Deformable Free Space Tilings for Kinetic Collision Detection

Pankaj K. Agarwal* Julien Basch† Leonidas J. Guibas† John Hershberger‡
Li Zhang†

Abstract

We present kinetic data structures for detecting collisions between a set of polygons that are not only moving continuously but whose shapes can also vary continuously with time. Unlike classical collision detection methods that rely on bounding volume hierarchies, our method is based on deformable tilings of the free space surrounding the polygons. The basic shape of our tiles is that of a *pseudo-triangle*, a shape sufficiently flexible to allow extensive deformation, yet structured enough to make detection of self-collisions easy. We show different schemes for maintaining pseudo-triangulations as a kinetic data structure, and we analyze their performance. Specifically, we first describe an algorithm for maintaining a pseudo-triangulation of a point set, and show that the pseudo-triangulation changes only quadratically many times if points move along algebraic arcs of constant degree. We then describe an algorithm for maintaining a pseudo-triangulation of a set of convex polygons. Finally, we extend our algorithm to the general case of maintaining a pseudo-triangulation of a set of moving or deforming simple polygons.

1 Introduction

Collision detection between moving objects is a fundamental problem in computational simulations of the physical world. Because of its universality, it has been studied by several different communities, including robotics, computer graphics, computer-aided design, and computational geometry. Several methods have been developed for the case of rigid bodies moving freely in two and three dimensions. Though a physical simulation involves several other computational tasks, such as motion dynamics integration, graphics rendering, and collision response, collision detection remains one of the bottlenecks in such a system. A commonly used approach to expedite the collision detection between complex shapes is based on hierarchies of simple bounding volumes surrounding each of the objects. For a given placement of two non-intersecting objects, their respective hierarchies are refined only to the coarsest level at which the primitive shapes in the two hierarchies can be shown to be pairwise disjoint.

Motion in the physical world is in general continuous over time, and many systems attempt to speed up collision checking by exploiting this temporal coherence, instead of repeating a full collision check *ab initio* at each time step [11]. Swept volumes in space or space-time have been used towards this goal [4, 9]. Though fixed time-sampling is customary for motion integration, collisions tend to be rather irregularly spaced over time. If we know precisely the motion laws of the objects, then it makes sense to try to predict exactly when collisions will happen, instead of hoping to locate them with time sampling. There have been a few theoretical papers in computational geometry

*Center for Geometric Computing, Department of Computer Science, Box 90129, Duke University, Durham, NC 27708, USA.

†Computer Science Department, Stanford University, Stanford, CA 94305, USA.

‡Mentor Graphics Corp., 8005 SW Boeckman Road, Wilsonville, OR 97070, USA.

along these lines [5, 8, 12], but their results are not so useful in practice because they use complex data structures and are only applicable for limited types of motion.

Recently Basch *et al.* [2] and Erickson *et al.* [6] presented algorithms for detecting collision between two polygons using the *kinetic data structures* framework, which was originally introduced by Basch *et al.* [3, 7]. Their algorithms avoid many of the problems that arise in the fixed time-sampling method. A kinetic data structure, or KDS for short, is built on the idea of maintaining a discrete attribute of objects in motion by animating a proof of its correctness through time. The proof consists of a set of elementary conditions, called *certificates*, based on the kinds of tests performed by ordinary geometric algorithms (CCW tests in our case). Those certificates that can fail as a result of the motion of the polygons are placed in an event queue, ordered according to their earliest failure time. When a certificate fails, the proof needs to be updated. Unless a collision has occurred, we perform this update and continue the simulation. In contrast to fixed time step methods, for which the fastest moving object gates the time step for the entire system, a kinetic method is based on *events* (the certificate failures) that have a natural significance in terms of the problem being addressed (collision detection in this case).

Unlike the previous hierarchy-based algorithms, the algorithm by Basch *et al.* [2] maintains a decomposition of the common exterior of the two moving polygons. The cells of this decomposition deform continuously as the objects move. As long as all the cells in the decomposition remain disjoint, the decomposition itself acts as a KDS proof of non-collision between the objects. At certain times, cells become invalid because they self-intersect and the decomposition has to be modified. Unfortunately, in their approach an extension to collision detection between many polygons is very expensive — we have to construct such a decomposition for every pair of polygons.

In this paper we present an algorithm for detecting collision between members of arbitrary sets of simple polygons as they move and/or deform in the plane. As in [2], we maintain a decomposition of the common exterior of polygons, the free space, into deformable tiles. Ideally, we would like to maintain a decomposition so that the self-intersection of a cell of the decomposition is easy to detect, the decomposition is simple to update when a self-intersection occurs, and the decomposition conforms to the motion of polygons so that self-intersections do not happen too many times. An obvious choice for the decomposition is a triangulation of the free space. Although a triangulation satisfies the first two criteria, it contain too many cells and therefore has to be updated frequently. We will therefore use a *pseudo-triangulation* as the decomposition, which has considerably fewer cells compared to a triangulation. Pseudo-triangles can flex as the objects move, and therefore the combinatorial structure of the tilings needs fewer updates. At the same time, the cells in a pseudo-triangulation have sufficiently simple shapes so that their self-intersections are easy to detect and the triangulation is easy to update. Pseudo-triangulations have been used in the past, but primarily for various visibility problems [10]. An additional benefit of our structure is that it can gracefully adapt to object shapes that are themselves flexible. As our polygons move they can also deform and change shape. This additional flexibility impacts our data structure primarily on the number of events it has to process — but its basic nature and certification remains unchanged.

In Section 2, we describe our model for motion, define pseudo-triangulations, and describe the certificates needed to maintain a pseudo-triangulation. In Section 3, we first describe how to maintain a pseudo-triangulation for a set $\mathcal{P}$ of n moving points in the plane, and show that it can be maintained in an output-sensitive manner. For low-degree algebraic motion, the pseudo-triangulation changes about n^2 times. We can further refine the pseudo-triangulation to maintain a triangulation of $\mathcal{P}$, which changes $O(n^{7/3})$ times if each point in $\mathcal{P}$ is moving with fixed velocity. To our knowledge, this is the first triangulation (without Steiner points) that changes a sub-cubic number of times.

In Section 4, we describe a scheme for maintaining the pseudo-triangulation of a set $\mathcal{P}$ of k

disjoint polygons with a total of n vertices. We show that the greedy vertical pseudo-triangulation proposed by Pocchiola and Vegter [10] can be maintained efficiently. A nice feature of this (or any minimal) pseudo-triangulation is that the number of cells is only $O(k)$, which is typically much smaller than n, the total complexity of the polygons. On the other hand, the size of any triangulation has to be at least $\Omega(n)$. We will show that we need a kinetic data structure of size $O(k)$ to maintain this pseudo-triangulation. But unlike the set of points case, we do not have sharp bounds on the number of events.

Finally, in Section 5, we combine our algorithm for the convex case with the one in [2] to construct a pseudo-triangulation for a set of pairwise-disjoint simple polygons moving in the plane. It can easily be updated when a certificate fails. The number of certificates needed to maintain the correctness of the pseudo-triangulation is proportional to the size of a *minimum link subdivision* separating the polygons [13], and is thus optimal in some sense. Our separation proof automatically adapts to the complexity of the relative placement of the polygons, and its size will vary between $O(k)$ (when the polygons are far from each other) and $O(n)$ (when they are closely intertwined). A compact separation proof is important when objects are allowed to change their *motion plan* unpredictably.

Traditionally collision detection has been divided into two phases: the *broad phase*, in which one uses a simple-minded algorithm (typically a bounding box check) to determine which pairs of objects might collide and thus need further testing, and the *narrow phase*, in which these candidate pairs get a more detailed collision test using a sophisticated algorithm. Note that our approach based on deformable free space tilings completely obviates this distinction. The tiling effectively 'hides' features of objects that are far away and treats the objects as equivalent to their convex hulls. As the objects get closer and more intertwined, their features get progressively revealed and participate in collision checks. Furthermore our framework can be extended in a straightforward way to deformable objects, a setting that no previous collision detection method had successfully addressed. Though this paper describes how to implement the deformable tiling idea only in 2D, we are hopeful that 3D extensions will also be possible.

2 Pseudo-Triangulations and their Certification

In this section we define pseudo-triangulations and discuss how we maintain them as objects move or deform continuously. Due to lack of space, we do not elaborate on the kinds of motions and deformations allowed. The two key properties that we need for our methods to work are that

- for each certificate or condition in the kinetic proof, we must be able to predict when it will fail next; we assume that our knowledge of the polygon motions allows us to compute the time of such an event at $O(1)$ cost.
- for our analyses we need the property that the polygon motions are such that any particular certificate can fail at most a constant number of times; motions satisfying this constraint are called *pseudo-algebraic*.

2.1 Pseudo-triangles

A *pseudo-triangle* in the plane is a simple polygon whose boundary consists of three concave chains, called *side chains*, that join at their endpoints. The three endpoints of a pseudo-triangle are called *corners*. The edges incident upon the corners are called *corner edges* (Figure 1 (i)). For a vertex p on the boundary of a pseudo triangle, we denote p_L, p_R be the predecessor and the successor of

Figure 1. (i) The examples of some pseudo-triangles and non-pseudo-triangles. The corner vertices are shown in the pseudo-triangles. (ii) An example of pseudo triangulation of a polygon with holes. The dotted edges are the diagonal edges.

p along $\partial\Delta$ in the counterclockwise direction. Let S be a polygon with holes; some of the holes may be degenerate, i.e., they can be points or segments. A *pseudo-triangulation* $\mathcal{T}(S)$ of S is a planar subdivision of the closure of S, so that each face of $\mathcal{T}(S)$ is a pseudo-triangle and so that the interior of each face lies in the interior of S; see Figure 1 (ii) for an example. In other words, $\mathcal{T}(S)$ is a collection of pseudo-triangles with pairwise-disjoint interiors, each lying inside S, that cover S. The vertices of $\mathcal{T}(S)$ are the same as the vertices of S, and each edge of $\mathcal{T}(S)$ is either an edge of ∂S or a segment whose interior lies inside S; edges of the latter type are called *diagonals*.

We are interested in maintaining the pseudo-triangulation of $\mathcal{F}(\mathcal{P})$, which, for brevity, we denote by $\mathcal{T}(\mathcal{P})$.

Since a polygon (even with degenerate holes) can always be triangulated and a triangulated planar subdivision is obviously a pseudo-triangulation, a pseudo-triangulation of S always exists. A pseudo-triangulation $\mathcal{T}$ is called *minimal* if the union of any two faces in $\mathcal{T}$ is not a pseudo-triangle. If $\mathcal{T}$ is not minimal, then there exists a diagonal d in $\mathcal{T}$ so that $\mathcal{T}$ remains a pseudo-triangulation after the removal of d from it. Although minimal pseudo-triangulations are not unique, we can prove the following result on the size of minimal pseudo-triangulations of $\mathcal{F}(\mathcal{P})$.

Lemma 2.1. *Let $\mathcal{P}$ be a set of k simple pairwise-disjoint polygons as defined above. Suppose m of these polygons are not points, and suppose there are a total of r reflex vertices[1] in $\mathcal{P}$. Then there are $k + m + r - 2$ pseudo-triangles in any minimal pseudo-triangulation of $\mathcal{F}(\mathcal{P})$.*

2.2 Maintaining a pseudo-triangulation

We are interested in maintaining $\mathcal{T}(\mathcal{P})$, as $\mathcal{F}(\mathcal{P})$ deforms continuously. Specifically, we want to maintain $\mathcal{T}(\mathcal{P})$ as a kinetic data structure. As mentioned in the introduction, this is accomplished by maintaining a proof of the correctness of $\mathcal{T}(\mathcal{P})$, which consists of a small set of elementary conditions called *certificates*. As long as the certificates remain valid, the current combinatorial structure of $\mathcal{T}(\mathcal{P})$ is valid. $\mathcal{T}(\mathcal{P})$ is updated only when some certificate fails. A certificate failure is called an *event*. All the events are placed in an event queue according to their failure time. The structure is then maintained by processing those events one by one. The efficiency of such a data structure depends on the size of the proof, the number of events, and the number of certificates that need to be updated at each event.

In our set-up, we certify that each face in $\mathcal{T}(\mathcal{P})$ is a pseudo-triangle and that the faces cover $\mathcal{F}(\mathcal{P})$. It turns out that the latter claim does not have to be explicitly checked, assuming it was

[1]For polygonal lines, all the interior vertices are assumed to be reflex vertices while the end vertices are not. A single point is always assumed non-reflex.

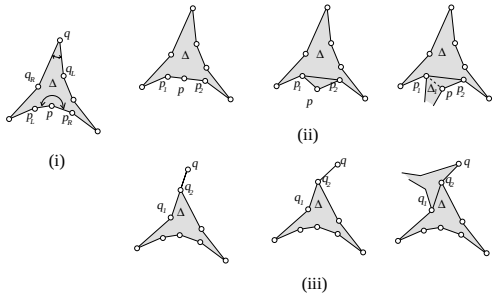


Figure 2. The certificates and updates for pseudo triangulation. (i) The reflex and corner certificates; (ii) the update for a reflex certificate failure; (iii) the update for a corner certificate failure.

true at the beginning of time. Thus we need to certify only the following two conditions for each pseudo-triangle Δ in $\mathcal{T}(\mathcal{P})$:

1. Each side chain C is concave, i.e. for each interior vertex p of C, the angle $\angle p_L p p_R > \pi$ (or the signed area of $\triangle p_L p p_R$ is negative). We call these certificates *reflex* certificates.
2. The side chains of a pseudo-triangle Δ join only at their endpoints, i.e. for each corner vertex q of Δ, the angle $\angle q_L q q_R < \pi$ (or the signed area of $\triangle q_L q q_R$ is positive). We call such certificates *corner* certificates.

When the reflex certificate of p fails, we connect the vertices adjacent to p, say p_1, p_2, to maintain the pseudo-triangularity of Δ. To maintain the pseudo-triangulation, we may also have to update other pseudo-triangles, depending on the edges adjacent to p. If both edges adjacent to p are boundary edges, then we add a triangle pp_1p_2 to $\mathcal{T}(\mathcal{P})$. If only one edge, say pp_1, is a diagonal edge, then pp_1 must be on another pseudo-triangle Δ_1. In this case, we delete the edge pp_1 and merge the triangle pp_1p_2 into the pseudo-triangle Δ_1. If both pp_1, pp_2 are diagonal edges, we then delete one of pp_1 or pp_2: which edge is deleted depends on the specific pseudo-triangulation we are maintaining; we will explain the choice for each pseudo-triangulation later in the paper.

When the corner certificate of q fails, we collapse the corner. Suppose that the vertices adjacent to q are q_1, q_2 and when the concave certificate corresponding to q fails, q_2 lies on the edge qq_1. We then delete the edge qq_1 and add the edge q_1q_2 to maintain the pseudo-triangularity of Δ. This process also effectively adds the point q_2 to the side chain of the other pseudo-triangle incident to the edge qq_1. Since there is only one way to update the pseudo-triangulation when a corner certificate fails, we will not describe updates for this case in the following sections.

In addition, we also have to maintain the boundary $\partial\mathcal{F}(\mathcal{P})$. $\partial\mathcal{F}(\mathcal{P}) \setminus \partial\overline{\mathcal{P}}$ is automatically maintained by the algorithm. By regarding $\partial\overline{\mathcal{P}}$ as a concave chain with respect to the exterior of $\overline{\mathcal{P}}$, it can also be maintained using reflex certificates for each vertex on the hull. We omit the details.

Since the pseudo-triangulation is not a canonical structure (there may be many pseudo triangulations of $\mathcal{F}(\mathcal{P})$), it is not clear which pseudo-triangulation we are maintaining. To avoid this ambiguity, we first describe a static algorithm that constructs a "specific" pseudo-triangulation of $\mathcal{F}(\mathcal{P})$. We then assume that, at any time t, the kinetic data structure maintains the pseudo-triangulation that the static algorithm would have constructed on $\mathcal{P}(t)$. This way, the structure is *canonical*, and it is more convenient for the analysis. Such approach has been taken in maintaining other structures, for example, the binary space partitioning structure in [1].

To maintain this invariant, we need additional certificates. When these certificates fail, we usually update the structure by the *flipping* operation: For any diagonal edge e, consider the two pseudo-triangles Δ_1, Δ_2 that contain e. The union of Δ_1, Δ_2 forms a pseudo-quadrangle $\diamond$—a cell

Figure 3. The incremental pseudo-triangulation of a point set. In (i), the sweep line has just passed p; (ii) shows the final pseudo-triangulation.

whose boundary consists of four concave chains. Besides e, there is exactly one other diagonal edge inside $\diamond$, which is called the *shadow edge* of e. The quadrangle $\diamond$ can be decomposed into two pseudo-triangles in two ways, one by adding e and the other by adding e's shadow edge. The flipping operation replaces e by e'.

3 Pseudo-Triangulation for Points

In this section, we assume $\mathcal{P}$ to be a set of n points. we define the *incremental pseudo-triangulation* for $\mathcal{P}$, and show how to maintain it efficiently. We also show it can be refined to maintain a triangulation of $\mathcal{P}$.

3.1 Incremental pseudo-triangulation

The pseudo-triangulation for points is built in an incremental manner as follows. We first sort the points in increasing order of their x-coordinates. Suppose that $p_1, p_2, \ldots, p_n$ is the sorted sequence. We then place a vertical line at p_1 and sweep it from left to right to build the pseudo-triangulation incrementally. When the sweep line passes the point p_k, we draw two tangent segments from p_k to the convex hull of $\mathcal{P}_{k-1}$. Denote by $u(p_k)$ and $d(p_k)$ the left endpoints of the upper and lower tangent segments to $\mathcal{P}_{k-1}$ from p_k, respectively (Figure 3). The concave chain on $\overline{\mathcal{P}}_{k-1}$ between $u(p_k)$ and $d(p_k)$ and line segments $p_k u(p_k)$, $p_k d(p_k)$ form a pseudo-triangle $\Delta(p_k)$ — its boundary consists of a concave chain and two single edges. Let $C(p_k)$ denote the concave chain on $\Delta(p_k)$. The three corners of $\Delta(p_k)$ are the points p_k, $d(p_k)$, and $u(p_k)$. After processing every point in $\mathcal{P}$, we obtain a pseudo-triangulation, which is called the *incremental pseudo-triangulation (IPT)* of $\mathcal{P}$.

3.2 Maintaining the incremental pseudo-triangulation

As the points move, $IPT(\mathcal{P})$ changes continuously, but its combinatorial structure changes only at discrete times.

Lemma 3.1. *As long the x-ordering of the points in $\mathcal{P}$ does not change, the reflex and corner certificates certify the incremental pseudo-triangulations.*

In view of this lemma, as long as the x-ordering does not change, we have to update $IPT(\mathcal{P})$ only when a reflex or corner certificate fails. When a corner event fails, the structure can be updated as described in Section 2.2. When a reflex certificate fails, we have two choices as which edge to delete — such choice is not difficult, and we omit the details in this abstract.

Figure 4. (i) Two points exchange in x-order. (ii) Fan triangulation of $IPT(\mathcal{P})$.

Next, we consider the case in which two points exchange in x-order. Although such an event does not affect the validity of the pseudo-triangulation, it does cause a change to $IPT(\mathcal{P})$ since the incremental ordering of the points changes. Suppose that p passes q from the left at time t and p is above q (Figure 4). Let r be $d(p)$ just before time t, and let r' be $u(q)$ immediately after time t. Then the structure is updated by switching the edge rp with the edge $r'q$ — a flipping operation defined in Section 2.2. The point r' can be found by computing a tangent segment from q to the concave chain between $u(p)$ and $d(q)$, which can be done in $O(\log n)$ time. Notice that in this description, each event changes $IPT(\mathcal{P})$. Thus, the method is output-sensitive. We have shown that

Theorem 3.2. *The incremental pseudo-triangulation can be maintained in an output-sensitive manner. Each update of the structure takes $O(\log n)$ time.*

In the following, we shall bound the number of changes to $IPT(\mathcal{P})$ for points in constant degree algebraic motion.

3.3 Combinatorial changes to $IPT(\mathcal{P})$

We now obtain an upper bound on the number of combinatorial changes to $IPT(\mathcal{P})$, i.e., the number of events, under the assumption that the degree of motion of $\mathcal{P}$ is fixed. We define a function $\phi_p^q(t)$ for two distinct points $p, q \in \mathcal{P}$: if at time t, q is to the left of p, then $\phi_p^q(t)$ is the slope of the line that passes through p and q; otherwise, $\phi_p^q(t)$ is undefined. Since the motion has constant degree, the x-order of a pair of points can switch only a constant number of times. Thus, each $\phi_p^q(t)$ consists of a constant number of arcs, all of which are portions of a fixed-degree polynomial. Consider the family of functions $\Phi_p = \{\phi_p^q \,|\, p \neq q \in \mathcal{P}\}$. By our assumptions any two arcs in Φ_p intersect a constant number of times, say, $s-2$. For a point q to be $d(p)$ at time t, $\phi_p^q(t)$ must have the largest value among all the points in $\mathcal{P}$. This is to say, the number of changes to $d(p)$ is the same as the combinatorial complexity of the upper envelope of Φ_p, which is bounded by $\lambda_s(n)$ ($\lambda_s(n)$ is the maximum length of an (n, s) Davenport-Schinzel sequence and is roughly linear.). Similarly, we can bound the number of changes to $u(p)$ by $\lambda_s(n)$. Summing over all the vertices in $\mathcal{P}$, we thus have

Theorem 3.3. *When the points of $\mathcal{P}$ move algebraically with constant degree, $IPT(\mathcal{P})$ changes $O(n\lambda_s(n))$ times, where s is a constant that depends on the degree of the motion.*

3.4 Triangulation of $\mathcal{P}$

It is very easy to obtain a triangulation of $\mathcal{P}$ from $IPT(\mathcal{P})$ by connecting each point p to every interior point on $C(p)$, thereby creating a fan inside each $\Delta(p)$. We call such a triangulation a *fan triangulation*. Although $IPT(\mathcal{P})$ changes only nearly quadratically many times, we are not able

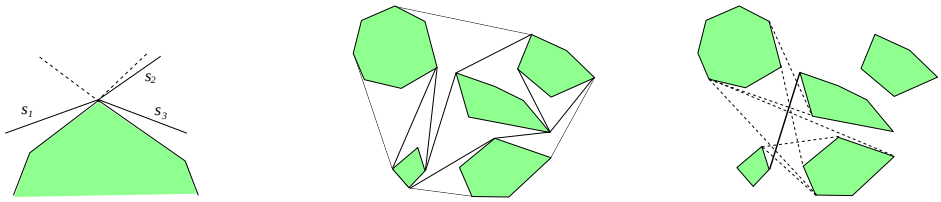


Figure 5. (i) Intersecting and non-intersecting tangents: s_1, s_2 are intersecting, but s_1, s_3 and s_2, s_3 are non-intersecting tangents. (ii) Greedy vertical pseudo-triangulation. (iii) Left-to-right property. The solid segment is an edge in $GVPT$. The dotted ones are the free tangents it crosses.

to prove a similar bound on the number of changes in the fan triangulation. The problem comes from the x-ordering event, which we process by switching a fan of p to a fan of q, or vice versa (Figure 4 (ii)). Using the previous notation, the cost of such switching is proportional to the length of the chain between r and r'. In the worst case, it might be $\Theta(n)$. This would give us a naïve bound of $O(n^3)$ on the number of changes to the fan triangulation.

For linear motion, we obtain a better upper bound on the number of changes to the fan triangulation by a global argument. We omit the proof from this abstract.

Theorem 3.4. *For points in linear motion, the number of changes to the fan triangulation is bounded by $O(n^{4/3}\lambda_s(n))$, where s is a constant.*

The obvious open problems are whether it is possible to extend this result to algebraic motions and whether the bound is tight.

4 Pseudo-Triangulation for Convex Polygons

Next we consider the case in which $\mathcal{P}$ is a set of k convex polygons with a total of n vertices. We describe a different pseudo-triangulation for $\mathcal{P}$, called the *greedy vertical pseudo-triangulation*, which is based on a result by Pocchiola and Vegter [10]. They introduced *greedy pseudo-triangulation* as a tool to compute the visibility complex of a set of convex polygons. We will modify their algorithm for our application

4.1 Greedy pseudo-triangulation for convex polygons

There are exactly four bi-tangent segments between any two disjoint convex polygons. A bi-tangent is called *free* if it does not intersect the interior of any object in $\mathcal{P}$. Let s and s' be two free bi-tangents with a common endpoint v, which is a vertex of a polygon $P \in \mathcal{P}$. We say that s and s' cross at v if P lies in the wedge of angle greater than π formed by s and s'; see Figure 5.[2] We say that two free bi-tangents intersect either if they share an interior point or if they have a common endpoint and they cross at that endpoint. For any linear ordering $\prec$ on the free tangents, we can build a corresponding greedy triangulation $\mathcal{T}_{\prec}(\mathcal{P})$ as follows. We first sort the tangents by $\prec$ ordering. We then scan through this list and maintain a set S of tangents that we have added to the pseudo-triangulation so far. At each step, we pick the next tangent segment s in the sorted sequence and check whether it intersects any tangents in S. If s does not intersect any segment of

[2]Intuitively, if we smooth the polygon P by taking the Minkowski sum of P with a disk of a sufficiently small radius, say δ, and we translate s and s' by at most δ so that they become tangents to the resulting polygons at their endpoints, then s and s' cross at v if and only if the translated copies of s and s' intersect.

S, we add it to S; otherwise, we discard s. After processing all the segments, we obtain a set of non-intersecting free tangents. It is shown in [10] that $|S| = 3k - 3$, and that together with the object boundaries, S forms a pseudo-triangulation of $\mathcal{F}(\mathcal{P})$ consisting of $2k - 2$ pseudo-triangles. By Lemma 2.1, this procedure constructs a minimal pseudo-triangulation.

For any line segment s, define $\theta(s)$ to be the minimum positive angle by which we have to rotate a vertical segment in the clockwise direction so that it becomes parallel to s. We order all the free tangents in increasing order of $\theta(\cdot)$. The greedy pseudo-triangulation created using this ordering is called the *greedy vertical pseudo-triangulation (GVPT)* and is denoted by $\mathcal{G}(\mathcal{P})$ (Figure 5 (ii)). It is shown in [10] that $\mathcal{G}(\mathcal{P})$ can be constructed in $O(k \log n)$ time, provided that each polygon is represented as an array storing its vertices in a clockwise (or counterclockwise) order.

The following local rule determines whether a free bi-tangent is in $\mathcal{G}(\mathcal{P})$.

Lemma 4.1 (Left-to-right property). *A bi-tangent segment s is in $\mathcal{G}(\mathcal{P})$ if and only if $\theta(s) < \theta(s')$ for each free bi-tangent s' intersected by s (Figure 5 (iii)).*

Recall that if we replace an edge in a pseudo-triangulation with its shadow edge, we still obtain a pseudo-triangulation. It turns out that in $\mathcal{G}(\mathcal{P})$ the shadow edge of an edge s is minimal in the ordering $\prec$ among all free bi-tangents crossing s. This implies the following property:

Lemma 4.2 (Local property). *Suppose that s is an edge in $\mathcal{T}_{\prec}(\mathcal{P})$. Denote by $\prec'$ the same ordering as $\prec$ except assigning s as the maximum element of $\prec'$. Then $\mathcal{T}_{\prec'}(\mathcal{P})$ can be obtained from $\mathcal{T}_{\prec}(\mathcal{P})$ by replacing s with its shadow edge.*

By this lemma, if the ordering of an edge changes, we just perform a local flip operation to fix the greedy pseudo-triangulation.

4.2 Maintaining the greedy pseudo-triangulation

We now describe how to maintain $\mathcal{G}(\mathcal{P})$ as a KDS so that it can be updated efficiently as the polygons in $\mathcal{P}$ move or deform. As described in Section 2.2, we maintain corner and reflex certificates for each pseudo-triangle in $\mathcal{G}(\mathcal{P})$. In addition, for each diagonal edge s of $\mathcal{G}(\mathcal{P})$, we maintain a *diagonal certificate* to certify that $\theta(s) < \theta(s')$, where s' is the shadow edge of s in $\mathcal{G}(\mathcal{P})$. This adds the requirement to maintain the shadow edges although such edges do not appear in $\mathcal{G}(\mathcal{P})$. Again, the shadow edges can be maintained in the same manner by corner and reflex certificates. We will show that these certificates are sufficient to maintain $\mathcal{G}(\mathcal{P})$.

As in Section 3, we can prove the following analogue to Lemma 3.1. The proof is omitted from this version.

Lemma 4.3. *If no diagonal certificate fails, then $\mathcal{G}(\mathcal{P})$ changes only when one of the corner or reflex certificates fails.*

When a corner or reflex certificate fails, we can update $\mathcal{G}(\mathcal{P})$ as described in Section 2.2, except for one subtle issue in the case of reflex certificates. When an interior vertex p of a side chain $\ldots p_L p p_R \ldots$ ceases to be a reflex vertex, we add the edge $p_L p_R$ and we have to delete one of the edges $p_L p$ and $p p_R$. Instead of deleting one of them arbitrarily, we delete $p_L p$ if $\theta(p_L p) > \theta(p p_R)$; see Figure 6.

Next, we consider diagonal certificates. If the certificate of a diagonal edge s fails, then either s or its shadow edge is vertical. When such an event happens, the slope ordering of an edge jumps from the minimum to maximum, or vice versa. By the local property of greedy pseudo-triangulations, we can simply perform a flipping operation to s to maintain $\mathcal{G}(\mathcal{P})$.

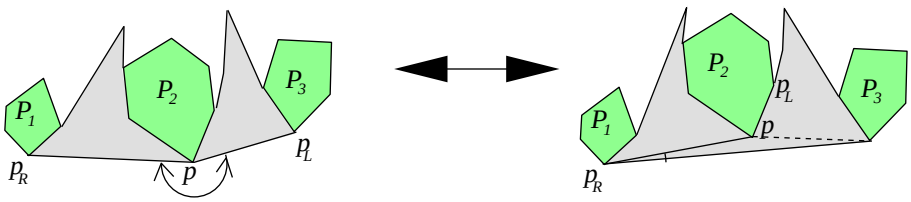


Figure 6. The reflex(from left to right) and corner(from right to left) events and the updates. When a reflex certificate fails, we choose the edge with smaller slope, as shown in the right figure.

Since $\mathcal{G}(\mathcal{P})$ has $O(k)$ diagonals and pseudo-triangles, the number of corner and diagonal certificates is $O(k)$. We assume that the polygons remain convex at all times, therefore it suffices to maintain reflex certificates for a vertex only if it is adjacent to a diagonal edge, i.e., a free bi-tangent. The number of such vertices is also $O(k)$. Hence, we obtain the following:

Theorem 4.4. *$\mathcal{G}(\mathcal{P})$ can be maintained by using a structure with $O(k)$ certificates. Each event can be processed in time $O(\log n)$.*

This data structure works even if the polygons deform continuously over time, as long as they remain convex at all times. What matters is the number of events. The rigid motion can give us better bounds as stated in Theorem 4.6.

4.3 Combinatorial changes to $\mathcal{G}(\mathcal{P})$

Next, we bound the number of changes to $\mathcal{G}(\mathcal{P})$ if the degree of motion of $\mathcal{P}$ is fixed. It can be shown that a corner or reflex certificate fails when three polygons of $\mathcal{P}$ become collinear, i.e., a line is tangent to three polygons. Hence, a combinatorial change in $\mathcal{G}(\mathcal{P})$ happens only when three polygons of $\mathcal{P}$ are collinear or when a bi-tangent becomes vertical. The number of such events can be bounded by the following lemma.

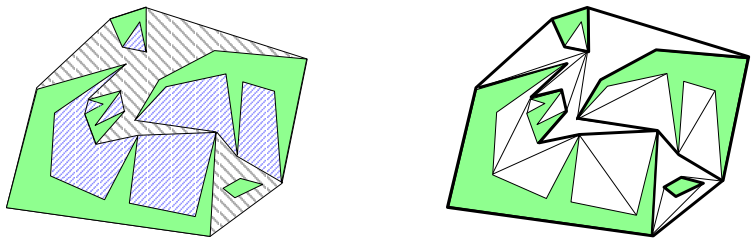
Lemma 4.5. *Suppose that P_1, P_2, P_3 are convex polygons with n_1, n_2, n_3 vertices, respectively, and the degree of their motion is constant. They can become collinear $O(n_1 + n_2 + n_3)$ and $O(n_1n_2 + n_1n_3 + n_2n_3)$ times for translational and rigid motions, respectively. The bi-tangent between P_1, P_2 can become vertical $O(1)$ and $O(n_1 + n_2)$ times for translational and rigid motions, respectively.*

Applying this lemma to all triplets of $\mathcal{P}$, we obtain the following weak upper bounds on the number of changes to $\mathcal{G}(\mathcal{P})$.

Theorem 4.6. *Let $\mathcal{P}$ be a set of k polygons with a total of n vertices. $\mathcal{G}(\mathcal{P})$ changes $O(kn^2)$ times if the degree of motion of $\mathcal{P}$ is fixed. If the motion is translational, the number of changes is only $O(k^2n)$. If the polygons may deform, the number of changes is bounded by $O(n^3)$.*

5 Pseudo-Triangulation for Simple Polygons

In this section, we consider the case in which $\mathcal{P}$ is a set of k pairwise-disjoint simple polygons with a total of n vertices. We will describe a method for maintaining a pseudo-triangulation of $\mathcal{P}$ by combining our algorithm for convex polygons with the algorithm by Basch *et al.* [2] for two simple polygons.



(i) (ii)

Figure 7. The mixed pseudo-triangulation. (i) $\mathcal{F}_1(\mathcal{P})$ and $\mathcal{F}_2(\mathcal{P})$ are shaded differently. (ii) Mixed pseudo-triangulation; thick edges denote the boundary between $\mathcal{F}_1(\mathcal{P})$ and $\mathcal{F}_2(\mathcal{P})$.

5.1 The mixed pseudo-triangulation

For each $P \in \mathcal{P}$, define the *relative geodesic cycle* $C(P)$ of P to be the shortest cycle in $\mathcal{F}(\mathcal{P})$ with the same homotopy type as ∂P. The region enclosed by $C(P)$ is called the *relative convex hull* and is denoted by $\overline{P}$. We decompose $\mathcal{F}(\mathcal{P})$ into two parts $\mathcal{F}_1(\mathcal{P}) = \mathcal{F}(\mathcal{P}) \cap (\bigcup \overline{P})$ and $\mathcal{F}_2(\mathcal{P}) = \mathcal{F}(\mathcal{P}) \setminus \bigcup \overline{P}$. We compute pseudo-triangulations $\mathcal{T}(\mathcal{F}_1(\mathcal{P}))$ and $\mathcal{T}(\mathcal{F}_2(\mathcal{P}))$ separately. These two triangulations together give the *mixed pseudo-triangulation* of $\mathcal{P}$.

First, we consider $\mathcal{F}_1(\mathcal{P})$. Since the interiors of all the relative convex hulls are pairwise disjoint, we can compute a pseudo-triangulation of each of them separately. We triangulate each $\overline{P}$ as in [2]. Following [2], we define the *pinned relative geodesic cycle* of P, with respect to a pinning point set B (a subset of the vertices of P), to be the shortest cycle in $\mathcal{F}(\mathcal{P})$ that passes through the vertices in B and has the same homotopy type as ∂P. The *pinned relative convex hull* of P with respect to B is the region enclosed by the pinned relative geodesic cycle. If B contains all the vertices of P, the pinned relative convex hull of P is P itself. Consider a balanced binary tree $T(P)$ with the vertices of P as leaves, in the order they appear on ∂P; each internal node stores the vertex from its left child. We can then build $\log |P|$ pinned relative geodesic cycles, where the pinning set of each cycle is the set of vertices on the same level of $T(P)$. By overlaying these layers, we can obtain a pseudo-triangulation of $\overline{P} \setminus P$. This pseudo-triangulation is called the *external relative geodesic triangulation*. Refer to [2] for the maintenance and bounds on the number of changes of this structure. Now, we consider $\mathcal{F}_2(\mathcal{P})$. In the following, we generalize the greedy pseudo-triangulation, defined in Section 4, to a connected polygonal subdivision. Consider a connected polygonal region $\mathcal{F}$. A vertex v of $\mathcal{F}$ is called a *corner* vertex if the interior angle at v is less than 180°; otherwise, v is called a *reflex* vertex. For a point p on ∂F, a line segment pq in $\mathcal{F}$ is called *tangent* to ∂F at p if p is a corner vertex or the line passing through p, q locally supports ∂F at p. (The intuition behind the definition of "tangent" for corner vertices comes from the case in which a corner is formed by two separate convex objects whose boundaries touch. Then a "tangent" segment to the corner is indeed tangent to one of the two convex objects.) For two vertices p, q on ∂F, the line segment pq is called a free bi-tangent if pq lies in $\mathcal{F}$ and is tangent to ∂F at both p and q. We now construct the greedy vertical pseudo-triangulation of $\mathcal{F}$, as described in Section 4. It can be shown that the algorithm constructs a minimal pseudo-triangulation of $\mathcal{F}$. The following lemma bounds the number of pseudo-triangles in such a pseudo-triangulation.

Lemma 5.1. *If ∂F consists of k connected components and m corner vertices, the number of pseudo-triangles in the greedy pseudo triangulation of $\mathcal{F}$ is $O(k + m)$.*

The relative geodesic triangulation $\mathcal{T}(\mathcal{F}_1(\mathcal{P})$ and greedy pseudo-triangulation $\mathcal{T}(\mathcal{F}_2(\mathcal{P}))$ together form a pseudo-triangulation of $\mathcal{F}(\mathcal{P})$, which we call the *mixed pseudo-triangulation* (Fig-

ure 7). Next, we bound the size of the mixed pseudo-triangulation.

We call a diagonal edge of the mixed pseudo-triangulation a *bridge* edge if it connects two different polygons of $\mathcal{P}$. As we will see later, bridge edges play an important role in maintenance of the pseudo-triangulation.

Lemma 5.2. *$\partial\mathcal{F}_2(\mathcal{P})$ consists of $O(k)$ connected components and $O(k)$ bridge edges and corner vertices.*

Lemmas 5.1 and 5.2 imply the following.

Corollary 5.3. *The greedy triangulation $\mathcal{T}(\mathcal{F}_2(\mathcal{P}))$ of $\mathcal{F}_2(\mathcal{P})$ has $O(k)$ pseudo-triangles.*

Next, we bound the size of $\mathcal{T}(\mathcal{F}_1(\mathcal{P}))$. We will focus on the bridge edges in $\mathcal{F}_1(\mathcal{P})$. Basch *et al.* [2] have shown that any line segment lying inside $\mathcal{F}(\mathcal{P})$ crosses $O(\log n)$ bridge edges, and thus the number of bridge edges is $O(\tau \log n)$, where τ is the size of the min-link separator between two simple polygons. Here, we generalize their result to our setup by arguing that any free line segment can cross at most two relative convex hulls.

Lemma 5.4. *Any free line segment crosses $O(\log n)$ bridge edges of $\mathcal{T}(\mathcal{F}_1(\mathcal{P}))$.*

A *minimum link subdivision* is a polygonal subdivision of the plane, using as few (line segment) edges as possible, such that each $P \in \mathcal{P}$ is contained in its own face of the subdivision. A *minimum link separator* for a polygon $P \in \mathcal{P}$ is a simple polygon homotopic to ∂P with as few edges as possible.

Consider any Jordan cycle C in $\mathcal{F}(\mathcal{P})$ that is homotopic to ∂P. Clearly, C must intersect all the bridge edges incident to P because C separates P from other objects in $\mathcal{P}$. Suppose that C consists of τ line segments. Since each line segment on C crosses at most $O(\log n)$ bridge edges connected to P and each bridge edge is crossed at least once, we can conclude that the number of bridge edges incident to P is bounded by $O(\tau \log n)$. Therefore, we have the following result on the number of bridge edges.

Lemma 5.5. *The number of bridge edges in $\mathcal{T}(\mathcal{F}_1(\mathcal{P}))$ is bounded by $O(\kappa \log n)$, where κ is the number of edges in a minimum link subdivision of $\mathcal{P}$. For each polygon P in $\mathcal{P}$, the number of bridge edges connected to P is bounded by $O(\tau(P) \log n)$, where $\tau(P)$ is the number of edges in a minimum link separator of P.*

Thus, we obtain the following bounds on the size of the mixed pseudo triangulation by combining Corollary 5.3 and Lemma 5.5.

Theorem 5.6. *The size of the mixed pseudo-triangulation is $O(n)$, and among all the diagonal edges, only $O(\kappa \log n)$ edges are bridge edges, where κ is the number of edges in a minimum link subdivision of $\mathcal{P}$.*

5.2 Maintaining the mixed pseudo-triangulation

As shown in [2], the external relative geodesic triangulation can be maintained by corner and reflex certificates only, and there are local rules to decide which edge to select when a reflex certificate fails. Therefore, the mixed triangulation can be easily maintained, although it comprises two completely different structures. As argued before, the number of certificates in the certification structure of $\mathcal{T}(\mathcal{P})$ is proportional to the number of bridge edges in $\mathcal{T}(\mathcal{P})$. According to Lemmas 5.2 and 5.5, we may bound the certificates for certifying $\mathcal{T}(\mathcal{P})$ as follows.

Theorem 5.7. *The number of certificates in $\mathcal{T}(\mathcal{P})$ is bounded by $O(\kappa \log n)$, where κ is the number of edges in the minimum link subdivision of $\mathcal{P}$. For each polygon P in $\mathcal{P}$, the number of certificates involving P is bounded by $O(\tau(P) \log n + k)$, where $\tau(P)$ is the number of edges in the minimum link separator of P.*

A trivial bound on the number of changes to the mixed pseudo-triangulation for polygons in rigid motion is $O(n^3)$, which is the number of times when three vertices become collinear.

6 Conclusions

We have shown how to efficiently maintain pseudo-triangulations of the free space for points, convex, and simple polygons moving around in the plane. In addition to collision detection, the pseudo-triangulation structure provides a simple way for walking around the free space and performing operations such as ray shooting and other visibility queries. Unlike any previous collision detection method, our deforming tilings can be adapted to work for deformable objects, thus enabling new applications where simulation of flexible shapes is important. An extension of our ideas to 3D presents the next obvious challenge.

References

[1] P. K. Agarwal, L. J. Guibas, T. M. Murali, and J. S. Vitter. Cylindrical static and kinetic binary space partitions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 39–48, 1997.

[2] J. Basch, J. Erickson, L. J. Guibas, J. Hershberger, and L. Zhang. Kinetic collision detection for two simple polygons. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 102–111, 1999.

[3] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756, 1997.

[4] S. Cameron. Collision detection by four-dimensional intersection testing. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 291–302, 1990.

[5] D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 58–67, 1998.

[6] J. Erickson, L. J. Guibas, J. Stofi, and L. Zhang. Separation-sensitive kinetic collision detection for convex objects. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, 1999.

[7] L. J. Guibas. Kinetic data structures: A state of the art report. In *Proc. 3rd Workshop on Algorithmic Foundations of Robotics*, page to appear, 1998.

[8] P. Gupta, R. Janardan, and M. Smid. Fast algorithms for collision and proximity problems involving moving geometric objects. *Comput. Geom. Theory Appl.*, 6:371–391, 1996.

[9] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Trans. Visualization and Computer Graphics*, 1(3):218–230, Sept. 1995.

[10] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudo-triangulations. *Discrete Comput. Geom.*, 16:419–453, Dec. 1996.

[11] M. K. Ponamgi, D. Manocha, and M. C. Lin. Incremental algorithms for collision detection between general solid models. In *Proc. ACM Siggraph Sympos. Solid Modeling*, pages 293–304, 1995.

[12] E. Schömer and C. Thiel. Efficient collision detection for moving polyhedra. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 51–60, 1995.

[13] S. Suri. *Minimum link paths in polygons and related problems.* Ph.D. thesis, Dept. Comput. Sci., Johns Hopkins Univ., Baltimore, MD, 1987.