Efficient Algorithms for Counting and Reporting Pairwise Intersections between Convex Polygons

Prosenjit Gupta*† Ravi Janardan‡* Michiel Smid§

April 16, 1996

Abstract

Let $\mathcal{S}$ be a set of convex polygons in the plane with a total of n vertices, where a polygon consists of the boundary as well as the interior. Efficient algorithms are presented for the problem of reporting output-sensitively (resp. counting) the I pairs of polygons that intersect. The algorithm for the reporting (resp. counting) problem runs in time $O(n^{4/3+\epsilon}+I)$ (resp. $O(n^{4/3+\epsilon})$), where $\epsilon > 0$ is an arbitrarily small constant. This result is based on an interesting characterization of the intersection of two convex polygons in terms of the intersection of certain trapezoids from their trapezoidal decomposition. Also given is an alternative solution to the reporting problem, which runs in $O(n^{4/3}\log^{O(1)} n + I)$ time, and is based on characterizing the intersection of two convex polygons via the intersection of their upper and lower chains and their leftmost vertices. The problems are interesting and challenging because the output size, I, can be much smaller than the total number of intersections between the boundaries of the polygons and because the number of polygons and their sizes can depend on n.

1 Introduction

Let $\mathcal{S}$ be a set of r bounded, convex polygons in the plane with a total of n vertices. By a *polygon*, we mean the region consisting of the boundary as well as the interior. Polygons P and Q are said to *intersect* if they share a point; in particular, they intersect if one is completely contained inside the other or if their boundaries intersect.

We consider efficient algorithms for reporting output-sensitively (resp. counting) all intersecting pairs of polygons in $\mathcal{S}$. By *output size* we mean the number of intersecting

*The research of these authors was supported in part by NSF grant CCR-92-00270.

†Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany. E-mail: pgupta@mpi-sb.mpg.de.

‡Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A. E-mail: janardan@cs.umn.edu. Also supported in part by a Grant–in–Aid of Research from the Graduate School of the University of Minnesota.

§Department of Computer Science, King's College London, Strand, London WC2R 2LS, United Kingdom. E-mail: michiel@dcs.kcl.ac.uk. Part of this work was done while the author was at the Max-Planck-Institut für Informatik, Saarbrücken.

pairs of polygons in $\mathcal{S}$; we denote this by I. Let K denote the total number of intersections between the polygons in $\mathcal{S}$. That is, if P is completely contained in Q, or vice versa, we count this in K as one intersection; otherwise, if the boundaries of P and Q intersect, then we count all the boundary intersections in K. Note that I is no larger than K and, in general, it can be much smaller: The boundaries of P and Q can intersect $O(|P| + |Q|)$ times; however, this is counted in I as just a single intersection between P and Q.

Output-sensitivity is only one of the reasons why the problem we consider is interesting and non-trivial. A second reason has to do with the number of polygons and their sizes. We note at the outset that the problem is straightforward if either (a) r is a constant, or (b) all the polygons are of constant size. Consider case (a): Assuming that the polygons P and Q are represented appropriately, we can use the algorithm of Chazelle and Dobkin [CD87] to decide if they intersect in time $O(\log(|P| + |Q|)) = O(\log n)$. Considering all pairs of polygons in this way, we can solve the problem in time $O(r^2 \log n + I) = O(\log n)$. (Here $I = O(1)$, since $r = O(1)$.) Next, consider case (b): Note that $I = \Theta(K)$ since all the polygons are of constant size. We can compute the at most K boundary intersections among the polygons in $\mathcal{S}$ by using the algorithm of Chazelle and Edelsbrunner [CE92] in time $O(n \log n + K)$. Moreover, we can determine the at most K complete containments among the polygons by triangulating them and then stabbing the triangles with a vertex of each polygon. This takes time $O(n^{4/3} \log^{O(1)} n + K)$ using the algorithm given in [Aga91, Corollary 5.14]. It follows that the total time is $O(n^{4/3} \log^{O(1)} n + I)$.

The interesting case is when both r and the polygon sizes depend on n, for then the two approaches outlined above are not efficient. For example, assume that each polygon has size n^α, for some α, $0 < \alpha < 2/3$; thus $r = n^{1-\alpha}$. Assume further that the polygons intersect pairwise in the maximum number of edges. Thus there are $\Theta(n^\alpha)$ intersections in each of the $\Theta(n^{2-2\alpha})$ pairs, so that $K = \Theta(n^{2-\alpha})$. Moreover, $I = \Theta(n^{2-2\alpha}) = o(K)$. The first approach above takes time $O(r^2 \log n + I) = O(n^{2-2\alpha} \log n)$. The second approach takes time $O(n^{4/3} \log^{O(1)} n + K)$, which is $\Theta(K)$ since $K = \Theta(n^{2-\alpha})$ and $\alpha < 2/3$. The first solution can be made nearly quadratic by a suitable choice of α, while the second solution is not sensitive to I.

The challenge then is to devise an algorithm for reporting the intersecting pairs of polygons in output-sensitive fashion, in time $O(f(n) + I)$, where $f(n)$ is subquadratic and small. For the more difficult counting problem, we seek an algorithm with running time $O(f(n))$. (Note that the counting problem can be solved using a reporting algorithm, but this is not efficient.) We remark that we are not aware of any previous work on this problem.

2 Summary of results

Our first result is a data structure of size $O(m^{1+\epsilon})$ (for any m satisfying $n \le m \le n^2$), which stores a set $\mathcal{S}$ of convex polygons with a total of n vertices, such that given a query convex polygon, Q, the I_Q polygons in $\mathcal{S}$ intersecting Q can be counted (resp. reported) in time $O(|Q| \cdot n^{1+\epsilon}/m^{1/2})$ (resp. $O(|Q| \cdot n^{1+\epsilon}/m^{1/2} + I_Q)$). Using the counting (resp. reporting) version of this data structure, we can count the pairs (resp.

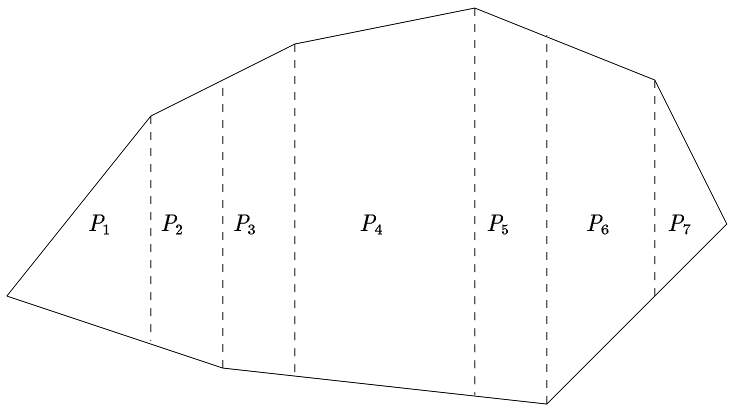

Figure 1: *Trapezoidal decomposition of a convex polygon*

report the I pairs) of intersecting polygons in time $O(n^{4/3+\epsilon})$ (resp. $O(n^{4/3+\epsilon} + I)$), for any constant $\epsilon > 0$. This algorithm is based on an interesting characterization of the intersection of two convex polygons in terms of intersecting pairs of trapezoids from their trapezoidal decomposition. We also give an alternative algorithm for the reporting problem, which runs in time $O(n^{4/3} \log^{O(1)} n + I)$. This algorithm is based on a different characterization of the intersection of a pair of convex polygons in terms of their upper and lower chains and their leftmost vertices.

3 The counting problem

In this section, we give an algorithm for counting the pairs of convex polygons that intersect. The technique works for the reporting problem as well.

3.1 Characterizing the intersection of two polygons

We need the notion of a *trapezoidal decomposition.* Let P be a convex polygon. Draw a vertical line through each vertex of P. This partitions P into $O(|P|)$ trapezoids and triangles $P_1, P_2, \ldots$, sorted from left to right. We consider a triangle as a degenerate trapezoid. (See Figure 1.) We define an artificial trapezoid P_0, which is immediately to the left of P_1, and which "behaves" like an empty trapezoid. By definition, P_0 does not intersect anything. (P_0 can be represented by four halfplanes whose intersection is empty.) Note that a trapezoid also consists of a boundary together with its interior.

For each polygon P of $\mathcal{S}$, we define a new set $\mathcal{P}$ consisting of all pairs (P_i, P_{i+1}), $i \geq 0$. We call each pair (P_i, P_{i+1}) a *trapezoidal pair* of P.

Definition 1 *Let P and Q be two convex polygons. Consider the sets $\mathcal{P}$ and $\mathcal{Q}$. We say that the elements $(P_i, P_{i+1}) \in \mathcal{P}$ and $(Q_j, Q_{j+1}) \in \mathcal{Q}$ have a* conflict *if*

1. *$P_{i+1} \cap Q_{j+1} \neq \emptyset$, and*
2. *$P_i \cap Q_{j+1} = \emptyset$, and*
3. *$P_{i+1} \cap Q_j = \emptyset$.*

Theorem 1 *Let P and Q be two convex polygons. Then P and Q intersect if and only if there are indices i and j such that (P_i, P_{i+1}) and (Q_j, Q_{j+1}) have a conflict. Moreover, if such indices i and j exist, then they are unique.*

Proof: Suppose there are indices i and j such that (P_i, P_{i+1}) and (Q_j, Q_{j+1}) have a conflict. Then P_{i+1} and Q_{j+1} have a point in common. Hence, P and Q also have a point in common, i.e., P and Q intersect.

To prove the converse, assume that P and Q intersect. Let x be the leftmost point in the intersection of P and Q. (If there is no unique leftmost point of intersection, then we take for x the leftmost point with minimum y-coordinate.) We distinguish three cases.

Case 1: x is in the interior of Q.

In this case, x must be the leftmost point of P. In particular, x is a point of the trapezoid P_1. Let $j \geq 0$ be the index such that $x \in Q_{j+1}$. (If x is on the boundaries of two trapezoids of Q, then we choose j such that x is on the right boundary—which is vertical—of Q_{j+1}.) Note that x does not belong to Q_j. We claim that the elements (P_0, P_1) and (Q_j, Q_{j+1}) have a conflict. Indeed, since $x \in P_1 \cap Q_{j+1}$, we have $P_1 \cap Q_{j+1} \neq \emptyset$. Also, by definition of the artificial trapezoid P_0, we have $P_0 \cap Q_{j+1} = \emptyset$. Finally, we have $P_1 \cap Q_j = \emptyset$: This follows from the facts that (i) x is the leftmost common point of P and Q, (ii) x does not belong to Q_j and (iii) Q_j is to the left of Q_{j+1}.

Case 2: x is in the interior of P. This case is symmetric to Case 1.

Case 3: x is on the boundaries of both P and Q.

Let i (resp. j) be the index such that $x \in P_{i+1}$ (resp. $x \in Q_{j+1}$). (If x is on the boundaries of two trapezoids of P (resp. Q), then we choose i (resp. j) such that x is on the right boundary of P_{i+1} (resp. Q_{j+1}).) We claim that the pairs (P_i, P_{i+1}) and (Q_j, Q_{j+1}) have a conflict. The case $P_{i+1} \cap Q_{j+1} \neq \emptyset$ is obvious. If we had $P_i \cap Q_{j+1} \neq \emptyset$, then x could not be the leftmost intersection since P_i is to the left of P_{i+1}. The case involving P_{i+1} and Q_j is similar.

This proves the first part of the theorem. Now assume there are indices i and j such that (P_i, P_{i+1}) and (Q_j, Q_{j+1}) have a conflict. We will prove that i and j are unique.

Among all indices $i \geq 0$ and $j \geq 0$ such that (P_i, P_{i+1}) and (Q_j, Q_{j+1}) have a conflict, choose those for which the pair (i, j) is lexicographically maximal. We consider four cases.

Case A: $i = j = 0$.

The way we chose the pair (i, j) immediately implies that i and j are unique.

Case B: $i = 0$ and $j \neq 0$.

We claim that there is no k such that $k < j$ and (P_0, P_1) and (Q_k, Q_{k+1}) have a conflict. Clearly, this claim will prove that i and j are unique.

To prove the claim, assume there is a $k < j$ such that (P_0, P_1) and (Q_k, Q_{k+1}) have a conflict. We know that $P_1 \cap Q_{k+1} \neq \emptyset$ and $P_1 \cap Q_{j+1} \neq \emptyset$. Let a and b be points of $P_1 \cap Q_{k+1}$ and $P_1 \cap Q_{j+1}$, respectively. Then, by convexity, the segment ab is completely contained inside

$$P_1 \cap (Q_{k+1} \cup Q_{k+2} \cup \ldots \cup Q_{j+1}).$$

Moreover, this segment passes through the trapezoid Q_j. Hence, $P_1 \cap Q_j \neq \emptyset$ which contradicts the fact that (P_0, P_1) and (Q_j, Q_{j+1}) have a conflict.

Case C: $i \neq 0$ and $j = 0$.

We claim that there are no k and ℓ such that $k < i$, $\ell \geq 0$, and (P_k, P_{k+1}) and $(Q_\ell, Q_{\ell+1})$ have a conflict. Clearly, this claim will prove that i and j are unique.

To prove the claim, assume there are $k < i$ and $\ell \geq 0$ such that (P_k, P_{k+1}) and $(Q_\ell, Q_{\ell+1})$ have a conflict. We know that $P_{k+1} \cap Q_{\ell+1} \neq \emptyset$ and $P_{i+1} \cap Q_1 \neq \emptyset$. Since $k < i$, the trapezoid P_{k+1} is to the left of P_{i+1}. But then ℓ must be equal to zero. As in Case B, let a and b be points of $P_{k+1} \cap Q_1$ and $P_{i+1} \cap Q_1$, respectively. Then the segment ab is completely contained inside

$$(P_{k+1} \cup P_{k+2} \cup \ldots \cup P_{i+1}) \cap Q_1.$$

Since this segment passes through the trapezoid P_i, it follows that $P_i \cap Q_1 \neq \emptyset$, which is a contradiction.

Case D: $i \neq 0$ and $j \neq 0$.

Let L and L' denote the left vertical sides of P_{i+1} and Q_{j+1}, respectively. Note that L and L' exist and that they are also the right vertical sides of P_i and Q_j, respectively. Since $P_i \cap Q_{j+1} = \emptyset$, L lies completely outside Q_{j+1}. Similarly, since $P_{i+1} \cap Q_j = \emptyset$, L' lies completely outside P_{i+1}. These two facts, together with the fact that $P_{i+1} \cap Q_{j+1} \neq \emptyset$, imply that the boundaries of P_{i+1} and Q_{j+1} intersect. In particular, the top side, t, of Q_{j+1} intersects the bottom side, b, of P_{i+1} or, symmetrically, the bottom side of Q_{j+1} intersects the top side of P_{i+1}. Assume without loss of generality that t intersects b. Then the slope of t is larger than that of b. (Otherwise, L would intersect Q_{j+1} or L' would intersect P_{i+1}.) By convexity, the polygon Q lies below the supporting line of t, and the polygon P lies above the supporting line of b. Hence, the polygon $P \cap Q$ does not contain any point to the left of the intersection of t and b. In particular, there are no indices k and ℓ such that (k, ℓ) is lexicographically smaller than (i, j), and (P_k, P_{k+1}) and $(Q_\ell, Q_{\ell+1})$ have a conflict. This proves that i and j are unique.

This completes the proof of the theorem. ■

In Section 3.2 below we review a useful query composition technique. Combining this with the characterization given in Theorem 1 leads to the counting algorithm we present in Section 3.3.

3.2 Review of a query composition technique

In Section 3.3 below, we will express intersection conditions as the conjunction of $h > 1$ halfplane range queries, where $h = O(1)$. Towards this end, we review a useful query composition result due to van Kreveld [vK92], which we will use. (This result is based on multi-level range searching structures [DE87, Mat92, CSW92].)

Let S be a set of n geometric objects. Let D be a data structure for some query problem on S, with building time, space and query time bounds of $p(n)$, $f(n)$ and $g(n)$, respectively. Suppose that we now wish to answer queries not on the entire set S but on a subset S' of S, where S' is specified by putting S in 1–1 correspondence with a set P of points in $\mathbb{R}^d$ and letting S' correspond to the subset P' of P lying in a query simplex. (In [vK92], this technique is called *simplex composition* on P to D.) The following theorem from [vK92] states how fast the query problem on S' can be solved. (We state only the part of the theorem relevant to us. Moreover, the building time is not explicitly mentioned in [vK92], but follows easily from the discussions there.)

Theorem 2 [vK92] *Let S, D, P and n be as above. For an arbitrarily small constant $\epsilon > 0$, simplex composition on P to D yields a data structure with building time $O(m^\epsilon(m + p(n)))$, size $O(m^\epsilon(m + f(n)))$, and query time $O(n^\epsilon(g(n) + n/m^{1/d}))$, for any $n \leq m \leq n^d$, assuming $f(n)/n$ is nondecreasing and $g(n)/n$ is nonincreasing. For the reporting problem, the output size, denoted by k, must be included in the query time as an additive term.* ■

In our application, the simplex will always be a halfplane. Given the $h = O(1)$ halfplanes, we proceed as follows: We design an initial data structure D. Then we apply Theorem 2, with one of the h halfplanes. This gives a new structure D' to which we apply Theorem 2 using a second halfplane and so on. Since $h = O(1)$, the space and query time bounds of the resulting structure are asymptotically the same as the ones given in Theorem 2.

We illustrate the above idea with the following simple example: Let S be a set of n vertical line segments in $\mathbb{R}^2$. We wish to construct a data structure for the following query problem on S: "Given a query line ℓ, count or report the k segments of S that are intersected by ℓ." We take D to be a linked list of the objects of S and store its size at its head. Clearly, D solves the trivial "query" problem, "count or report the segments of S", in $f(n) = O(n)$ space and $g(n) = O(1)$ query time. Since a segment $s \in S$ intersects ℓ iff its upper endpoint is in ℓ^+ and its lower endpoint is in ℓ^-, we can cast the intersection condition as two halfplane compositions: In the first, we associate S with the set P of upper endpoints and use ℓ^+; in the second, we associate S with the set P of lower endpoints and use ℓ^-. We then apply these two compositions successively using Theorem 2. This gives a data structure of size and building time $O(m^{1+\epsilon})$ and a query time of $O(n^{1+\epsilon}/m^{1/2})$ for the counting problem and $O(n^{1+\epsilon}/m^{1/2} + k)$ for the reporting problem.

3.3 The algorithm for the counting problem

Consider two trapezoids P and Q. The following lemma is easy to show.

Lemma 1 *P and Q intersect iff*
(i) P has a vertex inside Q, or
(ii) Q has a vertex inside P, or
(iii) an edge of P intersects an edge of Q. ■

First let us consider the following problem: Preprocess a set S of trapezoids in the plane, such that given a query trapezoid Q, the trapezoids in S that intersect Q can be counted efficiently. We can define a boolean formula $B(P, Q)$ in disjunctive normal form (DNF) such that $B(P, Q)$ is true iff P intersects Q. Each minterm in $B(P, Q)$ is the conjunction of literals of the form $p \in H$, where p is a point and H is a halfplane (either open or closed). The point p is either a vertex of P or the dual of a line supporting an edge of P, while H is a halfplane whose bounding line is either a supporting line of an edge of Q or is the dual of a vertex of Q. For example, the condition that a vertex $p \in P$ is in Q in Lemma 1 can be written as the conjunction of four closed halfplane membership conditions, where each halfplane is bounded by a supporting line of an edge of Q and contains Q. The condition that a vertex of Q is in P can be expressed similarly using duality. The intersection of an edge of P and an edge of Q can also be written similarly. (Note that the condition $p \notin H$—which arises when we want to express the non-intersection of two trapezoids, as in Definition 1—can also be expressed in a similar form as $p \in H^c$, where H^c is the open halfplane that is complementary to H.) Let M_i, $i = 1, 2, \ldots, k$, be the minterms of $B(P, Q)$. We may assume that $B(P, Q)$ is written in a form such that it is true iff exactly one of the M_i's is true. We can ensure this easily: we create the truth table for $B(P, Q)$ and, for each instance of a 1 as the truth value, we write out the corresponding minterm. Since P and Q are of constant size, $B(P, Q)$ has a constant number of minterms, each of constant size.

A trapezoid is composed of (at most) four vertices $p_i, i = 1, \ldots, 4$ and at most four edges $e_i, i = 1, \ldots, 4$. Let p_i^* denote the point that is dual to the line supporting e_i. Any of the minterms M can be written as

$$M = G_1 \wedge G_2 \wedge G_3 \wedge G_4 \wedge G_1^* \wedge G_2^* \wedge G_3^* \wedge G_4^*$$

where G_i (resp. G_i^*) is the AND of literals of the form $p_i \in H$ (resp. $p_i^* \in H$). We build a data structure corresponding to each minterm. Each such data structure is built on all the trapezoids P that we have. Each data structure is built on eight levels, corresponding to the p_i's and p_i^*'s. Given a query trapezoid Q, we query each data structure. If level j is built on p_i, then we look at the literals in $G_i = (p_i \in H_1 \wedge p_i \in H_2 \wedge \ldots \wedge p_i \in H_s)$. Each $H_k, k = 1, \ldots, s$ (where $s \leq 16$) is a halfplane bounded by one of the edges of Q or is dual to one of the vertices of Q. We query the data structure at level j by first searching using H_1, then searching with H_2 below nodes selected by H_1 and so on. Once we are done at level j, we explore the level $j+1$ structures at nodes selected at level j. For a given Q, a particular trapezoid P intersecting Q will be included in the count for the query on the data structure corresponding to only one of the minterms (because $B(P, Q)$ is written such that it is true iff exactly one of its minterms is true). Hence the counts from the queries of the different data structures can be simply added up. We apply Theorem 2 with $d = 2$ to get:

Lemma 2 *For any m satisfying $n \leq m \leq n^2$, and any constant $\epsilon > 0$, a set S of n trapezoids in the plane can be preprocessed in time $O(m^{1+\epsilon})$ into a data structure of size $O(m^{1+\epsilon})$ such that the trapezoids that intersect a query trapezoid Q can be counted in time $O(n^{1+\epsilon}/m^{1/2})$.* ∎

Now we turn to another problem. Let PS denote the union of the sets $\mathcal{P}$ of trapezoidal pairs corresponding to all polygons $P \in \mathcal{S}$. We wish to preprocess PS into a data structure such that given a query trapezoidal pair T_Q, the trapezoidal pairs in PS that have a conflict with T_Q can be counted efficiently. We consider two trapezoidal pairs $T_P = (P', P'')$ and $T_Q = (Q', Q'')$ and the three conditions enumerated in Definition 1. From the previous discussions, we know how to construct the boolean formula for the first condition of Definition 1. For the second and the third conditions of Definition 1, we construct the boolean formula for the corresponding intersection condition, negate it and again write it in DNF. From the AND of the three formulas thus constructed, we can construct a boolean formula $B_T(T_P, T_Q)$ which is true iff T_P and T_Q have a conflict. Moreover, we can write $B_T(T_P, T_Q)$ in DNF such that it is true iff exactly one of its minterms is true. We conclude:

Lemma 3 *Let PS be a set of $O(n)$ trapezoidal pairs in the plane. For any m satisfying $n \leq m \leq n^2$, and any constant $\epsilon > 0$, we can preprocess PS in time $O(m^{1+\epsilon})$ into a data structure of size $O(m^{1+\epsilon})$, such that the trapezoidal pairs in PS that conflict with a query trapezoidal pair T_Q can be counted in time $O(n^{1+\epsilon}/m^{1/2})$.* ∎

Finally we would like to preprocess a set $\mathcal{S}$ of r convex polygons with a total of n vertices (where r and the polygon sizes can depend on n), into a data structure such that given a query convex polygon Q, the polygons in $\mathcal{S}$ intersected by Q can be reported efficiently.

Theorem 3 *Let $\mathcal{S}$ be a set of convex polygons in the plane with a total of n vertices. For any m such that $n \leq m \leq n^2$, and any constant $\epsilon > 0$, $\mathcal{S}$ can be preprocessed in time $O(m^{1+\epsilon})$ into a data structure of size $O(m^{1+\epsilon})$ such that the polygons in $\mathcal{S}$ that intersect a query polygon Q can be counted in time $O(|Q| \cdot n^{1+\epsilon}/m^{1/2})$.*

Proof: We store all the trapezoidal pairs from all the polygons in $\mathcal{S}$ into an instance D of the data structure of Lemma 3. Given a query polygon Q, we decompose it into trapezoidal pairs and query D with each such pair. From Theorem 1, it follows that if Q intersects any $P \in \mathcal{S}$ there are unique trapezoidal pairs T_P of P and T_Q of Q such that T_P and T_Q are in conflict. This pair is detected while querying D with T_Q. The space and time bounds follow from Lemma 3. ∎

To count pairwise intersections of the polygons, we simply build an instance of the data structure of Theorem 3 and query with each polygon in turn.

Theorem 4 *Given a set $\mathcal{S}$ of convex polygons in the plane with a total of n vertices, the number of intersecting pairs of polygons in $\mathcal{S}$ can be counted in time $O(n^{4/3+\epsilon})$, for any constant $\epsilon > 0$.*

Proof: From Theorem 3, the building time is $O(m^{1+\epsilon})$. Adding the query time $O(|Q| \cdot n^{1+\epsilon}/m^{1/2})$, over all polygons Q, we get a total query time of $O(n^{2+\epsilon}/m^{1/2})$. Choosing $m = n^{4/3}$, we get an overall running time of $O(n^{4/3+\epsilon})$. ∎

4 The reporting problem: an alternative algorithm

The technique presented in the previous section can also be used to report the I intersecting pairs of polygons in time $O(n^{4/3+\epsilon} + I)$. In this section, we use a different approach to obtain a slightly faster algorithm.

We preprocess each polygon to remove any vertex between two adjacent edges supported by the same line. This can be achieved in $O(n)$ time. We also assume that no polygon has a vertical side, which can be achieved by appropriate rotation.

Now each polygon P has a leftmost vertex $\ell(P)$, a rightmost vertex $r(P)$ and can be uniquely decomposed into an upper chain $U(P)$ and a lower chain $L(P)$ at these vertices. We give a different characterization for the intersection of two convex polygons P and Q, which forms the basis of the reporting algorithm.

Observation 1 *For convex polygons P and Q, if $L(P)$ intersects $U(Q)$, then at most a total of eight edges of $L(P)$ and $U(Q)$ are involved in the intersection.*

The worst case of eight edges in Observation 1 occurs when $L(P)$ and $U(Q)$ intersect at exactly two points, each of which coincides with a vertex from $L(P)$ and a vertex from $U(Q)$.

Theorem 5 *Two convex polygons P and Q intersect iff*
(i) $\ell(P) \in Q$ or $\ell(Q) \in P$, or
(ii) $L(P)$ intersects $U(Q)$ or $U(P)$ intersects $L(Q)$.

Proof: If (i) or (ii) is satisfied, then P and Q intersect. To prove the converse, assume that P and Q intersect. Without loss of generality, assume that $\ell(P)$ is not to the right of $\ell(Q)$. Draw a vertical line L through $\ell(Q)$ and mark the two points, s and t, where L intersects the boundary of P (it is possible that $s = t$.) Note that s and t must exist since (i) the x-spans of P and Q must overlap if they intersect and (ii) $\ell(P)$ is not to the right of $\ell(Q)$. With a slight abuse of notation, we denote the y-coordinates of s, t and $\ell(Q)$ by the points themselves. Assume without loss of generality that $s \geq t$. Three cases can arise. (See Figure 2.) (i) If $s \geq \ell(Q) \geq t$, then $\ell(Q) \in P$; (ii) If $s \geq t \geq \ell(Q)$, then for P and Q to intersect, $L(P)$ must intersect $U(Q)$; (iii) if $\ell(Q) \geq s \geq t$, then for P and Q to intersect, $U(P)$ must intersect $L(Q)$. ■

Given r convex polygons with a total of n vertices, we break them up into upper and lower chains. We color the segments from the upper chains red and those from the lower chains blue. Then we run the red-blue segment intersection algorithm given in [Aga91, Theorem 5.10] to compute all intersections involving a red segment and a blue segment. This takes time $O(n^{4/3} \log^{O(1)} n + k)$, where k is the number of red-blue intersections. This time bound is also $O(n^{4/3} \log^{O(1)} n + I)$, since $k = O(I)$ by Observation 1. (Note that there can be intersections between red segments and, similarly, between blue segments; therefore we cannot use the algorithms given in [MS88, PS93].)

Next we take the r leftmost points of the polygons, triangulate the polygons into $O(n)$ triangles and solve the following problem: Given r points and $O(n)$ triangles, report all k' pairs (p, t) such that point p lies in triangle t. Using an algorithm given in [Aga91, Corollary 5.14], this problem can be solved in time $O(n^{4/3} \log^{O(1)} n + k')$, which is $O(n^{4/3} \log^{O(1)} n + I)$, since $k' = O(I)$. We conclude:

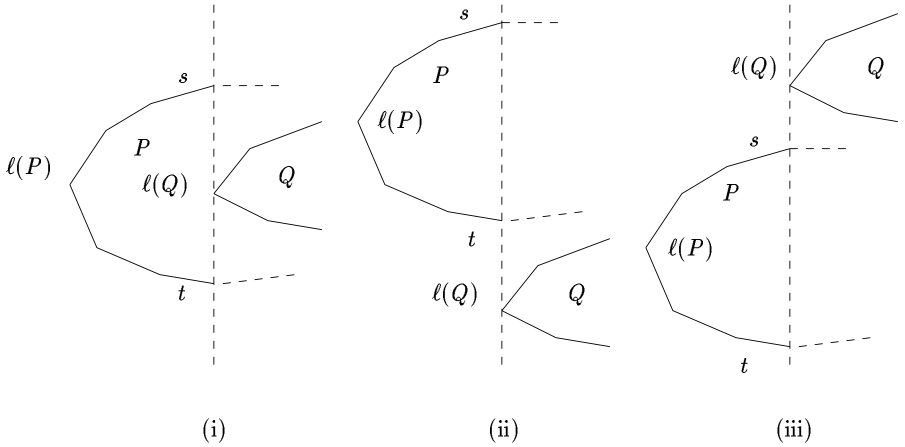

Figure 2: *Illustrating the proof of Theorem 5*

Theorem 6 *Given a set $\mathcal{S}$ of convex polygons in the plane, with a total of n vertices, the I pairs of polygons that intersect can be reported in time $O(n^{4/3}\log^{O(1)} n + I)$.* ∎

Remark 1 Let $\mathcal{A}$ and $\mathcal{B}$ be sets of convex polygons, with a total of n vertices, where no two polygons in the same set intersect. Using the above approach, we can report in time $O(n \log n + I)$ the I pairwise intersections between polygons in $\mathcal{A}$ and $\mathcal{B}$.

Note that Observation 1 and Theorem 5 still hold for $P \in \mathcal{A}$ and $Q \in \mathcal{B}$. We color red (resp. blue) the segments belonging to the upper (resp. lower) chains of polygons in $\mathcal{A}$ (resp. $\mathcal{B}$). Then we compute all k red-blue intersections using the algorithm of [MS88] or [PS93]. These algorithms are applicable since no two red segments and no two blue segments intersect. This takes time $O(n \log n + k) = O(n \log n + I)$, since $k = O(I)$ by Observation 1. We then repeat the above step with the lower chains in $\mathcal{A}$ and the upper chains in $\mathcal{B}$.

Next, we take the leftmost vertex of each polygon in $\mathcal{B}$ and determine which polygon (if any) in $\mathcal{A}$ contains it. Since the polygons in $\mathcal{A}$ are non-intersecting, this step can be done by building, in $O(n \log n)$ time, a point-location structure for the subdivision induced by the polygons in $\mathcal{A}$ and querying with the leftmost vertex of each polygon in $\mathcal{B}$. The total time for the queries is $O(|\mathcal{B}| \log n) = O(n \log n)$ time. It follows that the overall time for the algorithm is $O(n \log n + I)$.

5 Conclusions and open problems

In this paper, we have shown how to count and report efficiently the pairwise intersections between convex polygons in the plane, where the number of polygons and their sizes both depend on n—the total number of vertices. An interesting open problem is to design algorithms for the case when the polygons are arbitrary simple polygons.

References

[Aga91] P.K. Agarwal. *Intersection and decomposition algorithms for planar arrangements.* Cambridge University Press, New York, 1991.

[CD87] B. Chazelle and D.P. Dobkin. Intersection of convex objects in two and three dimensions. *Journal of the ACM*, 34(1):1–27, 1987.

[CE92] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM*, 39(1):1–54, 1992.

[CSW92] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, 1992.

[DE87] D.P. Dobkin and H. Edelsbrunner. Space searching for intersecting objects. *Journal of Algorithms*, 8:348–361, 1987.

[Mat92] J. Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8:315–334, 1992.

[MS88] H.G. Mairson and J. Stolfi. Reporting and counting intersections between two sets of line segments. *Theoretical Foundations of Computer Graphics and CAD*, NATO–ASI Series (R.A. Earnshaw, ed.) 307–325, 1988.

[PS93] L. Palazzi and J. Snoeyink. Counting and reporting red/blue segment intersections. *Proceedings of the Third Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science, Vol. 709, Springer-Verlag, Berlin, 530–540, 1993.

[vK92] M. van Kreveld. *New results on data structures in computational geometry.* PhD thesis, Department of Computer Science, University of Utrecht, Utrecht, the Netherlands, 1992.