Large $k$-gons in a 1.5D Terrain

Vahideh Keikha

Institute of Computer Science, the Czech Academy of Sciences, Department of Applied Mathematics, Charles University, Prague, Czech Republic, keikha@cs.cas.cz

Abstract. Given is a 1.5D terrain $T$, i.e., an $x$-monotone polygonal chain in $\mathbb{R}^2$. For a given $2 \leq k \leq n$, our objective is to approximate the largest area or perimeter convex polygon of exactly or at most $k$ vertices inside $T$. For a constant $k > 3$, we design an FPTAS that efficiently approximates the largest convex polygons with at most $k$ vertices, within a factor $(1 - \epsilon)$. For the case where $k = 2$, we design an $O(n)$ time exact algorithm for computing the longest line segment in $T$, and for $k = 3$, we design an $O(n \log n)$ time exact algorithm for computing the largest-perimeter triangle that lies within $T$.

1 Introduction

Recently, much attention is given to problems of the following form: Given is a set $P$ of complexity $n$ and a parameter $k \leq n$. The objective is computing a subset of size $k$ of $P$ that optimizes some cost function among all choices.

A well-studied variant is “potato peeling” [11] also known as “convex skull” [20], which assumes the input set $P$ is an arbitrary simple polygon of $n$ vertices, and seeks the inscribed object having the largest area or perimeter in $P$. It is shown that computing an inscribed convex polygon having the largest area in $P$ is solvable in $O(n^7)$ time [6]. In the same paper, an $O(n^6)$ time algorithm is also presented for computing the largest-perimeter inscribed convex polygon. There also exists a 4-approximation algorithm for the largest-area contained convex polygon in $P$ with running time $O(n \log n)$ [13]. In the same paper, the authors also discussed several approximations and randomized algorithms for the other variants of the problem, including the largest-area rectangle, ellipse, and the $\delta$-fat triangle in $P$, where an $\delta$-fat triangle is a triangle at which all three angles are at least $\delta$. In [4], computing the largest inscribed convex polygon is studied for both the area and the perimeter measures, and the authors introduced a randomized $(1 - \epsilon)$-approximation algorithm that gives this result with probability at least $2/3$, and runs in $O(n(\log n)^2 + \frac{1}{\epsilon^3} \log n + \frac{1}{\epsilon^4})$ time.

The exact algorithm for computing the largest area triangle that is inscribed in $P$ is also studied in [19]. The authors presented an $O(n^4)$ time algorithm that also works for the perimeter measure keeping the same running time.

In [9], an $O(n \log^2 n)$ time algorithm is presented for computing the largest area axis-aligned rectangle in a simple polygon, and it is shown that this problem has an $\Omega(n \log n)$ lower bound. See [15] and the references therein for the results on the potato peeling problem in higher dimensions.
Motivation There are several applications for different variants of the potato peeling problems. One of the most important applications is collision detection\footnote{also known as interference detection or contact determination.} with the aim of finding the biggest object of bounded complexity which could be moved within an environment without being collapsed\cite{16}. Computing the biggest static or moving object in an environment is considered a major computational bottleneck. The other application is geometric shape approximation which is the act of approximating one geometric shape with another simpler object, see, e.g.,\cite{1} which gives several approximations of a convex polygon with rectangles, circles, and polygons with fewer edges. See also\cite{18} for a survey on different applications and approaches to these problems.

Related Work The longest line segment inside a simple polygon of $n$ vertices can be computed in $O(n^{1.99})$ time\cite{7} and can be approximated within a factor $(1-\epsilon)$ in $O(n\log^2 n)$ time\cite{14}. As mentioned above, the best-known algorithms for computing the largest area and the largest perimeter triangle inside a simple polygon take $O(n^4)$ time\cite{19}. However, these problems seem to be easier if the input is a terrain. It is shown that the largest-area triangle that is inscribed in a 1.5D terrain $\mathcal{T}$ of $n$ vertices can be computed in $O(n^2)$ time\cite{10}. This algorithm recently improved to $O(n\log n)$ time\cite{5}. In\cite{10}, a 2-approximation algorithm and an FPTAS with running times $O(n\log n)$ and $O(\frac{n}{\varepsilon}\log^2 n)$, respectively, are designed for computing the largest-area triangle. The largest area axis-aligned rectangle inside $\mathcal{T}$ can also be computed in $O(n)$ time\cite{8} since any terrain can be considered as a vertically separated horizontally convex polygon\cite{8}.

Contribution In the following, we formally define our problems.

\textbf{Problem 1.} Given is a 1.5D terrain $\mathcal{T}$, compute the largest perimeter triangle in $\mathcal{T}$.

\textbf{Problem 2.} Given is a 1.5D terrain $\mathcal{T}$ and a constant integer $2 \leq k < n$, compute a largest perimeter/area polygon of at most $k$ vertices in $\mathcal{T}$.
See Fig. 1 for an illustration. For the case where \( k = 2 \), we seek the longest line segment inside the terrain and refer to it as the diameter of the terrain. We show that the diameter of the terrain can be computed in \( O(n) \) time. For \( k = 3 \), we design an \( O(n \log n) \) time exact algorithm for the largest perimeter contained triangle in \( T \), that matches the result for the area measure \( [3] \). We then give a \((1 - \epsilon)\)-approximation for Problem 2, i.e., computing the largest perimeter polygon of at most \( k > 3 \) vertices. Our results are successfully extended to the area measure. See Table 1 for a summary of the new and known results.

### Preliminaries

Let \( T \) be a 1.5D terrain of \( n \) vertices, and let \( B \) be the base of the terrain. Let \( A_k^* \) denote a largest perimeter polygon of at most \( k \) vertices inside \( T \). W.l.o.g, suppose \( B \) is always a horizontal segment, otherwise one can transform the coordinate system to satisfy this. Let \( Q \) be any convex polygon inside \( T \). The sides of \( Q \) which have exactly one endpoint on \( B \) are called the legs of \( Q \), and the side which has two vertices at \( B \) is called the base of \( Q \). For a point \( p \in T \), \( x(p) \) denotes the \( x \)-coordinate of \( p \).

### Case \( k = 3 \): Largest Perimeter Triangle

In contrast to the largest area triangle inside a terrain, a largest perimeter triangle in a terrain does not necessarily have a side coincident with \( B \). See Fig. 2a.

We show that there exists a \( A_3^* \) which has a vertex or a side coincident with \( B \). We consider these two cases independently, and report the best solution.

**Lemma 1.** There is a largest perimeter triangle inside the terrain which has its base or one of its vertices coincident with the base of the terrain.

**Proof.** If \( A_3^* \) does not have any vertex on the base of the terrain, since the terrain is monotone, we can translate \( A_3^* \) downward until it has a vertex or a side on the base of the terrain. See Fig. 2b. \( \square \)
**A₃⁺ having its base on B** We first seek a $A₃⁺$ which has its base coincident with $B$. We will use the following lemmas in some of our proofs.

**Lemma 2.** Let the base of $A₃⁺$ coincides with $B$, and let $α$ and $β$ be the internal angles of the legs of $A₃⁺$. Then $0 < α, β ≤ \frac{π}{2}$.

*Proof.* Suppose, by contradiction, there is a solution to $A₃⁺$ at which (w.l.o.g) $α > \frac{π}{2}$. Let $A, B$ and $C$ denote the sides of $A₃⁺$ in counter clockwise (CCW), where $A$ and $C$ are the legs of $A₃⁺$, as illustrated in Fig. 3. With a slight abuse of the notation, let each side also denote its length. Consider the right triangle $A'$ which is constructed by drawing the perpendicular to the base of the terrain through the top vertex of $A₃⁺$. Let $a, b$ and $C$ denote the sides of $A'$, where $C$ denotes the hypotenuse of this triangle and $a, b$ denote the adjacent sides of the right angle. Let $b' = b - B$. From the triangle inequality, we have $a + b' > A$. By adding a $B$ to either side of the inequality we have $a + b' + B > A + B$, hence $a + b > A + B$, which means the perimeter of $A'$ is larger than the perimeter of $A₃⁺$, a contradiction. Consequently, we have $0 < α, β ≤ \frac{π}{2}$. \(\square\)

**Lemma 3.** A largest perimeter triangle which has its base on $B$, either (i) has its topmost vertex on the boundary of the terrain, in which case each of its legs is passing through at least a vertex of the terrain, or (ii) each of its sides is supported by two vertices of the terrain.

*Proof.* The sides or the vertices of $A₃⁺$ must be blocked by the boundary of the terrain, otherwise, we still can improve the perimeter.

In the following, we prove for restricting any further possible enlargement, $A₃⁺$ must have either its top vertex on the boundary of the terrain in which case each of its legs is passing through a vertex, or it has its legs passing through two vertices of the terrain. In case (i), for the sake of contradiction, suppose $A₃⁺$ has its top vertex on the terrain, but the legs of $A₃⁺$ are not supported by any...
vertices of the terrain. But then, while we keep the top vertex fixed, we improve
the perimeter by moving the vertices on the base of the terrain in opposite
directions to get further from each other, until each leg encounters a vertex of
the terrain which restricts its further movement. This gives a contradiction with
the optimality of $\Lambda^*_3$.

In case (ii), suppose there is a solution to $\Lambda^*_3$ where a leg $\ell$ of $\Lambda^*_3$ is supported
by only one vertex (otherwise, while the top vertex is fixed, similar to above,
we can improve the perimeter by moving the vertices of $\Lambda^*_3$ on the base on
opposite directions until each leg touches at least one vertex of the terrain).
From Lemma 2 the interior angles of $\Lambda^*_3 = ABC$ are at most $\frac{\pi}{2}$. Now suppose,
by contradiction, at least one of the legs of $\Lambda^*_3$, say the right one, is supported
by only one vertex. Let $l_r$ denote the right leg, and let $l_l$ denote the left leg.
Observe that if $l_r$ is supported by a convex vertex, the top vertex of $\Lambda^*_3$ lies at
the boundary of the terrain. Hence let $l_r$ be supported by a reflex vertex $O$. We
consider the changes of the perimeter of $\Lambda^*_3$ as a function of $l_r$, while we rotate
$l_r$ around $O$ in CW or CCW directions.

$l_r$ can be rotated around $O$ until it encounters another vertex at which $l_r$ is
blocked and further rotation is not possible.

First, suppose the point $O$ lies on the same side as $l_l$, with reference to the
supporting line through the bisector of $l_l$ and $B$ (the case where the point $O$
lies on the other side of the bisector is symmetric); see Fig. 1. Consider two
triangles $Ab_2c_1$ and $Ab_0c_3$ by a slight rotation of $l_r$ in CW and CCW direction,
respectively, where $b_0B = Bb_2$. Note that such triangles always exist even for a
very small value of $b_0B$, as $l_r$ is pivoted at only one vertex. Suppose by rotating
$l_r$, around $O$, in the CW direction, the perimeter of the resulting triangle is
decreased. Hence, $Ab_2c_1 < ABC$ (1). We show that further rotations of $l_r$ in the
CCW direction would increase the perimeter of $ABC$.

Consider $b_3c_0$ as a further rotation of $l_r$ in CW direction, at which $c_0$ lies at
$B$, and $b_3$ lies at the supporting line of $AB$. Let $Oc_0$ is vertical to $B$.

From (1) we have $Ab_0 + b_0B + Bb_2 + b_2c_1 + Ac_0 + c_0c_1 < Ab_0 + b_0B + BC +
c_1C + c_0c_1 + Ac_0$. Since $b_0B = Bb_2$, we have $Bb_2 + b_2c_1 < BC + c_1C$.

First let $BC$ is vertical to $l_l$. We show that if we rotate $l_r$ in CCW, the
resulting triangle is bigger than $BC$ and has an internal angle bigger than $\frac{\pi}{2}$,
which cannot happen according to Lemma 2. If $BC$ is vertical to $l_l$, the rotation
of $l_r$ in the CW direction is allowed, but it should decrease the perimeter of
the resulting triangle. But then, we show that the assumption of by rotating
$l_r$ in CW direction the perimeter of the resulting triangle is decreased gives a
contradiction with the optimality of $ABC$. 

Fig. 4: Illustration of Lemma 3.
If \( O \) lies on the bisector of the angle \( A \), the triangles \( ABO \) and \( AOC_0 \) have the same perimeter, while \( O_{C_0} \) is vertical to \( B \). The same argument also holds for \( OBb_0 \) and \( O_{C_0}C_1 \). But if \( O \) lies to the left of the bisector of angle \( A \), directed from \( A \) to \( O \), the triangles are still similar, but \( ABO < AOC_0 \) and \( b_0BO < O_{C_0}C_1 \), since the length of one side of each of these triangles, is decreased. Consider a line parallel to \( AB \) through \( c_1 \). Let \( x_1 \) be the intersection of this parallel line and \( BC \).

Since \( O \) lies to the left of the bisector of angle \( A \), directed from \( A \) to \( O \), and the similarity of the triangles \( OBb_0 \) and \( O_{C_0}C_1 \), the fact that \( O_{C_0}x_1 \) falls inside \( O_{C_0}C \), and \( c_1x_1 = b_0B \), we have \( b_0B < c_1C \), which implies \( b_2c_1 < BC \), because \( Bb_2 + b_2c_1 < BC + c_1C \) and \( Ab_2C_1 < ABC \).

Again using the fact that \( O \) lies to the left of the bisector of \( l_l \) and \( l_r \), with the same argument as above we have \( C_{C_3} > b_0B \). But then, since \( b_0O > BO \), \( O_{C_0} > OB \) (since \( O \) lies to the left of the bisector) and \( O_{C_3} > O_{C_0} \) (since \( O_{C_0}C_3 \) is a right triangle with \( O_{C_0} \) as hypotenuse), we have \( b_0c_3 > BC \), which implies \( Ab_0c_3 > ABC \), while \( Ab_0c_3 \) has also one internal angle bigger than \( \frac{\pi}{2} \), contradiction with the optimality of \( ABC \) and Lemma 2.

Now suppose \( BC \) is not vertical to \( l_l \). But then the rotation of \( l_r \) is allowed in both CW and CCW directions. The proof is as above, using the fact that the side next to the largest angle in a triangle has the largest length. Hence, we again reach \( Ab_0c_3 > ABC \), which gives a contradiction with the optimality of \( ABC \).

The proof in the case where \( O \) lies on the bisector of the angle \( A \) is much simpler. Observe that among all possible rotations of \( l_r \), the resulting maximum perimeter triangle is one of the two triangles at which further rotations of \( l_r \) is not possible because either \( l_r \) is blocked because it encounters another reflex vertex or \( l_r \) reaches an endpoint of \( l_l \). But in both of these cases, \( l_r \) must be supported by two vertices, where one of them is \( O \). The Lemma follows.

The relation between the triangles refers to their perimeter.

**Algorithm** In [5,10] it is shown that a largest-area triangle inside \( T \) which has a base on the base of the terrain has the properties which we proved in Lemma 3. The presented algorithms in [5,10] enumerate all maximal triangles with these properties. Hence, we use the same algorithm, but we just report the largest-perimeter triangle instead of the largest-area triangle. For the case (i) of Lemma 3 of [5] at which the top vertex lies at the boundary of the terrain, we apply Lemma 2 of [5] to treat this case in \( O(n \log n) \) time. For the case (ii) we use the following results of [5].

In Section 4 in [5], it is shown that each segment which is connecting two vertices of the terrain and may realize a side of \( A'_{3} \), would have two prolongations inside the terrain, at which in the prolongation in one direction the segment intersects the base, and in the other direction, the segment intersects the upper boundary of the terrain. Let \( S \) denote all such segments. In Lemma 4 in [5], it is shown that the set of candidates of the left and the right sides of \( A'_{3} \) which consists of the prolongations of the segments in \( S \), in both directions, can be
computed in $O(n)$ time, as it uses the Guibas et al. [12] idea on how to compute the shortest path trees inside a simple polygon in $O(n)$ time. Let $L$ denote the segments of $S$ with positive slopes, and let $R$ denote the segments of $S$ with negative slopes. In the case where the prolongations of one element from $L$ and one element from $R$ lies inside the terrain, a triangle would be constructed which realizes a candidate of $A^*_3$. The set of all such intersection points (Lemma 5 in [5]) realize the candidates of the top vertex of $A^*_3$, which can be computed in $O(n \log n)$ time (Lemma 4,5 in [5]). Hence, we also construct all such triangles and report the one having the largest perimeter.

Consequently, a largest perimeter triangle contained in a 1.5D terrain of $n$ vertices which has a side on the base of the terrain can be computed in $O(n \log n)$ time. In the following, we consider the case where a largest perimeter triangle has a vertex on the base of the terrain.

**$A^*_3$ having a vertex on $B$** A largest perimeter triangle with a vertex on the base of the terrain has this property that the side opposite to the vertex on $B$ has a maximal length since otherwise the perimeter can be improved. In Lemma 4 we show that a maximal length line segment in $B$ must be supported by two vertices of $T$. Here we seek the maximal length line segments for which they are supported by two vertices of the terrain, but they do not necessarily have a prolongation which intersects the base of the terrain. Such maximal line segments are still supported by two vertices, and are a subset of the edges of the shortest path tree inside a simple polygon, that can be computed in $O(n)$ time, as discussed in detail in Section 3 in [5]. Thus we already know the two vertices for each potential largest perimeter triangle, and it remains to compute the optimal placement of the third vertex which lies on $B$. According to Lemma 7 the third vertex chooses an extreme placement on the line segment which it lies on, i.e., the leftmost or the rightmost possible locations on $B$, at which the triangle does not intersect the terrain boundary. Hence, for each maximal line segment, two maximal perimeter triangle needs to be considered, and we choose the largest one. Let $a,b$ denote the vertices of a maximal line segment $\ell$, and let $c \in B$ be the third vertex of the maximal triangle having $\ell$ as a side. W.l.o.g., let $a$ be the left endpoint of $\ell$. Then obviously either $c$ is a vertex of $B$ or $ac$ is blocked by a reflex vertex $r$ (to stop further improvement of the perimeter), or it is passing through the terrain edge containing the vertex $a$. Handling the first and the last cases are straightforward. In the case where there is a reflex vertex $r$ that blocks $ac$, we need to find $r$ precisely. But then, again, the line segment connecting $r$ to the leftmost vertex of the terrain edge containing $a$ is a subset of $L$, as it is supported by two terrain vertices and also has a vertex on $B$. Hence, all such reflex vertices can be computed in $O(n)$ time. The intersection of $B$ and the prolongation of the line segment passing through $r$ and $a$ can be computed in constant time. Consequently, a largest perimeter triangle which has a vertex on the base of the terrain can be computed in $O(n)$ time.

**Theorem 1.** The largest perimeter triangle contained in a 1.5D terrain of $n$ vertices can be computed in $O(n \log n)$ time.
Case $k = 2$: Diameter of the Terrain

We define the diameter of a 1.5D terrain $\mathcal{T}$ as the longest line segment within $\mathcal{T}$ and denote it by $l^*$. In the following, we discuss how we can compute $l^*$ in $O(n)$ time. We design an approximation algorithm for Problem 2 based on $l^*$.

**Lemma 4.** $l^*$ is either (1) supported by two convex vertices, (2) supported by a reflex vertex and a convex vertex, or (3) supported by two reflex vertices of $\mathcal{T}$.

**Proof.** The longest line segment in the terrain must be blocked by the vertices of the terrain, otherwise, we still can increase its length by rotation or translation. Case (2) have two different combinatorial structures, where $l^*$ passes through an edge adjacent to a reflex vertex and ends up at a convex vertex, or $l^*$ originated from a convex vertex and passes through a reflex vertex. See Fig. 5. If the diameter is not a diagonal between two convex vertices, it must be blocked by at least one reflex vertex; if not we can improve the length by translation and/or rotation until the segment encounters a vertex to be blocked from further improvement on the lengths. Let $r$ be the reflex vertex that blocks $l^*$. If we rotate $l^*$ around $r$, in at least one of the CW or CCW directions the length of $l^*$ would be increased, until $l^*$ encounters another reflex vertex or a convex vertex that further rotation is not possible. □

For computing $l^*$, the existence of an $O(n^2 \log n)$ time algorithm by considering any pair of vertices and ray shooting queries is obvious. In the following, we discuss the improvement of this running time.

**Theorem 2.** There is a solution to the diameter of the terrain which has a vertex on $\mathcal{B}$ and can be computed in $O(n)$ time.

**Proof.** From Lemma 4 we know any maximal length line segment $l^*$ in $\mathcal{T}$ is supported by two vertices of $\mathcal{T}$. If $l^*$ does not have any vertex on $\mathcal{B}$, we can translate it downward until one of its vertices lies at $\mathcal{B}$, and then we either can improve the length or keep it unchanged if its blocked from further changes. Hence, the set $S$ discussed in Section 2 is a superset for all maximal length line segments in the terrain. In Lemma 4 in [5], it is shown that the set $S$ which consists of the prolongations of all the maximal length line segments passing through two terrain vertices can be computed in $O(n)$ time. □
3 An FPTAS for $A_k^*$

We first discuss $k = 3$. Any triangle which has the diameter of the terrain as a side gives a 3-approximation to $A_3^*$. We use this simple observation to design our FPTAS for Problems 1, 2. We first explain the algorithm for $k = 3$ and discuss the extension afterwards.

Let $A$ be a 3-approximation of $A_3^*$. Also, let $|A|$ denote the perimeter of $A$. Consider a grid of big cells of side length $6|A|$. Let the bottom left corner of a big cell in the grid lies at the leftmost vertex of $T$, and let $(0, 0)$ denote its coordinates. Consider three copies of this big cell with the bottom left corners at coordinates $(3|A|, 0)$, $(0, 3|A|)$ and $(3|A|, 3|A|)$, respectively. Since $T$ is monotone, the diameter of the terrain at least equals the length of $B$, and the longest vertical line segment inside the terrain is smaller than the diameter of the terrain, the union of the four big cells covers the terrain entirely.

Lemma 5. $A_3^*$ is contained in one of the big cells entirely.

Proof. The side length of a big cell, say $C$ is $6|A|$. Since any triangle has at most one obtuse angle, in the worst case, two consecutive edges of $A$ fulfills at most a $1/9$ fraction of the width of $C$. Hence, any feasible solution to $A$ entirely lies within $C$. On the other hand, $A$ gives a $1/3$ approximation for $A_3^*$, and $A_3^*$ does not have any obtuse angle, but any of its sides is at most trice of a side of $A$. Hence, $A_3^*$ also entirely lies within a big cell. \[\square\]

From the construction of the grid, each edge of $T$ intersects at most all the four big cells. We decompose each big cell by $O\left(\frac{1}{\epsilon}\right)$ finer cells of side length $\epsilon|P|$, for a given $\epsilon > 0$. Let $X$ denote the set of finer cells. Observe that the total complexity of the intersection of $X$ and $T$ is in $O(n)$ since each edge of $T$ is intersecting with at most four big cells. So computing the set $X' \subseteq X$ of finer cells that are intersecting with $T$ takes $O(n)$ time. We consider any of the four big cells independently.

Let $c_i, c_j$ and $c_k$ be any triple of finer cells in $X'$, all in a specific big cell $C$. For two line segments $s_i \subset c_i$ and $s_j \subset c_j$, we consider the intersection of the visibility polygons by assuming an edge guard at $s_i$ and another one at $s_j$. However, we do not need to compute the visibility polygons explicitly. Let $V^{ji}_q$ denote the ranges on the corresponding side of $c_i$ that are visible to $s_j$. We are interested in determining whether there are three line segments in $T \cap C$ whose endpoints lie at the intersection of the pairwise visibility ranges of the segments $s_i^p$, $s_j^q$ and $s_k^r$. This happens when the intersection of the pairwise visibility ranges of $s_i^p$, $s_j^q$ and $s_k^r$ have a non-empty intersection, i.e., $V^{ij}_p \cap V^{kj}_q \neq \emptyset$, $V^{ij}_q \cap V^{kj}_r \neq \emptyset$ and $V^{ij}_r \cap V^{kj}_p \neq \emptyset$, as illustrated in Fig. 6.

3.1 Preparation for Using the Data Structures

The visible parts of a given segment $s$ from any other segment, as well as the visible parts of every segment from $s$, can be computed in linear time and space.
Fig. 6: Computing the visibility ranges on a triple of finer cells that are introducing a candidate triangle for $\Lambda^*_3$ (its sides denoted by dash). The visibility ranges are shown in yellow. This triple of segments has a non-empty intersection for the pairwise visibility ranges of the segments. Here, there also exists another combinatorially different triangle (in dash-dotted) on these three finer cells.

using the shortest path algorithm [12]. This imposes an extra $O(n^2)$ time in the running time of the algorithm. Next, we try to improve this running time using the technique presented in [14] for computing the visibility between intervals on the edges of a simple polygon. Here we recall the main technique of the presented algorithm in [14] for the self-completeness of our algorithm.

Let $c_i, c_j$ and $c_k$ be three finer cells in a big cell. For any three distinct sides of the cells $c_i, c_j$ and $c_k$, we first make the corresponding visibility ranges, i.e., the maximal length line segments on a side of a finer cell that lies inside the terrain entirely. We refer to each visibility range as an interval. Let $s^i_1, \ldots, s^i_{n_i}$, $s^j_1, \ldots, s^j_{n_j}$ and $s^k_1, \ldots, s^k_{n_k}$ denote the sequence of the intervals on $c_i, c_j$ and $c_k$, respectively, such that they lie within $\mathcal{F}$.

Let $V^i \cap c_j$ be the constructed visibility range for the interval $s^i_p$, by considering an edge guard on it, with respect to the intervals on $c_j$. If one or both endpoints of $V^i \cap c_j$ lie in any visibility ranges of $c_j$, $s^i_p$ sees any of these intervals. This can be determined in logarithmic time.
We can represent \( V_{ij}^{pq} \) by the pair of indices corresponding to the intervals that are contained in it. Then we can do range searching queries and 1D windowing queries to determine the intervals which have a non-empty intersection between their visibility ranges.

For each pair of \( s_p^i \) and \( s_q^j \), for efficiently determining whether there is any segment that lies within \( T \) entirely, and with endpoints at these intervals, we use a combination of two data structures.

For two points \( p \in s_p^i \) and \( q \in s_q^j \), the line segment \( pq \) must lie in the visibility range of both \( s_p^i \) and \( s_q^j \). The visibility range of any of \( s_p^i \) and \( s_q^j \) can be computed in \( O(\log n) \) time by performing shortest path queries on preprocessed connected components achieved in \( O(n) \) time \[13\], and checking whether they can see each other or not can be done at the same time by simple visibility queries inside simple polygons. We first make a 1D range tree at the indices of the intervals in \( c_i \) in \( O(n_i \log n_i) \) time, and of the height \( O(\log n_i) \). Then we make an interval tree at each node of the constructed range tree. Thus our data structure has a size \( O(n_i \log n_i) \) and takes \( O(n_i \log^2 n_i) \) time at each level of the range tree. The queries are is there any interval \( s_p^i \) in \( c_i \) which is visible from an interval \( s_q^j \) in \( c_j \). This can be answered in \( O(\log n_i) \) time. Performing \( n_j \) queries like this gives us the running time \( O((n_i + n_j) \log^2 n_i) \) for each pair of cells in a big cell. Observe that \( n_i, n_j \in O(n) \).

**Lemma 6.** For each pair of finer cells, computing all pairwise visible intervals takes \( O(n \log^2 n) \) time \[14\].

There are \( O(\epsilon^{-2}) \) cells, and we consider any triple of cells. For any triple of intervals that the intersection of the pairwise visibility regions is non-empty, we compute the largest perimeter triangle as below.

### 3.2 A Largest perimeter triangle on three intervals

For any three points on the boundary of terrain that define a triangle that lies inside the terrain entirely, it is necessary and sufficient that they are pairwise visible. Now we have three intervals such that they are pairwise visible, and we seek the largest perimeter triangle having its vertices on these intervals.

Any triple \( s_p^i, s_q^j, s_r^k \) of segments that may contain the vertices of \( A^*_3 \) have a horizontal or vertical direction. Suppose the locations of two points \( p, q \) on their segments, say \( s_p^i, s_q^j \) are fixed, and the position of \( r \in s_r^k \) is a variable. The function that describes the longest perimeter changes as a symmetric hyperbolic function with a unique minimum. Thus there always exists one direction in which a point can be moved such that the size has not decreased. In the following, we prove it formally.

**Lemma 7.** For any triple \( s_p^i, s_q^j \) and \( s_r^k \) of segments, the largest perimeter triangle \( A^*_3 \) with one vertex at each segment would have its vertices at the endpoints of the segments.
Fig. 7: $E_{ij}$ is the constructed ellipse at $A^*_3 = \triangle pqt$. If there is at least one line segment $s^k_r$ that does not contribute an endpoint to $A^*_3$, at least one of the two ellipses which are constructed at $p,q$ and one of the endpoints of $s^k_r$ contains $E_{ij}$ entirely. (Note that two of the ellipses coincide if $s^k_r$ is parallel to $pq$).

Proof. Suppose, by contradiction, that $A^*_3 = \triangle pqr$ has at least one vertex that is not at an endpoint. W.l.o.g, let $p \in s^k_p$ and $q \in s^k_q$ be the vertices of $A^*_3$ that lie at the endpoints. Consider the ellipse $E_{ij}$ which is passing through $r$ and with $p$ and $q$ as its foci, see Fig. 7. Consider the two (possibly one) ellipses that are passing through the endpoints of the line segment $s^k_r$ with the same foci as $E_{ij}$. Then since $r$ is an interior point of $s^k_r$, $E_{ij}$ entirely lies within one of the constructed ellipses. Because of this, moving $r$ along $s^k_r$ to at least one of the endpoints of $s^k_r$ would strictly increase the perimeter of $A^*_3$; contradiction. \( \Box \)

Thus, a $(1-\epsilon)$-approximation of the largest perimeter triangle in $\mathcal{T}$ can be computed in $O(\epsilon^{-6}n \log^2 n)$ time, however, this is not efficient as the problem has an $O(n \log n)$ time exact algorithm. But this algorithm can be generalized for computing largest area/perimeter polygons of $k > 3$ vertices.

4 Extension to Convex Polygons of at most $k$ Vertices

The extension of the presented algorithm to convex polygons of $k > 3$ vertices is straightforward. We consider any set of $k$ intervals in $k$ finer cells instead of three intervals, where all the $k$ intervals lie inside the terrain and are pairwise visible.

The non-empty intersection condition is necessary again to find a largest perimeter convex polygon of $k$ vertices that entirely lies inside $\mathcal{T}$, however, in the algorithm, we may report a convex polygon of less than $k$ vertices if it has a larger area/perimeter than any convex $k$-gon with vertices at the selected $k$ intervals. It remained to find the optimal placements of the points on each set of $k$ segments that admits a non-empty set for the intersection of the pairwise visibility ranges.

4.1 Perimeter Measure

Löffler and van Kreveld have shown that if there is a set of $O(n)$ squares or line segments of two directions, the maximum perimeter convex polygon that is inter-
secting all of them and has at most one vertex at any of them can be computed in $O(n^{10})$ time [17], and the optimal solution always chooses the endpoints of the segments. In our problem, the intervals have horizontal or vertical directions since they lie at the sides of the finer cells. However, in the algorithm of [17], it is necessary for the segments to be disjoint. On the other hand, we do not have the restriction of choosing exactly one point from each interval. But we can adjust our intervals to satisfy the input conditions of the algorithm presented in [17].

We need to allow an interval to contribute to both of its endpoints. Hence, for each interval, we send two tiny length sub-intervals to the algorithm, such that each of which is contained in the original interval and contains exactly one of the endpoints of the original interval. This way, a horizontal line segment in our problem is transformed into two tiny length horizontal line segments. With a careful selection of the length of the tiny intervals which should be a fraction of $\epsilon$ (to keep the approximation factor unchanged), the tiny intervals would not intersect each other and the selection of each of their endpoints does not change the combinatorial structure of the resulting polygon. Hence, from now on, we work with the tiny intervals instead of the original intervals.

We note that we shortened the intervals because we know from [17] that only the endpoints of the tiny length intervals contribute to the optimal solution. Also notice that we cannot simply consider the endpoints (degenerate segments) since we miss the pairwise visibility information by only considering the points.

Any convex at most $k$-gon $A$ which has the diameter of the terrain as an edge gives a $1/k$ approximation for the largest perimeter convex $k$-gon in the terrain. Hence, we would have four big cells of side length $2k|A|$ to cover the terrain. So we use the presented dynamic program algorithm in [17]. We just need to ensure the next vertex from the current tiny interval is visible from the current vertex. A natural correction is we keep a refined list of tiny intervals for each of the tiny intervals, where only visible tiny intervals are included in the refined list of each tiny interval. This increases the space complexity by $O(n)$, but the running time remains unchanged.

Hence, we can use the presented algorithm in [17] for computing a largest perimeter polygon of at most $k$ vertices with vertices on $k$ intervals (see [17] for details). Notice that the constructed polygon has at most $2k$ vertices, but the adjustment for having at most $k$ vertices is straightforward. Consequently, we achieve an $O(k^{10}\epsilon^{-2k}n\log^2 n)$ time algorithm that approximates the largest perimeter convex polygon of at most $k$ vertices within a factor $(1-\epsilon)$.

**Theorem 3.** One can compute a $(1-\epsilon)$-approximation of the largest perimeter $k$-gon in a 1.5D terrain $T$ of $n$ vertices in $O(k^{10}\epsilon^{-2k}n\log^2 n)$ time and $O(n^2)$ space.

**Proof.** Computing the largest perimeter polygon on $k$ intervals takes $O(k^{10})$ time. There are $O(\epsilon^{-2k})$ sets of $k$ intervals in $O(\epsilon^{-2})$ finer cells. Considering a pairwise visibility query takes $O(n\log^2 n)$ time. Hence, we have $O(k^{10}\epsilon^{-2k}n\log^2 n)$

---

2 The extension of the Lemma 5 to this case follows from the convexity of the $k$-gon and is straightforward.
computations and we remember the largest computed polygon which has at most $k$ vertices. The space complexity of the algorithm of [17] is $O(n)$. Hence, in our problem, the space complexity is $O(n^2)$.

\section{4.2 Area Measure}

In this section, we show that our algorithm is extendable to the area measure. We first prove that if two triangles have a $(1 - \epsilon)$-approximation for the ratio of their corresponding sides, the ratio of their area is also within a factor $(1 - \epsilon)$.

\begin{lemma}
Suppose the ratio between the length of corresponding edges of two triangles $\triangle abc$ and $\triangle a^*b^*c^*$ is $(1 - \epsilon)$. Then the area of $abc$ is within a factor $(1 - \epsilon)$ of the area of $a^*b^*c^*$.
\end{lemma}

\begin{proof}
From the statement of the lemma, $\frac{ab}{a^*b^*} = \frac{bc}{b^*c^*} = \frac{ac}{a^*c^*}$. This implies that $\triangle abc$ and $\triangle a^*b^*c^*$ are similar. A basic theorem in the similarity of the triangles states that for two similar triangles with similarity ratio $(1 - \epsilon)$, the corresponding heights is also within a factor $(1 - \epsilon)$, and the ratio of the areas equals $(1 - \epsilon)^2$, which is still a $(1 - \epsilon)$ approximation.
\end{proof}

\begin{algorithm}
Similar to the largest-perimeter at most $k$-gon, we should find the optimal placements of the points on each set of $k$ intervals that admits a non-empty set for the intersection of all the pairwise visibility ranges. In [17], an $O(n^7)$ time algorithm is designed to compute the largest-area convex hull with vertices given from a set of disjoint axis-aligned squares of arbitrary size, where each square contributes at most one vertex. The authors proved that only two corners of each square can contribute a vertex to the optimal solution, and they reduced their problem to the problem of finding the largest-area convex polygon on a set of line segments of four different orientations; horizontal, vertical, and diagonal, where the diagonal orientations come from the diagonal of the squares. In our problem, we only have two orientations for the intervals, so we can use the same algorithm to compute the largest area convex hull of at most $2k$ vertices, where the vertices lie at horizontal or vertical tiny intervals, as discussed in the previous section. This gives us an $O(k^7\epsilon^{-2k}n \log^2 n)$ time algorithm for having a $(1 - \epsilon)$-approximation algorithm.

\begin{theorem}
One can compute a $(1 - \epsilon)$-approximation of the largest area $k$-gon in a terrain $T$ of $n$ vertices in $O(k^7\epsilon^{-2k}n \log^2 n)$ time and $O(n^2)$ space.
\end{theorem}

\section{5 Discussion}

We discussed several algorithms for computing a largest area or perimeter polygon of bounded complexity $k$. For $k = 2, 3$, we discussed exact algorithms for the perimeter measure. Furthermore, for $k = 3$ our running time is the same as the running time of the algorithm presented in [17] for computing the largest area triangle. The efficiency of the algorithms for $k = 3$ in both the area and the perimeter measures remained open. Also, the problem of finding a largest area/perimeter convex polygon of at most $k$ vertices for general $k$ remained open.
References

1. H. Alt, J. Blömer, and H. Wagener. Approximation of convex polygons. In *International Colloquium on Automata, Languages, and Programming*, pages 703–716. Springer, 1990.
2. J. E. Boyce, D. P. Dobkin, R. L. Drysdale III, and L. J. Guibas. Finding extremal polygons. *SIAM Journal on Computing*, 14(1):134–147, 1985.
3. S. Cabello, O. Cheong, C. Knauer, and L. Schlipf. Finding largest rectangles in convex polygons. *Computational Geometry*, 51:67–74, 2016.
4. S. Cabello, J. Cibulka, J. Kyncl, M. Saumell, and P. Valtr. Peeling potatoes near-optimally in near-linear time. *SIAM J. Comput.*, 46(5):1574–1602, 2017.
5. S. Cabello, A. K. Das, S. Das, and J. Mukherjee. Finding a largest-area triangle in a terrain in near-linear time. In A. Lubiw and M. Salavatipour, editors, *Algorithms and Data Structures*, pages 258–270, Cham, 2021. Springer.
6. J. Chang and C. Yap. A polynomial solution for the potato-peeling problem. *Discret. Comput. Geom.*, 1:155–182, 1986.
7. B. Chazelle and M. Sharir. An algorithm for generalized point location and its applications. *Journal of symbolic computation*, 10(3-4):281–309, 1990.
8. K. Daniels, V. Milenkovic, and D. Roth. Finding the largest rectangle in several classes of polygons. *Harvard Computer Science Group, Technical Report TR-22-95.*, 1995.
9. K. L. Daniels, V. J. Milenkovic, and D. Roth. Finding the largest area axis-parallel rectangle in a polygon. *Comput. Geom.*, 7:125–148, 1997.
10. A. K. Das, S. Das, and J. Mukherjee. Largest triangle inside a terrain. *Theoretical Computer Science*, 858:90–99, 2021.
11. J. E. Goodman. On the largest convex polygon contained in a non-convex n-gon, or how to peel a potato. *Geometriae Dedicata*, 11(1):99–106, 1981.
12. L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1):209–233, 1987.
13. L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.
14. O. Hall-Holt, M. J. Katz, P. Kumar, J. S. Mitchell, and A. Sityon. Finding large sticks and potatoes in polygons. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, volume 6, pages 474–483, 2006.
15. N. Karmakar and A. Biswas. Construction of an approximate 3d orthogonal convex skull. In *International Workshop on Computational Topology in Image Context*, pages 180–192. Springer, 2016.
16. M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA conference on mathematics of surfaces*, volume 1, pages 602–608. Citeseer, 1998.
17. M. Löffler and M. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–260, 2010.
18. D. P. Luebke. A developer’s survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.
19. E. A. Melissaratos and D. L. Souvaine. On solving geometric optimization problems using shortest paths. In *Proceedings of the Sixth Annual Symposium on Computational Geometry, Berkeley*, pages 350–359. ACM, 1990.
20. T. C. Woo. The convex skull problem. Technical report, Technical report TR 86-31, Department of Industrial and Operations, 1986.