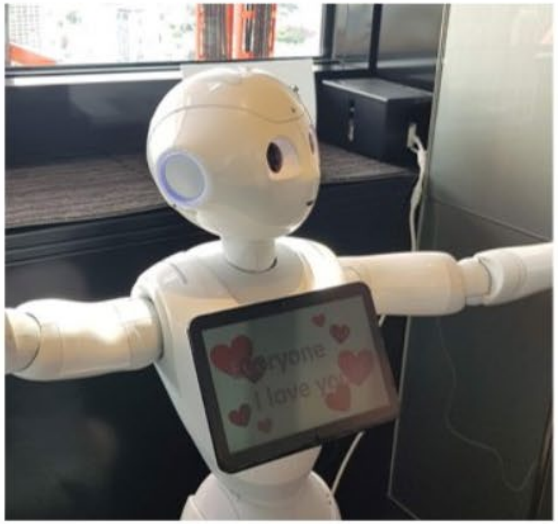
2 – What is Planning?

Dr. Marija Popović

UNIVERSITÄT BONN

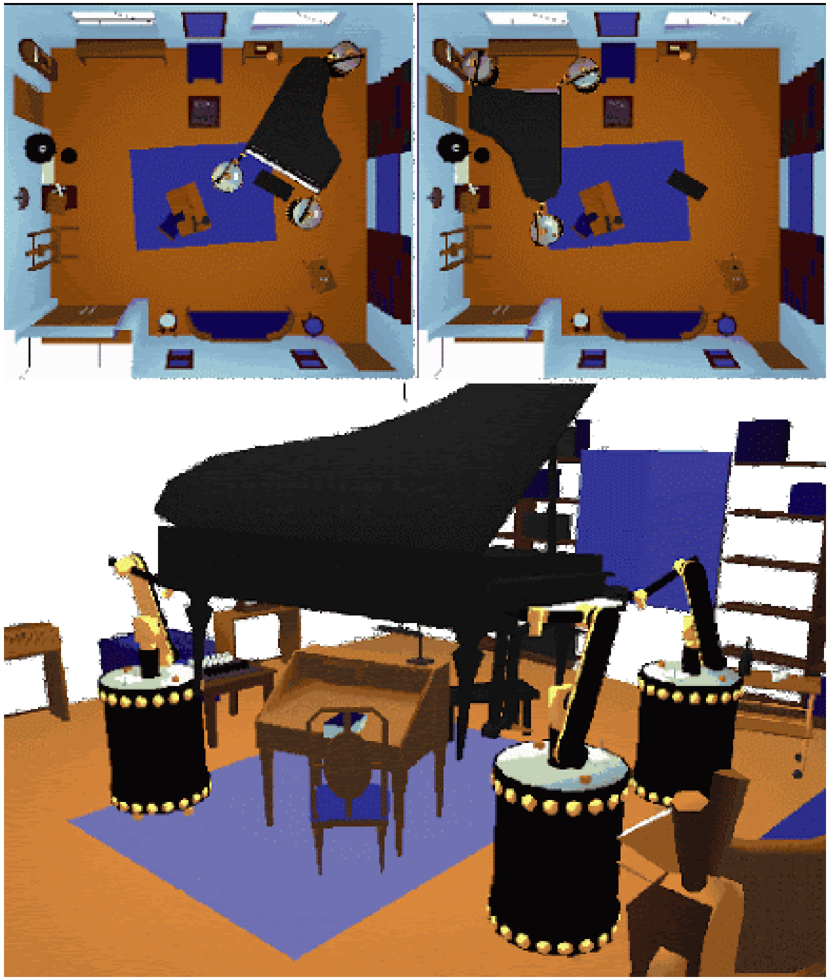
Motivation



http://www.cs.columbia.edu/~allen/F17/NOTES/Lecture_2.pdf
http://asl.stanford.edu/aa274a/pdfs/lecture/lecture_1.pdf
Hitz et al. (2014), "Fully autonomous focused exploration for robotic environmental monitoring," in: IEEE ICRA.
Lehnert et al. (2020), "Performance improvements of a sweet pepper harvesting robot in protected cropping environments," in: JFR. 37(7): pp.1197-1223.

Piano Mover's Problem

- Find the **shortest (optimal) path**



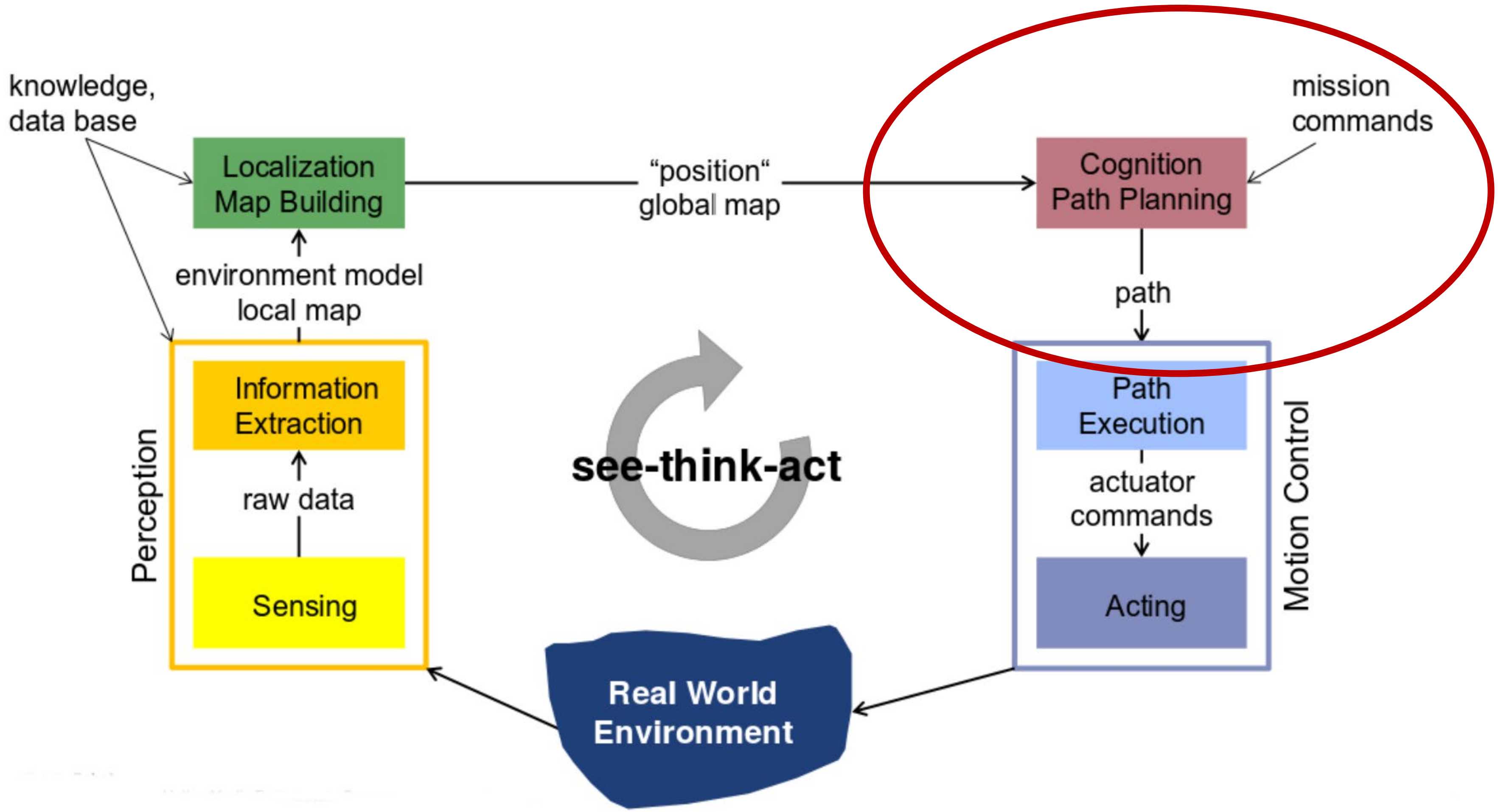
http://lavalle.pl/planning/node10.html

Robotics Challenges

- Find the **shortest (optimal) path**

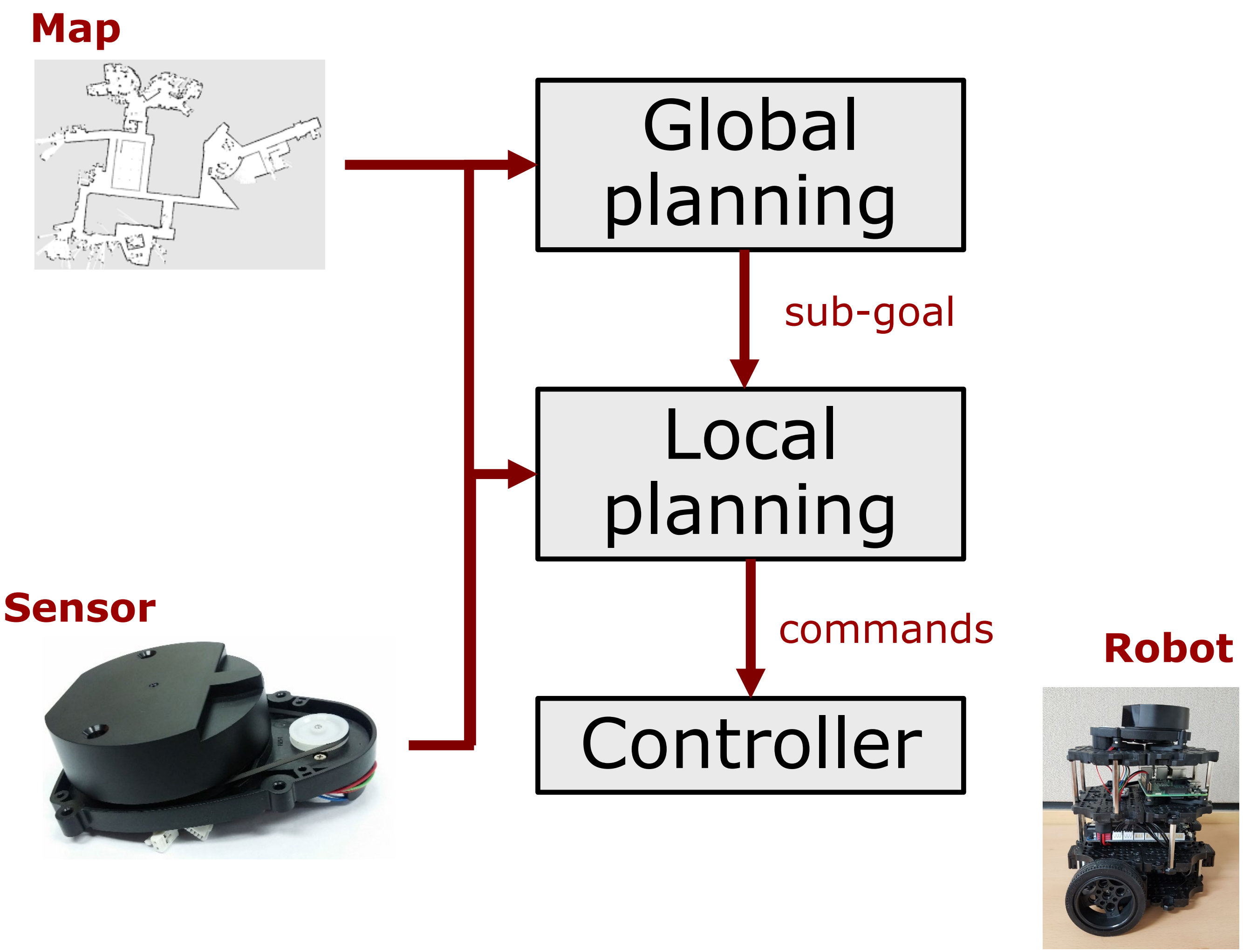
- Physical robot constraints
- Uncertainties: sensing and actuation
- Scalability and computational efficiency
- Dynamic/moving obstacles

An Autonomous System



http://www.cs.columbia.edu/~allen/F17/NOTES/Lecture_2.pdf

Global vs. Local Planning

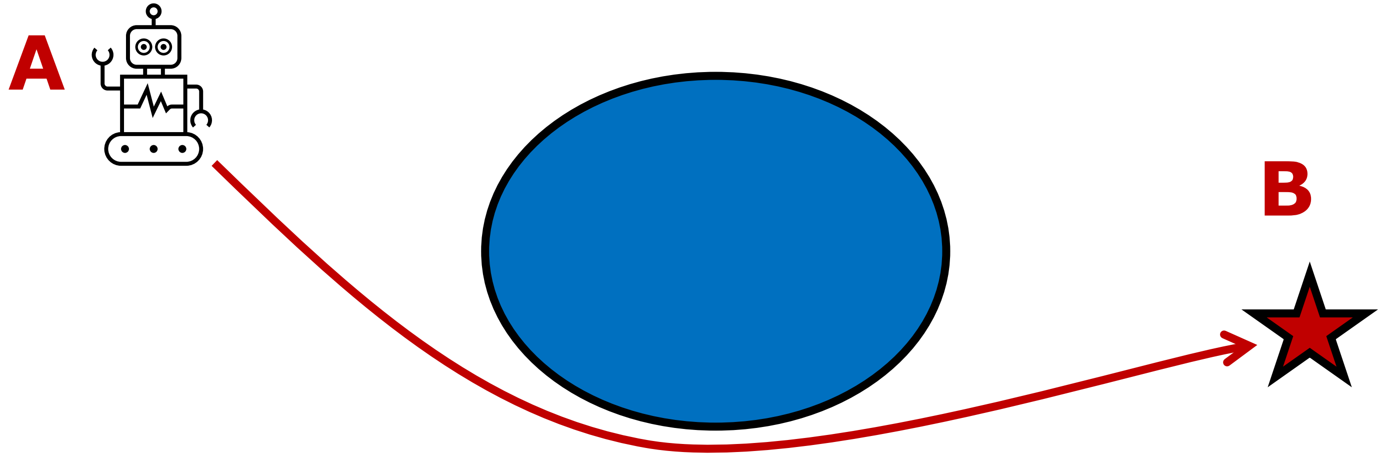


http://emanual.robotis.com

http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/pdf/slam10-gridmaps-4.pdf

What is Planning?

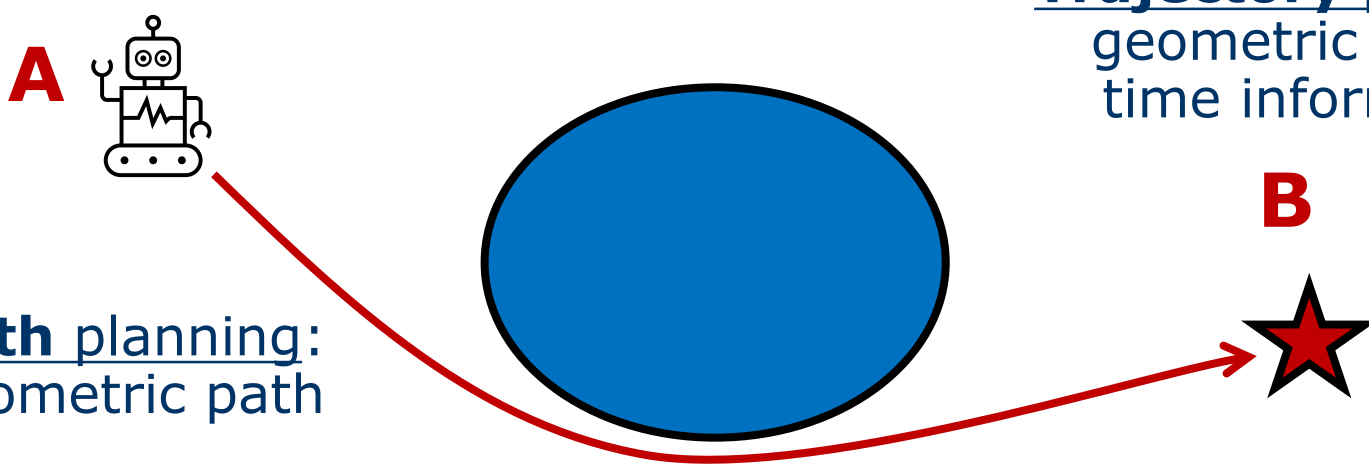
- Find a sequence of valid configurations to move a robot from point A to point B – *how?*



What is Planning?

- Find a sequence of valid configurations to move a robot from point A to point B – *how*?
- Given:
  - Initial configuration (A)
  - Goal configuration (B)
  - Model of the robot
  - Map of the environment

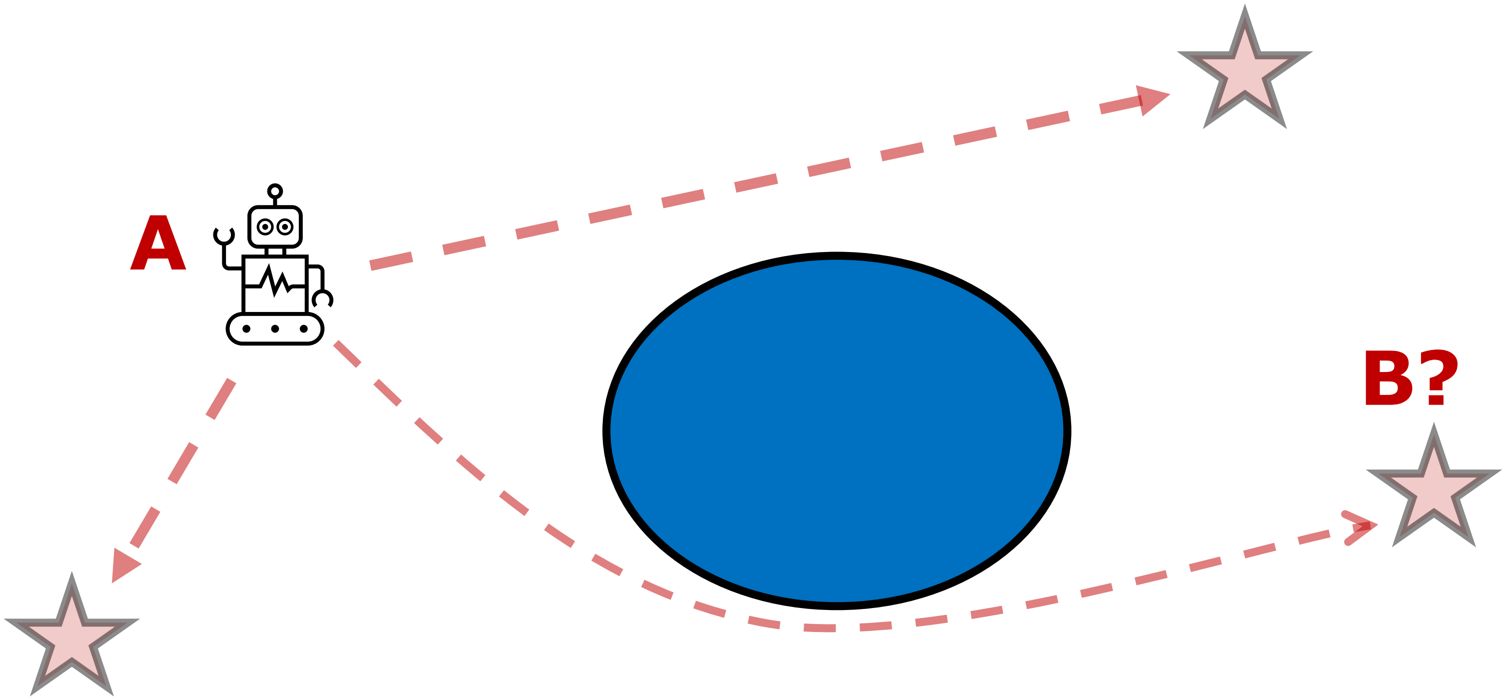
Trajectory planning: geometric path + time information



Path planning: geometric path

What is Decision-Making?

- Find goal configuration(s) (B) to fulfill a particular task – *where*?

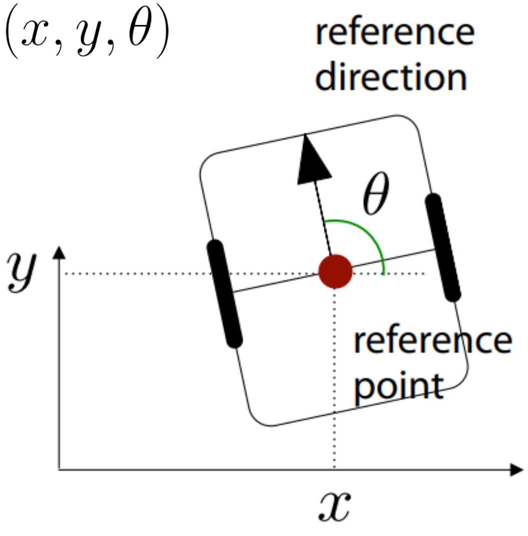


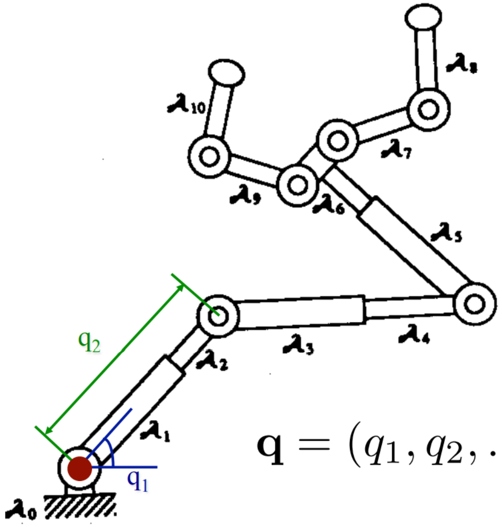
Configurations

- Robot configuration $\mathbf{q}$: specifies *all* robot points relative to a fixed coordinate system

- Examples:

$\mathbf{q} = (x, y, \theta)$





$\mathbf{q} = (q_1, q_2, \ldots, q_{10})$

http://srl.informatik.uni-freiburg.de/teachingdir/ws13/slides/12-RobotMotionPlanning.pdf

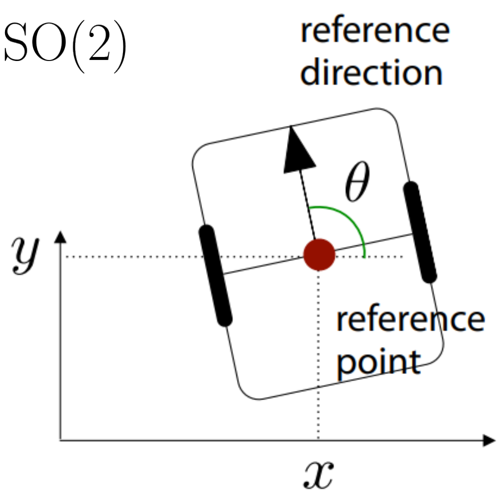
Configuration Space

- Configuration space (C-space): space of all possible configurations.
- Workspace: set of points which a robot can reach.

- Examples:

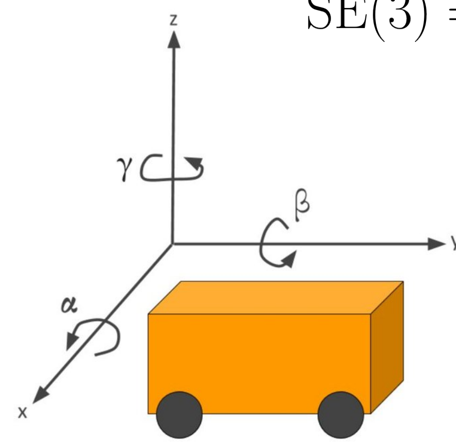
$$\mathbf{q} = (x, y, \theta)$$

$$\mathrm{SE}(2) = \mathbb{R}^2 \times \mathrm{SO}(2)$$



$$\mathbf{q} = (x, y, z, \alpha, \beta, \gamma)$$

$$\mathrm{SE}(3) = \mathbb{R}^3 \times \mathrm{SO}(3)$$



http://srl.informatik.uni-freiburg.de/teachingdir/ws13/slides/12-RobotMotionPlanning.pdf
https://automaticaddison.com/how-to-describe-the-rotation-of-a-robot-in-3d/

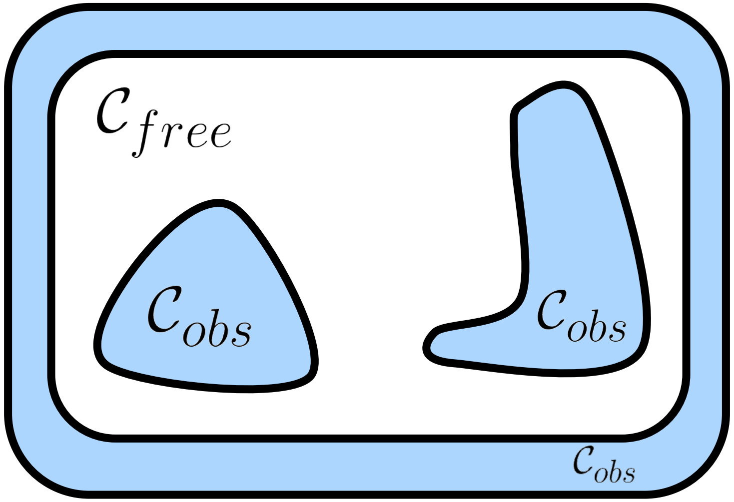
Configuration Space

- Free space $\mathcal{C}_{free}$ and obstacle space $\mathcal{C}_{obs}$

$$\mathcal{C}_{obs} = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{A}(\mathbf{q}) \cap \mathcal{O} \neq \emptyset\}$$

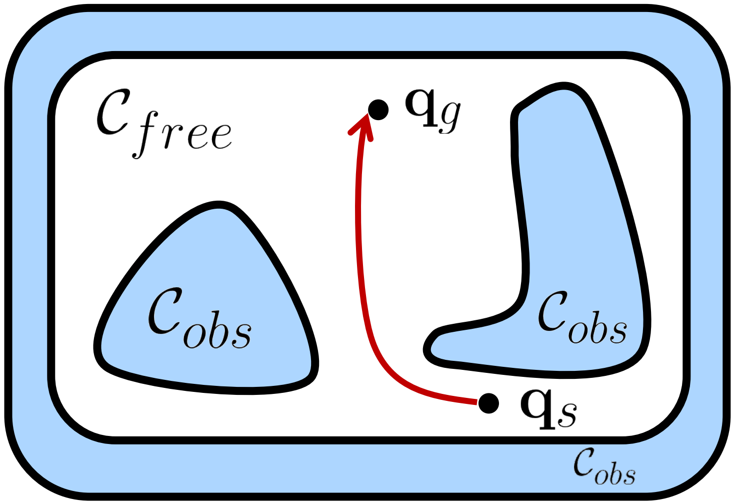
$$\mathcal{C}_{free} = \mathcal{C} / \mathcal{C}_{obs}$$

where $\mathcal{W} = \mathbb{R}^m$ is the robot workspace, $\mathcal{O} \in \mathcal{W}$ is the set of obstacles, and $\mathcal{A}(\mathbf{q})$ is the robot in configuration $\mathbf{q} \in \mathcal{C}$.



Motion Planning Problem

- Given a start configuration $\mathbf{q}_s$ and a goal configuration $\mathbf{q}_g$, find a continuous path that satisfies $\tau(0) = \mathbf{q}_s$, $\tau(1) = \mathbf{q}_g$, and $\tau : [0, 1] \to \mathcal{C}_{free}$



Key Considerations

- Robot characteristics
  - Degrees of freedom
  - Physical shape
  - Motion constraints
  - Dynamic constraints

- Algorithm properties
  - Optimality
  - Computational cost
  - Memory cost
  - Completeness
    - Probabilistic
    - Resolution
  - Online vs. offline
  - Anytime
  - Paths vs. trajectories
  - Exact vs. approximate

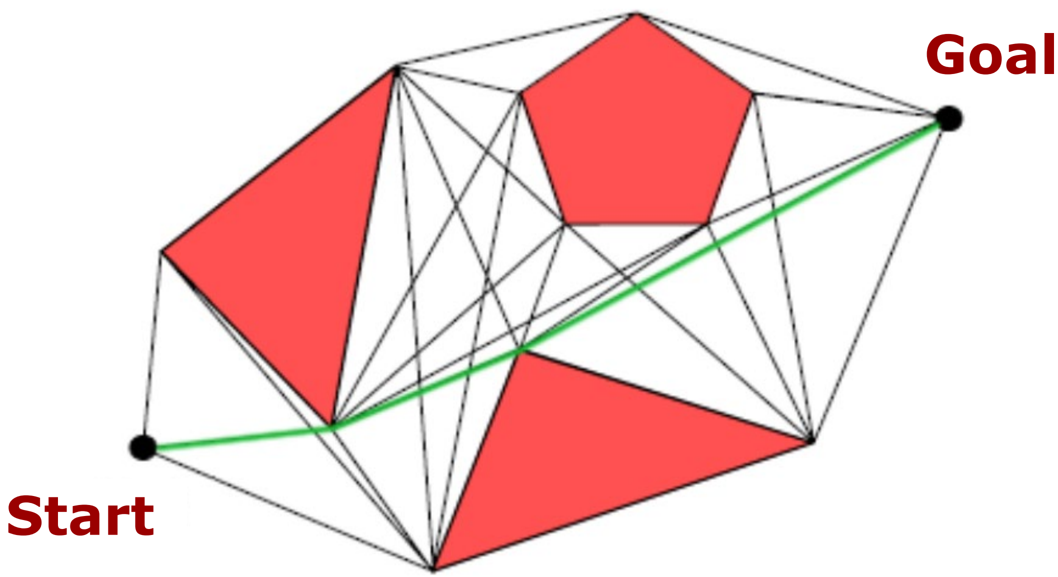
Planning Methods

Overview of Planning Methods

- Geometric
  - Visibility graphs, cell decomposition, Voronoi diagrams, etc.
- Potential field
  - Wavefront planner, navigation function, etc.
- Search-based
  - Dijkstra, A*, D*, D* Lite, etc.
- Sampling-based
  - RRT, RRT*, PRM, BIT, etc.
- Trajectory
  - Minimum time/energy/jerk/snap, etc.
- Bioinspired
  - Neural networks, genetic algorithms, ant colony optimisation, etc.

Geometric Methods

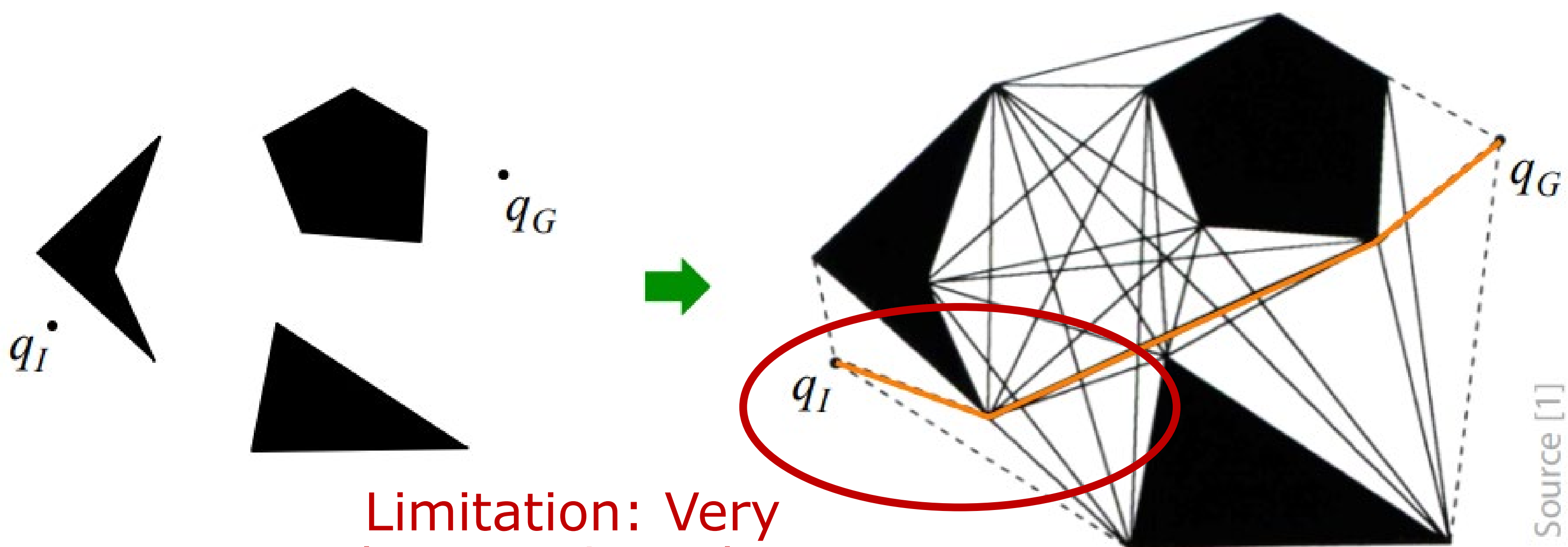
- Roadmap to capture connectivity of free space
  - Vertex: configuration in $\mathcal{C}_{free}$
  - Edge: collision-free path in $\mathcal{C}_{free}$
- Plan paths using search-based algorithm



https://resources.mpi-inf.mpg.de/departments/d1/teaching/ss10/Seminar_CGGC/Slides/06_Bazhenova_RMP.pdf

Geometric Methods

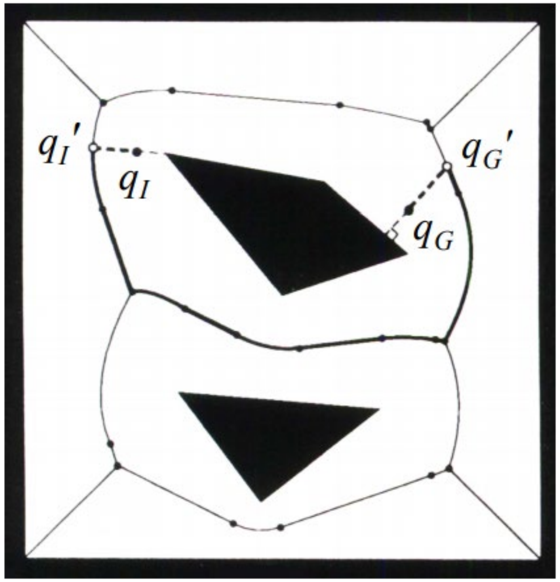
- **Visibility graphs**
- <u>Idea</u>: Connect all intervisible vertices of obstacles
- Plan a path from start to goal location along these edges
- Shortest path for polygonal obstacles



Limitation: Very close to obstacles!

Geometric Methods

- **Generalised Voronoi diagrams**
- Idea: Connect points that are equidistant from the closest two or more obstacle boundaries, including workspace edges
- Plan a path from start to goal location along these edges



Geometric Methods

- **Generalised Voronoi diagrams**
- Benefits:
  - Conservative paths
  - Similar to human behaviour
- Limitations:
  - Difficult to compute in higher dimensions
  - *Too* conservative paths
  - Unstable – small changes in environment lead to large changes in diagram
  - Issues with short-range sensors

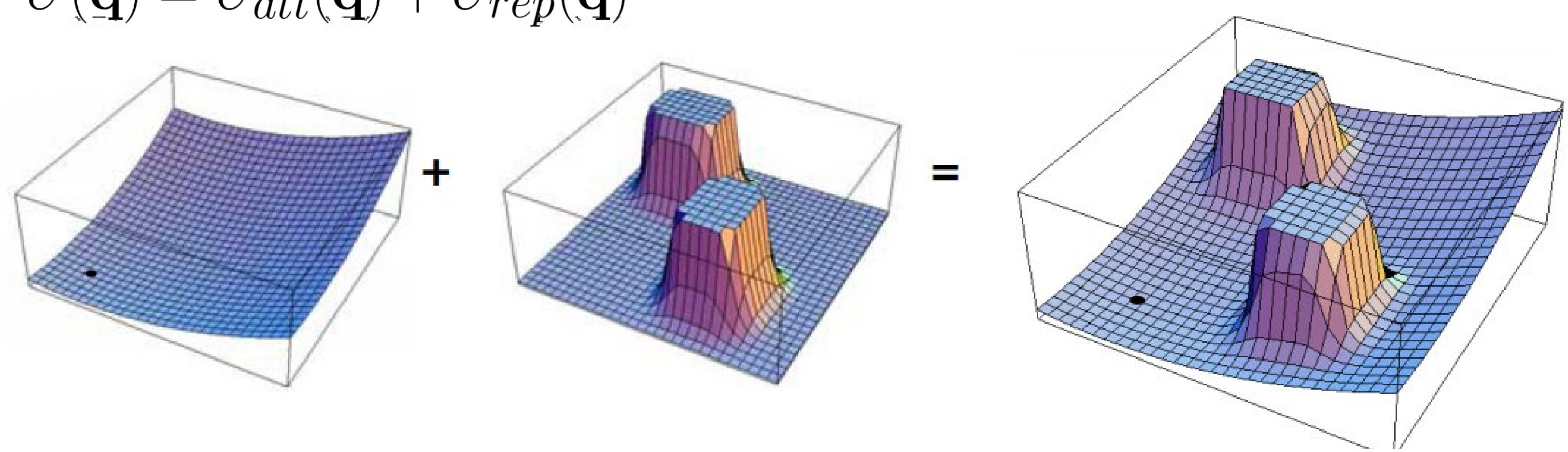
Potential Field Methods

- Robot is a point mass in a **potential field**
- Potential field is a differentiable function

$$U : \mathbb{R}^m \rightarrow \mathbb{R}$$

- Attractive potential $U_{att}(\mathbf{q})$ - attracts to goal
- Repulsive potential $U_{rep}(\mathbf{q})$ - repels from obstacles

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

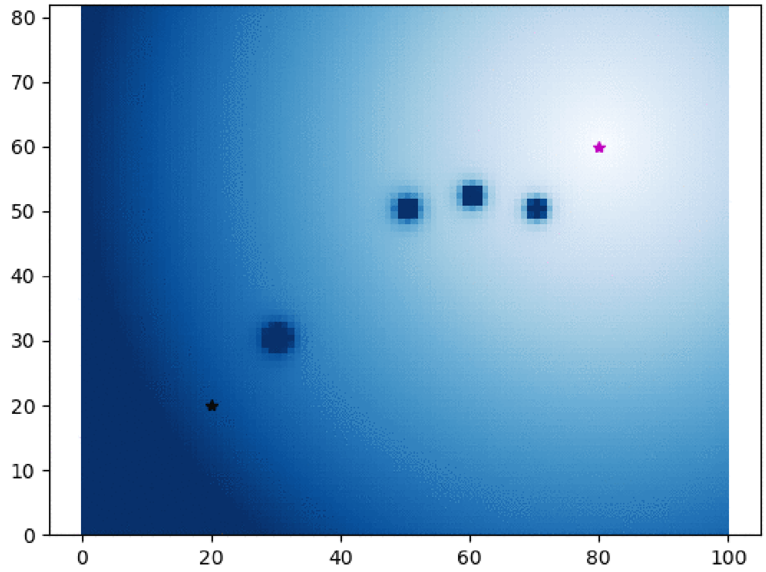


https://cs.gmu.edu/~kosecka/cs685/cs685-potential-fields.pdf

Potential Field Methods

- Considerations:
  - Modelling the potential
  - Solution method
- <u>Idea</u>: follow negative gradient using gradient descent

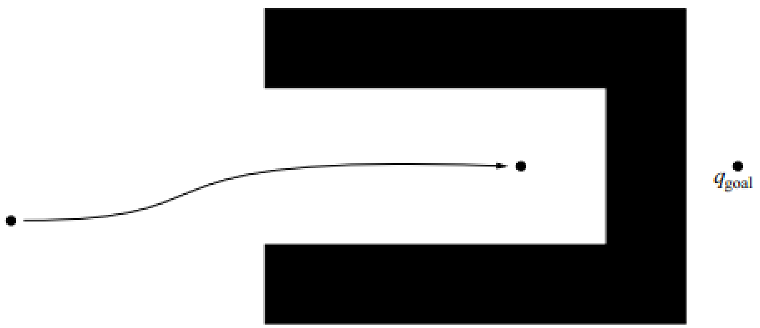
$$F(\mathbf{q}) = -\Delta U(\mathbf{q})$$

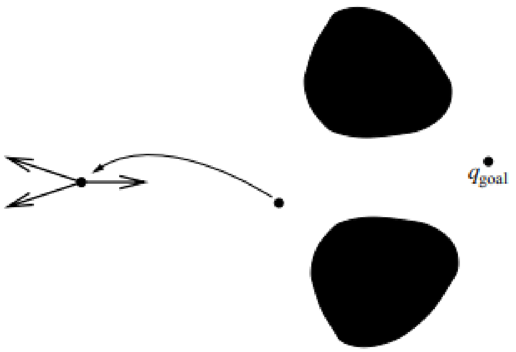


https://pythonrobotics.readthedocs.io/en/latest/modules/path_planning.html

Potential Field Methods

- Benefits:
  - Simple implementation
  - Online collision avoidance
- Limitations:
  - Scalability – explicit modelling of obstacles and free space
  - Local minima



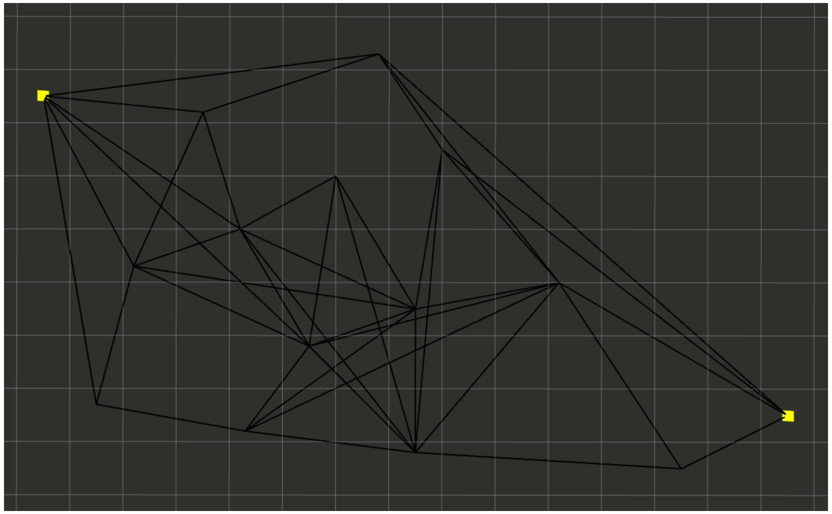


http://srl.informatik.uni-freiburg.de/teachingdir/ws13/slides/12-RobotMotionPlanning.pdf

Search-based Methods

- Discrete-space planning
- Planning graph $G = (V,E)$
  - V is a set of vertices → configurations
  - E is a set of edges → collision-free connections
- Apply a search-based algorithm to find a path
  - Depth-first, breadth-first, Dijkstra, A*, etc.

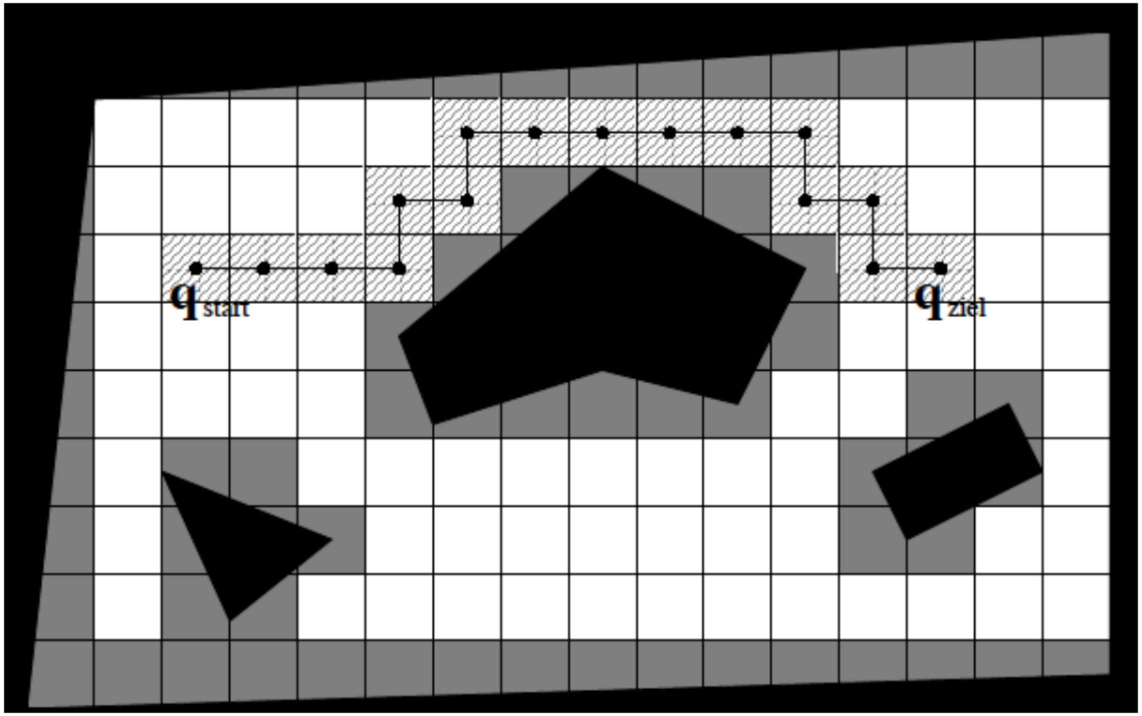
- Example: visibility graph + Dijkstra's algorithm



https://github.com/tedklin/pathviz

Search-based Methods

- **Grid map**: special case of graphs
- Subdivide $\mathcal{C}_{free}$ into smaller cells
- Enables planning with discrete methods

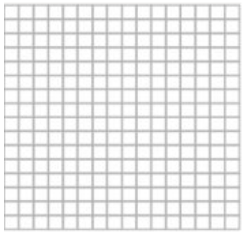


http://gki.informatik.uni-freiburg.de/teaching/ws1011/imap/05_SearchAlgorithmsAndPathPlanningMAS_PartB.pdf)

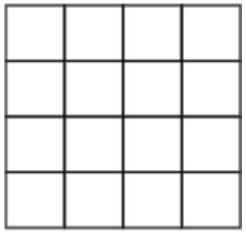
Search-based Methods

- **Grid map**: special case of graphs
- Considerations:
  - Resolution

high

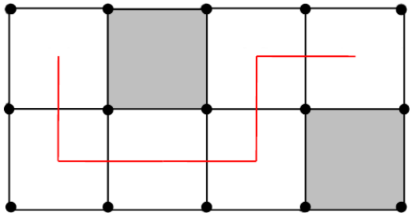


low

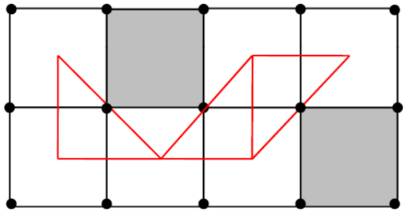


  - Connectivity

4 neighbours



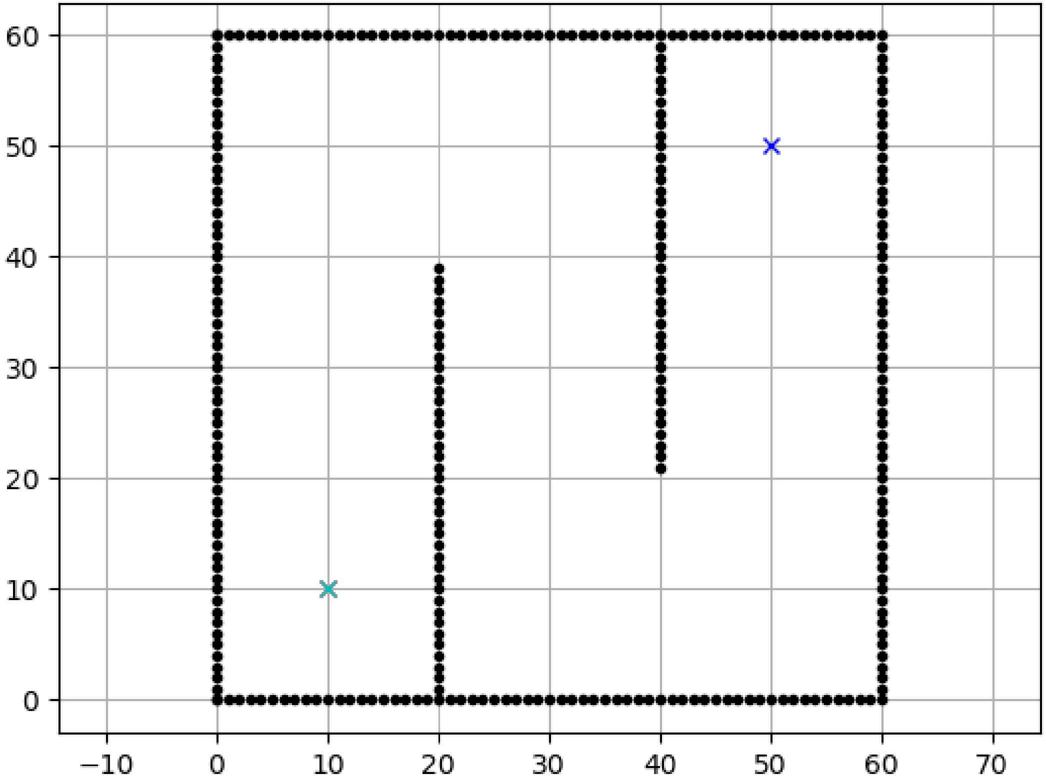
8 neighbours



https://cw.fel.cvut.cz/old/_media/courses/b4m36uir/lectures/b4m36uir-lec04-handout.pdf

Search-based Methods

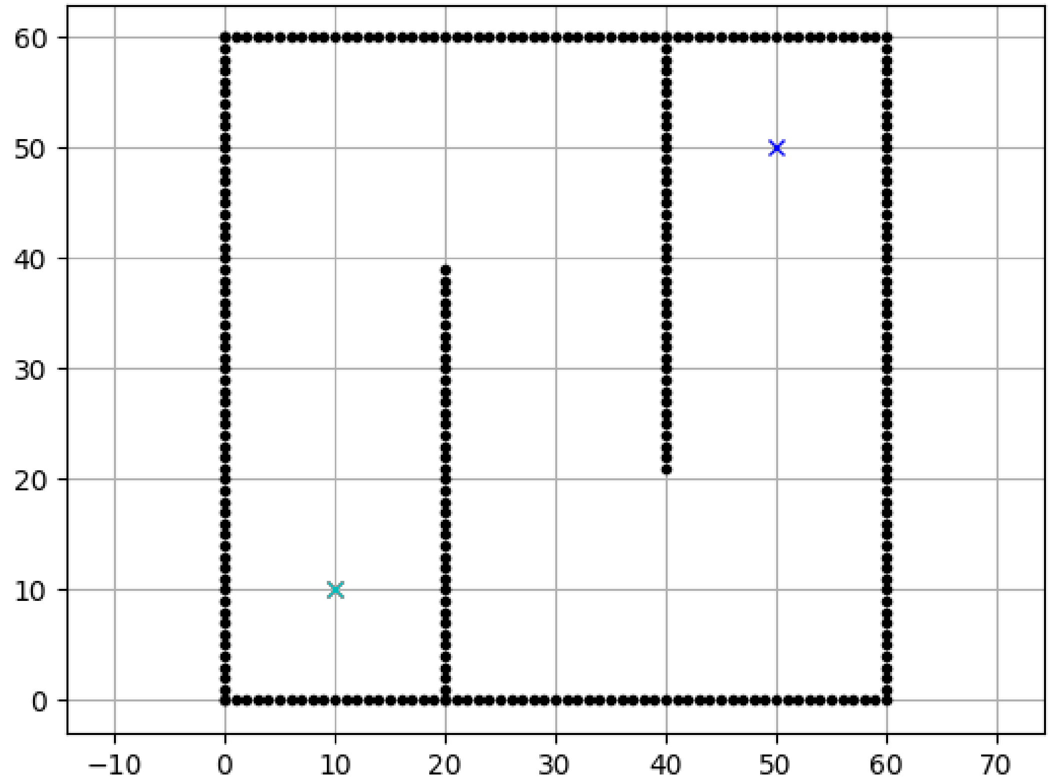
- <u>Example</u>: **Dijkstra's algorithm**
  - Expand nodes with minimal distance to the initial node



https://pythonrobotics.readthedocs.io/en/latest/modules/path_planning.html

Search-based Methods

- Example: **A* algorithm**
  - Also considers heuristic based on distance to the goal.



https://pythonrobotics.readthedocs.io/en/latest/modules/path_planning.html

So far...

1. Define the configuration space $\mathcal{C}$
2. Discretise the configuration space $\mathcal{C}$
3. Search the configuration space $\mathcal{C}$

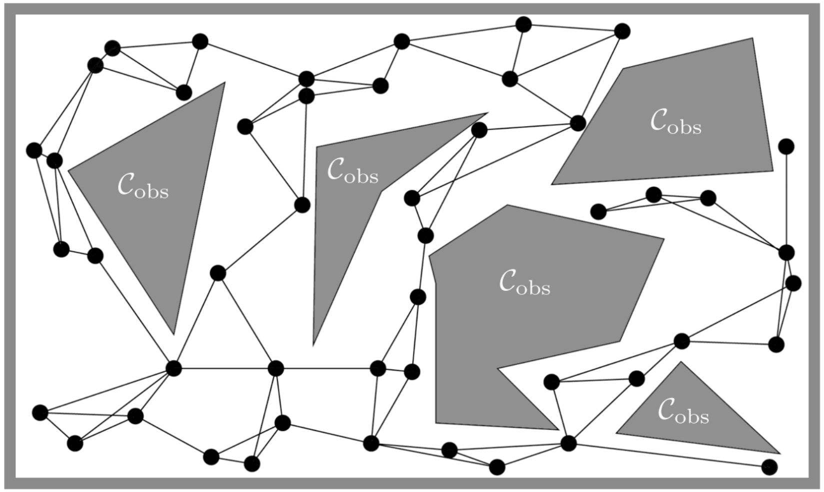
Sampling-based Methods

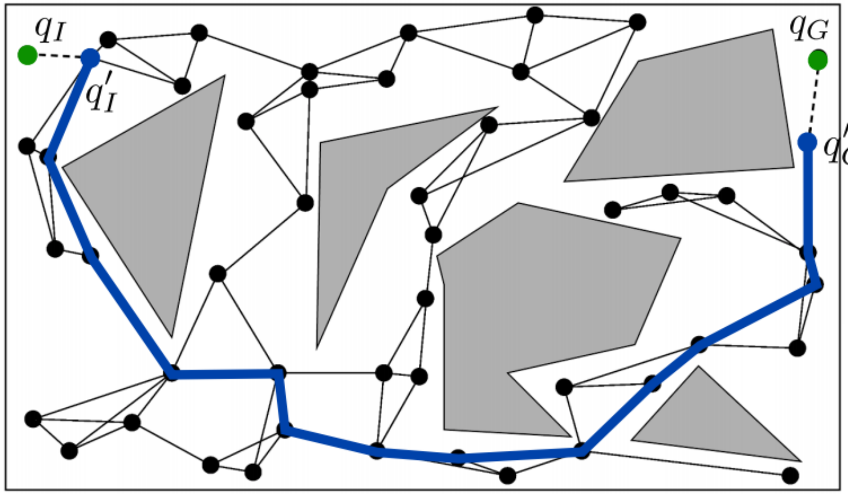
- Discretisation is expensive, especially in high-dimensional spaces
- Idea: Sample in configuration space
- Only check sampled configurations for collisions
- Construct a graph that consists of sampled configurations
- Trade off completeness for efficiency
- Examples:
  - Probabilistic roadmap (PRM), Rapidly-exploring random tree (RRT), RRT*, etc.

Sampling-based Methods

- Example: Probabilistic roadmap (PRM)

1. Learning phase
2. Query phase



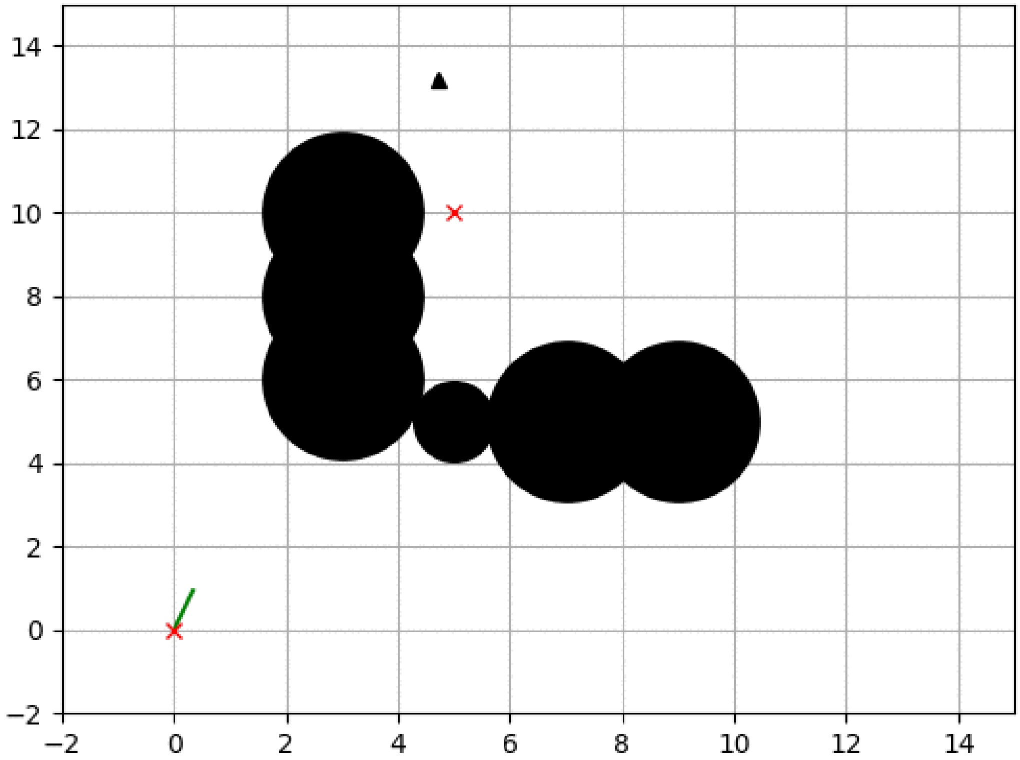


Sampling-based Methods

- Example: Rapidly-exploring random tree (RRT)

Sampling-based Methods

- Example: Rapidly-exploring random tree (RRT)



http://srl.informatik.uni-freiburg.de/teachingdir/ws13/slides/12-RobotMotionPlanning.pdf

Comparison of Methods

Method	Complete	Optimal	Scalability to higher DoFs	Comments
Visibility	Yes	Yes	No	+ Explicit free space - Poor scalability - Robot might travel close to obstacles
Voronoi	Yes	No	No	+ Explicit free space + Maximum clearance - Paths may be too conservative - Poor scalability
Potential field	Yes	No	Environment-dependent	+ Easy to implement + Can account for uncertainty - Susceptible to local minima
Dijkstra/A*	Yes	Grid	No	+ Faster than uninformed search + A* uses a heuristic function to drive the search more efficiently - Poor scalability
PRM	Yes	Graph	Yes	+ Efficient for multi-query problems + Probabilistic completeness - Jagged path
RRT	Yes	No	Yes	+ Efficient for single-query problems + Probabilistic completeness - Jagged path

Summary

- Autonomous systems overview
- Planning problem
  - <u>Aim</u>: Find a sequence of collision-free configurations between a start and goal
  - Configuration space approach
  - Key considerations
- Planning methods
  - Geometric, potential field, search-based, sampling-based

Further Reading

- Map representations (stanford.edu)
- Sampling-Based Robot Motion Planning | October 2019 | Communications of the ACM
- Introduction to Robotics #4: Path-Planning | Correll Lab (colorado.edu)
- Great animations - Path Planning — PythonRobotics documentation
- Planning Algorithms – Steven M. LaValle (2006)
  - Planning Algorithms / Motion Planning (lavalle.pl)
  - Esp. Ch. 1+2
- Principles of Robot Motion: Theory, Algorithms, and Implementations – H. Choset *et al*. (2005)
  - Principles of Robot Motion (cmu.edu)