Counting Convex k-gons in an Arrangement of Line Segments

Martin Fink* Neeraj Kumar* Subhash Suri*

Abstract

Let $\mathcal{A}(S)$ be the arrangement formed by a set of n line segments S in the plane. A subset of arrangement vertices $p_1, p_2, \ldots, p_k$ is called a *convex k-gon* of $\mathcal{A}(S)$ if $(p_1, p_2, \ldots, p_k)$ forms a convex polygon and each of its sides, namely, (p_i, p_{i+1}) is part of an input segment. We want to count the number of distinct convex k-gons in the arrangement $\mathcal{A}(S)$, of which there can be $\Theta(n^k)$ in the worst-case. We present an $O(n \log n + mn)$ time algorithm, for any fixed constant k, where m is the number of pairwise segment intersections. We can also *report* all the convex k-gons in time $O(n \log n + mn + |K|)$, where K is the output set. We also prove that the k-gon counting problem is 3SUM-hard for $k = 3$ and $k = 4$.

1 Introduction

We consider the problem of counting, and enumerating, all convex k-gons formed by the arrangement $\mathcal{A}(S)$ of a set S of n line segments in the plane. A set of vertices $p_1, p_2, \ldots, p_k$ of the arrangement $\mathcal{A}(S)$ is called a *convex k-gon* if $(p_1, p_2, \ldots, p_k)$ forms a convex polygon and each of its sides $(p_1, p_2), (p_2, p_3), \ldots, (p_{k-1}, p_k), (p_k, p_1)$ is part of an input segment. We note that such a k-gon is not necessarily a *face* of the arrangement, and in general there can be $\Theta(n^k)$ convex k-gons, for any fixed k. We are interested in the problem of *counting* these k-gons. That is, given a set of n line segments in the plane, how many convex k-gons exist in their arrangement?

Surprisingly, this natural-sounding problem appears not to have been explored in computational geometry. We are motivated by an application in computer vision where the case of counting, and enumerating, *convex quadrilaterals* arises. Specifically, the arrangement is the camera image representing linear boundaries of objects in the scene, and the goal is to estimate (the counting problem) the number of "rectangular" objects in the input scene, which may represent important features such as desks, door frames, walls etc. Due to the perspective transformation, the rectangles in the scene map to convex quadrilaterals in the image.

The computational problem then becomes the following: can we find all convex quadrilaterals in the n-segment arrangement in better than the naive $O(n^4)$ time? Counting convex k-gons is a natural generalization of this problem. We call this the *k-gon reporting* problem, which leads to a natural *counting* version of the problem, where we are just interested in counting the number of k-gons formed by the segments. Unlike the segment intersection problem [2, 3], in which the maximum number of intersecting line segments is $O(n^2)$, the number of combinatorially distinct k-gons in an n-segment arrangement can be $\Omega(n^k)$. See Fig. 1. Therefore, it is desirable to be able to count the k-gons in time much faster than the number of combinatorially distinct k-gons.

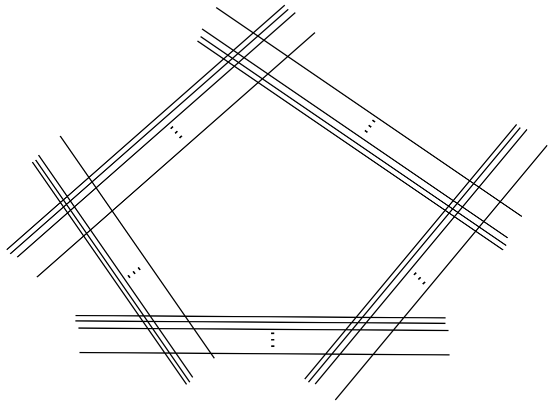


Figure 1: An arrangement of n segments with $\Omega(n^k)$ convex k-gons. Each of the $k = 5$ groups contains $\lfloor \frac{n}{k} \rfloor$ segments.

Our Contribution. We present a sweep-line algorithm for counting the number of k-gons in worst-case time $O(n \log n + mn)$, using $O(n^2)$ space, for any constant k, where m is the number of pairwise segment intersections. The algorithm works for non-constant values of k as well, but in that case takes $O(n \log n + mn^2)$ time and $O(n^3)$ space. In either case, the running time is independent of k.

By maintaining additional information during the counting algorithm, we can also recover all the k-gons, in worst-case time $O(n \log n + mn + |K|)$ time, using $O(mn + n^2)$ space, where K is the output.

Finally, we show that counting the number of triangles and the number of quadrilaterals are both 3SUM-hard, suggesting that a running time significantly better than $O(n^2)$ is unlikely.

*Department of Computer Science, University of California, Santa Barbara, {fink|neeraj|suri}@cs.ucsb.edu

Related Work. The problem of counting convex k-gons in a set of n points has been considered by several researchers [9, 8, 7]. The algorithm in [9] achieves a running time of $O(n^{k-2})$, which was then improved to $O(n^{\lceil k/2 \rceil})$ in [8]. This bound was improved significantly to $O(n^3)$ by Mitchell et al. [7] using dynamic programming. In [4], Eppstein et al. study the related problem of finding minimum area k-gons for point sets.

Some results are also known for the restricted problem of counting faces in line arrangements. For instance, every arrangement of lines (or pseudolines) in the plane results in $\Omega(n)$ triangular faces [5].

Another related problem is one of counting and reporting simple cycles of given length in a graph. In general, the time bound for counting the cycles is exponential in k [1], however for $k \leq 7$ the problem can be solved in $O(n^{2.376})$ time. These cycle counting results in graphs, however, do not solve our k-gon problem because of the *convexity* constraint. Indeed, an arrangement $\mathcal{A}(S)$ of segments can be easily viewed as a graph, whose vertices are the segments and whose edges correspond to pairs of intersecting segments. However, cycles in this graph are not necessarily convex polygons. An exception is the case of triangles which are always convex: every 3-cycle correspond to a triangle in the segment arrangement, but only for non-degenerate input, namely, no three segments intersecting in a common point.

2 Counting Convex k-gons

Let P be a convex k-gon in the arrangement $\mathcal{A}(S)$ formed by the n line segments of S.[1] Let L be a vertical line intersecting P. Since P is convex, L can only intersect two sides of P. The *span* of P with respect to L is the (ordered) pair of segments of P that intersect L. Although the number of k-gons can be exponential in k, the number of distinct spans is only quadratic.

Observation 1 *There are $O(n^2)$ distinct spans among all k-gons of $\mathcal{A}(S)$ with respect to a vertical line L.*

In other words, Observation 1 tells us that although there could be $\Omega(n^k)$ k-gons, at a given vertical line L, all k-gons intersecting L can be assigned to one of the $O(n^2)$ distinct segment pairs. This suggests the existence of a natural sweep line based approach for the counting problem. The key idea is to keep track of convex *open* polygons with up to k sides as we sweep a vertical line L across the arrangement. When sweeping over an intersection, some open polygons may become closed k-gons, which we must count, while other open polygons can be extended using the intersection vertex, and new open polygons start growing at the intersection. We start by fixing some notation.

[1]For the rest of the paper, we drop the qualifier "convex" and simply refer to P as a k-gon.

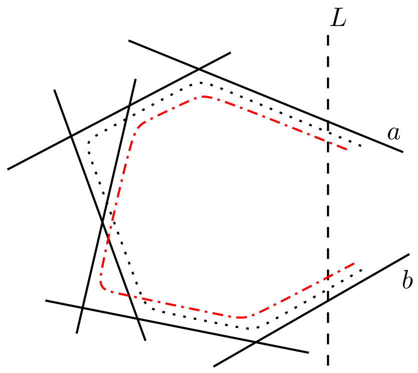


Figure 2: Open 5-gons: $\Sigma(a, b, 5)$ at a given sweep line L; two members of the set are shown by dotted lines.

Notation. Observe that when we are sweeping a vertical line L across the arrangement, at a given x-coordinate x_L, we may have come across two types of potential k-gons:

- *Closed k-gons:* these are the k-gons all of whose sides are to the left of the line L.
- *Open j-gons:* these are j-gons, for $j \leq k$, whose j sides lie (partially or fully) to the left of L. More precisely, L intersects the two open sides of these convex polygons and their remaining $j-2$ sides are to the left of L. See Fig. 2.

Observe that open j-gons are only potential candidates for closed k-gons; not all of them necessarily become k-gons.

At an x-coordinate x_L, we can now represent an open j-gon P by the triplet (a, b, j), where a and b are the segments forming the top and bottom sides of P at x_L and j is the number of sides we have seen so far. Note that $2 \leq j \leq k$ and this includes the sides of P formed by the segments a and b. The triplet (a, b, j) succinctly combines all open j-gons for which a and b are the open sides intersecting the line L; we let $\Sigma(a, b, j)$ denote the set of these open j-gons, and we let $\sigma(a, b, j) = |\Sigma(a, b, j)|$.

2.1 Algorithm

Our algorithm moves a sweep line L across the arrangement $\mathcal{A}(S)$ with the segment intersections as key event points. For the sake of simplicity, we assume no degeneracies for now, that is, every intersection involves exactly two line segments of S. (We will show how to handle degenerate cases later in this section.) We maintain an array of counters $\sigma(a, b, j)$ which keep track of all open j-gons whose span is (a, b) on line L, for all $2 \leq j \leq k$. A global counter keeps track of all the closed k-gons that have been encountered already. In the following, we explain these steps in detail.

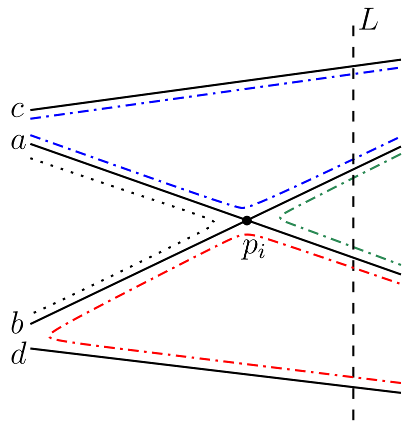


Figure 3: After the intersection of a and b, the sweep line L gets new open polygons for $\Sigma(b, a, 2)$, $\Sigma(c, b, j+1)$, and $\Sigma(a, d, j+1)$ shown in dash dotted; parent polygons in $\Sigma(c, a, j)$ and $\Sigma(b, d, j)$ are still active.

1. Set *count* $= 0$ and $\sigma(a, b, j) = 0$ for all segments a, b and $2 \leq j \leq k$.
2. Compute all m intersections of the n line segments using [2] and order them from left to right.
3. Process the intersections one by one from left to right, moving the sweep line L accordingly.

   When L contains the intersection of segments a and b (where to the left of L, a is above b), we perform the following updates. See Fig. 3.

   (a) Open k-gons of $\Sigma(a, b, k)$ become closed k-gons. We update the count:
   $$count \ \mathrel{+}= \ \sigma(a, b, k)$$
   (b) An open 2-gon of b and a begins at this intersection. We initialize the count:
   $$\sigma(b, a, 2) \quad = \quad 1$$
   (c) For all $2 \leq j < k$, each open j-gon with a as the lower side can be extended to an open $j+1$-gon with b as the lower side. Let S'_L be the set of segments intersecting sweep line L *above* the intersection point (a, b). We update the count:
   $$\forall c \in S'_L \quad \sigma(c, b, j+1) \ \mathrel{+}= \ \sigma(c, a, j)$$
   Similarly, for each $2 \leq j < k$, each open j-gon with b as the upper side can be extended to an open $j+1$-gon with a as the upper side. Let S''_L be the set of segments that intersect sweep line L *below* the intersection point (a, b).
   $$\forall d \in S''_L \quad \sigma(a, d, j+1) \ \mathrel{+}= \ \sigma(b, d, j)$$
4. Return *count*.

Correctness. Let S_L be the set of all segments that intersect the sweep line L. Let A_L be the set of all possible spans for the k-gons intersecting L. That is, $A_L = \{(a, b) \mid a, b \in S_L, \ a \text{ above } b \text{ on } L\}$. As the sweep line moves from left to right we maintain the following invariant:

> *count* is the total number of closed k-gons to the left of L; and $\sigma(a, b, j)$ is the number of open j-gons with span (a, b) on L, for all $(a, b) \in A_L$ and $2 \leq j \leq k$.

The invariant is trivially satisfied before processing the first intersection. For the general case, after moving the sweep line L over the intersection (a, b), the segments a and b switch their vertical order. Therefore,

1. A_L no longer includes the span (a, b).
2. *count* now includes the new k-gons that complete at the intersection (Step 3a); these correspond to the open k-gons counted by $\sigma(a, b, k)$.
3. A_L includes the new span (b, a). Right after the crossing, the open 2-gon formed by b and a is the only polygon with that span, covered by $\sigma(b, a, 2) = 1$.
4. Open polygons with span $(c, b) \in A_L$ can now also use the vertex (a, b). Any such open j-gon ($j \leq k$) must consist of the new intersection and an open $j-1$-gon with span (c, a) right before the intersection (compare Step 3c).
5. Analogously, open polygons with span $(a, d) \in A_L$ can now use vertex (a, b). Such a new open j-gon ($j \leq k$) consists of the new intersection and an open $j-1$-gon with span (b, d).

It is easy to see that the algorithm maintains the invariant after processing each intersection. Hence, when eventually the sweep line L is right of all intersections, *count* is the total number of k-gons.

Analysis. Computing and storing all m intersections takes $O((n+m) \log n)$ time and $O(m+n)$ space [2]. Since we perform $O(n)$ updates for each of the m events, the total running time for our algorithm is $O(n \log n + mn)$. The total space requirement is $O(n^2)$ since we store information for all pairs of segments that may intersect the sweep line.

Handling Degenerate Cases. We now show how to extend our algorithm so that it can also handle segment arrangement with degeneracies, that is, with three or more segments intersecting in a single point. (For parallel segments, we do not need to do anything special). For an intersection point p_i of a set S_i of more than two segments, we first update the number of closed k-gons for every pair of segments in S_i. For extending open j-gons ($2 \leq j < k$), we need to be a bit more careful. Since we do not want degenerate k-gons, we should only extend the j-gons which we have seen before the current intersection. More precisely, we compute the updates for every pair of segments in S_i, and apply them collectively. One way to achieve this is to process the updates in Step 3(b) and 3(c) in decreasing order of j as follows:

- For j in $k-1$ down to 3 perform updates in Step 3(c) for every segment pair in S_i.
- Perform updates in Step 3(b) for every segment pair in S_i.

Observe that these modifications do not affect the overall runtime since $m \in O(n^2)$ is the number of pairwise intersections.

3 Reporting Convex k-gons

We now turn to the problem of reporting all the k-gons in an arrangement of n line segments. We solve this problem by extending our algorithm for counting k-gons. The key idea is to keep track of how the values $\sigma(a, b, j)$ are updated as we move the sweep line across the arrangement, and to remember the values that contributed to the total number. Recall that the total number of k-gons formed by n segments can be $\Omega(n^k)$. Therefore, we would like the total running time to be linear in size of the output. We start by describing a *reporting graph* that will help us reconstruct all k-gons.

Reporting Graph. Our reporting graph is a *labeled* directed acyclic graph $G = (V, E, \mathcal{L})$. Its vertices represent the sets of polygons $\Sigma(a, b, j)$ and its edges keep track of how these sets grow. The function $\mathcal{L}\colon E \to \mathbb{N}$ assigns a label $\mathcal{L}(e)$ to each edge $e \in E$. The label is a timestamp and represents the intersection at which the edge was created.

To construct the digraph G, we extend the counting algorithm from Section 2.1 as follows:

1. For every pair (a, b) of segments and $2 \leq j \leq k$ add a vertex (a, b, j) to G.
2. Define $Q = \emptyset$ to be the set that keeps track of closed k-gons grouped by their rightmost vertex.
3. Refer to step 3 of the counting algorithm. Suppose the sweep line L is currently at the i^{th} intersection event (a, b). Recall that S'_L and S''_L are respectively the sets of segments that intersect L above and below the intersection point (a, b). We modify the reporting graph as follows; see Fig. 4 for an example.
   (a) If $\sigma(a, b, k) > 0$, insert (a, b, k) to Q.
   (b) For all values $2 \leq j < k$ and each segment $c \in S'_L$ with $\sigma(c, a, j) > 0$, create an edge $\big((c, b, j+1), (c, a, j)\big)$.
   (c) Similarly, for all $2 \leq j < k$ and a segment $d \in S''_L$ with $\sigma(b, d, j) > 0$, create an edge $\big((a, d, j+1), (b, d, j)\big)$.

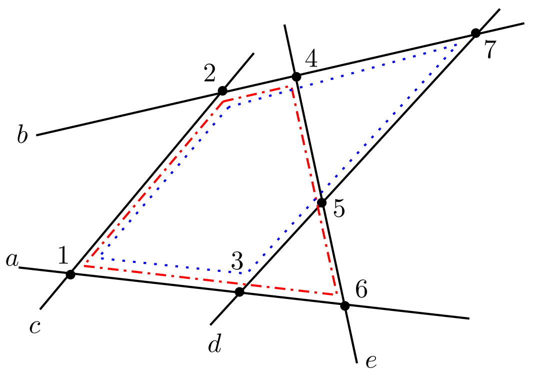
   Label each edge e created for this intersection event with the timestamp $\mathcal{L}(e) = i$.

The reporting graph G has the following properties.

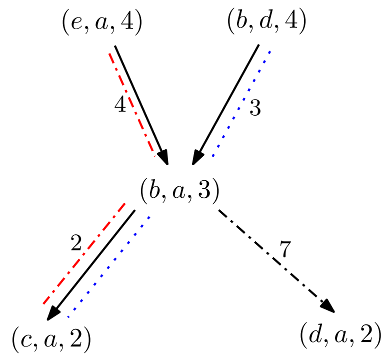
- G has $O(n^2)$ vertices and $O(mn)$ edges. The vertices of the form (a, b, k) have in-degree zero (sources) and vertices of the form $(a, b, 2)$ have out-degree zero (sinks).
- An edge of G represent the extension of an open j-gon into a $j+1$-gon ($2 \leq j < k$), with the intersection point (a, b) being the newly added corner. As a result, we get two new sets of $j+1$-gons: one with b as the lower side and another with a as the upper side.
- There is exactly one label on every edge since two segments can only intersect once.
- Each complete k-gon corresponds to a path that starts at a vertex in Q (a source in G) and ends at a sink of G. Since G is acyclic, every such path has exactly $k-2$ edges. The intersection points corresponding to these $k-2$ edges along with the two intersection points for the source and sink will be the k vertices of the output k-gon.

Enumerating all k-gons. With the reporting graph G, enumerating all k-gons seems pretty straightforward. We can simply start at vertices in Q one by one and recursively explore all distinct paths to sinks. However, there is one small caveat. Since the segments may continue to grow further after we close a k-gon, it is possible that a vertex v of G gets an additional successor w' after it got a predecessor u; see Fig. 4. Observe that in such a case, the path $(u \to v \to w' \to \cdots)$ does not correspond to a valid k-gon since the corresponding vertices are not ordered chronologically.

In order to fix this we can use the timestamps of the edges: we only recurse using the edges whose timestamp is smaller than that of the parent edge. Because of our construction, we are guaranteed to find at least one such edge. Moreover, since the edges are added in the order of their timestamps, we can stop at the first edge for which the timestamp is higher than the parent value. This way we spend no extra time on objects that are not a member of our output set. Consequently, we get a running time of $O(n \log n + mn)$ for constructing the G, and $O(|K|)$ time for reporting the k-gons that form our



(a) Arrangement of five line segments. Numbers 1 through 7 indicate intersections from left to right events. The two valid 4-gons formed by the segments are shown by red and blue dotted lines.



(b) Reporting graph. The edge shown as dash-dotted was added after its predecessors and therefore does not contribute to a valid 4-gon. The other two valid dotted paths to the sink $(c, a, 2)$ represent the closed 4-gons with vertices $(1, 2, 4, 6)$ and $(1, 2, 3, 7)$.

Figure 4: An arrangement and the corresponding reporting graph.

output set K. Since G has $O(n^2)$ vertices and $O(mn)$ edges, the total space requirement is $O(n^2 + mn)$.

4 3SUM-Hardness

In this section, we show that counting the number of triangles in an arrangement of straight-line segments is at least as hard as the 3SUM problem. Since it is widely believed that 3SUM cannot be solved in $o(n^2)$ time, this also holds for the problem of counting triangles.

Theorem 1 *Counting the number of triangles in an arrangement of straight-line segments is 3SUM-hard.*

Proof. We reduce the problem Point-on-3-Lines to counting triangles. Gajentaan and Overmars showed that Point-on-3-Lines is as hard as 3SUM [6]. In Point-on-3-Lines one has to decide whether a given arrangement of straight-lines contains a point in which at least three lines intersect. It is easy to see that the problem remains 3SUM-hard even if no pair of lines is parallel. We transform such an arrangement of lines (with no parallel pairs) to an arrangement of straight-line segments by shortening all lines to segments. We must ensure that all crossings of lines are preserved as crossings of the corresponding segments. To this end, we determine the bounding box of the line arrangement, which is not hard to achieve in $O(n \log n)$ time.

Consider the resulting arrangement. Since it contains all crossings, each triple of segments forms a triangle unless either the three segments intersect in a single point, or (ii) two of the segments are parallel—which cannot happen since the input lines did not contain parallel pairs. Therefore, the arrangement of segments contains $\binom{n}{3}$ triangles if and only if there is no point in which three or more lines intersect.

We have seen that we can check the existence of a point lying on at least three lines by counting the triangles in the arrangement of segments. Furthermore, transforming the instance and determining the number α needed only constant time. Hence, an $o(n^2)$-time algorithm for counting triangles in segment arrangements implies an $o(n^2)$-time algorithm for Point-on-3-Lines. □

We can use almost the same 3SUM-hardness proof for convex quadrilaterals rather than triangles. Observe that any arrangement of four straight-lines (without parallel pairs) forms exactly one quadrilateral face unless three of the lines meet in a point. Hence, the number of quadrilaterals is $\binom{n}{4}$ if and only if there is no triple of lines meeting in a point.

Theorem 2 *Counting the number of convex quadrilaterals in an arrangement of straight-line segments is 3SUM-hard.*

Unfortunately, for larger values of k, e.g., $k = 5$ the hardness reduction does not seem easy to adjust. The problem is that not every set of five straight lines forms a 5-gon, even if they are in general position.

5 Conclusion

We introduced the problem of counting and reporting k-gons in an arrangement of line segments, and presented an $O(n \log n + mn)$ time algorithm for counting all the k-gons, for any fixed constant k, where m is the number of intersecting segment pairs. Our algorithm for reporting all the k-gons runs in time $O(n \log n + mn + |K|)$, where K is the output set. We also prove that the k-gon counting problem is 3SUM-hard for $k = 3$ and $k = 4$.

References

[1] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17:209–223, 1997.

[2] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, 100(9):643–647, 1979.

[3] B. Chazelle. Reporting and counting segment intersections. *Journal of Computer and System Sciences*, 32(2):156–182, 1986.

[4] D. Eppstein, M. Overmars, G. Rote, and G. Woeginger. Finding minimum area k-gons. *Discrete & Computational Geometry*, 7(1):45–58, 1992.

[5] S. Felsner and K. Krieger. Triangles in euclidean arrangements. In *Graph-Theoretic Concepts in Computer Science*, pages 137–148. Springer, 1998.

[6] A. Gajentaan and M. H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165 – 185, 1995.

[7] J. S. Mitchell, G. Rote, G. Sundaram, and G. Woeginger. Counting convex polygons in planar point sets. *Information Processing Letters*, 56(1):45–49, 1995.

[8] G. Rote and G. Woeginger. Counting convex k-gons in planar point sets. *Information Processing Letters*, 41(4):191–194, 1992.

[9] G. Rote, G. Woeginger, Z. Binhai, and W. Zhengyan. Counting k-subsets and convex k-gons in the plane. *Information Processing Letters*, 38(3):149–151, 1991.