Sublinear Geometric Algorithms

B. Chazelle, D. Liu, A. Magen
Princeton University / University of Toronto

STOC 2003

CS 468
Geometric Algorithms Seminar
Winter 2005/2006

Overview

What is this paper about?

Sublinear geometric algorithms, for

convex polygons (2D) and *convex polyhedra* (3D):

- Intersection
- Ray shooting
- Volume approximation
- Shortest path approximation

Sublinear Algorithms

What means sublinear?

- *Not* reading the complete input
- Hence: *No preprocessing*
- Assuming "standard" input formats
- *Randomized* Las-Vegas algorithms
  - No wrong answers, but run time may vary
  - *Expected* runtimes are sublinear

Overview

Overview:

- Basic technique
  - Sublinear search algorithm
  - Optimality
- Polygons / polyhedra intersection
- Ray shooting & applications
- Volume approximation
- Shortest path approximation

Overview

Overview:

- Basic technique
  - Sublinear search algorithm
  - Optimality
- Polygons / polyhedra intersection
- Ray shooting & applications
- Volume approximation
- Shortest path approximation

each new technique builds upon previous techniques

Basic Technique

Search Problem

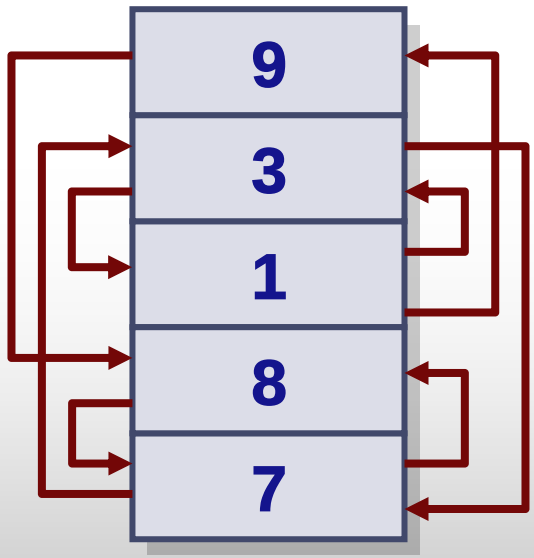
Successor Search:

- Given a *sorted doubly-linked list* of numbers
- And a search value *k*
- Find first element greater than *k*
- List stored in *continuous memory block* (allowing sampling of list elements)

Data Structure

Looks like this:

Memory Layout:



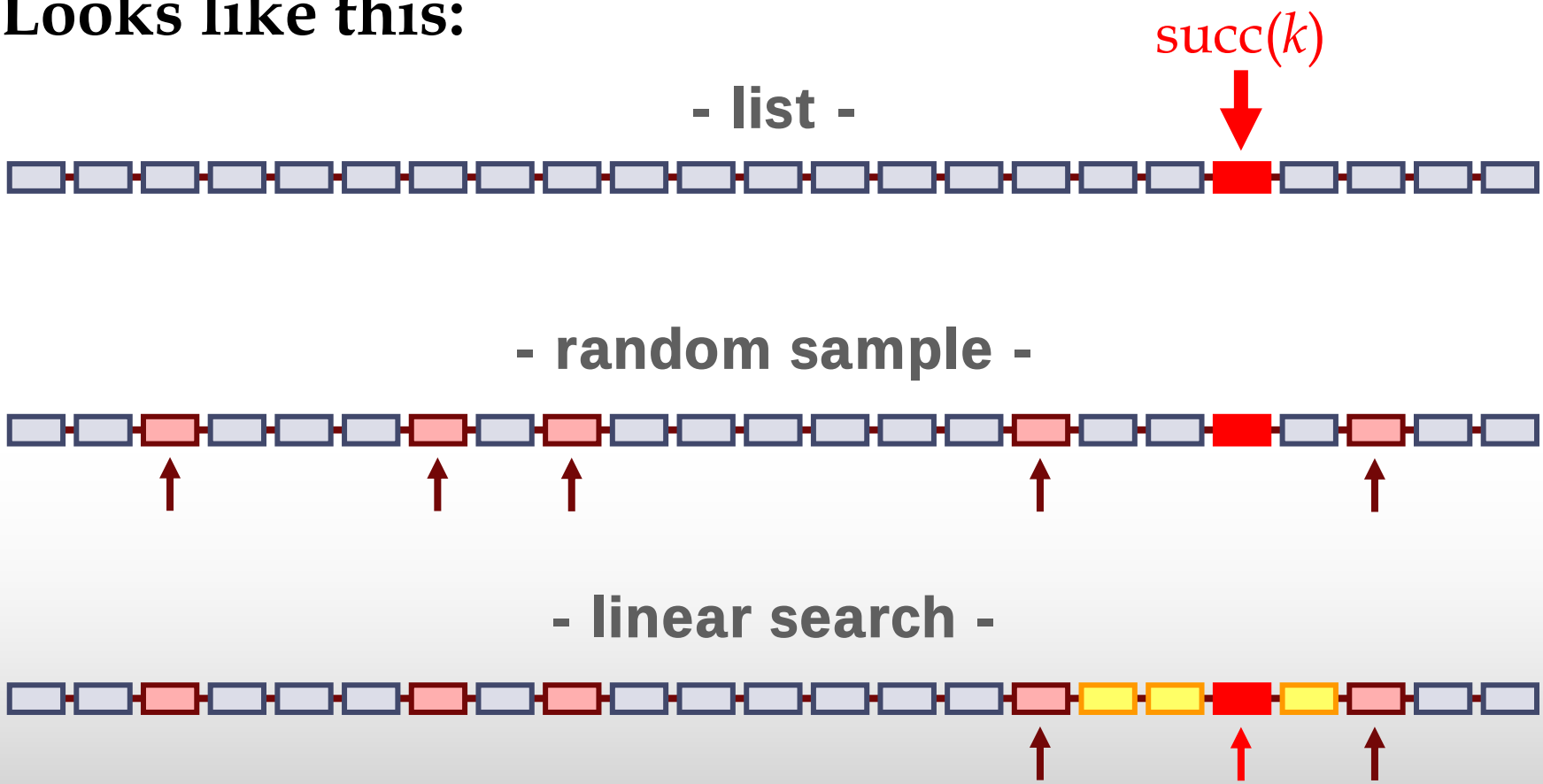
The Search Algorithm

Search Algorithm:

- Pick $\sqrt{n}$ random list elements
- Determine closest predecessor and successor of k (within the random sample)
- Perform plain, linear search towards k (starting from these two elements)

The Search Algorithm (2)

Looks like this:



Analysis

The Algorithm:

- Takes expected $O(\sqrt{n})$ time.
- This is *optimal*.

Why?

Runtime:

Sampling:

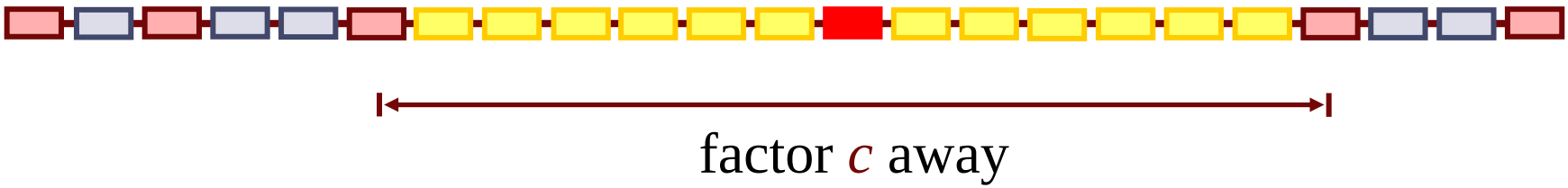
Good case:

Bad case:

Runtime

Runtime Analysis

Runtime depends on "empty" block around $succ(k)$



Probability of no hit within Factor $c\sqrt{n}$ of target is $O(\exp(-c))$

Hence: Expected distance $O(\sqrt{n})$ after $\sqrt{n}$ trials.

Optimality

Why can't we do O(any better)?

The proof employs Yao's minimax principle:

The expected running time of an optimal deterministic algorithm for any chosen input distribution

is a lower bound

on the expected running time of the optimal Las Vegas randomized algorithm.

Constructing the Bound

A Lower Bound for an Opt. Determ. Algorithm

Choose the following scenario:

- Input: Random permutation
- Search for last element

What can the algorithm do?

- Op. "**A**": Visit neighbor of known node
- Op. "**B**": Go to unknown node in memory

Constructing the Bound (2)

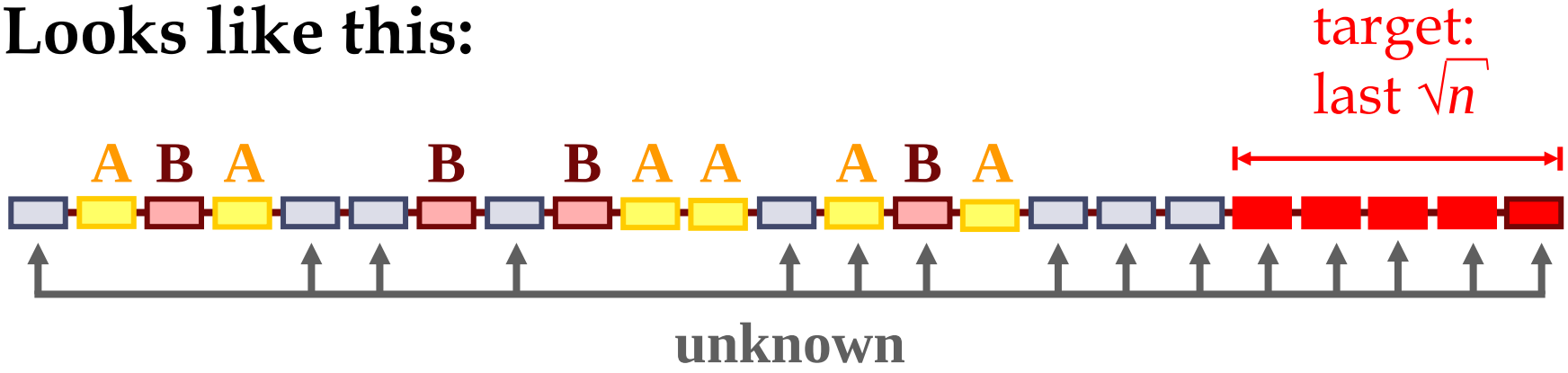
Random input:

- Consider the question: How likely is it, to discover one of the last $\sqrt{n}$ elements?
- With every newly visited node, the chance of discovery increases
- After a A-Op's and b B-Op's:

$$\Pr(\text{"saw not last } \sqrt{n} \text{ elem."}) \geq \left(1 - \frac{\sqrt{n} + a + b}{n}\right)^b$$

Constructing the Bound (3)

Looks like this:



new random choice

$$\Pr(\text{"saw not last } \sqrt{n} \text{ elem."}) \geq \left(1 - \underbrace{\frac{\sqrt{n} + a + b}{n}}\right)^{b}$$

no-go

Pr("no discovery")

Constructing the Bound (3)

Number of B-Ops:

- Expected value of **A** and **B**-op's until visiting one of the last $\sqrt{n}$ is $O(\sqrt{n})$.
- If last $\sqrt{n}$ elements have not been visited
  $\Rightarrow \sqrt{n}$ more **A**-Ops necessary

What's next?

Now: Construct sublinear geometric algorithms

- Similar basic idea
- All with $O(\sqrt{n})$ time complexity
- Some problems are "more general" than successor-searching

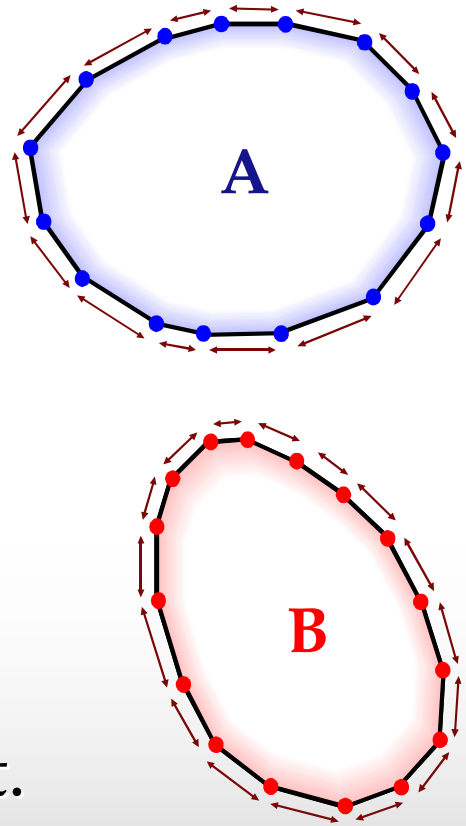
  Thus: run-time also *optimal*

Polygon Intersection

Problem Statement

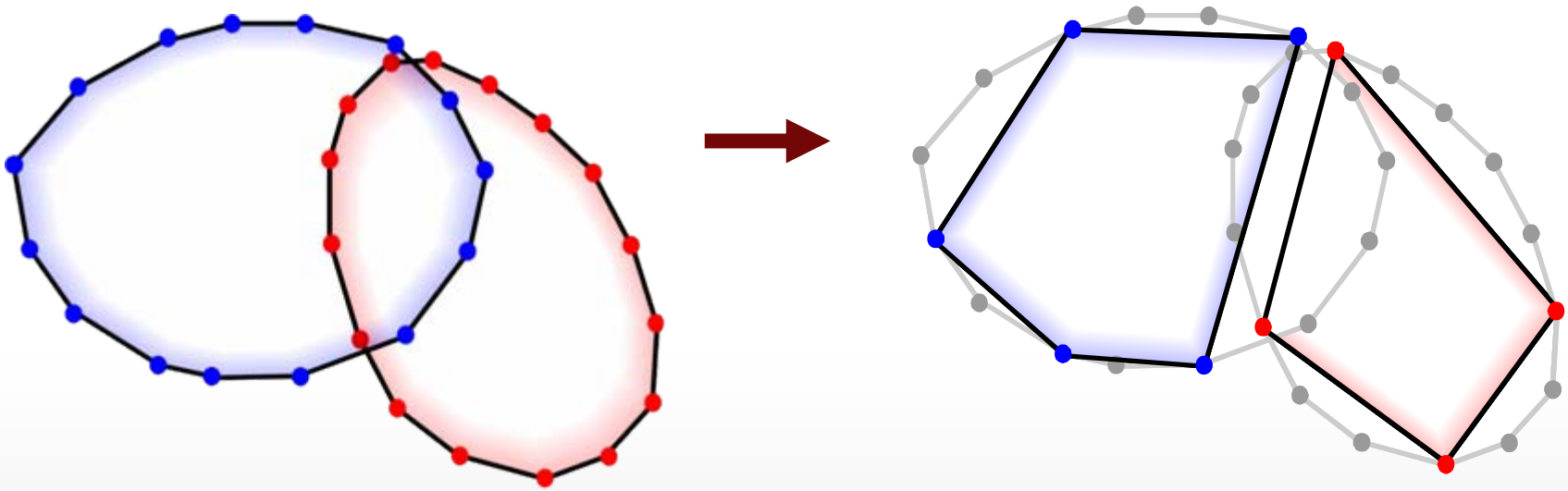
Polygon Intersection:

- Given two polygons **A**, **B**
- Encoded as densely stored linked lists of vertices (same as before)
- Do they intersect?
- If so, report one intersection point.



Idea of the Algorithm

Idea: Employ the same sampling approach

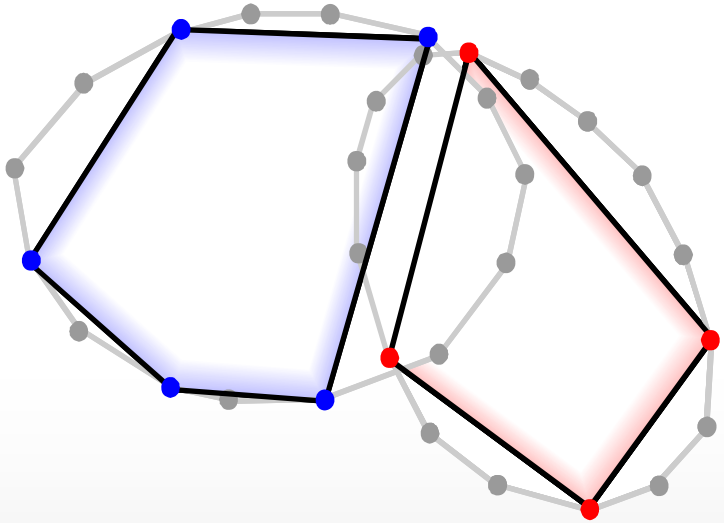


two input polygons

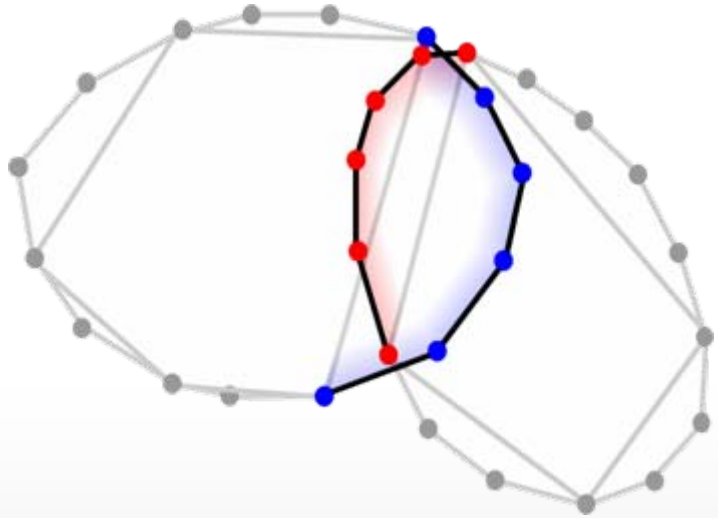
simplified by
random subsampling

Idea of the Algorithm

Two step Approach:

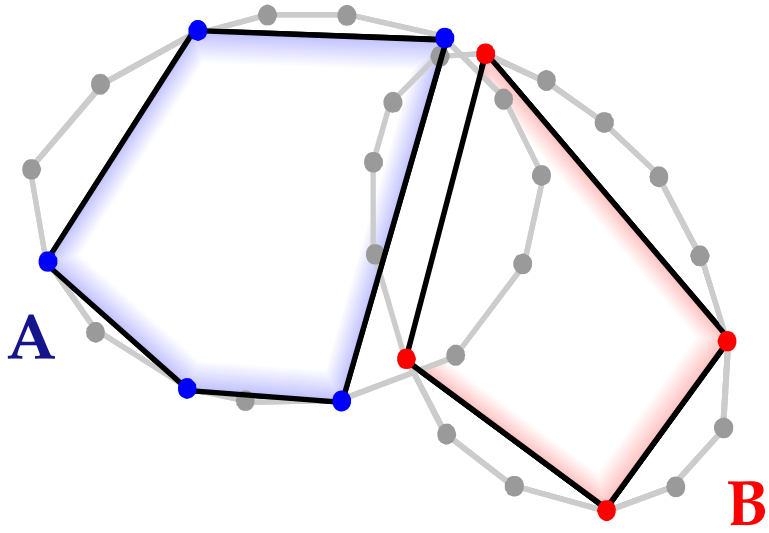


(I) test for simplified polys intersection



(II) test potentially overlapping remaining region

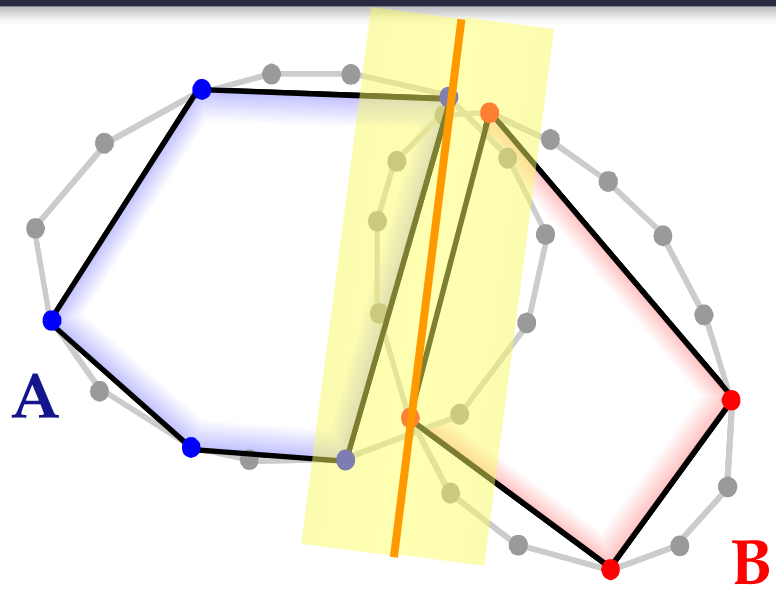
First Step



First Step:

- Random subsampling
- Choose $O(\sqrt{n})$ sample points
- Compute intersection in linear time

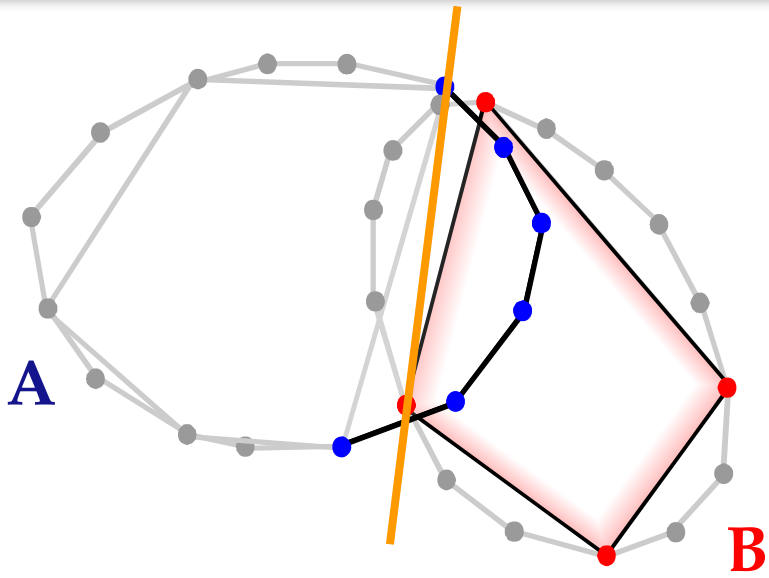
First Step (2)



First Step:

- If no intersection found:
  - Compute separating plane
  - Go to step 2

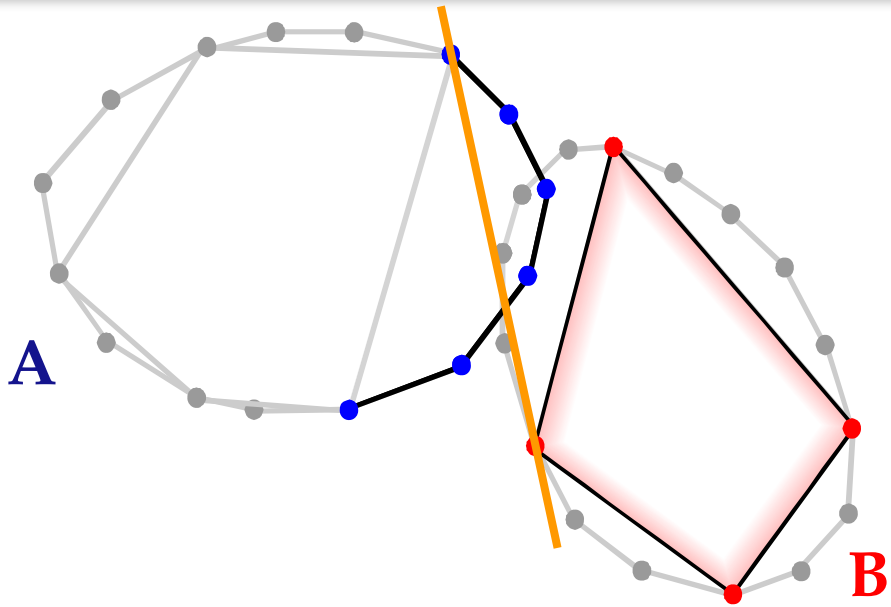
Second Step



Second Step:

- Compute expected $O(\sqrt{n})$ vertices of **A** exceeding separating plane
- Test intersection with *simplified* **B**

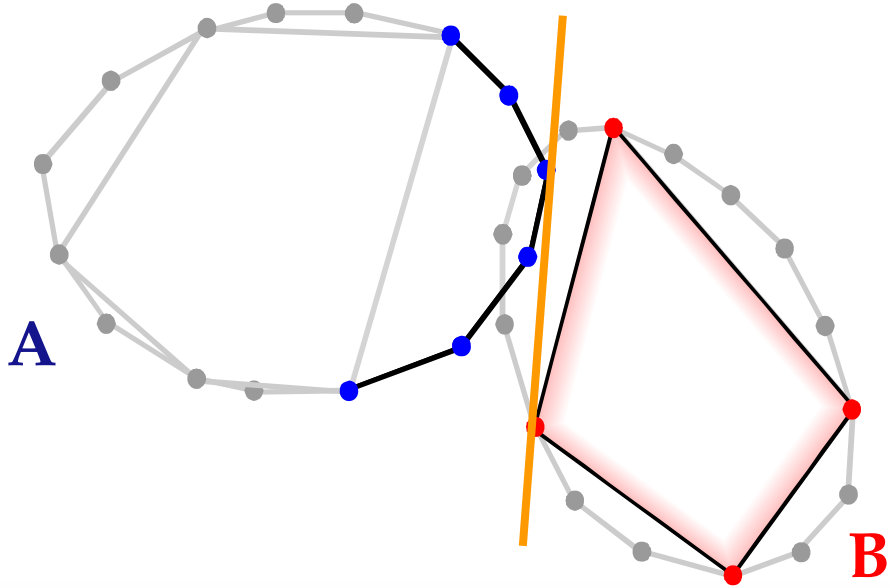
Second Step (2)



Second Step:

- If no intersection is found:
  - Compute new separating plane
  - Test against corresponding part of **B**

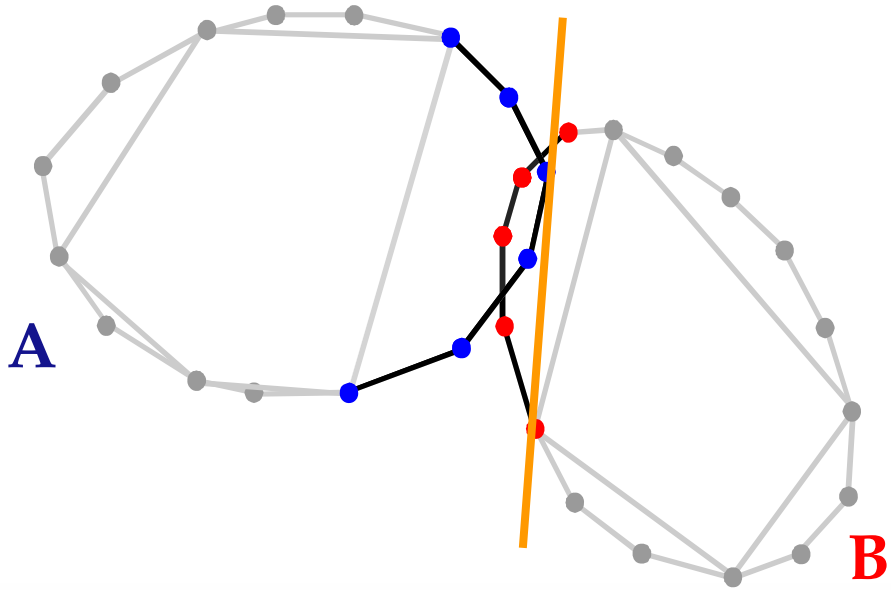
Second Step (3)



Second Step:

- If no intersection is found:
  - Compute new separating plane
  - Test against corresponding part of **B**

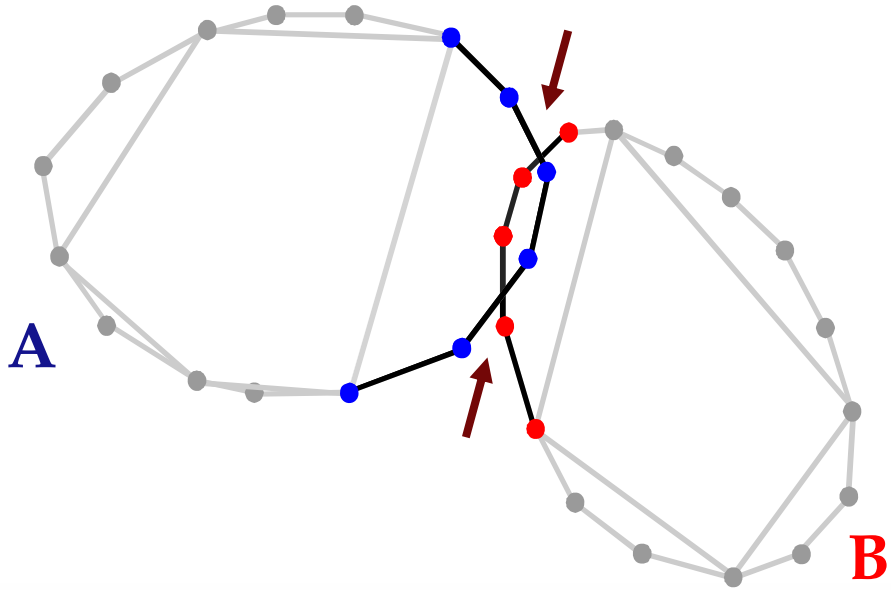
Second Step (4)



Second Step:

- If no intersection is found:
  - Compute new separating plane
  - Test against corresponding part of **B**

Second Step (5)

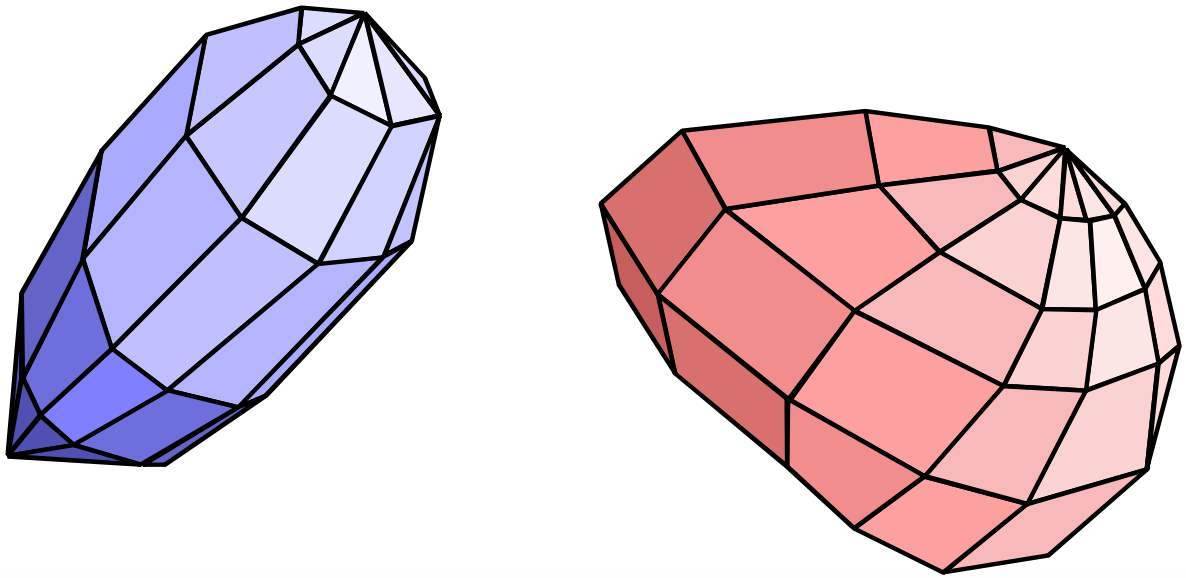


Second Step:

- Report intersection if found
- Otherwise repeat second step with **A** and **B** exchanged for final answer

Polyhedra Intersection

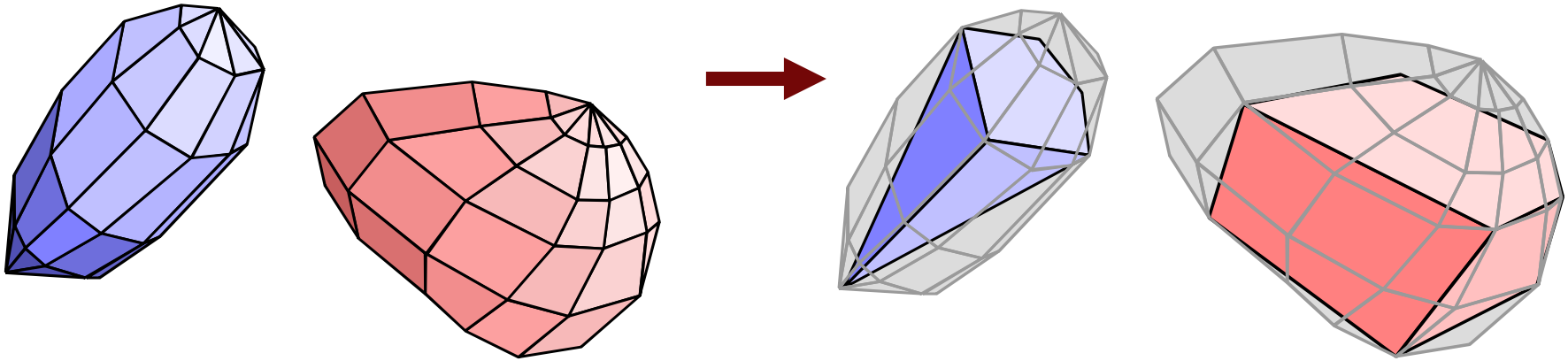
Polyhedral Intersection



Intersection of Two Convex Polyhedra

- Same approach as for polygons
- Some additional technical problems...

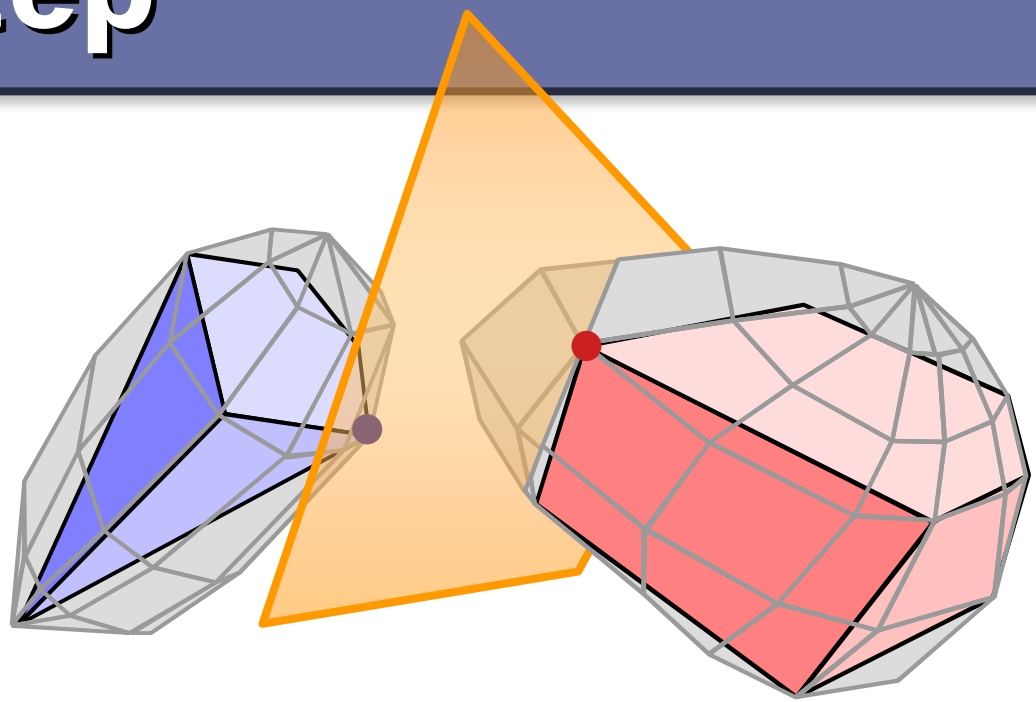
First Step



First Step:

- Simplify Polyhedra
- Take $O(\sqrt{n})$ sample vertices

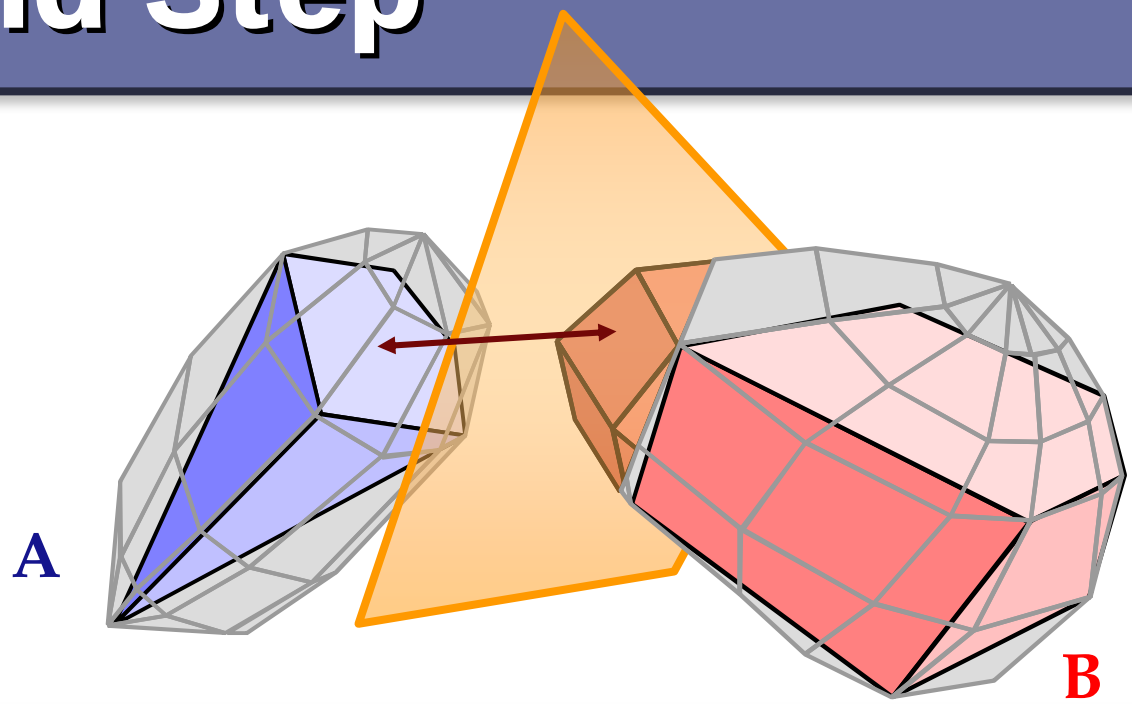
First Step



First Step:

- Test for intersection (lin. programming)
- If no intersection: Compute separating plane

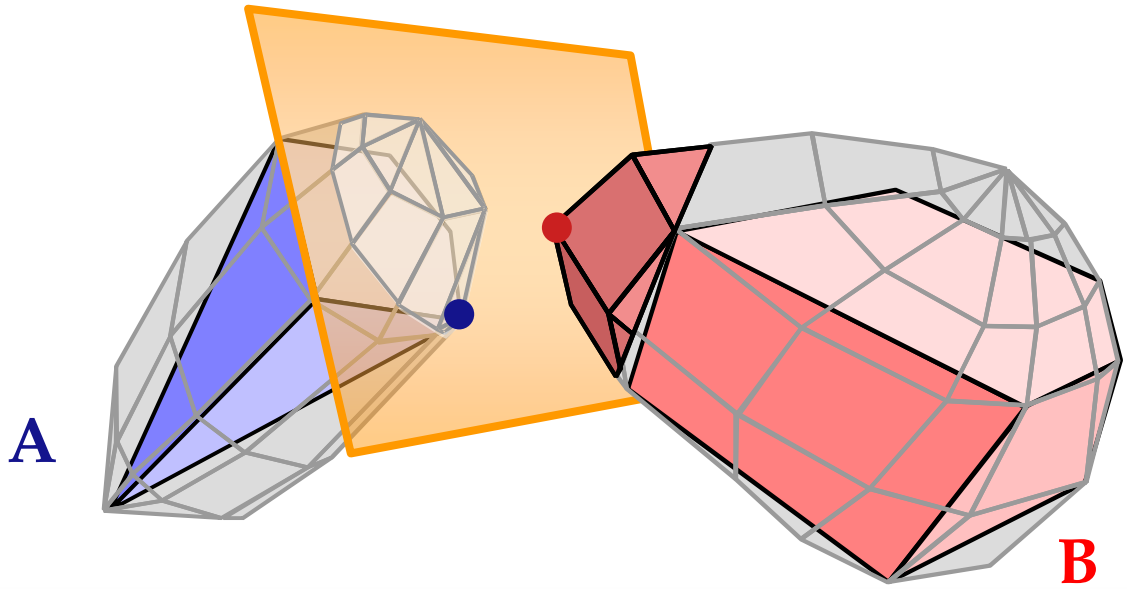
Second Step



Second Step:

- Compute full resolution part of **B** which exceeds separating plane
- Test with simplified **A**

Second Step

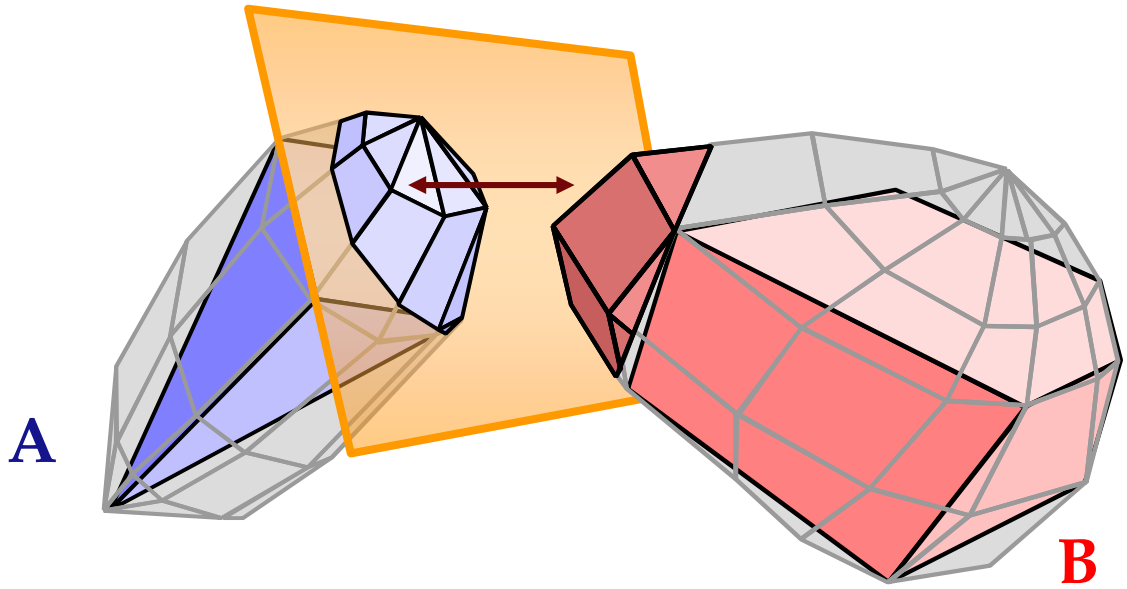


Second Step:

- If no intersection is found:

  Compute new supporting plane...

Second Step



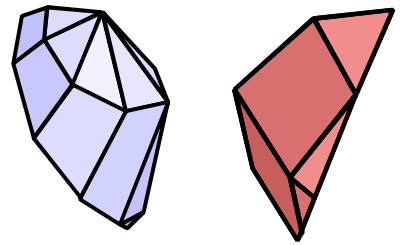
Second Step:

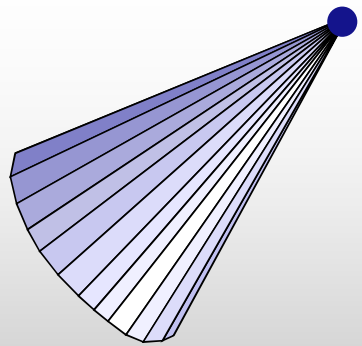
- Test for intersection with full resolution piece of **A** on **B**-side of plane
- If no intersection found: repeat *second step* with **A** and **B** swapped (as for polygons)

Additional Problem

Additional Problem for 3D-Polyhedra

- How compute full resolution cut-off pieces?
- Start at a low-res point
- Adjacency search
- Problem: Starting vertex with large out-degree





Solution

Handle Vertices with Large Out-Degree

- First, sample $O(\sqrt{n})$ edges randomly from polyhedron (only global sampling possible!)
- If out-degree larger than $O(\sqrt{n})$, then one can expect to find some of those edges
- Hence: Start linear search at edges closest to plane (c.f. successor search)
- Otherwise (small out-degree): All edges can be searched linearly

Complexity

Complexity

- The cut-off caps are $O(\sqrt{n})$
  - Intuition: Similar to successor searching
  - Formal proof: See paper
- All other steps are also $O(\sqrt{n})$
- Yields $O(\sqrt{n})$ overall complexity
- The problem is a generalization of successor searching. Hence: *Optimal complexity*.

Ray Shooting & Applications

Ray Shooting

Same Technique Can Handle Ray Shooting:

- Rays are very skinny polyhedra
- Can use the same algorithm
- Computes intersections in $O(\sqrt{n})$ time
- This is also *optimal*

Applications

Point Location

- Point location in Delaunay triangulations
- Shoot vertical ray towards convex hull of $(x, y, x^2 + y^2)$
- Similar construction also works for Voronoi Diagrams
- Yields point location algorithms with *optimal* $O(\sqrt{n})$ complexity

Other Applications

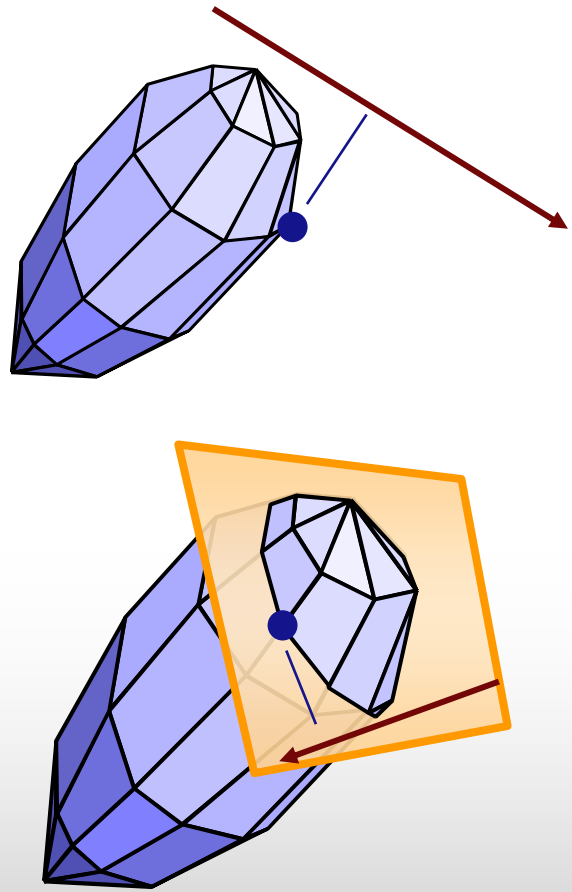
Nearest Neighbor Search:

- Find nearest neighbor in polyhedron to a given point in space in $O(\sqrt{n})$ time
- Uses (slightly) modified polyhedron intersection algorithm

Other Applications (2)

Direction Search

- Find point with maximum projection on a line
- Find point with maximum projection on a line restricting input polyhedron to a plane
- Both in $O(\sqrt{n})$



Volume Approximation

Volume Approximation

The Task:

- Determine volume of polyhedron
- With accuracy up to factor ε

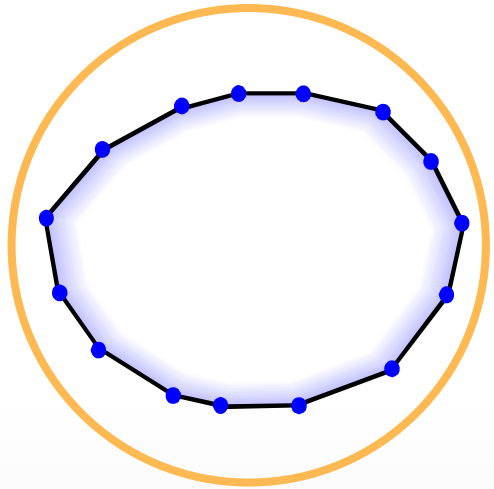
Main Idea:

- Dudley approximation
- Non-uniform rescaling to be able to guarantee ε-precision

Dudley Approximation

Dudley Approximation of Convex Polyhedra

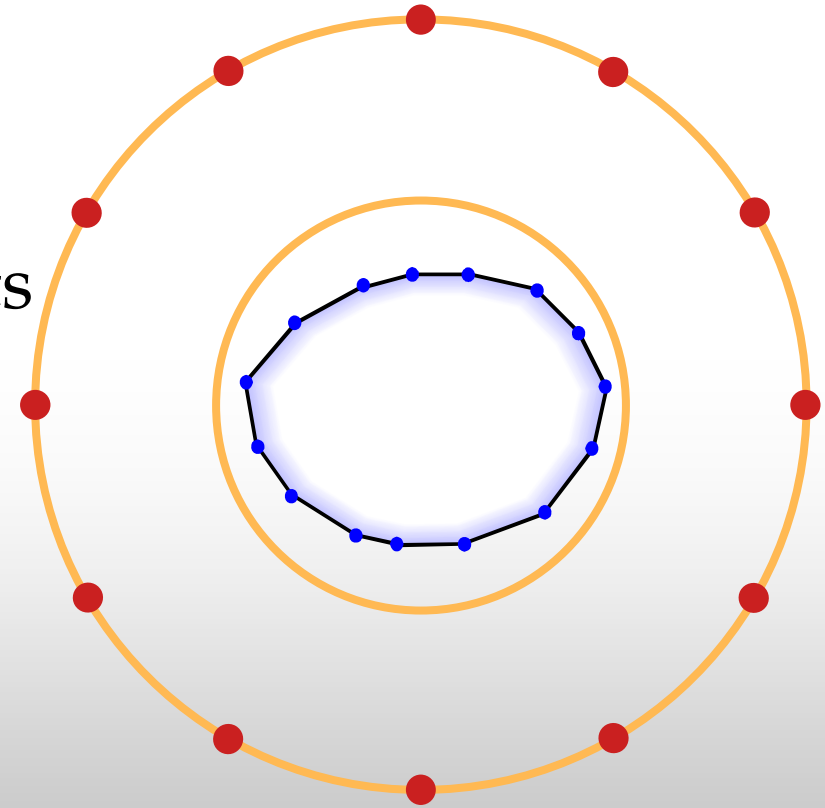
- Given a convex polygon / polyhedron enclosed in a unit sphere
- An approximation of complexity $O(1/\varepsilon^{(d-1)/2})$ with max. Hausdorff-distance ε can be constructed



Dudley Approximation (2)

Dudley Approximation of Convex Polyhedra

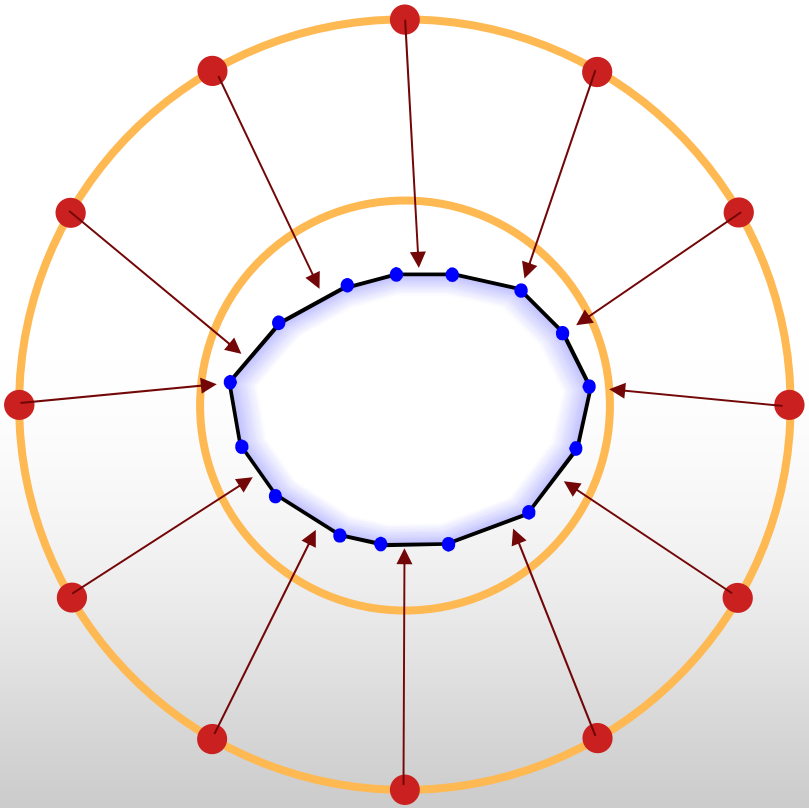
$O(1/\varepsilon^{(d-1)/2})$ uniformely distributed sample points



Dudley Approximation (3)

Dudley Approximation of Convex Polyhedra

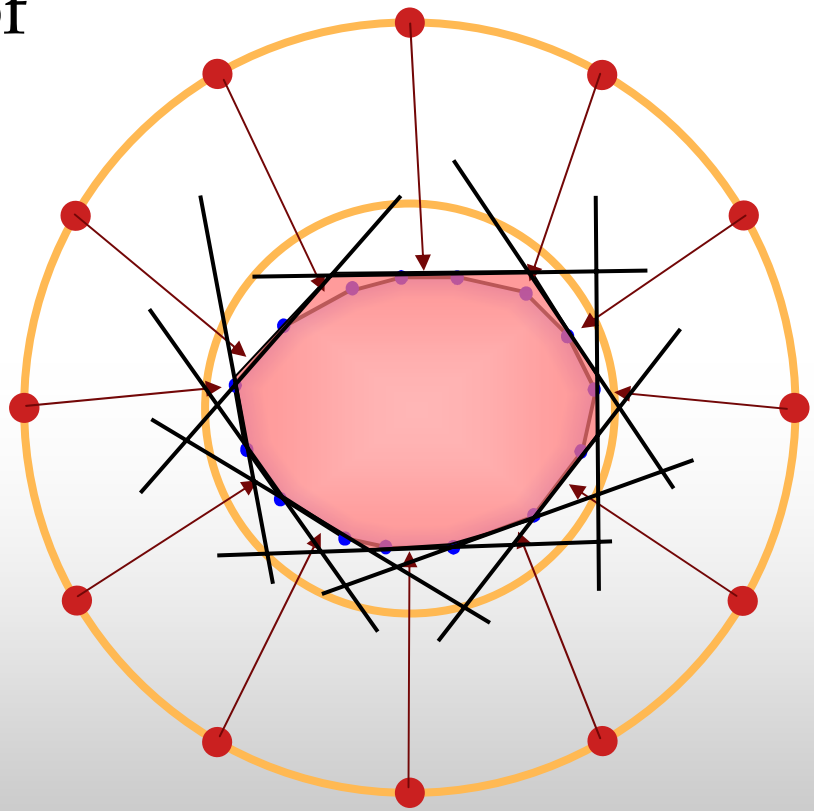
find nearest neighbor of each point



Dudley Approximation (4)

Dudley Approximation of Convex Polyhedra

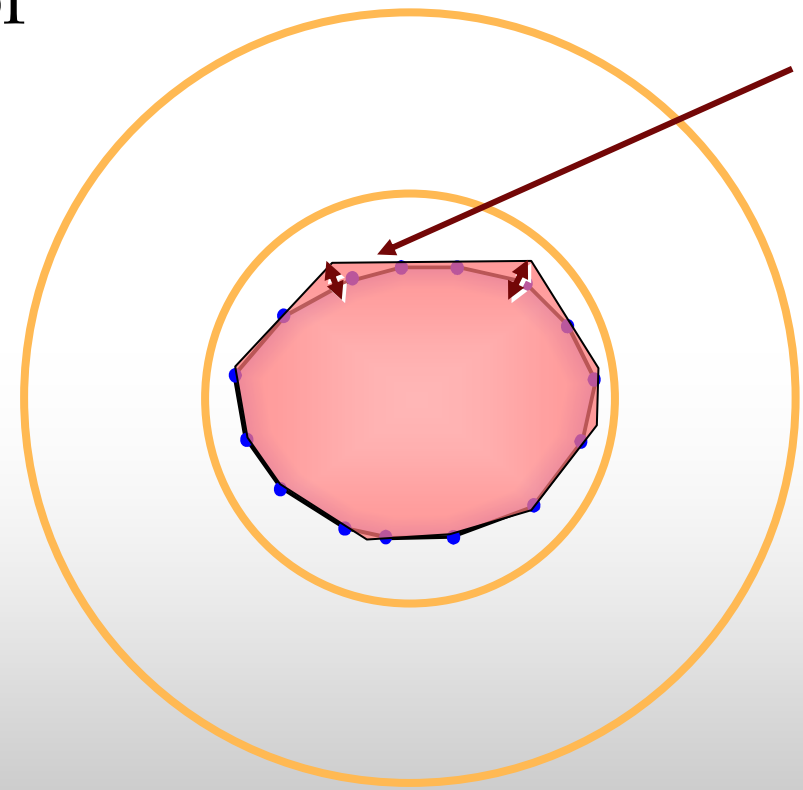
convex hull of "supporting" planes



Dudley Approximation (5)

Dudley Approximation of Convex Polyhedra

convex hull of "supporting" planes

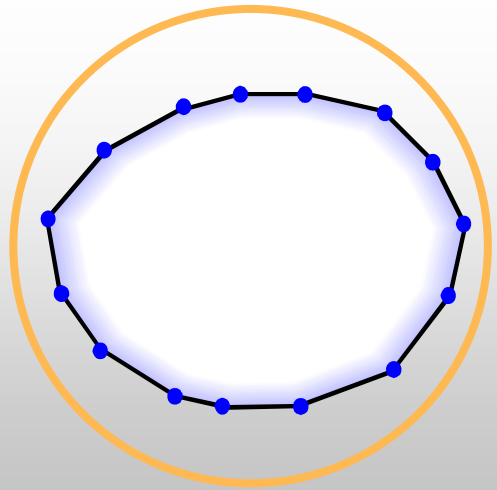


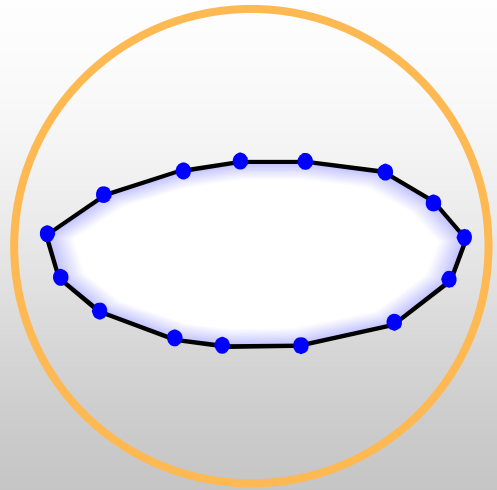
maximum error: ε (with respect to unit sphere)

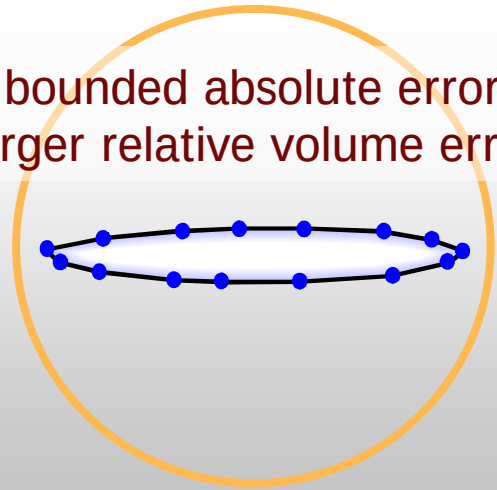
Volume Approximation

Volume Approximation using Dudley Construction

- Observation: All steps in Dudley Construction can be peformed in $O(\sqrt{n})$ time
- Remaining problem: Poor approximation for skinny polyhedra

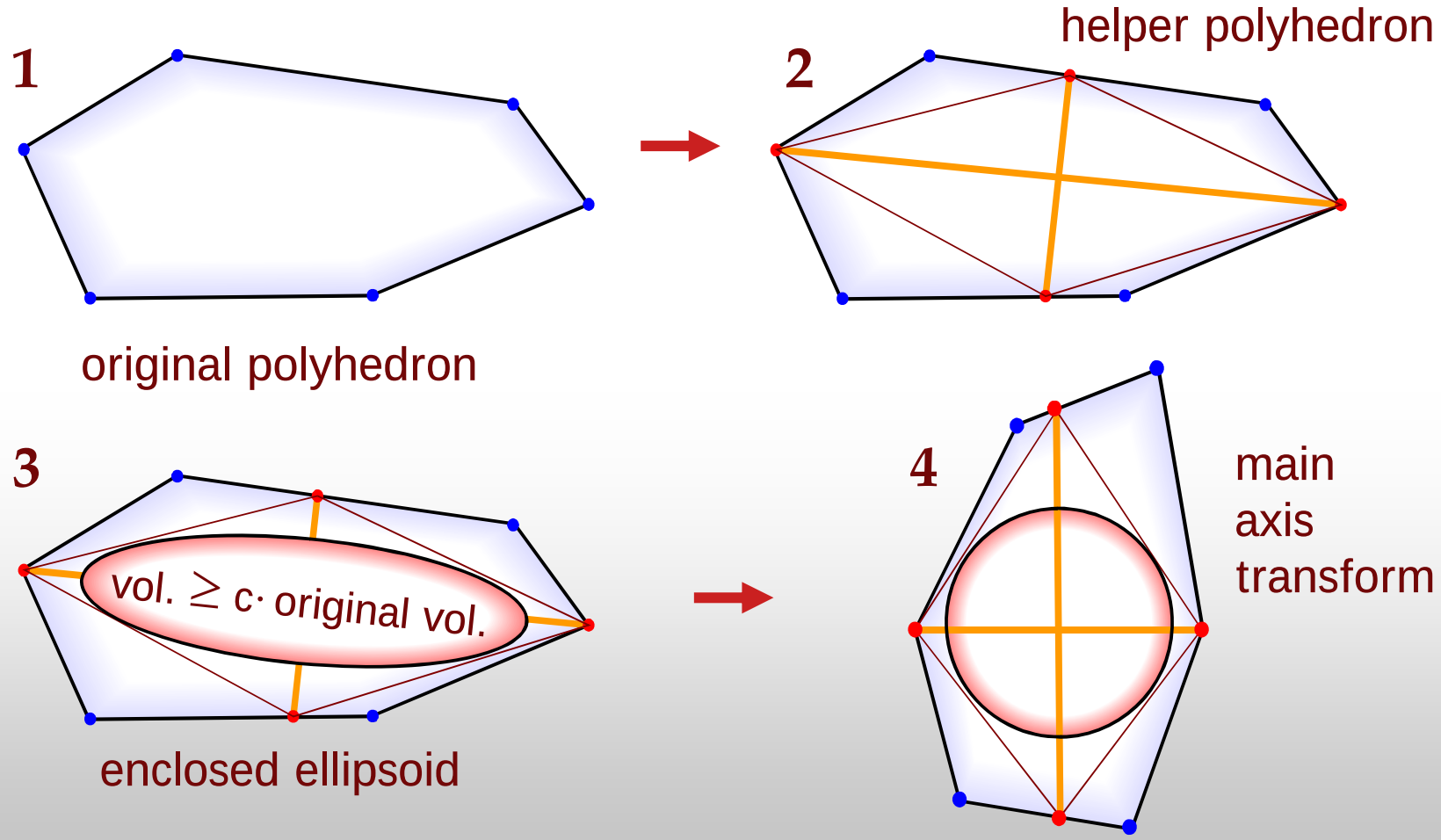






Rescaling

Solution: Non-Uniform Rescaling



Result

Rescaling:

- Enclosed ellipsoid has
  $volume \geq c \cdot original\ volume$
- Volume can be measured with arbitrary, constant accuracy by rescaling

Overall:

- Sublinear operations only
- Overall: $O(\varepsilon^{-1}\sqrt{n})$ running time for ε-approximation of volume

Shortest-Path Approximation

Approx. Shortest Path

The Task:

- Compute *shortest path* between *two surface points* on *convex polyhedron*
- Approximate up to factor ε
- Allow path to leave surface of polyhedron (otherwise, any path might be of complexity $\Omega(n)$)

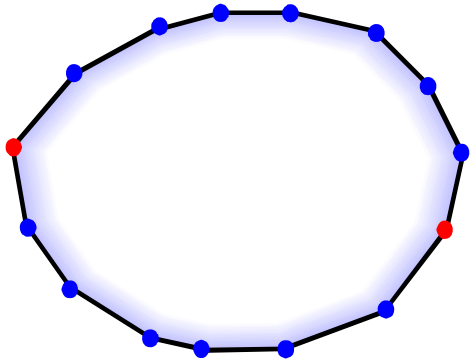
How does it work?

Main Idea:

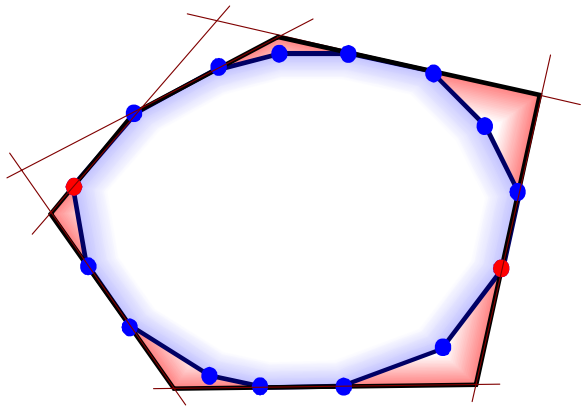
- Same basic idea: Dudley approximation
  - Construct simplified polyhedron
  - Constant number of faces

  $\Rightarrow$ Can then use standard algorithm to compute shortest path
- Problem: Guaranteeing accuracy (again)
- Solution: Clipping to O(*path-length*)-Box

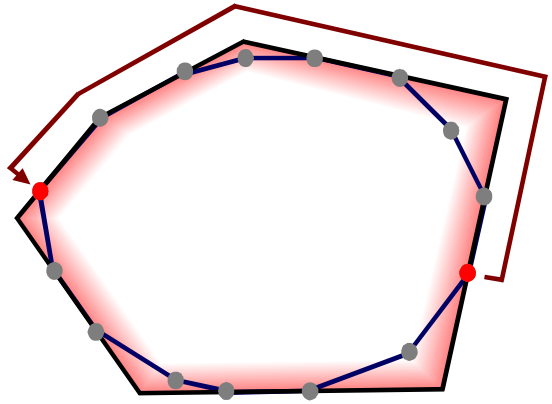
Using the Dudley Approx.



original



simplified

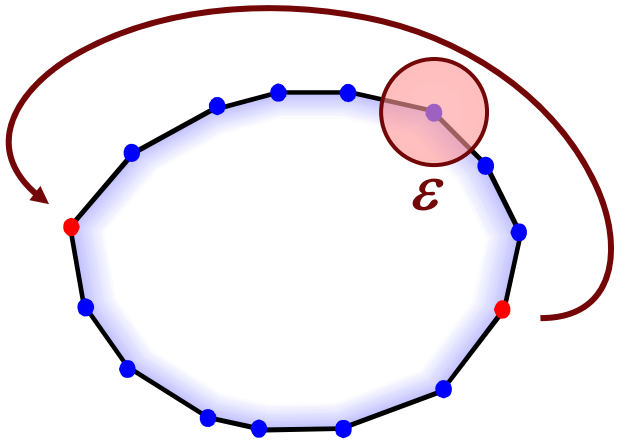


approximate path

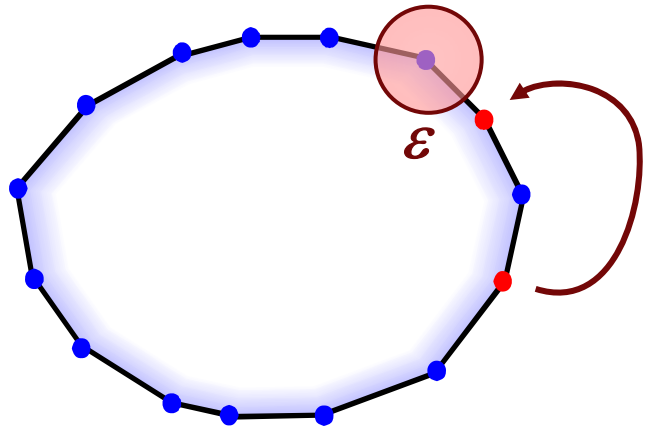
Dudley Approximation:

- Simplify polyhedron (Dudley)
- Build path along simplified w. std. method

The Problem



long path – smaller relative error



short path – large relative error

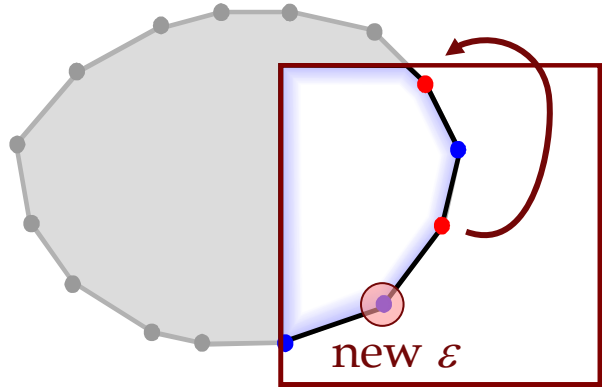
Remaining Problem: Relative Accuracy

- ε-approx for long paths
- Larger relative error for short paths

Solution

Clipping before Dudleying

- Estimate length of shortest path up to factor 8
- Place box of side length $16 \cdot est$ around start point



- Clip simplified polyhedron to box (additional constraints during computation)
- By scaling ε correspondingly, this guarantees ε-approximation of shortest path

Result

Overall result:

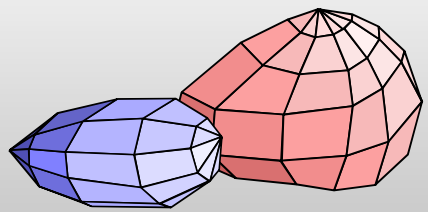
- Can approximate shortest path up to factor $(1 + \varepsilon)$
- Using time $O(\varepsilon^{-5/4} \sqrt{n}) + f(\varepsilon^{-5/4})$ where f is time for exact solution
- Known: $f(k) \in O(k \log^2 k)$
- Authors use *modified* Dudley construction to achieve given bounds; see paper for details.

Conclusions

Conclusions

Sublinear Algorithms for Geometric Problems

- Convex polygons / polyhedra intersection
- Ray shooting (convex P's, and similar problems)
- Volume approximation (convex P's)
- Shortest path approximation (convex P's)

Properties

- All in $O(\sqrt{n})$
- Techniques are generalizations of the successor searching algorithm