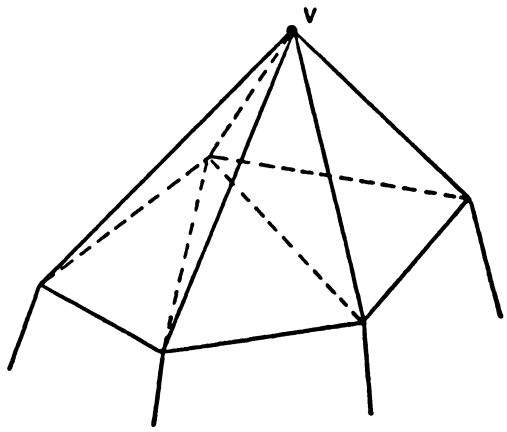

FIG. 2.2. *Removing v and its "umbrella" of incident faces.*

can now repeat this process with respect to P_1 and define a sequence of nested convex polytopes $P_0 \supset P_1 \supset \cdots \supset P_\alpha$, where (i) $P_0 = P$, (ii) P_α has constant size, and (iii) each cl $(P_i \backslash P_{i+1})$ is a collection of three-dimensional cones whose interiors are disjoint. If the interior of P_1 is empty, then at most two vertices were popped out and P has at most $12+2$ vertices. To avoid any difficulty, we do not bother with P_1 and set $\alpha = 0$ whenever P has at most 14 vertices. We use the same criterion to terminate the iteration.

Our next step is to compute a triangulation of P that is compatible with all the nested polytopes. This might be awkward to do during the shelling phase, because we may inherit the "wrong" triangulation from outer polytopes and create tetrahedra with empty interiors (Fig. 2.3). The difficulty is that a facet incident upon a popped-out vertex v of P_i might still contribute a portion of a facet of P_{i+1}. A simple fix is to retriangulate inside out. Assume that P_i has been given a triangulation compatible with $P_{i+1}, \cdots, P_\alpha$. Each cone of cl $(P_{i-1} \backslash P_i)$ can be triangulated directly by connecting its apex to the triangulation of its base provided by P_i. This will lead to a compatible triangulation of P in $O(n)$ time. Note that the resulting triangulation of ∂P might be different from the one we started with.

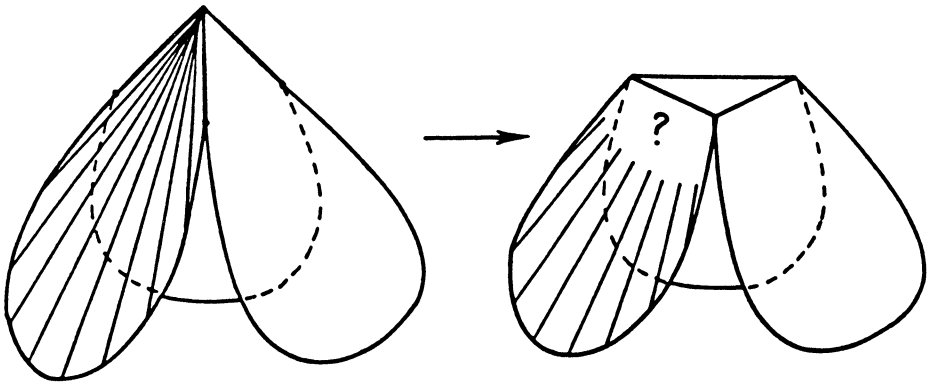

FIG. 2.3. *Inheriting the wrong triangulation.*

There is nothing startlingly novel here. The only slight twist from the Dobkin-Kirkpatrick construction is to transform the nested sequence into a three-dimensional triangulation. This will allow us to discover P from various angles by traversing it along straight lines, shooting rays through it, and in general exploring its geometry from within. Unfortunately, we also need to approximate P from the outside. What is unfortunate about this is that the complement of P is not convex, so we cannot play the same game over. Its dual polytope is convex, however, so it might just be the right time to jump into dual space.

After ensuring that no two facets are coplanar, by removing edges if necessary, we choose an origin O in the interior of P and we form its dual polytope P^δ. Then we triangulate ∂P^δ and submit P^δ to the in-growing process described above. This results in a sequence of nested convex polytopes $P^\delta = \Pi_0 \supset \Pi_1 \supset \cdots \supset \Pi_\beta$, where Π_β has constant size. The triangulation of $P_0 \backslash \text{int } P_\alpha$ is the *primal shield* of P; its counterpart between Π_0 and Π_β is called the *dual shield.* Unless specified otherwise, the term "shield" refers to both its primal and dual parts. Given any integer k such that $0 \leqq k \leqq \min\{\alpha, \beta\}$, the triangulations of $P_0 \backslash \text{int } P_k$ and $\Pi_0 \backslash \text{int } \Pi_k$ form the *k-shield* of P. Primal and dual k-shields are defined in the obvious way. All logarithms below are to the base 2.

LEMMA 2.1. *Let P be a convex polytope with nonempty interior, and let m be its total number of vertices and bounding planes. The shield of P can be constructed in $O(m)$ time. The total number of vertices and bounding planes in each P_k (and Π_k) is at most $3(1-1/7)^k m$. The total number of nested polytopes is less than $9 \log m + O(1)$.*

Proof. Let v_k and f_k be the number of vertices and bounding planes in P_k, respectively. The reason for dealing with $m_k = v_k + f_k$ is that this quantity is invariant under duality. Since the boundary of a convex polytope has Euler characteristic 2 and every facet has at least three incident edges, we derive $f_k \leqq 2v_k - 4$. Each pass in the algorithm removes at least one-seventh of the vertices; therefore, (i) the preprocessing is linear and (ii) $m_k \leqq 3(1-1/7)^k v_0 - 4$. □

C. NAVIGATING THROUGH A SHIELD. The usefulness of a shield owes to its being both an approximation scheme and a cell complex. Indeed, it gives us a "two-way" sequence of approximations for P, through which we can easily navigate and "discover" the boundary of P from any desired angle. This assumes that we use a proper representation, such as the Dobkin-Laszlo structure [8]. Without getting into the details of the data structure, let us just say that from each tetrahedron of a shield we can gain access to any of its four incident facets in constant time. Conversely, any facet leads us directly to its two incident tetrahedra. Also, the tetrahedra and facets incident upon an edge are accessible in cyclical order around the edge.

Let us consider a simple operation, such as being given a ray $\vec{r}$ with a starting point in a tetrahedron of, say, the primal shield of P, and being asked to traverse the primal shield along $\vec{r}$. In general, the ray will cut through a sequence of cells alternating between tetrahedra and triangles. When this is the case, there is no difficulty in discovering the sequences of cuts on the fly, at a cost of $O(1)$ per cut. Let us call a facet of the primal shield *primitive* if it lies on the boundary of one of the nested polytopes. Since the popped-out cones are bounded by primitive triangles, the ray $\vec{r}$ cannot cut more than a constant number of nonprimitive triangles in a row. Consequently, the total size of the cutting sequence is proportional to the number of primitive triangles intersected by the ray. A minor difficulty arises when the ray cuts through an edge or a vertex of the shield. In the case of an edge we are faced with two candidate triangles to be visited next. We can break ties by making arbitrary navigational conventions. For example, we might agree always to choose the triangle that is (locally) highest (or leftmost if there are several highest ones). We can also submit the ray $\vec{r}$ to a *symbolic* perturbation—see Edelsbrunner and Mücke [13] and Yap [28]. Note that we can easily generalize the mode of traversal to polygonal lines embedded in 3-space. The following summarizes our discussion.

LEMMA 2.2. *The complexity of traversing the primal (respectively, dual) shield along a polygonal line in three dimensions, knowing the starting cell, is proportional to the complexity of the polygonal line plus the number of nested polytopes P_i (respectively, Π_i) whose boundaries are cut during the traversal (counting multiplicities).*

3. Intersecting two convex polytopes. We begin with a brief discussion of what makes the problem not so easy. We can assume that we have a point O inside both convex polytopes P and Q, since such a witness (if any) can be discovered in linear time [7], [10], [19]. What remains to be done is in some sense merging P and Q. Imagine a sphere centered at O, on which ∂P and ∂Q are centrally projected. This gives us two spherical subdivisions S_P and S_Q. Merging the two subdivisions would do the job, but this might cause a quadratic blowup. The next "smartest" move might appear to be locating each vertex of S_P in S_Q and vice versa, which we can do in $O(n \log n)$ time, where n is the total number of vertices in P and Q [18]. Unfortunately,

it is easy to prove that this complexity is optimal. Clearly, we are still seeking too much information, and the subdivisions S_P and S_Q appear essentially worthless. So, let us bring shields into the picture. How about locating each vertex of P (respectively, Q) in the primal shield of Q (respectively, P)? Such information might be a good start from which to launch our intersecting attack. But actually this is still asking too much: Indeed, it can be proven that locating the vertices of P in the primal shield of Q requires $\Omega(n \log\log n)$ time in the worst case. All these attempts fail because we are working too far from the boundaries of P and Q and thus are giving free rein to our adversary. Pairwise intersections of convex polytopes possess a rich geometric structure into which we have yet to tap seriously. The time has come for a closer look at the geometry of the intersection problem.

A. BROADCASTING. We devote this section to defining the notion of broadcasting and showing that it is the essential operation in computing the intersection of two polytopes (Lemma 3.1). Let us restate our assumptions: P and Q are two convex polytopes with a total of n vertices, and their interiors contain the origin O. To simplify our discussion, we shall assume that P and Q have no two coplanar facets and are in general position relative to each other: No facet (respectively, edge) of one is coplanar (respectively, colinear) with a facet (respectively, edge) of the other, no vertex of one lies on the boundary of the other, etc. To borrow a cliché, relaxing these assumptions is tedious but not difficult.

Assume that the boundaries of P and Q have been triangulated by inheritance from their primal shields. Since the two polytopes are convex and contain the origin in their interiors, the boundary $\Sigma = \partial(P \cup Q)$ is the graph of $\max\{f, g\}$, where f and g are continuous functions $S^2 \mapsto \mathbb{R}^+$. It follows that Σ is homeomorphic to S^2. Let us now color ∂P blue and ∂Q red. (We apologize to the reader for not using a more evocative terminology: If it is any help for future reference, P and the first letter of "blue" sound somewhat alike.) Points that are both blue and red are said to be purple. The facets of Σ become monochromatic polygons. Beware: Some of them may be of nonconstant size, nonconvex, and even perforated. However, Σ can still be regarded as a two-dimensional cell complex. Of particular interest to us are the connected components of $\partial P \cap \partial Q$. These are disjoint, purple, simple cycles in the facial graph of Σ, which we call *laces*. removing all the laces from Σ creates relatively open polyhedral surfaces, called *regions* (white and dotted areas in Fig. 3.1).

Regions and laces partition Σ into maximal monochromatic connected subsets. Assume for the time being that Σ has at least one lace. Then the closure of a region

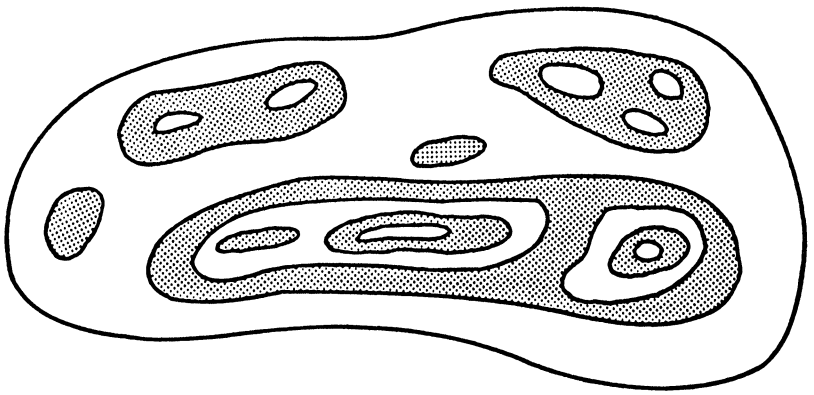

FIG. 3.1. *Creation of regions by removing all the laces from* Σ.

R is a (topological) manifold with boundary, which is homeomorphic to S^2 perforated by $k \geqq 1$ disjoint copies of D^2. Unlike some of the sets we will encounter later, this is a rather friendly one: It is an orientable bounded surface, its number of boundary components is k, and its Euler characteristic is $2-k$. The boundary of the manifold is also the frontier of R in the relative topology of Σ: By extension, we call it the *boundary* of R. The k connected components of the boundary are called the *bounding* laces of the region. Note that each lace of Σ bounds exactly two regions (of opposite color). A region R, being a monochromatic component of the graph $\max\{f, g\}$, is paired with a homeomorphic component R^c of the graph $\min\{f, g\}$: This component has the opposite color of the region R and is called its *co-region* (dotted area in Fig. 3.2—boundaries are left untriangulated for clarity). Here are more formal definitions of all these concepts.

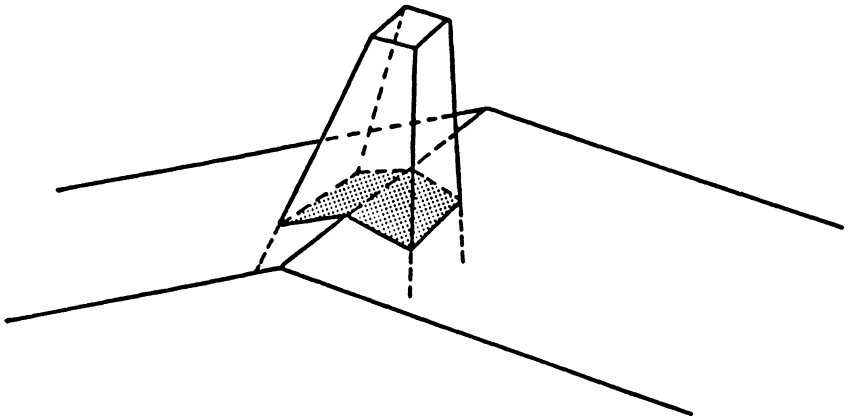

FIG. 3.2. *Co-region of R (dotted area).*

DEFINITIONS. A *lace* is a connected component of $\partial P \cap \partial Q$. A *region* is a connected component of $\partial(P \cup Q) \backslash (\partial P \cap \partial Q)$. A *co-region* is a connected component of $\partial(P \cap Q) \backslash (\partial P \cap \partial Q)$.

We are now ready to define the notion of broadcasting. A *broadcaster* is an algorithm that takes as input a vertex v on a lace and a color c and returns at least one vertex on each of the laces bounding the unique c-colored region incident upon v. Each vertex of a lace is determined by the intersection of a facet of P (or Q) and an edge of Q (or P): It is understood that this correspondence should be provided in full by the broadcaster. Suppose without loss of generality that the broadcaster is given the color red as input. Then one solution for the broadcaster is to traverse the relevant co-region on the boundary of P and keep track of the current location in the primal shield of Q until all the desired laces have been reached. In that case we say that the broadcasting is *anchored* to P. As we shall see, there is another, slightly more complicated solution, which is to stay anchored to Q and traverse the relevant portion of its boundary while tracing the navigation in the dual shield of P. In all cases the term *anchor* refers to the polytope on whose boundary the traversal takes place, the other polytope providing the guiding shield.

LEMMA 3.1. *If a broadcaster is available, it is sufficient to know a vertex of one lace of* Σ *(or to know that there is no lace) in order to compute the entire intersection of* P *and* Q *while incurring only linear overhead on top of the broadcasting time.*

Proof. If Σ has no lace, just knowing this fact will be enough for us to compute $P \cap Q$ in linear time, by checking whether a vertex v of P lies inside or outside Q.

Indeed, $P \cap Q = P$ (respectively, $P \cap Q = Q$) if and only if $v \in Q$ (respectively, $v \notin Q$). Suppose now that Σ has at least one lace. They key observation is that if the regions of Σ are to be made into the nodes of a graph, with arcs connecting nodes whose associated regions are bounded by a common lace, then the graph in question will be connected. Therefore, starting from the vertex given to us by the input, a standard graph traversal algorithm will allow us to discover a vertex for each lace of Σ. Since the facets of P and Q are triangles, it is elementary to compute an entire lace in time linear in its size by tracing it from one of its vertices. Once we know all the laces in full, we can easily compute $P \cap Q$ in linear time by marking the laces in both ∂P and ∂Q and exploring the facets of the co-regions. □

B. AN INTERSECTION ALGORITHM BASED ON CO-ANCHORED NAVIGATION. To illustrate the utility of broadcasting further and bring out some of its main features, we describe an $O(n \log n)$ time algorithm for intersecting P and Q that relies on a symmetric form of broadcasting, one where the anchoring alternates between P and Q (hence the term *co-anchored*). The idea is to stay glued to the boundary of $P \cap Q$ and confine our traversals to co-regions. In one broadcast we are anchored to P; that is, we navigate across the primal shield of Q while exploring a co-region in ∂P. The next time around, roles are reversed and we find ourselves broadcasting across the primal shield of P while being anchored to Q. Dual shields are never used. For this reason co-anchored navigation relies solely on what we call *primal broadcasting.*

Co-anchored navigation is simple, but it leaves us little room for improvement. Later we will develop a more complex but more promising form of navigation in which we always remain anchored to the same polytope. This requires keeping track of where we are within the primal shield of Q while we lie inside Q (primal broadcasting) and where we are in the dual shield of Q when we navigate outside Q (dual broadcasting). This also necessitates the use of mutations, the transition operation we mentioned earlier. Briefly, the reason why this more complicated form of navigation is preferable is that by always remaining anchored to the same polytope, we remain free to choose the anchor, whereas, before, the choice of anchors was dictated by the color of the co-regions. This added freedom is the key to being able to balance costs between the two polytopes and achieve linear time.

Let us now describe the simpler, co-anchored form of navigation that, as we said, relies exclusively on primal broadcasting. Given a vertex v of a certain lace γ of Σ, let R be the red region of Σ incident upon v and let R^c be its co-region. Suppose now that the broadcaster is given the vertex v and the color red as input. Its goal is to find vertices on all the laces $\gamma, \gamma_1, \cdots, \gamma_l$ of R. First of all, we (the broadcaster) can easily compute the entire lace γ by tracing the connected component of $\partial P \cap \partial Q$ passing through v. Since the boundaries are triangulated, the computation is linear in the size of γ.

Now we must reach out to all the other laces γ_i. Our strategy relies on the fact that the closure of a co-region is an *edge-connected* bounded surface (meaning that the vertices and edges form a connected graph). This seemingly obvious fact should not be taken for granted, because it does not necessarily hold for the closure of a region. In Fig. 3.3, for example, the dotted area represents a region facet that is biconnected and therefore is not edge-connected. So we must prove our claim. Suppose that two vertices of Σ in the closure of a co-region R^c cannot be joined by a path of edges in cl R^c. Then, either one of these vertices can be separated from the other by a simple closed curve that lies entirely in cl R^c but does not cut across any edge or vertex. It easily follows that this curve must lie in a single facet f of cl R^c and that f is perforated. A local analysis of the perforation reveals that f contains two points p

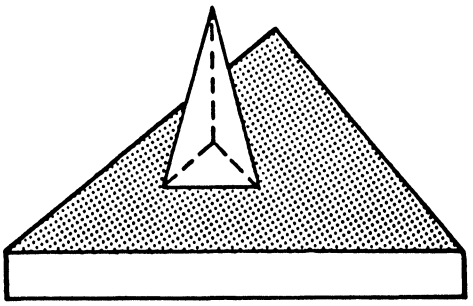

FIG. 3.3. *Region facet (dotted area) that is biconnected and therefore is not edge-connected.*

and q such that $pq \not\subseteq P \cap Q$. This contradicts the convexity of $P \cap Q$ and proves our claim.

Observe that cl R^c has vertices in $\partial P \cap \partial Q$ as well as possibly in $\partial P \backslash \partial Q$ (we previously assumed that R is red). The main difference is that the latter vertices are known ahead of time, whereas the former (the lace vertices) are discovered during the broadcast. Using standard graph traversal techniques, we can reach the vertices of all the γ_i's from v. The only problem is that whenever we visit an edge of P that crosses the boundary of Q, we need to know about it. The solution is to keep track—at all times during the broadcast—of where we are in the primal shield of Q. This means computing the intersections of the visited edges of P with the facets of the primal shield of Q. It follows trivially from Lemmas 2.1 and 2.2 that the broadcast will take $O(\rho \log n)$ time, where ρ denotes the number of edges in cl R^c. Obviously, we shall exchange the roles of P and Q if the color blue is given as input to the broadcaster. To summarize, the cost of all the broadcasts will be proportional, up to a logarithmic factor, to the size of all the co-regions. This gives us a total broadcasting time of $O(n \log n)$.

We will now be in a position to apply Lemma 3.1 and compute the entire intersection of P and Q as soon as we know one lace vertex. To do this, we take an arbitrary *starting vertex* v of P and check whether it lies inside Q. If the answer is yes, we pursue the search and locate v in the primal shield of Q. From there, we start a regular broadcast-like routine, which involves traversing ∂P and keeping track of where we are in the primal shield of Q at all times. Either we will reach the boundary of Q and, hence, a vertex of a lace, or we won't. In the latter case, we know that $P \cap Q = P$. If v lies outside Q, on the other hand, we locate the point $w = Ov \cap \partial Q$ in the primal shield of P. Let z be one of the vertices incident upon the unique (simplicial) face of Q that contains w. Let us traverse the primal shield of P along the oriented segment wz. If we do not exit P (or if $w = z$) we are just back to the previous case, with the roles of P and Q reversed. If we do leave P, however, the exit point belongs to a lace and therefore is a valid starting vertex (or it lies on an edge incident upon one). In view of Lemmas 2.1, 2.2, and 3.1, we now have an intersection algorithm with $O(n \log n)$ running time.

C. A TOPOLOGICAL EXCURSION. Before we can switch to single-anchored navigation and describe dual broadcasting, we must further explicate the relationship between polytopes and their duals. This section introduces the notions of *belts*, *bracelets*, and *co-dual regions*, on which dual broadcasting is founded. This introduction might be a little tedious, but it is indispensable for a proper understanding of why the intersection algorithm actually works.

Everything we said of P and Q (e.g., laces, regions, co-regions) applies just the same to P^δ and Q^δ. In the following, Σ^δ will designate the analog of Σ vis-à-vis $P^\delta \cup Q^\delta$; that is, $\Sigma^\delta = \partial(P^\delta \cup Q^\delta)$. We assume that the boundaries of all four polytopes P, P^δ, Q, Q^δ have been triangulated in accordance with their shields (and, hence, may have coplanar facets). Correspondence between a polytope X and its dual is ensured by the usual pointers between a k-face and its dual $(2-k)$-face. This concerns the state of faces prior to boundary triangulation. With the introduction of simplicial faces, however, this representation must be slightly amended. Let us distinguish between the *old* faces of X (before triangulation of ∂X) and the *new* ones. Note that some of them, vertices in particular, are both old and new. Each vertex of X points to any one of the new facets to which it is dual, and each new facet points to its unique dual vertex. Each old edge of X points to the unique old edge of X^δ that is dual to it. Each new edge e points to the four old edges of X that are both adjacent to e and incident upon the old facet where e lies. It is a simple exercise to set up all these pointers in linear time. An attractive aspect of this representation is that, given a new facet abc of X, we can gain constant time access not only to its dual vertex v, but also to three new facets of X^δ that are dual to a, b, and c, respectively, and incident upon v (Fig. 3.4).

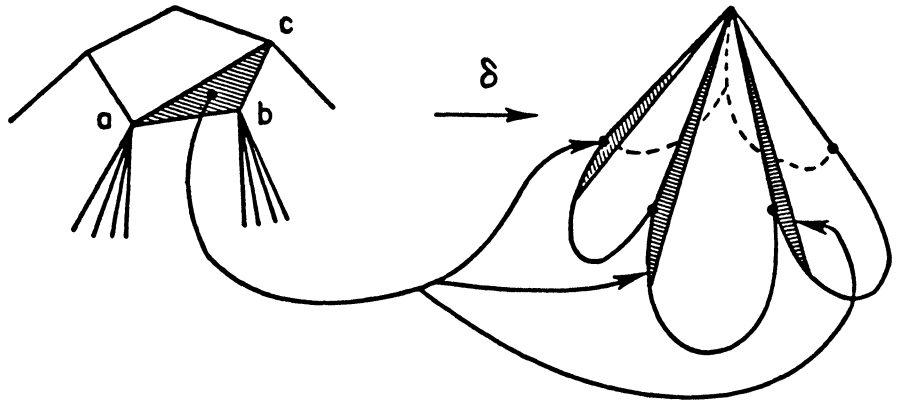

FIG 3.4. *Relationship between polytopes and their duals.*

Belts. We will show that the laces of Σ can be individually "covered" by disjoint *belts* that dualize to laces of Σ^δ. Let us color yellow all the faces of the convex hull H of $P \cup Q$ that are faces of neither P nor Q. A maximal connected subset of yellow faces is a polyhedron (in our definition), which we call a *belt* of Σ. Formally, we define *belts* and *co-belts* as follows:

DEFINITIONS. Let H be the convex hull of $P \cup Q$. A *belt* is a connected component of $(\partial H) \backslash \partial(P \cup Q)$. A *co-belt* is a connected component of $\partial(P \cup Q) \backslash \partial H$.

Note that belts and co-belts are relatively open. To be able to say more about them, we define the *envelope* of a polyhedron X (in a slightly nonstandard manner) as the set of planes π such that (i) the affine closure of $X \cap \pi$ has dimension at least 1 and (ii) X lies entirely in either one of the closed halfspaces defined by π. It easily follows from the incidence-preserving properties of a polarity that the envelope of a belt B dualizes to a lace B^δ of Σ^δ. More specifically, as we walk along the lace B^δ, a certain plane $\pi(x)$ *rolls* around the belt in a continuous fashion as x goes around S^1. Therefore, B is a simple cycle of simplicial facets and edges (and no vertices), $t_0, e_0, \cdots, t_m, e_m$, where each facet t_i is incident upon the two edges e_i and e_{i-1}

(mod $m+1$). It follows that a belt is topologically equivalent to an open annulus $S^1 \times (0, 1)$. Its frontier consists of two monochromatic connected components: One of them, denoted b_c, is blue, and the other, b_m, is red. Let us look at these components more closely. We may restrict ourselves to one of them, say b_c.

In general, b_c will be a simple closed polygonal curve, but unfortunately this might not always be the case.[1] Indeed, consider an old (i.e., untriangulated) facet f of P^δ that contributes at least one edge to the lace B^δ. It is certainly possible for B^δ to come in and out of f repeatedly, as is shown in Fig. 3.5 (where f is the horizontal face of the lower polyhedron). To translate this into the language of belts, let $u_0, \cdots, u_k$ be the cycle of edges of b_c encountered while visiting $t_0, \cdots, t_m$ in that order. Identifications of the form $u_i = u_j$ might occur. The topological type of a belt enforces two important restrictions on the allowable identification patterns. First, three or more edges cannot be identified together. Second, in any subcycle u_i, u_j, u_k, u_l we cannot have both $u_i = u_k$ and $u_j = u_l$, which means that the identifications form a parenthesis system. Of course, we might have vertex identifications only and no edge identifications at all. It might even be the case that b_c consists of a single vertex, which will happen if B^δ lies entirely in a single facet of P^δ—see the top vertex in Fig. 3.6. The only reason that b_c is not always topologically equivalent to S^1 is that the boundaries of P and Q are not smooth. If they were, then b_c would actually be diffeomorphic to S^1. Thus, if we look at ∂P and ∂Q as limit sets of smooth 2-manifolds, it appears immediately that b_c is a simple closed curve that may have been pinched and collapsed along vertices and edges according to a parenthesis system. A traversal of such a curve is

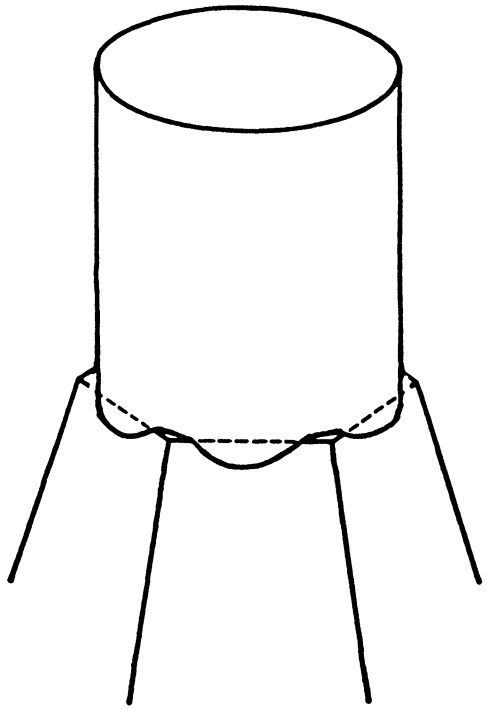

FIG. 3.5. *B^δ may come in and out of f repeatedly.*

[1] A similar situation occurs in the merge step of Preparata and Hong's convex hull algorithm [24], which is also discussed in detail in Edelsbrunner [11].

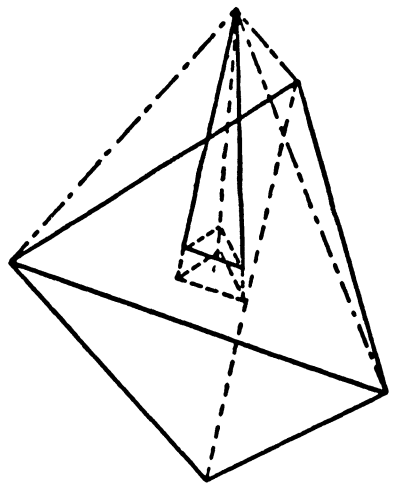

FIG. 3.6. *A case in which b_c consists of a single vertex.*

shown in Fig. 3.7: The curve can be obtained from a circle by pinching it and gluing it at various places.

Observe that two distinct laces cannot share vertices or edges, but they can pass through the same (old) facet of P or Q. This implies that although two belts cannot share a common edge or facet, their frontiers might have vertices and edges in common (recall that a belt does not contain its frontier). This fact will require that special measures be added to the intersection algorithm.

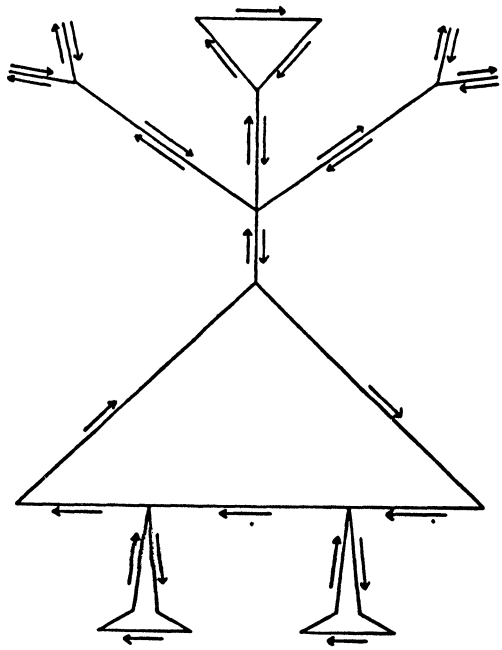

FIG. 3.7. *Traversal of a simple closed curve that has been pinched and collapsed along vertices and edges according to a parenthesis system.*

Bracelets. The space between the boundary of the convex hull H and the polytopes P and Q consists of disjoint donut-like objects, which are called *bracelets.*

DEFINITION. Let H be the convex hull of $P \cup Q$. A *bracelet* is the closure of a connected component of $H \backslash (P \cup Q)$.

A different perspective on belts and bracelets comes from looking at Σ and ∂H as the graphs of two continuous functions, respectively, φ and $\psi: S^2 \mapsto \mathbb{R}^+$. Removing the kernel of $\psi - \varphi$ from S^2 leaves relatively open connected components $S_1, S_2, \cdots$, and each belt B is the graph of ψ's restriction to some distinct domain S_i. By analogy, the graph of φ's restriction to S_i is the co-belt B^c of B. Belts and co-belts are homeomorphic and have the same frontiers. Therefore, the closure of $B \cup B^c$ is the frontier of a compact polyhedron $\mathcal{B}$, which is a bracelet of Σ: Its interior is homeomorphic to an open filled torus $D^2 \times S^1$. A bracelet is the closure of a connected component of $H \backslash (P \cup Q)$, and its belt is the relative interior of the intersection of that component with ∂H. One should not hastily conclude that a bracelet is always topologically equivalent to a filled torus. It can actually assume rather contrived shapes, because as the frontiers b_c and b_m might cause (homeomorphically) multiple point identifications along nonnull homologous paths on the torus. This might give us a filled torus pinched at various places or, as in Fig. 3.6, a closed 3-ball pinched at a pair of antipodal points. Note that general position alone cannot prevent this type of pathology.

Again, let b_c and b_m be the frontier components of B. Since Σ is locally blue (respectively, red) around b_c (respectively, b_m), we can define the maximal blue (respectively, red) connected subset B_c (respectively, B_m) of B^c whose closure contains b_c (respectively, b_m). We claim that B_c and B_m are joined together along a single lace. To prove this claim, let $\gamma = B^c \backslash (B_c \cup B_m)$ and suppose, by contradiction, that γ contains a point p of a color different from purple, say, red. Let C be the maximal connected red subset of B^c that contains p (the dotted region containing p in Fig. 3.8). The closure of C does not intersect b_c or b_m; therefore, C is a full region of Σ (and not just the portion of a region that lies within B^c). Figure 3.8 shows Σ with the white area denoting B^c and with C in the middle: It is bordered by red (i.e., dotted) material on one side and blue (i.e., hatched) material on the other. Like every region, C contains a point in ∂H. To see why, consider a plane π supporting a facet of its co-region, and let π^+ be the open halfspace bounded by π that does not contain the origin. Since P lies entirely outside π^+, we have $\pi^+ \cap \partial Q \subseteq C$; therefore, the point of $\pi^+ \cap \partial Q$ farthest away from π is a vertex of Q in $C \cap \partial H$. But this contradicts our previous observation that the portion of the bracelet $\mathcal{B}$ that lies in ∂H is confined to the closure of its belt. Consequently, γ is a purple curve whose removal from B^c creates two monochromatic surfaces of opposite color, each homeomorphic to an open annulus. It follows that γ is homeomorphic to S^1 and therefore is a lace of Σ. Thus, we have shown that $\partial \mathcal{B}$ is the disjoint union of B, b_c, B_c, γ, B_m, and b_m. In general, the removal of any one of these six sets makes $\partial \mathcal{B}$ homeomorphic to an annulus (open for the lower-case sets, closed for the upper-case ones). But one should not count on it. Removing the belt or the co-belt makes $\partial \mathcal{B}$ equivalent to a closed 2-disk with usually one open perforation, but with possibly zero (Fig. 3.6) or an arbitrarily large number of them.

Dual bracelets and co-dual regions. Each belt of Σ is associated with a unique lace of Σ. Since this is true in dual space as well, this association is bijective. Therefore, the envelope of a belt B of any bracelet $\mathcal{B}$ of Σ dualizes to the lace γ^* of a bracelet $\mathcal{B}^\delta$ of Σ^δ, whose belt B^* has for an envelope the dual of the lace γ of $\mathcal{B}$. If λ and β map a bracelet to its lace and belt, respectively, we can extend the commutative diagram

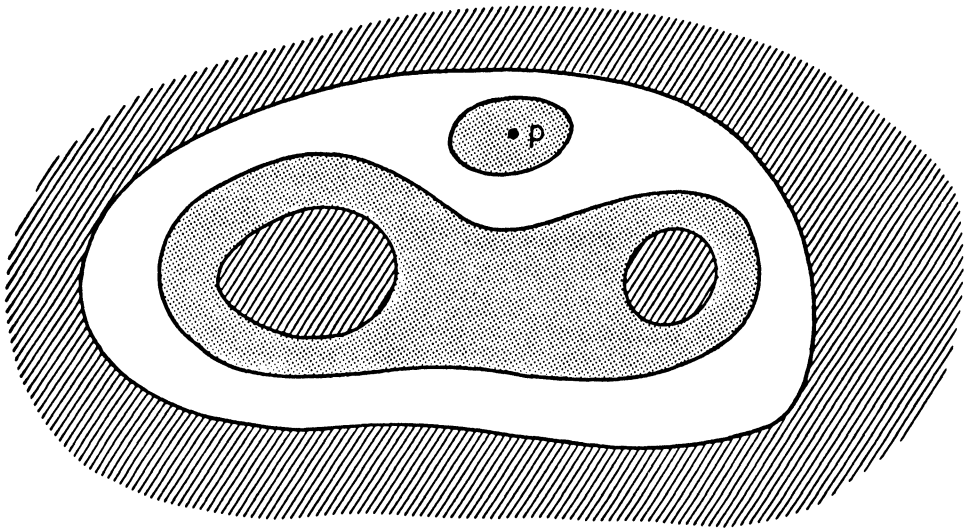

FIG. 3.8. *The maximal connected red subset of B^c that contains p (dotted region containing p) is a full region.*

of § 2.A as follows:

$$\begin{array}{ccc} \gamma & \xrightarrow{\delta} & B^* \\ \downarrow{\scriptstyle \lambda^{-1}} & & \downarrow{\scriptstyle \beta^{-1}} \\ \mathscr{B} & \xrightarrow{\delta} & \mathscr{B}^\delta \\ \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle \lambda} \\ B & \xrightarrow{\delta} & \gamma^* \end{array}$$

By carefully rolling a plane around the boundary of $\mathscr{B}$ in the appropriate manner, we trace the entire boundary of $\mathscr{B}^\delta$ in dual space. (The rolling has to be "appropriate" in the sense that the passage from a belt to the co-belt and the passage across the lace must cut through the bracelet instead of tracing its envelope.) Obviously, the association between $\mathscr{B}$ and $\mathscr{B}^\delta$ is involutory. For all these good reasons, we call $\mathscr{B}^\delta$ the *dual bracelet* of $\mathscr{B}$. Note that a lace alone does not provide sufficient information to reconstruct its associated belt in dual space. One needs to add the planes supporting the facets incident to it. Another subtle point is that although the envelope of a belt in primal space is dual to a lace in dual space, the facial structures of these objects may not be in bijection. The reason is that all facets have been triangulated. As a result, a belt in primal (respectively, dual) space might end up being facially less "refined" than its corresponding lace in dual (respectively, primal) space. This will complicate our algorithm a little because of the ensuing loss of information incurred when switching to dual space.

We are now in a position to establish the most useful connection between the input polytopes and their duals, namely, a bijection between the regions of Σ and those of Σ^δ. This can be seen as a further extension of the commutative diagram. As usual, we must use caution, however, because of the nonsmoothness of polyhedral boundaries. Every bounding lace of a region is covered by a band (a co-belt) whose frontier usually consists of two simple closed curves: Figure 3.9 shows four laces (the thick black rings) surrounded by their co-belts. As we saw earlier, things might not be

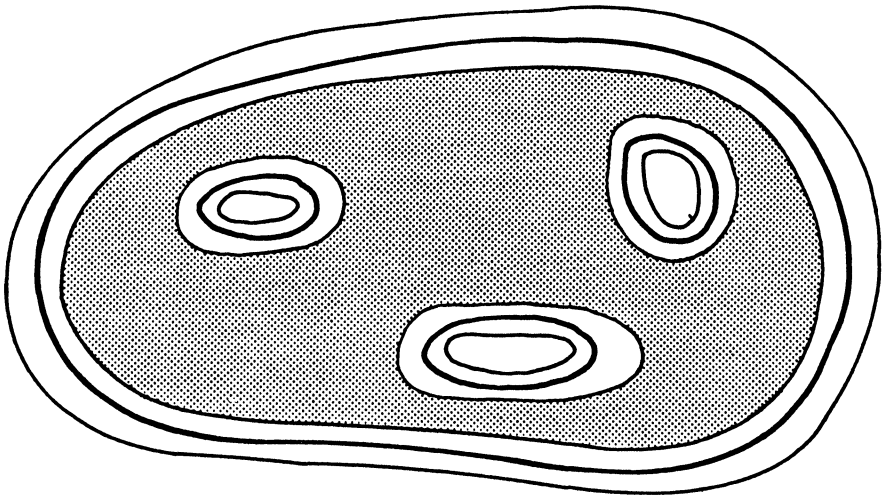

FIG. 3.9. *Four laces (thick black rings) surrounded by their co-belts.*

nearly as simple. Let R be a blue region of Σ and let $\mathcal{B}_1, \cdots, \mathcal{B}_l$ be the bracelets of its bounding laces. Since the corresponding co-belts $B_1^c, \cdots, B_l^c$ are pairwise disjoint, the set $K = R \backslash \cup_{1 \leqq i \leqq l} B_i^c$ is a connected subset of ∂H—the dotted area in Fig. 3.9. If again we think of P and Q as limit sets of infinite sequences of isotopic smooth manifolds, we can interpret K as a deformation retract of a copy of cl D^2 with zero, one, or several open perforations (Fig. 3.10). Thus, any two planes supporting H at points of K can be brought together by continuous rolling around H without ever leaving contact with K. This proves that among the laces of the dual bracelets $\mathcal{B}_1^\delta, \cdots, \mathcal{B}_l^\delta$, any two can be connected by a blue path in a co-region of Σ^δ. An immediate consequence is that the laces of $\mathcal{B}_1^\delta, \cdots, \mathcal{B}_l^\delta$ bound a common red region of Σ^δ. Since the argument is involutory, the region in question must be entirely bounded by these laces. For this reason it is only natural to call it the *co-dual region* R^δ of R (the prefix "co" is a reminder that it is really its co-region that is dual to R). Again, this map is involutory and, of course, consistent with the bijection previously defined between laces in primal and dual spaces.

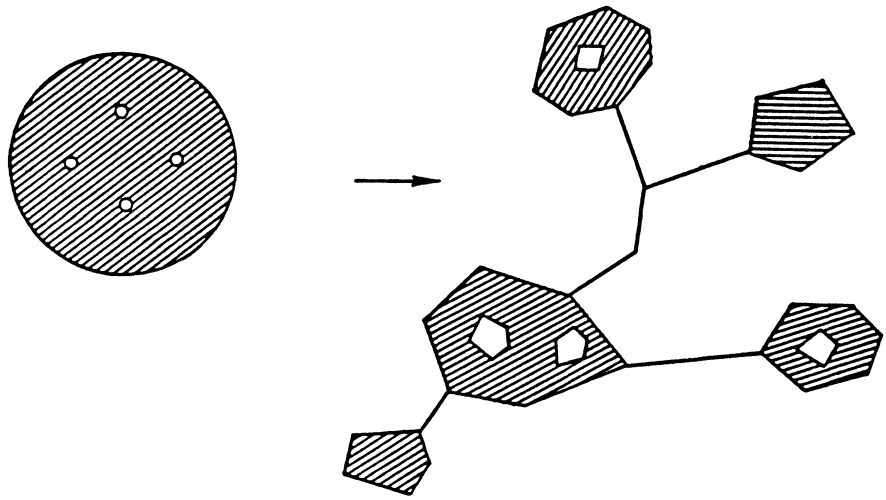

FIG. 3.10. *Interpretation of K as a deformation retract of a copy of* cl D^2 *with four open perforations.*

Unlike k-faces, which become $(2-k)$-faces in dual space, or, for that matter, laces and belts, the type "bracelet" (respectively, "region") is invariant under duality (respectively, co-duality). The best illustration of this comes from the self-dual case, where $Q = P^\delta$ (Fig. 3.6). In that remarkable situation, dual bracelets and co-dual regions remain invariant; Only their colors change! The following lemma summarizes most of our discussion so far.

LEMMA 3.2. *The interior of a bracelet $\mathcal{B}$ of Σ is homeomorphic to an open filled torus. Its boundary contains exactly one belt and one lace of Σ, which are homeomorphic to $S^1 \times (0, 1)$ and S^1, respectively. With $\mathcal{B}$ is associated a unique dual bracelet $\mathcal{B}^\delta$ of Σ^δ: The dual of the envelope of the belt of $\mathcal{B}$ is the lace of $\mathcal{B}^\delta$; conversely, the dual of the lace of $\mathcal{B}$ is the envelope of the belt of $\mathcal{B}^\delta$. Also, to each region R corresponds a unique co-dual region R^δ of Σ^δ of opposite color, and the bracelets of their bounding laces are dual to each other.*

D. DUAL BROADCASTING. As we said earlier, our goal is to break the symmetry between P and Q provided by co-anchored navigation. This will oblige the broadcaster to alternate between two modes of operation: primal and dual. The motivation for this move is to leave us the choice of anchor. Suppose once and for all that P is the anchor. (How to choose the right anchor will be discussed later.) Using the previous notation, what shall the broadcaster do if presented with the input pair (v, red)? Since we mean to let the boundary of P lead the search, this is the easy case where primal broadcasting through Q can be used (§ 3.B).

Assume now that the input is of the form (v, blue), where v is a vertex of a lace γ, and let R be the blue region to be explored. Traversing R entails leaving the polytope Q altogether. As usual, if γ is the only lace of R and we know that for a fact, everything is easy. But what if we have other laces $\gamma_1, \cdots, \gamma_l$? The difficulty is not to traverse R per se, but to tell when we might be re-entering Q and hitting upon vertices of γ_i. Primal shields are useless at this point, and we must turn to the dual shield of Q for help. Dual broadcasting from a lace of Σ to another one will be accomplished in three stages:

1. Starting from the belt of Σ^δ associated with the starting lace, navigate in dual space to the lace of its bracelet.
2. Primal broadcast through Q^δ in dual space.
3. Starting from the belt of Σ associated with a (dual space) lace newly discovered, navigate to the lace of its bracelet.

Either of the tasks performed in steps 1 and 3 is called a *mutation*: We are given a vertex on a lace of a bracelet $\mathcal{B}$, and we must find one vertex on the lace of the dual bracelet of $\mathcal{B}$. Note that from the algorithm description a mutation involves navigating from a belt to its associated lace and not the other way around. One might think that reversing the process is just dualizing it and, hence, is computationally equivalent. That is not quite true. The subtlety here relates to our previous remark about belts being facially less refined than the laces of their dual bracelets. As a result, navigating toward belts can be more difficult than toward laces (albeit still doable). We now describe an efficient implementation of a mutation. It consists of two parts: First we move to dual space, then we navigate around the boundary of the dual bracelet from somewhere on its belt to some place on its lace.

Going from the lace to its dual belt. To mutate from the lace of a bracelet $\mathcal{B}$ of Σ to the lace of its dual bracelet $\mathcal{B}^\delta$, our first action is to collect some relevant information from the input vertex v. As we indicated earlier, a vertex of a lace is never given by itself, but along with the new facet of P (or Q) and the new edge of Q (or P) upon which it is in contact. General position ensures that v is incident upon a constant

number of faces, so we can easily get two (new) facets, one in ∂P and the other in ∂Q, whose intersection contributes one edge e of the lace of $\mathcal{B}$. By duality, these two facets specify an edge ab of the belt of $\mathcal{B}^\delta$. Note that each facet contributes at least one of its own vertices to the bracelet: Which one can be determined in constant time by local examination. Dualizing these chosen vertices gives old facets f_a, f_b that, locally around a and b, lie on the boundary of $\mathcal{B}^\delta$. Note that these facets need not lie entirely in $\partial\mathcal{B}^\delta$. For example, in Fig. 3.11, the two (intersecting) triangles shown on the left dualize to the two vertices a and b shown on the right. The two vertices in primal space from which the arrows emanate point to their dual facets f_a, f_b, both of which extend beyond the dual bracelet. Instead of computing these old facets, which might be large, we retrieve one new facet within f_a (respectively, f_b) that is incident upon a (respectively, b). These are the hatched triangles in Fig. 3.11. Recall that we added a special provision to the correspondence between a polytope and its dual to make this possible in constant time.

Climbing down around the dual bracelet from its belt to its lace. We are now in possession of an edge ab of the belt of $\mathcal{B}^\delta$ and a simplicial facet A (respectively, B) incident upon a (respectively, b) that contributes a facet to $\mathcal{B}^\delta$ (but might not be one itself). Let P^* (respectively, Q^*) be the intersection of P^δ (respectively, Q^δ) with the plane π passing through O, a, b, and let H^* be the convex hull of P^* and Q^*. Note that, in general, H^* is *not* the intersection of π with the convex hull of $P^\delta \cup Q^\delta$. Because the origin lies in the interior of both P^* and Q^* and neither polygon contains the other, the curves ∂P^* and ∂Q^* must intersect. Furthermore, the closure $\mathcal{B}^*$ of the connected component of $H^* \backslash (P^* \cup Q^*)$ that contains the edge ab is the two-dimensional equivalent of a bracelet (Fig. 3.12): It is simple to analyze, so we will assume its basic properties. The edge ab is the "belt" of $\mathcal{B}^*$ (more appropriately called a *bridge*) and its "lace" is the point $p = \mathcal{B}^* \cap \partial\mathcal{P}^* \cap \partial\mathcal{Q}^*$. Using standard techniques, we can compute p by a simultaneous traversal of ∂P^* and ∂Q^* starting from a and b. With a bit of care, we can find p in time proportional to the number of vertices in $\mathcal{B}^*$.

Of course, this assumes that we have full knowledge of P^* and Q^*. But we do not, and we do not wish to. Since the boundaries of P^δ and Q^δ have been triangulated, however, it is easy to go from one edge to an adjacent one in constant time and thus achieve the same effect. To obtain the starting edge may require a little extra work, since A (respectively, B) might not intersect π (recall that a facet does not contain its incident vertices). But we know that A (respectively, B) lies in $\partial\mathcal{B}^\delta$ locally around a

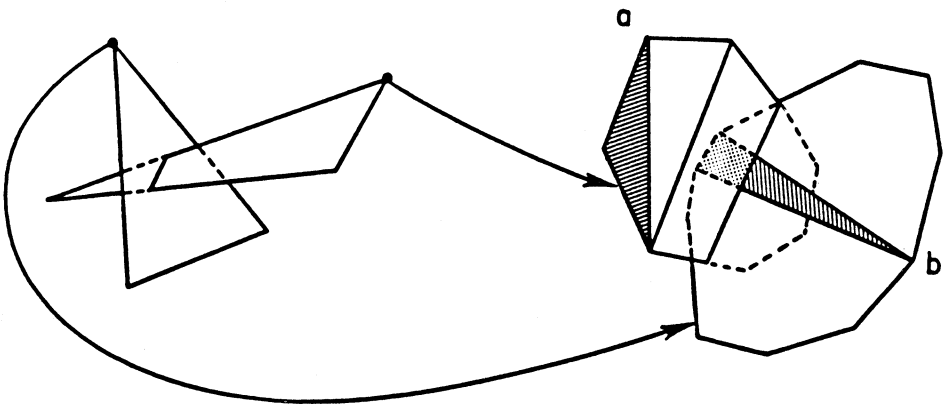

FIG. 3.11. *Two intersecting triangles (left) dualize to vertices a and b (right).*

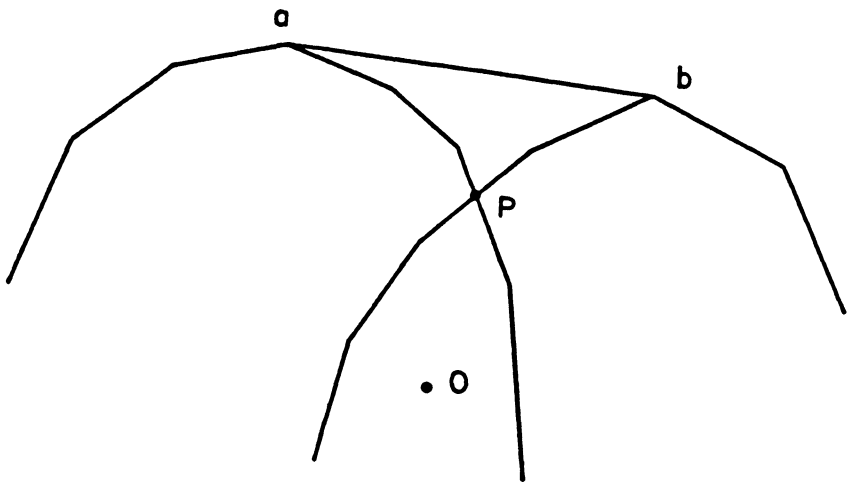

FIG. 3.12. *Two-dimensional equivalent of a bracelet.*

(respectively, b). Therefore, beginning at A (respectively, B), we can go around the cyclical order of new faces around a (respectively, b) until we find one that intersects π. If we are careful to go in the right direction, this preliminary work should involve looking only at new faces of P^δ and Q^δ that contribute to the boundary of $\mathcal{B}^\delta$. Once we have p, we also know two simplicial facets whose intersection contributes an edge to the desired lace, so the mutation is over. The total running time is at most proportional to the size of the dual bracelet.

LEMMA 3.3. *Mutating from a lace of a bracelet can be performed in time proportional to the number of edges in its dual bracelet.*

The lemma gives us all the ammunition we need to primal broadcast through Q^δ. By definition, this will reveal to us one vertex for each lace bounding the co-dual region of R. By virtue of Lemma 3.2, mutating back from each lace bounding R^δ will finally take us to the remaining laces of R and complete our dual broadcasting routine. Thus, to summarize, dual broadcasting is effected in a three-step sequence: (1) mutate to dual space, (2) primal broadcast in dual space, and (3) mutate back to primal space. One should appreciate that dual broadcasting is more than simply primal broadcasting in dual space. Another basic observation is that the input to a primal broadcast need not be a vertex of a lace: Any nonlace vertex v in a co-region of Σ (respectively, Σ^δ) will work just as well, as long as we know the location of v in the primal (respectively, dual) shield of Q. This might be handy when looking for a starting vertex.

What is the cost of broadcasting as a whole? Let κ (respectively, κ') be the maximum number of nested polytopes in the primal (respectively, dual) shield of Q whose boundaries are cut by a single (old or new) edge of P (respectively, P^δ). It follows from Lemma 2.2 that the broadcasting time will be $O((\kappa+\kappa'+1)n)$, not counting mutations. But from Lemma 3.3 the cost of all mutations is at most proportional to the number of edges in Σ and Σ^δ, which is $O(n)$. In light of Lemmas 2.1 and 3.1, we can draw the following conclusions:

LEMMA 3.4. *Given a starting vertex, we can compute the intersection of P and Q in time $O(n+\kappa n+\kappa' n)$, where κ (respectively, κ') is the maximum number of nested polytopes in the primal (respectively, dual) shield of Q whose boundaries are cut by a single edge of P (respectively, P^δ).*

What is a valid starting vertex in the context of the lemma? Any vertex on a lace of Σ or Σ^δ will do, as will any vertex of P (respectively, P^δ) that lies in the primal

(respectively, dual) shield of Q and has been located in it. It is not difficult to compute a starting vertex in linear time. Therefore, we could package our findings into yet another $O(n \log n)$ algorithm for intersecting two convex polyhedra of size n. But we can do better than that. To begin with, we must trade our full shields for k-shields.

E. USING PARTIAL SHIELDS. The only data structure we shall need is the k-shield of Q, where k is a fixed constant to be determined later. Let $Q = Q_0 \supset Q_1 \supset \cdots \supset Q_k$ and $Q^\delta = Q'_0 \supset Q'_1 \supset \cdots \supset Q'_k$ be the sequences of nested polytopes provided by, respectively, the primal and dual parts of the k-shield. Suppose that the intersections $P \cap Q_k$ and $P^\delta \cap Q'_k$ are fully available. We will show that we can emulate primal broadcasting through Q and Q^δ, even though we have only a portion of the shield at our disposal. Let us discuss the case of P and Q_k, with the understanding that the same applies to P^δ and Q'_k. Assume that both the boundaries and the interiors of P and Q_k intersect. Then, the entire theory of regions, co-regions, laces, belts, and bracelets applies verbatim to the surface $\partial(P \cup Q_k)$. Since the intersection of P and Q_k is available, we can *precompute* all primal broadcasting through Q_k anchored to P. We do this by marking the (new) facets of P and Q_k that contribute an edge to a lace of $\partial(P \cup Q_k)$. Also, for each region of $\partial(P \cup Q_k)$, we link together representative vertices of its bounding laces into a circular list. In this way, we are able to primal broadcast from any vertex of a lace in $\partial(P \cup Q_k)$ by tracing the lace in question until we hit a representative vertex. From there, we jump to all the other laces bounding the desired region in time proportional to their number. This gives us the capability to primal broadcast through the polytope Q with P as anchor, even though we might know only the outer layers of its primal shield. Obviously, the same trick can be used for primal broadcasting in dual space. Note that mutations are not affected by this change. The advantage of this new scenario is to place an upper bound of k on the values of κ and κ' in Lemma 3.4.

What now qualifies as a starting vertex? As usual, any vertex on a lace of Σ or Σ^δ will do. Another valid situation is a vertex of P (respectively, P^δ) that lies in the (possibly disconnected) set $Q \setminus Q_k$ (respectively, $Q^\delta \setminus Q'_k$), along with its location in the primal (respectively, dual) part of the k-shield of Q. Finally, any intersection between ∂Q_k (respectively, $\partial Q'_k$) and an edge of P (respectively, P^δ) and its location in the k-shield would form an appropriate starting vertex. Note that we do not extend this qualification to just any point in $P \cap \partial Q_k$ or $P^\delta \cap \partial Q'_k$ because these points might not be reached by any broadcast. Indeed, primal broadcasting anchored to P involves traversing the edges of P and *not* its facets. Thus, there could be a nonempty intersection between ∂P and ∂Q_k, even though no edge of P intersects Q_k. In that case, the primal broadcast would never make contact with Q_k, so, obviously, no point of ∂Q_k should serve as a valid starting vertex.

The intersection algorithm.

1. Check whether the interiors of P and Q intersect and conclude immediately if they do not, using the information provided by the Dobkin–Kirkpatrick algorithm. Else, pick a point O in the interior of both P and Q, and compute their dual polytopes P^δ and Q^δ.

2. Unless the anchor has already been chosen, declare P the anchor and compute the k-shield of Q. Let Q_k (respectively, Q'_k) be the innermost polytope in the primal (respectively, dual) part of the k-shield.

3. Compute $P \cap Q_k$ and $P^\delta \cap Q'_k$ recursively (the boundary case where any of the polytopes involved has constant size can be handled directly in linear time). *Crucial point*: Make Q_k and Q'_k the anchors in the recursive calls.

4. If $P \cap Q_k = P$, return P as the intersection of P and Q, and stop. If $P^\delta \cap Q'_k = P^\delta$, return Q as the intersection of P and Q, and stop.

5. If the interiors and boundaries of P (respectively, P^δ) and Q_k (respectively, Q'_k) intersect, then precompute all primal broadcasting through Q_k (respectively, Q'_k) anchored to P (respectively, P^δ).

6. Compute a starting vertex (see below).

7. Launch a broadcast from the starting vertex and pursue it until all the laces of $\partial(P \cup Q)$ have been found.

8. Use the laces to compute $P \cap Q$ explicitly.

A few comments about the algorithm are in order. Step 1 uses the linear time algorithm of Dobkin and Kirkpatrick [7]. If there is no intersection, the algorithm will say so and report the two closest points in P and Q. If P and Q intersect only at their boundaries (which, incidentally, is against our general position assumption), the Dobkin–Kirkpatrick method will still allow us to compute the full intersection in linear time. If we have a full-fledged intersection, however, the method will return a point interior to both polytopes. The dual polytopes of P and Q are easily computed in linear time. In step 2, we declare either one of the two polytopes, say, P, the anchor, unless we are responding to a recursive call, in which case the choice of anchor is forced upon us. From Lemma 2.1, the k-shield of Q can be computed in linear time. Step 3 consists of two recursive calls. As the analysis will show, switching anchors is a crucial feature of the algorithm. Failure to do so would jeopardize the linearity of the algorithm. Step 4 takes care of two trivial terminating cases. In step 5, we build the shortcuts, if any, provided by $P \cap Q_k$ and $P^\delta \cap Q'_k$.

Step 6 determines a starting vertex. If the boundaries of P and Q_k intersect, we pick a starting vertex among the vertices of $\partial P \cap \partial Q_k$ that emanate from an edge of P (if any). Then we locate the vertex in question in the primal part of the k-shield of Q. If this does not work, we try the same operation in dual space. Namely, if the boundaries of P^δ and Q'_k intersect, we pick a starting vertex among the vertices of $\partial P^\delta \cap \partial Q'_k$ that emanate from an edge of P^δ (if any). Then we locate the vertex in question in the dual part of the k-shield of Q. If this also fails, then because of step 4 we know that the skeleton (i.e., set of vertices and edges) of P (respectively, P^δ) lies entirely outside Q_k (respectively, Q'_k). Therefore, we pick a vertex v of P and check whether it lies in Q. If it does, then it must be sandwiched between Q and Q_k, so we can locate v in the primal part of the k-shield and make it the starting vertex. Otherwise, let f be a facet of P incident upon v. If the plane passing through f does not intersect Q, then its dual is a vertex of P^δ in $Q^\delta \backslash Q'_k$ and therefore qualifies as a starting vertex. Otherwise, computing the intersection allows us to identify a point w in $\partial Q \backslash P$ or $\partial P \cap \partial Q$. The cross section of P and Q by the plane passing through O, v, w consists of two convex polygons whose boundaries intersect. Any boundary intersection qualifies as a starting vertex. So, in all cases, finding a starting vertex (step 6) takes linear time. With such a vertex in hand, Lemma 3.4 tells us how to compute the intersection of P and Q. Figure 3.13 attempts to illustrate the main phases of the algorithm in two dimensions. The polytope P is the nonconvex (sorry about that) blob wiggling across the primal part of the k-shield of Q (which itself, obviously, should not be made of *disjoint* rings $\cdots$).

Complexity analysis. Put $m = p + q$, where p (respectively, q) is the total number of vertices and bounding planes in P (respectively, Q), and let $T(p, q)$ be the worst-case running time of the algorithm. If either $p = O(1)$ or $q = O(1)$, then, trivially, $T(p, q) = O(p+q)$. From Lemmas 2.1 and 3.4 we derive the general relation

$$T(p, q) = 2T(p, 3(1-1/7)^k q) + O(p+q).$$

This recurrence alone is rather ominous looking. However, the trick of switching anchors at each recursive call will now pay off. Indeed, the recurrence can be more

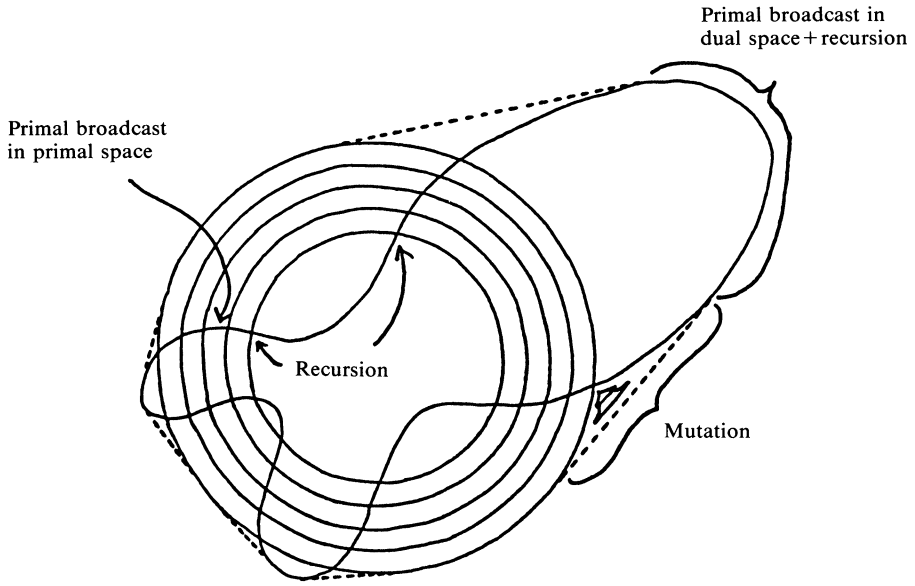

FIG. 3.13. *A bird's-eye view of the algorithm.*

accurately expressed as

$$T(p, q) = 2T'(p, 3(1-1/7)^k q) + O(p+q)$$

and

$$T'(p, q) = 2T(3(1-1/7)^k p, q) + O(p+q),$$

which, after substitution, yields

$$T(p, q) = 4T(3(1-1/7)^k p, 3(1-1/7)^k q) + O(p+q)$$

or, more simply,

$$T^*(m) = 4T^*(m/5) + O(m),$$

where $T^*(m) = T(m, m)$ and $k = 18$. We have $T^*(m) = O(m)$, and thus ends our search for a linear time algorithm for intersecting two convex polyhedra.

Reflecting back on the algorithm, it is interesting to observe that switching anchors at each recursive call makes all the difference. The process can be regarded as a form of *branching dovetailing.* A second observation is that, in the end, all the laces in both Σ and Σ^δ will have been fully computed (or at least this can be easily ensured at no extra asymptotic cost). Since the laces of Σ^δ dualize to the belts of Σ, we get the convex hull of P and Q as a bonus. Of course, another method is to compute the intersection of the dual polytopes and dualize back. If P and Q are disjoint, then we can use the Preparata–Hong linear time wrapping routine. In all cases, therefore, we are able to compute the convex hull of two convex polytopes in linear time.

THEOREM 3.5. *It is possible to compute the intersection (and the convex hull) of two three-dimensional convex polyhedra in linear time. It is assumed that the polyhedra are given in standard representation.*

4. Miscellaneous applications. The intersection algorithm can be put to use for improving or simplifying the solutions to a number of geometric problems. These applications are all very simple, so we keep our discussion to a minimum.

A. INTERSECTING SEVERAL CONVEX POLYHEDRA. Consider the problem of computing the common intersection of k convex polyhedra $P_1, \cdots, P_k$, given in standard representation. We can do this in optimal $O(n \log k)$ time, where n is the total number of vertices among the k polytopes. We use a straightforward scheme, borrowed from multi-way merging: Put the polyhedra in bijection with the leaves of a complete binary tree, and compute intersections in an order consistent with the tree. The $O(n \log k)$ running time of this algorithm is worst-case optimal even in two dimensions, because we can reduce any k-way merge to polygon intersection. To see this, consider a collection of k sorted lists $L_1, \cdots, L_k$ with distinct elements. Form the polygons $P_1, \cdots, P_k$, where P_i is the unbounded polygon defined by the intersection of the halfplanes $y \geqq x_j(2x - x_j)$: P_i is bounded by the tangents to the parabola $y = x^2$ at the points (x_j, x_j^2), for $j = 1, \cdots, m$, where $L_i = (x_1, \cdots, x_m)$. Now, observe that the boundary of the polygon $P = \bigcap_{1 \leqq i \leqq k} P_i$ contains all the points in $\{(x, x^2) \mid x \in \bigcup_{1 \leqq i \leqq k} L_i\}$. Therefore, the merged sequence of all the k lists can be read off by going around the boundary of P. To obtain the desired lower bound, we form k lists of size $m = n/k$ and observe that they can be merged in $M = \binom{n}{m_1, \cdots, m_k}$ ways, where $m_i = m$. The lower bound follows from the fact that $\log M = \Omega(n \log k)$ and by-now standard algebraic decision-tree arguments [25].

B. CONVEX HULLS. Bentley and Shamos [4] have shown how to take advantage of certain point distributions to obtain linear expected-time algorithms for computing convex hulls. The idea is to use divide-and-conquer by splitting the input set in a fixed manner (independent of the point set itself) and build the convex hull bottom-up. For their method to work efficiently, the merge step must be capable of computing the convex hull of two (possibly intersecting) convex polytopes reasonably fast. As observed by Seidel [11], we can use the Preparata–Hong algorithm for that purpose and get linear expected complexity for a wide class of point distributions. Using Theorem 3.5 widens that class. Specifically, any distribution for which the average size of the convex hull of a random set of n points is $O(n/\log^{1+\varepsilon} n)$ will trivially yield a linear expected-time complexity.

C. MERGING VORONOI DIAGRAMS. Kirkpatrick [17] has shown that two planar Voronoi diagrams can be "merged" in linear time. His algorithm is ingenious but somewhat complicated. Standard reductions cause the same result to fall straight out of Theorem 3.5. The problem is this: Given two sets of n points in the plane with their respective Voronoi diagrams, compute the diagram of their union. By using a reduction due to Edelsbrunner and Seidel [11], [14], we compute a Voronoi diagram of n points by intersecting n halfspaces. Let $h(p)$ denote the closed halfspace bounded below by the tangent to the paraboloid $z = x^2 + y^2$ at the point whose xy projection is p. The Voronoi diagram of $p_1, \cdots, p_n$ is the xy projection of the two-dimensional cell complex formed by the boundary of the convex polyhedron $\bigcap_{1 \leqq i \leqq n} h(p_i)$. Thus, merging Voronoi diagrams becomes a special case of intersecting two convex polyhedra. Applications include computing the Voronoi diagram of a polygon (Kirkpatrick [17]) and of the vertices of a convex polygon (Aggarwal et al. [1]).

5. Conclusions. Our main result is a linear-time algorithm for intersecting two convex polyhedra in 3-space. Whether the algorithm lends itself to efficient and robust implementations remains to be seen. In practice the recursion might be stopped after only a few steps, when the polytopes are small enough that we can use more naive methods. One advantage of this algorithm is that it does not use any dynamic data structure. Except for the work needed to build the shields, all other activity is purely pointer chasing through a three-dimensional cell complex. It is likely that adding randomization might lead to certain simplifications. Looking into this possibility might

be worthwhile. One must live with the fact, however, that intersecting even two tetrahedra is difficult to implement correctly, so the goal of an intersection algorithm that is truly simple to implement might be elusive. An outstanding open problem is that of intersecting two nonconvex polyhedra efficiently. The problem of intersecting arbitrarily placed triangles in 3-space has been investigated by Aronov and Sharir [2]. How much we can gain by having collections of faces structured into the boundaries of simple polyhedra is an intriguing open question.

Acknowledgments. I wish to thank Herbert Edelsbrunner, David Kirkpatrick, and Chee Yap for helpful conversations. I also thank the referees for their help in improving the presentation of the results.

REFERENCES

[1] A. AGGARWAL, L. J. GUIBAS, J. SAXE, AND P. SHOR, *A linear time algorithm for computing the Voronoi diagram of a convex polygon*, in Proc. 19th Annual ACM Symposium on the Theory of Computing, Association for Computing Machinery, New York, 1987, pp. 39–45.

[2] B. ARONOV AND M. SHARIR, *Triangles in space, or building and analyzing castles in the air*, in Proc. 4th Annual ACM Symposium on Computational Geometry, Association for Computing Machinery, New York, 1988, pp. 381–391.

[3] B. G. BAUMGART, *A polyhedron representation for computer vision*, in AFIPS Conference Proceedings, Vol. 44, AFIPS Press, 1975, pp. 589–596.

[4] J. L. BENTLEY AND M. I. SHAMOS, *Divide and conquer for linear expected time*, Inform. Process. Lett., 7 (1978), pp. 87–91.

[5] B. CHAZELLE AND D. P. DOBKIN, *Intersection of convex objects in two and three dimensions*, J. ACM, 34 (1987), pp. 1–27.

[6] D. P. DOBKIN AND D. G. KIRKPATRICK, *Fast detection of polyhedral intersection*, Theoret. Comput. Sci., 27 (1983), pp. 241–253.

[7] ———, *A linear algorithm for determining the separation of convex polyhedra*, J. Algorithms, 6 (1985), pp. 381–392.

[8] D. P. DOBKIN AND M. J. LASZLO, *Primitives for the manipulation of three-dimensional subdivisions*, in Proc. 3rd Annual ACM Symposium on Computational Geometry, Association for Computing Machinery, New York, 1987, pp. 86–99.

[9] D. P. DOBKIN AND J. I. MUNRO, *Efficient uses of the past*, J. Algorithms, 6 (1985), pp. 455–465.

[10] M. E. DYER, *Linear time algorithms for two and three-variable linear programs*, SIAM J. Comput., 13 (1984), pp. 31–45.

[11] H. EDELSBRUNNER, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Berlin, New York, 1987.

[12] H. EDELSBRUNNER AND H. MAURER, *Finding extreme points in three dimensions and solving the post-office problem in the plane*, Inform. Process. Lett., 21 (1985), pp. 39–47.

[13] H. EDELSBRUNNER AND E. P. MÜCKE, *Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms*, in Proc. 4th Annual ACM Symposium on Computational Geometry, Association for Computing Machinery, New York, 1988, pp. 118–133.

[14] H. EDELSBRUNNER AND R. SEIDEL, *Voronoi diagrams and arrangements*, Discrete Comput. Geom., 1 (1986), pp. 25–44.

[15] L. J. GUIBAS AND J. STOLFI, *Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams*, ACM Trans. Graphics, 4 (1985), pp. 75–123.

[16] S. HERTEL, M. MÄNTYLÄ, K. MEHLHORN, AND J. NIEVERGELT, *Space sweep solves intersection of convex polyhedra*, Acta Inform., 21 (1984), pp. 501–519.

[17] D. G. KIRKPATRICK, *Efficient computation of continuous skeletons*, in Proc. 20th Annual IEEE Symposium on the Foundations of Computer Science, IEEE Computer Society, Washington, DC, 1979, pp. 18–27.

[18] ———, *Optimal search in planar subdivisions*, SIAM J. Comput., 12 (1983), pp. 28–35.

[19] N. MEGIDDO, *Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems*, SIAM J. Comput., 12 (1983), pp. 759–776.

[20] K. MEHLHORN, *Data Structures and Algorithms 3: Multidimensional searching and computational geometry*, Springer-Verlag, Berlin, New York, 1984.

[21] K. MEHLHORN AND K. SIMON, *Intersecting two polyhedra one of which is convex*, University of Saarland, Tech. Report, Saarbrücken, Federal Republic of Germany, 1986.

[22] D. E. MULLER AND F. P. PREPARATA, *Finding the intersection of two convex polyhedra*, Theoret. Comput. Sci., 7 (1978), pp. 217–236.

[23] J. O'ROURKE, C. B. CHIEN, T. OLSON, AND D. NADDOR, *A new linear algorithm for intersecting convex polygons*, Comput. Graphics and Image Process., 19 (1982), pp. 384–391.

[24] F. P. PREPARATA AND S. J. HONG, *Convex hulls of finite sets of points in two and three dimensions*, Comm. ACM, 20 (1977), pp. 87–93.

[25] F. P. PREPARATA AND M. I. SHAMOS, *Computational geometry*, Springer-Verlag, Berlin, New York, 1985.

[26] C. P. ROURKE AND B. J. SANDERSON, *Introduction to Piecewise-Linear Topology*, Springer-Verlag, Berlin, New York, 1982.

[27] M. I. SHAMOS AND D. HOEY, *Geometric intersection problems*, in Proc. 17th Annual IEEE Symposium on the Foundations of Computer Science, IEEE Computer Society, Washington, DC, 1976, pp. 208–215.

[28] C. K. YAP, *A geometric consistency theorem for a symbolic perturbation scheme*, in Proc. 4th Annual ACM Symposium on Computational Geometry, Association for Computing Machinery, New York, 1988, pp. 134–142.