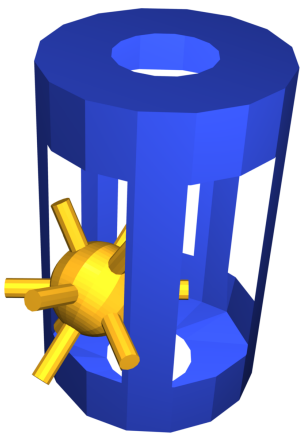
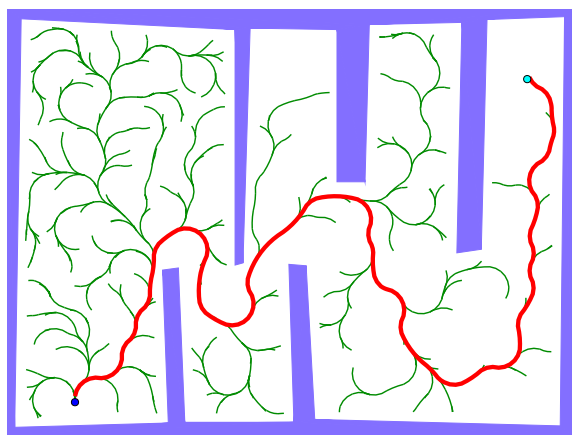
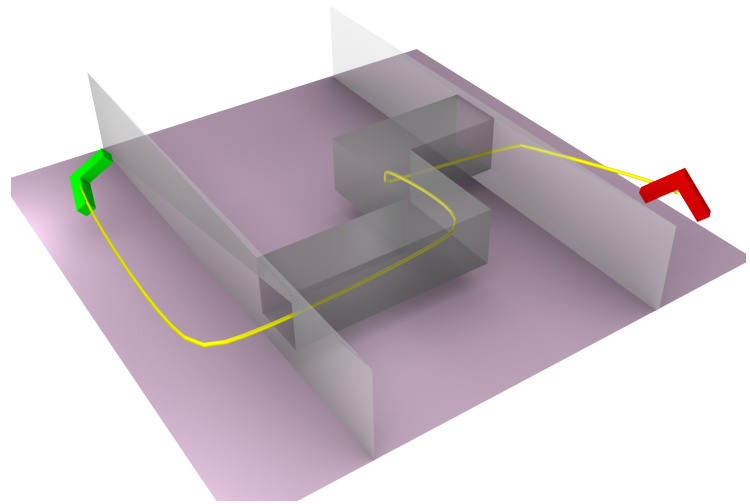
Motion planning: combinatorial path planning

Vojtěch Vonásek

Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

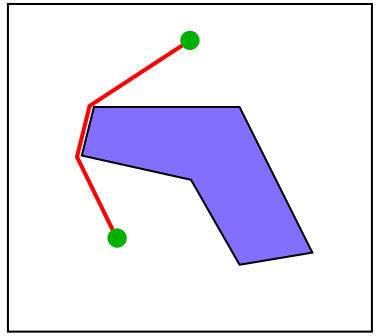






Combinatorial (geometric) path planning

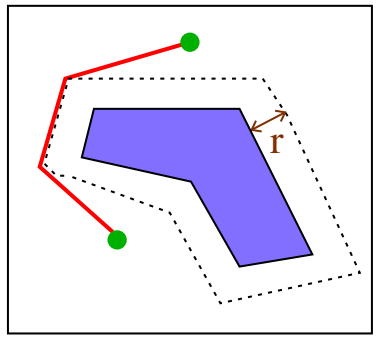
- Assume point/disc robots
- Use geometric (usually polygonal) representation of W
- In these cases, representation of $\mathcal{W}$ is also representation of $\mathcal{C}$
- The representation is explicit $\rightarrow$ enumeration of obstacles is easy
- Voronoi diagram, Visibility map, Decomposition-based methods



2D $\mathcal{W}$ and a path for the point robot

Point robot in 2D or 3D $\mathcal{W}$

- The map of $\mathcal{W}$ is also representation of $\mathcal{C}$
- Polygons/polyhedrons are suitable



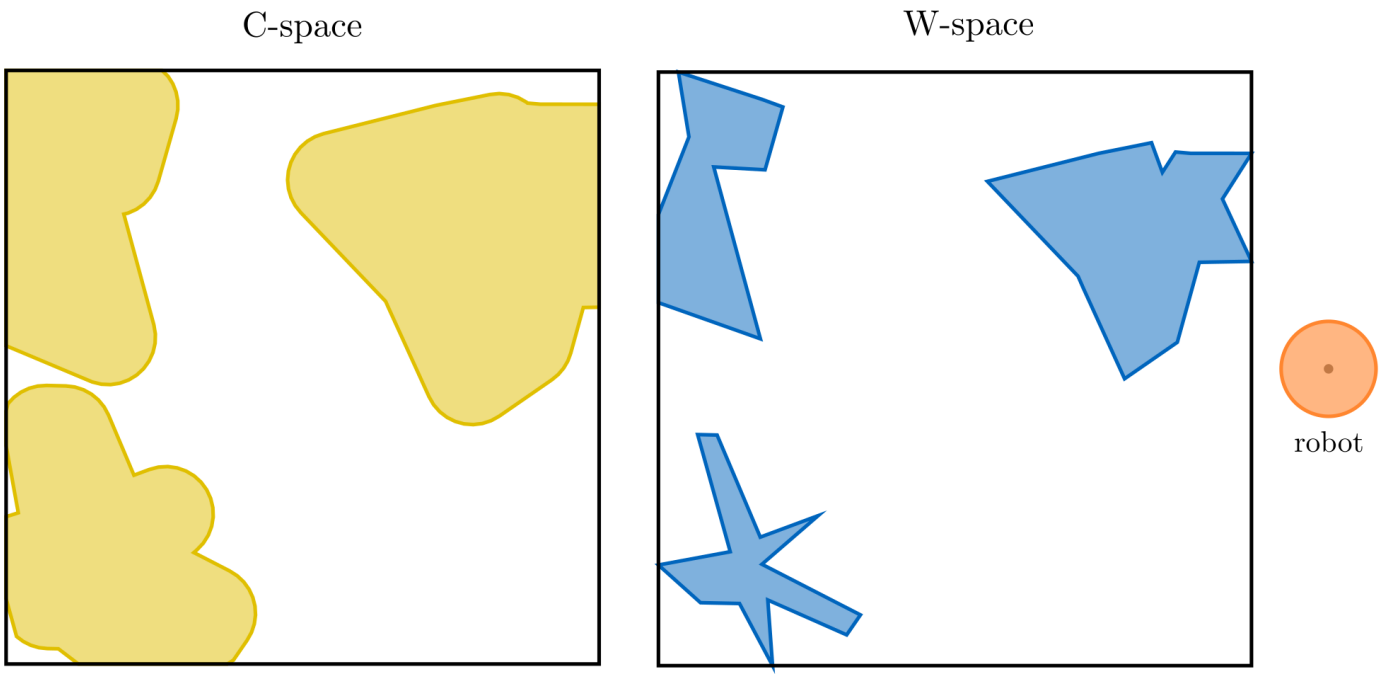
2D $\mathcal{W}$ + enlargement of obstacles, and a path for the disc robot

Disc/sphere robot in 2D or 3D $\mathcal{W}$

- The obstacles are “enlarged” by the radius of the robot (Minkowski sum)

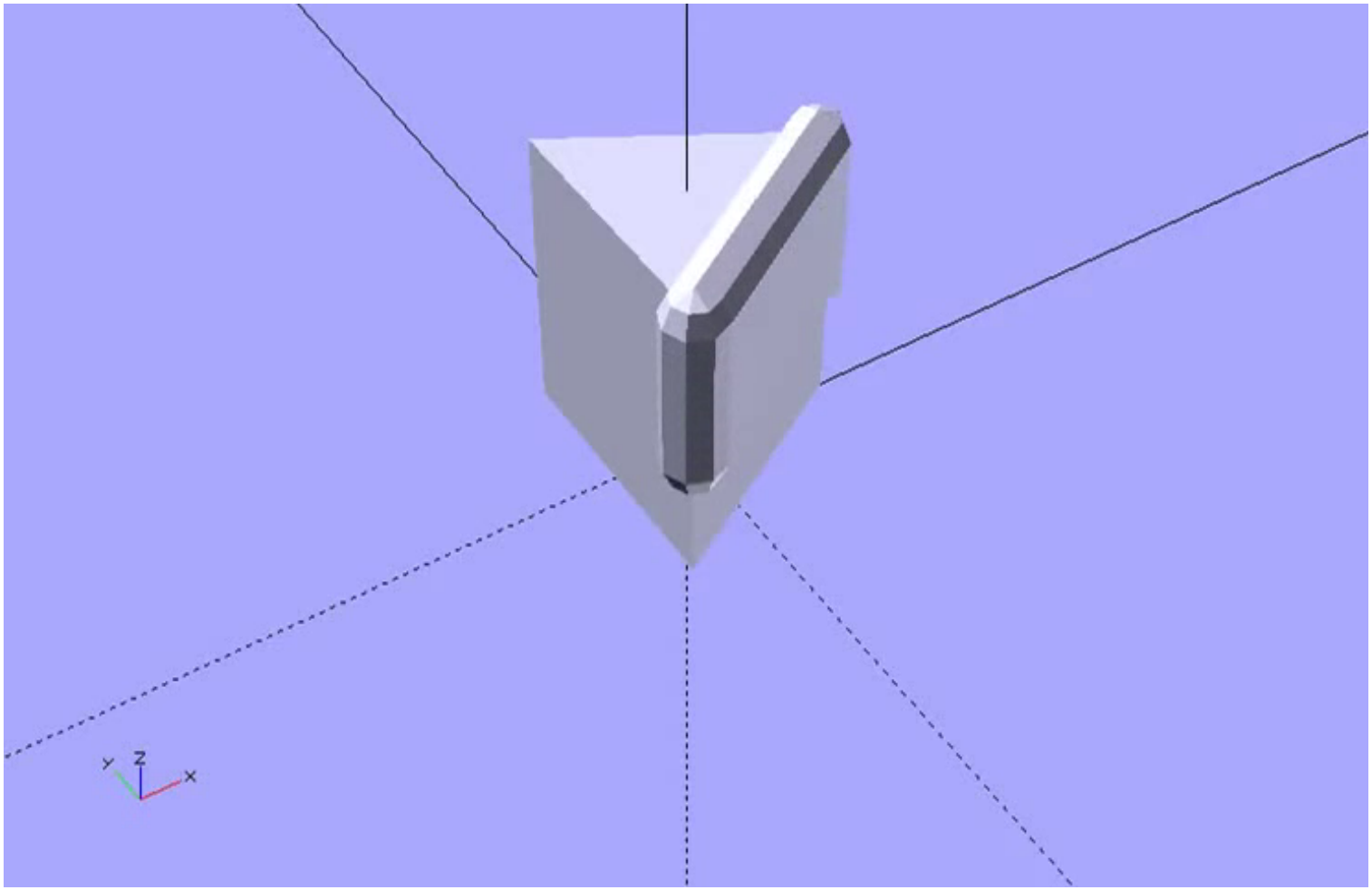
Point and disc robots

- The obstacles are “enlarged” by the radius of the robot (Minkowski sum)
- For point robot, $\mathcal{W} = \mathcal{C}$
- For disc robot, $\mathcal{W}$ enlarged with Minkowski sum leads to $\mathcal{C}$
  - Easy, robust implementation



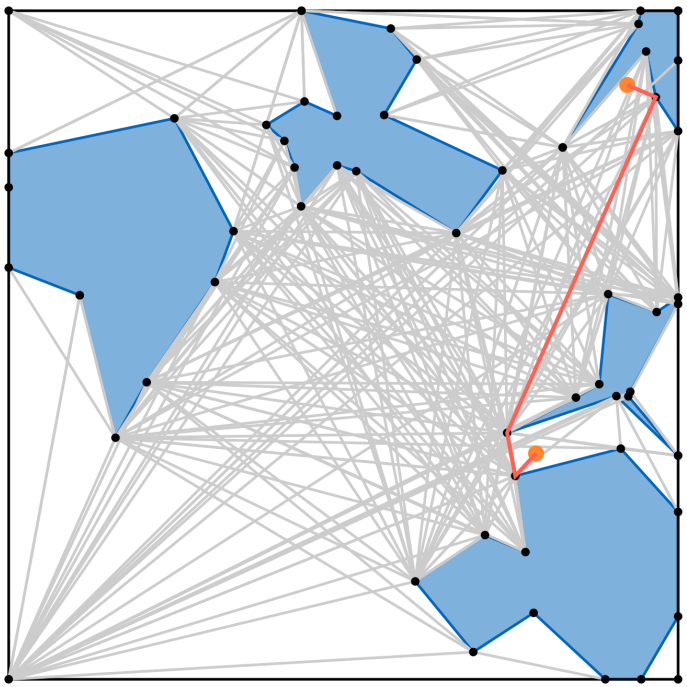
Point and disc robots

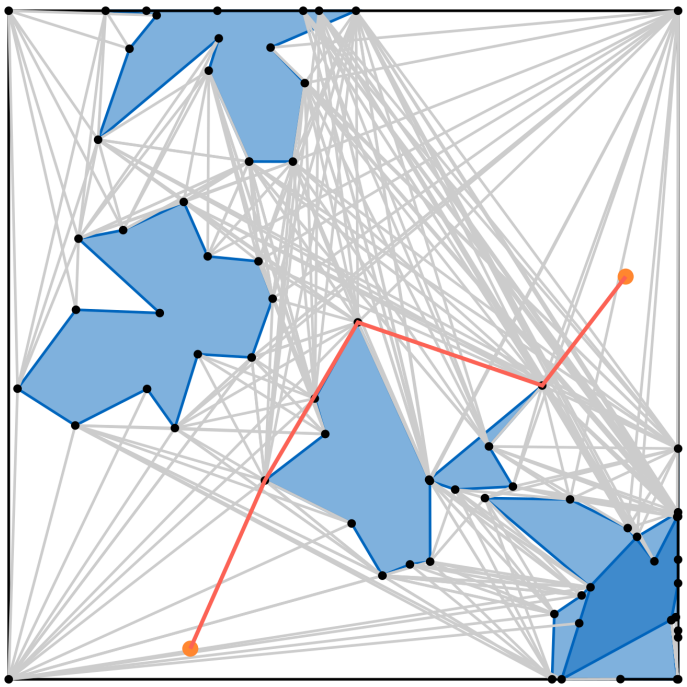
- The obstacles are “enlarged” by the radius of the robot (Minkowski sum)
- For point robot, $\mathcal{W} = \mathcal{C}$
- For disc robot, $\mathcal{W}$ enlarged with Minkowski sum leads to $\mathcal{C}$
  - Easy, robust implementation



Visibility graph (VG)

- Two points v_i, v_j are visible $\iff$ $(sv_i + (1-s)v_j) \in \mathcal{C}_{\text{free}}, \quad s \in [0,1]$
- Visibility graph (V, E), V are vertices of polygons (optionally +boundary), E are edges between visible points
- Start/goal are connected in the same manner to visible vertices





- No clearance
- Suitable only for 2D

Visibility graph (VG)

Naïve implementation

- n^2 pairs of vertices
- Check each of them for collision
- Collision check is $O(n)$
- Total complexity $O(n^3)$

Input: polygonal obstacle
Output: visibility graph $G = (V, E)$

```
1 V = all vertices of polygonal obstacles
2 foreach u, v ∈ V do
3     foreach obstacle edge e do
4         if segment u, v intersects e then
5             continue;
6         add edge u, v to E
```

Fast methods

- Lee's algorithm[1] $O(n^2 \log n)$
- Overmars/Welz method[2] $O(n^2)$
- Ghosh/Mount method[3] $O(|E| n \log n)$

[1] D. Coleman. *Lee's O(n² log n) Visibility Graph Algorithm Implementation and Analysis*. 2012.

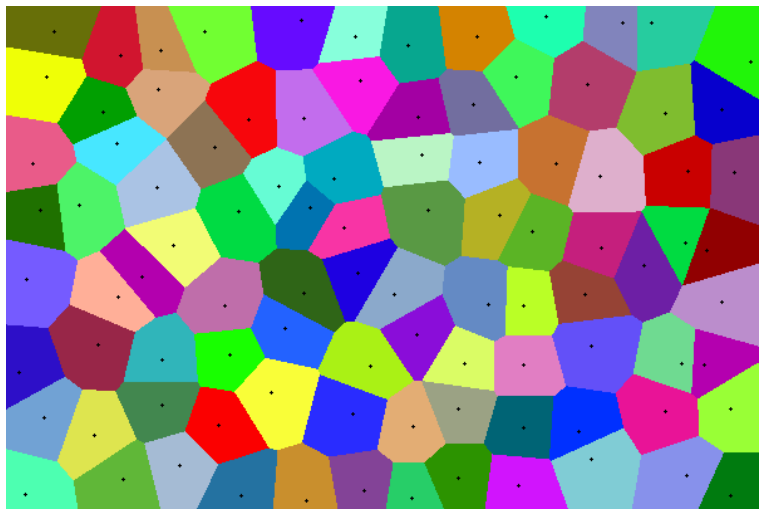
[2] M. H. Overmars and E. Welzl. "New methods for computing visibility graphs". In: *SCG '88*. 1988.

[3] S. K. Ghosh and D. M. Mount. "An output sensitive algorithm for computing visibility graphs". In: *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. 1987, pp. 11–19. DOI: 10.1109/SFCS.1987.6.

Voronoi diagram

- Let $P = v_1, \ldots, v_n$ are n distinct points ("input sites") in a $d-$dimensional space
- Voronoi Diagram (VD) divides P into n cells $V(p_i)$

$$V(p_i) = \{x \in \mathbf{R}^d : ||x - p_i|| \leq ||x - p_j|| \ \ \forall j \leq n\}$$



- Cells are convex
- Used in point location (1-nn search), closest-pair search, spatial analysis
- Construction using Fortune's method[a] in $O(n \log n)$

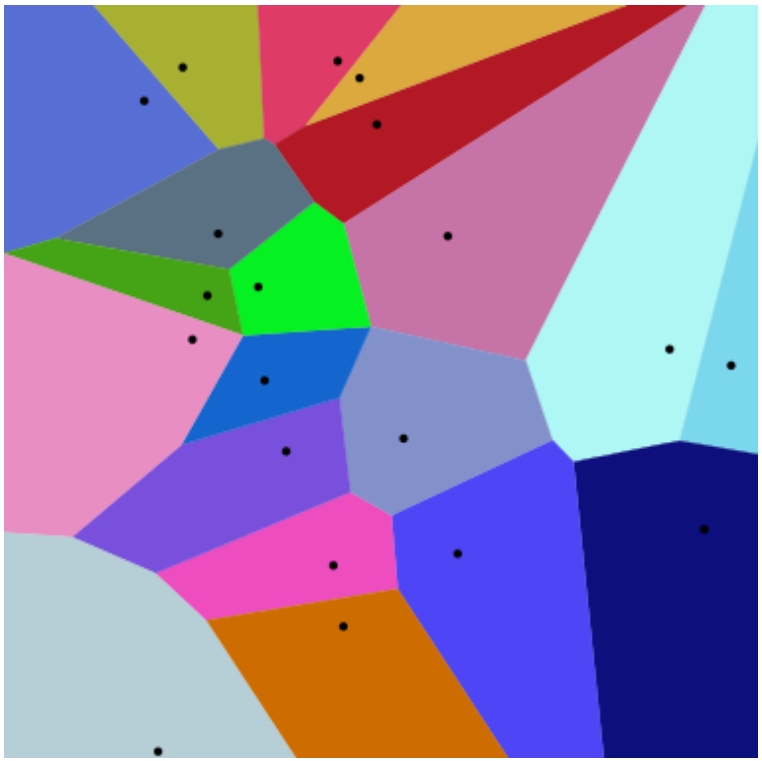
[a] S. Fortune. "A Sweepline Algorithm for Voronoi Diagrams". In: *Algorithmica* 2.1-4 (1987), pp. 153–174. DOI: 10.1007/BF01840357.

Voronoi diagram

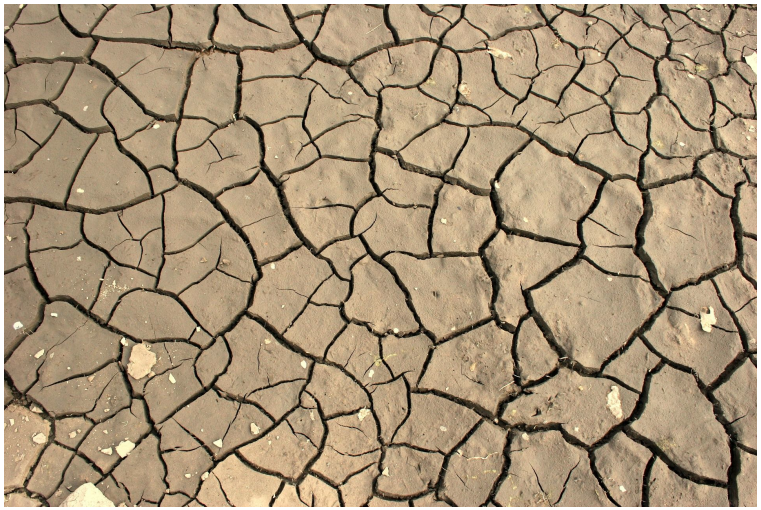
- Let $P = v_1, \ldots, v_n$ are n distinct points ("input sites") in a $d-$dimensional space
- Voronoi Diagram (VD) divides P into n cells $V(p_i)$

$$V(p_i) = \{x \in \mathbf{R}^d : ||x - p_i|| \leq ||x - p_j|| \ \ \forall j \leq n\}$$

- Note, that other metrics can be considered



Voronoi diagrams are everywhere

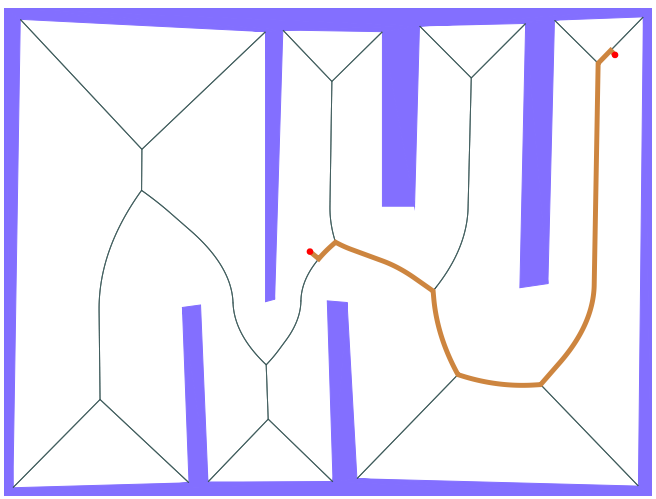


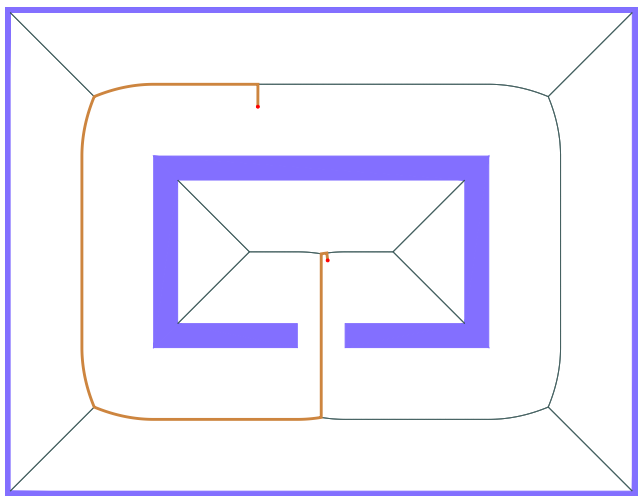
Voronoi diagram in robotics

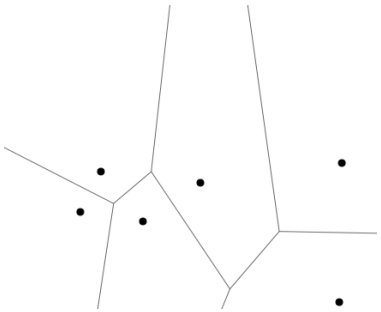
- (Basic) Voronoi diagram: computed on points
- Generalized Voronoi Diagram: computed on e.g., points + weights, segments, spheres, . . .

Segment Voronoi Diagram (SVD)

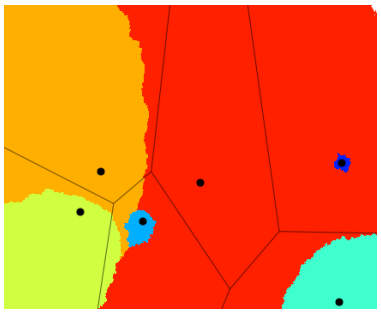
- computed on line-segments describing obstacles
- requires polygonal map or line/segment map
- ✓ Maximal clearance
  - largest distance between a path and the nearest obstacle
- Is it optimal? Is it complete?



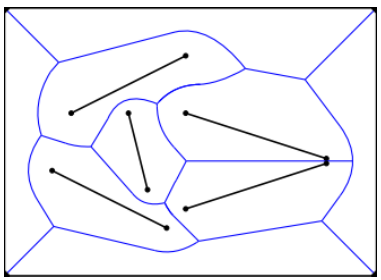




Classic VD



Weighted VD

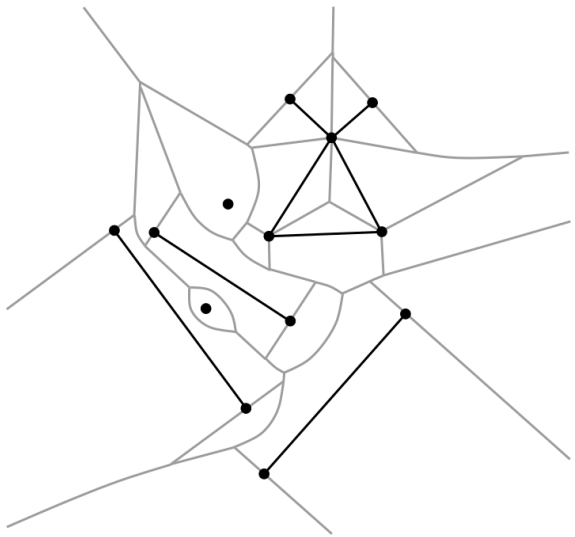


Segment VD

Segment Voronoi diagram: complexity

Algorithms for computing Segment Voronoi diagram of n segments

- Lee & Drysdale[4] $O(n \log^2 n)$, no intersections
- Karavelas[5] $O((n+m) \log^2 n)$, m intersections between segments



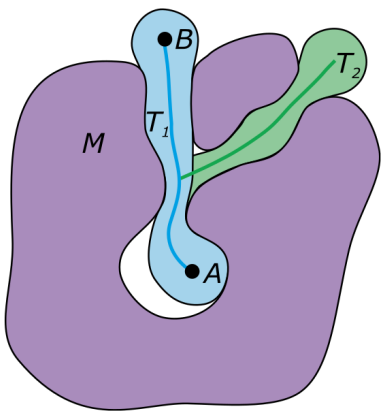
Karavelas 2004

[4] D. T. Lee and R. L. Drysdale Iii. "Generalization of Voronoi Diagrams in the Plane". In: *SIAM Journal on Computing* 10.1 (Feb. 1981), pp. 73–87. DOI: 10.1137/0210006.

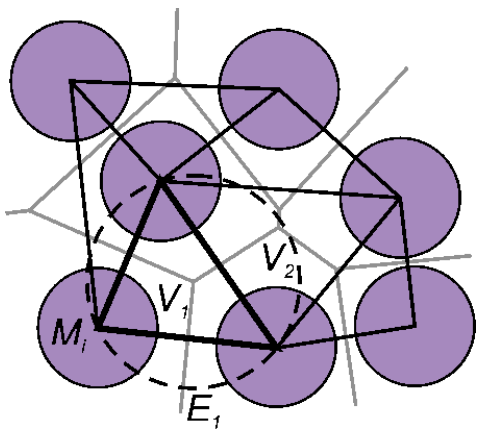
[5] M. I. Karavelas. "A robust and efficient implementation for the segment Voronoi diagram". In: *International symposium on Voronoi diagrams in science and engineering*. Vol. 1. 2004.

Voronoi diagrams in bioinformatics

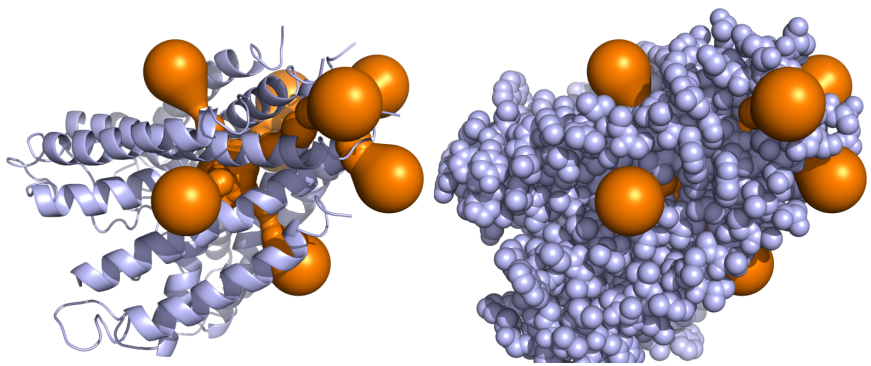
- Proteins are modeled using hard-sphere model
- Weighted Voronoi diagram of the spheres (weight is the atom radii — Van der Waals radii)
- Path in the Voronoi diagram reveals “void space” and “tunnels”
- Tunnel properties (e.g. bottleneck) estimate possibility of interaction between protein and a ligand
- Implemented, e.g., in CAVER[6]



Tunnels*



Voronoi diagram *

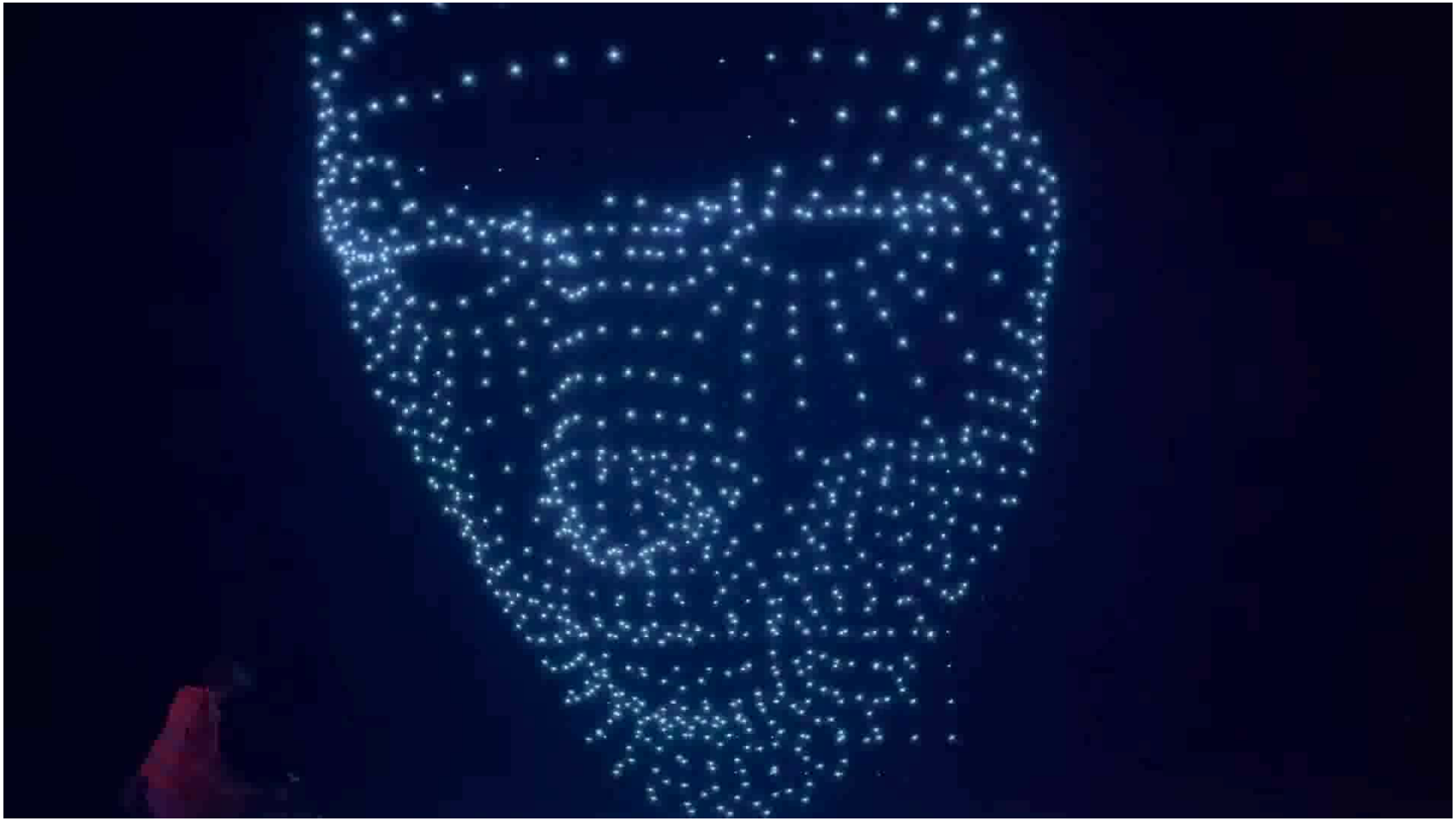


Tunnels on 1BL8

[6] A. Pavelka et al. “CAVER: algorithms for analyzing dynamics of tunnels in macromolecules”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 13.3 (2015), pp. 505–517.

Voronoi diagram for collision avoidance

- Change of positions between various formations[7]
- Drone art [8]

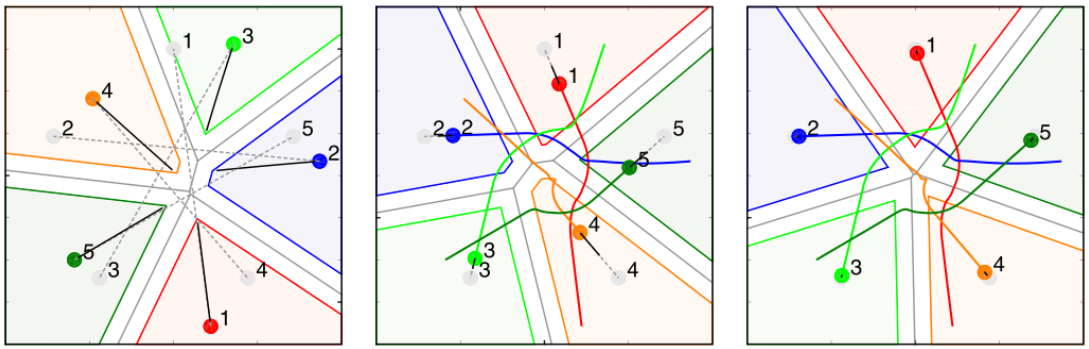


www.youtube.com/watch?v=YH1BD7kKqKw

[7] D. Zhou et al. "Fast, On-line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1047–1054. DOI:

Voronoi diagram for collision avoidance

- Change of positions between various formations[7]
- Drone art [8]

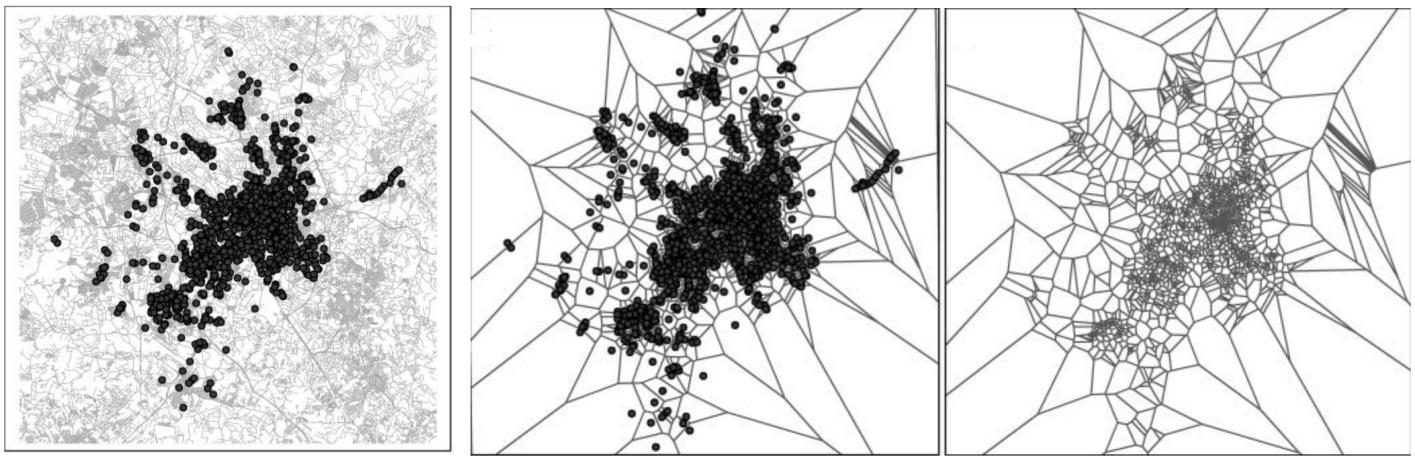


[7] D. Zhou et al. "Fast, On-line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1047–1054. DOI: 10.1109/LRA.2017.2656241.

[8] https://en.wikipedia.org/wiki/Drone_art

Voronoi diagram for spatial analysis

- One of first analysis was Cholera epidemic in London[a]
- Often used in criminology



[a] S. N. d. Melo et al. "Voronoi diagrams and spatial analysis of crime". In: *The Professional Geographer* 69.4 (2017), pp. 579–590.

Voronoi diagram in computer graphics

- Used in many low-level routines (e.g., point location)
- Modeling fractures
  - Object is filled with some random points
  - VD is computed to provide set of convex cells
  - Interaction between cells can be modeled e.g. using rigid body dynamics

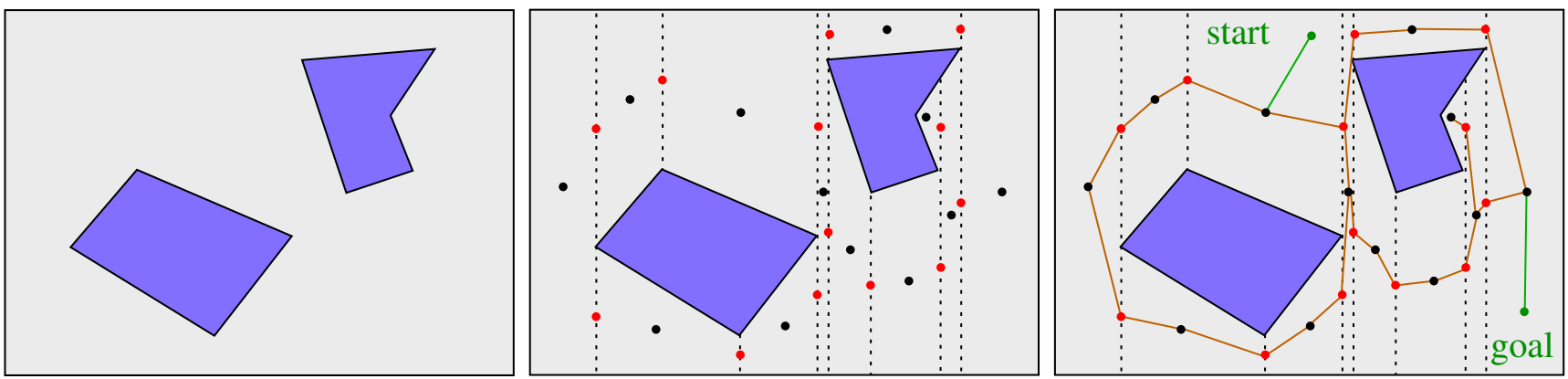
www.youtube.com/watch?v=FIPu9_OGFgc

Decomposition-based methods

- The free space is partitioned into a finite set of cell
  - Determination of cell containing a point should be trivial
  - Computing paths inside the cells should be trivial
- The relations between the cells is described by a graph

Vertical cell decomposition

- Make vertical line from each vertex, stop at obstacles
- Determine centroids of the cells, centers of each segments
- Graph connects the neighbor centroids through the centers
- Connect start/goal to centroid of their cells
- Can be built in $O(n \log n)$ time

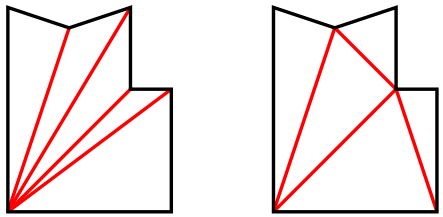


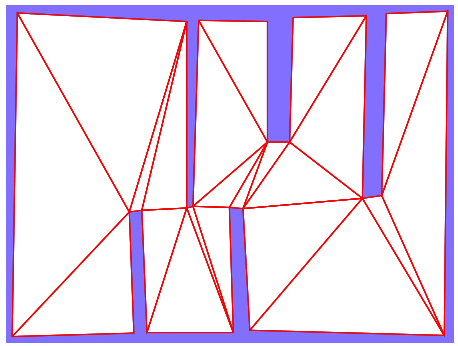
Decomposition via triangulation I

- Variant of decomposition-based methods
- $\mathcal{C}_{\text{free}}$ is triangulated
- Can be computed in $O(n \log \log n)$ time
- Polygons can be triangulated in many ways
- $\mathcal{C}_{\text{free}}$ is represented by graph $G = (V, E)$
  - V are centroids of the triangles
  - $E = (e_{i,j})$ if Δ_i is neighbor of Δ_j

  Or

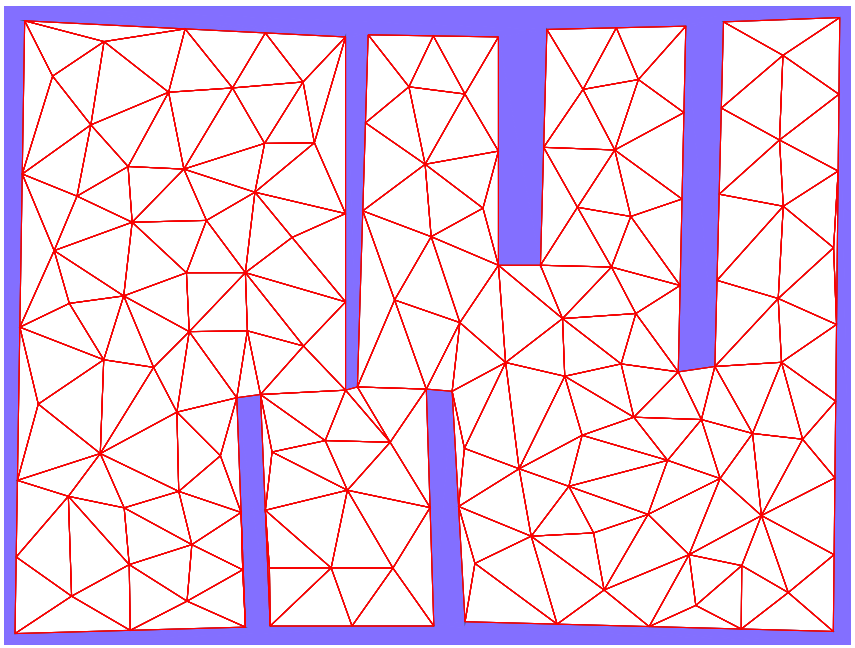
  - V are vertices of the triangulation
  - E are edges of the triangulation
- Planning: start/goal are connected to graph, then graph search



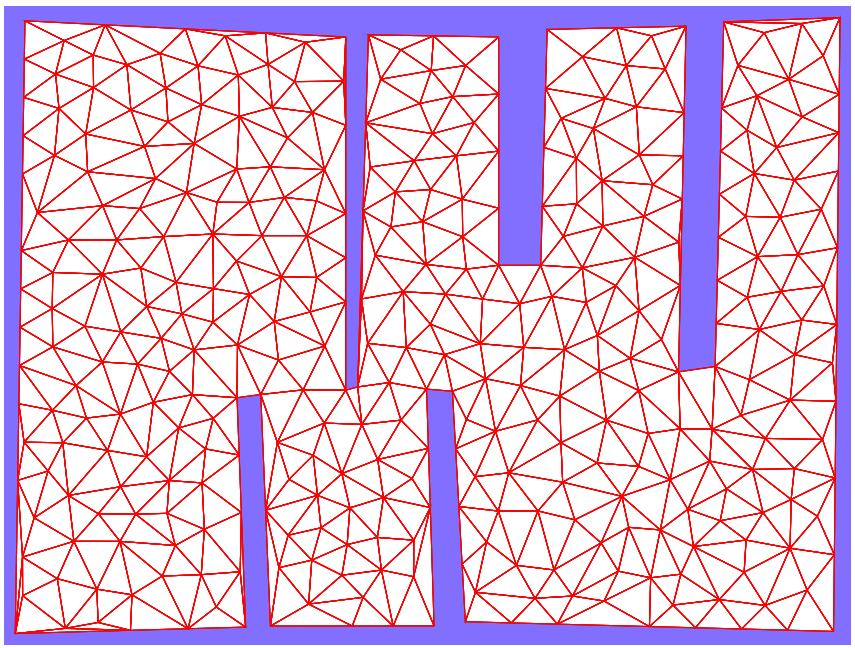


Decomposition via triangulation II

- Finer triangulation via Constrained Delaunay Triangulation (CDT)
  - if a triangle does not meet a criteria, it is further triangulated
  - criteria: triangle area or the largest angle



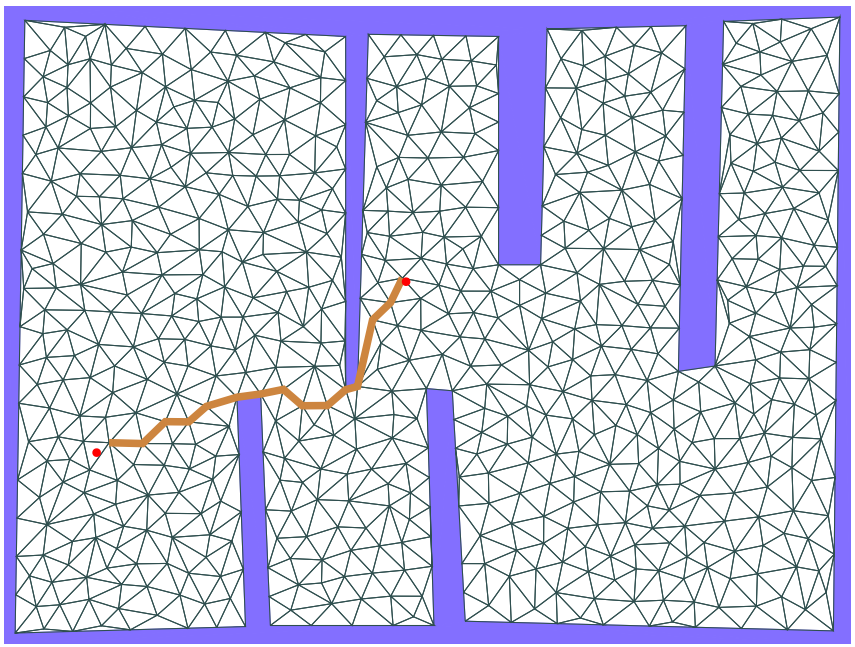
CDT



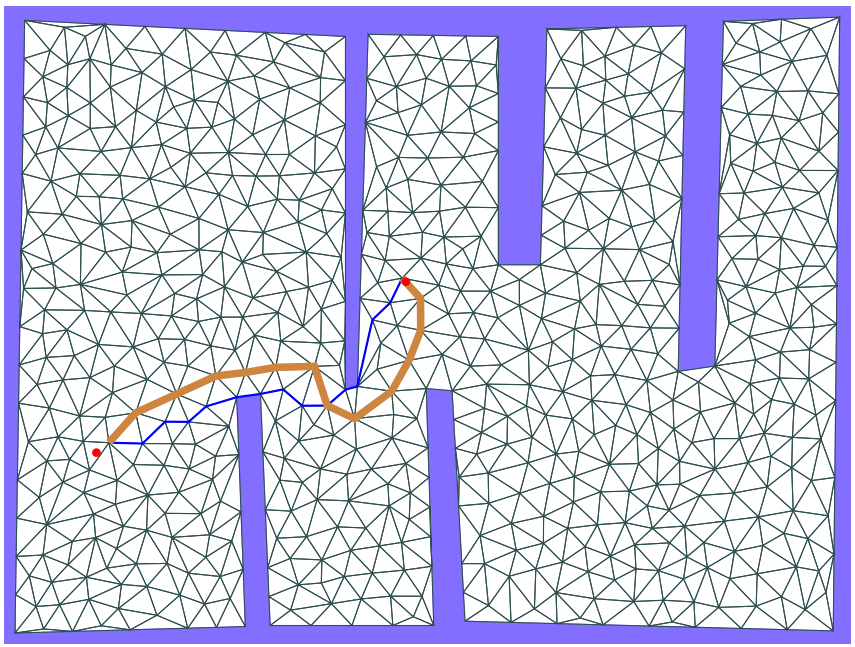
Finer CDT (area of Δ)

Decomposition via triangulation II

- Finer triangulation via Constrained Delaunay Triangulation (CDT)
  - if a triangle does not meet a criteria, it is further triangulated
  - criteria: triangle area or the largest angle



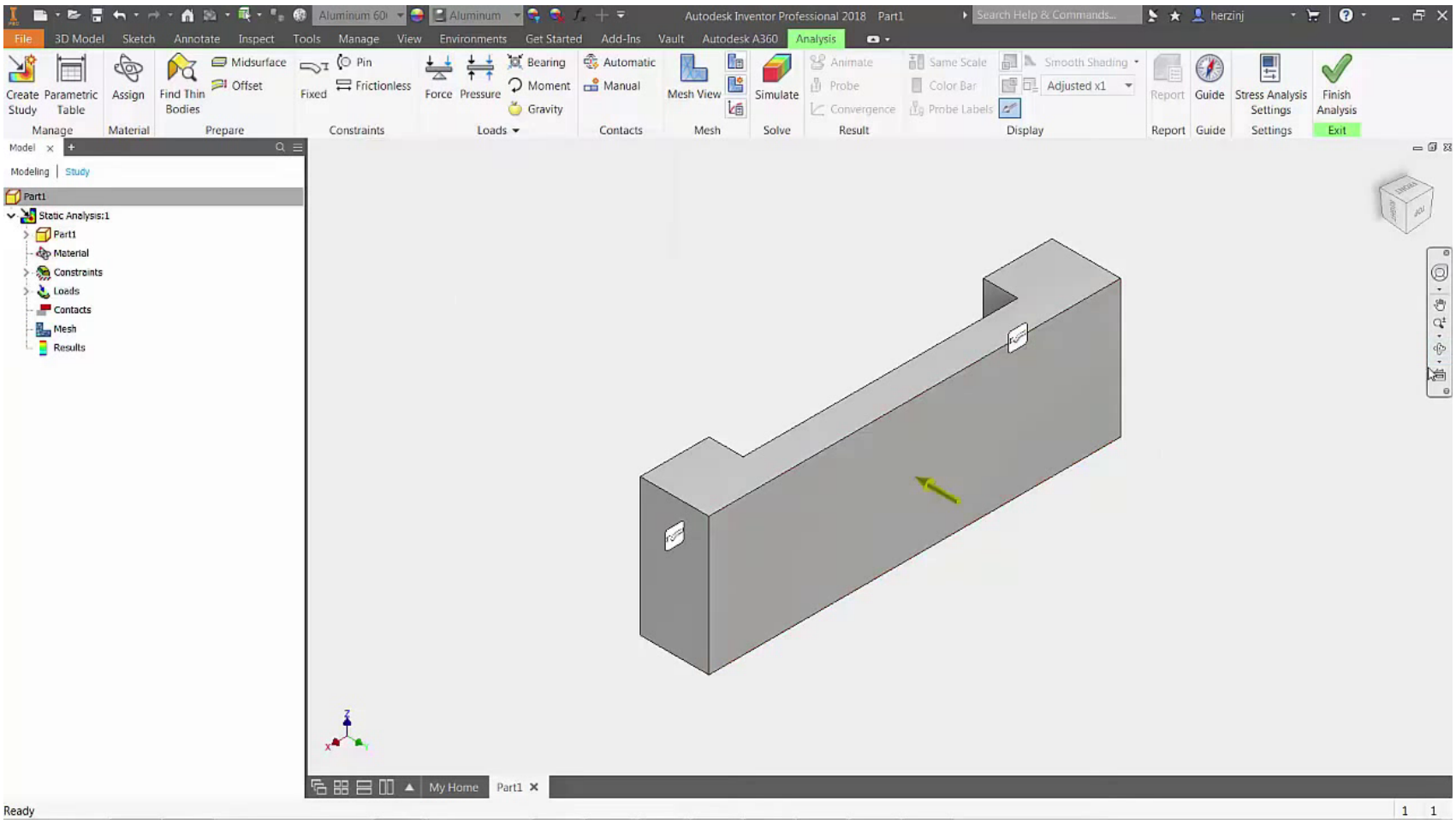
Path on edges



Modification: ignore segments connecting obstacles

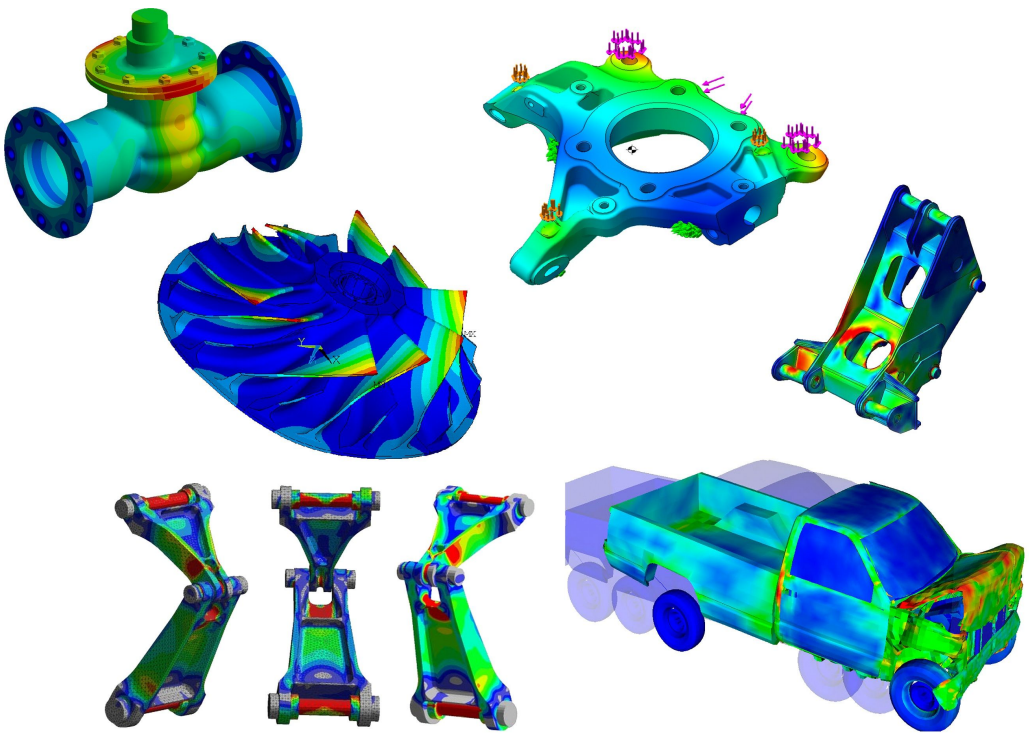
CDT in civil engineering

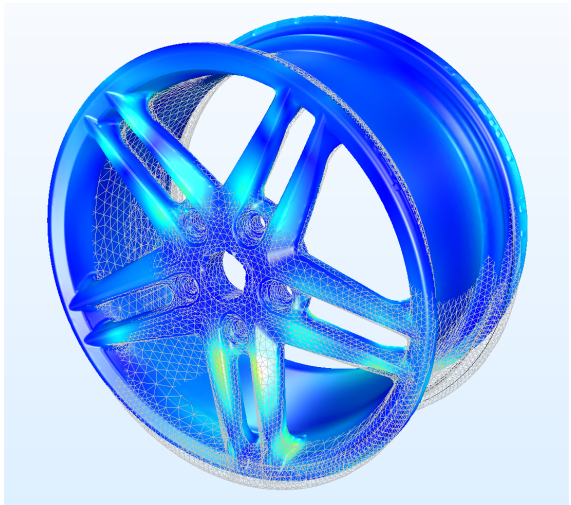
- Structural analysis: modeling behavior of a structure under load, wind, pressure, . . .
- Finite element method

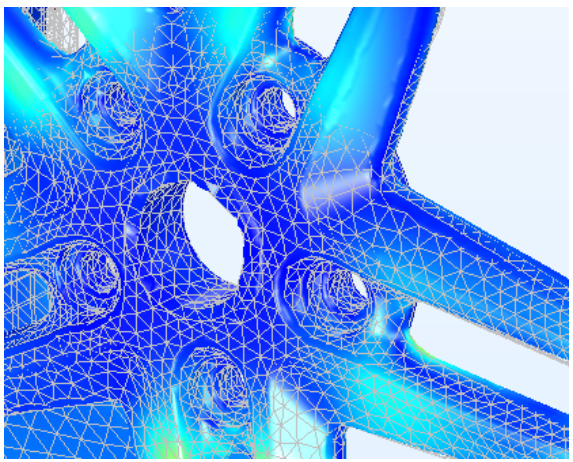


CDT in civil engineering

- Structural analysis: modeling behavior of a structure under load, wind, pressure, . . .
- Finite element method

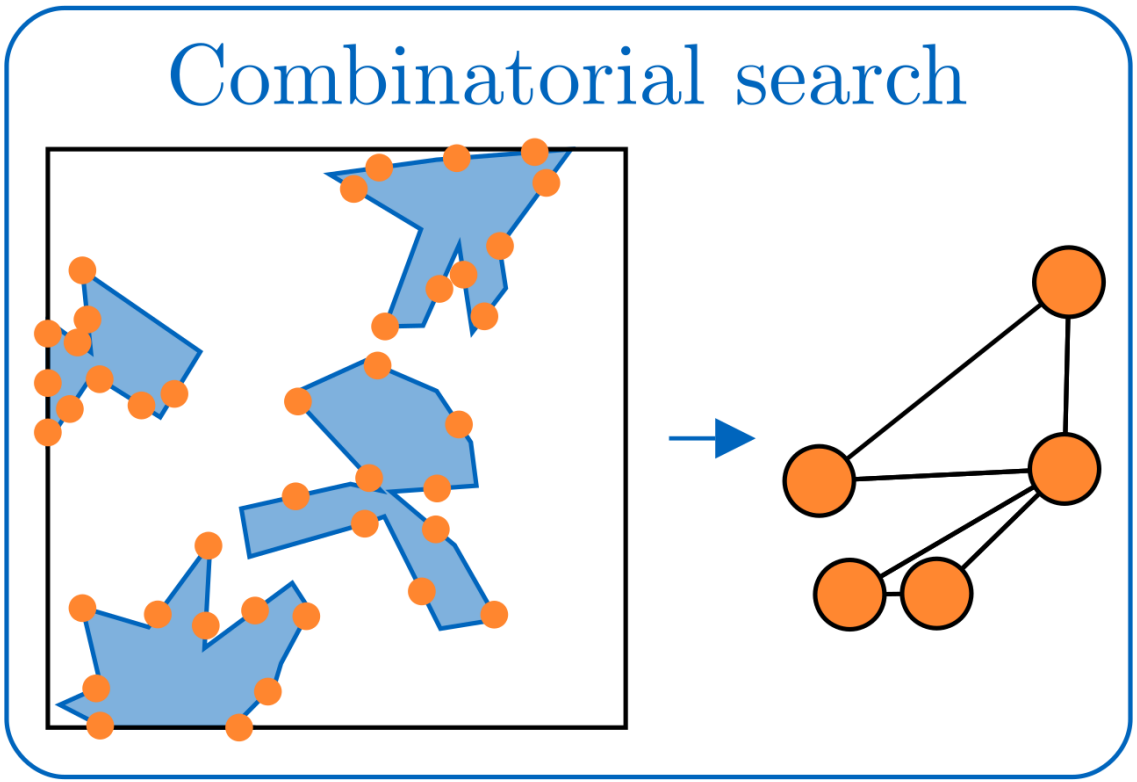






Combinatorial methods: summary

- Visibility graphs, Voronoi diagrams, Decomposition-based planners
- "Convert" $\mathcal{C}$-space to a graph
- Path in the graph is search by graph-search
- Path in the graph defines motion in $\mathcal{W}$



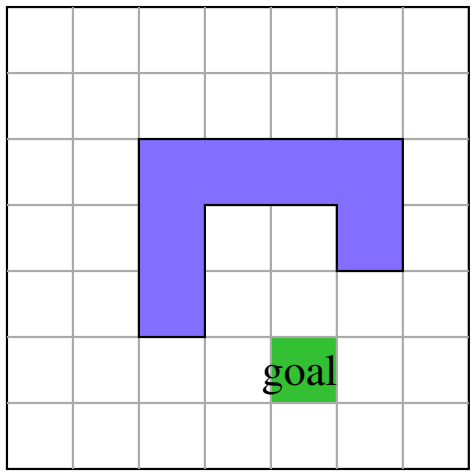
Navigation functions

- Let's assume a forward motion model

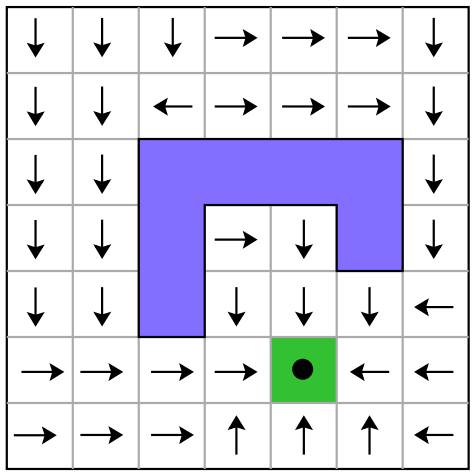
$$\dot{q} = f(q, u)$$

  where $q \in \mathcal{C}$ and $u \in \mathcal{U}$ ($\mathcal{U}$ are allowed actions)
- The navigation function $F(q)$ $\mathcal{C} \rightarrow \mathcal{U}$ tells which action to take at q to reach the goal

Example: robot moving on grid, actions $\mathcal{U} = \{\rightarrow, \leftarrow, \uparrow, \downarrow, \bullet\}$



Discrete planning problem



Navigation function

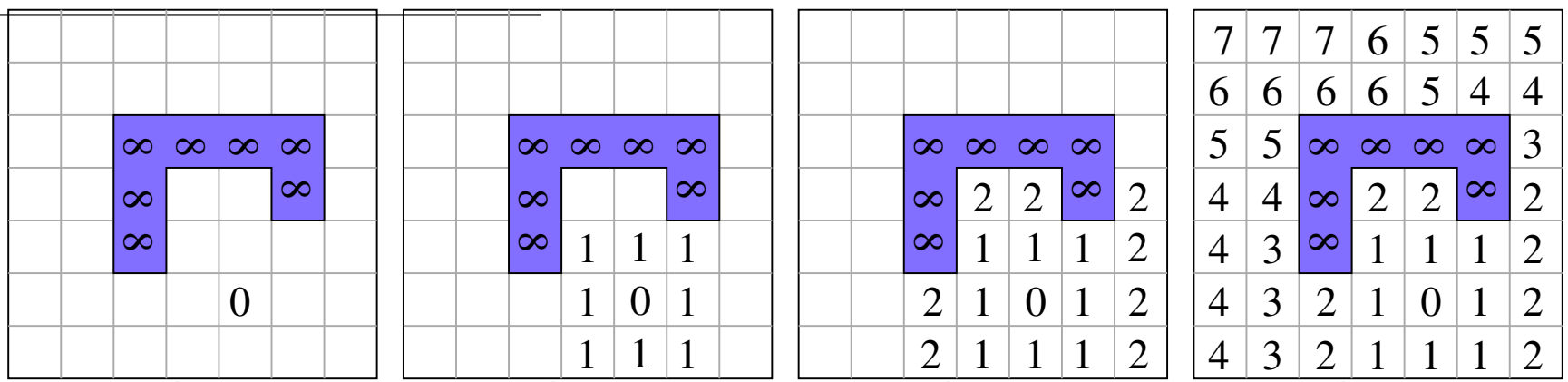
- In discrete space, navigation f. is a by-product of graph-search methods

Wavefront planner

- Explores X in "waves" starting from goal until all states are explored

```
1  open = {goal}
2  i = 0
3  while open ≠ ∅ do
4      wave = ∅        // new wave
5      foreach x ∈ open do
6          value(x) = i
7          foreach y ∈ N(x) do
8              if y is not explored
                 then
9                add y to wave
10     i = i + 1
11     open = wave
```

- $N(x)$ are neighbors of x
- 4-/8-point connectivity
- The increase of the wave value i should reflect the distance between x and its neighbors
- Path is retrieved by gradient-descent from start
- $O(n)$ time for n reachable states

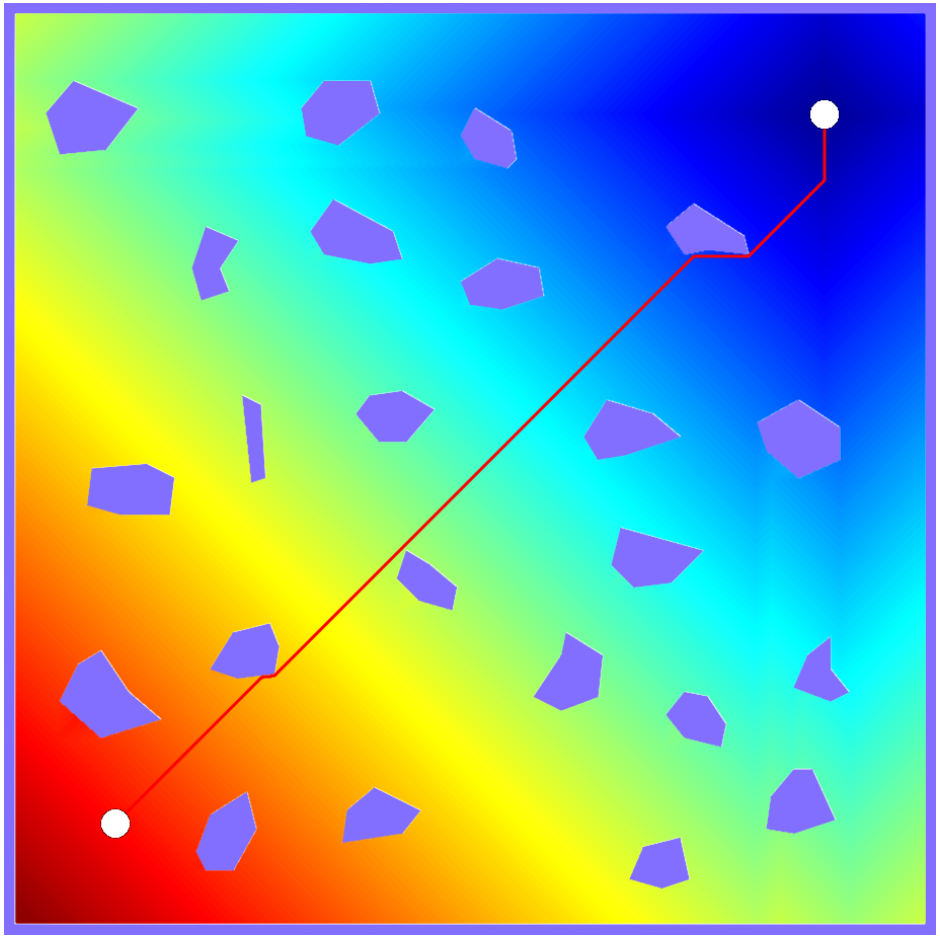


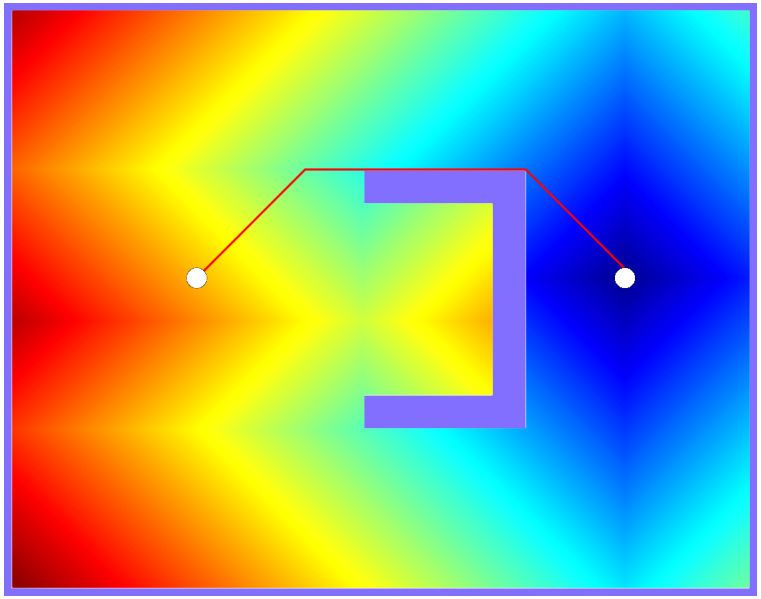
Wavefront planner

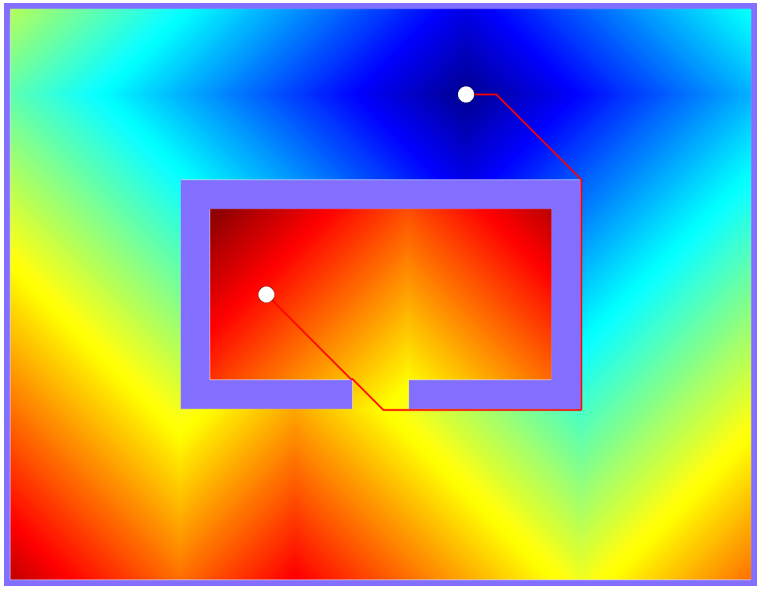
- Similar to BFS, it just propagates the cost of reaching the goal
- Suitable for 2D occupancy grids[9]

[9] I. Ibrahim et al. "Exact Wavefront Propagation for Globally Optimal One-to-All Path Planning on 2D Cartesian Grids". In: *IEEE Robotics and Automation Letters* 9.11 (Nov. 2024), pp. 9431–9437. ISSN: 2377-3774. DOI: 10.1109/lra.2024.3460409.

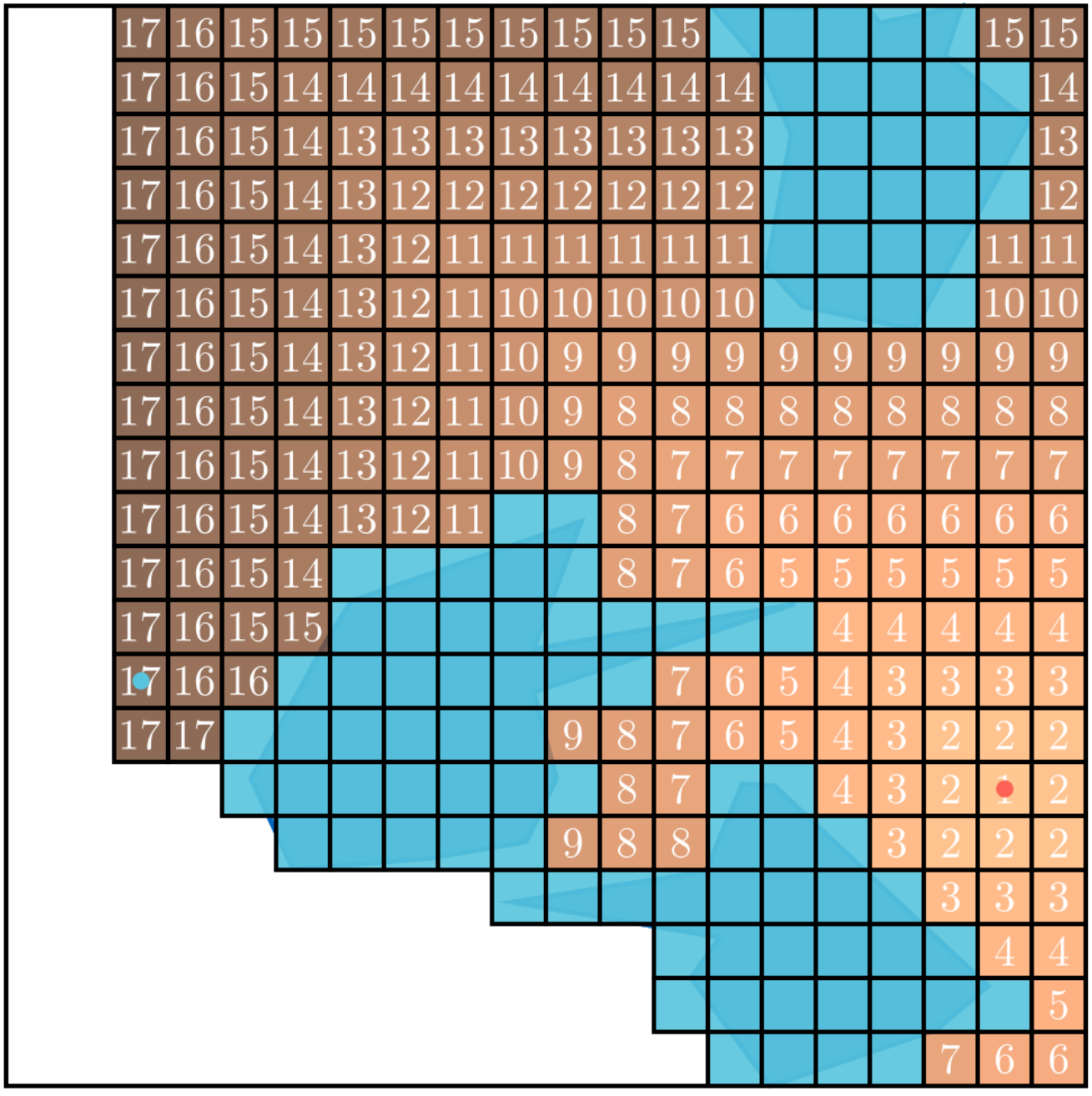
Wavefront planner





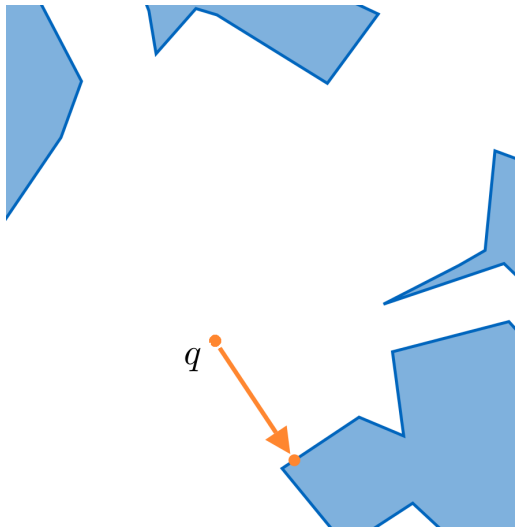


Wavefront planner



Potential field: principle

Idea: the robot is repelled by obstacles and attracted by the goal[10]



- Attractive potential U_{att} (with weight K_{att}):

$$U_{att}(q) = \frac{1}{2} K_{att} dist(q, q_{\text{goal}})^2$$

- Repulsive potential U_{rep} (with weight K_{rep}):

$$U_{rep}(q) = \begin{cases} \frac{1}{2} K_{rep}(1/d_{\mathcal{O}} - 1/\varrho)^2 & \text{if } d_{\mathcal{O}} \leq \varrho \\ 0 & \text{otherwise} \end{cases}$$

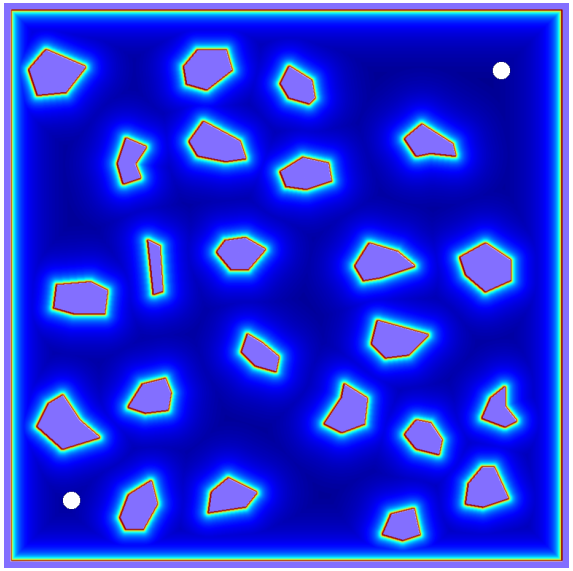
- $d_{\mathcal{O}}$ is the distance to the nearest obstacle, ϱ is radius of influence
- Combined potential

$$U(q) = U_{att}(q) + U_{rep}(q)$$

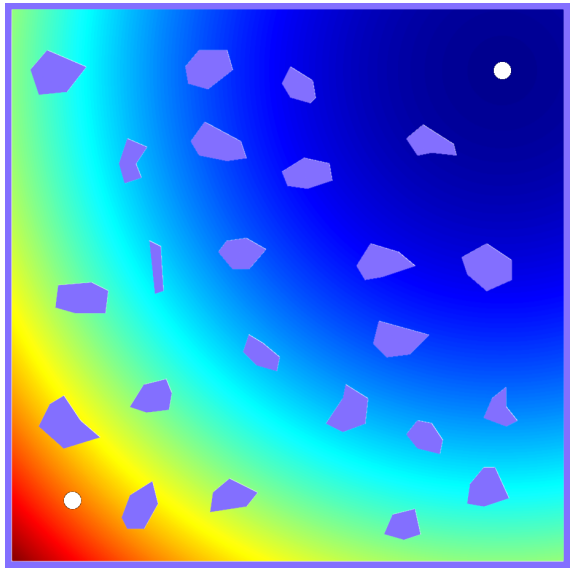
- Gradient descent to goal: follow the negative gradient $-\nabla U(q)$

[10] Y. Hwang and N. Ahuja. "A potential field approach to path planning". In: *IEEE Transactions on Robotics and Automation* 8.1 (1992), pp. 23–32. DOI: 10.1109/70.127236.

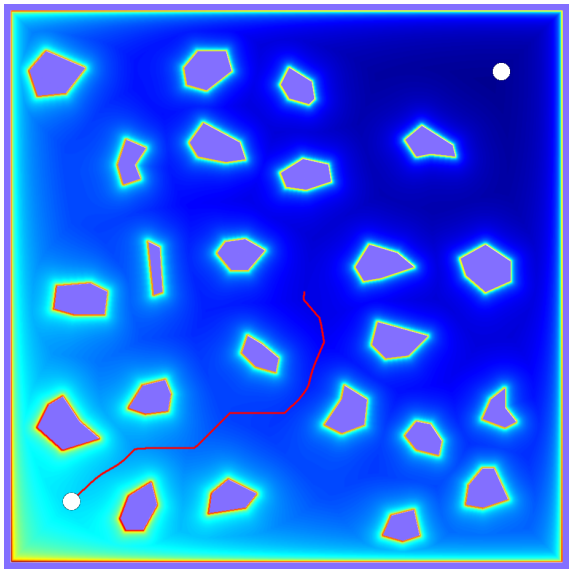
Potential field: parameters



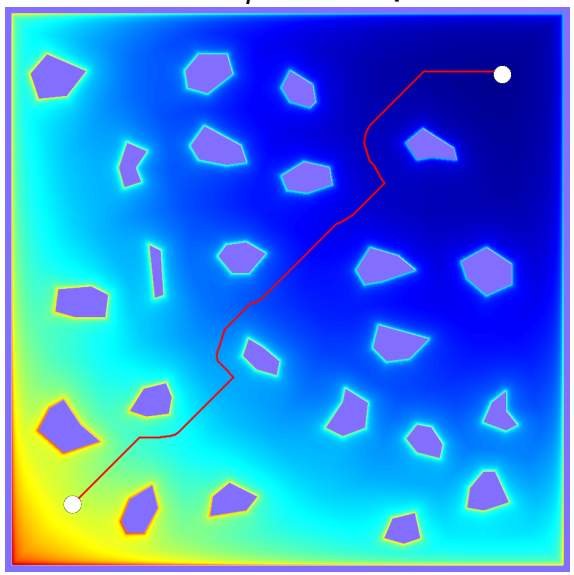
$K_{att} = 0$, no attraction



$K_{att} \gg K_{rep}$, no repulsion



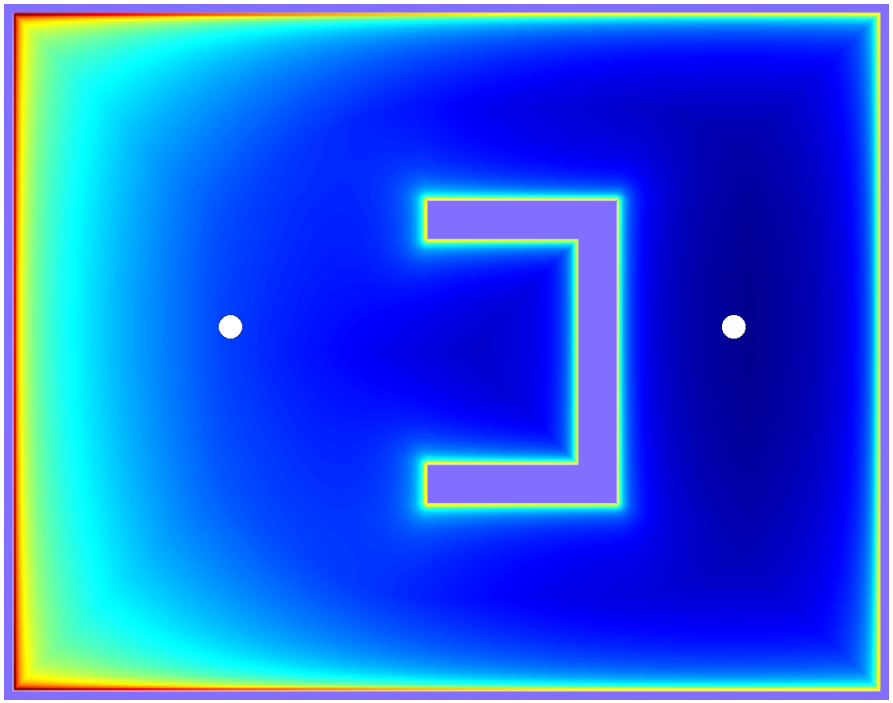
$K_{att} \sim K_{rep}$



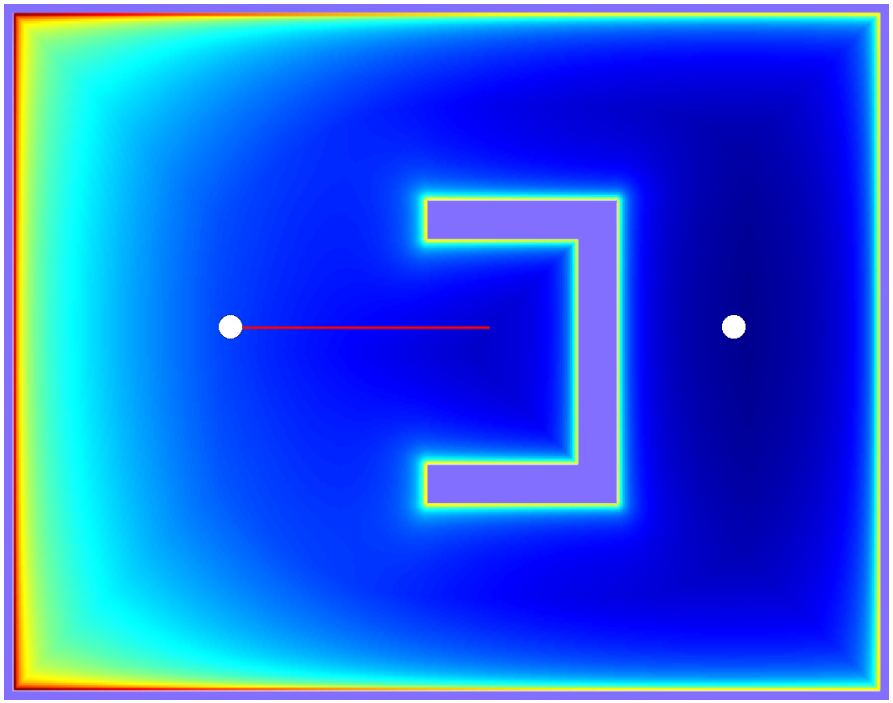
optimal settings

Potential field: local minima problem

- Potential field may have more local minima/maxima
- Gradient-descent stucks there



potential field

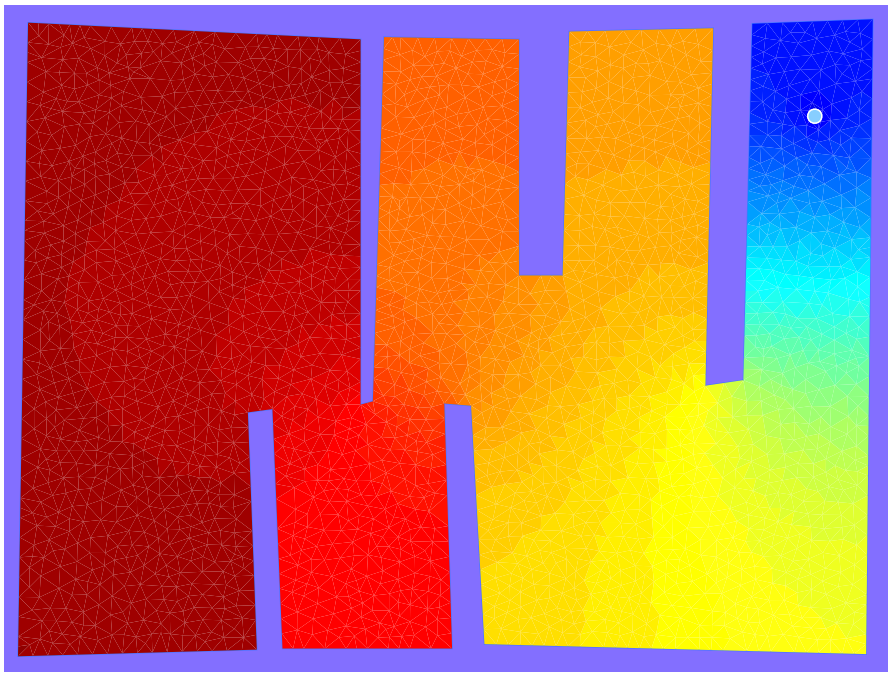


gradient-descent to minimum

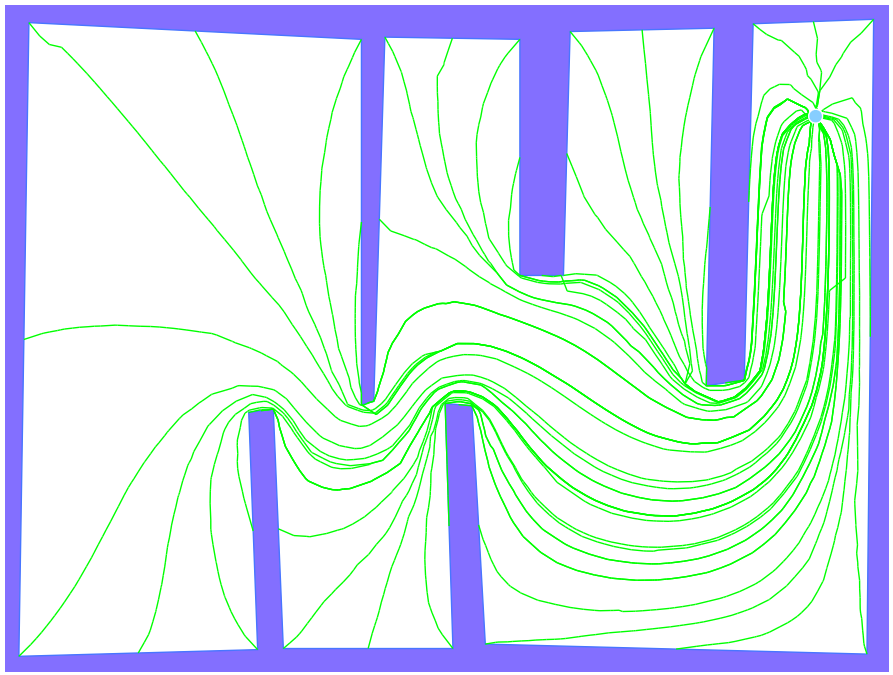
- Escape using random walks
- Use a better potential function without multiple local minima — harmonic field

Harmonic field

- Harmonic field is an ideal potential function: only one extreme[11]
- Many available implementations [12]



Harmonic field



Paths from different configurations

Images by[13]

[11] S. Wang et al. "Customize Harmonic Potential Fields via Hybrid Optimization Over Homotopic Paths". In: *IEEE Robotics and Automation Letters* 10.8 (2025), pp. 8594–8601. DOI: 10.1109/LRA.2025.3587840.

[12] https://github.com/mhagenow01/2Dplanning

[13] J. Mačák. *Multi-robotic cooperative inspection*. Master thesis. 2009.

Potential field: summary

- Usually computed using grid or a triangulation of the $\mathcal{W}$
- Suitable for 2D/3D $\mathcal{C}$-space
  - memory requirements (in case of grid-based computation)
  - requires to compute distance d to the nearest obstacle in $\mathcal{C}$!
- Parameters K_{att}, K_{rep} and ϱ need to be tuned
- Problem with local minima $\rightarrow$ harmonic fields

Completeness and optimality

Visibility graph

- Complete and optimal

Voronoi diagram, decomposition-based method

- Complete, non-optimal

Navigation function

- Complete
- Optimal for Wavefront/Dijkstra/-based navigation functions

Potential field

- Complete only if harmonic field is used (one local minima!)

Consider the limits of these methods!

- Point/Disc robots, low-dimensional $\mathcal{C}$-space

Optimality of planning methods

Do we always need optimal solution?

- No! in many cases, non-optimal solution is fine
  - e.g. for assembly/disassembly studies, computational biology
  - generally: if the **existence of a solution** is enough for subsequent decisions
- in industry:
  - scenarios, where robot waits due to mandatory technological breaks
  - e.g., in robotic welding and painting

Optimality of planning methods

When to prefer optimal one?

- Repetitive executing of the same plan
- Benchmarking of algorithms

It is necessary to carefully design the criteria!

Shortest path vs. fastest path vs. path for good spraying