Shooting permanent rays among disjoint polygons in the plane

SIAM J. COMPUT.
Vol. 41, No. 4, pp. 1005–1027

SHOOTING PERMANENT RAYS AMONG DISJOINT POLYGONS IN THE PLANE*

MASHHOOD ISHAQUE†, BETTINA SPECKMANN‡, AND CSABA D. TÓTH§

Abstract. We present a data structure for ray shooting and insertion in the free space between disjoint polygonal obstacles with a total of n vertices in the plane, where each ray starts at the boundary of some obstacle. The portion of each query ray between the starting point and the first obstacle hit is inserted permanently as a new obstacle. Our data structure uses $O(n \log n)$ space and preprocessing time, and it supports m successive ray shooting and insertion queries in $O((n+m)\log^2 n + m \log m)$ total time in the real RAM model of computation. We present two applications: (1) Our data structure supports efficient implementation of *auto-partitions* in the plane, that is, binary space partitions where each partition is done along the supporting line of an input segment. If n input line segments are fragmented into m pieces by an auto-partition, then it can now be implemented in $O(n \log^2 n + m \log m)$ time. This improves the expected runtime of Patersen and Yao's classical randomized auto-partition algorithm for n disjoint line segments in the plane to $O(n \log^2 n)$. (2) If we are given disjoint polygonal obstacles with a total of n vertices in the plane, a permutation of the reflex vertices, and a half-line at each reflex vertex that partitions the reflex angle into two convex angles, then the *convex partitioning* algorithm draws a ray emanating from each reflex vertex in the prescribed order in the given direction until it hits another obstacle, a previous ray, or infinity. The previously best implementation (with a semidynamic ray shooting data structure) requires $O(n^{3/2-\varepsilon/2})$ time using $O(n^{1+\varepsilon})$ space for any $\varepsilon > 0$. Our data structure improves the runtime to $O(n \log^2 n)$.

Key words. ray shooting, tessellation, hierarchical decomposition

AMS subject classifications. 68P05, 05C10

DOI. 10.1137/100804310

1. Introduction.

Ray shooting data structures are a classical core component of computational geometry. They preprocess a set of objects in space such that one can efficiently find the first object hit by a query ray. A simple polygon with n vertices can be preprocessed in $O(n)$ time to answer ray shooting queries in $O(\log n)$ time, using either a balanced geodesic triangulation [10] or a Steiner triangulation [21]. However, the free space between disjoint polygonal obstacles with a total of n vertices (e.g., $n/2$ disjoint line segments) cannot be handled as easily: the best ray shooting data structures can answer a query in $O(n/\sqrt{s})$ time using $O(s)$ space and preprocessing, based on range searching data structures via parametric search. That is, an average query takes $O(\sqrt{n})$ time using $O(n)$ space (ignoring polylogarithmic factors). Geometric algorithms often rely on a ray shooting data structure, in which the result of

*Received by the editors August 3, 2010; accepted for publication (in revised form) June 11, 2012; published electronically August 28, 2012. A preliminary version of this work appeared in *Proceedings of the 25th Annual Symposium on Computational Geometry*, Aarhus, Denmark, 2009, ACM Press, pp. 51–60.

http://www.siam.org/journals/sicomp/41-4/80431.html

†Department of Computer Science, Tufts University, Medford, MA 02155 (mishaq01@cs.tufts.edu). This author's work was partially supported by NSF grants CCF-0431027 and CCF-0830734.

‡Department of Mathematics and Computer Science, TU Eindhoven, 5612 AZ Eindhoven, The Netherlands (speckman@win.tue.nl). This author's work was supported by the Netherlands Organisation for Scientific Research under project 639.022.707.

§Department of Mathematics, University of Calgary, Calgary, AB, T2N 1N4 Canada, and Department of Computer Science, Tufts University, Medford, MA 02155 (cdtoth@ucalgary.ca). This author's research was conducted at Tufts University, supported in part by NSERC grant RGPIN 35586.

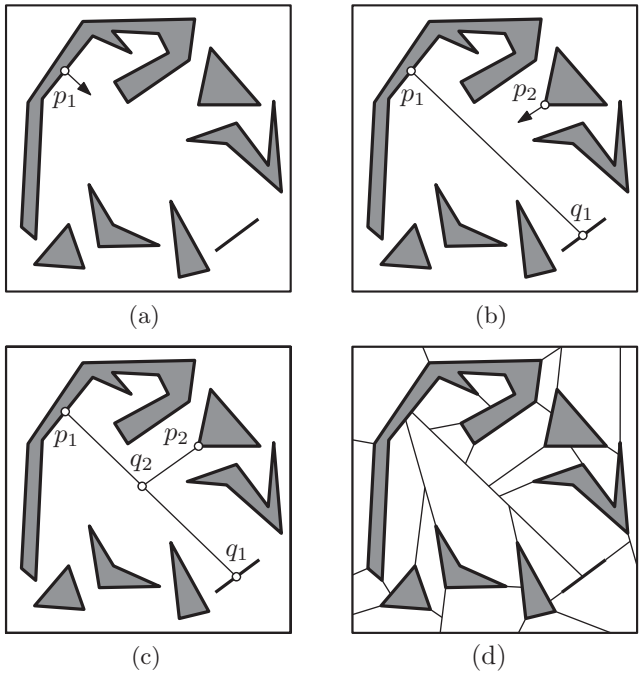


FIG. 1. (a) *Disjoint polygonal objects in the plane.* (b) *A first ray shooting and insertion query at* p_1. (c) *A second ray shooting and insertion query at* p_2. (d) *A convex partition of the free space.*

each query may affect the course of the algorithm and modify the data. In this paper, we show that by inserting successive query rays as new obstacles, the amortized query time can become polylogarithmic. To achieve this, we introduce a new hierarchical tessellation of the free space between the obstacles.

1.1. Results. We present a data structure for ray shooting and insertion queries among disjoint polygonal obstacles lying in a bounding box B in the plane. Each query is a point p on the boundary of an obstacle and a direction d_p; we report the first point q where the ray emanating from p in direction d_p hits an obstacle or the bounding box (ray shooting) *and* insert the segment pq as a new obstacle edge (see Figure 1). If the input polygons have a total of n vertices, our data structure uses $O(n \log n)$ preprocessing time and it supports m ray shooting and insertion queries in $O((n+m)\log^2 n + m \log m)$ total time and $O((n+m)\log(n+m))$ space. For instance, if $m = \Theta(n)$, the amortized query time becomes $O(\log^2 n)$, and if $m = O(1)$, it is $O(n \log^2 n)$. However, by using a tessellation of low stabbing number [9], we can reduce the cost of a single query to $O(\sqrt{n}+\log m)$. Furthermore, if the m ray shooting queries consist of $O(n)$ groups of consecutive collinear queries, we can slightly improve the total runtime to $O(n \log^2 n + m \log m)$.

1.2. Techniques. The biggest challenge for the design of our data structure was bridging the gap between the $O(\log n)$ query time for ray shooting in a simple polygon and the $O(\sqrt{n})$ query time among disjoint obstacles with n vertices. Our data structure is based on two tools which we describe in detail in the next section: geometric partition trees in two dimensions and geodesic hulls. In a nutshell, our data structure works as follows. We construct a geometric partition tree for all reflex vertices of the

free space (i.e., convex vertices of the obstacles). Based on this partition tree, we construct a novel hierarchical tiling of the free space between the obstacles. Specifically, in each convex cell of the geometric partition tree, we maintain the geodesic hull of the reflex vertices lying in the cell. This geodesic hull separates the obstacles lying in the interior of the cell from all other obstacles. The geodesic hulls associated with all nodes of the partition tree subdivide the free space into simple polygons. These simple polygons form our tiling. Each tile is bounded by a constant number of convex or reflex chains, that hence can easily be processed for fast ray shooting queries. A ray shooting query is now answered by tracing the query ray along these tiles.

The use of geodesic hulls allows us to control the total complexity of m ray insertion queries. Basically, a query ray intersects the boundary of a geodesic hull only if it partitions the set of reflex vertices into two nonempty subsets. Since a set of k points can recursively be partitioned into nonempty subsets at most $k-1$ times, we can charge the total number of intersections between rays and geodesic hulls to the number of such partition steps.

1.3. Applications. Successive ray shooting queries are responsible for a bottleneck in the runtime of some geometric algorithms, which recursively partition the plane along rays (along the portion of rays between their starting points and the first obstacles hit, to be precise). We present two specific applications.

(1) Computing binary space partitions for disjoint line segments. A *binary space partition* (BSP) for a set L of disjoint line segments in the plane is a recursive decomposition of the plane into convex cells. Each step partitions the plane along a line and recurses on the segments clipped in each open halfplane. An *auto-partition* is a BSP where each partition is made along the supporting line of an input segment [15]. Patersen and Yao [28] proved that a simple randomized auto-partition, which recursively partitions along the supporting lines of randomly selected segments, fragments the input into an expected $O(n \log n)$ fragments. A dynamic ray shooting data structure for Patersen and Yao's algorithm leads to an expected runtime of $O(n^{3/2-\varepsilon/2})$ time using $O(n^{1+\varepsilon})$ space; and an implementation with a "somewhat disappointing" runtime of $O(n^2 \log n)$ was presented in [15]. Recently, Tóth [33] presented a deterministic auto-partition algorithm that fragments the n input segments into $O(n \log n / \log\log n)$ pieces, which is the best possible [32]. Another deterministic auto-partition algorithm [31] fragments n input segments with $O(1)$ distinct slopes into $O(n)$ pieces. These auto-partition algorithms are recursive, the partition lines depend on the previous steps of the algorithm, and dynamic ray shooting data structures support $O(n^{3/2} \log n)$ runtime.

We can implement BSPs and auto-partitions by inserting the partition lines as new barriers. To partition along the supporting line of segment $\ell \in L$, shoot rays from the endpoints of ℓ, and whenever a ray hits another segment ℓ', shoot a new ray from the opposite side of ℓ' in the same direction. An auto-partition that fragments the input segments into m pieces requires $O(m)$ ray shooting and insertion queries. However, all $O(m)$ queries come in $2n$ groups of consecutive collinear queries. Our data structure supports these queries in $O(n \log^2 n + m \log m)$ time. In particular, the classical randomized auto-partition algorithm by Patersen and Yao [28] can now be implemented in $O(n \log^2 n)$ expected time. Similarly, Tóth's deterministic auto-partitions [31, 33] can now be implemented in $O(n \log^2 n)$ time.

(2) Computing a convex partition of the free space between disjoint polygons. Assume that we are given a set of disjoint polygons in the plane with a total of n vertices, a permutation of the reflex vertices of the free space, and a

half-line at each reflex vertex that partitions the reflex angle into two convex angles. The *convex partitioning* algorithm processes the reflex vertices in the specified order. For each reflex vertex, it draws a segment emanating from the vertex along the given ray until it hits another obstacle, a previously drawn segment, or infinity. Since every reflex angle is split into convex angles, the free space is decomposed into convex faces.

If we are allowed to choose the permutation π, then it is easy to compute *a* convex partition in $O(n \log n)$ time: first process simultaneously all rays pointing to the right in a left-to-right line sweep; if two segments along the rays meet, give priority to one over the other arbitrarily, then process similarly all the rays pointing to the left in a right-to-left sweep. However, in many applications [5, 6, 22], the order of the reflex vertices and the rays is given online. If π is given (either in advance or online), then previously known best data structures require $O(n^{3/2-\varepsilon/2})$ time using $O(n^{1+\varepsilon})$ space for any $\varepsilon > 0$. Our data structure improves the runtime to $O(n \log^2 n)$ and uses $O(n \log n)$ space.

1.4. Limitations. The condition that every query point p is on the boundary of an obstacle is satisfied for the applications discussed above. This condition cannot easily be relaxed using our current techniques, since our data structure is built on the geodesic hull of all reflex vertices of the free space between the obstacles. If we insert a line segment along a query ray emanating from a point v in the interior of the free space, then v becomes a new reflex vertex of the free space. Introducing m new reflex vertices can generate $\Omega(nm)$ combinatorial changes in the geodesic hull, as was recently shown in [23].

1.5. Related work. In a simple polygon with n vertices, Chazelle et al. [10] showed that a balanced geodesic triangulation can be used to answer ray shooting queries in $O(\log n)$ time. Goodrich and Tamassia [19] generalized this data structure to dynamic subdivisions defined by noncrossing line segments where each face is a simple polygon. They maintain a balanced geodesic triangulation of each face. For m segments, the data structure has $O(m)$ size. Each segment insertion and deletion, point location, and ray shooting query takes $O(\log^2 m)$ time.

Among disjoint polygonal obstacles with a total of n vertices, Bar-Yehuda and Fogel [7] gave a ray shooting data structure with $O(n \log n)$ space and $O(\sqrt{n} \log n)$ query time by connecting the obstacles with a spanning tree of low stabbing number, thereby reducing the problem to a simple polygon. The same space and query time can be achieved for possibly intersecting polygons [3, 13] using geometric partition trees and parametric search. These techniques allow tradeoffs between space and query time: for every $n \leq s \leq n^2$, one can answer ray shooting queries in $O(\frac{n}{\sqrt{s}} \text{polylog}\, n)$ query time using $O(s^{1+\varepsilon})$ space for any $\varepsilon > 0$. Applying standard dynamization techniques, Cheng and Janardan [13] devised dynamic ray shooting data structures for (possibly intersecting) polygons with a total of n vertices, which use $O(n \log^4 n)$ space, $O(\sqrt{n} \log^2 n)$ query time, and $O(\log^4 n)$ update time (i.e., segment insertion or deletion). With their data structure, m ray shooting and insertion queries can be processed in $O(m\sqrt{n} \log^4 n)$ time. Agarwal and Sharir [2] proposed a ray shooting data structure among m disjoint *convex* obstacles with $O(n^{1+\varepsilon})$ preprocessing and $O(m^{1/2} n^{1/4+\varepsilon})$ query time for any $\varepsilon > 0$. Refer to [4, 29] for higher-dimensional variants and special cases.

A tiling of the free space between disjoint polygons in the plane can serve as a certificate that the polygons do not intersect. If the tiling is easily maintainable as the polygons move, then it can be the basis for kinetic algorithms for collision detection. Kirkpatrick, Snoeyink, and Speckman [24] and Kirkpatrick and Speckmann [25] and

independently Agarwal et al. [1] and Basch et al. [8] developed kinetic data structures that are based on tilings with *pseudo-triangles*—simple polygons with exactly three convex angles. The theoretical study of geodesics in the interior of a simple polygon was pioneered by Toussaint [34, 35]. He showed that the geodesic hull $\mathrm{gh}_D(S)$ of a set S of n points in a simple n-gon D can be computed in $O(n \log n)$ time and any line segment in the interior of P crosses at most two edges of $\mathrm{gh}_D(S)$. Mitchell [26] and Ghosh [18] survey results on geometric shortest paths in the plane.

For a set of points in the plane, the fully dynamic *convex hull* data structure of Overmars and van Leeuwen [27] and the semidynamic data structures of Chazelle [12] and Hershberger and Suri [20] rely on a binary hierarchy of nested convex hulls. This idea was recently extended to a semidynamic data structure for geodesic hulls that supports point deletion and obstacle insertion in a companion paper [23]; but it works only in the special case that every face in the free space is *simply connected* and the runtime uses additional polylogarithmic factors.

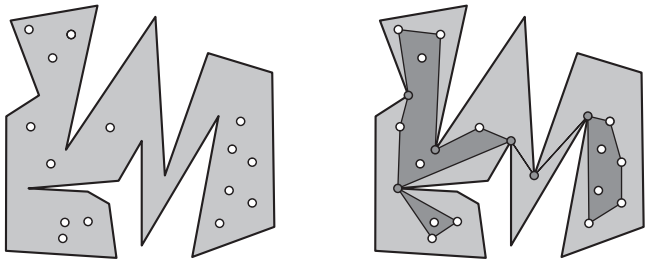
Our application in convex partitioning is reminiscent of the so-called *motorcycle problem*, introduced by Eppstein and Erickson [16]. We are given n rays in the plane, and n velocities, say d_i and $\sigma_i > 0$ for $i = 1, 2, \ldots n$. The *motorcycle graph* is constructed by drawing a line segment along each ray. Segment i is traced out by a point (motorcycle) moving continuously along the ray d_i with speed σ_i. The motorcycles start from the ray endpoints at the same time, and each motorcycle stops when it reaches the track left behind by some other motorcycle (including possible collisions). Cheng and Vigneron [14] showed that the motorcycle graph can be computed in $O(n^{3/2} \log n)$ time. Our result implies that if motorcycles run sequentially in a prescribed order (rather than simultaneously), then their tracks can be computed in $O(n \log^2 n)$ time.

2. Preliminaries. Geometric partition trees are at the core of many hierarchical data structures. A *geometric partition tree* for n points in a bounding box B in Euclidean d-space $\mathbb{R}^d$ is a rooted tree T of bounded degree, where (1) every node u corresponds to a convex cell C_u in $\mathbb{R}^d$; (2) the root, at level 0, corresponds to B; (3) the children of every node u correspond to a subdivision of C_u into convex cells; and (4) every cell C_u, $u \in T$, at level k of T contains at most $n/2^{ck}$ points, where $c > 0$ is a constant. The convex cells C_u for all leaf nodes $u \in T$ form a subdivision of the bounding box B. In particular, in a *binary* geometric partition tree for every nonleaf node $u \in T$, the cell C_u is partitioned into two convex cells along a hyperplane.

The simplest geometric partition tree for a set S of n points in the plane is a binary partition of the point set by vertical lines. All cells C_u are vertical slabs; if $|S \cap C_u| \geq 2$, then C_u is subdivided into two slabs by a vertical median of $S \cap C_u$. This vertical slab partition tree can be constructed in $O(n \log n)$ time, and it was used for dynamic convex hull computation [12] and many other early dynamic data structures [27].

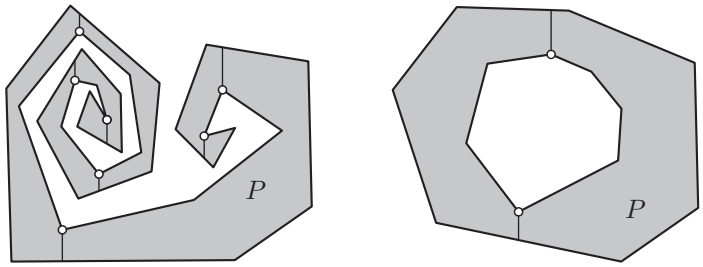
Geometric partition trees with low stabbing numbers were constructed by Chazelle, Sharir, and Welzl [11], and their result has recently been simplified and strengthened by Chan [9]. For n points in the plane, one can construct a geometric partition tree in $O(n \log n)$ time such that the total complexity of all cells is $O(n)$ and any line stabs at most $O(\sqrt{n})$ cells. Furthermore, the tree can be represented as a binary geometric partition tree where every cell is a convex polygon with at most $O(1)$ vertices.

2.1. Geodesic hulls. The geodesic hull, which is also known as a relative convex hull, was introduced in the digital imaging community [30] and later rediscovered in computational geometry (c.f. [34]). It is a key concept for the best available data struc-



(a) (b)

FIG. 2. (a) *A point set S in a simply connected polygonal domain D.* (b) *The geodesic hull* $\mathrm{gh}_D(S)$.



(a) (b)

FIG. 3. (a) *A simple crescent polygon.* (b) *A non-simple crescent polygon.*

tures for motion planning, collision detection [1, 8], and robotics [17]. The geodesic hull is a generalization of the convex hull, restricted to some simply connected domain. For a set S of points in the plane, the convex hull is the minimum set that contains S and is convex (i.e., contains the line segment between any two of its points). For a set S of points and a simply connected domain D, the geodesic hull is the minimum set that contains $S \cap D$, lies in D, and contains the shortest path with respect to D between any two of its points.

The boundary of the convex hull is a convex polygon, denoted by $\mathrm{ch}(S)$. Toussaint [35] showed that the boundary of a geodesic hull is a *weakly simple polygon*, denoted by $\mathrm{gh}_D(S)$ (albeit not necessarily a simple polygon; see Figure 2(b)). A weakly simple polygon on k vertices is a closed polygonal chain $(p_1, p_2, \ldots, p_k)$ such that for any $\varepsilon > 0$ the points p_i can be moved to a location p_i' in the ε-neighborhood of p_i so that $(p_i', p_2', \ldots, p_k')$ is a simple polygon. Toussaint [35] also showed that $\mathrm{gh}_D(S)$ has the property that any line segment lying in D intersects $\mathrm{gh}_D(S)$ in at most two points.

2.2. Crescent polygons. Data structures for ray shooting and collision detection in the plane typically use tessellations of the free space between obstacles by pseudo-triangles—simple polygons with exactly three convex angles. We construct a tessellation by crescent polygons, which are a superset of *spiral polygons*. Crescent polygons are bounded by one convex polygonal chain and at most one reflex polygonal chain (see Figure 3). A polygonal chain on the boundary of a polygon P is convex (reflex) if the interior angle of P is less (more) than π at every internal vertex of the chain. As opposed to spiral polygons, crescent polygons do not have to be simple—the two polygonal chains can be disjoint. That is, the family of crescent polygons does

not only include all spiral polygons (which, in particular, include convex polygons) but also convex polygons with a convex hole (the boundary of the hole is the reflex chain of the polygon).

We build a *monotone decomposition* to store a crescent polygon P. A reflex vertex v of P is *y-minimal* if both incident vertices are in the closed halfplane above the horizontal line passing through v. Similarly, v is *y-maximal* if both incident vertices are in the closed halfplane below the horizontal line passing through v. For each y-minimal (y-maximal) reflex vertex v, shoot a vertical ray downward (upward) into the interior of P; the ray necessarily hits the convex chain of P. The vertical rays partition P into y-monotone subpolygons (see Figure 3). Each subpolygon is bounded by two y-monotone polygonal chains: one y-monotone chain is a reflex chain together with up to two vertical segments (at most one downward and at most one upward ray), and the other y-monotone chain is a convex chain (part of the convex chain of P). We store the edges of the convex and the reflex chain of each subpolygon in a self-balancing search tree; we also store the adjacency relations between the subpolygons. For a crescent polygon with n edges, this monotone decomposition has $O(n)$ size and can easily be built in $O(n \log n)$ time. The following lemmata formulate some important properties of crescent polygons.

LEMMA 2.1. *A line segment inside a crescent polygon P intersects at most two subpolygons in the monotone decomposition of P.*

Proof. If the line segment s is vertical, then it is disjoint from the separators between subpolygons and can hence intersect at most one subpolygon. Assume now that s is not vertical and it intersects at least two subpolygons of P. Then s crosses a vertical separator t between two adjacent subpolygons, say P' and P''. Assume, without loss of generality, P' is on the left and P'' is on the right side of t, and direct s from left to right. Then s enters the interior of P' crossing t from left to right. If s exits P' on another vertical separator, then it crosses it in the same direction (left to right). However, there are at most two vertical separators on the boundary of every subpolygon, which are part of a y-monotone polygonal chain. Therefore s cannot cross any other separator adjacent to P'. Similarly, s cannot cross any other vertical separator adjacent to P''. Hence s intersects exactly two subpolygons of P. ∎

We will answer a ray shooting query among disjoint obstacles by tracing the ray along a sequence of adjacent polygons in a tessellation of the free space. A ray shooting query in a polygon P is a triple $(e, p, \overrightarrow{d})$, where e is an edge of P, p is a point on e, and $\overrightarrow{d}$ is a direction.

LEMMA 2.2. *Let P be a crescent polygon with a reflex chain of n_1 edges, a convex chain of n_2 edges, and a monotone decomposition. A ray shooting query in polygon P can be answered in $O(\log n_1)$ time if the ray hits the reflex chain and in $O(\log n_1 + \log n_2)$ time if it hits the convex chain.*

Proof. The edge e is adjacent to at most two monotone subpolygons of P by Lemma 2.1, hence we can decide in $O(1)$ time which subpolygon the ray starts from. By Lemma 2.1, the ray intersects at most two subpolygons of P. We trace the ray through the subpolygons of P. In each subpolygon P', first detect any intersection with the reflex chain of P'. This takes $O(\log n_1)$ time, since the reflex chain of P' is y-monotone and its edges are sorted by slope. If the ray intersects the reflex chain, then it ends there and we can report the first point hit. Otherwise detect any intersection with the vertical separating segments on the boundary of P' in $O(1)$ time. If the ray crosses a separator, then it enters an adjacent subpolygon. Otherwise the ray must hit the convex chain of P'. We can compute the first edge hit in $O(\log n_2)$ time. ∎

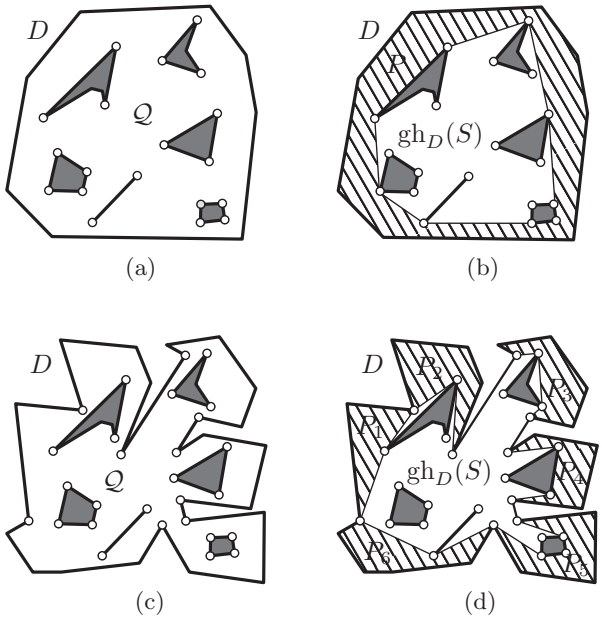


FIG. 4. (a) *A simply connected polygonal domain D containing a set $\mathcal{Q}$ of polygonal obstacles in the interior; empty circles mark the set S.* (b) *The exterior polygon is a crescent polygon P.* (c) *Another simply connected polygonal domain D containing a set $\mathcal{Q}$ of polygonal obstacles.* (d) *The exterior polygons of* $\mathrm{gh}_D(S)$ *are* $P_1, \ldots, P_6$.

2.3. Exterior and padding polygons. Let D be a simple polygonal domain, and let $\mathcal{Q}$ be a set of pairwise disjoint polygonal obstacles in the interior of D. Denote by S the set of reflex vertices of the free space $\mathrm{int}(D) \setminus (\bigcup \mathcal{Q})$ in the interior of D between the obstacles. Specifically, all reflex vertices of D are in S. Polygon D contains the geodesic hull $\mathrm{gh}_D(S)$. The space in the interior of D and the exterior of $\mathrm{gh}_D(S)$ decomposes naturally into connected components which we call the exterior polygons of $\mathrm{gh}_D(S)$; see Figure 4. We show that every exterior polygon is a crescent polygon.

LEMMA 2.3. *Every exterior polygon of* $\mathrm{gh}_D(S)$ *is a crescent polygon, where the convex chain is a maximal convex chain of D and the reflex chain is a subchain of* $\mathrm{gh}_D(S)$.

Proof. If D is convex, then $\mathrm{gh}_D(S) = \mathrm{ch}(S)$ lies in the interior of D. The boundary of the free space between D and $\mathrm{gh}_D(S)$ has two connected components: an outer boundary D and a hole $\mathrm{gh}_D(S) = \mathrm{ch}(S)$. They determine a nonsimple crescent polygon.

Now assume that D has $k \geq 1$ reflex vertices. All k reflex vertices of D are in S. The boundary of the geodesic hull $\mathrm{gh}_D(S)$ passes through the reflex vertices of D, since they are in $\mathrm{gh}_D(S)$ by definition, and they are also on the boundary of $\mathrm{gh}_D(S)$ since $\mathrm{gh}_D(S) \subseteq D$. The convex vertices of D, however, are in the exterior of $\mathrm{gh}_D(S)$, since they are not in S, and any polygonal path between two points in S that passes through a convex vertex of D can be shortened.

The two endpoints of any maximal convex chain of D are connected by two polygonal chains: the maximal convex chain of D and a polygonal chain along $\mathrm{gh}_D(S)$. The chain along $\mathrm{gh}_D(S)$ is reflex, since if it had a convex angle at an internal vertex $p \in S$, the geodesic hull of S would not contain the shortest path between two neighbors of p. If the length of a maximal convex chain of D is at least two, then the two polygonal

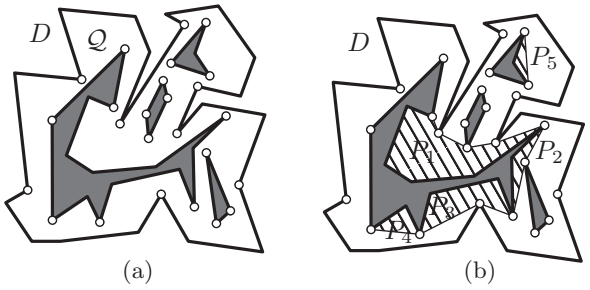


FIG. 5. (a) *A set $\mathcal{Q}$ of obstacles in a simply connected domain D.* (b) *Five padding polygons along the boundaries of the obstacles.*

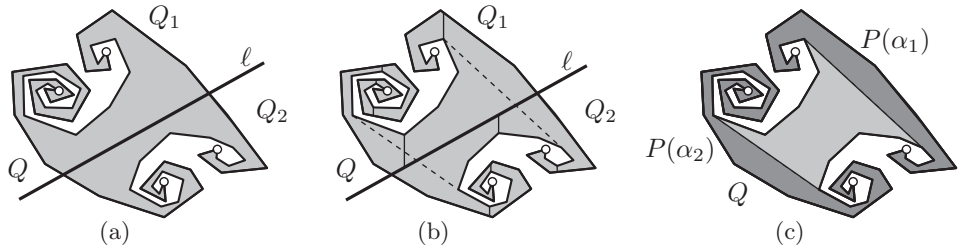


FIG. 6. (a) *Double-crescent Q decomposed into two crescent polygons.* (b) *The common external tangents of the two reflex chains in Q.* (c) *The crescent polygons $P(\alpha_1)$ and $P(\alpha_2)$ and a bridge polygon.*

chains are different and they bound a simple polygon. Otherwise both chains are reduced to the same line segment, and they do not enclose a simple polygon. □

A padding polygon (Figure 5) is a crescent polygon whose convex chain is a maximal reflex chain β of at least two edges on the boundary of an obstacle and whose reflex chain is the shortest path between the two endpoints of β homeomorphic to β in the free space $D \setminus (\bigcup \mathcal{Q})$. Padding polygons play an important role in our data structure. It is clear that the interior of a padding polygon is disjoint from the shortest path between any two points in S.

2.4. Double-crescent and bridge polygons. In our data structure, we have to deal with pairs of adjacent crescent polygons separated by a line. Let a double-crescent Q be a polygon whose boundary can be partitioned into two convex chains and two (possibly degenerate) reflex chains such that the two reflex chains are separated by a line ℓ and the line ℓ decomposes Q into two crescent polygons Q_1 and Q_2 (see Figure 6). Let α_1 and α_2 be the convex chains of Q. We define two crescent polygons $P(\alpha_1)$ and $P(\alpha_2)$ in Q. Specifically, let $P(\alpha_1)$ be the polygon bounded by α_1 and the shortest path between the endpoints of α_1 inside Q. This shortest path consists of some initial portions of the reflex chains of Q incident to the endpoints of α_1 and a common external tangent of the two reflex chains. $P(\alpha_2)$ is defined analogously, bounded by α_2 and the shortest path between the endpoints of α_2 inside Q. The crescent polygons $P(\alpha_1)$ and $P(\alpha_2)$ are adjacent to α_1 and α_2, respectively, and they are interior disjoint since they are separated by the two common external tangents of the two reflex chains of Q. If $P(\alpha_1)$ and $P(\alpha_2)$ do not fill Q entirely, then the space between them is called the bridge polygon of Q.

LEMMA 2.4. *A double-crescent Q has at most one bridge polygon, which is a simple polygon bounded by two reflex chains on opposite sides of ℓ and two common tangents of the reflex chains of Q.*

Proof. Every internal vertex of the reflex chains of $P(\alpha_1)$ and $P(\alpha_2)$ is a reflex vertex of Q. Since the reflex chains of Q are separated by a line, the overlap of the reflex chain of $P(\alpha_i)$, $i = 1, 2$, and a reflex chain of Q is a continuous polygonal chain. The reflex chain of $P(\alpha_i)$, $i = 1, 2$, consists of subchains of the two reflex chains of Q connected by their common external tangents; see Figure 6(c). If these common tangents are not identical, then $P(\alpha_1)$ and $P(\alpha_2)$ do not fill Q entirely and the remaining space is bounded by a subchain of each reflex chain of Q (a subchain may also be a single edge or a vertex) and the two common external tangents of the reflex chains. □

LEMMA 2.5. *Let Q be a double-crescent polygon Q with n vertices and ℓ be a line that decomposes Q into two crescent polygons Q_1 and Q_2, which are given with their monotone decompositions. Then the common external tangents of the two reflex chains of Q (and hence the decomposition of Q into $P(\alpha_1)$, $P(\alpha_2)$, and a possible bridge polygon) can be computed in $O(\log n)$ time.*

Proof. Each crescent polygon Q_1 and Q_2 is given with a decomposition into y-monotone subpolygons. Within each of Q_1 and Q_2, at most two subpolygons are adjacent to ℓ by Lemma 2.1. A common tangent of the two reflex chain of Q crosses line ℓ, hence it can can intersect the subpolygons along ℓ and any adjacent subpolygon in the decomposition of Q_1 and Q_2. A common external tangent of the two reflex chains of Q is a common external tangent of two reflex chains in some pair of subpolygons of Q_1 and Q_2 on opposite sides of ℓ. We have $O(1)$ pairs of subpolygons of Q_1 and Q_2 to consider. In each pair, the common tangents can be computed in $O(\log n)$ time [27], since the reflex chain in each subpolygon is y-monotone. Out of $O(1)$ common tangents of reflex subchains, we can select the common external tangents of the two reflex subchains of Q in $O(1)$ time. □

Our data structure is a hierarchical system of geodesic hulls. The edges of the geodesic hulls subdivide the free space between the obstacles into crescent and bridge polygons, where the crescent polygons are either interior or exterior to a geodesic hull. The details can be found in the next section.

3. Hierarchical tessellation. In this section, we define a hierarchical system of geodesic hulls for a set of pairwise disjoint obstacles in a bounding box B. The geodesic hulls in this hierarchy are pairwise noncrossing, and they jointly decompose the free space between obstacles into crescent polygons and double-crescent polygons. These polygons, together with the obstacles, form a tessellation of the bounding box. We can *answer* a ray shooting query by simply tracing the ray through the tessellation. In section 4, we present our semidynamic data structure for efficiently maintaining the hierarchy of geodesic hulls and the tessellation. The ray shooting and insertion queries are discussed in section 5.

3.1. Hierarchy of geodesic hulls. We are given a set $\mathcal{P}$ of pairwise disjoint polygons with a total of n vertices lying in the interior of a bounding box B (see Figure 7). Denote by $F = \text{int}(B)\backslash(\bigcup \mathcal{P})$ the free space between the obstacles and by S the set of reflex vertices of the free space (that is, convex vertices of the obstacles). Let T be a binary geometric partition tree for the point set S, where the root corresponds to the bounding box B. We augment the geometric partition tree T with additional structures. For each node $u \in T$, the following hold:

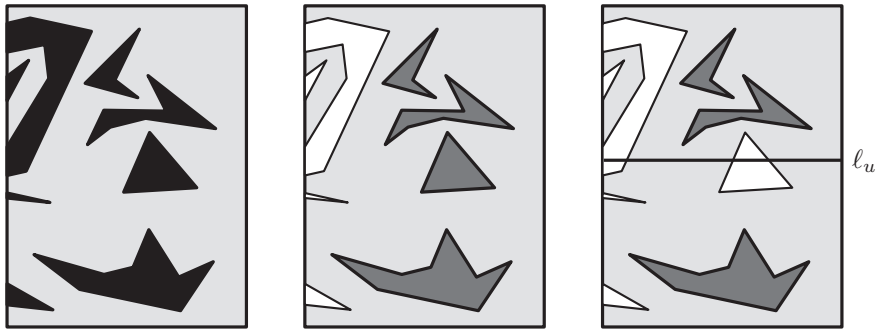


FIG. 7. (a) *A set $\mathcal{P}$ of polygonal obstacles clipped in the rectangular cell C_u.* (b) *The obstacles in $\mathcal{P}_u$ that intersect the boundary of C_u are white, the obstacles in $\mathcal{Q}_u$ in the interior of C_u are dark gray.* (c) *A line ℓ_u splits C_u into two subcells; the obstacles in $\mathcal{Q}_u^* \subseteq \mathcal{Q}_u$ intersect ℓ_u.*

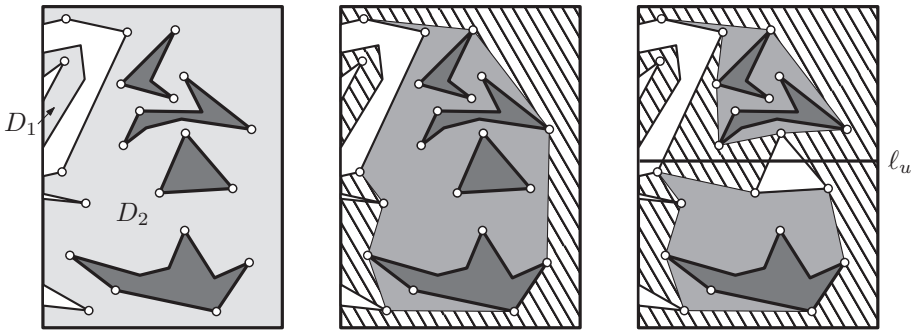


FIG. 8. (a) *The reflex vertices of the free space $C_u \setminus (\bigcup \mathcal{P})$ are marked with empty circles; the free space has two domains D_1 and D_2.* (b) *The geodesic hull $\mathrm{gh}_{D_1}(S)$ consists of a single vertex, and $\mathrm{gh}_{D_2}(S)$ is a simple polygon with 10 vertices; the exterior polygons are striped.* (c) *The splitting line ℓ_u decomposes D_2 into two domains D_2^- and D_2^+; the geodesic hulls $\mathrm{gh}_{D_2^-}(S)$ and $\mathrm{gh}_{D_2^+}(S)$. The exterior polygons on each side of ℓ_u are striped.*

- C_u is the convex cell associated with node $u \in T$; and
- ℓ_u is the line splitting cell C_u into two subcells at a nonleaf node $u \in T$.
- Let $S_u = S \cap C_u$ be the set of reflex vertices of the free space in cell C_u;
- let $\mathcal{P}_u$ denote the set of obstacles that intersect the boundary of C_u (Figure 7(b));
- let $\mathcal{Q}_u$ denote the obstacles lying in the interior of C_u (Figure 7(b));
- let $\mathcal{Q}_u^* \subseteq \mathcal{Q}_u$ denote the obstacles $\mathcal{Q}_u$ that intersect the splitting line ℓ_u. (In particular, obstacles in $\mathcal{Q}_u^*$ are in the interior of cell C_u but not in the interior of any subcell of C_u.) See Figure 7(c).

Note that the set $C_u \setminus (\bigcup \mathcal{P}_u)$ is the union of the free space clipped in C_u and all obstacles lying in the interior of C_u. Thus, every connected component of $C_u \setminus (\bigcup \mathcal{P}_u)$ is a simply connected polygonal domain.

- Let $\mathcal{D}_u$ denote the set of the components of $C_u \setminus (\bigcup \mathcal{P}_u)$ that are incident to some vertex in S_u (see Figure 8(a)).

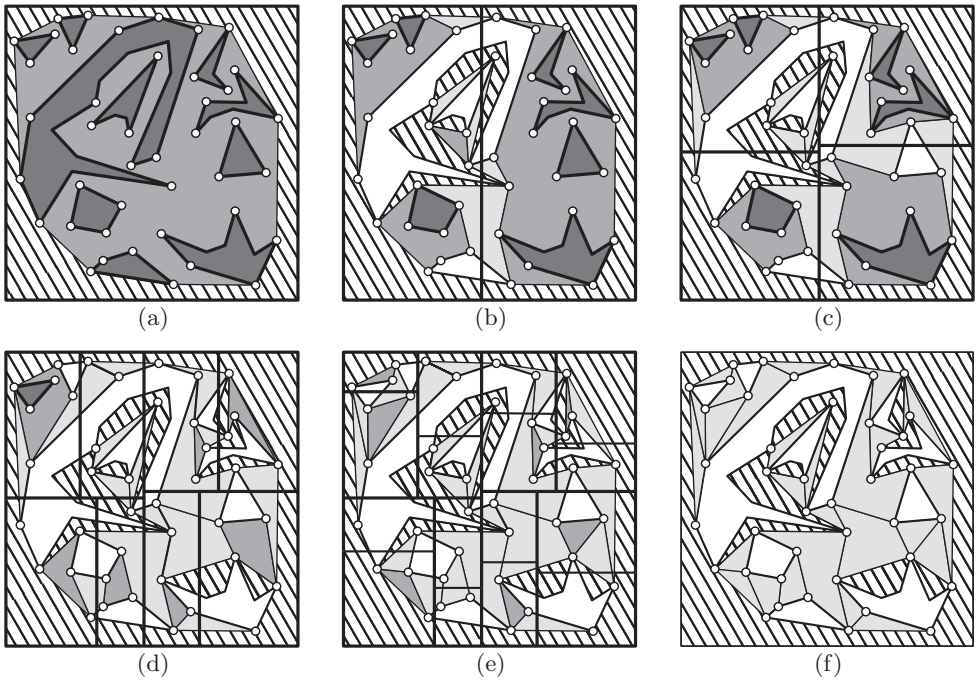


FIG. 9. (a)–(e) *The geodesic hulls* $\mathrm{gh}_D(S)$ *corresponding to the first* 5 *levels of a binary geometric partition tree* T. *Thick lines indicate the splitting lines that subdivide* B *into cells. The obstacles in the interiors of cells are dark gray; the obstacles intersecting the boundary of cells are white; the exterior and padding polygons are striped, and the geodesic hulls* $\mathrm{gh}_D(S)$ *are light gray.* (f) *The resulting tessellation of the bounding box with obstacles, exterior polygons, and bridge polygons.*

The components of $C_u \setminus (\bigcup \mathcal{P}_u)$ that are not incident to any reflex vertices of obstacles are irrelevant our data structure (and they are not in $\mathcal{D}_u$). For each polygonal domain $D \in \mathcal{D}_u$, we maintain the geodesic hull $\mathrm{gh}_D(S)$. Refer to Figure 8.

3.2. Tessellation of the bounding box B. We show that the boundaries of the geodesic hulls $\mathrm{gh}_D(S)$ defined above are pairwise noncrossing.

LEMMA 3.1. *The edges of the geodesic hulls* $\mathrm{gh}_D(S)$ *for all* $D \in \mathcal{D}_u$, $u \in T$ *are pairwise noncrossing.*

Proof. Any two cells C_u and C_v, for $u, v \in T$, are either interior disjoint or nested (that is, one contains the other). Similarly, any two domains $D_1 \in \mathcal{D}_u$ and $D_2 \in \mathcal{D}_v$, are either interior disjoint or nested. The geodesic hulls $\mathrm{gh}_{D_1}(S)$ and $\mathrm{gh}_{D_2}(S)$ are interior disjoint if the domains D_1 and D_2 are interior disjoint; and $\mathrm{gh}_{D_1}(S)$ lies inside $\mathrm{gh}_{D_2}(S)$ if $D_1 \subseteq D_2$. At any rate, the edges of $\mathrm{gh}_{D_1}(S)$ and $\mathrm{gh}_{D_2}(S)$ do not cross. □

The boundaries of the obstacles, their padding polygons, and the geodesic hulls $\mathrm{gh}_D(S)$ defined above decompose the bounding box into tiles. See Figure 9. Formally, we define a planar straight-line graph G as follows. The vertices of G are the four corners of B and all vertices of the obstacles; the edges of G are the four sides of B, the edges of all obstacles, the edges of the padding polygons, and the union of the edges of the geodesic hulls $\mathrm{gh}_D(S)$ for all $D \in \mathcal{D}_u$ and $u \in T$. By Lemma 3.1, G is a planar straight-line graph. The bounded faces of G form a tessellation of the bounding box B. Our data structure (presented in section 4) maintains the hierarchy of geodesic hulls $\mathrm{gh}_D(S)$ and the graph G.

3.3. Structural properties of hierarchical geodesic hulls. At the root $u_0 \in T$, we have $C_{u_0} = B$, $\mathcal{P}_{u_0} = \emptyset$, $\mathcal{Q}_{u_0} = \mathcal{P}$, there is a single domain $\mathcal{D}_{r_0} = \{B\}$, and the geodesic hull $\mathrm{gh}_B(S)$ is the convex hull of *all* obstacles. If $u \in T$ is a leaf, then C_u contains exactly one vertex from S and so the geodesic hull $\mathrm{gh}_D(S)$ is a single point. The binary partition tree T induces a parent-chid relationship among the domains and the geodesic hulls. Specifically, we say that domain $D \in \mathcal{D}_u$ is the parent of domain $D' \in \mathcal{D}_v$ if $u \in T$ is the parent of $v \in T$ and $D' \subseteq D$. Similarly, the geodesic hull $\mathrm{gh}_D(S)$ is the parent of $\mathrm{gh}_{D'}(S)$ if D is the parent of D'. Even though every nonleaf node $u \in T$ has exactly two children, a domain $D \in \mathcal{D}_u$ may have arbitrarily many children: If the splitting line ℓ_u decomposes D into several subdomains, then each subdomain incident to a point in S is a child of D. If, however, ℓ_u is disjoint from D, then D has exactly one child (namely, $D \in \mathcal{D}_v$).

The relationship between a geodesic hull $\mathrm{gh}_D(S)$, $D \in \mathcal{D}_u$, and its children is crucial for our data structure. We construct and update the geodesic hulls in a bottom up traversal of T, that is, we compute $\mathrm{gh}_D(S)$ assuming that all children of $\mathrm{gh}_D(S)$ have already been computed. If the splitting line ℓ_u is disjoint from $\mathrm{gh}_D(S)$, then $\mathrm{gh}_D(S)$ has a unique child, which is identical to $\mathrm{gh}_D(S)$, so no computation is necessary. Lemma 3.2 shows that every edge of $\mathrm{gh}_D(S)$ that lies entirely on one side of ℓ_u is already an edge of some child of $\mathrm{gh}_D(S)$, and hence it is enough to compute the "new" edges of $\mathrm{gh}_D(S)$ that cross the line ℓ_u. In the remainder of this section, we consider a nonleaf node $u \in T$ with two children v and w and a domain $D \in \mathcal{D}_u$ such that the splitting line ℓ_u intersects the geodesic hull $\mathrm{gh}_D(S)$.

LEMMA 3.2. *Let e be an edge of the geodesic hull $\mathrm{gh}_D(S)$ such that e not an edge of any child of $\mathrm{gh}_D(S)$.*

(i) *e lies in the union of a domain $D' \in \mathcal{D}_v$ and a domain $D'' \in \mathcal{D}_w$.*
(ii) *e lies in the union of an exterior polygon P_v of $\mathrm{gh}_{D'}(S)$ and an exterior polygon P_w of $\mathrm{gh}_{D''}(S)$.*
(iii) *Both P_v and P_w are adjacent to the splitting line ℓ_u and to the boundary ∂D of D.*
(iv) *e is the common tangent of the two reflex chains of the double-crescent polygon $P_v \cup P_w$.*

Proof. The splitting line ℓ_u decomposes the domain D into connected components, which are called subdomains. Every edge of $\mathrm{gh}_D(S)$ that lies entirely in a subdomain $D' \in \mathcal{D}_v \cup \mathcal{D}_w$ is an edge of $\mathrm{gh}_{D'}(S)$. So we may assume that edge e is not contained in any domain $D' \in \mathcal{D}_v \cup \mathcal{D}_w$; therefore e crosses the splitting line ℓ_u. Refer to Figure 10.

(i) Since e crosses ℓ_u at most once, it can intersect at most two subdomains of D, at most one on each side of ℓ_u. Denote these domains by $D' \in \mathcal{D}_v$ and $D'' \in \mathcal{D}_w$, respectively.

(ii) Since the geodesic hulls $\mathrm{gh}_{D'}(S)$ and $\mathrm{gh}_{D''}(S)$ lie in $\mathrm{gh}_D(S)$, the edge e cannot cross any edge of $\mathrm{gh}_{D'}(S)$ or $\mathrm{gh}_{D''}(S)$. It must lie in the exterior of both $\mathrm{gh}_{D'}(S)$ and $\mathrm{gh}_{D''}(S)$. Since D' and D'' are separated by the line ℓ_u, edge e can intersect at most one exterior polygon of each of $\mathrm{gh}_{D'}(S)$ and $\mathrm{gh}_{D''}(S)$. Denote these exterior polygons by $P_v \subset E_v$ and $P_w \subset E_w$, respectively.

(iii) If P_v is not adjacent to ℓ_u, then no shortest path between $S \cap D$ can enter the interior of P_v. If P_v is adjacent to ℓ_u but not adjacent to the boundary of D, then it is adjacent to some obstacles in $\mathcal{Q}_u$, which lie in the interior of D. Since every such obstacle in $\mathcal{Q}_u$ is contained in $\mathrm{gh}_D(S)$, the polygon P_v is also contained in $\mathrm{gh}_D(S)$. Therefore, if e lies in P_v, then P_v is adjacent to ℓ_u and to ∂D. By the same argument, P_w is also adjacent to both ℓ_u and ∂D.

(iv) On the boundaries of the exterior polygons P_v and P_w, the vertices of S are in two reflex chains, which we denote by $\beta_v \subset \mathrm{gh}_{D'}(S)$ and $\beta_w \subset \mathrm{gh}_{D''}(S)$, respectively.

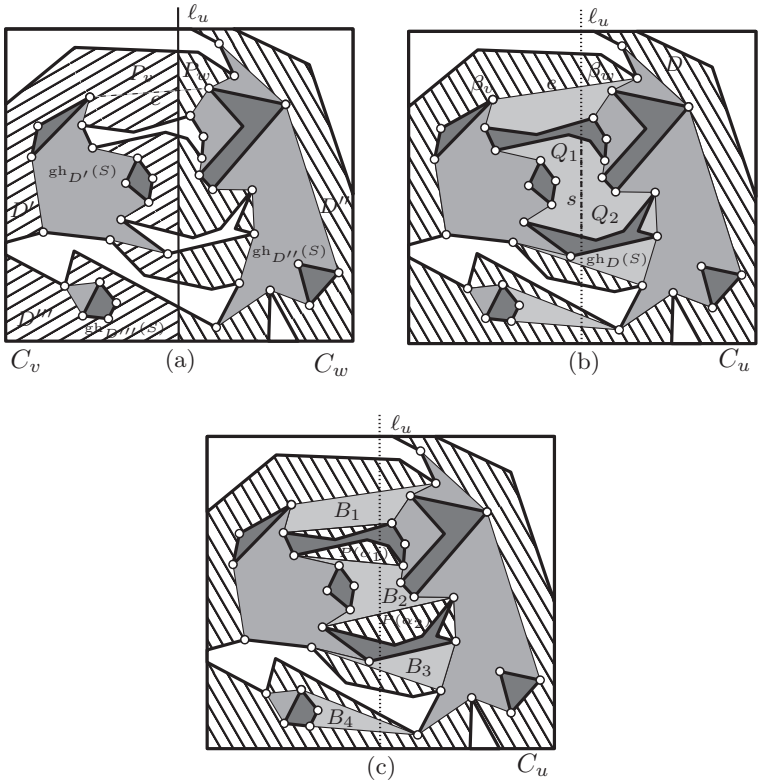


FIG. 10. (a) *Line ℓ_u split a domain into three subdomains D', D'', and D''', each containing a geodesic hull. The edge e of $\mathrm{gh}_D(S)$ lies in $D' \cup D''$ and in the union $P_v \cup P_w$ of two adjacent exterior polygons.* (b) *The geodesic hull $\mathrm{gh}_D(S)$ in domain D.* (c) *The interior of the geodesic hull $\mathrm{gh}_D(S)$ is tiled with its three children, obstacles in $\mathcal{Q}_u$, their padding polygons, and bridge polygons.*

The two chains are separated by the line ℓ_u. Since the geodesic hull of $S \cap D$ with respect to D contains the shortest path between any two points of S, it contains all line segments between the reflex chains on the boundaries of P_v and P_w. The edge e is on the convex hull $\mathrm{ch}(\beta_v \cup \beta_w)$, and so e is a common tangent of the arcs β_v and β_w. ∎

Next, we show that the faces of G are obstacles and crescent and double-crescent polygons. In is sufficient to study the tiling of the interior of a geodesic hull $\mathrm{gh}_D(S)$. The children of $\mathrm{gh}_D(S)$ are obviously contained in $\mathrm{gh}_D(S)$. The remaining space is filled by obstacles from $\mathcal{Q}_u^*$, adjacent exterior polygons, and certain bridge polygons.

LEMMA 3.3. *The interior of $\mathrm{gh}_D(S)$ is tiled by the following interior disjoint polygons:*

(i) *the children of the geodesic hull $\mathrm{gh}_D(S)$;*
(ii) *the obstacles in $\mathcal{Q}_u^*$ which are in the interior of domain D;*
(iii) *padding polygons along the maximal reflex chains along obstacles in $\mathcal{Q}_u^*$ in D;*
(iv) *bridge polygons lying in the union of two adjacent exterior polygons of two children of $\mathrm{gh}_D(S)$ on opposite sides of the splitting line ℓ_u.*

Proof. It is clear that $\mathrm{gh}_D(S)$ contains the geodesic hulls $\mathrm{gh}_{D'}(S)$ for all $D' \in \mathcal{D}_v \cup \mathcal{D}_w$, $D' \subset D$. See Figure 10. Every obstacle in the interior of C_v or C_w is contained in the geodesic hull $\mathrm{gh}_{D'}(S)$ for some $D' \in \mathcal{D}_v \cup \mathcal{D}_w$. All obstacles in $\mathcal{Q}_u$, which lie in the interior of D, are contained in $\mathrm{gh}_D(S)$. However, only the obstacles

in $\mathcal{Q}_u^*$ are not already contained in the children of $\mathrm{gh}_D(S)$. If an obstacle is contained in $\mathrm{gh}_D(S)$, then all adjacent padding polygons are also contained in $\mathrm{gh}_D(S)$, since a geodesic hull of reflex vertices cannot separate an obstacle from an adjacent padding polygon. It follows that the polygons of type (i), (ii), and (iii) are all contained in $\mathrm{gh}_D(S)$. It is clear that they are interior disjoint. It remains to show that any remaining tile is a bridge polygon described in (iv).

Delete all polygons of type (i), (ii), and (iii) from the interior of $\mathrm{gh}_D(S)$, and let P be a connected component of the remaining space. Since P is outside of the geodesic hulls $\mathrm{gh}_{D'}(S)$ for any $D' \in \mathcal{D}_v \cup \mathcal{D}_w$, it must be contained in the union of the exterior polygons of the children of $\mathrm{gh}_D(S)$. An exterior polygon of $\mathrm{gh}_{D'}(S)$, $D' \in \mathcal{D}_v \cup \mathcal{D}_w$, is a crescent polygon bounded by a convex chain β and a subchain of $\mathrm{gh}_{D'}(S)$. If β is disjoint from the boundary of C_v, then β is a maximal convex chain along an obstacle in $\mathcal{Q}_u^*$; hence the exterior polygon is a crescent polygon adjacent to an obstacle in $\mathcal{Q}_u^*$. If β touches the boundary of C_u but is disjoint from ℓ_u, then the exterior polygon is exterior to $\mathrm{gh}_D(S)$, too, and it is disjoint from P. So P is contained in the exterior polygons of some $\mathrm{gh}_{D'}(S)$, $D' \in \mathcal{D}_v \cup \mathcal{D}_w$, that lie in the interior of C_u and are adjacent to ℓ_u.

Consider two adjacent exterior polygons Q_1 and Q_2 of $\mathrm{gh}_{D'}(S)$ and $\mathrm{gh}_{D''}(S)$, respectively, on opposite sides of ℓ_u such that P intersects their union $Q_1 \cup Q_2$ (see Figure 10(b)–(c)). Let $s \subset \ell_u$ be a common edge of Q_1 and Q_2, and let α_1 and α_2 be the two maximal convex chains along obstacles that contain the endpoints of s. By Lemma 2.4, the union $Q_1 \cup Q_2$ of the two exterior polygons is covered by the padding polygons $P(\alpha_1)$ and $P(\alpha_2)$ and a possible bridge polygon R. Since P is disjoint from all padding polygons, we have $P \subseteq R$. The bridge polygon R lies in the free space and it is bounded by two geodesics of $\mathrm{gh}_{D'}(S)$, $D' \in \mathcal{D}_v$, and $\mathrm{gh}_{D''}(S)$, $D'' \in \mathcal{D}_w$, and by two of their common external tangents. Therefore, R is interior disjoint from any polygon of type (i), (ii), and (iii) and so $P = R$. □

Applying Lemma 3.3 for the geodesic hulls at all levels of the hierarchy, we can now characterize all faces of the graph G (see Figure 9).

COROLLARY 1. *The faces of graph G are obstacles in $\mathcal{P}$, crescent polygons (where the convex chain is a maximal convex chain of the free space $B \setminus (\bigcup \mathcal{P})$), and double-crescent polygons.*

A ray shooting query (p, d_p) returns a line segment pq that lies in the free space $F = B \setminus (\bigcup \mathcal{P})$, and points p and q are on the boundary of some obstacles or the bounding box. In the following lemma, we study the position of an arbitrary line segment in the free space with respect to the domains $D \in \mathcal{D}_u$, $u \in T$, and the tessellation G.

LEMMA 3.4. *Let pq be a line segment lying in the free space.*

(i) *For every $u \in T$, segment pq intersects at most one domain $D \in \mathcal{D}_u$.*
(ii) *Segment pq intersects at most two crescent polygons in the tessellation G of B.*
(iii) *For every $u \in T$, segment pq intersects at most one bridge polygon within $\mathrm{gh}_D(S)$ for some $D \in \mathcal{D}_u$.*

Proof. (i) Suppose, to the contrary, that pq contains points $a \in D_a$ and $b \in D_b$ from two distinct domains $D_a, D_b \in \mathcal{D}_u$. Then the line segment $ab \subseteq pq$ lies in the convex cell C_u. Since $D_a, D_b \subset C_u$ are separated by obstacles in $\mathcal{P}$, segment ab must traverse an obstacle, contradicting the assumption that pq lies in the free space.

(ii) Each crescent polygon in the tessellation G is bounded by a convex chain along an obstacle P and a reflex chain. If a ray $\overrightarrow{pq}$ enters a crescent polygon P, it enters through its reflex chain and hits the convex chain along P. Since the rays $\overrightarrow{pq}$

and $\overrightarrow{qp}$ can hit at most two convex chains along obstacles, pq intersects at most two crescent polygons.

(iii) Suppose, to the contrary, that pq intersects at least two distinct bridge polygons in the tiling of $\mathrm{gh}_D(S)$. Denote by $R_1, R_2 \subset D$ two consecutive bridge polygons along pq. Let $a, b \in pq$ be the closest points on the boundaries of R_1 and R_2, respectively. Since bridge polygons are not adjacent, we have $a \neq b$ and segment $ab \subset pq$ does not pass through any other bridge in $\mathrm{gh}_D(S)$. The segment ab cannot cross ℓ_u, since all points of ℓ_u in the interior of $\mathrm{gh}_D(S)$ are covered by obstacles, crescent polygons, and bridges. Hence, ab lies on one side of the line ℓ_u, and so ab lies in some domain $D' \in \mathcal{D}_v \cup \mathcal{D}_w$, $D' \subset D$. That is, ab connects two points along $\mathrm{gh}_{D'}(S)$, while the relative interior of ab is in the exterior of $\mathrm{gh}_{D'}(S)$. This contradicts the fact that the shortest path between any two points along $\mathrm{gh}_{D'}(S)$ passes in the geodesic hull of $S \cap D'$ with respect to D'. □

4. Data structure. In this section, we present our data structure and establish bounds on its space complexity and preprocessing time. We are given a set $\mathcal{P}$ of pairwise disjoint polygons with a total of n vertices lying in the interior of a bounding box B. As before, let $F = \mathrm{int}(B) \setminus (\bigcup \mathcal{P})$ denote the free space between the obstacles. Denote by S the set of reflex vertices of the free space (that is, convex vertices of the obstacles).

The backbone of our data structure is a binary binary geometric partition tree T for the point set S. At each node $u \in T$, we store the cell C_u, the splitting line ℓ_u (if u is not a leaf), the reflex vertices $S_u = S \cap C_u$, and the obstacles $\mathcal{P}_u$ clipped in C_u. Indicator variables are maintained to record whether polygons in $\mathcal{P}_u$ are in $\mathcal{Q}_u$ (lying in the interior of cell C_u) or in $\mathcal{Q}_u^* \subseteq \mathcal{Q}_u$ (lying in the interior of C_u but intersecting line ℓ_u). For every obstacle in $\mathcal{P}$, we maintain all their padding polygons along its boundary. Each padding polygon is a crescent polygon, and we maintain their monotone decomposition as described in section 2.

We also store the domains in $\mathcal{D}_u$. Each polygonal domain $D \in \mathcal{D}_u$ is stored in a doubly linked edge list (DCEL). We also maintain pointers to indicate the parent-child relationship between domains. For each polygonal domain $D \in \mathcal{D}_u$, we store the geodesic hull $\mathrm{gh}_D(S)$ and all exterior polygons of $\mathrm{gh}_D(S)$. We maintain pointers between pairs of exterior polygons that correspond to siblings and are adjacent to each other (on opposite sides of a splitting line). We store the planar straight-line graph G in a DCEL data structure. This completes the description of our data structure.

4.1. Space requirement. A binary geometric partition tree T for n points in the plane has $O(n)$ nodes and $O(\log n)$ height. A node $u \in T$ stores a convex cell C_u of constant complexity and a splitting line ℓ_u. We show next that the total complexity of the domains in $\mathcal{D}_u$ for all $u \in T$ is $O(n \log n)$. Every vertex of a domain $D \in \mathcal{D}_u$ is either a vertex of an obstacle or the intersection point of an edge of an obstacle and the boundary of the cell C_u. Since C_u has $O(1)$ edges, a domain $D \in \mathcal{D}_u$ has at most $O(1)$ consecutive vertices that are not vertices of any obstacle. Recall that a domain $D \in \mathcal{D}_u$ is stored only if it contains at least one reflex vertex in S. That is, the complexity of each domain $D \in \mathcal{D}_u$ is proportional to the number of vertices in S on the boundary of D. Every vertex in S lies in at most $O(\log n)$ cells C_u (at most one on each level of T); hence the domains $D \in \mathcal{D}_u$ for all $u \in T$ have a total of $O(n \log n)$ vertices.

All vertices of a geodesic hull $\mathrm{gh}_D(S)$ are in S. Since each vertex $p \in S$ occurs in $O(\log n)$ geodesic hulls (one at each level of T), the total complexity of $\mathrm{gh}_D(S)$ for all $D \in \mathcal{D}_u$ and $u \in T$ is $O(n \log n)$. The number of pointers representing parent-child

and adjacency relations is upper bounded by the total complexity of domains and geodesic hulls, which is $O(n \log n)$. Finally, the planar straight line graph G has n vertices and $O(n)$ edges. So the total size of all padding and bridge polygon faces of G, together with their monotone decompositions, is $O(n)$.

4.2. Preprocessing time. A binary geometric partition tree T for n points can be computed in $O(n \log n)$ time for the vertical slab tree [27] or the one with low stabbing number [9]. In a top-down traversal of the tree T, we can compute all domains $D \in \mathcal{D}_u$, the sets S_u of reflex vertices lying in C_u, and the sets $\mathcal{Q}_u$ of obstacles in the interior of C_u. At the root u_0, we have $\mathcal{D}_{u_0} = \{B\}$. If a domain $D \in \mathcal{D}_u$ intersects the splitting line ℓ_u, it is decomposed into subdomains as follows: test every edge along the boundary of D and every edge of obstacles lying in the interior of D for whether they intersect ℓ_u. Sort the intersection points of ℓ_u with the edges of ∂D along ℓ_u by traversing the boundary of the simple polygon D. Insert into this sorted list each intersection with obstacles in $\mathcal{Q}_u$ in $O(\log n)$ time. Trace out the boundary of each subdomain of D by traversing the edges of D, the portions of ℓ_u between consecutive intersection points, and the boundaries of obstacles $\mathcal{Q}_u$ crossed by line ℓ_u. The subdomains of D that are not incident to any reflex vertex in S are discarded. It takes $O(\log n)$ time per edge for sorting edges of the obstacles in $\mathcal{Q}_u$ stabbed by ℓ_u, which were in the interior of domain D but will be on the boundary of subdomains of D; and it takes $O(1)$ time for each edge of each domain $D \in \mathcal{D}_u$. Over each of the $O(\log n)$ levels of T, we spend $O(n)$ time traversing the edges of the domains and the obstacles. For each of the $O(n)$ obstacles, we spend $O(\log n)$ time when it is first crossed by a splitting line ℓ_u, and we insert it into the sorted order of the intersections of ℓ_u with the boundary of a domain. For all $u \in T$, all domains $D \in \mathcal{D}_u$, sets S_u, and sets $\mathcal{Q}_u$ in can be computed $O(n \log n)$ total time.

We compute the geodesic hulls $\mathrm{gh}_D(S)$ for all $D \in \mathcal{D}_u$, $u \in T$, and the padding polygons of the obstacles in a bottom-up traversal of T. Note that for any polygonal domain $D \in \mathcal{D}_u$, $u \in T$, all reflex vertices of D are in S_u. If $u \in T$ is a leaf, then cell C_u contains exactly one vertex of S, and the geodesic hull $\mathrm{gh}_D(S)$ is a single point. Assume now that $u \in T$ is a nonleaf node with children v and w, the geodesic hulls $\mathrm{gh}_{D'}(S)$ have already been computed for all subdomains $D' \in \mathcal{D}_v \cup \mathcal{D}_w$, $D' \subset D$, and we want to compute $\mathrm{gh}_D(S)$. By Lemma 3.2, we obtain all new edges of $\mathrm{gh}_D(S)$ and the padding polygons of the obstacles in $\mathcal{Q}_u$ contained in $\mathrm{gh}_D(S)$ by computing the common external tangents of the two reflex polygonal arcs along adjacent exterior polygons in $D' \in \mathcal{D}_v$ and $D'' \in \mathcal{D}_w$ on opposite sides of line ℓ_u. By Lemma 2.5, each common external tangent can be computed in $O(\log n)$ time. Since G is a planar graph with n vertices and $O(n)$ edges, a total of $O(n)$ common tangents are computed. Therefore all geodesic hulls $\mathrm{gh}_D(S)$, their exterior polygons, and all padding polygons can be computed in $O(n \log n)$ total time. The total size of all padding polygons in the tiling G is $O(n)$, so we can compute the monotone decompositions for all padding polygons in the tessellation in $O(n \log n)$ total time.

5. Ray tracing and update operations.

5.1. A single ray shooting and insertion query. Given a point p on the boundary of an obstacle and a direction d_p, we answer the associated ray shooting query by tracing the ray through the bridge and padding polygons of the tessellation G until it hits the boundary of an obstacle or the bounding box B. We can trace a ray through each face of G in logarithmic time in terms of the number of vertices. Recall that inserting $O(m)$ rays, the number of convex vertices is $O(n + m)$ but the

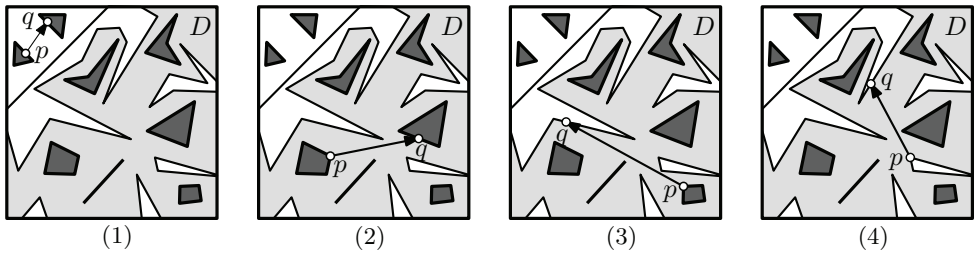


FIG. 11. *Cases for updating a domain $D \in \mathcal{D}_u$ depending on the position of the segment pq with respect to D.*

number of reflex vertices remains $O(n)$. By Lemma 2.2, a ray shooting query in a crescent or bridge polygon takes $O(\log n)$ time if the ray hits a reflex chain or the splitting line ℓ_u; and it takes $O(\log(m+n))$ time if it hits a convex chain. Since the convex chains in the padding (i.e., crescent) polygons in our data structure are on the boundary of obstacles, each ray hits at most one convex chain. So if the segment pq traverses k faces of G, then we can trace the ray trough the tessellation in $O(k \log n + \log(m+n)) = O(k \log n + \log m)$ time.

After answering the ray shooting query, we also insert the line segment pq between the starting point p and the first point q where it hits an obstacle as a new obstacle into our data structure.

If points p and q are not in the set S (that is, not reflex vertices of the free space), then the geometric partition tree T and the convex cells C_u associated with each node $u \in T$ remain the same. If point p is a reflex vertex of the free space and segment pq partitions the reflex angle into two convex angles, then we remove p from S. In this case, we also remove the leaf $u_p \in T$ associated with p and its sibling from the tree T, and we remove all additional structures and pointers associated with these nodes. The resulting tree T' is a valid binary geometric partition tree for $S' = S \setminus \{p\}$, in which the parent of u becomes a leaf corresponding to a single point in S'. Similarly, if $q \in S$ and pq partitions the reflex angle into two convex angles, then we remove q from S and update the partition tree T.

Next we update the set of domains $\mathcal{D}_u$ for all $u \in T$. As before, let v and w denote the children of a nonleaf node $u \in T$. In a top-down traversal of the tree T, detect all domains $D \in \mathcal{D}_u$ that intersect pq. By Lemma 3.4, pq intersects at most one domain $D \in \mathcal{D}_u$ for each $u \in T$. We distinguish four cases depending on the position of pq within $D \in \mathcal{D}_u$ (refer to Figure 11).

(1) If pq is disjoint from D, then no update is necessary and there is no need to descend to the polygons of $\mathcal{D}_v$ and $\mathcal{D}_w$ contained in D.
(2) If pq lies in the interior of D (that is, its endpoints lie on obstacles in the interior of D), then D is not updated, but we descend to the domains of $\mathcal{D}_v$ and $\mathcal{D}_w$ contained in D.
(3) If pq has exactly one endpoint in the interior of D, say p, which is incident to an obstacle P in the interior of D, then we update D by appending the edges $pq \cap D$ and all edges of P to the boundary of D and descend to its subdomains in $\mathcal{D}_v$ and $\mathcal{D}_w$.
(4) Finally, if pq intersects the boundary of D twice, then we split D into two domains and descend to its subdomains in $\mathcal{D}_v$ and $\mathcal{D}_w$.

We have identified all domains D that intersect the segment pq. In each domain $D \in \mathcal{D}_u$ along pq, we also update the geodesic hulls $\mathrm{gh}_D(S)$ and the padding polygons

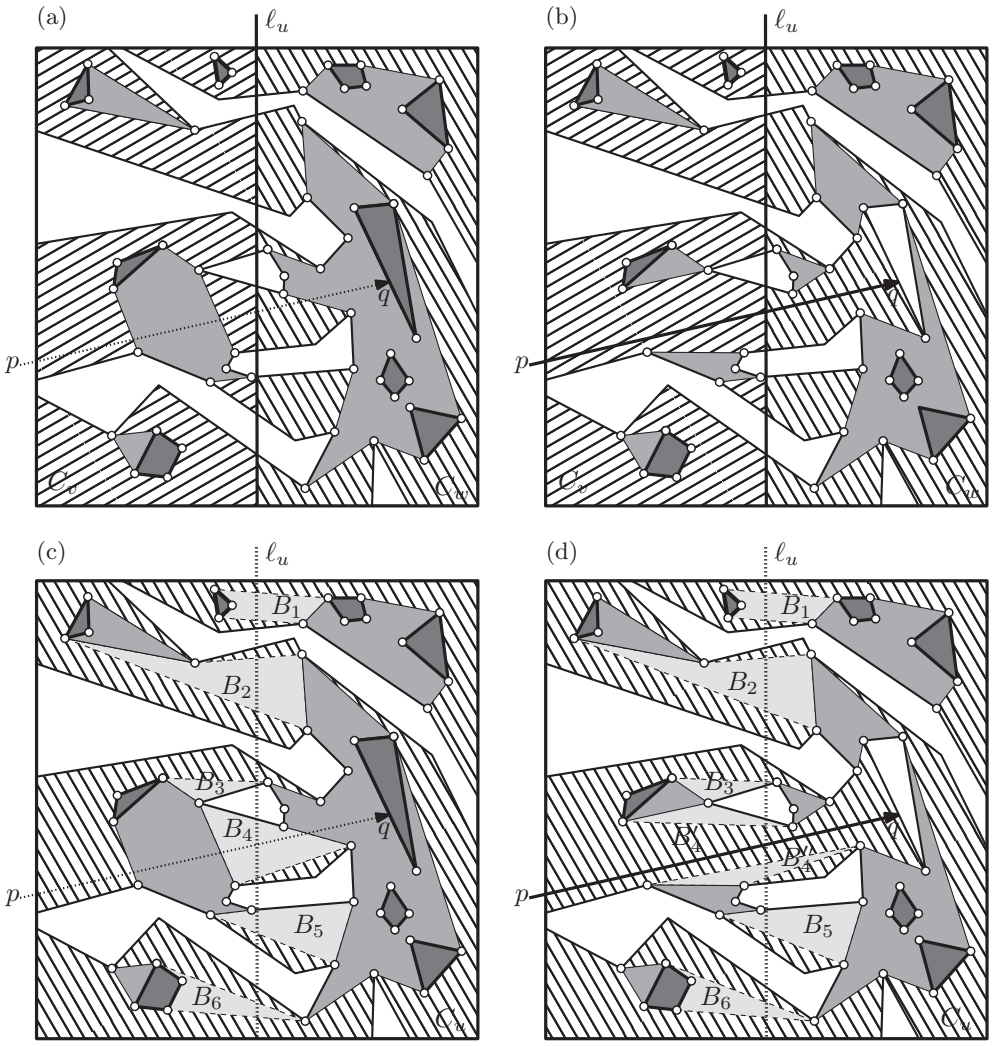


FIG. 12. *The geodesic hulls* $\mathrm{gh}_D(S)$ *for* $D \in \mathcal{D}_v \cup D_w$ *before* (a) *and after* (b) *inserting segment* pq. *The geodesic hulls* $\mathrm{gh}_D(S)$ *for* $D \in \mathcal{D}_u$ *before* (c) *and after* (d) *inserting segment* pq. *The bridge polygons are* $B_1, \ldots, B_6$. *After the insertion of segment* pq, *bridge* B_4 *is replaced by the bridges* B_4' *and* B_4'' *on opposite sides of* pq.

of the obstacles in $\mathcal{Q}_u$ lying in D. This is done in a bottom up traversal of the tree T. When processing the geodesic hull with respect to $D \in \mathcal{D}_u$, $u \in T$, we assume that the geodesic hulls $\mathrm{gh}_{D'}(S)$ have been updated for all $D' \in \mathcal{D}_v \cup \mathcal{D}_w$ at the children $v, w \in T$ of u. Note that $\mathrm{gh}_D(S)$ changes only if the domain D changes, that is, in cases (3) and (4) above. An example for case (3) is shown in Figure 12. By Lemma 3.4, segment pq crosses at most two edges of $\mathrm{gh}_D(S)$; these edges have to be removed. By Lemma 3.2, every edge of a crescent polygon or $\mathrm{gh}_D(S)$ that is not an edge of any $\mathrm{gh}_{D'}(S)$, $D' \in \mathcal{D}_v \cup \mathcal{D}_w$, is a common external tangent of two reflex arcs in two exterior polygons of some $\mathrm{gh}_{D'}(S)$ and $\mathrm{gh}_{D''}(S)$ for some $D' \in \mathcal{D}_v$ and $D'' \in \mathcal{D}_w$ that are adjacent to line ℓ_u. It follows that an update is necessary only if the two exterior polygons are adjacent to line ℓ_u and they intersect the segment pq.

There are two pairs of such extremal polygons, one on each side of pq. Compute the common external tangents between $\mathrm{gh}_{D'}(S)$ and $\mathrm{gh}_{D''}(S)$ in the two double-crescent polygons formed by these pairs of exterior polygons. In this way, we can update the geodesic $\mathrm{gh}_D(S)$ and the padding polygons of $\mathcal{Q}_u^*$ by deleting two edges crossed by pq and inserting two new common external tangents between two reflex arcs of $\mathrm{gh}_{D'}(S)$ and $\mathrm{gh}_{D''}(S)$. Since any two reflex arcs have a total of $O(n)$ vertices, we can update the geodesic hull in $\mathrm{gh}_D(S)$ in $O(\log n)$ time by Lemma 2.5.

5.2. Time complexity of m successive queries. For a segment pq in the free space, denote by $\mathrm{tr}(pq)$ the number of geodesic hulls $\mathrm{gh}_D(S)$ on all levels of T whose edges cross pq. We need to update $\mathrm{tr}(pq)$ geodesic hulls along pq (some of which may split into two parts). We have seen that the ith ray shooting query takes $O(\mathrm{tr}(p_iq_i)\log n + \log i)$ time. The insertion of the ith segment p_iq_i as a new obstacle takes $O(\mathrm{tr}(p_iq_i)\log n))$ time. The total runtime of processing m successive rays shooting and insertion queries is

$$O\left(\left(\sum_{i=1}^{m}\mathrm{tr}(p_iq_i)\right)\log n + m\log m\right).$$

LEMMA 5.1. *For m successive ray shooting and insertion queries, we have*

$$\sum_{i=1}^{m}\mathrm{tr}(p_iq_i) = O((n+m)\log n).$$

Proof. The tree T has $O(\log n)$ levels. The domains $D \in \mathcal{D}_u$ for all nodes u on one level of T are pairwise interior-disjoint, and they partition S into subsets $S \cap D$. Hence, the geodesic hulls $\mathrm{gh}_D(S)$ are pairwise disjoint on each level of T. We deduce an upper bound on the number of crossings between query segments p_iq_i and the edges of the geodesic hulls on one level of T. By Lemma 3.4, a segment p_iq_i crosses at most two edges of each geodesic hull $\mathrm{gh}_D(S)$.

We distinguish the following two cases.

(1) If p_iq_i crosses one edge of $\mathrm{gh}_D(S)$, then one endpoint of p_iq_i lies in the interior of $\mathrm{gh}_D(S)$. Since the geodesic hulls are disjoint a level of T, each point p_i or q_i lies in the interior of at most one geodesic hull $\mathrm{gh}_D(S)$. The number of this type of crossings is at most $2m$.

(2) If p_iq_i crosses two edges of $\mathrm{gh}_D(S)$, then it partitions D into two domains, each containing some vertices of S. Since a set of cardinality k can recursively be partitioned into nonempty subsets at most $k-1$ times, the sets $D \cap S$ on each level are partitioned at most $O(n)$ times. Hence the number of this type of crossings is at most $O(n)$. Summing both types of crossings over all $O(\log n)$ levels of T, we obtain $\sum_{i=1}^{m}\mathrm{tr}(p_iq_i) = O((n+m)\log n)$. □

The total runtime of m successive ray shooting and insertion queries is $O((n+m)\log^2 n + m\log m)$. If the geometric partition tree T is arbitrary, then $\mathrm{tr}(pq) = \Theta(n\log n)$ is possible and the ith ray shooting and insertion query takes $O(n\log^2 n + \log i)$ time.

However, if we use a geometric partition tree T of low stabbing number [9], then every line intersects at most $O(\sqrt{n})$ cells C_u, $u \in T$. By Lemma 3.4, a line segment in the free space intersects at most one domain $D \in \mathcal{D}_u$ and at most one geodesic hull $\mathrm{gh}_D(S)$ in each cell C_u, and so $\mathrm{tr}(pq) = O(\sqrt{n}\log n)$. Hence the ith ray shooting and insertion query takes $O(\sqrt{n}\log^2 n + \log i)$ time.

5.3. Collinear ray shooting and insertion queries. In our application in auto-partitions, many consecutive ray shooting and insertion queries are collinear. Recall that for a partition step along the supporting line of an obstacle segment $\ell \in L$, we shoot rays from the endpoints of ℓ, and whenever a ray hits another obstacle segment ℓ', we shoot a new ray from the opposite side of ℓ' in the same direction. An auto-partition that fragments n disjoint input segments into m pieces can be implemented with $2n+m$ ray shooting and insertion queries. Since there are 2n groups of consecutive collinear queries, we can slightly improve the general runtime of $O((n+m)\log^2 n + m\log m)$ to $O(n\log^2 n + m\log m)$. In particular, if $m = O(n\log n)$, then the total runtime becomes $O(n\log^2 n)$.

LEMMA 5.2. *Suppose that m successive ray shooting and insertion queries are arranged into m_0 groups of consecutive collinear queries such that if i and $i+1$ are in the same group, then points q_i and p_{i+1} are on the boundary of the same obstacle. Then we have $\sum_{i=1}^{m} \mathrm{tr}(p_i q_i) = O((n+m_0)\log n)$.*

Proof. For $j = 1, \ldots, m_0$, denote by $a_j b_j$ the convex hull of the jth group of collinear segments $p_i q_i$. The tree T has $O(\log n)$ levels. The domains $D \in \mathcal{D}_u$ for all nodes u on each level of T are pairwise interior-disjoint, and they partition S into subsets $S \cap D$. Hence, the geodesic hulls $\mathrm{gh}_D(S)$ are pairwise disjoint on each level of T.

We deduce an upper bound on the number of crossings between query segments $p_i q_i$ and the edges of the geodesic hulls on one level of T. Recall that each geodesic hull $\mathrm{gh}_D(S)$ is simply connected. If the consecutive and collinear query segments along $a_j b_j$ cross r edges of a geodesic hull $\mathrm{gh}_D(S)$ and $a_j b_j$ has 0 (resp., 1 or 2) endpoints in the interior of $\mathrm{gh}_D(S)$, then $a_j b_j$ decomposes $\mathrm{gh}_D(S)$ into $\frac{r}{2}$ (resp., $\frac{r-1}{2}$ or $\frac{r-2}{2}$) pieces; and it also partitions the set $D \cap S$ into the same number of subsets. If a_j (b_j) lies in the interior of $\mathrm{gh}_D(S)$, then we charge the nearest crossing to point a_j (b_j). We charge all other crossings to the partitioning of set $D \cap S$. Since the geodesic hulls $\mathrm{gh}_D(S)$ are pairwise disjoint on one level of T, we charge at most $2m_0$ crossings to the points a_j, b_j for $j = 1, 2 \ldots, m_0$. Since a set of cardinality k can recursively be partitioned into nonempty subsets at most $k-1$ times, we charge at most $O(n)$ to the partitioning of the sets $D \cap S$. Summing the crossings over $O(\log n)$ levels, we obtain $\sum_{i=1}^{m} \mathrm{tr}(p_i q_i) = O((n+m_0)\log n)$. ∎

6. Conclusion. We proposed a data structure for disjoint polygonal obstacles in the plane with a total of n vertices that supports m ray shooting and insertion queries in $O((n+m)\log^2 n + m\log m)$ total time and $O((n+m)\log n)$ space. Our data structure applies, with minimal adjustments, to polygons with holes having a total of n vertices. With our data structure, we improve the expected runtime of Patersen and Yao's classical randomized auto-partition algorithm for n disjoint line segments in the plane to $O(n\log^2 n)$. Also the convex partition of the free space between n disjoint line segments in the plane (with $m = n$), where the segments are extended beyond their endpoints in a specified order, can now be computed in $O(n\log^2 n)$ time and $O(n\log n)$ space. It remains a challenge for future research to find an $O(n\log n)$ time algorithm.

It is essential for our techniques that the obstacles are *polygonal*. We believe that our techniques can be extended to handle curvilinear polygons, bounded by arcs of algebraic curves of bounded degree. Typically, n such arcs can be decomposed into a finite number of locally convex or concave arcs and be approximated well enough by polygonal arcs with a total of $O(n)$ vertices. The extension of our data structure to disjoint obstacles of bounded description complexity is left for future work.

It is unlikely that our results generalize to three or higher dimensions using linear "rays." In $\mathbb{R}^2$, rays either partition the free space into two disjoint components or decrease the first Betty number (that is, the number of holes) in a single component. In $\mathbb{R}^3$, however, a linear ray never partitions the free space; it may decrease the second Betty number of a component but increase the first Betty number. A natural generalization in $\mathbb{R}^3$ would ask for efficiently computing the convex partition of the free space between polyhedral obstacles with a total of n edges, where the partition steps are governed by queries. A convex partition of the free space may have $\Theta(n^2)$ vertices. The best we can hope for, therefore, is a data structure that supports computing the convex partition in output-sensitive near-linear total time.

REFERENCES

[1] P. K. Agarwal, J. Basch, L. J. Guibas, J. E. Hershberger, and L. Zhang, *Deformable free space tilings for kinetic collision detection*, Int. J. Robotics Research, 21 (2002), pp. 179–197.

[2] P. K. Agarwal and M. Sharir, *Ray shooting amidst convex polygons in* 2*D*, J. Algorithms, 21 (1996), pp. 508–519.

[3] P. K. Agarwal, *Ray shooting and other applications of spanning trees with low stabbing number*, SIAM J. Comput., 21 (1992), pp. 540–570.

[4] P. K. Agarwal, *Range searching*, in Handbook of Discrete and Computational Geometry, 2nd ed., J. E. Goodman and J. O'Rourke, eds., CRC Press, Boca Raton, FL, 2004, pp. 809–838.

[5] O. Aichholzer, S. Bereg, A. Dumitrescu, A. García, C. Huemer, F. Hurtado, M. Kano, A. Márquez, D. Rappaport, S. Smorodinsky, D. Souvaine, J. Urrutia, and D. R. Wood, *Compatible geometric matchings*, Comput. Geom., 42 (2009), pp. 617–626.

[6] M. Al-Jubeh, M. Hoffmann, M. Ishaque, D. L. Souvaine, and C. D. Tóth, *Convex partitions with* 2*-edge connected dual graphs*, J. Comb. Optim., 22 (2010), pp. 409–425.

[7] R. Bar-Yehuda and S. Fogel, *Variations on ray shooting*, Algorithmica, 11 (1994), pp. 133–145.

[8] J. Basch, J. Erickson, L. J. Guibas, J. Hershberger, and L. Zhang, *Kinetic collision detection between two simple polygons*, Comput. Geom., 27 (2004), pp. 211–235.

[9] T. M. Chan, *Optimal partition trees*, in Proc. Sympos. on Comput. Geom., ACM Press, New York, 2010, pp. 1–10.

[10] B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink, *Ray shooting in polygons using geodesic triangulations*, Algorithmica, 12 (1994), pp. 54–68.

[11] B. Chazelle, M. Sharir, and E. Welzl, *Quasi-optimal upper bounds for simplex range searching and new zone theorems*, Algorithmica, 8 (1992), pp. 407–429.

[12] B. Chazelle, *On the convex layers of a planar set*, IEEE Trans. Inform. Theory, IT-31 (1985), pp. 509–517.

[13] S.-W. Cheng and R. Janardan, *Algorithms for ray-shooting and intersecton searching*, J. Algorithms, 13 (1992), pp. 670–692.

[14] S.-W. Cheng and A. Vigneron, *Motorcycle graphs and straight skeletons*, Algorithmica, 47 (2007), pp. 159–182.

[15] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed., Springer, New York, 2008.

[16] D. Eppstein and J. Erickson, *Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions*, Discrete Comput. Geom., 22 (1999), pp. 569–592.

[17] A. Ganguli, J. Cortes, and F. Bullo, *Multirobot rendezvous with visibility sensors in nonconvex environments*, IEEE Trans. Robotics, 25 (2009), pp. 340–352.

[18] S. Ghosh, *Visibility Algorithms in the Plane*, Cambridge University Press, New York, 2007.

[19] M. T. Goodrich and R. Tamassia, *Dynamic ray shooting and shortest paths in planar subdivisions via balanced geodesic triangulations*, J. Algorithms, 23 (1997), pp. 51–73.

[20] J. Hershberger and S. Suri, *Applications of a semi-dynamic convex hull algorithm*, BIT, 32 (1992), pp. 249–267.

[21] J. Hershberger and S. Suri, *A pedestrian approach to ray shooting: Shoot a ray, take a walk*, J. Algorithms, 18 (1995), pp. 403–431.

[22] M. Hoffmann and C. D. Tóth, *Segment endpoint visibility graphs are Hamiltonian*, Comput. Geom., 26 (2003), pp. 47–68.
[23] M. Ishaque and C. D. Tóth, *Relative convex hulls in semi-dynamic subdivisions*, Algorithmica, in print.
[24] D. G. Kirkpatrick, J. Snoeyink, and B. Speckmann, *Kinetic collision detection for simple polygons*, Int. J. Comput. Geom. Appl., 12 (2002), pp. 3–27.
[25] D. G. Kirkpatrick and B. Speckmann, *Kinetic maintenance of context-sensitive hierarchical representations for disjoint simple polygons*, in Proceedings of the 18th Symposium on Computational Geometry, 2002, pp. 179–188.
[26] J. S. B. Mitchell, *Geometric shortest paths and network optimization*, in Handbook of Computational Geometry, Elsevier Science, North-Holland, Amsterdam, 2000.
[27] M. H. Overmars and J. van Leeuwen, *Maintenance of configurations in the plane*, J. Comput. System Sci., 23 (1981), pp. 166–204.
[28] M. S. Paterson and F. F. Yao, *Efficient binary space partitions for hidden-surface removal and solid modeling*, Discrete Comput. Geom., 5 (1990), pp. 485–503.
[29] M. Pellegrini, *Ray shooting and lines in space*, in Handbook of Discrete and Computational Geometry, J. E. Goodman and J. O'Rourke, eds., 2nd ed., CRC Press, Boca Raton, FL, 2004, pp. 239–256.
[30] J. Sklansky, R. L. Chazin, and B. J. Hansen, *Minimum perimeter polygons of digitized silhouettes*, IEEE Trans. Comput., C-21 (1972), pp. 260–268.
[31] C. D. Tóth, *Binary space partition for line segments with a limited number of directions*, SIAM J. Comput., 32 (2003), pp. 307–325.
[32] C. D. Tóth, *A note on binary plane partitions*, Discrete Comput. Geom., 30 (2003), pp. 3–16.
[33] C. D. Tóth, *Binary plane partitions for disjoint line segments*, Discrete Comput. Geom., 45 (2011), pp. 627–646.
[34] G. T. Toussaint, *Shortest Path Solves Translation Separability of Polygons*, School of Computer Science, McGill University, Montreal, Canada, 1985.
[35] G. T. Toussaint, *An optimal algorithm for computing the relative convex hull of a set of points in a polygon*, in Signal Processing III: Theories and Applications, North-Holland, Amsterdam, 1986, pp. 853–856.