JAMDA

J. Appl. Math. & Data. Anal.

Available online at https://jamda.abru.ac.ir/
Journal of Applied Mathematics & Data Analytics
Vol. 2, No. 2, 2026, 1–9
Research Article

A Time-Space Trade-Off for Computing the Geodesic Center of a Simple Polygon

Pardis Kavand*
Department of Mathematics, Faculty of Basic Sciences,
Ayatollah Boroujerdi University, Boroujerd, Iran

Ali Mohades
Department of Mathematics and Computer Science,
Amirkabir University of Technology, Tehran, Iran

Mohammad Reza Kazemi
Department of Mathematics and Computer Science,
Amirkabir University of Technology, Tehran, Iran

Abstract

Abstract: We study the problem of computing the geodesic center of a simple polygon when the available workspace is limited. For an n-vertex simple polygon, we present a time-space trade-off algorithm that finds the geodesic center in $O\left(T(n,s)\log^2 n + \frac{n^2}{s}\log n\right)$ expected time and uses $O(s)$ additional words of space, where $s \in \Omega(\log n) \cap O(n)$ and $T(n,s)$ denotes the time required to construct, in depth-first order, the shortest path tree of a given point inside a simple polygon using $O(s)$ extra space. Applying the time-space trade-off of Oh and Ahn (*Algorithmica*, 2019) for constructing a shortest path tree yields an expected running time of $O\left(\frac{n^2}{s}\log^3 n\right)$.
Keywords: Computational geometry, memory-constrained algorithms, constrained geodesic center, geodesic center.
2020 Mathematics Subject Classification: 68U05; 68Q25

1 Introduction

The geodesic center problem asks for a point inside a given simple polygon P with n vertices that minimizes the maximum geodesic distance to any point of P. Pollack et al. showed that the problem can be solved in $O(n \log n)$ time [1]. This remained the best known bound for many years, until Ahn et al. gave a linear-time algorithm [2]. Both algorithms use $O(n)$ space.

*Corresponding author: p.kavand@abru.ac.ir

We study the problem when the available workspace is sublinear. Our memory-constrained model is as follows. The input is stored in read-only memory and supports random access to each input item. In addition to the input, an algorithm may use $O(s)$ words of read-write workspace, where $s \in \{1, \ldots, n\}$. Each word contains $\Theta(\log n)$ bits. Because intermediate data structures may exceed the available workspace, they are not stored explicitly and are recomputed when needed. This model is commonly called the s-workspace model. Several computational-geometry problems have been studied in this setting, either by designing new algorithms or by adapting algorithms for unrestricted memory. For a survey of memory-constrained algorithms in computational geometry, see [3].

One problem studied in this model is computing a shortest path between two points inside a simple polygon. Har-Peled gave an s-workspace algorithm for computing a shortest path between two points in an n-vertex simple polygon in $O(n^2/s + n \log s \log^4(n/s))$ expected time [4]. Aronov et al. presented an algorithm for constructing the shortest path tree of a given point inside an n-vertex simple polygon. Their algorithm runs in

$$O\left(\frac{n^2}{s} \log n + n \log s \log^5(n/s)\right)$$

expected time and uses $O(s)$ words of space for $s \leq n$ [5]. Oh and Ahn subsequently improved these bounds [6]. Their s-workspace algorithm computes a shortest path in $O(n^2/s)$ deterministic time, while their algorithm for constructing a shortest path tree has expected running time $O((n^2/s) \log n)$.

We present an s-workspace algorithm that computes the geodesic center of a simple polygon P with n vertices in

$$O\left(T(n, s) \log^2 n + \frac{n^2}{s} \log n\right)$$

expected time, assuming $s \in \Omega(\log n) \cap O(n)$. Here, $T(n, s)$ is the time required to construct, in depth-first order, the shortest path tree of a given point inside P using $O(s)$ words of extra space. Thus, the running time decreases as the available workspace increases. Using the shortest path tree algorithm of Oh and Ahn [6], the expected running time becomes $O((n^2/s) \log^3 n)$.

The remainder of the paper is organized as follows. Section 2 introduces the required notation and preliminaries. Section 3 studies the geodesic center problem. We first consider the geodesic center constrained to a chord of the polygon in Subsection 3.1 and then solve the unconstrained problem in Subsection 3.2. Section 4 concludes the paper.

2 Preliminaries and notation

Let P be a simple polygon with n vertices, and let p and q be two points in P. A shortest path from p to q in P is a polygonal path of minimum length connecting p and q and contained entirely in P. The geodesic distance between p and q, denoted by $d_P(p, q)$, is the length of a shortest path between them. We use $\vec{v}(p, q)$ to denote the unit vector at p directed along the first segment of a shortest path from p to q in P. A *geodesic farthest neighbor* of p is a point of P having maximum geodesic distance from p. It is well known that every geodesic farthest neighbor is a vertex of P [1]. A point may have more than one geodesic farthest neighbor; we denote the set of all such vertices by $F_P(p)$.

The shortest path tree of p in P is the tree rooted at p whose nodes represent the vertices of P, with an edge between two nodes u and v whenever the segment uv occurs on a shortest path $\pi_P(p, u)$ or $\pi_P(p, v)$. We denote this tree by $ST_P(p)$.

A *chord* of P is a line segment whose endpoints lie on the boundary of P and whose interior does not intersect the exterior of P.

Throughout the paper, $T(n, s)$ denotes the time needed to construct, in depth-first order, the shortest path tree of a given point inside P using $O(s)$ words of extra space.

3 Geodesic centers

In this section, we present an s-workspace algorithm that computes the geodesic center of a simple polygon P with n vertices in

$$O\left(T(n, s)\log^2 n + \frac{n^2}{s}\log n\right)$$

expected time, where $s \in \Omega(\log n) \cap O(n)$. Our approach follows the framework of Pollack et al. [1], who gave an $O(n \log n)$-time, $O(n)$-space algorithm for the geodesic center problem. We adapt their approach to the memory-constrained setting.

We first outline the method. Given a chord of P, we determine on which side of the chord the geodesic center lies by computing the geodesic center constrained to that chord. Repeating this procedure in a binary-search framework yields a triangle inside P that contains the geodesic center.

For a center constrained to either a chord or a triangle, we first identify the shortest path tree of the unknown center. Once this tree is known, the geodesic center problem reduces to computing a smallest circle that encloses a linear number of circles. We show how to perform these steps in the memory-constrained model. Subsection 3.1 treats the constrained problem, and Subsection 3.2 treats the unconstrained problem.

3.1 Constrained geodesic centers

Let P be a simple polygon and let d be a chord of P. We seek the point $c_d \in d$ that minimizes the maximum geodesic distance to points of P. We give a memory-constrained algorithm that runs in $O(T(n, s) \log n)$ expected time and uses $O(s)$ words of space for $s \in \Omega(\log n) \cap O(n)$.

Algorithm. Let a and b be the endpoints of d. Consider the edges of $ST_P(a)$ and $ST_P(b)$ and the intersections of the supporting lines of these edges with d, whether or not the extended edges themselves meet the boundary of P. Let X be the sequence of these intersection points in their order along d, and let $m = |X|$. The points of X partition d into $m + 1$ subchords on each of which the combinatorial structure of the shortest path tree is unchanged. In particular, for any such subchord I and any vertex p of P, there is a reflex vertex r_p such that the shortest path from any $x \in I$ to p has the form

$$xr_p \cup \pi_P(r_p, p),$$

where xr_p denotes the line segment from x to r_p.

By performing a binary search over X, we identify the subchord containing the constrained geodesic center (Lemma 3.1). Within this subchord, the constrained geodesic center problem is equivalent to finding the smallest circle whose center lies on the subchord and that encloses the

circles centered at r_p with radii $d_P(r_p, p)$, one for each vertex p of P. We solve this problem by adapting Megiddo's prune-and-search method [7] to sublinear workspace (Lemma 3.2).

Lemma 3.1. *The shortest path tree of the unknown constrained geodesic center can be determined in $O(T(n, s) \log n)$ expected time using $O(s)$ additional words of space.*

Proof. It suffices to find the subchord containing the constrained geodesic center. We perform a binary search on X. After the ith step, we maintain a subchord $I_i \subseteq d$ that contains the constrained geodesic center and contains at most $(3/4)^i m$ points of X. Initially, $I_0 = d$.

While generating the edges of the two shortest path trees, we sample points of X until we obtain an approximate median. We need not store X: whenever its points are required, they are regenerated by the shortest path tree procedure. The expected number of trials needed to obtain an approximate median is constant. Thus, after an expected constant number of trials, we obtain $x_1 \in X$ that divides I_0 into two subchords, each containing at most $3m/4$ points of X.

After constructing $ST_P(x_1)$, we compute the geodesic farthest neighbors of x_1. Because the constrained geodesic center problem is a convex optimization problem [1], if the directions

$$\{\vec{v}(x_1, f) : f \in F_P(x_1)\}$$

occur on both sides of the line through x_1 perpendicular to d, then x_1 is the constrained geodesic center. Otherwise, all descent information points toward the side of d that contains the constrained geodesic center, and we retain that subchord as I_1.

Repeating the argument, at the beginning of step $i+1$ we have a subchord I_i containing at most $(3/4)^i m$ points of X and containing the constrained geodesic center. We regenerate the relevant intersection points from $ST_P(a)$ and $ST_P(b)$, find an approximate median x_i among the points in I_i, and determine the side containing the center from the geodesic farthest neighbors of x_i. Hence I_{i+1} contains at most $(3/4)^{i+1} m$ points of X.

Since each shortest path tree has linear size, after $O(\log m) = O(\log n)$ steps we obtain a subchord containing the constrained geodesic center and no point of X in its interior. The shortest path tree is therefore combinatorially unchanged throughout this subchord, so the shortest path tree of the constrained geodesic center is the same as that of any point in the subchord. Each step invokes the shortest path tree algorithm an expected constant number of times, giving the stated expected running time and workspace bound. ■

Using Lemma 3.1, let I^* be a subchord of d containing the constrained geodesic center such that the shortest path tree is unchanged on I^*. Let R^* be the set of children of the root in $ST_P(x)$ for any $x \in I^*$. For each $r \in R^*$, let f_r be the maximum geodesic distance from r to a vertex in the subtree of $ST_P(x)$ rooted at r. The constrained geodesic center problem on I^* is equivalent to finding the smallest circle, with center constrained to I^*, that encloses all circles $C(r, f_r)$ for $r \in R^*$.

We adapt Megiddo's prune-and-search algorithm for the smallest enclosing circle problem [7] by using a tournament-tree viewpoint. First consider a line ℓ and a set S of n points in the plane. The goal is to find the smallest circle containing S whose center is constrained to ℓ. Megiddo's method proceeds in $O(\log n)$ rounds. In each round, the active points are paired, the perpendicular bisectors of the pairs are intersected with ℓ, and a median intersection is used to determine on which side the optimal center lies. For each pair whose bisector crosses the discarded side, the point that cannot be extremal on the retained side is pruned. Thus a constant fraction of the active points is removed in every round.

The process can be viewed as a tournament tree with possible draws. The leaves represent all points of S. The ith round constructs level $i+1$ from level i. When one member of a pair is pruned, it is the loser and the other member is the winner; only the winner advances. If neither member can be pruned, the pair is a draw and both advance. Consequently, the number of active objects decreases geometrically.

We now return to the circles $C(r, f_r)$, $r \in R^*$, with the center constrained to I^*. Because the workspace is limited, lower tournament levels cannot all be stored explicitly; instead, they are regenerated as needed while the information defining the retained intervals is maintained.

Suppose we are constructing the second level. Choose any $x \in I^*$ and construct $ST_P(x)$. Let

$$C_1 = C(r_1, f_1), \ldots, C_{n'} = C(r_{n'}, f_{n'})$$

be the sequence of circles in the order in which they are generated, and pair C_{2j-1} with C_{2j}. For a pair $C(c_1, r_1)$ and $C(c_2, r_2)$, define its *covering bisector* as the locus of points z satisfying

$$\|z - c_1\| + r_1 = \|z - c_2\| + r_2.$$

This locus is a branch of a hyperbola, with the usual line or empty-set degeneracies; in particular, its intersection with the supporting line of I^* has at most two relevant boundary points. If the covering bisector does not meet I^*, one member of the pair is dominated throughout I^* and may be discarded.

Let $\mathcal{T}$ be the sequence of all intersection points between I^* and the covering bisectors of the pairs that meet I^*. By computing an approximate median of $\mathcal{T}$, as in Lemma 3.1, we first obtain a subinterval of I^* that contains at least one quarter of these intersection points but not the solution. Applying the same idea to the relevant intersections on the side containing the solution yields a subinterval $J_1 \subseteq I^*$ containing the solution and not intersected by at least $1/16$ of the covering bisectors. For every pair whose covering bisector does not intersect J_1, one circle is dominated throughout J_1 and can be discarded; for the remaining pairs, both circles remain active.

For $i \geq 1$, let A_i denote the procedure that computes an interval J_i at level i such that J_i contains the solution and at least $1/16$ of the covering bisectors of the active circles do not intersect J_i. Let B_i denote the procedure that generates level $i+1$ of the tournament tree. The procedures A_1 and B_1 are described above. For $i \geq 2$, A_i computes J_i by running B_{i-1} a constant number of times; once computed, J_i is retained and used by B_i. Thus, when B_i is subsequently executed, it generates the active pairs from level i by a single run of B_{i-1} and uses the already stored interval J_i to decide which member of each pair, if any, is dominated. Maintaining the intervals $J_1, \ldots, J_i$ requires only $O(i)$ words.

Let τ_i and s_i be the time and additional workspace needed to generate level $i+1$, assuming the intervals $J_1, \ldots, J_i$ have already been computed and retained. Since the number of active objects decreases geometrically, for constants $c > 0$ and $\sigma \geq 32/31$,

$$\tau_i = \tau_{i-1} + O\Big(\frac{n}{\sigma^{i-1}}\Big) = \tau_1 + O(n), \qquad s_i = s_{i-1} + c = ic.$$

The separate computation of J_i takes

$$c'\tau_{i-1} + O\Big(\frac{n}{\sigma^{i-1}}\Big) = c'\tau_1 + O(n)$$

time for a constant c'. Hence each level can be processed in $O(\tau_1 + n)$ expected time using only $O(i)$ additional words beyond the shortest path tree workspace.

After $O(\log n)$ levels, only a constant number of active circles remain, so the constrained geodesic center can be obtained directly.

Lemma 3.2. *Let P be a simple polygon with n vertices and let d be a chord of P. The geodesic center constrained to d can be computed in $O(T(n,s)\log n)$ expected time using $O(s)$ extra space, where $s \in \Omega(\log n) \cap O(n)$.*

Proof. A constant fraction of the active objects is removed at each level of the tournament tree, so the tree has $O(\log n)$ levels. By the discussion above, the total expected running time is

$$O(\log n)\big((c'+1)\tau_1 + O(n)\big).$$

The procedure B_1 constructs a shortest path tree only a constant number of times, so $\tau_1 = O(T(n,s))$. Since constructing a shortest path tree has output size $\Omega(n)$, the additive $O(n)$ term is absorbed by $O(T(n,s))$. Moreover, $i = O(\log n)$ and $s_i = ic$, so the tournament structure uses only $O(\log n)$ additional words. Because $s \in \Omega(\log n)$, this is absorbed by the available $O(s)$ workspace. The correctness follows from the prune-and-search dominance argument of Megiddo's method. ■

3.2 Geodesic centers

We now compute the geodesic center of a simple polygon P in the memory-constrained setting. We first find a chord that splits P into two subpolygons of almost equal size (Lemma 3.3). We determine on which side of the chord the geodesic center lies and recurse on that subpolygon. Eventually we obtain either the geodesic center itself or a triangle, whose sides are chords of P, that contains the center (Lemma 3.4). We then find a subregion of the triangle that contains the geodesic center and on which the shortest path tree does not change. Once the shortest path tree is known, the problem again reduces to finding the smallest circle enclosing a linear family of circles.

Oh and Ahn [6] give an $O(n^2/s)$-time, s-workspace method for subdividing an n-vertex simple polygon into $O(\min\{n/s, s\})$ subpolygons, each of complexity $O(\max\{n/s, s\})$. For our purpose, it is sufficient to split P into two nearly equal subpolygons. This gives the following lemma.

Lemma 3.3. *In $O(n^2/s)$ time and using $O(s)$ words of workspace, one can find a chord that decomposes P into two subpolygons, each containing at most $2n/3$ vertices up to the boundary vertices shared by the two parts.*

Proof. By Chazelle's polygon-cutting result [8], there exists a vertical chord ℓ that splits P into two subpolygons of size at most $2n/3$ each. Sliding such a chord horizontally while maintaining it as a chord allows it to be made incident to a vertex. Hence, it suffices to examine vertical chords through vertices. Lemma 1 of Oh and Ahn [6] explains how to generate the vertical chords through all vertices of a simple polygon in $O(n^2/s)$ time using $O(s)$ extra space. Therefore, the result follows. ■

Lemma 3.4. *Let P be a simple polygon with n vertices. In*

$$O\left(T(n,s)\log^2 n + \frac{n^2}{s}\log n\right)$$

expected time and $O(s)$ workspace, where $s \in \Omega(\log n) \cap O(n)$, we can either find the geodesic center of P or find a triangle whose sides are chords of P and that contains the geodesic center.

Proof. Using Lemma 3.3, we first find a chord that splits P into two nearly equal subpolygons. We then determine which subpolygon contains the geodesic center and recurse on that subpolygon. At each step, let P_i denote the current underlying subpolygon and let d_i be a balanced chord of P_i. The chord d_i also partitions the original polygon P into two subpolygons, denoted P_l^i and P_r^i. The chords d_i can be chosen vertical and, under the nested recursion, are pairwise noncrossing.

Let c_i be the geodesic center of P constrained to d_i, computed by Lemma 3.2. By constructing $ST_P(c_i)$, we obtain $F_P(c_i)$. Write $\vec{v}_p(q)$ for the unit vector at p along the first edge of $\pi_P(p, q)$. By Lemmas 2 and 3 of Pollack et al. [1], if

$$\{\vec{v}_{c_i}(f) : f \in F_P(c_i)\}$$

is not contained in any open half-plane through c_i, then c_i is the geodesic center of P. Otherwise, the cone generated by these directions identifies the side of d_i containing the geodesic center. Since a planar cone is determined by its two extreme rays, all directions need not be stored simultaneously.

Assume, without loss of generality, that P_l^i contains the geodesic center. We set $P_{i+1} = P_i \cap P_l^i$ and continue. Because each balanced cut reduces the size of the underlying subpolygon by a constant factor, after $O(\log n)$ steps the underlying subpolygon has constant complexity. Triangulating this constant-complexity polygon and applying the constrained-center test to its diagonals yields either the geodesic center or a triangle containing it.

The process has $O(\log n)$ steps. At each step, the underlying subpolygon is described implicitly by the $O(\log n)$ previously chosen chords and side decisions. Applying Lemmas 3.2 and 3.3 at each step gives the stated expected running time and workspace bound. ■

Let $\triangle\alpha\beta\gamma$ be the triangle obtained in Lemma 3.4. We next find a subregion of this triangle that contains the geodesic center and on which the shortest path tree is unchanged. We adapt the approach of Pollack et al. [1] to the memory-constrained setting.

The algorithm has $O(\log n)$ rounds. Each round identifies two half-planes containing the geodesic center. Consequently, after $O(\log n)$ rounds we obtain a set $\mathcal{L}^*$ of $O(\log n)$ half-planes whose intersection with $\triangle\alpha\beta\gamma$ contains the geodesic center and intersects only a constant number of the lines at which the combinatorial structure of the shortest path tree can change.

Let $\mathcal{L}$ be the set of supporting lines of all edges of $ST_P(\alpha)$, $ST_P(\beta)$, and $ST_P(\gamma)$. Initially, $\mathcal{L}^*$ is empty. Using random sampling as in Lemma 3.1, we find a line ℓ_m^1 whose slope is an approximate median of the slopes of the lines in $\mathcal{L}$. We then rotate the coordinate system so that ℓ_m^1 is the x-axis. The lines are paired so that, whenever possible, one line of each pair has positive slope and the other has negative slope. This pairing can be generated in limited workspace by running the shortest path tree procedure twice for each of $ST_P(\alpha)$, $ST_P(\beta)$, and $ST_P(\gamma)$, once to report positive-slope lines and once to report negative-slope lines.

Let x_m^1 be a median of the x-coordinates of the pairwise intersections. By solving the geodesic center problem constrained to the line $x = x_m^1$, we determine which of its two half-planes contains the geodesic center. Among the pairwise intersections on the opposite side, let y_m^1 be a median of their y-coordinates. We similarly solve the constrained problem on $y = y_m^1$ and retain the corresponding half-plane. Denote the two retained half-planes by h_x^1 and h_y^1 and add them to $\mathcal{L}^*$.

At least a constant fraction of the pairs have their intersection outside the retained region $h_x^1 \cap h_y^1$. For each such pair, one line does not intersect the retained region and can be discarded for the next round; all other lines remain active. Thus a constant fraction of the active lines is removed per round. Repeating the procedure for $O(\log n)$ rounds leaves only a constant number of lines of $\mathcal{L}$ that intersect the retained region. We solve the constrained geodesic center problem directly on each remaining line and add the corresponding half-plane containing the solution to $\mathcal{L}^*$.

The resulting region $R \subseteq \triangle\alpha\beta\gamma$ contains the geodesic center and has interior disjoint from every line in $\mathcal{L}$. Therefore the shortest path tree has the same combinatorial structure for all points of R. Choose any $x_0 \in R$. Let $q_1, \ldots, q_k$ be the vertices of P that form the first level of $ST_P(x_0)$, and let f_i be the maximum geodesic distance from q_i to a vertex in the subtree rooted at q_i. Then the geodesic center problem is equivalent to finding the smallest circle enclosing $C(q_1, f_1), \ldots, C(q_k, f_k)$. If z is the center of the enclosing circle and ρ its radius, the problem is

$$\begin{aligned} &\text{minimize} && \rho, \\ &\text{subject to} && \|z - q_i\| + f_i \le \rho, \qquad 1 \le i \le k. \end{aligned} \tag{1}$$

Problem (1) is a fixed-dimensional convex optimization problem and can be solved in linear time when unrestricted workspace is available. We adapt the methods of Megiddo [9,10] to the memory-constrained setting.

Although the constraints are not linear, Megiddo [9] shows that for each pair of constraints there is a separating plane with the following property: once the side of the optimal solution relative to that plane is known, one of the paired constraints can be discarded. Suppose we have an oracle that, for a given plane, determines whether the solution lies on the plane and, otherwise, which open half-space contains it. Megiddo's fixed-dimensional prune-and-search framework [10] uses a constant number of median computations, pairings, and oracle calls per stage to discard a constant fraction of the active constraints.

Median computations can be performed by regenerating the relevant objects through a constant expected number of shortest path tree traversals. Pairing requires only two further traversals, one producing objects of positive orientation and one producing those of negative orientation. As in the constrained-circle case, let A_i compute a retained subspace containing the solution and let B_i generate the next active level. Once each retained subspace is stored, the same tournament-style regeneration argument applies. Hence the running time is governed by the larger of the shortest path tree time and the oracle time.

It remains to implement the oracle. For Problem (1) and a given plane h, the oracle first solves the problem constrained to h. It then determines a descent direction for the unconstrained objective. The constrained problem has one lower dimension and can therefore be solved recursively. The base case is the one-dimensional problem, which is handled by essentially the same method as the geodesic center constrained to a line. Because the dimension is fixed, the recursion depth is constant, and the oracle has the same asymptotic running time as the constrained geodesic center procedure of Lemma 3.2. We obtain the following theorem.

Theorem 3.5. *There is an s-workspace algorithm that computes the geodesic center of a simple polygon with n vertices in*

$$O\left(T(n,s)\log^2 n + \frac{n^2}{s}\log n\right)$$

expected time, where $s \in \Omega(\log n) \cap O(n)$. In particular, the algorithm uses $O(s)$ words of workspace.

4 Conclusion

We studied the geodesic center problem for a simple polygon under a restricted-workspace model. For an n-vertex simple polygon P, we gave a time-space trade-off algorithm that computes the geodesic center in

$$O\left(T(n,s)\log^2 n + \frac{n^2}{s}\log n\right)$$

expected time using $O(s)$ words of workspace, for $s \in \Omega(\log n) \cap O(n)$. Here, $T(n,s)$ denotes the time required to construct, in depth-first order, the shortest path tree of a point inside P using $O(s)$ extra space. Substituting the shortest path tree bound of Oh and Ahn [6] gives an expected running time of $O((n^2/s)\log^3 n)$.

References

[1] Pollack, R., Sharir, M., & Rote, G. (1989). Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(1), 611–626.

[2] Ahn, H. K., Barba, L., Bose, P., De Carufel, J. L., Korman, M., & Oh, E. (2016). A linear-time algorithm for the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 56, 836–859.

[3] Banyassady, B., Korman, M., & Mulzer, W. (2018). Computational geometry column 67. *ACM SIGACT News*, 49(2), 77–94.

[4] Har-Peled, S. (2015). Shortest path in a polygon using sublinear space. *Journal of Computational Geometry*, 7(2), 19–45.

[5] Aronov, B., Korman, M., Pratt, S., van Renssen, A., & Roeloffzen, M. (2017). Time-space trade-off algorithms for triangulating a simple polygon. *Journal of Computational Geometry*, 8(1), 105–124.

[6] Oh, E., & Ahn, H. K. (2019). A new balanced subdivision of a simple polygon for time-space trade-off algorithms. *Algorithmica*, 81, 2829–2856.

[7] Megiddo, N. (1983). Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12(4), 759–776.

[8] Chazelle, B. (1982). A theorem on polygon cutting with applications. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 339–349.

[9] Megiddo, N. (1989). On the ball spanned by balls. *Discrete & Computational Geometry*, 4, 605–610.

[10] Megiddo, N. (1984). Linear programming in linear time when the dimension is fixed. *Journal of the Association for Computing Machinery*, 31(1), 114–127.