On Improved Dynamic Geodesic Nearest Neighbor Searching in a Simple Polygon

Haomin Li
University of Waterloo
h439li@uwaterloo.ca

Abstract

Nearest Neighbor Searching is one of the most fundamental problems in Computational Geometry. In the following, we summarize the paper of Improved Dynamic Geodesic Nearest Neighbor Searching in a Simple Polygon by Agarwal et al.[Agarwal et al., 2018]. Then, we will discuss about some further works of this paper.

1. Introduction

Nearest Searching can be defined as the problem that we preprocess a point set so that the point closest to a query point can be found efficiently.

Guibus and Hershberger [Guibas and Hershberger, 1987] proposed a data structure which supports efficient two-point shortest path queries in a simple polygon. Chan [Chan, 2010] presented an algorithm which takes in implicit shallow cuttings and maintains lower envelopes for 3d convex hulls and 2d nearest neighbor queries. Kaplan et al. [Kaplan et al., 2017] extended the result of Chan [Chan, 2010] to more general, constant complexity, distance functions. Chan also [Chan, 1998] also presented a randomized query algorithm for halfspace range reporting in R^3.

Mainly based on the above works, Agarwal et al. [Agarwal et al., 2018] demonstrated a fully dynamic data structure to support nearest neighbor queries for a set of sites S inside a simple polygon P.

The main data structure is a balanced binary tree, related to a balanced decomposition of P into subpolygons.Each node stores two copies of the data structure from Kaplan et al.[Kaplan et al., 2017] (construct a k-shallow cutting and realize the lower envelope) In this data structure, we can find the site in S closest to a query point $q \in P$ in $O(log^2nlog^2m)$ time where m is the number of sites in S and n is the number of vertices in P. This data structure also supports sites insertion and sites deletion in amortized $O(log^5nlogm+$

$log^4 nlog^3 m)$ time. The space usage is $O(nlog^3 nlogm + m)$.

2. Preliminary

Definition 2.1 (Geodesic distance)**.** The geodesic distance $\pi(u, v)$ between two vertices u, v is the length in terms of the length of the total edges of the shortest path between the vertices.

Definition 2.2 (Decomposable search problem)**.** Decomposable search problem is a searching problem for which one can decompose the searched domain into into groups, computing the solution for each group individually and take the solution that is the best of all.

2.1. Shortest Path

Lemma 2.1. Any simple polygon can be triangulated.

Proof. By the polygon cutting theorem of Chazelle [Chazelle, 1982], any simple polygon of at least four vertices has a diagonal that divides it into two subpolygons of roughly equal size. We apply this theorem recursively to split each subpolygon. Then, we can get a balanced, hierarchical decomposition of a simple polygon into triangles. These triangles form a triangulation. □

Definition 2.3 (Factor graph)**.** Denote S as the binary tree whose nodes correspond to the splitting diagonals of the triangulated polygon. Since shortest paths that pass through a cell enter and leave by a particular pair of, factor graph S^* is defined based on S to refer those pairs.

Lemma 2.2. The size of the factor graph S^* is $O(n)$, where n is number of the vertices of the corresponding polygon.

Proof. Pick a node p in S. Let the height of p be $h(p)$ which is also the length of the longest path to a leaf below p. Let l be the deepest leaf of p. $\Longrightarrow h(p) = depth(p) - depth(l)$. Since S is balanced, height of every node should be similar and many diagonals cannot have very large height. Suppose $h(p) = k$, we can prove by induction that the subpolygon split by diagonal p must contain at least $\lfloor (\frac{3}{2})^{k-1} + 1 \rfloor$ triangles. since cells belonging to the diagonals with the same height in S are all disjoint, we can get $O(\frac{3}{2}^k n)$ diagonals which have the same height k. Since S^* has $n - 3$ nodes and at most $2h(p)$ edges to descendants from node p, number of edges in S^* is bounded by $\sum_{p \in S^*} 2h(p) = \sum_{k \leq 1 + log_{\frac{3}{2}} n} k \frac{2}{3}^k n = O(n)$ □

2.2. Cuttings

Cuttings were introduced by Matoušek [Matoušek, 1992] as a tool for range searching, specifically, halfspace range reporting. Chazelle [Mehta and Sahni, 2004] stated in Handbook of Data Structures and Applications that cutting provides a space partitioning technique that extends the divide-and-conquer idea to higher dimension.

Definition 2.4 (Cuttings)**.** Let H be a set of n hyperplanes in R^d . Given a parameter $r \in [1, n]$ and a region $L \subseteq R^d$, a $\frac{1}{r}$-cutting for H covering L is a set of interior-disjoint cells such that

- The interior of every cell intersects at most $\frac{n}{r}$ hyperplanes of H
- The union of the cells covers L

Definition 2.5 (Level)**.** We define the level of a point $p \in R^d$ in H to be the number of hyperplanes of H that are below p. Denote $L_{\leq k}(H)$ as the region of all points with level in $H \leq k$.

Definition 2.6 (Conflict list)**.** The conflict list of a cell Δ refers to the subset of all planes of H intersecting Δ and is denoted by H_{Δ}.

Lemma 2.3. For any set H of n planes and parameter r, there exists an $O(\frac{1}{r})$- cutting of the $(\leq \frac{n}{r})$-level of H into $O(r)$ cells. We can construct this cutting, along with the conflict lists of all cells, in $O(nlogn)$ expected time.

Proof. We pick a random sample $R \subseteq H$ of size about r, form a canonical triangulation of the arrangement of R, and return the cells that intersect the $(\leq \frac{n}{r})$-level. The conflict list of each cell would have size close to $O(\frac{n}{r})$. By the algorithm of Ramos [Ramos, 1999], the expected running time of the construction is $O(nlogn)$. □

Definition 2.7 (Shallow cuttings)**.** Given a parameters $k, r \in [1, n]$, a k-shallow $\frac{1}{r}$-cutting $(\Lambda_k(H))$ is a $\frac{1}{r}$-cutting for H covering $L_{\leq k}(H)$.

2.3. Lower Envelope

Definition 2.8. Lower Envelope is the graph of the pointwise minimum of the (partially defined) functions

Fact 2.4. Voronoi diagram is the vertical projection on the R^2 plane of the lower envelope of the distance functions to the sites determining the diagram.

3. Core idea

To solve the nearest neighbor searching problem of the sites S in a simple polygon P, we may first think about using a Voronoi diagram. However, it is very hard to maintain a Voronoi diagram as we insert or delete sites. Since Bentley and Saxe [Bentley and Saxe, 1980] proved that the nearest neighbor searching problem is a decomposable search problem, we can use the idea of decomposable search problem in definition 2.2 which is very similar to divide-and-conquer method. Instead of using an actual Voronoi diagram, by Fact 2.4, we can represent the Voronoi diagram implicitly by computing the lower envelope of a given set of bivariate geodesic distance functions $F = \{f_s | s \in S\}$. ($f_s(x)$ returns the geodesic distance from x to s). The above lower envelope gives the function defined as the pointwise minimum of the given functions. In this way, we only need to calculate the geodesic distance between two sites in the polygon and calculate the lower envelope. Calculating the geodesic distance between each two points will be discussed in the following preprocessing section 4.1.

In order to calculate the lower envelope, Ramos [Ramos, 2000] showed that we can construct shallow cuttings $\Lambda_r(F)$ (definition 2.7) of the function set F . Let $\mathcal{A}(F)$ denote the arrangement of the functions in R^3. Kaplan[Kaplan et al., 2017] stated that the approximate k-level of $\mathcal{A}(F)$ is equivalent to the k-shallow cutting $\Lambda_k(F)$. In order to compute the k-shallow cutting, we can then compute the k-level of $\mathcal{A}(F)$ defined in definition 2.5.

Once we have the lower envelope of the set of geodesic distance functions F ($L_0(F)$), we query the nearest neighbor s of q by find the function f_s that realize $L_0(F)$ at q.

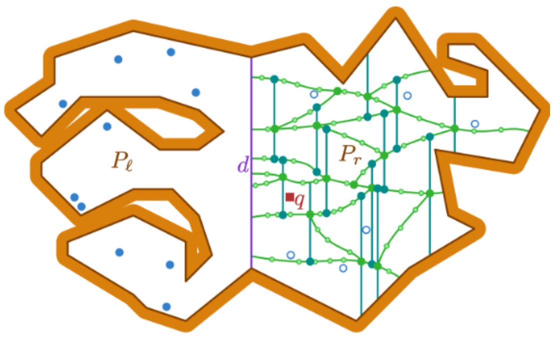

Figure 1. Implicit k-shallow cutting

Figure 1 displays the downward projection of an implicit k-shallow cutting $\Lambda_k(F_r)$ in P_r. The faces are pseudo-trapezoids. This is because of the decomposition of $L_0(F_r)$ into trapezoids. According to Ramos [Ramos, 2000], for Voronoi diagram, tracing a geodesic from each vertex in a face to the site determining the cell can result in such decomposition.

4. Overview of the approach

The approach involves several parts: Preprocessing, Recursively partition the polygon P into two roughly equal size sub-polygons P_r and P_l, Process on each sub-polygon, Query point q and find its nearest neighbor, Add or remove a site.

4.1. Preprocessing

Guibus and Hershberger [Guibas and Hershberger, 1987] provided a method to preprocess the polygon P so that we can find the length of the shortest path inside the polygon from a point $p \in P$ to a point $q \in P$ in $O(logn)$ time where n is the number of vertices of the polygon P.

In the paper of Guibus and hershberger, their method builds a hierarchy of nested subpolygons over an underlying triangulation, such that any shortest path crosses only a small number of subpolygons. Each subpolygons contains information about shortest paths. When we query the shortest distance between two points in the polygon, the method assembles path information from the subpolygons between them.

By lemma 2.1, lemma 2.2 and counting the number of principal separating diagonals (which is a subsequence of the separating diagonals that will be useful in finding the shortest path), they proved that they can extract a sequence of $O(logn)$ cells determining the shortest path between two points in $O(logn)$ time. Then, they introduced a geometric structure called hourglass which is associated with each edge of the factor graph S^* (definition 2.3). With this structure, we can get the represenation of shortest geodesic distance between any two points in $O(logn)$.

4.2. Process on each sub-polygon

In Chan's data structure [Chan, 2010], dynamic lower envelopes of planes in R^3 lies in the construction of small-sized shallow cuttings for planes. If we have small-sized shallow cuttings for surfaces, we are able to maintain dynamically the lower envelope of surfaces. By the lemma 2.3, we can compute the small-szized shallow cuttings in $O(nlogn)$. Then we can plug into the Chan's data structure and maintain the lower envelope. Here are the genener steps if the algorithm:

1. For the sites S_r in P_r, we consider their geodesic distance functions $F_r = \{f_s | s \in S_r\}$ restricted to P_l. f_s is a bivariate function. (i.e. $f_s(x) = \pi(s, x)$ where $x \in P_l$)
2. By lemma 2.3, we can represent an implicit vertical shallow cutting $\Lambda_r(F_r)$ for these functions F_r.
3. Then, we can build and maintain the lower envelope $L_0(F_r)$ by plugging the shallow cutting into Chan's data structure [Chan, 2010].
4. Similarly, we can build and maintain the lower envelope $L_0(F_l)$ for the sites in P_l

4.3. Query point q

To be more specific, we use the Chan's query algorithm [Chan, 1998] to query. Note that this algorithm may fail for some input parameter σ, we might need to adjust the σ until the algorithm succeeds. Instead of returning the k lowest planes intersecting the vertical line l, the algorithm returns the k lowest functions (k is also an input parameter in Chan's algorithm) intersecting l in out settings when it succeeds. Then, by repeatedly querying with the vertical line through the point q and doubling k , we can retrieve all functions that pass below the point q. So we can get the level of q by calculating the number of functions that pass below q and we can get the corresponding lower envelope. Below is the general steps:

1. Assume we get a query point $q \in P_r$ we use the lower envelope $L_0(F_r)$ to find the site in P_l closest to q in $O(log^2 nlogm)$
2. To find the site in P_r closest to q, we recursively query in sub-polygon P_r . In total we query $O(logm)$ levels, leading to an $O(log^2 nlog^2 m)$ query time.
3. After comparing the sites we derived from the above, we get the nearest neighbor for q.

4.4. Add or remove a site

When we add or remove a site, we insert or remove its distance function in the lower envelope data structures. The authors did not explicitly describe the insertion and deletion behaviors. But the scenario is quite similar to the scenario in Chen's fully dynamic data structure [Chan, 2010]. Instead of inserting a plane to the set of planes, we insert or remove the distance functions to the F. Then, we reconstruct the shallow cutting in the sub-polygon the site affects and rebuild the lower envelopes.

5. Further Work

5.1. K nearest neighbor

As we know, Agarwal et al. [Agarwal et al., 2018] generalized shallow cuttings to geodesic distance functions, but their shallow cuttings could not be applied to the k nearest neighbors problem. Based on Agarwal's idea, Liu [Liu, 2020] constructed linear-size shallow cuttings for the static data structure of k nearest neighbors problem and its related problem: circular range query problem. He discussed a new tool called *global and local conflicts of a configuration* to transfer from the shallow cuttings of Agarwal et al. [Agarwal et al., 2018] to the linear-size shallow cuttings.

5.2. Kinetic Geodesic Voronoi Diagrams in a Simple Polygon

Korman et al. [Korman et al., 2020] developed the first kinetic data structure (KDS) that maintains the geodesic Voronoi diagram as multiple sites move in a simple polygon. They carefully analyze when and how often the corresponding Voronoi diagram can change.

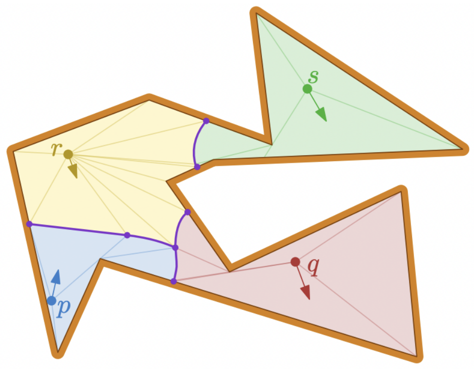

Figure 2 displays the changes of geometric Voronoi diagram while moving the sites r, p, s, q. Korman et al. [Korman et al., 2020] also proved tight bounds on the number of combinatorial changes in a single bisector, and on the trajectory of a Voronoi center.

Figure 2. Kinetic effects on Voronoi diagram

5.3. Enhancement

As the data structure of Agarwal et al. [Agarwal et al., 2018] supports sites insertion and sites deletion methods, we may want to access or update one of the previous versions of the data structure. For example, we may want to query the nearest neighbor of a point q before its current nearest neighbor site was inserted. We can construct a fully persistent structure for the problem of geodesic nearest neighbor searching in a simple polygon. Since the data structure of Agarwal et al. [Agarwal et al., 2018] is a balanced-binary-tree like structure, similar to the way that Driscoll et al. [Driscoll et al., 1989] converted red-black tree to fully persistent data structure, we construct a fully persistent version of the improved dynamic geodesic nearest neighbor searching data structure by node-splitting method. This will supports amortized time and space O(1) per update and a worst-case time of O(1) per access.

Furthermore, with the idea of the Kinetic Geodesic Voronoi Diagrams in a Simple Polygon, we can also modify the persistent data structure to make it support sites linear movement operation.

References

[Agarwal et al., 2018] Agarwal, P. K., Arge, L., and Staals, F. (2018). Improved Dynamic Geodesic Nearest Neighbor Searching in a Simple Polygon. In Speckmann, B. and Tóth, C. D., editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Bentley and Saxe, 1980] Bentley, J. L. and Saxe, J. B. (1980). Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301 – 358.

[Chan, 1998] Chan, T. M. (1998). Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 586–595.

[Chan, 2010] Chan, T. M. (2010). A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3).

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

[Driscoll et al., 1989] Driscoll, J. R., Sarnak, N., Sleator, D. D., and Tarjan, R. E. (1989). Making data structures persistent. *J. Comput. Syst. Sci.*, 38(1):86–124.

[Guibas and Hershberger, 1987] Guibas, L. J. and Hershberger, J. (1987). Optimal shortest path queries in a simple polygon. In *Proceedings of the Third Annual Symposium on Computational Geometry*, SCG ’87, page 50–63, New York, NY, USA. Association for Computing Machinery.

[Kaplan et al., 2017] Kaplan, H., Mulzer, W., Roditty, L., Seiferth, P., and Sharir, M. (2017). Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’17, page 2495–2504, USA. Society for Industrial and Applied Mathematics.

[Korman et al., 2020] Korman, M., Renssen, A., Roeloffzen, M., and Staals, F. (2020). Kinetic geodesic voronoi diagrams in a simple polygon.

[Liu, 2020] Liu, C.-H. (2020). *Nearly Optimal Planar k Nearest Neighbors Queries under General Distance Functions*, pages 2842–2859.

[Matoušek, 1992] Matoušek, J. (1992). Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3):169–186.

[Mehta and Sahni, 2004] Mehta, D. P. and Sahni, S. (2004). *Handbook Of Data Structures And Applications (Chapman Hall/Crc Computer and Information Science Series.)*. Chapman Hall/CRC.

[Ramos, 1999] Ramos, E. A. (1999). On range reporting, ray shooting and k-level construction. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry*, SCG '99, page 390–399, New York, NY, USA. Association for Computing Machinery.

[Ramos, 2000] Ramos, E. A. (2000). Deterministic algorithms for 3-d diameter and some 2-d lower envelopes. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, SCG '00, page 290–299, New York, NY, USA. Association for Computing Machinery.