# Shortest Path Map Equivalence Decompositions and Applications

**Haitao Wang**
Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

## Abstract

Given a polygonal domain $\mathcal{P}$ in the plane, the shortest path map with respect to a point $s$, denoted by $SPM(s)$, is the decomposition of $\mathcal{P}$ into cells such that shortest paths from $s$ to all points $t$ in the same cell have the same vertex sequence. The shortest path map equivalence decomposition of $\mathcal{P}$ is the decomposition of $\mathcal{P}$ into cells so that $SPM(s)$ is topologically equivalent for all points $s$ in the same cell. In this paper, we prove new upper bounds on the combinatorial complexities of the $SPM$-equivalence decompositions under various settings, depending on whether $s$ and/or $t$ are restricted to be the boundary of $\mathcal{P}$. We also propose new algorithms to compute these decompositions. Further, our results lead to new solutions to several other problems, including answering two-point shortest path queries in $\mathcal{P}$, and computing geodesic diameter and center of $\mathcal{P}$.

**2012 ACM Subject Classification** Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

**Keywords and phrases** shortest paths, shortest path maps, SPM-equivalent decompositions, two-point shortest path queries, geodesic diameter, geodesic center, polygononal domains

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2026.160

**Related Version** *Full Version*: https://arxiv.org/abs/2607.03416

## 1 Introduction

Let $\mathcal{P}$ be a polygonal domain of $h$ holes with a total of $n$ vertices in the plane, i.e., $\mathcal{P}$ is a closed and multiply-connected region bounded by $n$ segments that form $h+1$ cycles (holes and the region outside $\mathcal{P}$ are also called *obstacles*). The *shortest path map* (or SPM for short) with respect to a point $s \in \mathcal{P}$, denoted by $SPM(s)$, is the decomposition of $\mathcal{P}$ into cells such that shortest paths from $s$ to all points $t$ in the same cell are combinatorially the same (i.e., they have the same vertex sequence) [10, 23]; see Fig. 1. The *SPM-equivalence decomposition* of $\mathcal{P}$, denoted by $\Psi$, is the decomposition of $\mathcal{P}$ into cells so that $SPM(s)$ is topologically equivalent for all points $s$ in the same cell. Chiang and Mitchell [10] first studied the SPM-equivalence decomposition and used it to answer two-point shortest path queries in $\mathcal{P}$. They proved that the combinatorial complexity of $\Psi$ is bounded by $O(n^{10})$ and proposed an $O(n^{10}\log n)$ time algorithm to compute it.

We consider several variants of the SPM-equivalence decompositions, depending on whether $s$ and/or $t$ are required to be on the boundary of $P$. Specifically, let $\mathcal{B}$ denote the boundary of $\mathcal{P}$. We define the *boundary-restricted* shortest path map for a point $s$, denoted by $SPM_{\mathcal{B}}(s)$, as the decomposition of $\mathcal{B}$ into segments such that shortest paths from $s$ to all points $t$ in the same segment are combinatorially the same. We define the *boundary-restricted SPM-equivalence decomposition* with respect to $\mathcal{P}$ (resp., $\mathcal{B}$), denoted by $\Psi_{\mathcal{B}}(\mathcal{P})$ (resp., $\Psi_{\mathcal{B}}(\mathcal{B})$), as the decomposition of $\mathcal{B}$ into segments so that $SPM(s)$ (resp., $SPM_{\mathcal{B}}(s)$) are topologically equivalent for all points $s$ in the same segment. Similarly, we let $\Psi_{\mathcal{P}}(\mathcal{B})$ denote the decomposition of $\mathcal{P}$ into cells so that $SPM_{\mathcal{B}}(s)$ is topologically the same for all points $s$ in the same cell. If we follow the same convention for notation, then $\Psi_{\mathcal{P}}(\mathcal{P})$ is $\Psi$ (we will use the two notations interchangeably). Another way to view these four decompositions is to define them depending on whether $s$ and/or $t$ are restricted to be


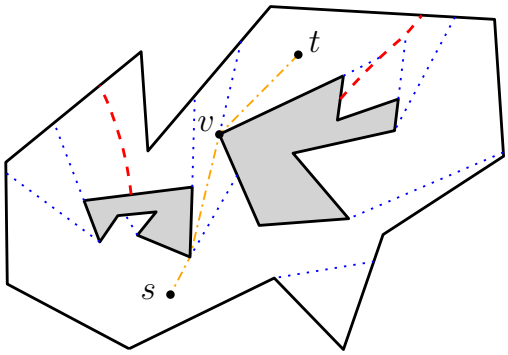


**Figure 1** Illustrating a shortest path map $SPM(s)$: The dotted blue segments are extension segments and the dashed red curves are bisector curves. The point $t$ is in a cell whose root is $v$.

on $\mathcal{B}$. For example, $\Psi_{\mathcal{P}}(\mathcal{B})$ is for the case where $s \in \mathcal{P}$ while $t \in \mathcal{B}$. Note that $\Psi_{\mathcal{P}}(\mathcal{P})$ is a refinement of $\Psi_{\mathcal{P}}(\mathcal{B})$ (i.e., each cell of $\Psi_{\mathcal{P}}(\mathcal{P})$ is completely inside a cell of $\Psi_{\mathcal{P}}(\mathcal{B})$), and $\Psi_{\mathcal{B}}(\mathcal{P})$ is a refinement of $\Psi_{\mathcal{B}}(\mathcal{B})$.

While we are not aware of any previous work on $\Psi_{\mathcal{B}}(\mathcal{B})$ and $\Psi_{\mathcal{B}}(\mathcal{P})$, $\Psi_{\mathcal{P}}(\mathcal{B})$ was implicitly discussed in [10]. Chiang and Mitchell's approach [10] also gave an $O(n^{10})$ upper bound for $|\Psi_{\mathcal{P}}(\mathcal{B})|$, the same as $|\Psi_{\mathcal{P}}(\mathcal{P})|$. Their algorithm computes $\Psi_{\mathcal{P}}(\mathcal{B})$ in $O(n^{10} \log n)$ time.

We obtain new bounds/algorithms for these decompositions (see Table 1 for a summary).

- For $\Psi_{\mathcal{B}}(\mathcal{B})$, we prove $|\Psi_{\mathcal{B}}(\mathcal{B})| = O(n^{4+\epsilon})$ and give an algorithm to compute $\Psi_{\mathcal{B}}(\mathcal{B})$ in $O(n^{4+\epsilon})$ time; throughout the paper, let $\epsilon$ represent an arbitrarily small positive constant. For comparison, a traditional method [3, 30, 38] can obtain $|\Psi_{\mathcal{B}}(\mathcal{B})| = O(n^5)$.
- For $\Psi_{\mathcal{B}}(\mathcal{P})$, we present an algorithm to compute it in $O((n^2 + |\Psi_{\mathcal{B}}(\mathcal{P})|) \cdot \log n)$ time, and prove that $|\Psi_{\mathcal{B}}(\mathcal{P})| = O(n^5)$. Note that $|\Psi_{\mathcal{B}}(\mathcal{P})|$ denotes the combinatorial size of $\Psi_{\mathcal{B}}(\mathcal{P})$ in the *worst case* among all input instances $\mathcal{P}$ (not the size in any specific input instance). We follow the same convention for other similar notation, e.g., $|\Psi_{\mathcal{P}}(\mathcal{P})|$, $|\Psi_{\mathcal{B}}(\mathcal{B})|$, $|\Psi_{\mathcal{P}}(\mathcal{B})|$. Therefore, a time complexity like $O((n^2 + |\Psi_{\mathcal{B}}(\mathcal{P})|) \cdot \log n)$ is a function of the worst-case size of $\Psi_{\mathcal{B}}(\mathcal{P})$.
- For the most general decomposition $\Psi_{\mathcal{P}}(\mathcal{P})$, we do not have a better bound than $O(n^{10})$ proved by Chiang and Mitchell [10]. However, we present an algorithm that can compute $\Psi_{\mathcal{P}}(\mathcal{P})$ in $O(n^{7.73} + |\Psi_{\mathcal{P}}(\mathcal{P})| \log n)$ time. As such, if $|\Psi_{\mathcal{P}}(\mathcal{P})|$ could be bounded by $o(n^{10})$, then our algorithm for constructing $\Psi_{\mathcal{P}}(\mathcal{P})$ would be faster than the $O(n^{10} \log n)$-time solution in [10]. Note that the time complexity of the algorithm of [10] does not depend on $|\Psi_{\mathcal{P}}(\mathcal{P})|$. The currently best-known lower bound for $|\Psi_{\mathcal{P}}(\mathcal{P})|$ is $\Omega(n^4)$ [10]. Hence, our result reduces the problem of designing efficient algorithms for computing $\Psi_{\mathcal{P}}(\mathcal{P})$ to proving smaller upper bounds for its combinatorial complexity.
- For $\Psi_{\mathcal{P}}(\mathcal{B})$, we prove $|\Psi_{\mathcal{P}}(\mathcal{B})| = O(n^8)$ and present an algorithm that can compute it in $O(n^5 \log^2 n + |\Psi_{\mathcal{P}}(\mathcal{B})| \cdot \log n)$ time.

Applying our results on these decompositions, we obtain new results on answering two-point shortest path queries, and computing geodesic diameter and center.

**Two-point shortest path queries.** The problem is to build a data structure so that a (Euclidean) shortest path from $s$ to $t$ in $\mathcal{P}$ can be quickly found for any two query points $s$ and $t$. Chiang and Mitchell [10] built a data structure in $O(n^{11})$ space and preprocessing time, and each query can be answered in $O(\log n)$ time (this is the time for computing shortest path length; reporting the path itself needs additional time linear in the number of

**Table 1** Summary of the results on the four decompositions.

| | Combinatorial Sizes | | Algorithm Complexities | |
|---|---|---|---|---|
| | Previous | Ours | Previous | Ours |
| $\Psi_{\mathcal{B}}(\mathcal{B})$ | $O(n^5)$ [3, 30, 38] | $O(n^{4+\epsilon})$ | $O(n^5)$ [3, 30, 38] | $O(n^{4+\epsilon})$ |
| $\Psi_{\mathcal{B}}(\mathcal{P})$ | | $O(n^5)$ | | $O((n^2 + \lvert\Psi_{\mathcal{B}}(\mathcal{P})\rvert) \cdot \log n)$ |
| $\Psi_{\mathcal{P}}(\mathcal{P})$ | $O(n^{10})$ [10] | | $O(n^{10} \log n)$ [10] | $O(n^{7.73} + \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert \cdot \log n)$ |
| $\Psi_{\mathcal{P}}(\mathcal{B})$ | $O(n^{10})$ [10] | $O(n^8)$ | $O(n^{10} \log n)$ [10] | $O(n^5 \log^2 n + \lvert\Psi_{\mathcal{P}}(\mathcal{B})\rvert \cdot \log n)$ |

edges of the path; we follow this convention throughout the paper). For simplicity, we use $O(P(n), S(n), Q(n))$ to denote the complexity of such a data structure, where $P(n)$, $S(n)$, and $Q(n)$ denote the preprocessing time, space, and query time, respectively. Chiang and Mitchell [10] also gave a data structure of $O(n^{10} \log n, n^{10} \log n, \log^2 n)$ complexity. In case the number of obstacles $h$ is much smaller than the number of vertices $n$, they also provided data structures with complexities sensitive to $h$. For example, they gave a data structure of complexity $O(n^5, n^5, h + \log n)$ and another data structure of $O(n + h^5)$ space with $O(h \log n)$ query time. Refer to [10] for other tradeoffs between preprocessing and query time.

Guo, Maheshwari, and Sack [17] also studied the problem and presented a data structure of complexity $O(n^2 \log n, n^2, h \log n)$. Bae and Okamoto [5] considered a special case where both query points are restricted on $\mathcal{B}$, and they derived a data structure of complexity roughly $O(n^5 \log n \log^* n, n^5 \log^* n, \log n)$. Very recently in SoCG 2024, by introducing a novel idea of using cuttings [8], de Berg, Miltzow, and Staals [6] revisited the problem and gave two data structures of complexities $O(n^{10+\epsilon}, n^{10+\epsilon}, \log n)$ and $O(n^{9+\epsilon}, n^{9+\epsilon}, \log^2 n)$, respectively; their preprocessing algorithm involves randomization (due to using the algorithm in [26]) and the preprocessing time is expected, while the space and query complexities are worst-case. For the case where both $s$ and $t$ are restricted on $\mathcal{B}$ (called the *$\mathcal{B}$-$\mathcal{B}$ case*), they gave a data structure of $O(n^{4+\epsilon}, n^{4+\epsilon}, \log n)$ complexity [6]. If only one of the query points is restricted on $\mathcal{B}$ (called the *$\mathcal{B}$-$\mathcal{P}$ case*), they gave a randomized data structure of $O(n^{6+\epsilon}, n^{6+\epsilon}, \log^2 n)$ complexity, where the preprocessing time is expected. Note that the method of [6] does not yield any new results on the SPM-equivalence decompositions (which were not used in [6]).

Building upon our new results on the SPM-equivalence decompositions, we achieve the following results on two-point shortest path queries (see Table 2 for a summary).

- For the $\mathcal{B}$-$\mathcal{B}$ case, we obtain a data structure of complexity $O(\lvert\Psi_{\mathcal{B}}(\mathcal{B})\rvert \log n, \lvert\Psi_{\mathcal{B}}(\mathcal{B})\rvert, \log n)$, assuming that $\Psi_{\mathcal{B}}(\mathcal{B})$ is available. With our $O(n^{4+\epsilon})$ time algorithm for computing $\Psi_{\mathcal{B}}(\mathcal{B})$, we obtain the complexity $O(n^{4+\epsilon}, n^{4+\epsilon}, \log n)$. This matches the result of [6].
- For the $\mathcal{B}$-$\mathcal{P}$ case, we obtain two data structures of complexities $O(n^2 \log n + \lvert\Psi_{\mathcal{B}}(\mathcal{P})\rvert \cdot n, \lvert\Psi_{\mathcal{B}}(\mathcal{P})\rvert \cdot n, \log n)$ and $O((n^2 + \lvert\Psi_{\mathcal{B}}(\mathcal{P})\rvert) \cdot \log n, \lvert\Psi_{\mathcal{B}}(\mathcal{P})\rvert \cdot \log n, \log^2 n)$, respectively. Using our bound $\lvert\Psi_{\mathcal{B}}(\mathcal{P})\rvert = O(n^5)$, these are $O(n^6, n^6, \log n)$ and $O(n^5 \log n, n^5 \log n, \log^2 n)$, respectively. This improves the result of $O(n^{6+\epsilon}, n^{6+\epsilon}, \log^2 n)$ from [6].
- For the general case (i.e., the $\mathcal{P}$-$\mathcal{P}$ case), we can have two data structures of complexities $O(n^{7.73} + \lvert\Psi\rvert \log n, (n^7 + \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert) \log n, \log^2 n)$ and $O(n^8 + n \cdot \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert, n^8 + n \cdot \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert, \log n)$, respectively. Using the current best bound $\lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert = O(n^{10})$, these match the results from [10] and are worse than the results from [6]. However, if the "true" size of $\lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert$ is much smaller than $O(n^{10})$ (e.g., $\lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert = O(n^9)$), then our result could be better than all previous work.

Computing shortest paths in polygonal domains is a classical problem and has been studied extensively in the past several decades, e.g., [9, 13–16, 20, 21, 23, 24, 27, 29, 30, 32, 36]. For the one-point query problem where $s$ is given in the input and $t$ is the only query point,

**Table 2** Summary of the results on two-point shortest path queries.

| | Previous | Ours |
|---|---|---|
| $\mathcal{B}$-$\mathcal{B}$ case | $O(n^{4+\epsilon}, n^{4+\epsilon}, \log n)$ [6] | $O(n^{4+\epsilon}, n^{4+\epsilon}, \log n)$ |
| $\mathcal{B}$-$\mathcal{P}$ case | $O(n^{6+\epsilon}, n^{6+\epsilon}, \log^2 n)$ [6] | $O(n^5 \log n, n^5 \log n, \log^2 n)$ or $O(n^6, n^6, \log n)$ |
| $\mathcal{P}$-$\mathcal{P}$ case | $O(n^{10+\epsilon}, n^{10+\epsilon}, \log n)$ [6] | $O(n^8 + n \cdot \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert, n^8 + n \cdot \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert, \log n)$ |
| | $O(n^{9+\epsilon}, n^{9+\epsilon}, \log^2 n)$ [6] | $O(n^{7.73} + \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert \log n, (n^7 + \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert) \log n, \log^2 n)$ |

the problem becomes much easier and $SPM(s)$, which is of space $O(n)$, along with a point location data structure [12, 25, 33], can be used to answer shortest path queries in $O(\log n)$ time. Constructing $SPM(s)$ can be done in $O(n \log n)$ time [23] or in $O(n + h \log h)$ time [39] after $\mathcal{P}$ is triangulated. If $\mathcal{P}$ is a simple polygon (i.e., $h = 0$), then there exists a data structure of $O(n, n, \log n)$ complexity for the two-point query problem [15].

**Geodesic diameter.** For two points in $\mathcal{P}$, the length of their shortest path in $\mathcal{P}$ is also called the *geodesic distance*. The pair of points of $\mathcal{P}$ whose geodesic distance is the largest is called a *diametral pair* and their geodesic distance is called the *geodesic diameter* of $\mathcal{P}$.

If $\mathcal{P}$ is a simple polygon, computing the geodesic diameter can be done in $O(n)$ time [22], improving the $O(n \log n)$ time results [15, 37]. For the polygonal domain case, the problem becomes much more challenging. One main difficulty lies in that a diametral pair of points could both be in the interior of $\mathcal{P}$. Bae, Korman, and Okamoto [3] thoroughly studied the problem and discovered many interesting properties. Their effort led to an algorithm of $O(n^{7.73})$ time (or $O(n^7(h + \log n))$ for small $h$) for computing the diameter. Their algorithm can be improved to $O(n^{7.4+\epsilon})$ time using the new shortest path query algorithm [6]. Other restricted cases of the problem were also studied in [3]. For the $\mathcal{B}$-$\mathcal{B}$ case, which aims to find a *restricted* diametral pair $(s, t)$ with $s \in \mathcal{B}$ and $t \in \mathcal{B}$, the method of [3] solved the problem in $O(n^5 \log n \log^* n)$ time. For the $\mathcal{B}$-$\mathcal{P}$ case, where one wishes to find a restricted diametral pair $(s, t)$ with $s \in \mathcal{B}$ and $t \in \mathcal{P}$, an algorithm of roughly $O(n^{5.91})$ time is presented in [3].

Based on our new results on the SPM-equivalence decompositions, we solve the $\mathcal{B}$-$\mathcal{B}$ case in $O(n^{4+\epsilon})$ time and the $\mathcal{B}$-$\mathcal{P}$ case in $O(n^5 \log n)$ time. These improve the $O(n^5 \log n \log^* n)$ time and $O(n^{5.91})$ time results in [3], respectively (see Table 3 for a summary; note that the $\mathcal{P}$-$\mathcal{B}$ case is symmetric to the $\mathcal{B}$-$\mathcal{P}$ case).

**Table 3** Summary of the results on the geodesic diameter and geodesic center.

| | Geodesic Diameter | | Geodesic Center | |
|---|---|---|---|---|
| | Previous | Ours | Previous | Ours |
| $\mathcal{B}$-$\mathcal{B}$ case | $O(n^5 \log n \log^* n)$ [3] | $O(n^{4+\epsilon})$ | $O(n^8 \log n)$ [38] | $O(n^{4+\epsilon})$ |
| $\mathcal{B}$-$\mathcal{P}$ case | $O(n^{5.91})$ [3] | $O(n^5 \log n)$ | $O(n^8 \log n)$ [38] | $O(n^5 \log n \log^* n)$ |
| $\mathcal{P}$-$\mathcal{B}$ case | $O(n^{5.91})$ [3] | $O(n^5 \log n)$ | $O(n^{11} \log n)$ [38] | $O(n^{10+\epsilon})$ |
| $\mathcal{P}$-$\mathcal{P}$ case | $O(n^{7.4+\epsilon})$ [6] | | $O(n^{11} \log n)$ [38] | $O((n^7 + \lvert\Psi_{\mathcal{P}}(\mathcal{P})\rvert) \cdot n^{2+\epsilon})$ |

**Geodesic center.** A closely related concept is the *geodesic center* of $\mathcal{P}$, which is a point that minimizes the maximum geodesic distance from it to any other point in $\mathcal{P}$. The problem of computing the geodesic center has attracted much attention [1, 2, 4, 31, 38]. If $\mathcal{P}$ is a simple polygon, a linear-time algorithm for computing a geodesic center is known [1, 28]. For the polygonal domain case, the problem again becomes significantly more challenging. One

reason is that a farthest point of a point may be in the interior of $\mathcal{P}$ [3]; this is in contrast to the simple polygon case in which a farthest point must be a vertex of $\mathcal{P}$. Bae, Korman, and Okamoto [4] gave the first-known algorithm that can compute a geodesic center in $O(n^{12+\epsilon})$ time. Later Wang [38] derived an $O(n^{11}\log n)$ time algorithm.

We obtain improved results on several restricted versions (see Table 3 for a summary).

- First, if the center is required to be on $\mathcal{B}$ and farthest points are also required to be on $\mathcal{B}$, i.e., finding a point on $\mathcal{B}$ to minimize the maximum geodesic distance from it to any other point in $\mathcal{B}$, then we solve this case in $O(|\Psi_{\mathcal{B}}(\mathcal{B})|\log n\log^* n)$ time, which is $O(n^{4+\epsilon})$ with our bound $|\Psi_{\mathcal{B}}(\mathcal{B})| = O(n^{4+\epsilon})$. To compare, [38] solves this problem in $O(n^8\log n)$ time.
- Second, if the center is required on $\mathcal{B}$ but farthest points can be anywhere in $\mathcal{P}$, then our algorithm runs in $O(|\Psi_{\mathcal{B}}(\mathcal{P})|\log n\log^* n)$ time, which is $O(n^5\log n\log^* n)$ with our bound $|\Psi_{\mathcal{B}}(\mathcal{P})| = O(n^5)$. The algorithm of [38] solves this problem in $O(n^8\log n)$ time.
- Third, if the center can be anywhere in $\mathcal{P}$ while the farthest points are required to be on $\mathcal{B}$, then we solve the problem in $O((n^5+|\Psi_{\mathcal{P}}(\mathcal{B})|)\cdot n^{2+\epsilon})$ time, which is $O(n^{10+\epsilon})$ with our bound $|\Psi_{\mathcal{P}}(\mathcal{B})| = O(n^8)$. The method of [38] solves this problem in $O(n^{11}\log n)$ time.
- Finally, for the most general problem, our algorithm runs in $O((n^7+|\Psi_{\mathcal{P}}(\mathcal{P})|)\cdot n^{2+\epsilon})$ time. As discussed above, the currently best known upper bound for $|\Psi_{\mathcal{P}}(\mathcal{P})|$ is $O(n^{10})$ [10].

## 1.1 An overview of our approach

To bound $|\Psi_{\mathcal{B}}(\mathcal{B})|$, we reduce the problem to analyzing the combinatorial complexities of lower envelopes of certain functions. For each vertex $u$ of $\mathcal{P}$, we define a set of functions $f_u(s,t)$ as follows. Let $\mathit{Vis}(u)$ denote the visibility polygon of $u$ in $\mathcal{P}$. For each edge $e_t \in \mathit{SPM}_{\mathcal{B}}(u)$, $e_t$ lies in a single cell of $\mathit{SPM}(u)$, and let $v_t$ be the root of the cell. For each edge $e_s$ of $\mathit{Vis}(u)\cap\mathcal{B}$, define a function $f_u(s,t) = |su| + d(u,v_t) + |v_t t|$, for $s \in e_s$ and $t \in e_t$, where $d(u,v_t)$ is the geodesic distance between $u$ and $v_t$. Since both $\mathit{Vis}(u)\cap\mathcal{B}$ and $\mathit{SPM}_{\mathcal{B}}(u)$ have $O(n)$ edges, the above defines $O(n^2)$ bivariate functions (of constant degree) for $u$, each of which defines a constant-sized two-dimensional surface patch in $\mathbb{R}^3$. As $\mathcal{P}$ has $O(n)$ vertices, we have $O(n^3)$ functions in total; let $F$ denote the set of all of these functions. Our first observation is that a vertex of $\Psi_{\mathcal{B}}(\mathcal{B})$ corresponds to a vertex of the lower envelope $\mathcal{L}(F)$ of (the surface patches defined by) the functions of $F$. Hence, it suffices to analyze the complexity of $\mathcal{L}(F)$.

To find a bound for $|\mathcal{L}(F)|$, one could apply the result of Sharir [34], who proved an $O(n^{d+\epsilon})$ upper bound for the size of the lower envelope of a collection of $n$ constant-sized (partially defined) $d$-variate algebraic functions of constant degree, for any $d \geq 2$ (the $d = 2$ case was first proved in [18]; see also [35, Chapter 7]). Since $|F| = O(n^3)$ and $d = 2$ in our problem, applying the above result directly could obtain an upper bound $O(n^{6+\epsilon})$ for $|\mathcal{L}(F)|$. We instead take a closer look at the problem. Our main idea is that we find a way to decompose the lower envelope $\mathcal{L}(F)$ into $O(n^2)$ regions so that for each region it suffices to consider $O(n)$ functions of $F$. Consequently, by applying the algorithm of [18, 34, 35] for each region, we can prove an upper bound of $O(n^{4+\epsilon})$ for $|\mathcal{L}(F)|$ and thus for $|\Psi_{\mathcal{B}}(\mathcal{B})|$, and also compute the vertices of $\mathcal{L}(F)$ and thus compute $\Psi_{\mathcal{B}}(\mathcal{B})$ in $O(n^{4+\epsilon})$ time.

For $\Psi_{\mathcal{B}}(\mathcal{P})$, a simple brute-force approach can provide a good upper bound for its size (similar techniques were used before for related problems [3, 10, 38]). To compute $\Psi_{\mathcal{B}}(\mathcal{P})$, since it is a decomposition of $\mathcal{B}$, which is one-dimensional, we use a sweeping point algorithm, i.e., moving a point $s$ on $\mathcal{B}$ and find those event points when $\mathit{SPM}(s)$ changes combinatorially.

For $\Psi_{\mathcal{P}}(\mathcal{P})$, unfortunately, our approach does not yield a better result than the $O(n^{10})$ bound from [10]. One reason is that the decomposition is on $\mathcal{P}$, which is two-dimensional, and a vertex of it might not correspond to a vertex of the lower envelope of the corresponding

functions. Hence, analyzing the complexity of the lower envelope does not immediately lead to an upper bound for $\Psi_{\mathcal{P}}(\mathcal{P})$. The same issue happens to $\Psi_{\mathcal{P}}(\mathcal{B})$ as well. In contrast, both $\Psi_{\mathcal{B}}(\mathcal{B})$ and $\Psi_{\mathcal{B}}(\mathcal{P})$ are decompositions of $\mathcal{B}$, which is essentially one-dimensional, and a vertex of them corresponds to a vertex of the corresponding lower envelope.

To construct $\Psi_{\mathcal{P}}(\mathcal{P})$ and $\Psi_{\mathcal{P}}(\mathcal{B})$, we first present efficient algorithms to compute the edges of the lower envelopes of the corresponding functions. We then compute these decompositions by using the edges of the lower envelopes.

To answer two-point shortest path queries, using a SPM-equivalent decomposition, a data structure can be built by constructing a "parameterized" shortest path map for points in each cell of the decomposition, which is the approach first proposed in [10] (a linear factor in the preprocessing can be saved by using persistent data structures [11], but the query time grows to $O(\log^2 n)$). This approach works for all SPM-equivalent decompostions, e.g., for the $\mathcal{B}$-$\mathcal{B}$ case, $\Psi_{\mathcal{B}}(\mathcal{B})$ is used (in this case, we actually show that $O(\log n)$ query time can still be achieved even when a persistent data structure is utilized).

For computing the geodesic diameter, using the results of Bae, Korman, and Okamoto [3], we show that a diametral pair corresponds to a vertex of the lower envelope of the corresponding functions. Consequently, once we construct the lower envelope, the diametral pairs can be easily found. This approach works for all cases.

To compute geodesic centers, an observation is that for any point $s \in \mathcal{P}$, its farthest point in $\mathcal{P}$ must be a vertex of $SPM(s)$ [4]. Based on this property, for each cell $\sigma$ of the SPM-equivalent decomposition, for each vertex $v$ of $SPM(s)$, we parameterize the geodesic distance $d(s, v)$ using the coordinate of $s \in \sigma$. Then, a geodesic center $s$ restricted to $s \in \sigma$ corresponds to a lowest vertex in the upper envelope of the functions $d(s, v)$ for all vertices $v$ of $SPM(s)$. This approach was used previously for the general case [4]. However, to make the algorithm efficient, one needs good upper bounds on SPM-equivalent decompositions and efficient algorithms to compute them. As we have new upper bounds and algorithms for the decompostions, we obtain new geodesic center algorithms.

**Outline.** The rest of the paper is organized as follows. Section 2 defines some notation and concepts that will be used throughout the paper. The decompositions $\Psi_{\mathcal{B}}(\mathcal{B})$ and $\Psi_{\mathcal{B}}(\mathcal{P})$ are discussed in Sections 3 and 4, respectively. Some of their applications are given in Appendix 5. Due to the space limit, our results for $\Psi_{\mathcal{P}}(\mathcal{P})$ and $\Psi_{\mathcal{P}}(\mathcal{B})$ are omitted but can be found in the full paper. A complete discussion of the applications of our results on the SPM-decompositions are described in the full paper.

## 2 Preliminaries

We define some notation and concepts that will be used throughout the paper, in addition to those already introduced in Section 1.

For any two points $p$ and $q$ in the plane, denote by $\overline{pq}$ the line segment with $p$ and $q$ as endpoints; denote by $|pq|$ the length of the segment. We say that $p$ is *visible* to $q$ if $\overline{pq} \subseteq \mathcal{P}$.

For any two points $s$ and $t$ in $\mathcal{P}$, we use $\pi(s, t)$ to denote a shortest path from $s$ to $t$ in $\mathcal{P}$. Denote by $d(s, t)$ the length of $\pi(s, t)$; we call $d(s, t)$ the *geodesic distance* between $s$ and $t$. The vertex of $\pi(s, t)$ adjacent to $s$ (resp., $t$) is called the *anchor* of $s$ (resp., $t$).

Define $d_{u,v}(s, t) = |su| + d(u, v) + |vt|$ for two vertices $u, v$ of $\mathcal{P}$ and two points $s, t \in \mathbb{R}^2$; note that when using this notation, we do not always guarantee that $s$ is visible to $u$ and $t$ is visible to $v$, and $s$ or/and $t$ may not even be inside $\mathcal{P}$.

For any compact region $R$ in the plane, let $\partial R$ denote its boundary. Recall that $\mathcal{B}$ is the boundary of $\mathcal{P}$. For differentiation, we often refer to the vertices of $\mathcal{P}$ as *obstacle vertices* and the edges of $\mathcal{P}$ as *obstacle edges*.

For any point $p$ in $\mathcal{P}$, let $Vis(p)$ denote the *visibility polygon* of $p$, i.e., $Vis(p)$ consists of all points $q \in \mathcal{P}$ such that $\overline{pq} \subseteq \mathcal{P}$. The size of $Vis(p)$ is $O(n)$ [19]. Define $Vis_{\mathcal{B}}(p) = Vis(p) \cap \mathcal{B}$, i.e., the portions of $\mathcal{B}$ that are visible to $p$, which consists of $O(n)$ segments on $\mathcal{B}$.

**Shortest path maps.** The *shortest path map* $SPM(s)$ of a point $s \in \mathcal{P}$ is a decomposition of $\mathcal{P}$ into cells such that all points $t$ in the cell $\sigma$ have the same anchor in any shortest $s$-$t$ path [23, 29]; the anchor is called the *root* of $\sigma$. Each edge of $SPM(s)$ is an obstacle edge fragment, an *extension segment* (i.e., extended from an obstacle vertex $u$ in the direction away from $v$, where $v$ is the anchor of $u$ in a shortest $s$-$u$ path), or a *bisector curve* which is the locus of points $p$ with $d(s,u) + |pu| = d(s,v) + |pv|$ for two obstacle vertices $u$ and $v$. See Fig. 1. Although an extension segment can be viewed as a degenerated bisector curve, we use bisector curve to refer to the general case only. As such, any point on a bisector curve has two topologically different paths from $s$. As in [13], we call the intersection of two bisector curves a *triple point*, which has at least three topologically different shortest paths from $s$.

It is known that $SPM(s)$ is of size $O(n)$ [23, 30] and can be computed in $O(n \log n)$ time [23] or in $O(n + h \log h)$ time after $\mathcal{P}$ is triangulated [39] (note that triangulating $\mathcal{P}$ can be done in $O(n + h \log h)$ time by a recent algorithm of Chan [7]).

Recall that $SPM_{\mathcal{B}}(s)$ refers to the portion of $SPM(s)$ restricted to $\mathcal{B}$, the boundary of $\mathcal{P}$. Hence, $SPM_{\mathcal{B}}(s)$ consists of a set of segments each of which belongs to a cell of $SPM(s)$.

**Shortest path trees (SPT) and SPT-equivalence decompositions.** For a point $s \in \mathcal{P}$, the *shortest path tree*, denoted by $SPT(s)$, is a spanning tree of $s$ and the vertices of $\mathcal{P}$ such that the path in the tree from $s$ to any vertex is a shortest path in $\mathcal{P}$.

The *SPT-equivalence decomposition* of $\mathcal{P}$, denoted by $\Psi^{spt}$, is the subdivision of $\mathcal{P}$ into cells such that $SPT(s)$ is combinatorially the same for all points $s$ in the same cell. The decomposition $\Psi^{spt}$ can be obtained by overlaying $SPM(u)$ for all obstacle vertices $u$ of $\mathcal{P}$ [10]. Since each $SPM(u)$ is of size $O(n)$, the worst-case complexity of $\Psi^{spt}$ is $\Theta(n^4)$ [10].

Define $\Psi^{spt}_{\mathcal{B}}$ as the subdivision of $\mathcal{B}$ into segments so that $SPT(s)$ is combinatorially the same for all points $s$ in the same segment. It is not difficult to see that $\Psi^{spt}_{\mathcal{B}}$ can be obtained by overlapping the segments of $SPM_{\mathcal{B}}(u)$ for all obstacle vertices $u$. As each $SPM_{\mathcal{B}}(u)$ has $O(n)$ segments, $\Psi^{spt}_{\mathcal{B}}$ has $O(n^2)$ segments. This bound is also tight in the worst case [10]. As such, $\Psi^{spt}_{\mathcal{B}}$ can be computed in $O(n^2 \log n)$ time.

**SPM-equivalence decompositions.** We follow the definitions of $\Psi_{\mathcal{B}}(\mathcal{B})$, $\Psi_{\mathcal{B}}(\mathcal{P})$, $\Psi_{\mathcal{P}}(\mathcal{B})$, and $\Psi_{\mathcal{P}}(\mathcal{P})$ (which is $\Psi$) in Sect. 1. As all obstacle vertices are on $\mathcal{B}$, both $\Psi_{\mathcal{B}}(\mathcal{B})$ and $\Psi_{\mathcal{B}}(\mathcal{P})$ are refinements of $\Psi^{spt}_{\mathcal{B}}$ and thus their sizes are $\Omega(n^2)$ in the worst case. Similarly, both $\Psi_{\mathcal{P}}(\mathcal{B})$ and $\Psi_{\mathcal{P}}(\mathcal{P})$ are refinements of $\Psi^{spt}$ and thus their sizes are $\Omega(n^4)$ in the worst case.

**General position assumption.** For ease of exposition, we make the following general position assumption for $\mathcal{P}$: (1) For any obstacle vertex $s$, $SPM(s)$ does not have a vertex of degree larger than three; (2) for any point $s \in \mathcal{B}$, $SPM(s)$ does not have a vertex of degree larger than four; (3) for any point $s \in \mathcal{P} \setminus \mathcal{B}$, $SPM(s)$ does not have a vertex of degree larger than five; (4) every pair of obstacle vertices have a unique shortest path; (5) no three obstacle vertices are collinear. These assumptions, which are consistent with those used in previous work, make our discussions less tedious but can be lifted without affecting our results. For example, the first assumption implies that if $s$ is an obstacle vertex, then there are at most three shortest paths from $s$ to $t$ for any point $t \in \mathcal{P}$.

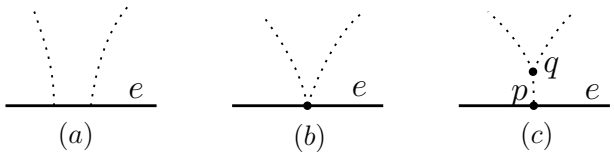


**Figure 2** Illustrating the combinatorial change of $SPM_{\mathcal{B}}(s)$.

## 3 The decomposition $\Psi_{\mathcal{B}}(\mathcal{B})$

To compute $\Psi_{\mathcal{B}}(\mathcal{B})$, we first compute $\Psi_{\mathcal{B}}^{spt}$ in $O(n^2 \log n)$ time as discussed in Section 2.

**Events.** Recall that $\Psi_{\mathcal{B}}(\mathcal{B})$ is a refinement of $\Psi_{\mathcal{B}}^{spt}$. Consider a segment $e_s$ of $\Psi_{\mathcal{B}}^{spt}$. Suppose we move a point $s$ on $e_s$ from one endpoint to the other. $SPM_{\mathcal{B}}(s)$ will change topologically in the following two situations; we call $s$ an *event point* when either situation happens.

1. At the event point, two bisector curves of $SPM(s)$ have their endpoints on $\mathcal{B}$ meet at an interior point of an obstacle edge $e$. Right before $s$ reaches the event point, the two bisector curves have their endpoints both on $e$ (i.e., as $s$ moves towards the event point, the two bisector endpoints move closer on $e$). See Fig. 2, where (a), (b), and (c) illustrate the scenarios when $s$ is before, at, and after the event point, respectively.
2. This case can be considered the inverse of the first case. At the event point, two bisector curves of $SPM(s)$ have their endpoints on $\mathcal{B}$ meet at an interior point of an obstacle edge $e$. Right before $s$ reaches the event point, the two bisector curves intersect at a triple point $q$ in the interior of $\mathcal{P}$ (i.e., there is a third bisector curve $\gamma$ whose endpoint is $q$ and whose other endpoint is at a point $p \in e$, and as $s$ moves towards the event point, $q$ moves on $\gamma$ towards $p$). See Fig. 2, where (c), (b), and (a) illustrate the scenarios when $s$ is before, at, and after the event point, respectively.

When $s$ moves between two adjacent event points on $e_s$, $SPM_{\mathcal{B}}(s)$ does not change combinatorially. Hence, to construct $\Psi_{\mathcal{B}}(\mathcal{B})$, it suffices to compute all event points on all segments $e_s$ of $\Psi_{\mathcal{B}}^{spt}$. In the following, we present an algorithm to compute all event points in $O(n^{4+\epsilon})$ time. Our approach also establishes an upper bound $O(n^{4+\epsilon})$ for $|\Psi_{\mathcal{B}}(\mathcal{B})|$.

### 3.1 Defining functions

For each obstacle vertex $u$, we define a function $f_u$ as follows. Recall that $Vis_{\mathcal{B}}(u)$ is the portion of the visibility polygon $Vis(u)$ of $u$ on $\mathcal{B}$.

▶ **Definition 1.** *For any obstacle vertex $u$, for any point $s \in Vis_{\mathcal{B}}(u)$ and any point $t \in \mathcal{B}$, define $f_u(s,t) = d_{u,v_t}(s,t)$, where $v_t$ is the root of the cell of $SPM(u)$ that contains $t$.*

For simplicity, we assume $f_u(s,t) = \infty$ if $s$ is not visible to $u$. If we use the positions of $s$ and $t$ on $\mathcal{B}$ as variables, then $f_u(s,t)$ is a bivariate piecewise algebraic function of constant degree (but not necessarily of constant complexity). Also, $f_u(s,t)$ defines a graph that is a surface patch in $\mathbb{R}^3$ in which the first two coordinates correspond to the positions of $s$ and $t$ on $\mathcal{B}$ and the third coordinate is $f_u(s,t)$. A convenient way to think about this is to place all edges of $\mathcal{B}$ on a line, in which $s$ and $t$ change. More specifically, obstacles can be arbitrarily ordered, but edges of the same obstacle of $\mathcal{P}$ should be placed consecutively following their order along the obstacle boundary (refer to [5] for a detailed treatment about this). We refer to the copy of the line for $s$ (resp., $t$) as the *$s$-axis* (resp., *$t$-axis*); they together form the $st$-plane. We also use $f_u(s,t)$ to refer to the graph or the surface patch defined by it.

For the function $f_u(s,t)$, while the domain of $t$ is the entire $\mathcal{B}$, the domain of $s$ comprises the $O(n)$ segments of $\mathit{Vis}_{\mathcal{B}}(u)$. We call the endpoints of the segments of $\mathit{Vis}_{\mathcal{B}}(u)$ the *breakpoints induced by* $u$ on the $s$-axis. Also, for each segment of $\mathit{SPM}_{\mathcal{B}}(u)$, the vertex $v_t$ (as in Definition 1) is the same for all points $t$ on the segment. We call the endpoints of the segments of $\mathit{SPM}_{\mathcal{B}}(u)$ the *breakpoints induced by* $u$ on the $t$-axis. For each segment $\sigma_s \in \mathit{Vis}_{\mathcal{B}}(u)$ and each segment $\sigma_t \in \mathit{SPM}_{\mathcal{B}}(u)$, $f_u(s,t) = d_{u,v}(s,t)$ for any $s \in \sigma_s$ and $t \in \sigma_t$, where $v$ is the root of the cell of $\mathit{SPM}(u)$ containing $\sigma_t$, and thus $f_u(s,t)$ is of constant complexity on $s \in \sigma_s$ and $t \in \sigma_t$; we refer to it as an *elementary function*. As such, since both $\mathit{Vis}_{\mathcal{B}}(u)$ and $\mathit{SPM}_{\mathcal{B}}(u)$ have $O(n)$ segments, $f_u(s,t)$ can be decomposed into $O(n^2)$ elementary functions each of which is defined on $s \in \sigma_s$ and $t \in \sigma_t$ for a segment $\sigma_s$ of $\mathit{Vis}_{\mathcal{B}}(u)$ and a segment $\sigma_t$ of $\mathit{SPM}_{\mathcal{B}}(u)$. For differentiation, we refer to the original function $f_u(s,t)$ for $s \in \mathit{Vis}_{\mathcal{B}}(u)$ and $t \in \mathcal{B}$ as a *compound function*.

Let $F$ denote the set of compound functions $f_u(s,t)$ for all obstacle vertices $u$. For convenience, we also consider $F$ the set of elementary functions of all these compound functions. As such, $F$ has $O(n^3)$ elementary functions.

**Lower envelope.** Define $\mathcal{L}(F)$ to be the lower envelope of all functions of $F$. By definition, for two points $s,t \in \mathcal{B}$, there is a unique point $p \in \mathcal{L}(F)$ whose projection onto the $st$-plane is $(s,t)$. Further, if $s$ is not visible to $t$, then the third coordinate of $p$ must be equal to $d(s,t)$, because a shortest $s$-$t$ path must contain an obstacle vertex.

Due to our general position assumption, the common intersection of three elementary functions of $F$ is a vertex of $\mathcal{L}(F)$. The following observation (with proof in Appendix A) follows from our general position assumption.

▶ **Observation 2.** *Every 3 elementary functions of $F$ have $O(1)$ common intersection points.*

Observation 3 (with proof in Appendix B) reduces the problem of bounding $|\Psi_{\mathcal{B}}(\mathcal{B})|$ to bounding $|\mathcal{L}(F)|$, by showing every event point of $\Psi_{\mathcal{B}}(\mathcal{B})$ corresponds to a vertex of $\mathcal{L}(F)$.

▶ **Observation 3.** *Every event point of $\Psi_{\mathcal{B}}(\mathcal{B})$ corresponds to a distinct vertex of $\mathcal{L}(F)$, and $|\Psi_{\mathcal{B}}(\mathcal{B})| = O(n^2 + |\mathcal{L}(F)|)$.*

In what follows, we focus on computing the vertices of $\mathcal{L}(F)$. As $F$ has $O(n^3)$ elementary bivariate functions, applying the result of [18, 34, 35] directly can give $|\mathcal{L}(F)| = O(n^{6+\epsilon})$ and compute all vertices in $O(n^{6+\epsilon})$. In the following, we present an improved algorithm of $O(n^{4+\epsilon})$ time and prove the bound $|\mathcal{L}(F)| = O(n^{4+\epsilon})$.

## 3.2 Computing the vertices of $\mathcal{L}(F)$

For convenience, when considering functions $f_u(s,t)$ of $F$, we use $\mathcal{B}_s$ (resp., $\mathcal{B}_t$) to refer to the copy of $\mathcal{B}$ for $s$ (resp., $t$), i.e., $s$ changes on $\mathcal{B}_s$ while $t$ changes on $\mathcal{B}_t$.

For a line segment $e_s$ on an obstacle edge of $\mathcal{B}_s$ and a line segment $e_t$ on an obstacle edge of $\mathcal{B}_t$, let $\mathcal{L}(F)[e_s, e_t]$ denote the portion of $\mathcal{L}(F)$ of the functions $f_u(s,t) \in F$ with $s \in e_s$ and $t \in e_t$. We will partition $\mathcal{B}_t$ into $O(n)$ line segments $e_t$ and present an algorithm to compute all vertices of $\mathcal{L}(F)[\mathcal{B}_s, e_t]$ in $O(n^{3+\epsilon})$ time. Doing this for all $O(n)$ such segments $e_t$ of $\mathcal{B}_t$ will compute all vertices of $\mathcal{L}(F)$ in $O(n^{4+\epsilon})$ time. To compute the vertices of $\mathcal{L}(F)[\mathcal{B}_s, e_t]$, we partition $\mathcal{B}_s$ into $O(n)$ line segments $e_s$ and present an algorithm to compute all vertices of $\mathcal{L}(F)[e_s, e_t]$ in $O(n^{2+\epsilon})$ time. Doing this for all $O(n)$ such segments $e_s$ will compute all vertices of $\mathcal{L}(F)[\mathcal{B}_s, e_t]$ in $O(n^{3+\epsilon})$ time.

**The partition of $\mathcal{B}_t$.** Recall that there are $O(n^2)$ breakpoints on $\mathcal{B}_t$, which partition $\mathcal{B}_t$ into $O(n^2)$ line segments, and we call them *elementary intervals*. For each obstacle edge $e \subseteq \mathcal{B}_t$, starting from an endpoint of $e$, we group every $n$ adjacent elementary intervals on $e$ into a *super interval*, except that the last super interval may contain less than $n$ elementary intervals. As $\mathcal{B}_t$ has $O(n^2)$ elementary intervals and $O(n)$ obstacle edges, $\mathcal{B}_t$ is partitioned into $O(n)$ super intervals each of which consists of at most $n$ elementary intervals. This is our partition for $\mathcal{B}_t$. Note that all super intervals can be computed in $O(n^2)$ time, assuming that the shortest path maps of all obstacle vertices are available. Since the shortest path map of each obstacle vertex can be computed in $O(n \log n)$ time [23], we conclude that all super intervals of $\mathcal{B}_t$ can be obtained in $O(n^2 \log n)$ time.

Let $e_t$ be a super interval of $\mathcal{B}_t$. We next show that computing $\mathcal{L}(F)[\mathcal{B}_s, e_t]$ can be done in $O(n^{3+\epsilon})$ time. To this end, we first introduce our partition for $\mathcal{B}_s$ with respect to $e_t$.

**The partition of $\mathcal{B}_s$.** Recall that $e_t$ contains at most $n$ elementary intervals. For notational convenience, we assume that $e_t$ contains exactly $n$ elementary intervals. Then, $e_t$ has $n-1$ breakpoints in its interior (these breakpoints partition $e_t$ into $n$ elementary intervals). For each obstacle vertex $u$, let $k_u$ denote the number of breakpoints of $e_t$ induced by $u$. As such, we have $\sum_{u \in V} k_u = n-1$ (recall that $V$ is the set of all obstacle vertices of $\mathcal{P}$).

Recall that $\mathcal{B}_s$ has $O(n^2)$ breakpoints, which partition $\mathcal{B}_s$ into $O(n^2)$ line segments, and we also call them *elementary intervals* for $s$. For each breakpoint $p$, we define $k_p = k_u$, where $u$ is the obstacle vertex that induces $p$. Consider an obstacle edge $e \subseteq \mathcal{B}_s$. Let $a$ and $b$ be the two endpoints of $e$. Starting from the elementary interval containing $a$, we group adjacent elementary intervals on $e$ as many as possible into a *super interval* $\xi$ until one of the following two cases to happen (the last elementary interval that causes one of these cases happens is included in $\xi$): (1) the sum of $k_p$ of all breakpoints in the interior of $\xi$ is larger than $2n$; (2) $\xi$ contains $b$. If $\xi$ does not contain $b$, then starting from the next elementary interval on $e$ after $\xi$, we create another super interval. We continue this until $b$ is included in last super interval. We create super intervals on other obstacle edges of $\mathcal{B}_s$ in the same way. Since $k_p \leq n-1$ for each breakpoint $p$, we obtain Lemma 4 (with proof in Appendix C).

▶ **Lemma 4.** *For each super interval of $\mathcal{B}_s$, the sum of $k_p$ of all breakpoints $p$ in its interior is no more than $3n$.*

Since $\sum_{u \in V} k_u = n-1$, we have the following lemma (with proof in Appendix D).

▶ **Lemma 5.** *The number of super intervals of $\mathcal{B}_s$ is $O(n)$.*

It is not difficult to see that the super intervals of $\mathcal{B}_s$ can be easily computed in $O(n^2)$ time, assuming that all breakpoints and their induced vertices are already available. The breakpoints can be obtained by computing the visibility polygons $Vis(u)$ of all obstacle vertices $u \in V$. As computing $Vis(u)$ for each $u$ can be done in $O(n \log n)$ time [19], we conclude that all super intervals of $\mathcal{B}_s$ can be computed in $O(n^2 \log n)$ time.

The following crucial lemma (whose proof is in Appendix E) leads to our result.

▶ **Lemma 6.** *For any super interval $e_s$ of $\mathcal{B}_s$, $\mathcal{L}(F)[e_s, e_t]$ has $O(n^{2+\epsilon})$ vertices and all vertices can be computed in $O(n^{2+\epsilon})$ time.*

With the above lemma, since $\mathcal{B}_s$ has $O(n)$ super intervals, it follows that $\mathcal{L}(F)[\mathcal{B}_s, e_t]$ has $O(n^{3+\epsilon})$ vertices and they can be computed in $O(n^{3+\epsilon})$ time. Furthermore, as $\mathcal{B}_t$ has $O(n)$ super intervals $e_t$, we conclude that $\mathcal{L}(F)[\mathcal{B}_s, \mathcal{B}_t]$, which is $\mathcal{L}(F)$, has $O(n^{4+\epsilon})$ vertices and they can be computed in $O(n^{4+\epsilon})$ time. We thus have the following theorem.

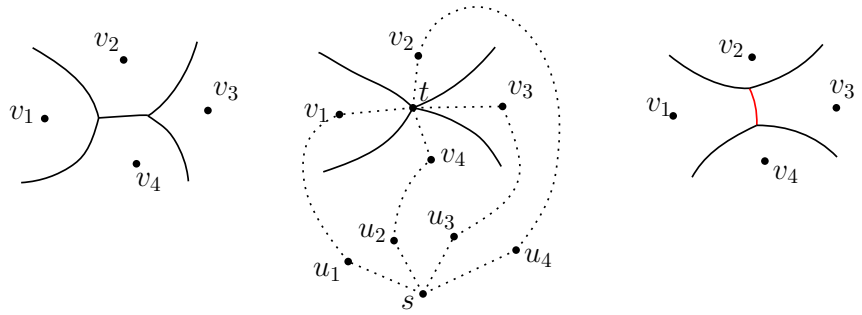


**Figure 3** The solid curves are bisector curves, which are defined by obstacle vertices $v_i$, $1 \leq i \leq 4$. (a) Before $s$ crosses an event point. (b) $s$ is at the event point: there are four shortest paths from $s$ to $t$, where a bisector contracts. (c) After $s$ passes the event point: the red curve is a new bisector curve.

▶ **Theorem 7.** *The lower envelope $\mathcal{L}(F)$ has $O(n^{4+\epsilon})$ vertices and all these vertices can be computed in $O(n^{4+\epsilon})$ time.*

Following our earlier discussion and Observation 3, we have the following result.

▶ **Corollary 8.** *The size of $\Psi_{\mathcal{B}}(\mathcal{B})$ is $O(n^{4+\epsilon})$ and $\Psi_{\mathcal{B}}(\mathcal{B})$ can be computed in $O(n^{4+\epsilon})$ time.*

## 4 The decomposition $\Psi_{\mathcal{B}}(\mathcal{P})$

We first present an algorithm to compute $\Psi_{\mathcal{B}}(\mathcal{P})$ in $O(|\Psi_{\mathcal{B}}(\mathcal{P})| \cdot \log n)$ time and then prove an $O(n^5)$ upper bound for its combinatorial complexity $|\Psi_{\mathcal{B}}(\mathcal{P})|$.

### 4.1 Algorithm for computing $\Psi_{\mathcal{B}}(\mathcal{P})$

Recall that $\Psi_{\mathcal{B}}(\mathcal{P})$ is a refinement of $\Psi_{\mathcal{B}}(\mathcal{B})$, which is a refinement of $\Psi_{\mathcal{B}}^{spt}$. We start with computing $\Psi_{\mathcal{B}}^{spt}$, which takes $O(n^2 \log n)$ time. Consider an edge $e_s$ of $\Psi_{\mathcal{B}}^{spt}$.

#### 4.1.1 Event points

Suppose that we move $s$ on $e_s$. Since $e_s$ is a segment of $\Psi_{\mathcal{B}}^{spt}$, as $s$ moves on $e_s$, there are two cases where combinatorial changes on $SPM(s)$ will happen (the location of $s$ where a combinatorial change of $SPM(s)$ happens is referred to as an *event point* for $s$). First (called the *boundary case*), a combinatorial change of $SPM(s)$ may happen on $\mathcal{B}$; in this case, the event point is also an event point for $\Psi_{\mathcal{B}}(\mathcal{B})$, which is the same as discussed in Section 3. Second (called the *interior case*), a combinatorial change of $SPM(s)$ may happen in the interior of $\mathcal{P}$. In this case, as discussed in [10], a bisector connecting two triple points contracts into a single point $t$ that has four shortest paths from $s$ (see Fig. 3). After the contraction, a new bisector is generated.

#### 4.1.2 Algorithm

We can compute all event points on $e_s$ as follows. Suppose initially we have $SPM(s)$ available when $s$ is at one endpoint of $e_s$. As $s$ moves on $e_s$, we maintain certain information.

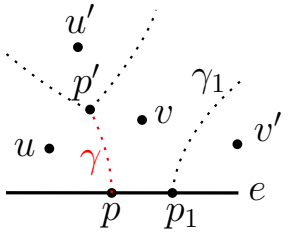


**Figure 4** $u$, $v$, $u'$, and $v'$ are obstacle vertices. $p'$ is a triple point defined by $u$, $v$, and $u'$.

**The boundary case.** To compute the boundary case event points, for each bisector curve $\gamma$ of $SPM(s)$ that has an endpoint $p$ on an obstacle edge $e$, we maintain the following information. Let $u$ and $v$ be the two obstacle vertices that define $\gamma$, i.e., $u$ and $v$ are the roots of the two cells of $SPM(s)$ bounded by $\gamma$; see Fig. 4.

As $s$ moves on $e_s$, $p$ will move on $e$. Note that unless $u$ and $v$ are the two endpoints of $e$, during the moving, $p$ cannot become collinear with $u$ and $v$, since otherwise the shortest path tree $SPT(s)$ would change combinatorially, contradicting that $e_s$ is an edge of $\Psi_{\mathcal{B}}^{spt}$. For a similar reason, $p$ will not cross either endpoint of $e$ during the moving of $s$. The position of $p$ on $e$ can be parameterized by that of $s$ on $e_s$. Let $p_1$ be a neighboring bisector curve endpoint of $p$ on $e$ (if any), and let $\gamma_1$ be the bisector curve containing $p_1$ (one of the two defining obstacle vertices of $\gamma_1$ is in $\{u, v\}$); see Fig. 4. As $s$ moves on $e_s$, we calculate the position of $s$ (if any) where $p_1$ will meet $p$. To this end, one needs to solve an equation of algebraic functions of constant degree to obtain $O(1)$ positions for $s$. We take the smallest value larger than the current position of $s$, i.e., this is the next position for $s$ when $p$ meets $p_1$; let $s_p^1$ denote this position. If $p$ has another neighboring bisector curve endpoint $p_2 \in e$, then we calculate the corresponding position $s_p^2$ for $s$. In addition, let $p'$ be the other endpoint of $\gamma$. If $p'$ is a triple point of $SPM(s)$ (see Fig. 4), then $p'$ also moves as $s$ moves, but its position can also be represented a function of $s$. We calculate the next position of $s$ (if any) so that $p'$ meets $p \in e$ (i.e., the bisector curve connecting $p$ and $p'$ contracts; see the red curve $\gamma$ in Fig. 4); let $s_p^0$ denote the position. Among all three positions $s_p^0$, $s_p^1$, and $s_p^2$, we only maintain the closer one to the current position of $s$, denoted by $s_p$ and called a *candidate event* of $s$ for $p$. We add $s_p$ to an *event heap* $H_s$ (with the position of $s_p$ as the *key*).

**The interior case.** To compute the interior case event points, we main the following information. For each bisector curve of $SPM(s)$ whose both endpoints are triple points, we calculate the next position of $s$ on $e_s$ (if any) where it will contract to a single point as a candidate event point and add it to $H_s$.

**Main loop.** The main loop of the algorithm works as follows. In each iteration, we extract the first candidate event $s_p$ from the event heap $H_s$ and process it as follows. First, we report $s_p$ as a "true" event point. Then, depending on whether the event belongs to the boundary case or the interior case, we update the events in $H_s$ related to $p$ as follows.

In the boundary case, $s_p$ is defined by a bisector curve endpoint on $\mathcal{B}$. We do the following. Depending on whether $s_p$ is $s_p^0$, there are two cases. If $s_p$ is $s_p^0$, then after $s$ moves across $s_p$, $p$ will be replaced by two new bisector endpoints on $e$ (see Fig. 5). Specifically, let $v, u, w$ be the three obstacle vertices defining $p$ such that $u$ is in the middle when $s$ is at $s_p$. Then, after $s$ crosses $s_p$, the bisector curve defined by $w$ (resp., $v$) and $u$ has a new endpoint at $e$. For each such endpoint $p'$, we compute its candidate event and add it to $H_s$. Further, we update the

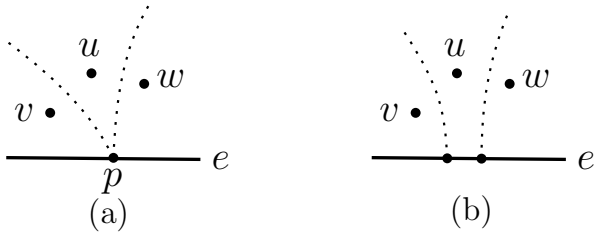


**Figure 5** (a) $s$ is at $s_p$; (b) after $s$ crosses $s_p$.

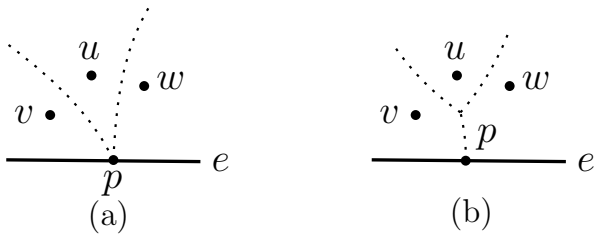


**Figure 6** (a) $s$ is at $s_p$; (b) after $s$ crosses $s_p$.

candidate event for the neighboring bisector event of $p'$, i.e, if $p'$ has a neighboring bisector curve endpoint $p_1$ on $e$, then we delete $p_1$'s candidate event from $H_s$, and recompute a new candidate event and insert it to $H_s$. If $s_p$ is either $s_p^1$ or $s_p^2$, we do the following. Without loss of generality, we assume that $s_p = s_p^1$. Hence, when $s$ is at $s_p$, $p$ has three anchors $v, u, w$ (see Fig. 6). After $s$ moves across $s_p$, a new bisector curve arises that is defined by two of $\{v, u, w\}$, say, $v$ and $w$, and the bisector curve has $p$ as one of its endpoint. Using $v$ and $w$, we recompute the candidate event of $s$ for this new endpoint $p$ and insert it to $H_s$. Further, for each of $p$'s neighboring bisector curve endpoint on $e$, we delete its candidate from $H_s$ and recompute a new candidate event and insert it to $H_s$.

In the interior case, $s_p$ is defined by a bisector curve of $SPM(s)$ connecting two triple points (see Fig. 3). We update $SPM(s)$ by removing the bisector curve and adding the new generated bisector. This is only a local change to $SPM(s)$ and costs $O(1)$ time since the degree of each vertex of $SPM(s)$ is at most 4 by our general position assumption. After the update of $SPM(s)$, for each bisector curve that has one of its incident triple points updated (including the new generated bisector), we update its candidate event in $H_s$. Again, only $O(1)$ bisector curves are affected.

The algorithm stops once $s$ reaches the other endpoint of $e_s$, at which time all event points on $e_s$ are computed. The proof of Theorem 9 in Appendix F gives the time analysis.

▶ **Theorem 9.** *The decomposition $\Psi_{\mathcal{B}}(\mathcal{P})$ can be computed in $O((n^2 + |\Psi_{\mathcal{B}}(\mathcal{P})|) \cdot \log n)$ time.*

Theorem 10 proves $|\Psi_{\mathcal{B}}(\mathcal{P})| = O(n^5)$ (see Appendix G for the proof), by a simple brute-force argument. Similar ideas have been used previously [3, 30, 38].

▶ **Theorem 10.** *The combinatorial complexity of $\Psi_{\mathcal{B}}(\mathcal{P})$ is $O(n^5)$.*

Combining Theorems 9 and 10 leads to the following result.

▶ **Corollary 11.** *The decomposition $\Psi_{\mathcal{B}}(\mathcal{P})$ can be computed in $O(n^5 \log n)$ time.*

# 5 Applications

In this section, we discuss a few "representative" applications of our results on $\Psi_{\mathcal{B}}(\mathcal{B})$ and $\Psi_{\mathcal{P}}(\mathcal{B})$. See the full paper for other applications.

## 5.1 Two-point shortest path queries

For the two-point shortest path query problem, we refer to it as the $\mathcal{B}$-$\mathcal{B}$ *case* if both query points $s$ and $t$ are required to be on $\mathcal{B}$, the $\mathcal{B}$-$\mathcal{P}$ *case* if only one of the query point is required to be on $\mathcal{B}$, and the *general case* otherwise. We discuss how to use our result for $\Psi_{\mathcal{P}}(\mathcal{B})$ to solve the $\mathcal{B}$-$\mathcal{P}$ case problem; other cases can be treated similarly.

In the preprocessing, we construct the SPM-equivalent decomposition $\Psi_{\mathcal{B}}(\mathcal{P})$. For each segment $\sigma$ of $\Psi_{\mathcal{B}}(\mathcal{P})$, following the method of Chiang and Mitchell [10], we construct a parameterized shortest path map $SPM(\sigma)$ for points $s \in \sigma$, which can answer each $\mathcal{B}$-$\mathcal{P}$ case two-point shortest path query in $O(\log n)$ time for two query points $s \in \sigma$ and $t \in \mathcal{P}$ (note that we also need to build a parameterized point location data structure on $SPM(\sigma)$ as discussed in [10]). As such, after building a parameterized shortest path map for each segment of $\Psi_{\mathcal{B}}(\mathcal{P})$, each $\mathcal{B}$-$\mathcal{P}$ case two-point shortest path query can be answered in $O(\log n)$ time. Constructing a parameterized shortest path map can be done in $O(n \log n)$ time and $O(n)$ space [10]. Hence, the total preprocessing time and space are bounded by $O(|\Psi_{\mathcal{B}}(\mathcal{P})| \cdot n \log n)$ and $O(|\Psi_{\mathcal{B}}(\mathcal{P})| \cdot n)$ space, respectively. The $O(\log n)$ factor can be reduced by observing that there are only $O(1)$ changes between two shortest path maps of adjacent cells (hence, if we have the map for one cell, obtaining the map for an adjacent cell can be easily done in $O(n)$ time). As in [10], using persistent data structures, the preprocessing can be further improved by roughly a linear factor, but the query time grows to $O(\log^2 n)$. Using our bound $|\Psi_{\mathcal{B}}(\mathcal{P})| = O(n^5)$ and our $O(n^5 \log n)$ time algorithm for computing $\Psi_{\mathcal{B}}(\mathcal{P})$, we can obtain two data structures for the $\mathcal{B}$-$\mathcal{P}$ case two-point shortest path query problem with the following complexities respectively: $O(n^6, n^6, \log n)$ and $O(n^5 \log n, n^5 \log n, \log^2 n)$. This improves the result of $O(n^{6+\epsilon}, n^{6+\epsilon}, \log^2 n)$ from [6].

## 5.2 Geodesic diameter

We demonstrate an application of $\Psi_{\mathcal{B}}(\mathcal{B})$ on computing the $\mathcal{B}$-$\mathcal{B}$ *case* geodesic diameter, i.e., computing a diametral pair $(s, t)$ with both $s, t \in \mathcal{B}$. In this case, Bae, Korman, and Okamoto [3] proved that any diametral pair $(s, t)$ has three different shortest $s$-$t$ paths. This observation implies that $(s, t)$ must correspond to an event point in the SPM-decomposition $\Psi_{\mathcal{B}}(\mathcal{B})$, i.e., $s$ is the event point while $t$ is the point on $\mathcal{B}$ where $SPM_{\mathcal{B}}(s)$ changes. As such, once $\Psi_{\mathcal{B}}(\mathcal{B})$ is computed, all $\mathcal{B}$-$\mathcal{B}$ case diametral pairs are available. More specifically, if $s$ is an event point $\Psi_{\mathcal{B}}(\mathcal{B})$, then $s$ is created due to a combinatorial change at a point $t$ on $\mathcal{B}$; we keep $(s, t)$ as a *candidate* diametral pair (and also keep their geodesic distance $d(s, t)$). We can slightly modify our algorithm for constructing $\Psi_{\mathcal{B}}(\mathcal{B})$ to compute all these candidate diametral pairs and their geodesic distances. The total time of the algorithm is still $O(n^{4+\epsilon})$. After having all these candidate diametral pairs, we return all pairs whose geodesic distances are the largest. As such, all $\mathcal{B}$-$\mathcal{B}$ case diametral pairs can be computed in $O(n^{4+\epsilon})$ time. For comparison, the method of [3] solves the problem in $O(n^5 \log n \log^* n)$ time.

Computing the $\mathcal{B}$-$\mathcal{P}$ *case* geodesic diameter can be done in a similar way in $O(n^5 \log n)$ time by slightly modifying our algorithm for $\Psi_{\mathcal{B}}(\mathcal{P})$. The algorithm in [3] runs in $O(n^{5.91})$ time for this case.

## 5.3 Geodesic center

We discuss the geodesic center problem. For two subsets $A, B \subseteq \mathcal{P}$, we use the *$A$-$B$ case* to refer to the problem in which one wishes to find a point $s \in A$ that minimizes the largest geodesic distance from $s$ to any point $t \in B$; we call $s$ an *$A$-restricted center with respect to $B$* and the geodesic distance between $s$ and its farthest point in $B$ is the corresponding *geodesic radius*. For example, the $\mathcal{B}$-$\mathcal{P}$ case refers to the problem of computing a point $s \in \mathcal{B}$ that minimizes the largest geodesic distance from $s$ to any point $t \in \mathcal{P}$. In particular, the *general case* refers to the $\mathcal{P}$-$\mathcal{P}$ case.

In the following, we mainly discuss the $\mathcal{B}$-$\mathcal{P}$ case (the algorithm for the $\mathcal{B}$-$\mathcal{B}$ case is very similar, but simpler). A useful observation that was proved by Bae, Korman, and Okamoto [4] is the following: For any point $s \in \mathcal{P}$, any of its farthest points must be a vertex of $SPM(s)$.

We first construct the SPM-equivalent decomposition $\Psi_{\mathcal{B}}(\mathcal{P})$. For each segment $\sigma$ of $\Psi_{\mathcal{B}}(\mathcal{P})$, by definition, when $s$ moves on $\sigma$, $SPM(s)$ does not change combinarially, and in particular, $SPM(s)$ has the same set of vertices. As $s$ moves on $\sigma$, for each vertex $t \in SPM(s)$, the geodesic distance $d(s,t)$ is a constant-degree univariate function of the position of $s \in \sigma$. Let $\mathcal{U}(\sigma)$ denote the upper envelope of the functions of all vertices $t$ of $SPM(s)$. We consider the problem of finding a $\sigma$-restricted center $s \in \sigma$ with respect to $\mathcal{P}$ and its corresponding geodesic radius. Observe that such a center must correspond to a lowest vertex in $\mathcal{U}(\sigma)$ and its geodesic radius is the height of the vertex. As such, an $\sigma$-restricted center and its radius can be easily found once $\mathcal{U}(\sigma)$ is constructed. The latter task can be done in $O(\lambda_{O(1)}(n)\log n)$ time [35] since there are $O(n)$ constant-degree algebraic functions of constant size, where $\lambda_{O(1)}(n)$ is the maximum length of a $DS(n, O(1))$-sequence and is bounded by $O(n\log^* n)$ (for notational simplicity, we will use the bound $O(n\log^* n)$ instead in the following). Constructing $\mathcal{U}(\sigma)$ for all segments $\sigma \in \Psi_{\mathcal{B}}(\mathcal{P})$ will compute all $\sigma$-restricted centers and their corresponding geodesic radii. Among them, we return those whose radii are the smallest. The total time is thus $O(|\Psi_{\mathcal{B}}(\mathcal{P})| \cdot n\log n\log^* n)$, which is bounded by $O(n^6\log n\log^* n)$ using our bound $|\Psi_{\mathcal{B}}(\mathcal{P})| = O(n^5)$.

We next improve the runtime by a linear factor. For each segment $\sigma \in \Psi_{\mathcal{B}}(\mathcal{B})$, let $F(\sigma)$ denote the set of functions $d(s,t)$ for all vertices $t$ of $SPM(\sigma)$, where $SPM(\sigma)$ denote the shortest path map $SPM(s)$ for points $s \in \sigma$. Our main idea is based on the observation that for two adjacent segments $\sigma_1$ and $\sigma_2$ of $\Psi_{\mathcal{B}}(\mathcal{P})$, their shortest path maps differs by at most $O(1)$ vertices. Hence, the set difference between $F(\sigma_1)$ and $F(\sigma_2)$ is $O(1)$. The details are given below.

Consider an obstacle edge $e$. We consider the problem of finding an $e$-restricted center $s \in e$ with respect to $\mathcal{P}$ and its corresponding geodesic radius. We consider the segments of $\Psi_{\mathcal{B}}(\mathcal{P})$ on $e$ in order. Let $\sigma_1$ and $\sigma_2$ be the first two segments. Suppose we already have the functions $F(\sigma_1)$, defined on $\sigma_1$. For each function in $F(\sigma_1) \cap F(\sigma_2)$, instead of creating a new function for $F(\sigma_2)$, we simply extend the domain of the function from $s \in \sigma_1$ to $s \in \sigma_1 \cup \sigma_2$. In this way, since $F(\sigma_1)$ and $F(\sigma_2)$ differs by $O(1)$, we only need to create $O(1)$ new functions for $F(\sigma_2)$. As such, the total number of functions of all segments of $\Psi_{\mathcal{B}}(\mathcal{P})$ on $e$ is $O(n+\kappa(e))$, where $\kappa(e)$ is the number of segments of $\Psi_{\mathcal{B}}(\mathcal{P})$ on $e$. Each $e$-restricted geodesic center corresponds to a lowest vertex of the upper envelope $\mathcal{U}(e)$ of all these functions (again the height of the vertex is the corresponding geodesic radius). Since these are $O(n+\kappa(e))$ constant-degree functions of constant size, the upper envelope $\mathcal{U}(e)$ can be computed in $O((n+\kappa(e))\log n\log^* n)$ time. Doing this for all obstacle edges $e$ will compute all $\mathcal{B}$-$\mathcal{P}$ case geodesic centers (i.e., among all edge restricted centers, we return those with smallest geodesic radii). Since $\sum_{e\in\mathcal{B}}\kappa(e) = |\Psi_{\mathcal{B}}(\mathcal{P})|$, the total time is bounded by $O((n^2+|\Psi_{\mathcal{B}}(\mathcal{P})|)\log n\log^* n)$. As $|\Psi_{\mathcal{B}}(\mathcal{P})| = \Omega(n^2)$ in the worst case, we obtain that

all $\mathcal{B}$-$\mathcal{P}$ case geodesic centers can be computed in $O(|\Psi_{\mathcal{B}}(\mathcal{P})| \log n \log^* n)$ time, which is $O(n^5 \log n \log^* n)$ with our $O(n^5)$ bound for $|\Psi_{\mathcal{B}}(\mathcal{P})|$. For comparison, the algorithm of [38] solves this problem in $O(n^8 \log n)$ time.

**The $\mathcal{B}$-$\mathcal{B}$ case.** For the $\mathcal{B}$-$\mathcal{B}$ case geodesic centers, we follow the similar approach. The difference is the we use the decomposition $\Psi_{\mathcal{B}}(\mathcal{B})$ instead. As such, all $\mathcal{B}$-$\mathcal{B}$ case geodesic centers can be computed in $O(n^{4+\epsilon})$ time. The algorithm of [38] for this case runs in $O(n^8 \log n)$ time.

## References


1 Hee-Kap Ahn, Luis Barba, Prosenjit Bose, Jean-Lou de Carufel, Matias Korman, and Eunjin Oh. A linear-time algorithm for the geodesic center of a simple polygon. *Discrete and Computational Geometry*, 56:836–859, 2016. doi:10.1007/s00454-016-9796-0.

2 Tetsuo Asano and Godfried Toussaint. Computing the geodesic center of a simple polygon. Technical Report SOCS-85.32, McGill University, Montreal, Canada, 1985.

3 Sang Won Bae, Matias Korman, and Yoshio Okamoto. The geodesic diameter of polygonal domains. *Discrete and Computational Geometry*, 50:306–329, 2013. doi:10.1007/s00454-013-9527-8.

4 Sang Won Bae, Matias Korman, and Yoshio Okamoto. Computing the geodesic centers of a polygonal domain. *Computational Geometry: Theory and Applications*, 48:495–505, 2015. doi:10.1016/j.comgeo.2015.10.009.

5 Sang Won Bae and Yoshio Okamato. Querying two boundary points for shortest paths in a polygonal domain. *Computational Geometry: Theory and Applications*, 45:284–293, 2012. doi:10.1016/j.comgeo.2012.01.012.

6 Sarita de Berg, Tillmann Miltzow, and Frank Staals. Towards space efficient two-point shortest path queries in a polygonal domain. In *Proceedings of the 40th International Symposium on Computational Geometry (SoCG)*, pages 17:1–17:16, 2024. doi:10.4230/LIPIcs.SoCG.2024.17.

7 Timothy M. Chan. Triangulating a polygon with holes in optimal (deterministic) time. In *Proceedings of the 42nd International Symposium on Computational Geometry (SoCG)*, pages 28:1–28:13, 2026. doi:10.4230/LIPIcs.SOCG.2026.28.

8 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993. doi:10.1007/BF02189314.

9 Danny Z. Chen, John Hershberger, and Haitao Wang. Computing shortest paths amid convex pseudodisks. *SIAM Journal on Computing*, 42(3):1158–1184, 2013. doi:10.1137/110840030.

10 Yi-Jen Chiang and Joseph S. B. Mitchell. Two-point Euclidean shortest path queries in the plane. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 215–224, 1999. doi:10.5555/314500.314560.

11 James R. Driscoll, Neil Sarnak, Daniel D. Sleator, and Robert E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989. doi:10.1016/0022-0000(89)90034-2.

12 Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986. doi:10.1137/0215023.

13 Sylvester D. Eriksson-Bique, John Hershberger, Valentin Polishchuk, Bettina Speckmann, Subhash Suri, Topi Talvitie, Kevin Verbeek, and Hakan Yıldız. Geometric $k$ shortest paths. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1616–1625, 2015. doi:10.1137/1.9781611973730.107.

14 Subir K. Ghosh and David M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20(5):888–910, 1991. doi:10.1137/0220055.

**15** Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989. doi:10.1016/0022-0000(89)90041-X.

**16** Leonidas J. Guibas, John Hershberger, D. Leven, Micha Sharir, and Robert E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987. doi:10.1007/BF01840360.

**17** Hua Guo, Anil Maheshwari, and Jörg-Rüdiger Sack. Shortest path queries in polygonal domains. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM)*, pages 200–211, 2008. doi:10.1007/978-3-540-68880-8_20.

**18** Dan Halperin and Micha Sharir. New bounds for lower envelopes in three dimensions, with applications to visibility in terrains. *Discrete and Computational Geometry*, 12:313–326, 1994. doi:10.1007/BF02574383.

**19** Paul J. Heffernan and Joseph S. B. Mitchell. An optimal algorithm for computing visibility in the plane. *SIAM Journal on Computing*, 24(1):184–201, 1995. doi:10.1137/S0097539791221505.

**20** John Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38(5):231–235, 1991. doi:10.1016/0020-0190(91)90064-O.

**21** John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4(2):63–97, 1994. doi:10.1016/0925-7721(94)90010-8.

**22** John Hershberger and Subhash Suri. Matrix searching with the shortest-path metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997. doi:10.1137/S0097539793253577.

**23** John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999. doi:10.1137/S0097539795289604.

**24** John Hershberger, Subhash Suri, and Hakan Yıldız. A near-optimal algorithm for shortest paths among curved obstacles in the plane. *SIAM Journal on Computing*, 51:1296–1340, 2022. doi:10.1137/21M1428248.

**25** David G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983. doi:10.1137/0212002.

**26** Vladlen Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *Journal of the ACM*, 51:699–730, 2004. doi:10.1145/1017460.1017461.

**27** Der-Tsai Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984. doi:10.1002/net.3230140304.

**28** Anna Lubiw and Anurag Murty Naredla. The geodesic edge center of a simple polygon. *Discrete and Computational Geometry*, 29(74):944–998, 2025. doi:10.1007/s00454-025-00768-9.

**29** Joseph S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991. doi:10.1007/BF01530888.

**30** Joseph S. B. Mitchell. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry and Applications*, 6(3):309–332, 1996. doi:10.1142/S0218195996000216.

**31** Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete and Computational Geometry*, 4:611–626, 1989. doi:10.1007/BF02187751.

**32** Hans Rohnert. Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters*, 23(2):71–76, 1986. doi:10.1016/0020-0190(86)90045-1.

**33** Neil Sarnak and Robert E. Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29:669–679, 1986. doi:10.1145/6138.6151.

**34** Micha Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete and Computational Geometry*, 12:327–345, 1994. doi:10.1007/BF02574384.

**35** Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

36 James A. Storer and John H. Reif. Shortest paths in the plane with polygonal obstacles. *Journal of the ACM*, 41(5):982–1012, 1994. doi:10.1145/185675.185795.

37 Subhash Suri. Computing geodesic furthest neighbors in simple polygons. *Journal of Computer and System Sciences*, 39:220–235, 1989. doi:10.1016/0022-0000(89)90045-7.

38 Haitao Wang. On the geodesic centers of polygonal domains. *Journal of Computational Geometry*, 9:131–190, 2018. doi:10.20382/jocg.v9i1a5.

39 Haitao Wang. A new algorithm for Euclidean shortest paths in the plane. *Journal of the ACM*, 70:11:1–11:62, 2023. doi:10.1145/3580475.


## A Proof of Observation 2

**Proof.** Consider three elementary functions $f_{u_i}(s,t) = d_{u_i,v_i}(s,t) = |su_i|+d(u_i,v_i)+|v_it|$, for $i = 1,2,3$. A common intersection point of these three functions corresponds to a pair of points $(s,t)$ that is a solution of the equation $d_{u_1,v_1}(s,t) = d_{u_2,v_2}(s,t) = d_{u_3,v_3}(s,t)$. Since these are three different elementary functions, for any $i,j \in \{1,2,3\}$ with $i \neq j$, either $u_i \neq u_j$ or $v_i \neq v_j$. Due to our general position assumption, the equation $d_{u_1,v_1}(s,t) = d_{u_2,v_2}(s,t) = d_{u_3,v_3}(s,t)$ has $O(1)$ solutions. The observation thus follows. ◀

## B Proof of Observation 3

**Proof.** Recall that $\Psi_{\mathcal{B}}(\mathcal{B})$ consists of vertices of $\Psi_{\mathcal{B}}^{spt}$ plus event points in the interior of segments of $\Psi_{\mathcal{B}}^{spt}$. Since $\Psi_{\mathcal{B}}^{spt}$ has $O(n^2)$ vertices, to prove $|\Psi_{\mathcal{B}}(\mathcal{B})| = O(n^2 + |\mathcal{L}(F)|)$, it suffices to show that the number of event points is at most $|\mathcal{L}(F)|$. To this end, we show that every event point of $\Psi_{\mathcal{B}}(\mathcal{B})$ corresponds to a distinct vertex of $\mathcal{L}(F)$.

Consider such an event point $s \in \Psi_{\mathcal{B}}(\mathcal{B})$. By definition, there is a point $t \in \mathcal{B}$ such that there are three topologically different shortest $s$-$t$ paths $\overline{su_i} \cup \pi(u_i,v_i) \cup \overline{v_it}$ with $i = 1,2,3$. Hence, $(s,t,d(s,t))$ must be a point in $\mathbb{R}^3$ on the lower envelope $\mathcal{L}(F)$ and is also a common intersection of three elementary functions of $F$: $d_{u_i,v_i}(s,t)$, $1 \leq i \leq 3$. As such, $(s,t,d(s,t))$ must be a vertex of $\mathcal{L}(F)$.

Note that no other event point corresponds to the same vertex $(s,t,d(s,t))$. Indeed, to have another event point $s'$ also correspond to $(s,t,d(s,t))$, $s = s'$ must hold.

We also remark that it is possible that there is another point $t' \in \mathcal{B}$ such that $t' \neq t$ and there are three topologically different shortest $s$-$t'$ paths. In that case, $s$ corresponding to another vertex $(s,t',d(s,t'))$ of $\mathcal{L}(F)$. Hence, this observation also holds if an event point $s$ is caused by multiple topological changes of $SPM_{\mathcal{B}}(s)$, in which case $s$ is considered multiple event points that are co-located. ◀

## C Proof of Lemma 4

**Proof.** Consider a super interval $\xi$. If $\xi$ consists of only one elementary interval, then the observation trivially follows as there is no breakpoint in the interior of $\xi$. Otherwise, let $\xi' \subseteq \xi$ be the sub-interval of $\xi$ excluding the last elementary interval, denoted by $\xi''$. Then, $\xi'$ and $\xi''$ are separated by a breakpoint $p'$. By definition, the sum of $k_p$ of all breakpoints $p$ in the interior of $\xi'$ must be no more than $2n$. Since $k_{p'} \leq n-1$ and the sum of $k_p$ of all breakpoints $p$ in the interior of $\xi$ is equal to $k_{p'}$ plus the sum of $k_p$ of all breakpoints $p$ in the interior of $\xi'$, we obtain that the sum of $k_p$ of all breakpoints $p$ in the interior of $\xi$ is at most $3n$. ◀

## D Proof of Lemma 5

**Proof.** By definition, there are two cases for a super interval to be created. For the second case, observe that each obstacle edge can have at most one super interval under the second case. Hence, the number of second case super intervals is $O(n)$.

We now discuss the first case. For each obstacle vertex $u$, let $P_u$ denote the set of breakpoints induced by $u$. Note that $|P_u| = O(n)$. Recall that $\sum_{u \in V} k_u = n - 1$. Hence, the sum of $k_p$ of all breakpoints $p$ on $\mathcal{B}_s$ is $\sum_{u \in V} \sum_{p \in P_u} k_p = \sum_{u \in V} \sum_{p \in P_u} k_u = O(n) \cdot \sum_{u \in V} k_u = O(n^2)$. For each super interval $\xi$ in the first case, by definition, the total sum of $k_p$ of all breakpoints $p$ in the interior of $\xi$ is larger than $2n$. Therefore, the total number of first case super intervals on $\mathcal{B}_s$ is $O(n)$. The lemma thus follows. ◀

## E Proof of Lemma 6

**Proof.** Recall that $e_s$ lies on a single obstacle edge, so does $e_t$. To prove the lemma, we will show that the number of elementary functions $f_u(s,t)$ defined on $s \in e_s$ and $t \in e_t$ is $O(n)$. Recall that each elementary function is a bivariate algebraic function of constant degree and constant size. Applying the result of [18, 34, 35] on these $O(n)$ elementary functions can give $|\mathcal{L}(F)[e_s, e_t]| = O(n^{2+\epsilon})$ and compute all vertices of $\mathcal{L}(F)[e_s, e_t]$ in $O(n^{2+\epsilon})$ time.

We now argue that the number of elementary functions $f_u(s,t)$ defined on $s \in e_s$ and $e_t$ is $O(n)$. For any subsets $e'_s \subseteq e_s$ and $e'_t \subseteq e_t$, let $F[e'_s, e'_t]$ denote the set of elementary functions $f_u(s,t)$ defined on $s \in e'_s$ and $t \in e'_t$. Our goal is to prove $|F[e_s, e_t]| = O(n)$.

We order the elementary intervals of $e_s$ from one end to the other. We do the same for $e_t$. Consider the first elementary interval $I_s$ of $e_s$ and the first elementary interval $I_t$ of $e_t$. Since $I_s$ is an elementary interval of $\mathcal{B}_s$ and $I_t$ is an elementary interval of $\mathcal{B}_t$, $|F[I_s, I_t]| \leq n$ holds since there are $n$ obstacle vertices. Let $m = |F[I_s, I_t]|$. Let $I'_t$ be the elementary interval of $e_t$ adjacent to $I_t$. By definition, $I_t$ and $I'_t$ are two line segments on the same obstacle edge separated by a breakpoint of $\mathcal{B}_t$. Hence, comparing $F[I_s, I'_t]$ to $F[I_s, I_t]$, at most one function of $F[I_s, I_t]$ is changed to a new function in $F[I_s, I'_t]$ while all other functions of $F[I_s, I_t]$ are still in $F[I_s, I'_t]$ (i.e., each of these functions is originally defined on $t \in I_t$ but we can extend its domain to $t \in I_t \cup I'_t$). As such, $F[I_s, I_t \cup I'_t]$ has $m + 1$ elementary functions. If we continue this analysis for the remaining elementary intervals of $e_t$, we obtain that $F[I_s, e_t]$ has at most $m + n \leq 2n$ elementary functions since $e_t$ has at most $n$ elementary intervals.

Now consider the elementary interval $I'_s$ of $e_s$ adjacent to $I_s$. Let $p$ be the breakpoint that separates $I_s$ and $I'_s$. We can follow a similar analysis to the above. When we move from $I_s$ to $I'_s$, at most $k_p$ functions are changed from $F[I_s, e_t]$ to $F[I'_s, e_t]$ while the other functions of $F[I_s, e_t]$ are still in $F[I'_s, e_t]$ (each of them is originally defined on $s \in I_s$ but we can extend their domain to $s \in I_s \cup I'_s$). Hence, the number of functions in $F[I_s \cup I'_s, e_t]$ is at most $k_p + |F[I_s, e_t]| \leq k_p + 2n$. If we continue the same analysis for the rest of the elementary intervals of $e_s$, then since the sum of $k_p$ for all breakpoints $p$ in the interior of $e_s$ is at most $3n$ by Lemma 4, we can obtain that $|F[e_s, e_t]| = O(n)$.

Note that following the above analysis, we can easily find all functions of $F[e_s, e_t]$ in $O(n)$ time, after which we can apply the algorithm of [18, 34, 35] to compute all vertices of $\mathcal{L}(F)[e_s, e_t]$ in $O(n^{2+\epsilon})$ time. The lemma thus follows. ◀

## F Proof of Theorem 9

**Proof.** We analyze the time complexity of our algorithm. First notice that $|H_s|$ is always bounded by $O(n)$ since the size of $SPM(s)$ is $O(n)$. Handling each event takes $O(\log n)$ time as each event only involves local changes on $O(1)$ bisector curves. As such, each iteration

can be performed in $O(\log n)$ time. Hence, the runtime for computing all event points on $e_s$ is $O(\kappa(e_s) \cdot \log n)$, where $\kappa(e_s)$ is the number of event points on $e_s$. If we construct $SPM(s)$ initially when $s$ is at an endpoint of $e_s$, which takes $O(n \log n)$ time, then the total time for computing the event points on $e_s$ is $O(n \log n + \kappa(e_s) \cdot \log n)$. Applying the algorithm to all segments $e_s \in \Psi_{\mathcal{B}}^{spt}$ will construct $\Psi_{\mathcal{B}}(\mathcal{P})$ in total $O(n^3 \log n + |\Psi_{\mathcal{B}}(\mathcal{P})| \log n)$ time since $\Psi_{\mathcal{B}}^{spt}$ has $O(n^2)$ segments.

We can further reduce the factor $O(n^3 \log n)$ to $O(n^2 \log n)$ as follows. Notice that the factor is dominated by the time for computing $SPM(s)$ initially for each $e_s \in \Psi_{\mathcal{B}}^{spt}$. Instead of doing this for each segment of $\Psi_{\mathcal{B}}^{spt}$, we process all segments of $\Psi_{\mathcal{B}}^{spt}$ on the same obstacle edge $e$ all together. Specifically, initially, we compute $SPM(s)$ when $s$ is at an endpoint of $e$. Then, we process the first segment $e_s$ of $\Psi_{\mathcal{B}}^{spt}$ on $e$. After $e_s$ is processed, we already have $SPM(s)$ available for the current position $s$, which is at the common endpoint of $e_s$ and the next segment $e'_s$. As such, before we process $e'_s$, we do not have to recompute the $SPM(s)$ again. In this way, we only need to compute $SPM(s)$ from scratch once for each obstacle edge, which takes $O(n^2 \log n)$ time in total. Therefore, the overall runtime of the algorithm can be bounded by $O(n^2 \log n + |\Psi_{\mathcal{B}}(\mathcal{P})| \log n)$. ◀

## G Proof of Theorem 10

**Proof.** According to our previous discussion, when $s$ moves on a segment $e_s$ of $\Psi_{\mathcal{B}}^{spt}$, there are two type of events: (1) The event is caused by a combinatorial change of $SPM(s)$ on $\mathcal{B}$; (2) the event is caused by a combinatorial change of $SPM(s)$ in the interior of $\mathcal{P}$.

The first type of events is actually included in $\Psi_{\mathcal{B}}(\mathcal{B})$, whose complexity is $O(n^{4+\epsilon})$ by Corollary 8. Below, we show that the number of the second type events is $O(n^5)$.

For every four (not necessarily distinct) obstacle vertices $v_i$, $1 \leq i \leq 4$, we do the following. Recall that $SPM_{\mathcal{B}}(v_i)$ has $O(n)$ segments on $\mathcal{B}$. We overlay $SPM_{\mathcal{B}}(v_i)$, for all $1 \leq i \leq 4$, resulting in $O(n)$ segments on $\mathcal{B}$. For each such segment $e$, for each $1 \leq i \leq 4$, $e$ is contained in a single cell of $SPM(v_i)$ and let $u_i$ be the root of the cell. Consider the equation $d_{u_1,v_1}(s,t) = d_{u_2,v_2}(s,t) = d_{u_3,v_3}(s,t) = d_{u_4,v_4}(s,t)$, for $s \in e$ and $t \in \mathbb{R}^2$. Observe that each second type event with $s \in e$ corresponds to a pair $(s,t)$ with $t$ in the interior of $\mathcal{P}$ such that there are four shortest $s$-$t$ paths (see Fig. 3(b)). If the anchors of $t$ in the four shortest paths are $v_i$, $1 \leq i \leq 4$, respectively, then we say that $s$ is *defined by* $v_i$, $1 \leq i \leq 4$, and $(s,t)$ must be a solution to the above equation. Since the equation has $O(1)$ solutions, $e$ contains $O(1)$ second type events for $s$ defined by $v_i$, $1 \leq i \leq 4$. As the above overlay has $O(n)$ segments, there are $O(n)$ second type events defined by $v_i$, $1 \leq i \leq 4$. Enumerating all combinations of four obstacle vertices gives the $O(n^5)$ upper bound for the total number of second type events. ◀