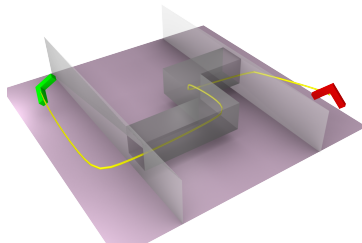
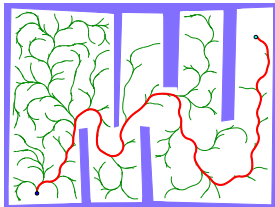


Motion planning: combinatorial path planning

Vojtěch Vonásek

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



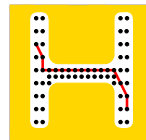
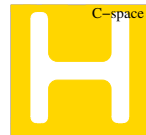
Continuous space

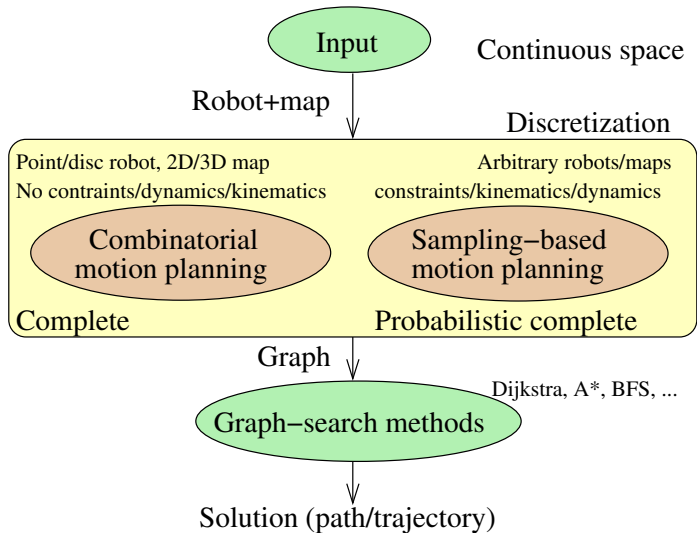


Discretization

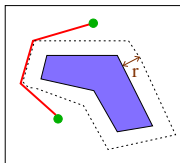
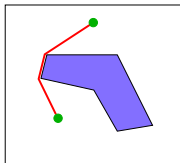


Search





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



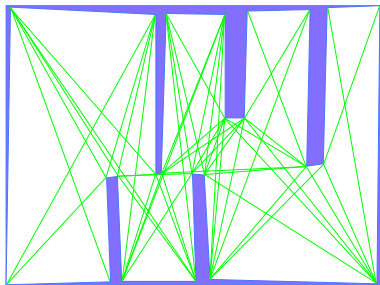
Point robot in 2D or 3D W

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

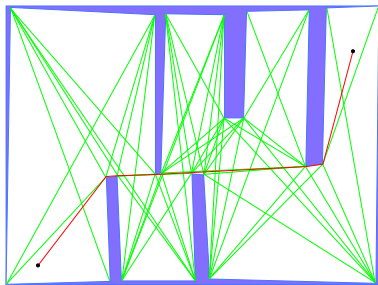
Disc/sphere robot in 2D or 3D W

- The obstacles are “enlarged” by radius of the robot (Minkowski sum)
- Then, representation of W is also representation of \mathcal{C}

- 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, E are edges between visible points
- Start/goal are connected in same manner to visible vertices



Visibility graph



After connecting start/goal + path

- No clearance
- Suitable only for 2D

- Straightforward, naïve implementation $O(n^3)$

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
```

- n^2 pairs of vertices
 - Complexity of checking one intersection is $O(n)$
- Total complexity $O(n^3)$

Fast methods

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

• Lee, Der-Tsai, Proximity and reachability in the plane, 1978

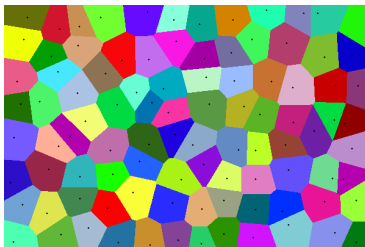
• D. Coleman, Lee's $O(n^2 \log n)$ Visibility Graph Algorithm Implementation and Analysis, 2012.

• M. H. Overmars, E. Welzl, New methods for Computing Visibility Graphs, Proc. of 4th Annual Symposium on Comp. Geometry, 1998

• S. Ghosh and D. M. Mount, An output-sensitive algorithm for computing visibility graphs, SIAM Journal on Computing, 1991

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

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



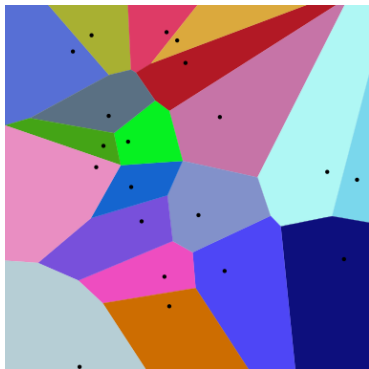
- Cells are convex
- Used in point location (1-nn search), closest-pair search, spatial analysis
- Construction using Fortune’s method in $O(n \log n)$

• S. Fortune. A sweepline algorithm for Voronoi diagrams. Proc. of the 2nd annual composition on Computational geometry. pages 313-322. 1986.

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

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

- Note, that other metrics can be considered



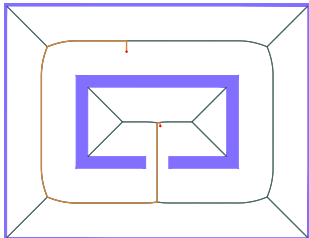
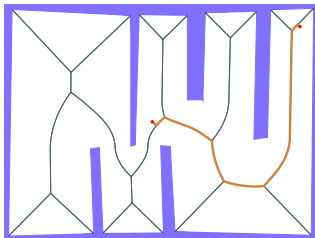
Voronoi diagrams are everywhere



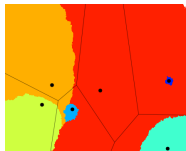
- (Basic) Voronoi diagram: computed on points
- Generalized Voronoi Diagram: computed on e.g., points + weights, segments, spheres, ...

Segment Voronoi Diagram (SVD)

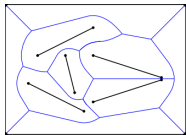
- computed on line-segments describing obstacles
- requires polygonal map or line/segment map
- ✓ Maximal clearance
 - largest distance between a path and the nearest obstacle
- Is it optimal? Is it complete?



Classic VD



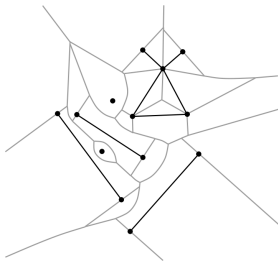
Weighted VD



Segment VD

Algorithms for computing Segment Voronoi diagram of n segments

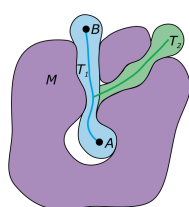
- Lee & Drysdale: $O(n \log^2 n)$, no intersections
- Karavelas: $O((n + m) \log^2 n)$, m intersections between segments



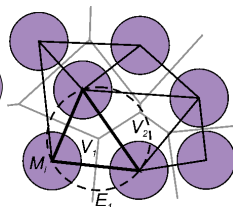
Karavelas 2004

- Karavelas, M. I. "A robust and efficient implementation for the segment Voronoi diagram." International symposium on Voronoi diagrams in science and engineering. 2004
- Lee, D. T, R. L. Drysdale, III. "Generalization of Voronoi diagrams in the plane." SIAM Journal on Computing 10.1 (1981): 73-87.

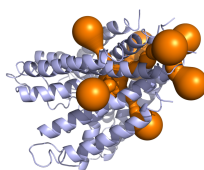
- Proteins are modeled using hard-sphere model
- Weighted Voronoi diagram of the spheres (weight is the atom radii — Van der Waals radii)
- Path in the Voronoi diagram reveals “void space” and “tunnels”
- Tunnel properties (e.g. bottleneck) estimate possibility of interaction between protein and a ligand



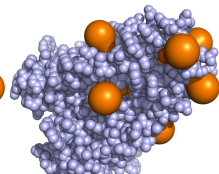
Tunnels*




Voronoi diagram *

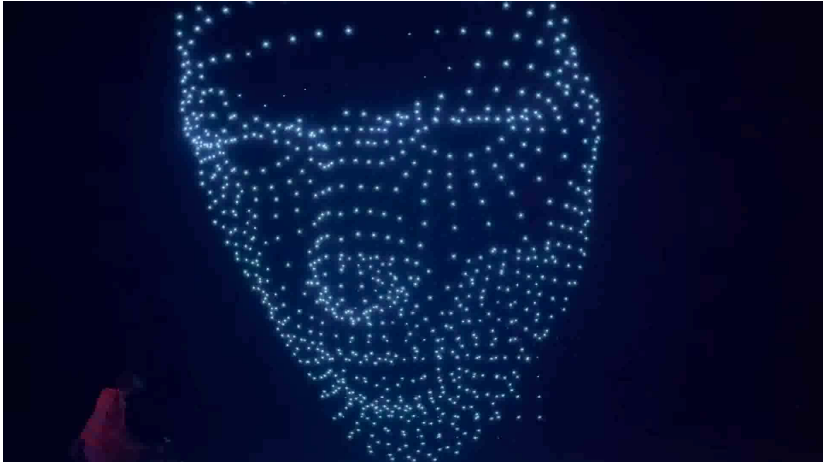


Tunnels on 1BL8

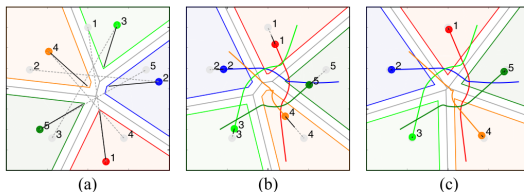


*  A. Pavelka, E. Sebestova, B. Kozlikova, J. Brezovsky, J. Sochor, J. Damborsky, CAVER: Algorithms for Analyzing Dynamics of Tunnels in Macromolecules, IEEE/ACM Trans. on comput. biology and bioinformatics, 13(3), 2016.

- Change of positions between various formations (e.g. in drone art)



- Change of positions between various formations (e.g. in drone art)



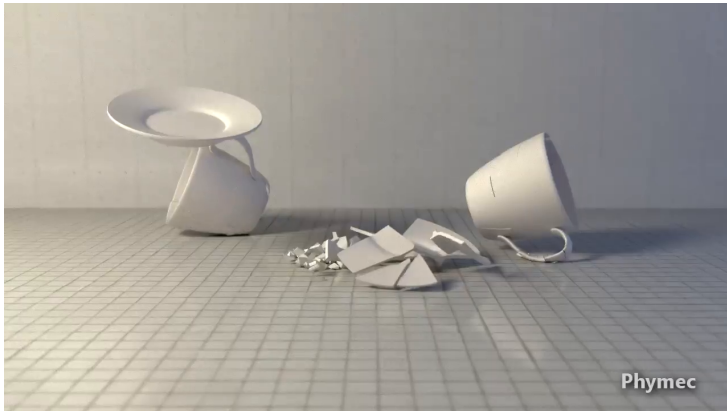
- Zhou, Dingjiang, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, On-Line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells. IEEE Robotics and Automation Letters, (2), 2017.

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



• Melo, S. N. D., Frank, R., Brantingham, P. (2017). Voronoi diagrams and spatial analysis of crime. *The Professional Geographer*, 69(4), 579-590.

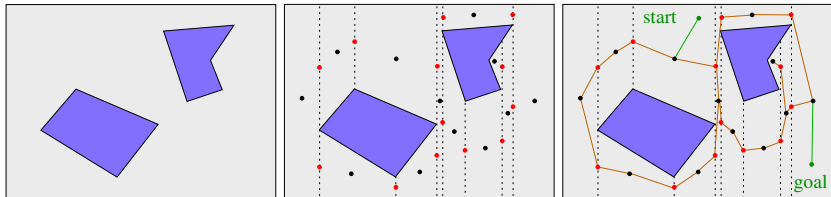
- Used in many low-level routines (e.g., point location)
- Modeling fractures
 - Object is filled with some random points
 - VD is computed to provide set of convex cells
 - Interaction between cells can be modeled e.g. using rigid body dynamics



- 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

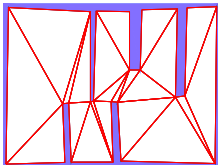
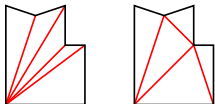


- Variant of decomposition-based methods
- $\mathcal{C}_{\text{free}}$ is triangulated
- Can be computed in $O(n \log \log n)$ time
- Polygons can be triangulated in many ways
- $\mathcal{C}_{\text{free}}$ is represented by graph $G = (V, E)$

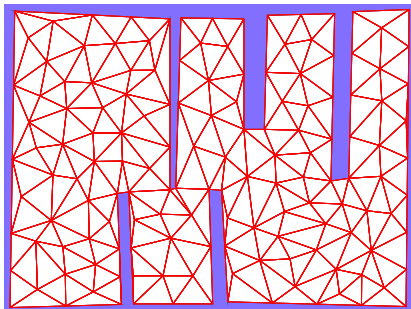
- V are centroids of the triangles
- $E = (e_{i,j})$ if Δ_i is neighbor of Δ_j

Or

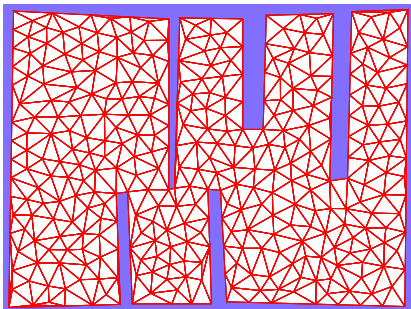
- V are vertices of the triangulation
 - E are edges of the triangulation
- Planning: start/goal are connected to graph, then graph search



- 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

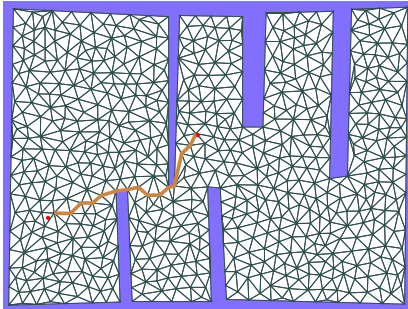


CDT

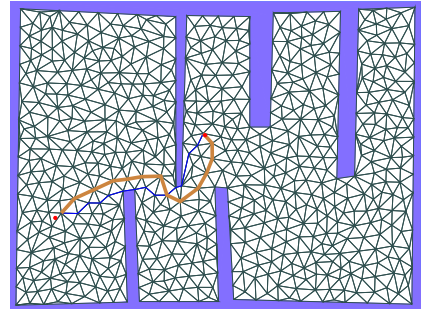


Finer CDT (area of Δ)

- 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

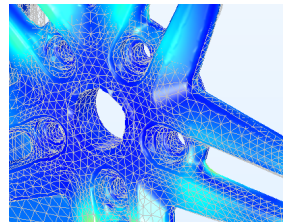
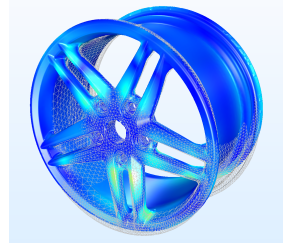
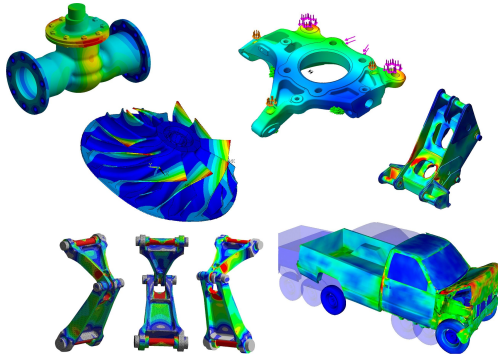


Path on edges



Modification: ignore segments connecting obstacles

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



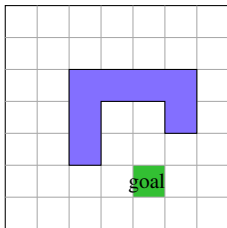
- Let's assume a forward motion model

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

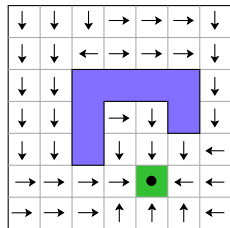
where $q \in \mathcal{C}$ and $u \in \mathcal{U}$; \mathcal{U} is the action space

- The navigation function $F(q)$ 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

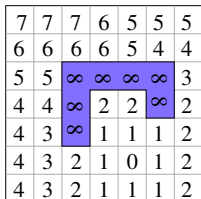
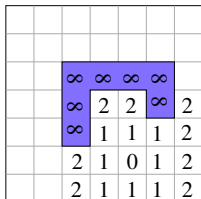
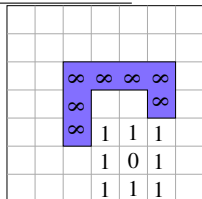
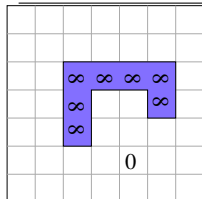
- In discrete space, navigation f is a by-product of graph-search methods

- Simple way to compute navigation function on discrete space X
- 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
9                  then
10                     add y to wave
11     i = i + 1
12     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 descend from start
- $O(n)$ time for n reachable states

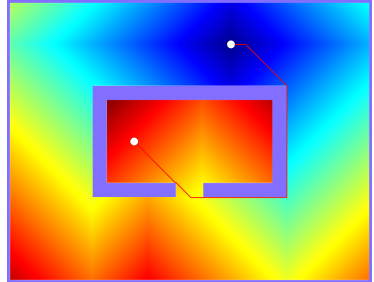
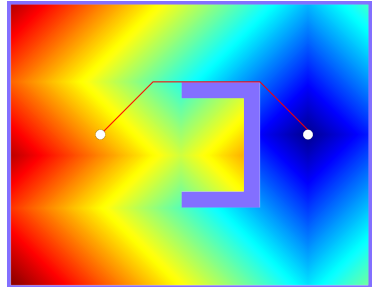
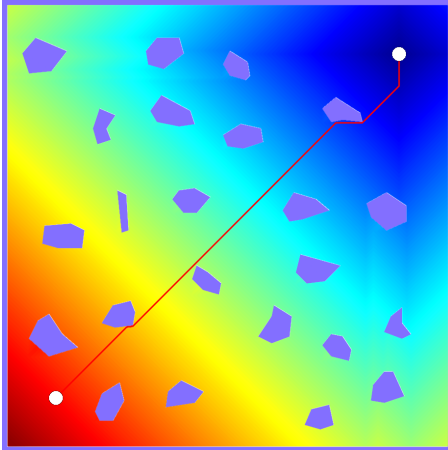


goal state

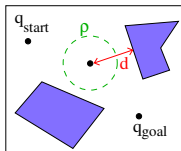
$i = 1$

$i = 2$

$i = 7$



- Potential field U : the robot is repelled by obstacles and attracted by q_{goal}
- Attractive potential U_{att} , repulsive potential U_{rep}
- Weights K_{att} and K_{rep} , d is the distance to the nearest obstacle, ϱ is radius of influence



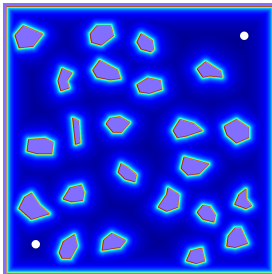
$$U_{\text{att}}(q) = \frac{1}{2} K_{\text{att}} \text{dist}(q, q_{\text{goal}})^2 \quad U_{\text{rep}}(q) = \begin{cases} \frac{1}{2} K_{\text{rep}} (1/d - 1/\varrho)^2 & \text{if } d \leq \varrho \\ 0 & \text{otherwise} \end{cases}$$

- Combined attractive/repulsive potential

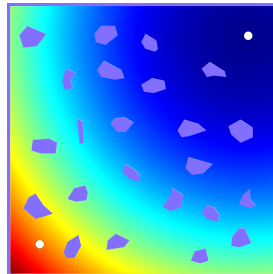
$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

- Goal is reached by following negative gradient $-\nabla U(q)$
- Gradient-descent method

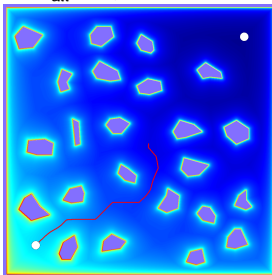
• Y. K. Hwang and N. Ahuja, A potential field approach to path planning, IEEE Transaction on Robotics and Automation, 8(1), 1992.



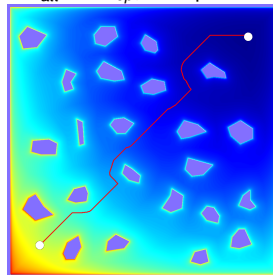
$K_{att} = 0$, no attraction



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

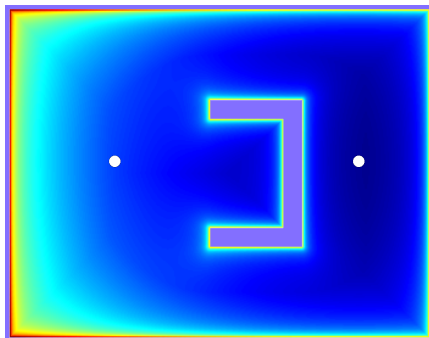


$K_{att} \sim K_{rep}$

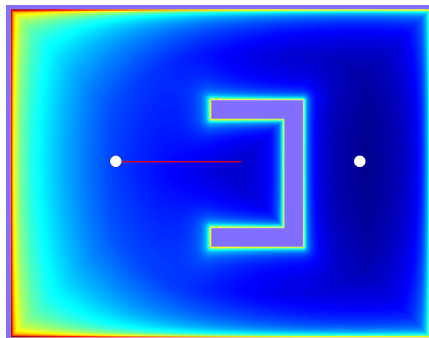


optimal settings

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



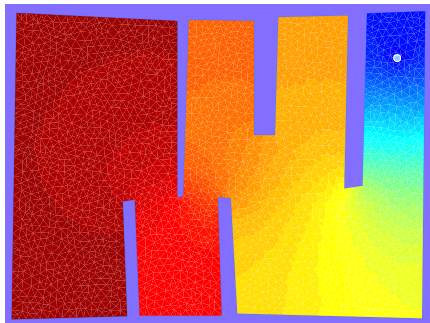
potential field



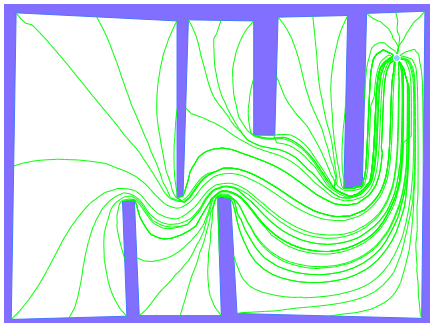
gradient-descent to minimum

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

- Harmonic field is an ideal potential function: only one extreme



Harmonic field



Paths from various q_{init}

- 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

But how to really find the path?

So far we know ...

- Visibility graphs, Voronoi diagrams, Decomposition-based planners
- Navigation functions & Potential fields

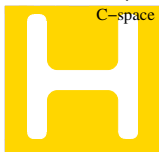
What they do?

- Discretize workspace/ \mathcal{C} -space by “converting” it to a graph structure
- The graph is also called **roadmap**
- The roadmap is a “discrete image” of the continuous \mathcal{C} -space
- The path is then found as path in the graph

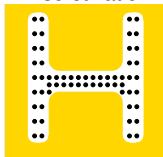
Graph-search

- Breath-first search
- Dijkstra
- A*, D* (and their variants)

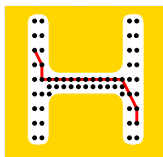
Continuous space



Discretization



Search



- Finds shortest path from $s \in V$ (source) to all nodes
- $\text{dist}(v)$ is the distance traveled from the source to the node s ; $\text{prev}(v)$ denotes the predecessor of node v

```
1  $Q = \emptyset$ 
2 for  $v \in V$  do
3    $\text{prev}[v] = -1$            // predecessor of  $v$ 
4    $\text{dist}[v] = \infty$        // distance to  $v$ 
5  $\text{dist}[s] = 0$ 
6 add all  $v \in V$  to  $Q$ 
7 while  $Q$  is not empty do
8    $u =$  vertex from  $Q$  with min  $\text{dist}[u]$ 
9   remove  $u$  from  $Q$ 
10  foreach neighbor  $v$  of  $u$  do
11     $dv = \text{dist}[u] + d_{u,v}$ 
12    if  $dv < \text{dist}[v]$  then
13       $\text{dist}[v] = dv$ 
14       $\text{prev}[v] = u$ 
```



- Path from $v \rightarrow s$: $v, \text{pred}[v], \text{pred}[\text{pred}[v]], \dots s$
- Dijkstra, E. W. "A note on two problems in connection with graphs." Numerische mathematik 1.1 (1959): 269-271.

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

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



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

- Motion planning: how to move objects and avoid obstacles
- Configuration space \mathcal{C}
- Generally, planning leads to search in continuous \mathcal{C}
- But we (generally) don't have explicit representation of \mathcal{C}
- We have to first create a discrete representation of \mathcal{C}
- and search it by graph-search methods
- Special cases: point robot and 2D/3D worlds
 - Explicit representation of \mathcal{W} is also rep. of \mathcal{C}
 - Geometric planning methods: Visibility graph, Voronoi diagram, decomposition-based
 - Also navigation functions + potential field