International Journal of Computational Geometry & Applications


# KINETIC COLLISION DETECTION FOR SIMPLE POLYGONS

DAVID KIRKPATRICK
*Department of Computer Science, University of British Columbia*
*2366 Main Mall, Vancouver, B.C., V6T 1Z4, Canada*
*E-mail: kirk@cs.ubc.ca*

JACK SNOEYINK
*Department of Computer Science, University of North Carolina at Chapel Hill*
*CB 3175 Sitterson Hall, Chapel Hill, N.C., 27516, USA*
*E-mail: snoeyink@cs.unc.edu*

BETTINA SPECKMANN
*Department of Computer Science, University of British Columbia*
*2366 Main Mall, Vancouver, B.C., V6T 1Z4, Canada*
*E-mail: speckman@cs.ubc.ca*

Received (received date)
Revised (revised date)
Communicated by (Name of Editor)

We design a simple and elegant kinetic data structure for detecting collisions between polygonal (but not necessarily convex) objects in motion in the plane. Our structure is compact, maintaining an active set of certificates whose number is proportional to a minimum-size set of separating polygons for the objects. It is also responsive; on the failure of a certificate invariants can be restored in time logarithmic in the total number of object vertices. It is difficult to characterize the efficiency of our structure for lack of a canonical definition of external events. Nevertheless we give an easy upper bound on the worst case number of certificate failures.

*Keywords*: collision detection; kinetic data structures; minimum link separation

## 1. Introduction

Algorithms and data structures for long-running simulations or continuous monitoring applications do not always fit into the style of big-$O$ asymptotic analysis that is based on batch processing (input, processing, output). The alternatives of amortized analysis, competitive analysis, or experimental evaluation do not always suit. Amortized analysis can hide unacceptably large per-operation costs, competitive analysis is difficult even on well-established algorithms like LRU page replacement, and experimental evaluations are difficult to compare and can obscure the portable ideas and concepts with non-portable implementation details. For this reason, we find the "kinetic data structures" (KDS) framework, which supports the design and analysis of algorithms and data structures that monitor properties of data in motion [4, 5], to be an exciting development in computational geometry. A kinetic data structure contains a set of certificates that constitutes a proof of the property that is to be maintained or monitored. Guibas [12] calls this "animating a proof of correctness through time."

Consider the problem of detecting collisions between moving objects in the plane, for example. We will assume that we have $k$ disjoint simple polygonal objects, consisting of a total of $N$ vertices. Each object has a known motion trajectory or *flight plan*, which may be subject to restrictions to make analysis tractable. Two common restrictions would be translation along paths parameterized by polynomials of fixed degree $d$, or translation and rotation described by algebraic curves.

Most approaches to collision detection work in two phases. First, a "broad phase" [14] (filtering) algorithm and data structure are used to determine pairs of objects that might possibly collide. A different "narrow phase" (refinement) algorithm and data structure then tests each pair. In general, two-phase approaches force the objects into a representation suited for the computer in the broad phase, and typically apply brute force computation on each pair in the narrow phase.

A KDS for collision detection, on the other hand, finds a set of geometric tests (elementary certificates) that together provide a proof that the input objects are disjoint (e.g. there exists a set of closed polygonal chains in the complement of the object set that separate the objects). Each certificate is placed in a priority queue based on the time of expiration calculated from the current flight plans. When a certificate expires – its test no longer holds – or when a flight plan is changed, the data structure must be updated.

Kinetic data structures and their accompanying maintenance algorithms can be evaluated and compared with respect to four desired characteristics. A good KDS is *compact* if it uses little space in addition to the input, *responsive* if the data structure invariants can be restored quickly after the failure of a certificate, *local* if it can be updated easily when the flight plan for an object changes, and *efficient* if the worst-case number of events handled by the data structure for a given motion is small compared to some worst-case number of "external events" that must be handled for that motion.

Our goal in looking for a kinetic data structure was to find a uniform structure that did not need to distinguish broad and narrow phases, but could adapt to the separation between objects while maintaining a minimal number of certificates. Furthermore our structure should handle several objects inside the convex hull of one (large) object. Such a structure might be suitable for dynamic data in Geographic Information Systems where the objects may have vastly different scales. Consider tracking in a marine environment, where the important question may be whether a boat runs aground on the land or whether several boats inside a bay collide. In this paper, we propose a simple and elegant structure for collision detection that fulfills our goals and is compact, responsive, and as local as possible.

The definition of our structure is developed over the next two sections of the paper. We start with $k$ disjoint polygonal objects and a triangulation of the free space between the objects. Note that the triangles in this triangulation taken together provide a proof of the disjointness of the objects in the sense that the dual of the triangulation contains (as a subgraph) a set of separating chains. This proof remains valid as long as the triangles remain *vital* (non-degenerate). Our motivation is to provide a similar but significantly more compact kinetic data structure by maintaining a coarser vital tessellation of the free space.

We show how to select an subset of $(2k - 2)$ triangles, referred to as *partition triangles*, whose removal serves to partition the free space into a set of at most $(3k - 3)$ regions called *corridors* (see Figure 1). Corridors have the property that they are bounded by polygonal chains from at most two objects. We show that the vitality of a corridor (the disjointness of its associated pair of chains $c_1$ and $c_2$) can be certified by a collection of $O(\kappa)$ certificates, where $\kappa$ is the size of a minimum link separator (cf [25]) for $c_1$ and $c_2$.

To maintain a proof of the disjointness of the polygonal objects it suffices to maintain elementary certificates for the vitality of the partition triangles and their associated corridors. Our structure is arguably minimally compact, requiring a number of elementary certificates proportional to the minimum number of line segments needed to form, for each object, a polygon that separates it from the other objects.

Our structure is also responsive. When a certificate fails, the invariants can be restored in $O(\log N)$ time. The details are many, but not otherwise difficult to implement.

For collision detection, no structure will be local unless locality is defined carefully. For example, consider a scene with one large object and $k-1$ smaller objects in close proximity to the large object, but not to one another. It is hard to imagine a proof of disjointness that would not require $\Theta(k)$ of its certificates to be updated when the large object changes its flight plan. Our structure, on the other hand, is local in a proximity-sensitive manner: when a flight plan for an object $P$ is changed, the number of certificates affected is proportional to the sum, over all corridors $C$ incident on $P$, of the size of the minimum link separator for the chains associated with $C$.

The main difficulty for establishing efficiency is inherent with the problem of collision detection itself: there are no canonical external events. When the problem specifies a structure to be maintained, such as the Voronoi diagram, then external events are easy to identify as changes to this structure. In collision detection, the desired output is "no collision" forever. However we are able to give straightforward bounds on the total number of events in the worst case. Furthermore the properties of minimality and hysteresis exhibited by our data structure suggest that it might be even more efficient in practice.

The ideas behind our structure come from a number of different results in computational geometry. The basic idea of decomposing the free space between objects into trapezoids or triangles is a common approach in computational geometry for dealing with one object moving in a static scene. There are many examples: point location in Delaunay triangulations [22], retraction motion planning using the Voronoi diagram [19, 21], shortest or minimum link paths computed in triangulated polygons [13], and trapezoidal decompositions for building arrangements [20]. When only one object is moving, configuration spaces can be used to reduce that object to a point [18, 24]. Decomposing into corridors, whose number is proportional to the number of objects rather than object complexity, can be found in Kapoor and Maheshwari's work [15, 16] on finding shortest paths in simple polygons with holes.

When multiple objects are moving, then it is more common to assume that all the objects are points [23, 6]. One recent exception is work of Guibas et al. [11] that studies the maintenance of the compact Voronoi diagram of McAllister et al. [19] for a set of disjoint convex polygons moving in the plane. Their work is different from ours in two ways: it specifically maintains the Delaunay triangulation, which is more difficult, but it restricts itself to convex objects, which is considerably easier.

Erickson et al. [9] have developed a kinetic collision detection structure for convex objects whose performance is sensitive to the amount of separation between the objects relative to their diameter. In a similar way, the kinetic data structure introduced in this paper can be modified to provide comparable separation sensitive bounds for kinetic collision detection of non-convex polygonal objects [17].

Agarwal et al. [2], extending work by Basch et al. [3], design a kinetic data structure for collision detection between simple polygonal objects that is based on a balanced geodesic triangulation [7]. Their structure maintains a number of certificates proportional to $\log N$ times the number of segments in a minimum link subdivision separating the objects. This logarithmic loss of efficiency is due to the hierarchy depth of the balanced geodesic triangulation. On the other hand, their structure is canonical, and exploiting this they are able to establish some non-trivial bounds on efficiency.

The options for comparing kinetic data structures are described in Agarwal et al. [1], where the authors consider lower bounds for classes of kinetic data structures.

## 2. Definitions and Basic Properties

Suppose we are given $k+1$ ($k \geq 2$) disjoint simple polygonal objects in the plane including an isolated point at infinity. We denote by $\mathcal{E}$ the free space between the objects (i.e. the complement of their union). An edge $e$ joining two object vertices is called a *(free space) cut* if its interior lies in the interior of $\mathcal{E}$. A cut is said to be *productive* if it joins vertices of distinct objects or if it joins two vertices of the same object and partitions $\mathcal{E}$ into two components neither of which is simply connected. Note that, because of the point at infinity, exactly one of the components associated with a non-productive cut $e$ is simply connected; this component is referred to as the *pocket* associated with $e$. Since every cut in the interior of a pocket $X$ is non-productive, each such cut has its own associated pocket which is nested within $X$. Thus the pocket inclusion relation induces a rooted forest structure on any collection of interior-disjoint non-productive cuts.

A triangle formed by three polygon edges or cuts is said to be *vital* if its interior is simply connected. A vital triangle formed by three productive cuts is called a *partition triangle*. Note that it is possible that two or three vertices of a partition triangle belong to the same object.

A *boundary chain* is a sequence of one or more vertices where successive vertices correspond to boundary edges of a fixed object. A simply connected region of $\mathcal{E}$ bounded by two boundary chains and two cuts is called *(free space) corridor*. Note that both boundary chains associated with a corridor can be part of the same object. We refer to the two cuts associated with a corridor as its *end-edges*. A corridor is said to be *degenerate* if its two associated boundary chains each contain only one vertex. A corridor is *vital* if its two associated boundary chains are disjoint.

Partition triangles are named for the fact that their removal decomposes the free space into disjoint regions (see Figure 1). The essential property of partition triangles is captured in the following lemma and its corollary.

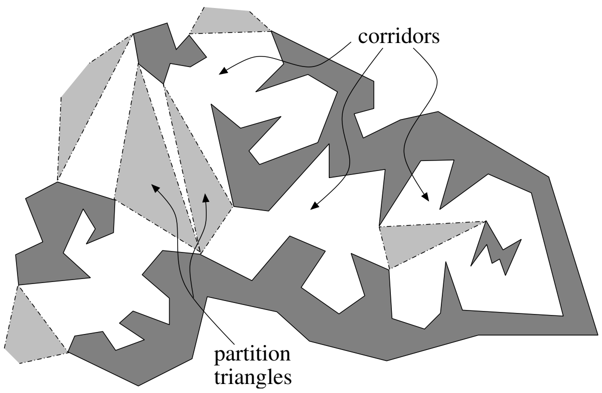


Figure 1: Decomposition of the free space between polygons.

**Lemma 1** *Every component of free space formed by the removal of a collection of interior-disjoint productive cuts either contains a partition triangle or forms a corridor.*

**Proof:** Suppose this is not the case. Let $R$ denote any minimal (under the inclusion ordering) counterexample. First, note that since $k \geq 2$, there must be at least one partition triangle in the full free space (in fact, one of the triangles formed by the point at infinity and an edge of the convex hull of the remaining objects must be a partition triangle). Thus $R$ must have at least one productive cut $f$ on its boundary.

Let $\mathcal{T}$ be any triangulation of $R$. Consider the triangle $T$ containing edge $f$.

At least one of the other edges of $T$, say $g$, must be productive, since otherwise $f$ would not be productive. The removal of $g$ partitions $R$ into two smaller regions neither of which can contain a partition triangle and so both of which, by the minimality of $R$, must be corridors. But the union of these two corridors, which share $g$ as a common end-edge, must be another corridor, contradicting our assumption. □

**Corollary 2** *The removal of any maximal set of interior-disjoint partition triangles decomposes the free space into corridors whose end-edges coincide with partition triangle edges.*

It turns out that *every* triangulation contains a minimum size maximal set of partition triangles.

**Theorem 3** *Any triangulation $\mathcal{T}$ of the free space $\mathcal{E}$ contains exactly $2k-2$ partition triangles. The removal of these partition triangles decomposes the free space into at most $3k-3$ non-degenerate corridors. Furthermore, given $\mathcal{T}$, the partition triangles and associated corridors can be identified in time linear in the size of $\mathcal{T}$.*

**Proof:** Let $t$ denote the number of partition triangles of $\mathcal{T}$. If a boundary chain associated with some corridor is degenerate (a single vertex) then replace this vertex by a pair of vertices joined by a zero length edge. In this way ($i$) the partition triangles all become vertex disjoint, ($ii$) the partition triangles decompose the object boundaries into a total of $3t$ chains, and ($iii$) each corridor is bounded by 2 partition triangle edges and 2 boundary chains.

The planar graph formed by the $3t$ partition triangle vertices, the $3t$ partition triangle edges, and the $3t$ boundary chains has exactly $t$ triangular faces, $k+1$ object faces, and $3t/2$ corridors. It follows directly from Euler's formula that $t = 2k-2$.

The edges of $\mathcal{T}$ can be classified into polygon boundary edges, productive cuts and non-productive cuts in time proportional to the size of $\mathcal{T}$ using the fact that non-productive cuts satisfy a tree ordering based on pocket inclusion. □

**Observation 4** *Each object $P$ is surrounded by an alternating sequence of partition triangles and (possibly degenerate) corridors, whereby $P$ contributes at least one vertex to each of the triangles and each of the corridors.*

We call a condition that states that a triangle $T$ in free space has a positive area (a necessary but not sufficient condition for a triangle to form a partition triangle) a *triangle certificate*. Similarly, a *corridor certificate* for a region $R$ bounded by two cuts and two boundary chains asserts that there exists a polygonal chain, whose interior lies in the interior of $R$, joining the two cuts and separating the two chains associated with $R$.

Triangle certificates are equivalent to the determinant sign test for the orientation of three points [8], so they are polynomials in the coordinates of the points.

Corridor certificates will be reduced to a set of primitive algebraic tests in the next section.

We can show that, for any continuous motion of the objects, a triangle or corridor certificate will fail before any pair of objects intersects. Specifically, a given configuration of disjoint polygons with partition triangles (and induced corridors) has an associated *topological structure* described by a planar graph whose (*i*) vertices correspond to partition triangles; (*ii*) edges correspond to separating chains joining end-edges of corridors, where the order of incidence of edges on a given vertex is given by the order of the end-edges on the associated partition triangle; and (*iii*) faces each contain exactly one polygon in their interior.

The following theorem, which is an immediate consequence of the definitions and observations above, allows us to focus on kinetic data structures that maintain triangle and corridor certificates.

**Theorem 5** *The topological structure of a given configuration of polygons is invariant under motions of the polygons that do not violate any triangle or corridor certificate.*

## 3. Corridor structure

To establish and maintain a corridor certificate for a region of free space it would suffice to maintain a vital triangulation of that region. Our goal, however, is to minimize the number of elementary certificates that must be maintained and hence we exploit a sparser tessellation, based on the notion of pseudo-triangles, that serves the same end.

A *pseudo-triangle* is a simple polygonal region bounded by three concave chains (called the *side chains*) that meet at their common endpoints (called the *corners*) of the pseudo-triangle. A pseudo-triangle is *vital* if the interior angle formed at each of its corners is greater than $0°$ and less than $180°$ and the interior angle formed at each of its other vertices is at least $180°$.

Following Basch et al. [3], we note that the side chains of a vital pseudo-triangle intersect only at their endpoints and that a pseudo-triangle remains vital as long as none of its vertex/angle constraints is violated. Accordingly, we refer to the angle constraint ($> 0°$) at pseudo-triangle corners as *corner certificates* and angle constraints ($\geq 180°$) at all other vertices as *side certificates.*

Suppose that we are given a corridor $C$ with two bounding polygonal chains labeled $\gamma$ and $\delta$. Recall that $\gamma$ and $\delta$ may or may not belong to the same object. In any triangulation of $C$ we can distinguish two different kinds of edges: *bridges* connecting $\gamma$ with $\delta$ and *chords* connecting $\gamma$ with $\gamma$ or $\delta$ with $\delta$. Edges of $\gamma$ and $\delta$ are in some sense degenerate chords; all other chords are referred to as *proper* chords. Similarly, end-edges are degenerate bridges and all other bridges are referred to as *proper* bridges.

Since the boundary of $C$ contains vertices of $\gamma$ or $\delta$, any triangle in $C$ contains an even number of bridges (and an odd number of chords). Thus, in any triangulation of $C$ the triangles adjacent to bridges form a *pathway* as illustrated in Figure 2*; their corresponding vertices in the dual graph of the triangulation form a single path between the partition triangles delimiting the corridor. The edges of this path are exactly the duals of bridges.

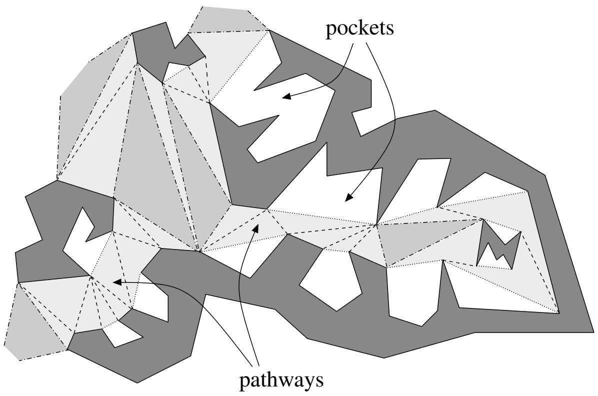


Figure 2: The decomposition of the corridors into pathways and pockets.

Like a corridor, a pathway is said to be *vital* if its two bounding chord chains are disjoint. It turns out that the vitality of corridors reduces to the vitality of pathways.

**Observation 6** *A corridor is vital if and only if it contains a vital pathway.*

Thus, to certify the vitality of a corridor over time it will suffice to maintain a vital pathway within the corridor. The vitality of a given pathway itself can be certified by the vitality of its constituent triangles. A more succinct certificate of vitality depends on a coarser pseudo-triangulation of the pathway described in the next subsection. Recall that each pathway triangle contains exactly two bridges (and one chord), two pathway triangles are either disjoint or intersect exactly on one bridge, each proper bridge belongs to exactly two pathway triangles, and each end-edge belongs to one pathway triangle and one partition triangle. Similarly, our pathway pseudo-triangles contain exactly two bridges (and one or more chords, not two of which meet at a pseudo-triangle corner), two pathway pseudo-triangles are either disjoint or intersect on exactly one bridge, each proper bridge belongs to exactly two pathway pseudo-triangles, and each end-edge belongs to exactly one pathway pseudo-triangle. Since all but at most four vertices of each pathway pseudo-triangle have incident edges from the same object, their associated side certificates never change (under our assumption of rigid motions). Hence, the vitality of pathway

*In Figure 2 and those that follow we depict the four different types of edges that appear in our tessellation of free space as follows: polygon boundary edges (solid), proper chords (dotted), partition triangle edges, which are corridor end-edges (dash-dotted), and proper bridges (dashed).

pseudo-triangles can be certified using $O(1)$ elementary certificates.

If we remove the pathway triangles from a triangulated corridor we are left with a (possibly empty) set of simply connected regions, each of which is the pocket associated with some proper chord (which we call the *extreme chord* for that pocket). As noted earlier, if a pocket is triangulated, then the edges of the dual induce a rooted binary tree on the chords in the pocket. We refer to the tree as the *chord tree* of the pocket. In Figure 3, for example, the extreme chord $e$ is the parent of chords $e_1$ and $e_2$.

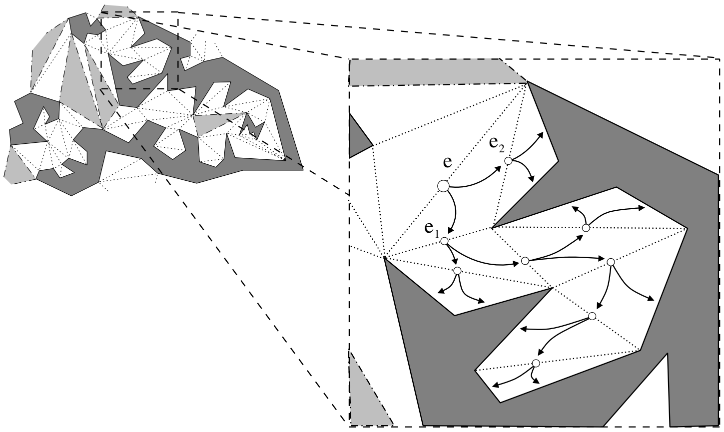


Figure 3: The chord tree rooted at extreme chord $e$.

In general we will find it more efficient to maintain pathways and pockets as unions of pseudo-triangles, as described in the next subsections. We do so without giving up any of the structural properties inherited from the dual of a full triangulation. In particular, the pseudo-triangles of a pathway will retain a linear structure and those of a pocket a rooted tree structure.

### 3.1. Pseudo-triangulations of pockets

As noted earlier any triangulation of a pocket induces a tree structure on the associated chords. The same is true of any pseudo-triangulation of a pocket with the following recursive structure: the extreme chord of the pocket forms one of the side chains of a pseudo-triangle (the *top* pseudo-triangle) and every proper chord on each of the other two side chains is the extreme chord of its own pseudo-triangulated pocket.

Note that pocket pseudo-triangles involve chords from one object only. As a consequence, the vitality of such pseudo-triangles never changes (i.e. requires no certificates).

We maintain a pseudo-triangulation (as opposed to a full triangulation) on pockets for consistency with the structure of adjacent pathways. Providing the flexibility of pseudo-triangles permits more efficient structural updates as objects may enter and subsequently vacate pockets (see section 4).

### 3.2. Pseudo-triangulations of pathways

Suppose we have a corridor $C$ with boundary chains $\gamma$ and $\delta$. Let $P$ be the object associated with $\gamma$ and let $Q$ be the object associated with $\delta$. (Recall that $P$ and $Q$ may denote the same object.) Suppose we are given a pathway in $C$ together with its defining triangulation (which of course is a legitimate pseudo-triangulation). We can produce a minimal associated pseudo-triangulation by successive removal of bridges whenever the two incident pseudo-triangles combine to form a single pseudo-triangle. Since the feasibility of this merge operation depends only on the edges incident to a given bridge, it can be checked in $O(1)$ time. Hence, we hereafter assume that all pathway pseudo-triangulations are minimal. Further coarsening follows from a simple classification of pathway pseudo-triangles.

Given that each pseudo-triangle contains exactly two bridges, these can be contained in either one or two sides of the triangle. We can therefore distinguish two types of pseudo-triangles (see Figure 4):

- *corner pseudo-triangles in $P$* that have bridges on only one side chain, and chords of only one object, $P$, on the other two side chains.
- *$P$ pseudo-triangles* (or *object pseudo-triangles in $P$*) that contain bridges on two side chains and chords of only one object, $P$, on the third.

($Q$ pseudo-triangles and corner pseudo-triangles in $Q$ are defined analogously.)

It is straightforward to check (using convexity of side chains) that two corner pseudo-triangles in the same object $P$ can never be adjacent; however, a corner in $P$ can be adjacent to a corner in a different object. Furthermore, the bridges of any $P$ pseudo-triangle must be incident on the two pseudo-triangle corners that delimit the one bridgeless side chain (this can be easily achieved while creating the minimal

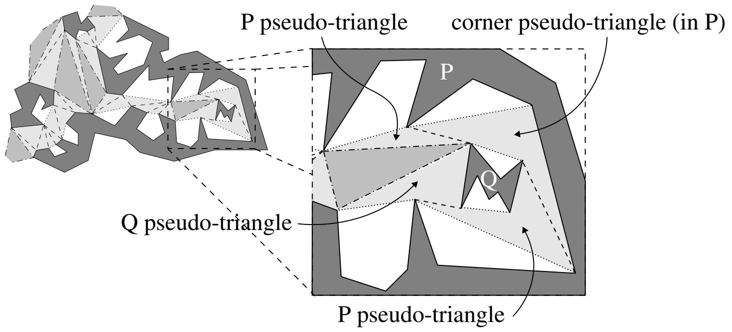


Figure 4: The different types of pseudo-triangles in a pathway.

pathway pseudo-triangulation). It follows that at least one of the other side chains consists of a bridge only.

**Observation 7** *Given two adjacent $P$ pseudo-triangles, one can either merge them into one $P$ pseudo-triangle or transform them into an $Q$ pseudo-triangle and a corner pseudo-triangle in $P$. See Figure 5. In either case the number of $P$ pseudo-triangles in the pathway decreases by one.*

The same observation holds with $P$ and $Q$ interchanged.

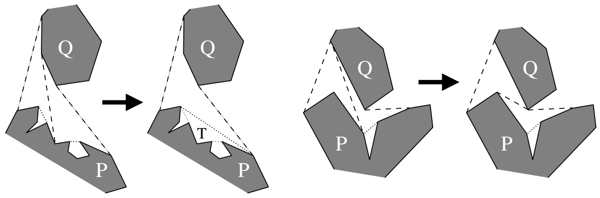


Figure 5: Merging and transforming two $P$ pseudo-triangles.

Note that merging two pseudo-triangles might involve the introduction of a new chord that cuts off a new pocket pseudo-triangle from the existing pathway. Such a pseudo-triangle $T$ becomes the top node of a chord tree that joins the chord trees rooted at the extreme chords belonging to the cut off side chains.

**Observation 8** *Given an $P$ pseudo-triangle adjacent to a $Q$ pseudo-triangle one can swap the diagonal between them to interchange their types (i.e. if they have the form $PQ$ the diagonal swap will yield $QP$).*

Note that a diagonal swap between two pseudo-triangles might also result in two corner pseudo-triangles or in an empty triangle that we can remove from the pseudo-triangulation.

Applying observations 7 and 8 we can now establish the following invariant:

**Invariant 9 (Pathway invariant)** *Any three consecutive pseudo-triangles in a pathway must include a corner pseudo-triangle. Furthermore, any two adjacent object pseudo-triangles must belong to different objects.*

Given any triangulation of a pathway we can initially establish the invariant by simply merging and transforming adjacent object triangles (see Figure 6). See Figure 7 for a fully merged and transformed version of the pathways of our example. Note that the pathway of a corridor between two polygons whose convex hulls are disjoint will contain at most two pseudo-triangles.

With the help of the pathway invariant we can bound the number of elementary certificates in any corridor certificate:

**Lemma 10** *To establish the vitality of a corridor $C$ bounded by polygonal chains $\pi_P$ and $\pi_Q$ requires $\Theta(\kappa)$ elementary certificates, where $\kappa$ is the size of a minimum link separator of $\pi_P$ and $\pi_Q$ joining the end-edges of $C$.*

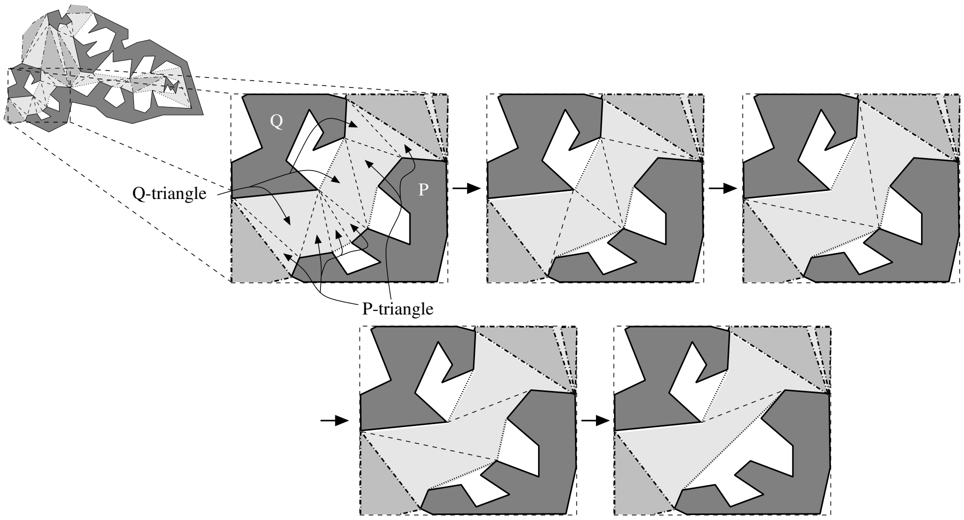


Figure 6: Merging and transforming adjacent object triangles yields a pathway consisting of only two pseudo-triangles if the convex hulls of the defining polygonal chains are disjoint.

**Proof:** Any pathway in $C$ (satisfying the pathway invariant) has at most five times as many pathway pseudo-triangles as there are corner pseudo-triangles (this can be the case if a corridor contains only one corner pseudo-triangle and two $P$ and $Q$ pseudo-triangles each). Furthermore, as noted earlier, the vitality of every pathway pseudo-triangle can be certified with at most four elementary certificates. Hence, the vitality of $C$ can be certified by a set of elementary certificates of size proportional to the number of corner pseudo-triangles in its pathway.

There are at least as many segments in a minimum link separator of polygonal chains $\pi_P$ and $\pi_Q$ as there are corner pseudo-triangles in any pathway in $C$. Furthermore, since pseudo-triangles are star-shaped one can construct a polygonal path separating $\pi_P$ and $\pi_Q$ by connecting the midpoints of the bridge edges with at most two segments per pseudo-triangle. Thus, any minimum link separator for $\pi_P$ and $\pi_Q$ contains between one and six times as many segments as there are corner pseudo-triangles inside the pathway of the corridor. $\square$

**Corollary 11** *The total number of elementary certificates needed to establish the vitality of all partition triangles and corridors (and hence the disjointness of all objects) is* $O(k + \sum_P \kappa_P)$*, where* $\kappa_P$ *is the length of a minimum link cycle around* $P$*.*

Note that the cost of initially creating a corridor structure with associated pathways satisfying the pathway invariant is dominated by the cost of triangulating the free space between the objects (cf. Theorem 3).

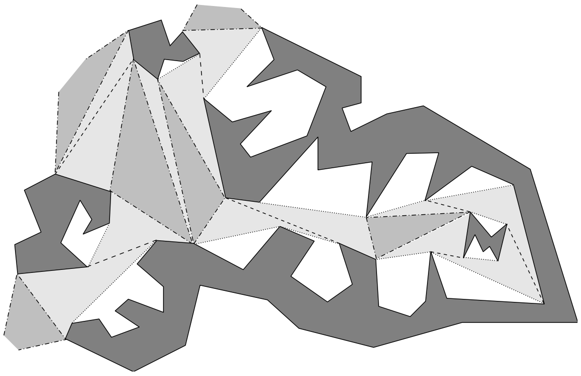

Figure 7: Pathway pseudo-triangulations satisfying the pathway invariant.

## 4. Maintenance

We now assume that we are given $k$ polygonal objects moving with continuous rigid motion and that we are able to compute the failure time of any elementary certificate associated with those objects in constant time.

When any certificate fails, we are able to update the data structure and restore the invariants by computation on a constant number of pseudo-triangles. If the convex side chains of pseudo-triangles are maintained in appropriate data structures, then this updating can be performed in $O(\log m)$ time for pseudo-triangles with $m$ boundary segments. (Note that these bounds only apply to the update of the data structure, see section 5 for an analysis that includes the update of the event queue.)

To maintain the decomposition of the free space $\mathcal{E}$ between the polygons into partition triangles and corridors we insert all partition triangle and corridor vitality certificates into an event queue sorted by time of failure. Since the structure of the partition triangles and the corridors does not change unless either a partition triangle or a corridor certificate fails, we only need to describe how to update the structure in the case of such an event. The update will always involve the descheduling and rescheduling in the event queue of the certificates affected.

Maintaining the pseudo-triangulation inside the pathway of a corridor during the motion of the objects will involve operations on the convex chains of chords that are part of the pseudo-triangle side chains. To do that efficiently we assume that these convex chains are stored in a data structure that allows us to find common tangents between two chains in $O(\log m)$ where $m$ is the number of vertices currently part of the chains. One possible structure that achieves these bounds is the *path hull data structure* as described in [10].

Processing events and updating the pathways will sometimes involve the relocation of partition triangles and the insertion and deletion of pseudo-triangles in pathways and pockets. If we insert an additional object pseudo-triangle or delete a corner pseudo-triangle the pathway invariant might be violated, necessitating a restructuring of the pathway.

**Lemma 12** *Given a pathway that satisfies the pathway invariant, one can perform a local update (pseudo-triangle insertion or deletion) and re-establish the pathway invariant in time $O(\log m)$, where $m$ is the number of vertices of the affected pseudo-triangles and their (at most four) immediate neighbors. This involves the destruction and creation of a constant number of edges and elementary certificates.*

**Proof:** Assume that after the insertion of a new pseudo-triangle there are three consecutive object pseudo-triangles (since the corridor satisfied the pathway invariant before there will be at most three). They will either have the form $PQP$ or $PPQ$ (up to symmetry). A diagonal swap between the $Q$ and the $P$ pseudo-triangle will transform the $PQP$ case into $PPQ$. The two $P$ pseudo-triangles can either be merged into one $P$ pseudo-triangle or transformed into a $P$ and a corner pseudo-triangle (see Figure 5).

Similarly, after the deletion of a corner pseudo-triangle out of a passage that satisfied the pathway invariant, there can be at most four consecutive object pseudo-triangles that can be merged and transformed as described above.

Swapping a diagonal, merging and transforming two pseudo-triangles takes time $O(\log m)$ where $m$ is the number of vertices of the pseudo-triangles involved. □

### 4.1. Partition triangles

Partition triangle certificates state that a partition triangle has positive area. These certificates expire whenever vertices $PQR$ of a partition triangle (see Figure 8) become collinear (without loss of generality, vertex $R$ intersects edge $PQ$). We distinguish two different kinds of these events: a *move* and a *swap* (see Figure 8).

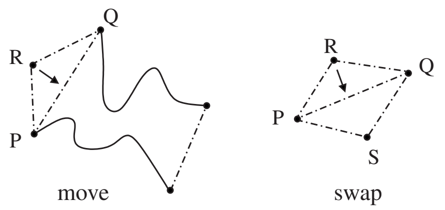


Figure 8: The two types of events for a triangle certificate failure.

A *move* event occurs whenever edge $PQ$ is the end-edge of a non-degenerate corridor. Otherwise, edge $PQ$ is shared by two partition triangles.

In the case of a move event we need to find two new endpoints for the partition triangle inside the $PQ$ corridor. The first pseudo-triangle $T$ inside the corridor can be either an object or a corner pseudo-triangle (see Figure 9 for the case of an $P$ pseudo-triangle).

Let us first assume that $T$ is a $P$ pseudo-triangle (the case for $T$ being a $Q$ pseudo-triangle can be handled analogously). To distinguish the three possible cases it is sufficient to test if $R$ can see the opposite corner of the pseudo-triangle ($P_2$ in Figure 9). If $R$ cannot see $P_2$ then we have to determine which side chain constitutes the obstacle. This can be done in $O(\log m)$ time by finding common tangents where $m$ is the number of vertices of the pseudo-triangle.

If $T$ is a corner pseudo-triangle in $P$ (again the case for $T$ being a corner pseudo-triangle in $Q$ is symmetric) then $Q$ can see the complete side chain that is part of $P$ and starts at $P_1$. Therefore we need only to find the vertex on this side chain that can be seen by $R$ as well. Again this can be done in $O(\log m)$ time by finding common tangents.

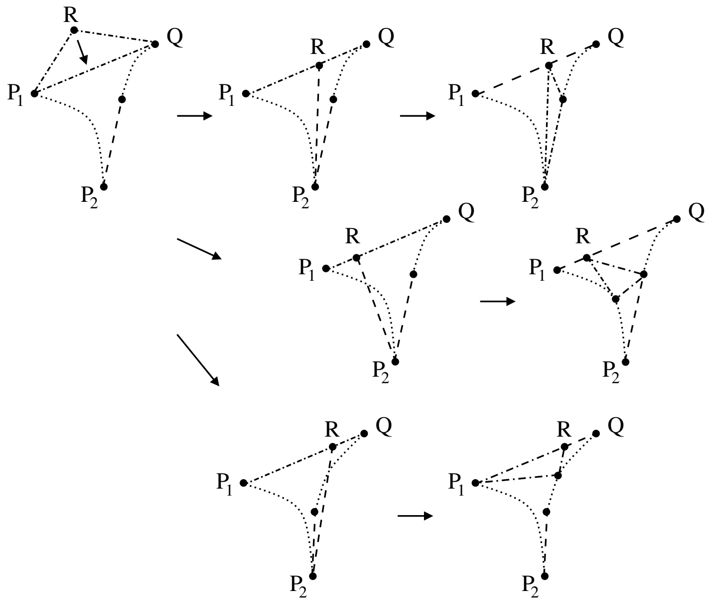


Figure 9: Examples of the different cases occurring during a move event.

The new partition triangle cuts the old object or corner pseudo-triangle into up to four pseudo-triangles (see Figure 9 for examples), two of which might need to be added to the $RP$ and $RQ$ corridors. After the insertion of those triangles into their respective corridors, the pathway invariants might have to be restored.

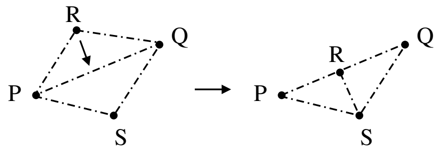


Figure 10: Swapping two partition triangles.

In the case of a swap event we swap the diagonal between the two partition triangles and update the two associated triangle certificates.

### 4.2. Pathways

While pathways may be altered when partition triangle certificates expire (as we have seen) they also change due to strictly internal events associated with the expiry of pathway pseudo-triangle vitality certificates.

As described above a pathway pseudo-triangle's vitality is certified by two types of certificates – corner and side certificates. Thus, there are two types of events we have to consider.

#### 4.2.1. Corner certificate failure

A pseudo-triangle corner certificate fails if its two incident edges collide. We say that the shorter edge collides with the longer one (see Figure 11). Should both edges have the same length, a collision between the two objects occurs.

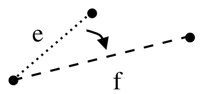


Figure 11: The chord $e$ collides with the bridge $f$.

Since we have identified four different kinds of edges, at most sixteen different collision types are possible. A careful analysis shows that only four types need to be distinguished.

Chords never collide with other chords because whenever they share an endpoint they belong to the same object and the objects move rigidly. Bridges can collide with other bridges only if they collide at the same time with a chord so it is not necessary to count those kind of collisions separately. Therefore the following four types of events can occur:

**Boundary event** Bridge $e$ collides with a polygon boundary edge $f$ – because the bridge is shorter, the endpoint of $e$ that is not shared with $f$ must hit edge $f$; we report this collision.

**Chord event** Bridge $e$ collides with a proper chord $f$ – we create a new corner pseudo-triangle (see Figure 12).

**Bridge event** A chord $e$ collides with a proper bridge $f$ – depending on the kind of pseudo-triangle $f$ is part of, we either update that pseudo-triangle or create a new pseudo-triangle (see Figures 13 - 16).

**End event** A chord $e$ collides with an end-edge $f$ – we transfer $e$ into the next corridor (see Figure 17).

The next paragraphs describe in detail how to update the data structure when a chord, bridge, or end event occurs. Whatever update is necessary to handle a boundary event (and the associated collision) highly depends on the particular application that uses the KDS. We will therefore exclude this case from our discussion.

### Chord event

Whenever one or two bridges collide with a proper chord, the chord gets *broken* and replaced by its two children chains in the chord tree (see Figure 12). The end-edges or bridges keep their type, i.e. each of the bridges in Figure 12 could be an end-edge and would remain an end-edge after the collision.

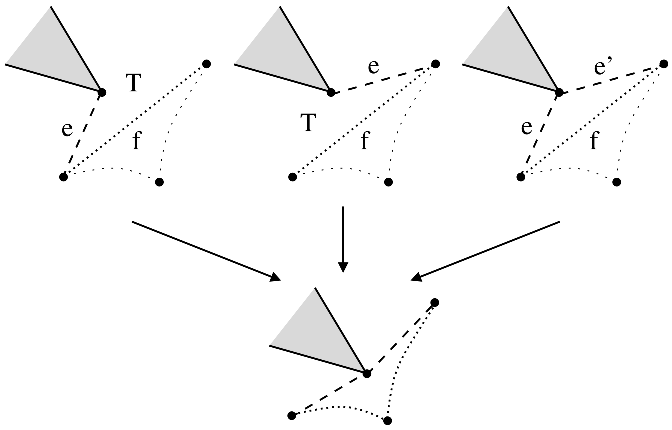


Figure 12: Chord event: An end-edge/bridge $e$ (and $e'$) colliding with chord $f$.

If the triangle $T$ is a corner pseudo-triangle then the collision turns it into an object pseudo-triangle. In this case it might be necessary to restore the pathway invariant.

### Bridge event

Two edges can collide only if the share a common endpoint. That implies that a chord can only collide with a proper bridge at a corner of a pseudo-triangle. Any bridge endpoint is a corner of at least one pseudo-triangle. Thus, up to symmetry, we can assume that the collision takes place at some corner $q$ belonging to object $Q$, and the chord $e$ colliding with bridge $f$ (drawn as a proper chord in the figures) lies clockwise of $f$ around $q$. We have three subcases to consider: the pseudo-triangle $T$ counterclockwise of $f$ is either $(i)$ a *pure* $P$ pseudo-triangle whose bridge sides contain only bridges, $(ii)$ an *impure* $P$ pseudo-triangle containing at least one chord of $Q$, or $(iii)$ a pure or impure $Q$ pseudo-triangle or a corner pseudo-triangle in $Q$.

Note that $T$ can not be a corner pseudo-triangle in $P$, since all endpoints of bridges in such a pseudo-triangle that are also corners of the pseudo-triangle are vertices of $P$ and we assume that the collision takes place at a vertex of $Q$. For the

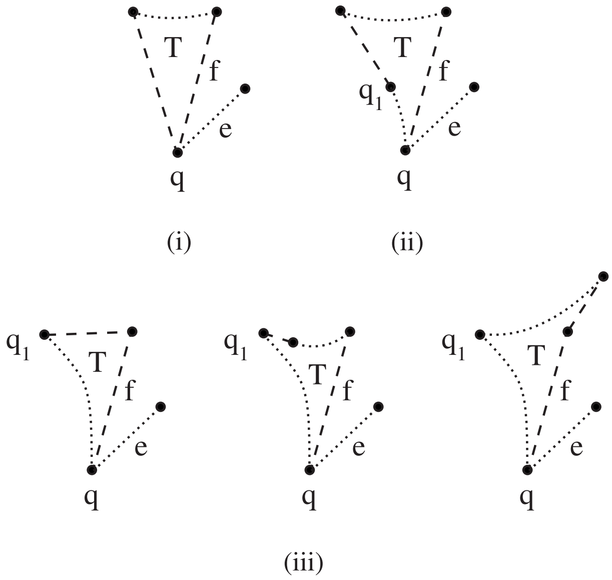


Figure 13: Bridge event: The different
types of pseudo-triangles a bridge $f$ can be part of.

same reason the pseudo-triangle $T'$ on the right of $f$ can not be a corner pseudo-triangle in $P$ either.

**Bridge event (i) :** $T$ is a pure $P$ pseudo-triangle. Its only possible neighbors, due to the pathway invariant, are a $Q$ pseudo-triangle or a corner pseudo-triangle in $Q$ (see Figure 14). After the collision the chord $e$ becomes part of $T$ and the bridge $f$ starts now at $q_1$. These operations take constant time.

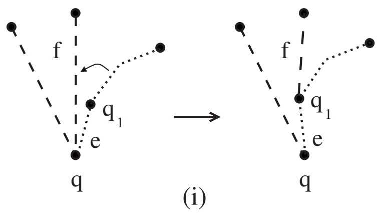


Figure 14: Bridge event $(i)$

If $e$ was the only edge on its side of pseudo-triangle $T'$, then $T'$ will cease to exist. This however can only be the case if $T'$ was an object pseudo-triangle, therefore the pathway invariant still holds.

**Bridge event (ii) :** $T$ is an impure $P$ pseudo-triangle (see Figure 15) and the colliding chord $e$ therefore has to be integrated with the edges of $Q$ that are part of $T$. There are two possibilities: the tangent from $q_3$ onto the convex chain of $Q$ edges between $q_1$ and $q_2$ either does or does not form a convex chain with the bridge starting at $q_2$. In the former case this tangent becomes part of the pseudo-triangle $T$, in the latter an additional triangle has to be created (see Figure 15) and the pathway invariant has to be restored. Both actions can take time $O(\log m)$ where $m$ is the number of vertices of the triangles involved (at most three, see Lemma 12).

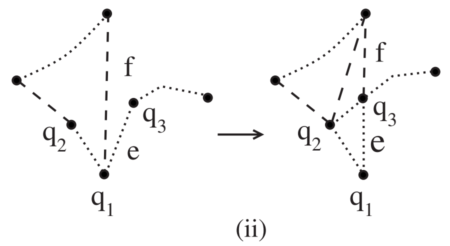


Figure 15: Bridge event (*ii*)

**Bridge event (iii) :** $T$ is a $Q$ pseudo-triangle or a corner pseudo-triangle in $Q$ (see Figure 16) and therefore contains a convex chain of chords of $Q$. Those edges have to be integrated with the colliding edge $e$. Again there are two possibilities if $T$ is a $Q$ pseudo-triangle depending on how the tangent from $q_3$ to the chain of edges between $q_1$ and $q_2$ intersects $T$. If the tangent lies inside the pseudo-triangle then this tangent becomes part of the bounding chain, and $Q$ chords that get cut off from the pathway by the tangent become part of the corresponding chord tree. If, on the other hand, the tangent intersects with opposite side of the pseudo-triangle, then a new corner pseudo-triangle has to be created (see Figure 16), which does not disturb the pathway invariant. Either action can be performed in $O(\log m)$ time.

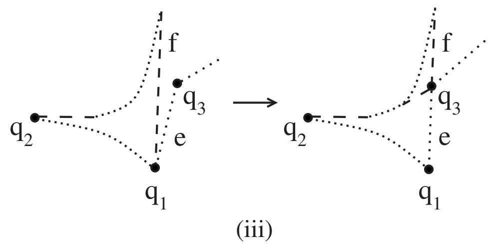


Figure 16: Bridge event (*iii*)

**End event**

A chord $e$ can collide only with an end-edge $f$ at a common endpoint $q$. We can therefore move the vertex of the partition triangle from $q$ to the other endpoint $q_1$ of $e$ and push the triangle $\Delta q_1 pq$ into the adjacent corridor (see Figure 17). Restoring the pathway invariant takes again time $O(\log m)$ where $m$ is the number of vertices of the (at most three) pseudo-triangles involved (see Lemma 12).

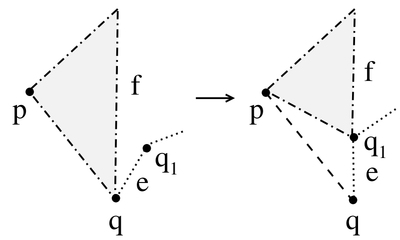


Figure 17: End event: The chord $e$ collides with the end-edge $f$

### 4.2.2. Side certificate failure

A side certificate fails whenever two adjacent edges on a side chain of some pseudo-triangle become colinear. When all objects move rigidly the angles between polygon edges or chords never change. Therefore the following three events have to be considered:

**Chord event** Two adjacent bridges – we create a new extreme chord $e$ and restore the pathway invariant (see Figure 18)

**Bridge event** A proper bridge $f$ adjacent to a chord $e$ – we move $e$ into the neighboring pseudo-triangle (see Figure 19)

**End event** An end-edge $f$ adjacent to a chord $e$ – we create a new object pseudo-triangle and restore the pathway invariant (see Figure 20)

The following paragraphs describe these events in detail.

#### Chord event

A side certificate between two bridges can fail only if those bridges are part of a corner pseudo-triangle. We will introduce a new extreme chord $e$ that connects the two endpoints the bridges do not have in common. This extreme chord becomes the parent of the two convex chains $e_1$ and $e_2$ that used to form the other two sides of the corner triangle (see Figure 18). Finally we create a new object pseudo-triangle $T$ and restore the pathway invariant. Since a corner pseudo-triangle is removed from the pathway and a new object pseudo-triangle is inserted, this might involve operations on up to five pseudo-triangles and will take time $O(\log m)$ where $m$ is the number of vertices of the triangles involved.

#### Bridge event

A side certificate between two edges can only fail if they share a common endpoint on the same side of a pseudo-triangle. That implies that a proper bridge $f$ and a chord

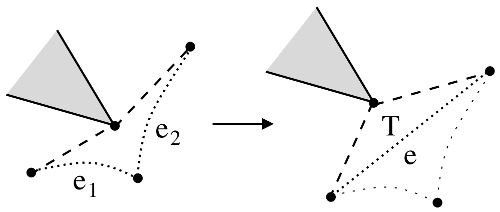


Figure 18: Chord event: Two bridges straightening.

$e$ can straighten only at a vertex $q$ that is not a corner of a pseudo-triangle. That leaves the following case to be considered (up to symmetry): The side certificate fails at the vertex $q$, part of the polygon $Q$, therefore the pseudo-triangle $T$ that contains $e$ and $f$ (on the left of $f$ in Figure 19) can only be a $P$ pseudo-triangle or a corner pseudo-triangle in $P$. Every endpoint of a proper bridge has to be a corner of a pseudo-triangle in at least one of the pseudo-triangles the bridge is part of. Since $q$ is not a corner of $T$ it has to be a corner of $T'$, $T$'s neighbor across $f$. Therefore we can simply move $e$ into $T'$ without changing the type of any pseudo-triangle involved (see Figure 19).

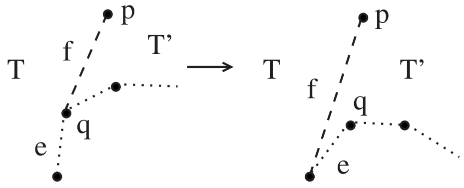


Figure 19: Bridge event: The side certificate associated with $e$ and $f$ fails.

**End event**

Since an end-edge $f$ belongs to only one pseudo-triangle we can insert a new object pseudo-triangle into the pathway as depicted in Figure 20. This might require the restoration of the pathway invariant.

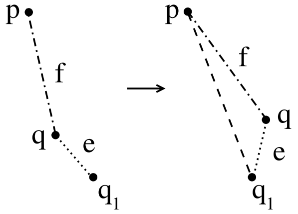


Figure 20: End event: An end-edge and a chord/polygon edge straighten.

## 5. Kinetic data structure properties

Recall that we have $k$ polygonal objects consisting of a total of $N$ vertices. Our data structure has the following desirable properties for a KDS.

### 5.1. Compactness

The size of a KDS as defined in [12] is the maximum number of certificates ever present in the active set. For our data structure that translates to the maximum number of triangle and elementary certificates for all polygons.

There are $\Theta(k)$ partition triangles (see Theorem 3) each of which will contribute a certificate to the active set. Each polygon $P$ contributes $\Theta(\kappa_P)$ elementary certificates, where $\kappa_P$ is the size of a minimum link cycle surrounding polygon $P$ (see Lemma 10 and section 3.2). Therefore the active set contains $\Theta(\sum_P \kappa_P)$ certificates at any time.

Note that if the objects are well separated (for example, if the convex hulls of each pair of objects are disjoint) then the number of certificates in the active set is $\Theta(k)$. However if the objects are closely intertwined, then the number of certificates in the active set can be $\Theta(N)$ in the worst case.

### 5.2. Locality

Observe that each polygon is surrounded by a cycle of pathways connected by partition triangles. Due to Lemma 10 we can therefore state that the number of certificates for any polygon $P$ is $O(k + \kappa_P)$.

In the worst case $\kappa_P$ could be $\Theta(N)$. However, it is hard to imagine that the disjointness of P from the other objects could be certified with fewer than $\kappa_P$ elementary certificates.

### 5.3. Responsiveness

A KDS is considered to be responsive as defined in [12] if the worst-case amount of time required to update it after a certificate failure is *small*.

Our data structure uses two kinds of certificates: Triangle certificates and corridor (i.e. elementary) certificates. The update of both kinds of certificates can be done in $O(\log m)$ as described in sections 4.1 and 4.2 where $m$ is the number of boundary segments of the (at most five) pseudo-triangles affected. In all cases the update involves the destruction and creation of a constant number of edges and associated corner and side certificates. Each elementary and triangle certificate that is disturbed during this process needs to be descheduled or rescheduled in the event queue which takes additional time logarithmic in the size of the active set (see section 5.1).

### 5.4. Efficiency

A key performance measure for kinetic data structures is the number of events processed in the worst case. A KDS is said to be *efficient* (see [12]) if the number of events processed by the structure in the worst case is asymptotically of the same order as the number of *external events*, i.e. the number of changes to the discrete attribute one wishes to maintain. As already pointed out by Agarwal et al. in [1] the notion of efficiency in the case of collision detecting data structures is not clear, since there is no canonical discrete attribute against which to compare the performance of the KDS, i.e. it is not canonically defined what constitutes an external event.

However it is somewhat natural to assume that any event that changes the spatial relation of the objects is considered an external event. We would like to argue that any change to the number of segments of the minimum link separator inside a passage constitutes an external event. Unfortunately there is no direct relation between the complexity of the minimum link separator for a specific passage and the number of events processed (see Figure 21 for an example where the number of segments in the minimum link separator remains constant while a number of events linear in the number of polygon vertices is processed).

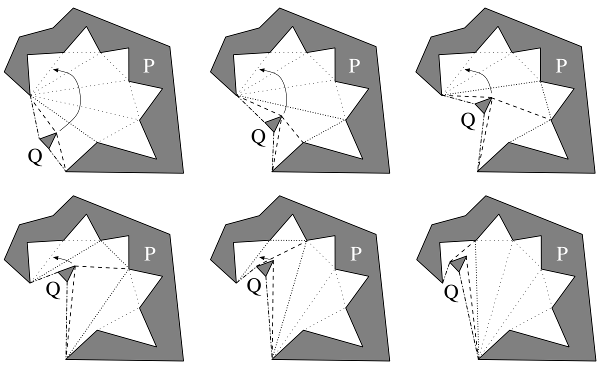


Figure 21: $Q$ translates along a convex path inside $P$.

Nevertheless it is possible to give an easy upper bound for the number of certificate failures if the polygons translate along algebraic trajectories of bounded degree.

**Theorem 13** *If $k$ simple polygons with a total of $N$ vertices translate along algebraic trajectories of degree $d$ then the number of events processed by the KDS is $O(dN^3)$.*

**Proof:** It suffices to observe that events happen, i.e. certificates fail, only when three vertices are collinear. □

Note that the partition triangles do not correspond to vertices of the compact Voronoi diagram as defined in [19]. Therefore we do not have to process events whenever four vertices are cocircular.

## 6. Future work

We would like to exploit the hysteresis exhibited by the partition triangles and the pseudo-triangles inside the pathways to obtain tighter bounds on the number of changes to the structure during various kinds of motions. Our structure tends to maximize the number of polygon edges that are part of corridors in which both boundary chains are on the same object and therefore do not have certificates associated with them. This fact has not been incorporated in the bounds for the compactness of the structure yet. The structure will be implemented and we will compare the experimental results with more traditional collision detection methods.

Since our initial goal included uniform structures suitable for dynamic data in Geographic Information Systems, there are several desirable extensions. First, there are likely to be many stationary objects; we need not maintain certificates that involve only stationary objects. Second, moving objects could be given approximate positions that are refined by closer observation or communication. Third, we would like to apply this work to flexible or articulated objects. The current structure benefits from the rigidity of objects by not maintaining certificates for the pseudo-triangles or triangles in the chord trees or in the corridors in which both boundary chains are on the same object. If the objects were subdivided so that parts that might be in relative motion were identified as corresponding to different objects (creating more partition triangles), or if the flexibility was limited to a range that does not disturb the data structure beyond what can be repaired responsively and locally (e.g., that pseudo-triangles in the chord tree would remain valid), or some combination of the two, then we expect that the structure could be extended. Reasonable conditions on the flexibility and experimental or practical analysis are interesting avenues to pursue.

## References

1. P. K. Agarwal, J. Basch, M. de Berg, L. J. Guibas, and J. Hershberger. Lower bounds for kinetic planar subdivisions. In *Proc. 15th ACM Sympos. Comp. Geom.*, pages 247–254, 1999.
2. P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tilings for kinetic collision detection. In *Proc. 5th Workshop Algorithmic Found. Robotics*, 2000.
3. J. Basch, J. Erickson, L. J. Guibas, J. Hershberger, and L. Zhang. Kinetic collision detection for two simple polygons. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 102–111, 1999.
4. J. Basch, L. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756, 1997.
5. J. Basch, L. Guibas, C. Silverstein, and L. Zhang. A practical evaluation of kinetic data structures. In *Proc. 13th ACM Sympos. Comp. Geom.*, pages 388–390, 1997.
6. J. Basch, L. J. Guibas, and L. Zhang. Proximity problems on moving points. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 344–351, 1997.
7. B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir,

and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12:54–68, 1994.

8. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.
9. J. Erickson, L. Guibas, J. Stolfi, and L. Zhang. Separation-sensitive collision detection for convex objects. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 327–336, 1999.
10. J. Friedman, J. Hershberger, and J. Snoeyink. Efficiently planning compliant motion in the plane. *SIAM J. Comput.*, 25:562–599, 1996.
11. L. Guibas, J. Snoeyink, and L. Zhang. Compact Voronoi diagrams for moving convex polygons. In *Proc. 7th Scandinavian Workshop on Algorithm Theory*, volume 1851 of *Lecture Notes Comp. Sci.*, pages 339–352. Springer Verlag, 2000.
12. L. J. Guibas. Kinetic data structures: A state of the art report. In *Proc. 3rd Workshop Algorithmic Found. Robotics*, pages 191–209, 1998.
13. L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, and J. S. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Internat. J. Comput. Geom. Appl.*, 3(4):383–415, Dec. 1993.
14. P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, Sept. 1995.
15. S. Kapoor and S. N. Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 172–182, 1988.
16. S. Kapoor, S. N. Maheshwari, and J. S. B. Mitchell. An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete Comput. Geom.*, 18:377–383, 1997.
17. D. Kirkpatrick and B. Speckmann. Separation sensitive kinetic collision detection for simple polygons. In *Proc. Japan Conference on Discrete and Computational Geometry*, Lecture Notes Comp. Sci. Springer Verlag, 2000. To appear.
18. T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, 1979.
19. M. McAllister, D. Kirkpatrick, and J. Snoeyink. A compact piecewise-linear Voronoi diagram for convex sites in the plane. *Discrete Comp. Geom.*, 15:73–105, 1996.
20. K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1993.
21. C. Ó'Dúnlaing and C. K. Yap. A "retraction" method for planning the motion of a disk. *J. Algorithms*, 6:104–111, 1985.
22. F. P. Preparata. Planar point location revisited. *Internat. J. Found. Comput. Sci.*, 1(1):71–86, 1990.
23. T. Roos. *Dynamic Voronoi diagrams.* Ph.D. thesis, Bayerische Julius-Maximilians-Univ., Würzburg, Germany, Sept. 1991.
24. M. Sharir and S. Sifrony. Coordinated motion planning for two independent robots. *Ann. Math. Artif. Intell.*, 3:107–130, 1991.
25. S. Suri. *Minimum link paths in polygons and related problems.* PhD thesis, Dept. of Computer Science, Johns Hopkins Univ., 1987.