MSAI Robotics course

Lecture 6

Path and Motion Planning

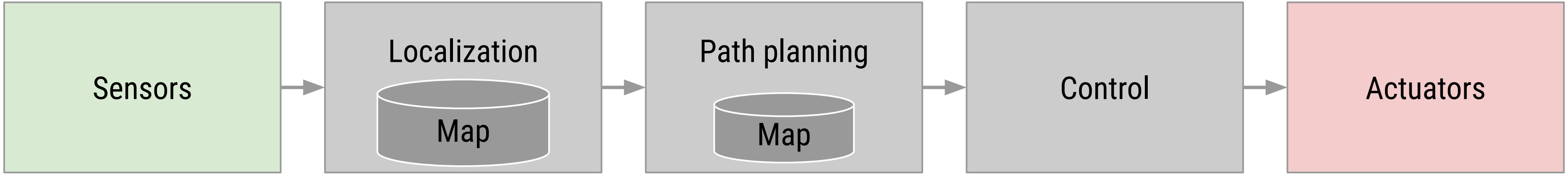
Vladislav Goncharenko
Materials by Oleg Shipitko
MIPT, 2022

girafe
ai

Outline

1. Planning problem definition
2. Configuration space
3. Cell decomposition (DFS, BFS, A*)
4. Road maps (visibility graphs, Voronoi diagram, RRT)
5. Potential field

(SIMPLIFIED) CONTROL SCHEME OF MODERN MOBILE ROBOT



Planning problem definition

girafe
ai

01

WHAT IS PATH AND MOTION PLANNING?

- **Path planning** — finding a collision-free transition (path) from some initial to some final configuration in space containing obstacles.
- **Motion planning** — planning and executing commands leading the robot to follow the planned path.

WHAT IS PATH AND MOTION PLANNING?

Path planning

"In 3 kilometers turn right..."

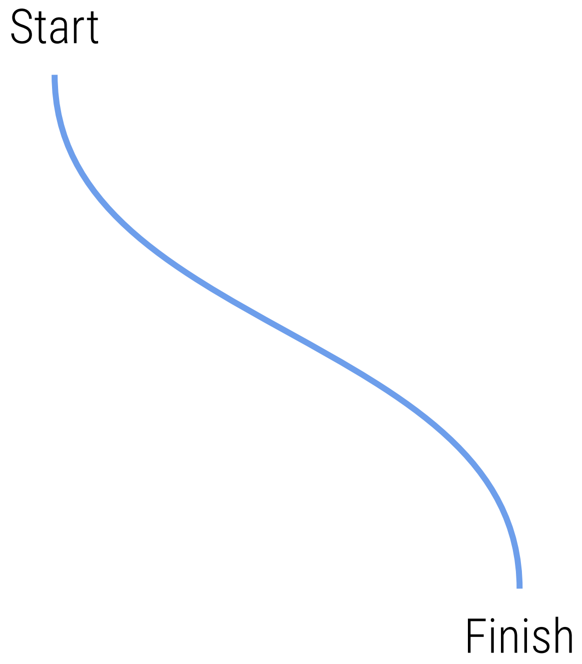
Motion planning

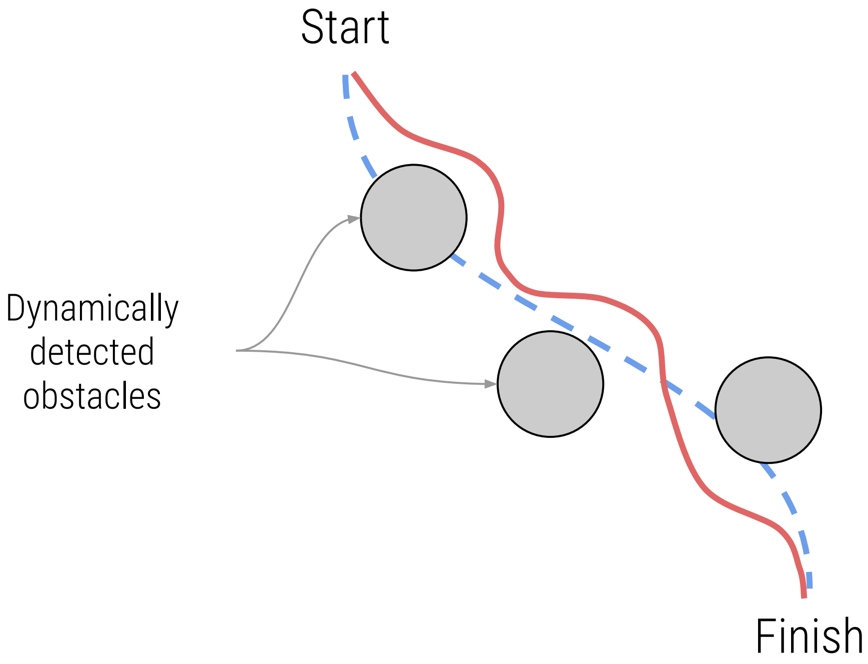
"Turn the steering wheel 20 degrees clockwise and press of the accelerator pedal by 10%..."

OTHER COMMON TERMS

- **Global path planning, global path, route** — given a start point, and an end point, and a map, one needs to find the (optimal) path. Often the kinematic constraints of the robot are not taken into account while planning.
- **Local path planning, local path, trajectory** — having information coming from sensors, it is necessary to find a local path (just a small part) that is free from collisions and deviates as little as possible from the previously planned global path.
- **Collision avoidance** — the same as local planning.

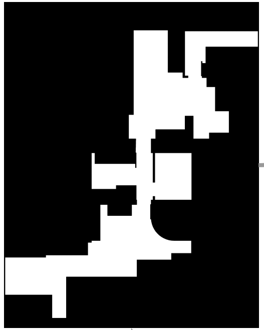
GLOBAL VS. LOCAL PATH





TWO LAYERED PLANNING ARCHITECTURE

Map



Information from sensors

Global planning

Global path

Local planning/ collision avoidance

Control commands

– small update frequency

– high update frequency

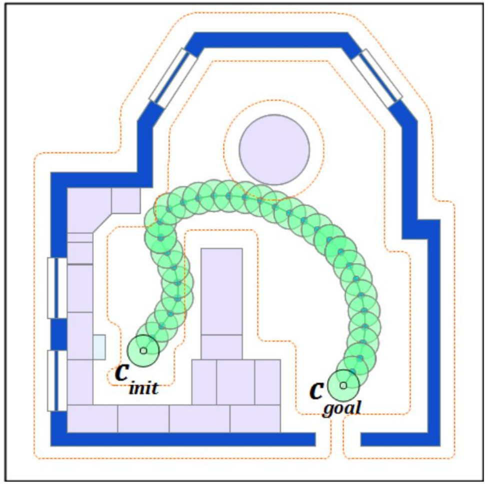
Configuration space

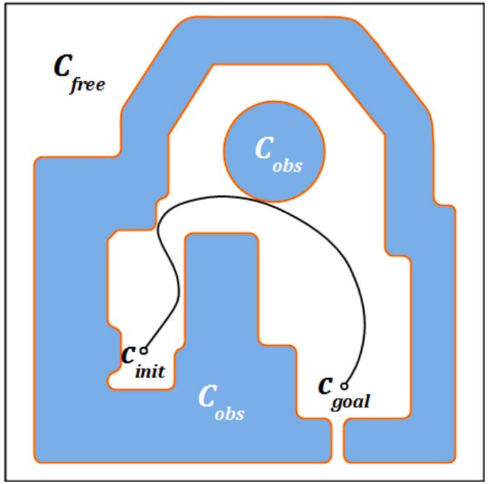
girafe
ai

02

CONFIGURATION SPACE

- Let $\mathbf{q}$ — vector uniquely defining the state of the robot
  - For example, the position of all links (joints) for a manipulator
  - For a mobile robot - position in space (x, y, θ)
- Then, configuration space $\mathbf{C}$ — the space of all possible states of the robot

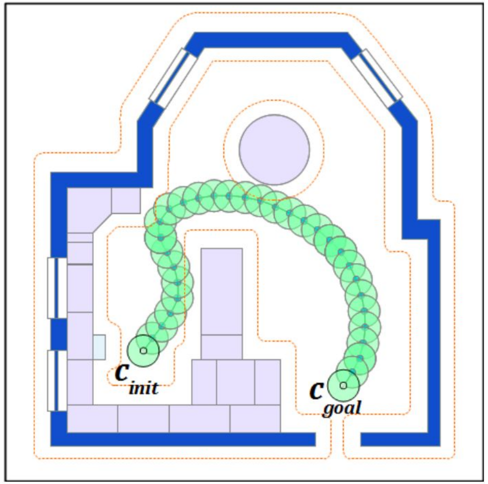


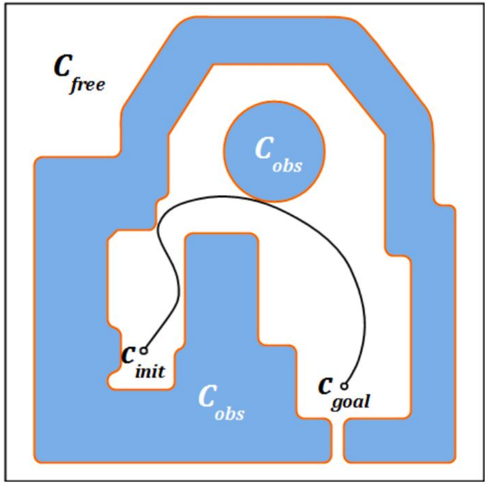


CONFIGURATION SPACE

- Let $W=R^m$ – robot working area (world representation)
- $C_{obs} \in W$ – obstacles set
- C_{init}, C_{goal} – initial and goal robot configurations
- Then:

$$C_{free} = \{q \in C \mid q \cap C_{obs} = \varnothing\}$$

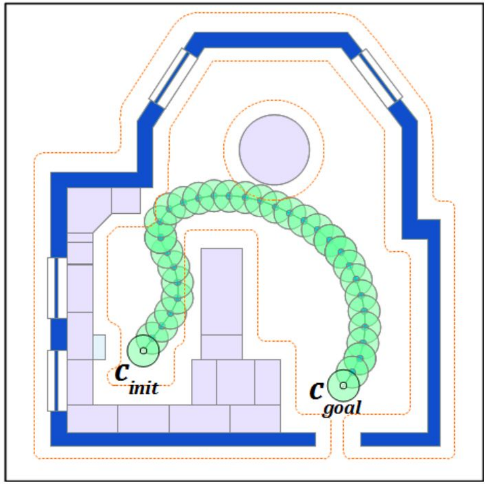


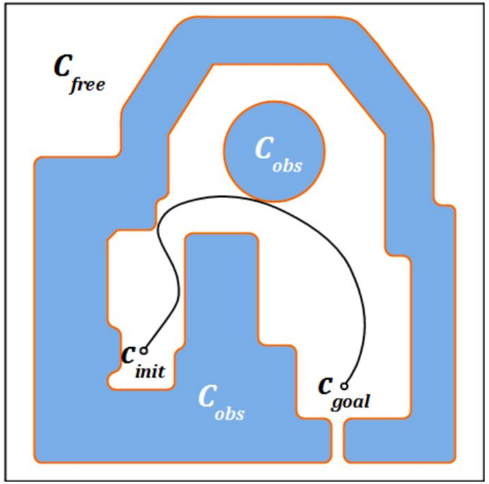


CONFIGURATION SPACE

- Then the path planning problem is reduced to finding the set $T = \{C_{init} \dots C_{goal}\} \subset C_{free}$

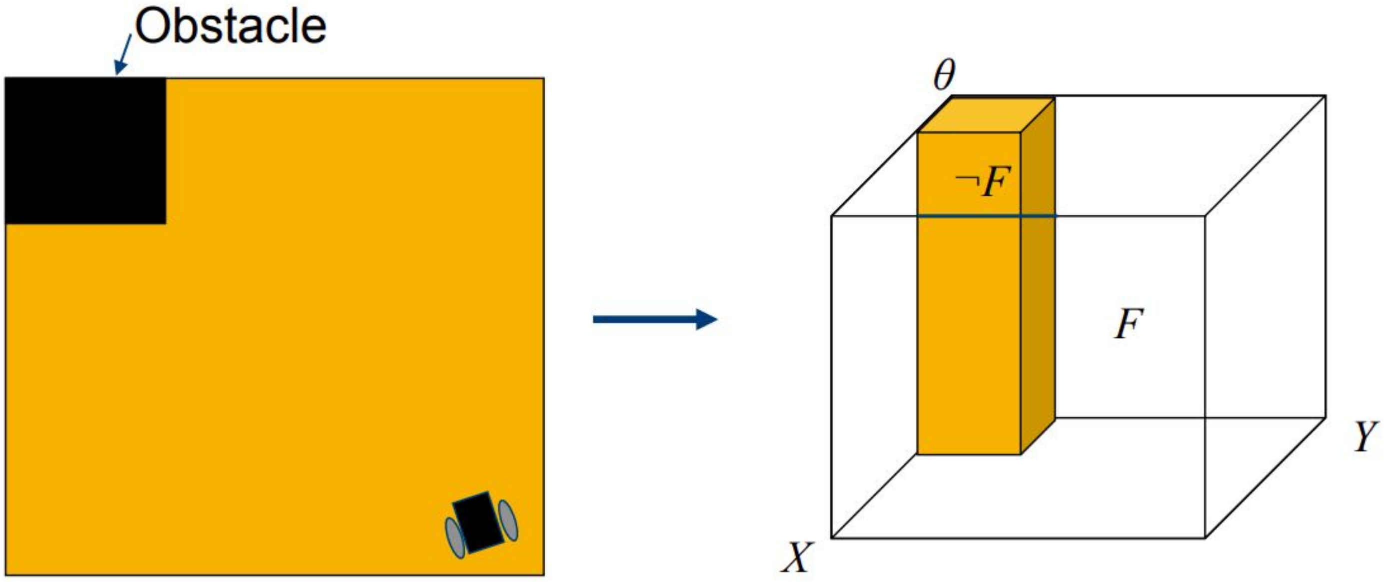
- In this setting, the robot can be considered as a point in $\mathbf{C}$-space.





CONFIGURATION SPACE

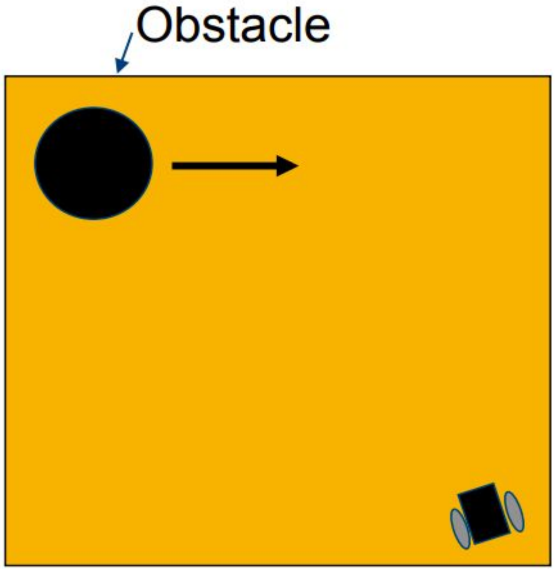
Mobile robot and static obstacle



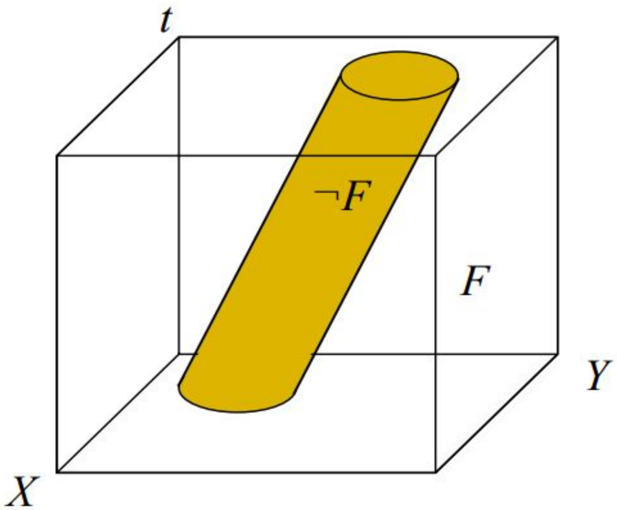
Workspace   Configuration Space

CONFIGURATION SPACE

Mobile robot and dynamic obstacle



Workspace



Configuration Space

COMMON APPROACH TO PATH PLANNING

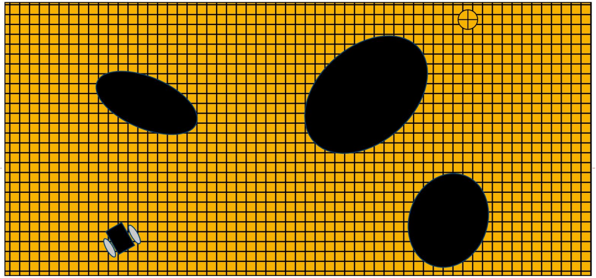
- Path planning most often boils down to the following set of steps:
  1. Define configuration space C
  2. Decompose (discretize) configuration spaces C
  3. Find the pat in the discretized space
     - Search algorithms
     - Sampling-based algorithms

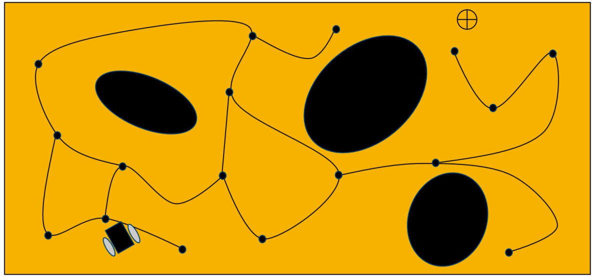
DEFINITION OF CONFIGURATION SPACE

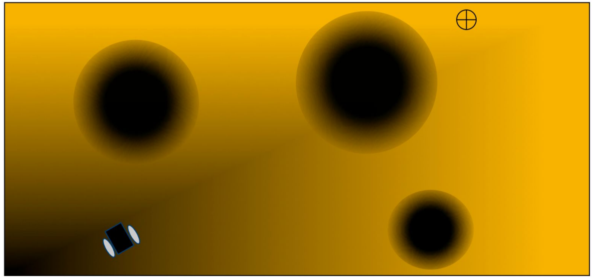
- Different planning problems may have different definitions of the configuration space. Many variants are possible for a wheeled robot:
  - For a holonomic robot (x, y)
  - For a nonholonomic robot (x, y, θ)
  - Taking velocities into account (x, y, θ, V, w)
  - ...

CONFIGURATION SPACE DECOMPOSITION

- Configuration space decomposition types C:
  - **Cell decomposition** — space is splitted into cells. The connectivity graph is built on the basis of the adjacency of cells.
  - **Roadmap decomposition** — represents free space as connected curves (routes).
  - **Potential field** — defines the potential field over the configurations space with a minimum at the target configuration and maximums at obstacles.





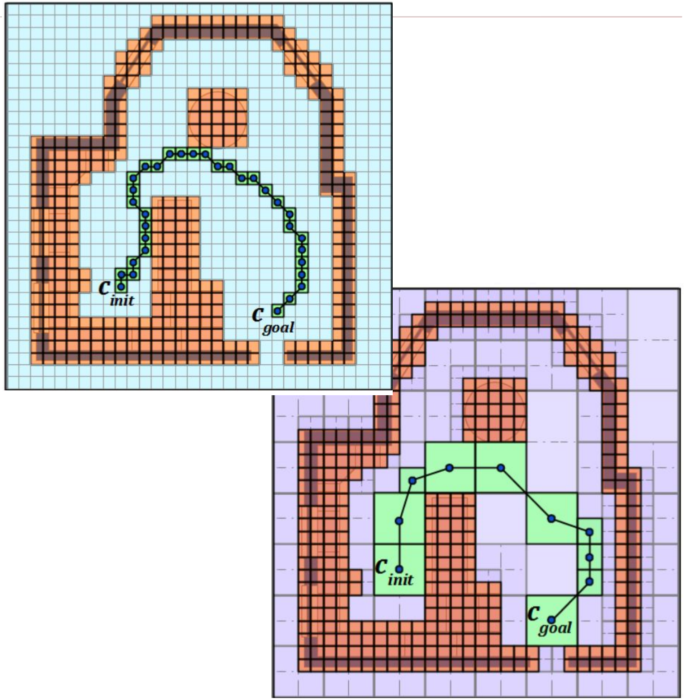


Cell decomposition

girafe
ai

03

CELL DECOMPOSITION BASED METHODS



- **All** space is divided into regions according to certain rules
- Adjacency of regions is established
- A connectivity graph is built
- The path planning problem is reduced to the problem of finding a path in the graph

UNINFORMED VS. INFORMED SEARCH

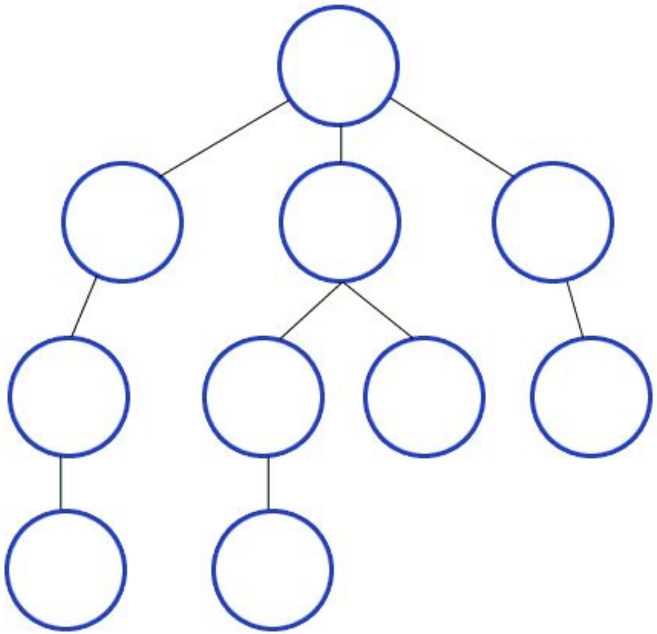
- **Uninformed search, blind search, brute-force search** — search for a path in the state space in which no auxiliary information is used. The only thing that can be done is to examine the nodes of the graph in different ways.
- **Informed search** — when additional information (heuristics) is available and can be used to say that one node is "more promising" than another.

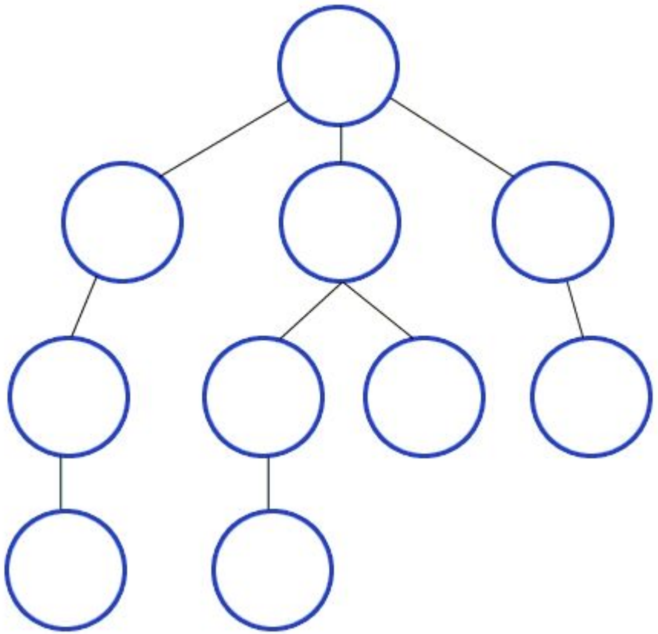
UNINFORMED SEARCH

Algorithms:

- ❑ Breadth-first search
- ❑ Depth-first search
- ❑ Bidirectional search
- ❑ Limited-depth search
- ❑ Uniform-cost search
- ❑ …

BREADTH-FIRST VS. DEPTH-FIRST SEARCH





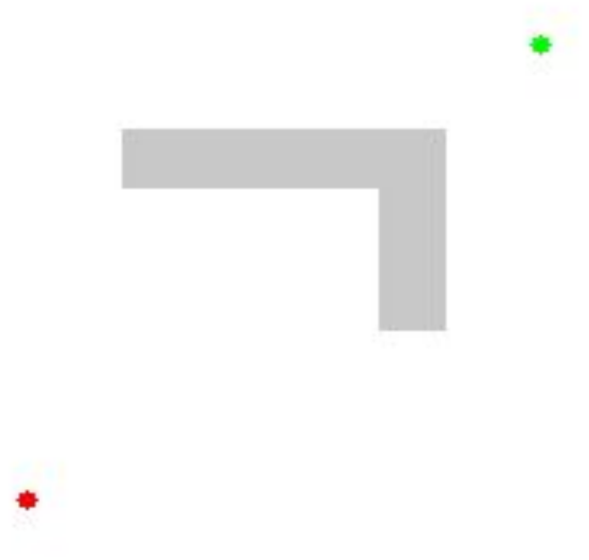
BREADTH-FIRST SEARCH

- **Breadth-First Search** is an optimal uninformed search algorithm if all nodes (actions) have equal cost.
- **Uniform-cost search** generalizes the idea of Breadth First Search when nodes have different costs.
- **Uniform-cost search** visits nodes in ascending order of path cost from the root (start) node (almost equivalent to Dijkstra's algorithm).

A*

- The priority of the vertex to be traversed is determined by the sum:
  - $f(v) = g(v) + h(v)$,
  - where $g(v)$ – the cost of the path from the initial vertex to the given one,
  - $h(v)$ – heuristic estimation of the path cost to the target vertex.
- Thus, A * is a generalization of Dijkstra's algorithm for $f(v) = g(v)$ and the greedy search algorithm for $f(v) = h(v)$.
- The algorithm finds the optimal solution provided that the heuristic estimate is optimistic: $h(v) \leqslant$ **exact cost of the path to the target vertex**
- Euclidean distance (L_2-norm) to the target is often used as a heuristic.

A*



A* AND FRIENDS

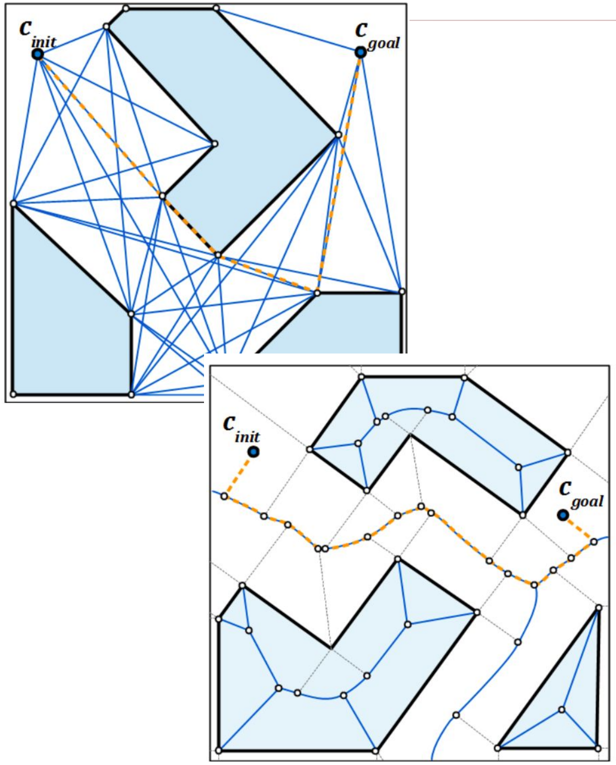
- **D*** family of algorithms for dynamic information accounting and replanning
- **Theta*** family of algorithms for planning paths with arbitrary direction
- **Lifelong Planning A*** family of algorithms is also used to replan using previous results when new obstacles appear in the scene
- …

Road maps

girafe
ai

04

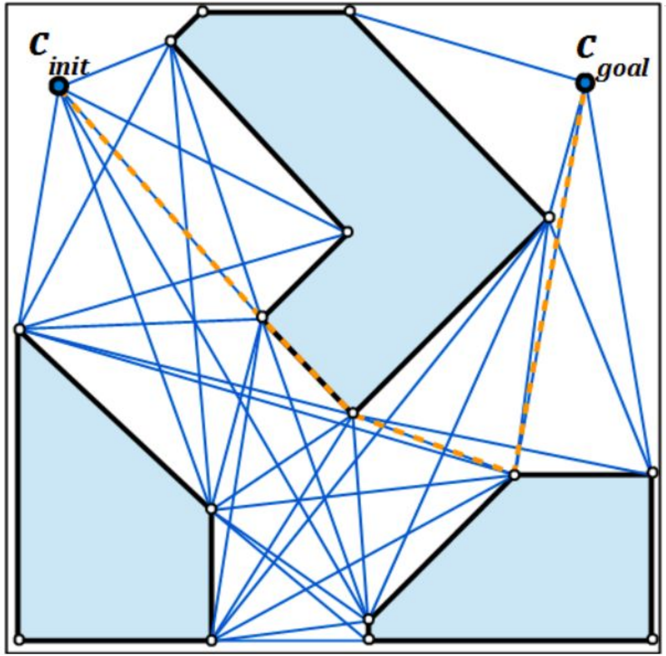
ROADMAP DECOMPOSITION BASED METHODS



- **Free** space is divided into regions according to certain rules
- Adjacency of regions is established
- A connectivity graph is built
- The path planning problem is reduced to the problem of finding a path in the graph

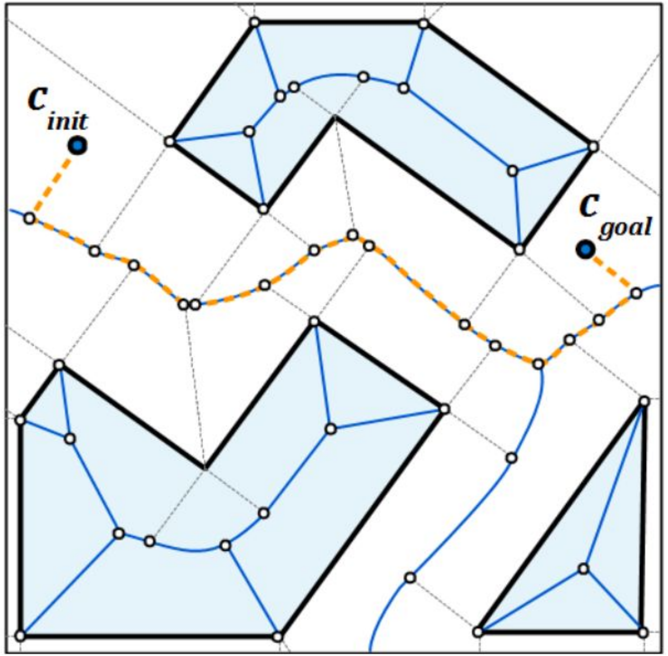
VISIBILITY GRAPHS

- All obstacles corners + start and end points of the path are used as nodes
- Two vertices are connected by edge if:
  - One vertex is "visible" from the other
  - The resulting edge does not cross obstacles
- The complexity of building a graph $O(n^2 \log n)$, where n — number of vertices



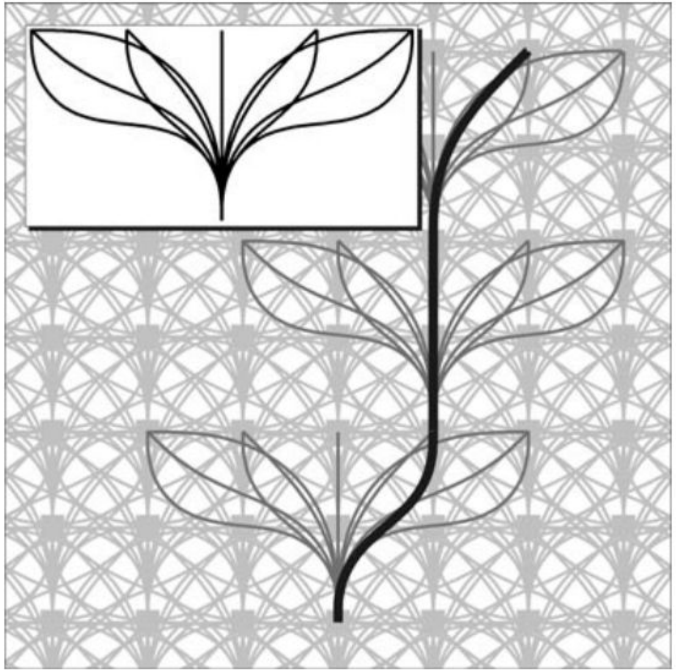
VORONOI DIAGRAM

- Partition of a finite set of points in which each region of the partition forms a set of points closer to one of the elements of the set than to any other element
- Paths built according to the Voronoi diagram are at the farthest distance from obstacles
- Naive algorithm — $O(n^4)$. Best — $O(n\ log\ n)$



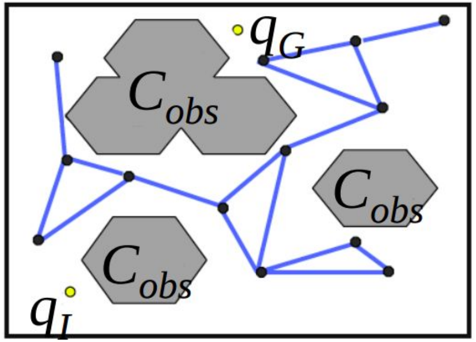
STATE LATTICE BASED METHODS

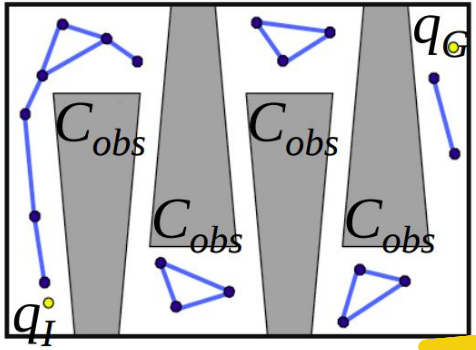
- The space is discretized in such a way that the transitions between the nodes are obviously kinematically reachable
- Transitions are built from predefined motion primitives
- Any of the existing directed graph search algorithms can be applied to find a path in the resulting state lattice



PROBABILISTIC ROAD MAPS

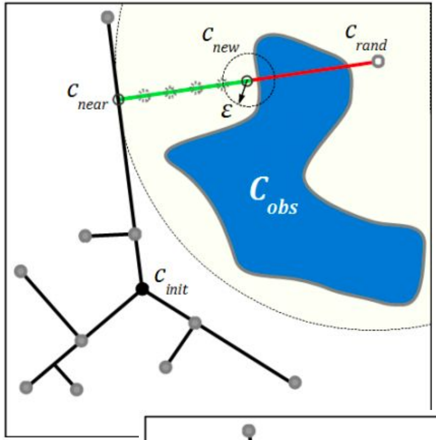
- ❑ Random samples are taken from C_{free}
- ❑ Close samples are connected to each other by an edge, provided there is no collision on the edge
- ❑ The proximity of nodes can be determined in different ways:
  - ❑ k-nearest neighbours
  - ❑ All nodes in a fixed radius
  - ❑ …
- ❑ The path in the resulting graph is searched for by any search algorithm

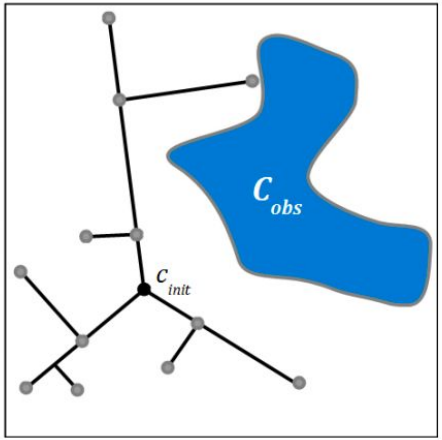




RAPIDLY EXPLORING RANDOM TREES

- The main idea is to explore space more efficiently
- The essence of the algorithm:
  - Initially, the tree consists of a start node C_{init}
  - Let's generate a random node $C_{rand} \in C_{free}$
  - Connect C_{rand} to the nearest tree node C_{near} by edge
    - Moreover, if an edge passes through an obstacle, then we leave the maximum part of the edge (C_{near}, C_{new}) that does not intersect obstacles
  - Every n-th iteration we will take C_{goal} as C_{rand}. If it is possible to connect C_{goal} to the tree, the path is found.



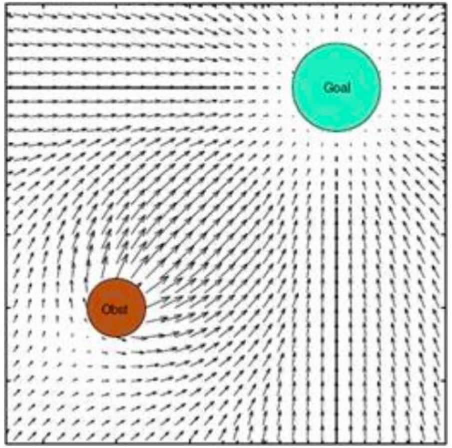


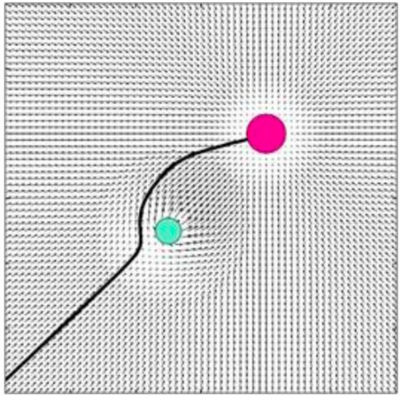
Potential field

05

POTENTIAL FIELD BASED METHODS

- The main idea is to set the **navigation** function f, indicating the direction to the target point from any current robot configuration
- ∇f (q) — indicates the direction to the target point
- f is set in such a way that obstacles push the robot away, while the target point attracts it





WHAT HAVE WE MISSED?

- ❑ A huge number of improvements and extensions of the considered basic methods
- ❑ Local planning methods based on all kinds of curves (polynomial planning, clothoids, Bezier curve, Dubins paths, etc.)
- ❑ Optimization planning methods
- ❑ Dynamic Programming Techniques / Markov Decision Making
- ❑ ...

ADDITIONAL RESOURCES

1. ARW – Lecture 03 Motion Planning. Chris Clark
2. Introduction to Mobile Robotics. Path and Motion Planning. Wolfram Burgard
3. Amit’s A* Pages
4. PythonRobotics

Thanks for attention!

Questions? Additions? Welcome!

girafe ai