Approximate Line Segment Nearest Neighbor Search *amid* Polyhedra *in* 3-Space

Ovidiu Daescu *and* **Ka Yaw Teo**

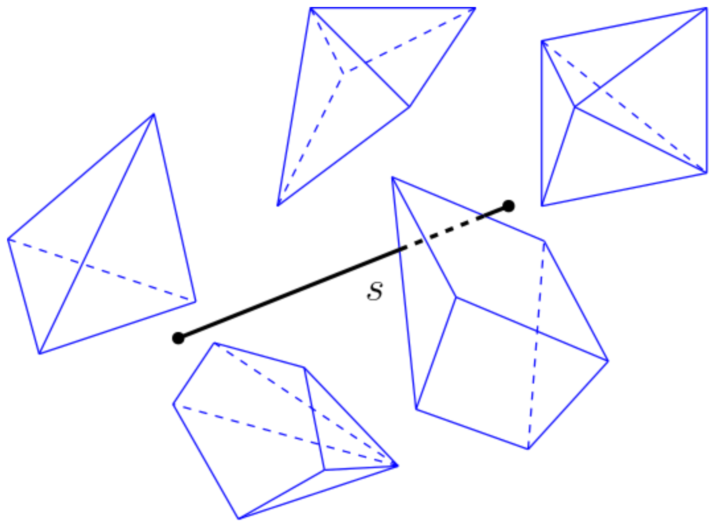
Department of Computer Science

University of Texas at Dallas, Richardson, TX

Problem statement

Input:

A set Π of **polyhedra** with total complexity n in $\mathbb{R}^3$.



Objective:

Preprocess Π so that given
any **query line segment *s***, one may quickly
find the **polyhedron in Π nearest to *s***.

In this study, we consider the **approximate** version:

- For any real parameter $\varepsilon > 0$, return an input polyhedron whose distance to s is **at most (1 + ε) times** the distance from the nearest input polyhedron to s.

Motivation:

Nearest neighbor search has applications in areas such as computational geometry, machine learning, data science, etc.

A particular relevance in **path planning** problems involving **collision or clearance queries** for **non-point objects** in three-dimensional space.

Related work: Nearest neighbor search (NNS) problems

How **difficult** is the problem? To get a rough idea…

NNS with $\boldsymbol{O(\log n)}$ query time has long been closely connected to **Voronoi diagram (VD).**

- In $\mathbb{R}^{d}$, the worst-case complexity of VD for **point sites** is $\boldsymbol{\Theta(n^{\lceil d/2 \rceil})}$.

  Note: Nearest neighbor search for non-point objects is at least as hard as for point objects.

- In $\mathbb{R}^{3}$, the worst-case complexity of VD for **general sites** is **not well understood**.
  - For a few non-point sites and metrics in $\mathbb{R}^3$, **known lower bounds** are **roughly quadratic** (e.g., see [Sharir, 1995], [Boissonnat et al., 1998], [Har-Peled, 2001], [Koltun and Sharir, 2004]).
- For any "reasonable" class of geometric sites, the maximum complexity of VD in $\mathbb{R}^d$ is **conjectured** to be close to $\boldsymbol{\Theta(n^{d-1})}$ [Boris, 2001].

Both **VD- and non-VD-style data structures** have been proposed for NNS problems.

Input and query objects are **points**:

- Efficient **exact** algorithms for **low dimensions** (e.g., see [Clarkson, 1988], [Meiser, 1993]).
- Efficient **approximate** algorithms for **high dimensions** (e.g., see [Andoni et al., 2014], [Andoni and Indyk, 2017], [Arya et al., 2017]).

Related work: Nearest neighbor search (NNS) problems

Input and/or query objects are **more complex than points**:

Input objects	**Query object**	**References**
Polyhedra in $\mathbb{R}^3$	Point	Koltun and Sharir, 2004
Lines		Chew et al., 1998; Har-Peled, 2001; Mahabadi, 2014
k-flats		Magen, 2007; Basri et al., 2010; Agarwal et al., 2017
Line segments		Abdelkader and Mount, 2021
Points	Line	Andoni et al., 2009
	k-flat	Mulzer et al., 2015
	Line segment in $\mathbb{R}^2$	Bespamyatnikh, 2003; Goswami et al., 2004; Segal and Zeitlin, 2008
Polygons in $\mathbb{R}^2$	Line segment	Daescu and Malik, 2018

Related work: Nearest neighbor search (NNS) problems

Input and/or query objects are **more complex than points**:

Input objects	**Query object**	**References**
Polyhedra in $\mathbb{R}^3$	Point	Koltun and Sharir, 2004
Lines		Chew et al., 1998; Har-Peled, 2001; Mahabadi, 2014
k-flats		Magen, 2007; Basri et al., 2010; Agarwal et al., 2017
Line segments		Abdelkader and Mount, 2021
Points	Line	Andoni et al., 2009
	k-flat	Mulzer et al., 2015
	Line segment in $\mathbb{R}^2$	Bespamyatnikh, 2003; Goswami et al., 2004; Segal and Zeitlin, 2008
Polygons in $\mathbb{R}^2$	Line segment	Daescu and Malik, 2018

- Most recent (related) work [Abdelkader and Mount, 2021]:
  - **Approximate nearest neighboring line segment** to a **query point** in $\mathbb{R}^{d}$.
  - $O((n^2/\varepsilon^d)\log(\Delta/\varepsilon))$ preprocessing time/space, $O(\log(\max\{n, \Delta\}/\varepsilon))$ query time, where Δ is the spread of the input line segments.

Related work: Nearest neighbor search (NNS) problems

Input and/or query objects are **more complex than points**:

Input objects	**Query object**	**References**
Polyhedra in $\mathbb{R}^3$	Point	Koltun and Sharir, 2004
Lines		Chew et al., 1998; Har-Peled, 2001; Mahabadi, 2014
k-flats		Magen, 2007; Basri et al., 2010; Agarwal et al., 2017
Line segments		Abdelkader and Mount, 2021
Points	Line	Andoni et al., 2009
	k-flat	Mulzer et al., 2015
	Line segment in $\mathbb{R}^2$	Bespamyatnikh, 2003; Goswami et al., 2004; Segal and Zeitlin, 2008
Polygons in $\mathbb{R}^2$	Line segment	Daescu and Malik, 2018

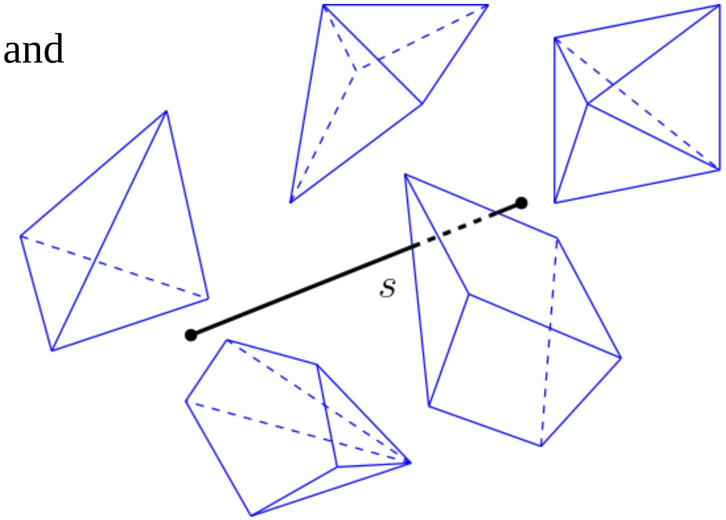
- **Two-dimensional variant** of our problem [Daescu and Malik, 2018]:
  - **Exact nearest neighboring polygon** to a **query line segment** in the **plane**.
  - $\boldsymbol{O(m \log n)}$ preprocessing time, $\boldsymbol{O(m)}$ space, $\boldsymbol{O((n/m^{1/2}) \text{ polylog } n)}$ query time, for any $n \leq m \leq n^2$.

Overview of our results

A non-trivial algorithm for **$(1 + \varepsilon)$-approximate line segment nearest neighbor** search among **polyhedra in** $\mathbb{R}^3$.

For any $n \le m \le n^3$, we obtain

$O((m/\varepsilon)\ \mathrm{polylog}\ n + n^{2+\varepsilon})$ preprocessing time/space and
$O((1/\varepsilon)(n/m^{1/3})\ \mathrm{polylog}\ n)$ query time.



Specifically, if we set $m = n$, we have

$O(n^{2+\varepsilon})$ preprocessing time/space and
$O((n^{2/3}/\varepsilon)\ \mathrm{polylog}\ n)$ query time.

Machineries and tools used:

Polyhedral metric (for approximating the Euclidean distance),
approximate Voronoi diagram,
real algebraic geometry,
multi-level partition trees.

Approximate line segment nearest neighbor

Suppose that query line segment s **intersects** a polyhedron in Π.

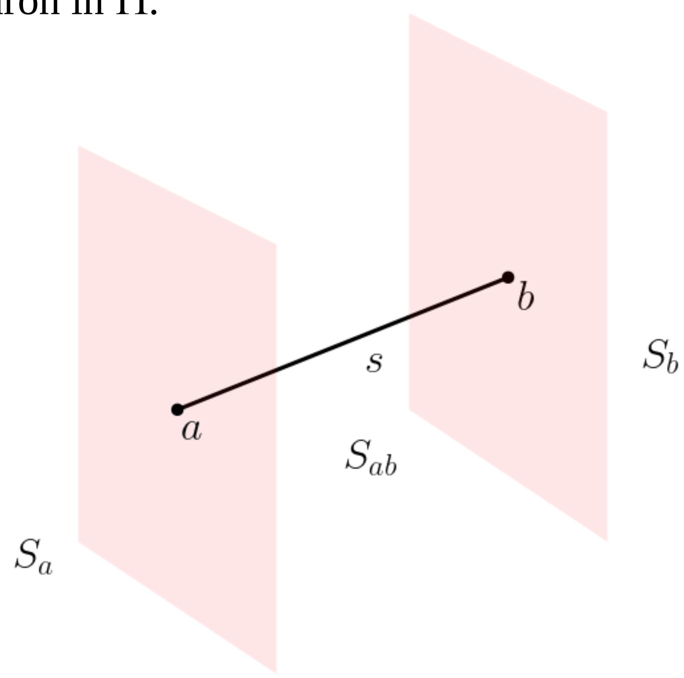
- Can be identified in $\boldsymbol{O(n^{1/2+\alpha})}$ time after a preprocessing that takes $\boldsymbol{O(n^{3/2+\alpha})}$ expected time and space [Ezra and Sharir, 2022].

Hereafter, assume that s **does not intersect** any polyhedron in Π.

1) Find the polyhedron in Π **closest to each endpoint of *s*** (a and b).

2) Find the polyhedron in $\Pi \cap S_{ab}$ closest to s (i.e., **closest orthogonal neighbor to *s***).

Of the polyhedra found above,
the one with the shortest distance to s is
the polyhedron in Π nearest to s.



Nearest neighbor to each endpoint of *s*

Subproblem 1.

Given a set Π of polyhedra with total complexity n in $\mathbb{R}^3$, preprocess Π so that for any query point p, one can quickly determine the polyhedron in Π closest to p.

Exact solution:

- Let T be the set of $\boldsymbol{O(n)}$ triangular faces of the polyhedral in Π.
- Reduce Subproblem 1 to finding the triangle of T nearest to p.
- For each triangle $\tau \in T$, define $f_\tau(p)$ to be the Euclidean distance from any point p to τ.
- Let $C_T = \{f_\tau(p) \mid \tau \in T\}$.
- Let M_T be the **lower envelope** of C_T.
- For any query point $p = (x, y, z)$, the triangle $\tau \in T$ nearest to p is given by $f_\tau(p)$ attaining M_T at (x, y, z).
- For any $\alpha > 0$, lower envelope M_T can be constructed in $\boldsymbol{O(n^{3+\alpha})}$ expected time and stored in a data structure of $\boldsymbol{O(n^{3+\alpha})}$ size such that a nearest triangle search query can be answered in $\boldsymbol{O(\log^2 n)}$ time [Agarwal et al., 1997].

Nearest neighbor to each endpoint of s

Subproblem 1.

Given a set Π of polyhedra with total complexity n in $\mathbb{R}^3$, preprocess Π so that for any query point p, one can quickly determine the polyhedron in Π closest to p.

Approximate solution:

- Define the **polyhedral distance** between any two points p and q:

  $$d_Q(p, q) = \sup\{t \mid q \notin p + tQ\},$$

  where Q is a symmetric convex polytope.

- For any $\varepsilon > 0$, there is a Q represented by the intersection of $\boldsymbol{O(1/\varepsilon)}$ half-spaces such that

  $$d(p, q) \le d_Q(p, q) \le (1 + \varepsilon)\, d(p, q),$$

  where $d(p, q)$ is the Euclidean distance between p and q [Dudley, 1974].

- Construct the **Voronoi diagram** V of Π under d_Q in $\boldsymbol{O(n^{2+\varepsilon})}$ time, and the complexity of V is $\boldsymbol{O(n^{2+\varepsilon})}$ [Kolton and Sharir, 2004].

- Using V, for any query point p, we can report, in $\boldsymbol{O(\log n)}$ time, a $(1 + \varepsilon)$-approximate nearest neighboring polyhedron in Π to p.

Nearest orthogonal neighbor to s

Subproblem 2.

Given a set Π of polyhedra with total complexity n in $\mathbb{R}^3$, preprocess Π so that for any query line segment s, one can quickly determine the polyhedron in $\Pi \cap S_{ab}$ closest to s.

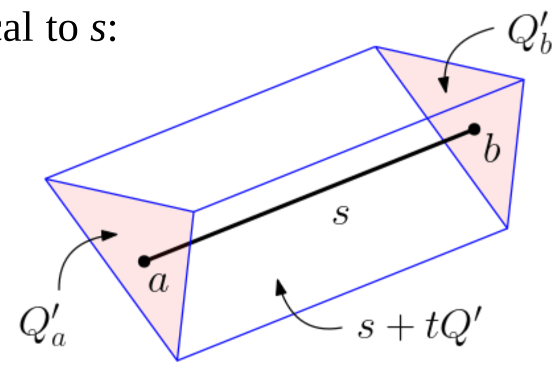
Approximate using a **polyhedral metric**:

- Define a convex polygonal prism T that is axially symmetrical to s: $s + tQ'$, where Q' is a symmetric convex $\boldsymbol{O(1/\varepsilon)}$-gon.

  T has two $\boldsymbol{O(1/\varepsilon)}$-gons Q'_a and Q'_b as base faces, connected by $\boldsymbol{O(1/\varepsilon)}$ rectangular sides.

- Using T, define the polyhedral distance between s and a polyhedron P in $\Pi \cap S_{ab}$:

$$d_{Q'}(p, q) = \sup\{t \mid P \cap (s + tQ') = \varnothing\}.$$



Process each of $\boldsymbol{O(1/\varepsilon)}$ faces and edges of T for **face- and edge-shooting queries**.

Two scenarios to be considered: A) A **shooting face** hits a **vertex** of P.

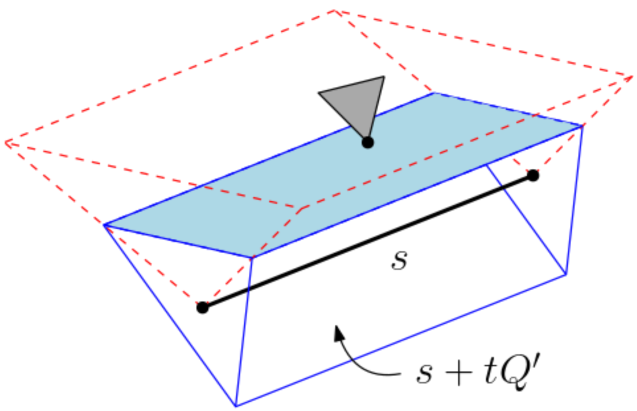
B) A **shooting edge** hits an **edge** of P.

Scenario A

A query shoots a **fixed-direction rectangular face** from *s*.

The expanding face traces a **3D infinite wedge**.

Look for the first time the expanding wedge hits a vertex of an input polyhedron.



Let V be the set of $\boldsymbol{O(n)}$ vertices of the polyhedra in Π.

Construct a **5-level partition tree** on V:

- First 4 levels are used to collect the vertices in V that lie within the infinite wedge as the union of a small number of canonical subsets.

  Each of first 4 levels supports **half-space range searching queries** among the points of V.

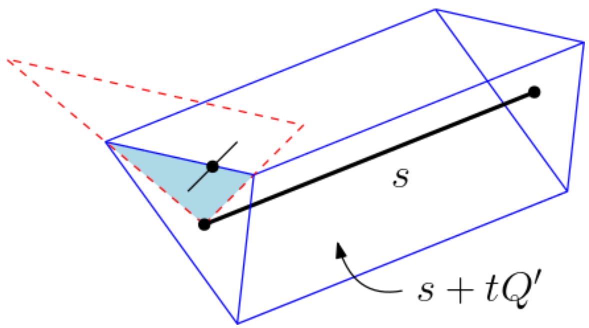
- 5-th level supports queries that ask for the vertex in the canonical subsets that is minimal in a fixed direction.

- Following standard methodology for constructing multi-level partition trees [Agarwal, 2017; Chan, 2012; Dobkin and Edelsbrunner, 1987; Matoušek, 1993]:

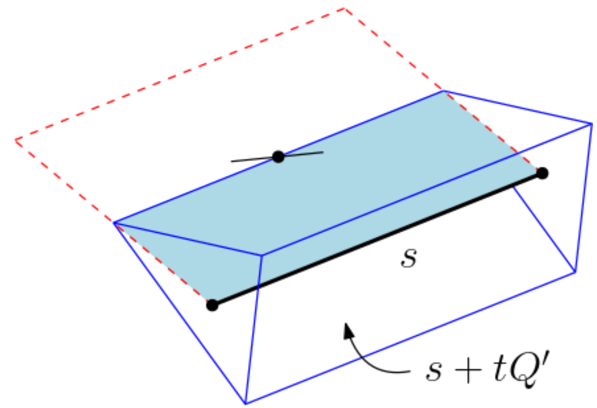
  Preprocessing time/space is $\boldsymbol{O(m \text{ polylog } n)}$, and query time is $\boldsymbol{O(n/m^{1/3} \text{ polylog } n)}$, for any $n \le m \le n^3$.

Scenario B

Two types of shooting edges:



(I) Normal to *s*



(II) Parallel to *s*

Type I

A query shoots a **fixed line segment normal to *s*** from an endpoint of *s*.

The expanding line segment traces a **2D infinite wedge**.

Seek for the first time the expanding wedge hits an edge of an input polyhedron.

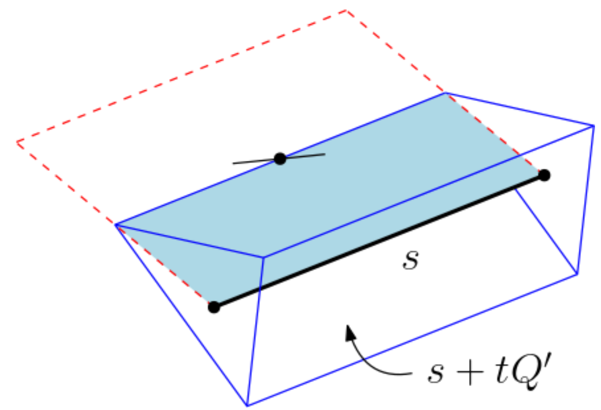
This scenario has been taken care of when solving Subproblem 1.

Scenario B – Type II

A query shoots a **fixed line segment identical and parallel to *s*** from *s*.

The moving line segment traces an **infinite rectangle**.

Look for the first time the expanding rectangle hits an edge of an input polyhedron.



Let E be the set of $\boldsymbol{O(n)}$ edges of the polyhedra in Π.

Let e denote the fixed shooting edge parallel to s.

Assume that e and s are in some plane $z = z_0$.

Let ℓ be the length of s.

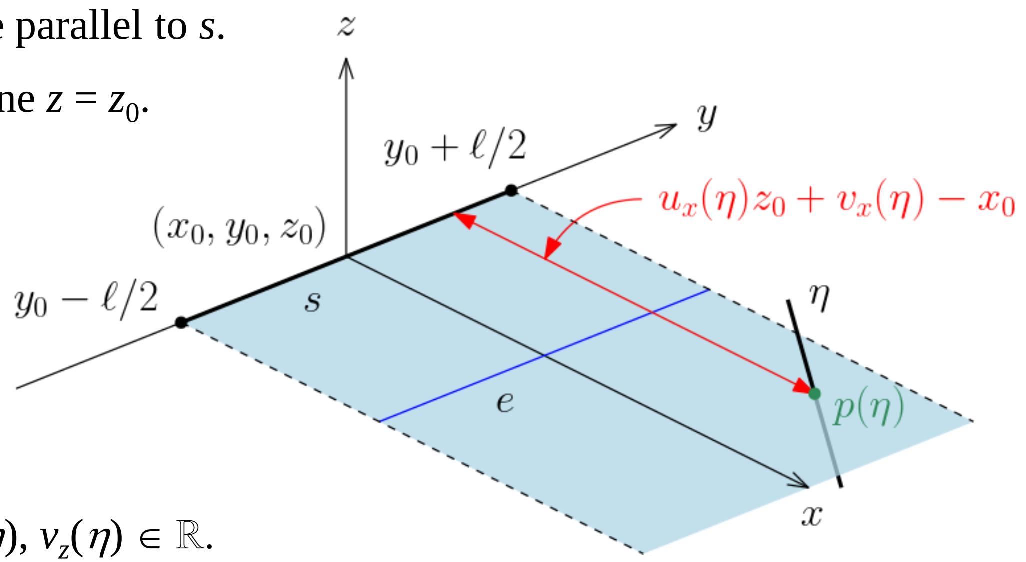
Parameterize each edge η in E:

$$x = u_x(\eta)z + v_x(\eta)$$

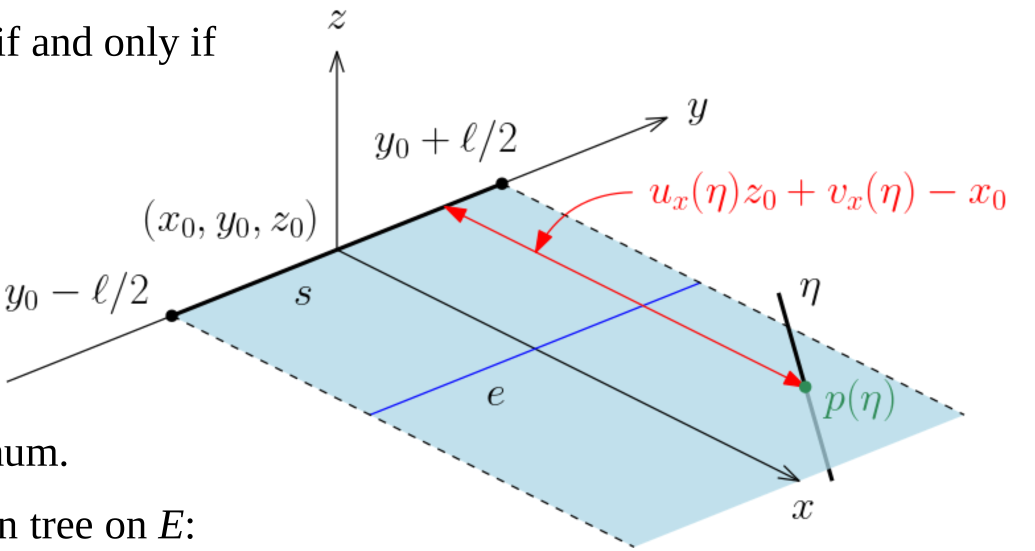
$$y = u_y(\eta)z + v_y(\eta)$$

$$z = u_z(\eta)z + v_z(\eta)$$

where $u_x(\eta), v_x(\eta), u_y(\eta), v_y(\eta), u_z(\eta), v_z(\eta) \in \mathbb{R}$.



Scenario B – Type II



$p(\eta)$ lies within the infinite rectangle if and only if

$$\left.\begin{aligned} &u_y(\eta)z_0 + v_y(\eta) \geq y_0 - l/2, \\ &u_y(\eta)z_0 + v_y(\eta) \leq y_0 + l/2, \\ &z_0 - u_z(\eta) \geq 0, \text{ and} \\ &z_0 - v_z(\eta) \leq 0. \end{aligned}\right\} \quad (1)$$

Find the η in E that satisfies (1)
such that $u_x(\eta)z_0 + v_x(\eta) - x_0$ is minimum.

To do that, construct a 5-level partition tree on E:

- First 4 levels are to collect the edges in E that satisfies (1).

  Define 4 planar point sets $P_1 = \{(u_y(\eta), v_y(\eta)) \mid \eta \in E\}$, $P_2 = \{(1, -u_z(\eta)) \mid \eta \in E\}$, $P_3 = \{(1, -v_z(\eta)) \mid \eta \in E\}$, and $P_4 = \{(u_x(\eta), v_x(\eta)) \mid \eta \in E\}$.

  1-st and 2-nd levels support **half-plane range searching queries** against P_1, 3-rd level against P_2, and 4-th level against P_3.

- 5-th level supports queries that ask for the edge that is minimal in a fixed direction.

- Preprocessing time/space is $\boldsymbol{O(m}$ **polylog** $\boldsymbol{n)}$, and
  query time is $\boldsymbol{O(n/m^{1/2}}$ **polylog** $\boldsymbol{n)}$, for any $n \leq m \leq n^2$.

Nearest orthogonal neighbor to *s*

Subproblem 2.

Given a set Π of polyhedra with total complexity n in $\mathbb{R}^3$, preprocess Π so that for any query line segment s, one can quickly determine the polyhedron in $\Pi \cap S_{ab}$ closest to s.

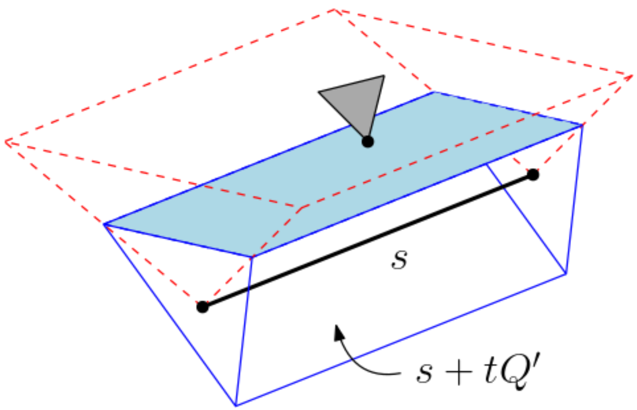
Prepare query data structures:

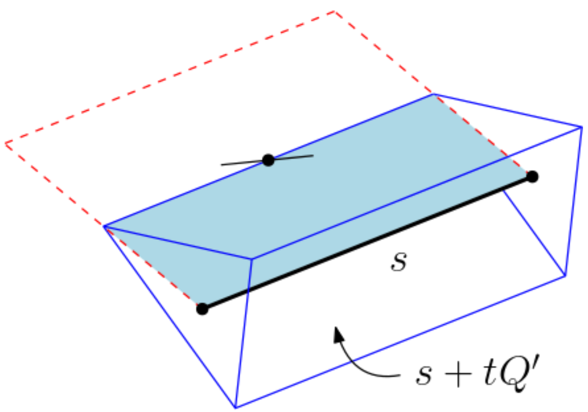
- For each shooting face of T in Scenario A:

  $\boldsymbol{O(m}$ **polylog** $\boldsymbol{n)}$ preprocessing time/space,
  $\boldsymbol{O(n/m^{1/3}}$ **polylog** $\boldsymbol{n)}$ query time, for any $n \le m \le n^3$.

- For each shooting edge of T in Scenario B – Type II:

  $\boldsymbol{O(m}$ **polylog** $\boldsymbol{n)}$ preprocessing time/space,
  $\boldsymbol{O(n/m^{1/2}}$ **polylog** $\boldsymbol{n)}$ query time, for any $n \le m \le n^2$.





Since there are $\boldsymbol{O(1/\varepsilon)}$ such faces and edges, this takes $\boldsymbol{O((m/\varepsilon)}$ **polylog** $\boldsymbol{n)}$ preprocessing time and storage total, for any $n \le m \le n^3$.

Total cost of a query is $\boldsymbol{O((1/\varepsilon)(n/m^{1/3})}$ **polylog** $\boldsymbol{n)}$.

Overview of our results

Subproblem 1: Finding nearest neighbor to each endpoint of query line segment s

$\boldsymbol{O(n^{2+\varepsilon})}$ preprocessing time/space, $\boldsymbol{O(\log n)}$ query time.

Subproblem 2: Finding nearest orthogonal neighbor to query line segment s

$\boldsymbol{O((m/\varepsilon) \text{ polylog } n)}$ preprocessing time/space, $\boldsymbol{O((1/\varepsilon)(n/m^{1/3}) \text{ polylog } n)}$ query time, for any $n \le m \le n^3$.

In conclusion, for any $n \le m \le n^3$, we can find a **$(1 + \varepsilon)$-approximate line segment nearest neighbor** among **polyhedra in** $\mathbb{R}^{\mathbf{3}}$ with

$\boldsymbol{O((m/\varepsilon) \text{ polylog } n + n^{2+\varepsilon})}$ preprocessing time/space and

$\boldsymbol{O((1/\varepsilon)(n/m^{1/3}) \text{ polylog } n)}$ query time.

Specifically,

$m = n$ $\boldsymbol{O(n^{2+\varepsilon})}$ preprocessing time/space and $\boldsymbol{O((n^{2/3}/\varepsilon) \text{ polylog } n)}$ query time.

$m = n^3$ $\boldsymbol{O((n^3/\varepsilon) \text{ polylog } n)}$ preprocessing time/space and $\boldsymbol{O((1/\varepsilon) \text{ polylog } n)}$ query time.

Concluding remarks

Partly motivated by **path planning applications**, where **paths** are suggested in real time and need to be verified quickly if they **satisfy certain constraints** such as having a given clearance from obstacles.

Given a set of polyhedra in 3-space, preprocess them so that for any query polygonal path and any real value $c > 0$, one can quickly

i) **report the clearance of the path**, and/or

ii) **determine if the path has a clearance of at least c**.

Query (i) can be answered by finding the exact nearest neighboring input polyhedron.

Query (ii) can be addressed after performing query (i).

Unfortunately, exact nearest neighbor search in such a setting is expensive, and approximation (using the results herein) may yield inconclusive answer.

Is it feasible to quickly answer query (ii) definitively without an exact solution to query (i)?

Note: After obtaining a decision algorithm for query (ii), query (i) can be answered using parametric search.

References

A. Abdelkader and D. M. Mount. Approximate nearest neighbor search for line segments. In *37th International Symposium on Computational Geometry*, 2021.

P. K. Agarwal. Simplex range searching and its variants: A review. *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 1–30, 2017.

P. K. Agarwal, B. Aronov, and M. Sharir. Computing envelopes in four dimensions with applications. *SIAM Journal on Computing*, 26(6):1714–1732, 1997.

P. K. Agarwal, N. Rubin, and M. Sharir. Approximate nearest neighbor search amid higher-dimensional flats. In *25th Annual European Symposium on Algorithms*, 2017.

A. Andoni and P. Indyk. Nearest neighbors in high dimensional spaces. In *Handbook of Discrete and Computational Geometry*, pages 1135–1155. Chapman and Hall/CRC, 2017.

A. Andoni, P. Indyk, R. Krauthgamer, and H. L. Nguyên. Approximate line nearest neighbor in high dimensions. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 293–301, 2009.

A. Andoni, P. Indyk, H. L. Nguyên, and I. Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete algorithms*, pages 1018–1028, 2014.

B. Aronov. A lower bound on Voronoi diagram complexity. *Information Processing Letters*, 83(4):183–185, 2002.

References

S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal approximate polytope membership. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 270–288, 2017.

R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate nearest subspace search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):266–278, 2010.

S. Bespamyatnikh. Computing closest points for segments. *International Journal of Computational Geometry & Applications*, 13(05):419–438, 2003.

J. D. Boissonnat, M. Sharir, B. Tagansky, and M. Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. *Discrete & Computational Geometry*, 19(4):473–484, 1998.

T. M. Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47:661–690, 2012.

L. P. Chew, K. Kedem, M. Sharir, B. Tagansky, and E. Welzl. Voronoi diagrams of lines in 3-space under polyhedral convex distance functions. *Journal of Algorithms*, 29(2):238–255, 1998.

K. L. Clarkson. A randomized algorithm for closest point queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

References

O. Daescu and H. Malik. Does a robot path have clearance c? In *Proceedings of the 12th International Conference on Combinatorial Optimization and Applications*, pages 509–521, 2018.

O. Daescu and R. Serfling. Extremal point queries with lines and line segments and related problems. *Computational Geometry*, 32(3):223–237, 2005.

D. P. Dobkin and H. Edelsbrunner. Space searching for intersecting objects. *Journal of Algorithms*, 8(3):348–361, 1987.

R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10(3):227–236, 1974.

E. Ezra and M. Sharir. On ray shooting for triangles in 3-space and related problems. *SIAM Journal on Computing*, 51(4):1065–1095, 2022.

P. P. Goswami, S. Das, and S. C. Nandy. Triangular range counting query in 2D and its application in finding k nearest neighbors of a line segment. *Computational Geometry*, 29(3):163–175, 2004.

S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 94–103, 2001.

V. Koltun and M. Sharir. Three dimensional Euclidean Voronoi diagrams of lines with a fixed number of orientations. *SIAM Journal of Computing*, 32:616–642, 2003.

References

V. Koltun and M. Sharir. Polyhedral Voronoi diagrams of polyhedra in three dimensions. *Discrete & Computational Geometry*, 31:83–124, 2004.

A. Magen. Dimensionality reductions in ℓ_2 that preserve volumes and distance to affine spaces. *Discrete & Computational Geometry*, 38(1):139–153, 2007.

S. Mahabadi. Approximate nearest line search in high dimensions. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 337–354, 2014.

J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10:157–182, 1993.

S. Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286–303, 1993.

W. Mulzer, H. L. Nguyên, P. Seiferth, and Y. Stein. Approximate k-flat nearest neighbor search. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, pages 783–792, 2015.

M. Segal and E. Zeitlin. Computing closest and farthest points for a query segment. *Theoretical computer science*, 393(1-3):294–300, 2008.

M. Sharir. Arrangements in higher dimensions: Voronoi diagrams, motion planning and other applications. In *Proceedings of the 4th Workshop on Algorithms and Data Structures*, pages 109–121, 1995.

Thank you!

Ovidiu Daescu *and* **Ka Yaw Teo**

Department of Computer Science

University of Texas at Dallas, Richardson, TX