Intelligent Robots Practice

Path and Motion Planning

Chungbuk National University, Korea
Intelligent Robots Lab. (IRL)

Prof. Gon-Woo Kim

Intelligent Robots Lab.
Chungbuk National University

Contents

- Introduction
- Motion Planning

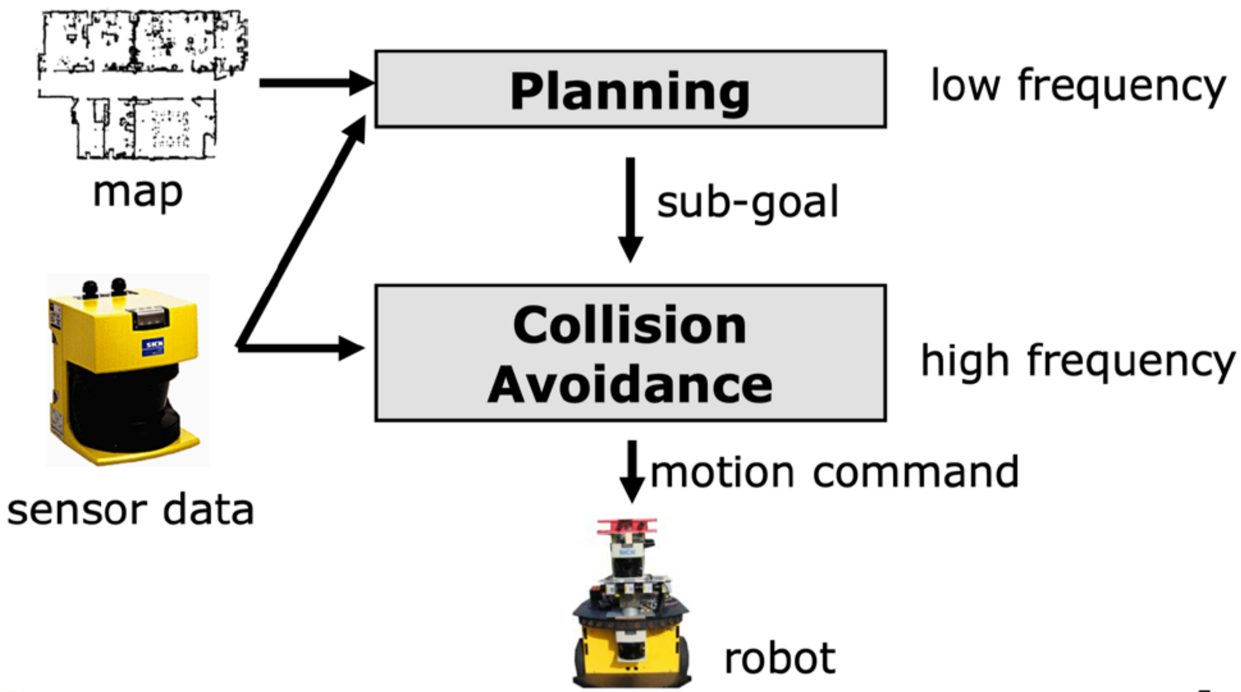
Introduction

Motion Planning

- Motion Planning
  - Goals:
    - Collision-free trajectories.
    - Robot should reach the goal location as quickly as possible.
- Motion Planning in Dynamic Environments
  - How to react to unforeseen obstacles?
    - Efficiency, Reliability
  - Methods
    - Dynamic Window Approaches
    - Grid-map-based planning
    - Nearness-Diagram-Navigation
    - Vector-Field-Histogram+
    - A*, D*, D* Lite, etc

Motion Planning

- Motion Planning
  - Two Challenges
    - Calculate the optimal path taking potential uncertainties in the actions into account
    - Quickly generate actions in the case of unforeseen objects
  - Classic Two-layered Architecture



Motion Planning

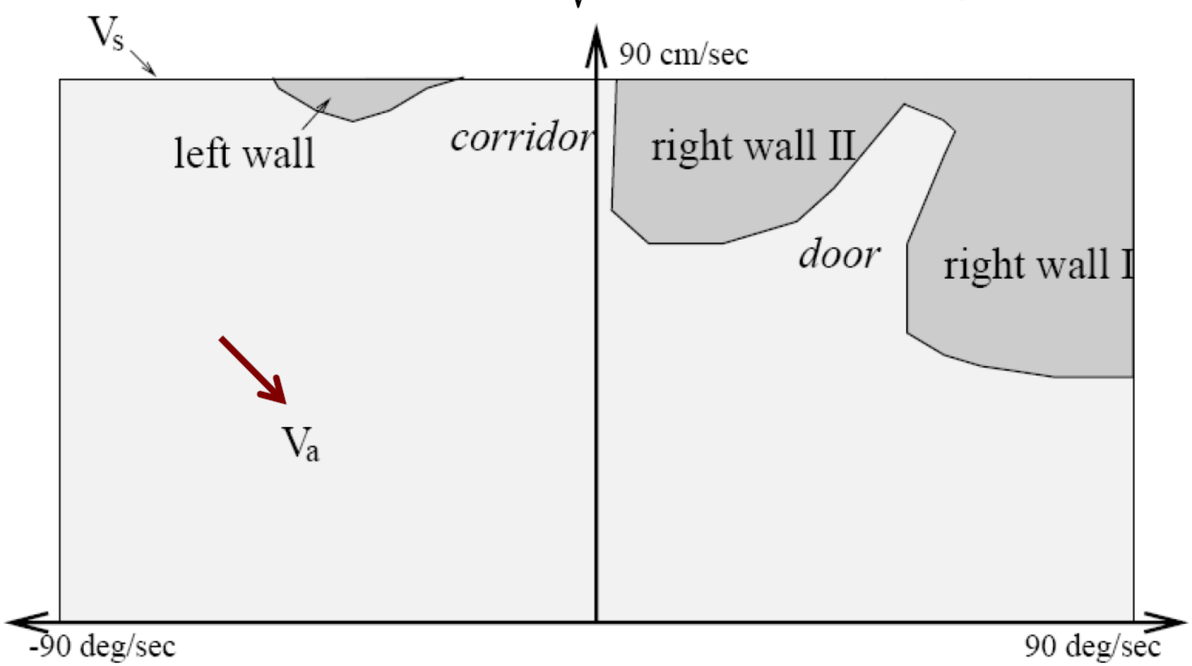
Motion Planning

- Dynamic Window Approach
  - Collision avoidance
    - Determine collision-free trajectories using geometric operations
  - Robot moves on circular arcs
  - Motion commands (v,ω)
  - Which (v,ω) are admissible and reachable?

Motion Planning

- Dynamic Window Approach
  - Admissible Velocities
    - Speeds are admissible if the robot would be able to stop before reaching the obstacle

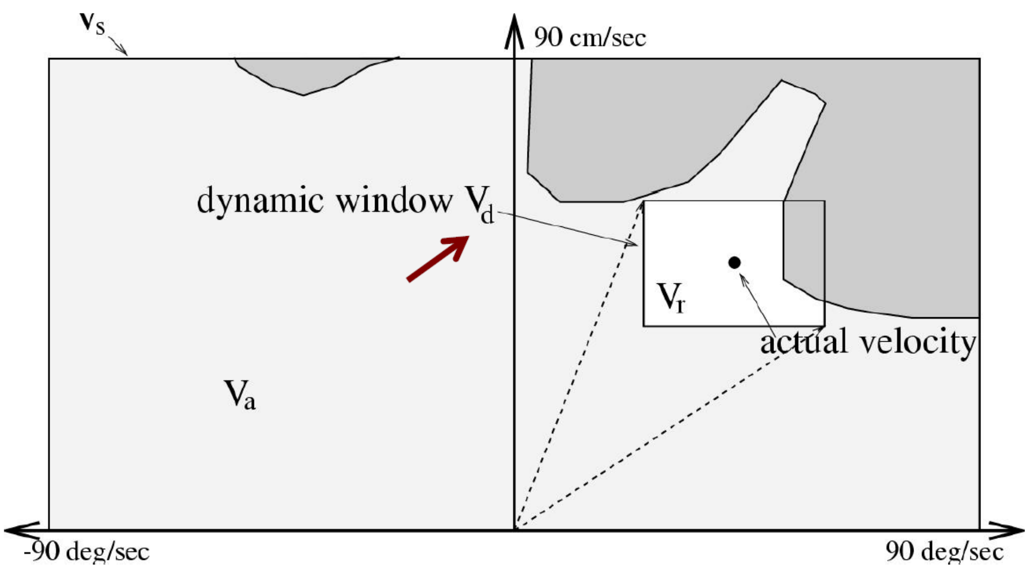
$$V_a = \{(v,\omega) \mid v \leq \sqrt{2\mathrm{dist}(v,\omega)a_{trans}} \;\wedge\; \omega \leq \sqrt{2\mathrm{dist}(v,\omega)a_{rot}}\}$$



Motion Planning

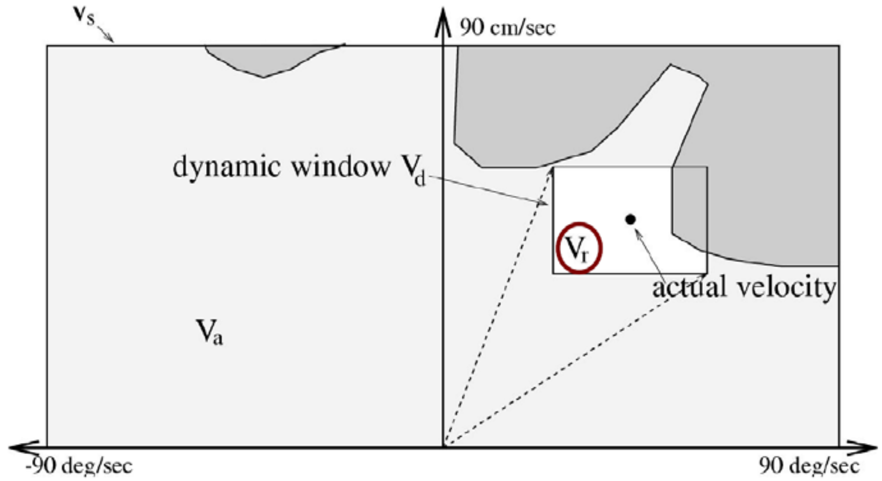
- Dynamic Window Approach
  - Reachable Velocities
    - Speeds that are reachable by acceleration

$$V_d = \{(v,\omega) \mid v \in [v - a_{trans}t, v + a_{trans}t] \wedge \omega \in [\omega - a_{rot}t, \omega + a_{rot}t]\}$$



Motion Planning

- Dynamic Window Approach
  - DWA Search Space



- V_s = all possible speeds of the robot.
- V_a = obstacle free area.
- V_d = speeds reachable within a certain time frame based on possible accelerations.

$$V_r = V_s \cap V_a \cap V_d$$

Motion Planning

- Dynamic Window Approach
  - How to choose <v,ω>?
  - Steering commands are chosen by a heuristic navigation function.
  - This function tries to minimize the travel-time by "**driving fast** into the right direction."

  - Navigation Function
    - Heuristic navigation function.
    - Planning restricted to <x,y>-space.
    - No planning in the velocity space.

Follows grid based path computed by A*

$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Maximizes velocity

Considers cost to reach the goal

Goal nearness

Motion Planning

- Dynamic Window Approach
  - Reacts quickly.
  - Low computational requirements.
  - Guides a robot along a collision-free path.
  - Successfully used in a lot of real-world scenarios.
  - Resulting trajectories sometimes sub-optimal.
  - Local minima might prevent the robot from reaching the goal location.

Motion Planning

- Motion Planning Formulation
  - Problem of motion planning
    - Given:
      - A **start** pose of the robot
      - A desired **goal** pose
      - A geometric description of the **robot**
      - A geometric representation of the **environment**
    - Find a path that moves the robot gradually from **start** to **goal** while **never touching** any obstacle

Motion Planning

- Configuration Space
  - Definition
    - Although the motion planning problem is defined in the regular world, it lives in another space: the **configuration space**
    - A robot configuration q is a specification of the positions of all robot points relative to a fixed coordinate system
    - Usually a configuration is expressed as a **vector of positions** and **orientations**

Motion Planning

- Configuration Space

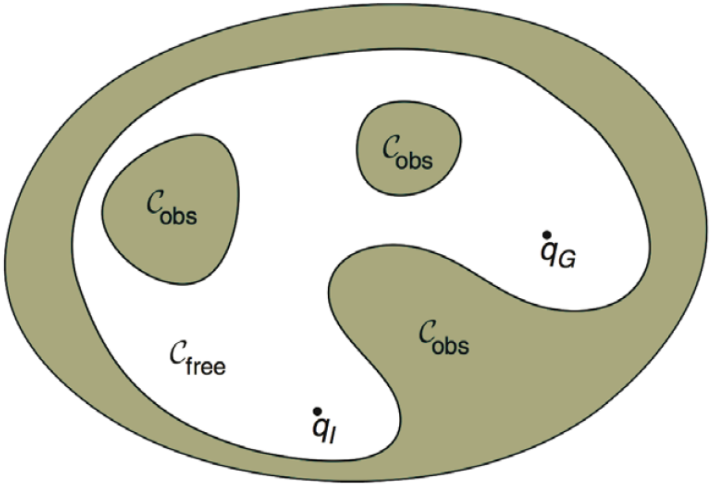
  - **Free space** and **obstacle region**
  - With $\mathcal{W} = \mathbb{R}^m$ being the work space, $\mathcal{O} \in \mathcal{W}$ the set of obstacles, $\mathcal{A}(q)$ the robot in configuration $q \in \mathcal{C}$

$$
\begin{aligned}
\mathcal{C}_{free} &= \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O} = \emptyset\} \\
\mathcal{C}_{obs} &= \mathcal{C} / \mathcal{C}_{free}
\end{aligned}
$$

  - We further define

q_I : start configuration

q_G : goal configuration



Motion Planning

■ Configuration Space

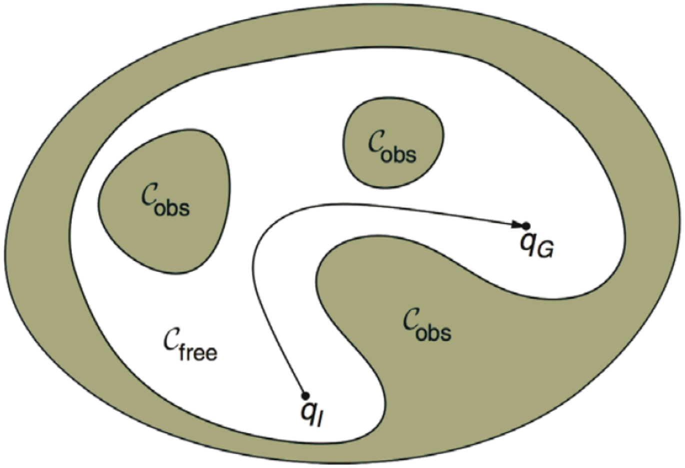
Then, motion planning amounts to

- Finding a continuous path

  $$\tau : [0, 1] \rightarrow \mathcal{C}_{free}$$

  with $\tau(0) = q_I \; , \; \tau(1) = q_G$

- Given this setting, we can do planning with the robot being a **point in C-space!**



Motion Planning

- Configuration Space
  - C-Space Discretization
    - Continuous terrain needs to be discretized for path planning
    - There are **two general approaches** to discretize C-spaces:
      - Combinatorial planning
        - Characterizes C_{free} explicitly by capturing the connectivity of C_{free} into a graph and finds solutions using search
      - Sampling-based planning
        - Uses collision-detection to probe and incrementally search the C-space for a solution

Motion Planning

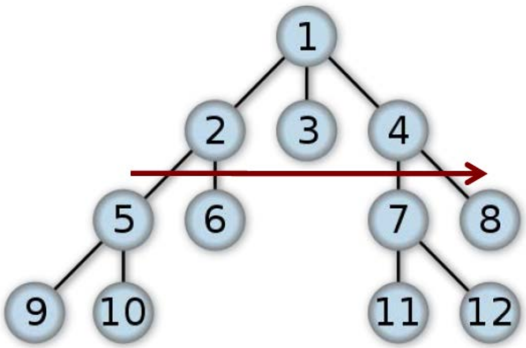
- Configuration Space
  - Search
    - finding a sequence of actions (a path) that leads to desirable states (a goal)
    - **Uninformed** search:
      - no further information about the domain (“blind search”)
      - The only thing one can do is to expand nodes differently
      - Example algorithms: breadth-first, uniform-cost, depth-first, bidirectional, etc.
    - **Informed** search:
      - further information about the domain through heuristics
      - Capability to say that a node is “more promising” than another node
      - Example algorithms: greedy best-first search, A*, many variants of A*, D*, etc.

Motion Planning

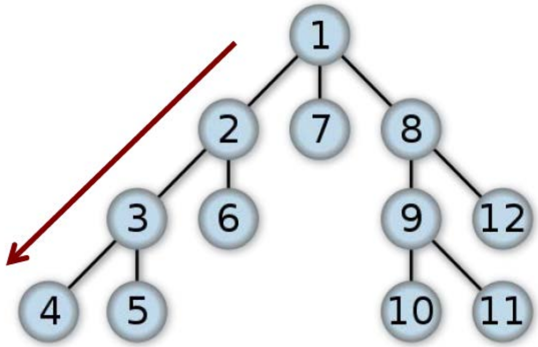
- Configuration Space
  - Performance of a search algorithm
    - **Completeness**
      - does the algorithm find a solution when there is one?
    - **Optimality**
      - is the solution the best one of all possible solutions in terms of path cost?
    - **Time complexity**
      - how long does it take to find a solution?
    - **Space complexity**
      - how much memory is needed to perform the search?

Motion Planning

- Configuration Space
  - Uninformed Search
    - **Breadth-first**
      - Complete
      - Optimal if action costs equal

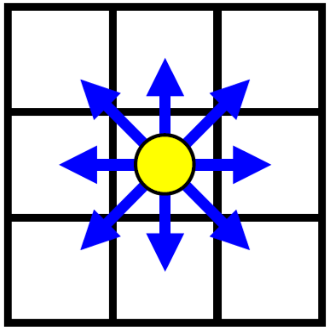


    - **Depth-first**
      - Not complete in infinite spaces
      - Not optimal



Motion Planning

- Configuration Space
  - Informed Search: A*
    - What about using A* to plan the path of a robot?
    - Finds the shortest path
    - Requires a graph structure
    - Limited number of edges
    - In robotics: planning on a 2d occupancy grid map



    - Minimize the Estimated Path Costs

$$f(n) = g(n) + h(n)$$

      - g(n) = actual cost from the initial state to n.
      - h(n) = estimated cost from n to the next goal.