Cognitive Robotics

11. Path Planning and Collision Avoidance

AbdElMoniem Bayoumi, PhD

Fall 2022

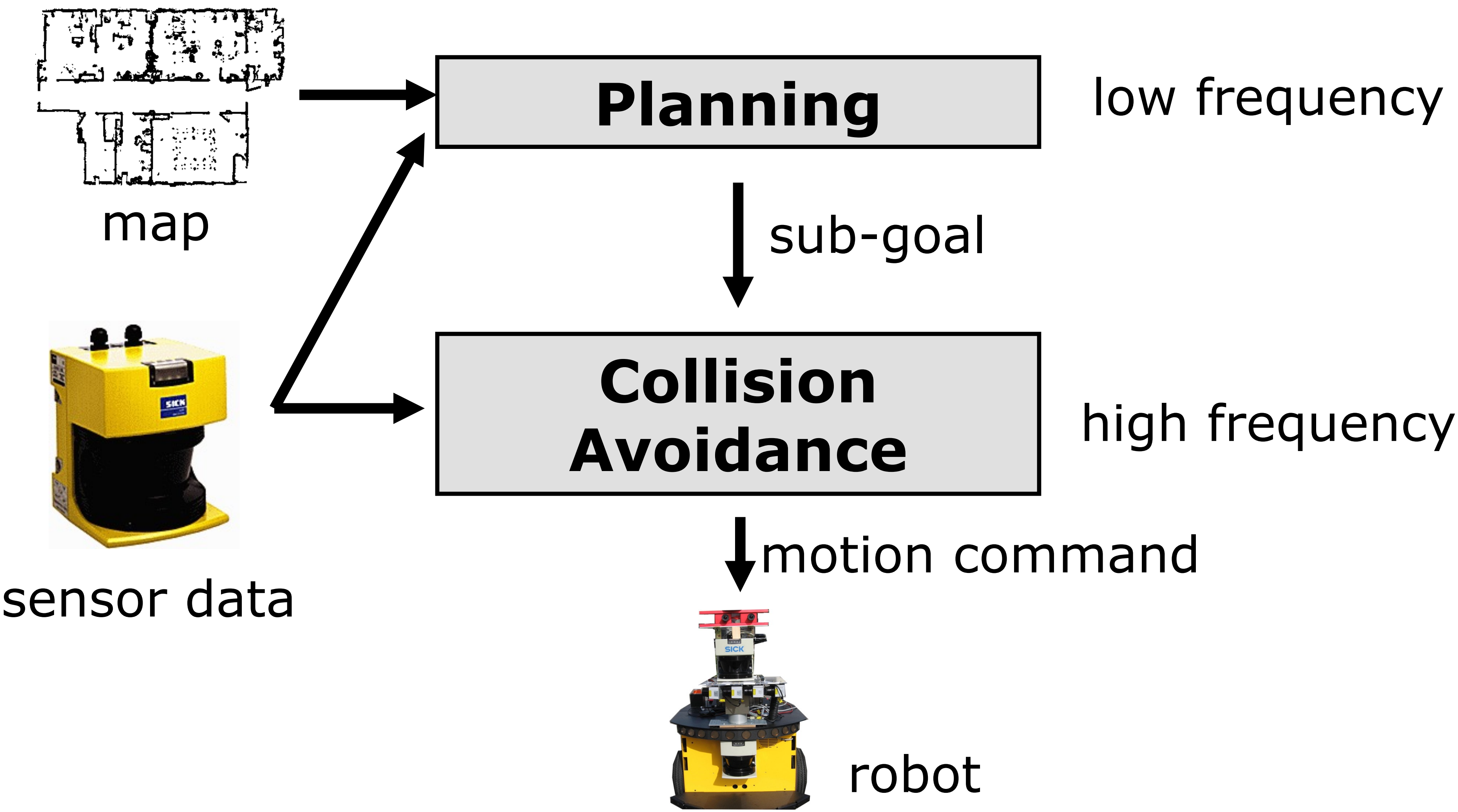
Today's Lecture

- So far:
  - State estimation
  - Localization
  - Mapping
- Today: action selection for navigation
  - Global path planning
  - Local collision-avoidance

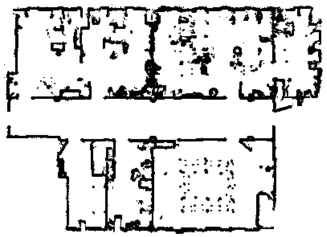
Motion Planning: Requirements

- Collision-free trajectory from the current robot pose to a given goal pose in a map of the environment
- The robot should reach the goal location as fast as possible
- The robot must react to unforeseen obstacles fast and reliably

Classic Two-Layered Architecture



Classic Two-Layered Architecture



map

sensor data

Planning

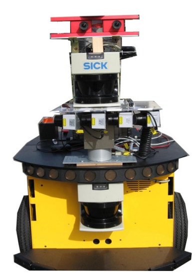
low frequency

sub-goal

Collision Avoidance

high frequency

motion command



robot

today: concentration on wheeled robots

Dynamic Window Approach

- Determine collision-free trajectories using geometric operations
- Motion commands: translational and rotational velocities v and ω

$$r = \left| \frac{v}{\omega} \right|$$

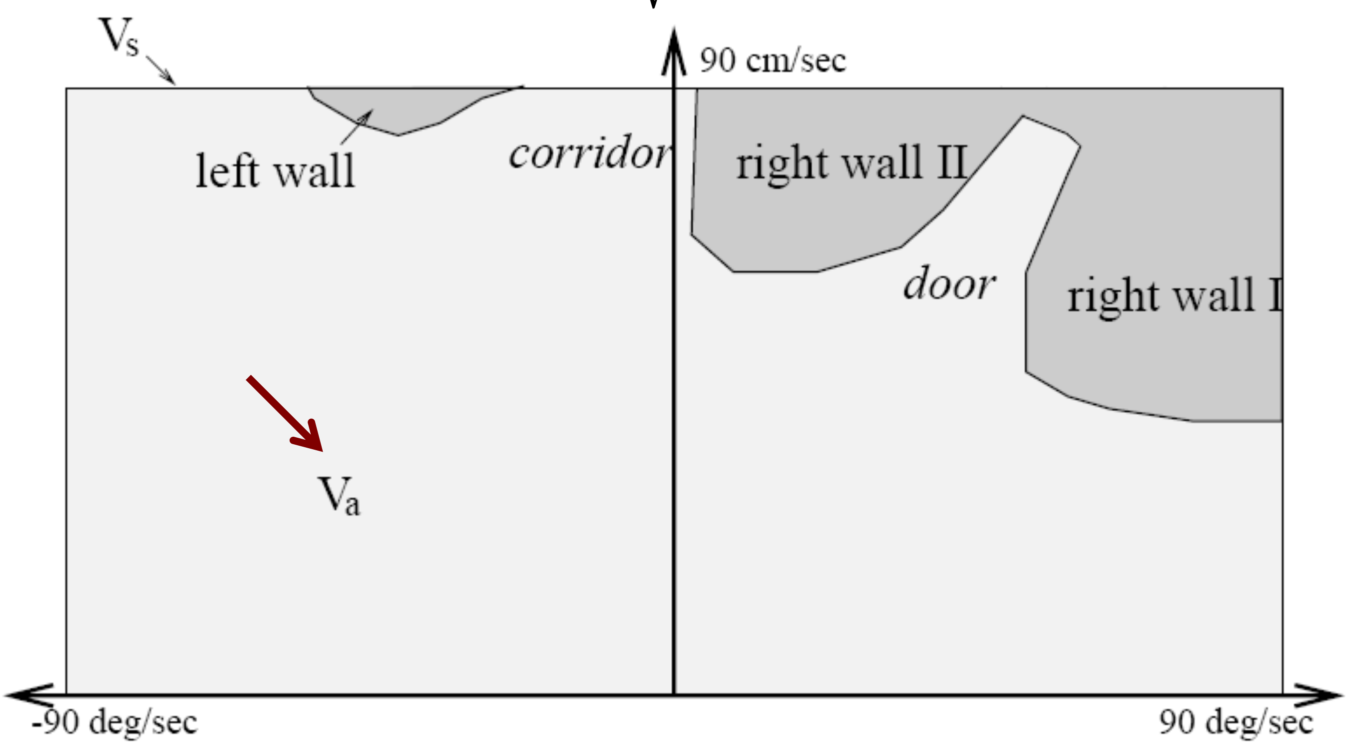
- Assumption: piecewise constant velocities within short time intervals, robot moves on circular arcs
- Question: Which (v,ω) are admissible and reachable?

[Simmons, 96], [Fox et al., 97], [Brock & Khatib, 99]

Admissible Velocities

A velocity is **admissible** if the robot is able to stop before colliding with an obstacle

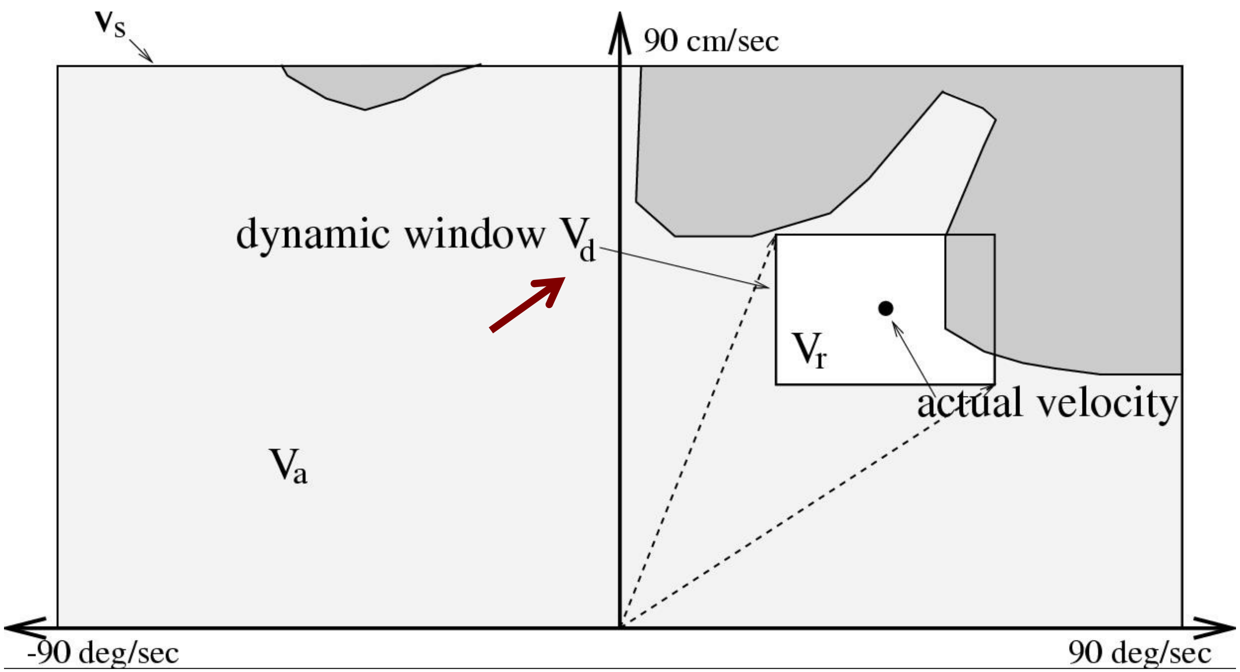
$$V_a = \{(v, \omega) \mid v \leq \sqrt{2\text{dist}(v,\omega)a_{trans}} \quad \wedge \\ \omega \leq \sqrt{2\text{dist}(v,\omega)a_{rot}}\}$$



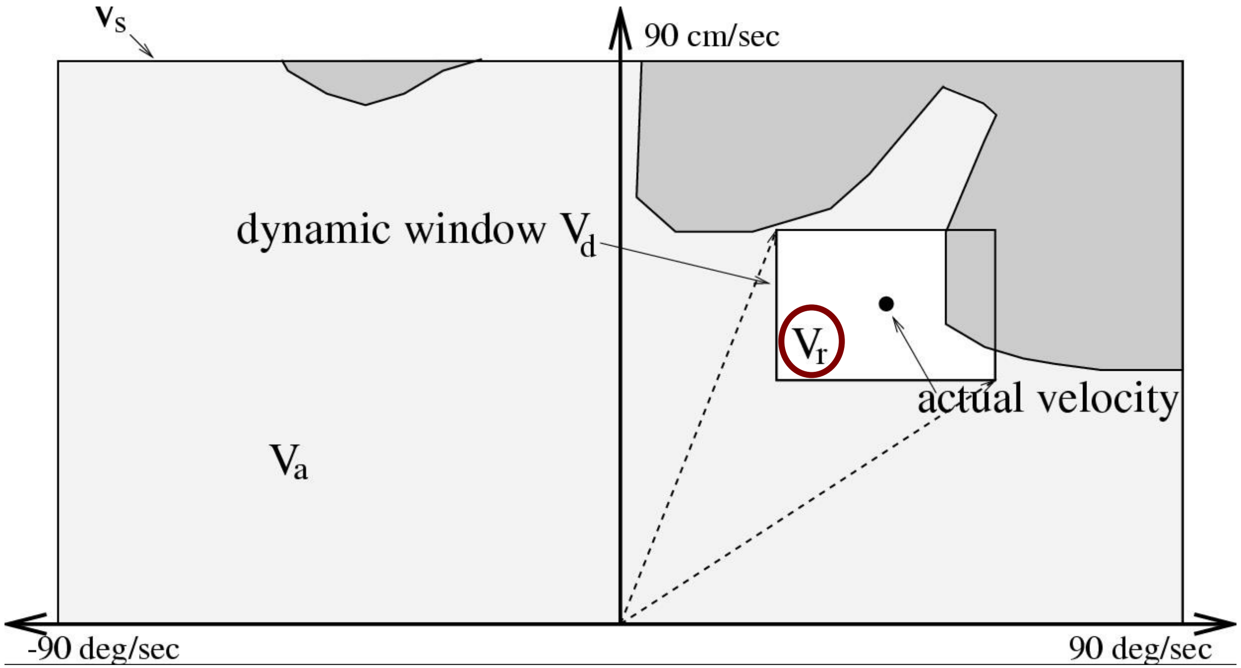
Reachable Velocities

Velocities that are **reachable** by acceleration within the time period t:

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



DWA Search Space



V_s = all possible velocities of the robot

V_a = obstacle free area (light grey)

V_d = velocities reachable within a certain time frame based on possible accelerations

Search space: $V_r = V_s \bigcap V_a \bigcap V_d$

Dynamic Window Approach

- How to choose <v,ω>?
- Use a **heuristic navigation function**
- Try to minimize travel time according to the principle: "**driving fast** in the **right direction**"

Dynamic Window Approach

- Heuristic navigation function
- Evaluation using <x,y>-space

Navigation function: [Brock & Khatib, 99]

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

Dynamic Window Approach

- Heuristic navigation function
- Evaluation using <x,y>-space

Navigation function: [Brock & Khatib, 99]

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

Maximizes velocity

Dynamic Window Approach

- Heuristic navigation function
- Evaluation using <x,y>-space

Navigation function: [Brock & Khatib, 99]

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

Maximizes velocity

Considers cost to reach the goal and alignment

Dynamic Window Approach

- Heuristic navigation function
- Evaluation using <x,y>-space

Navigation function: [Brock & Khatib, 99]

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

Maximizes velocity

Considers cost to reach the goal and alignment

Considers cost change

Dynamic Window Approach

- Heuristic navigation function
- Evaluation using <x,y>-space

Navigation function:

Goal nearness

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

Maximizes velocity

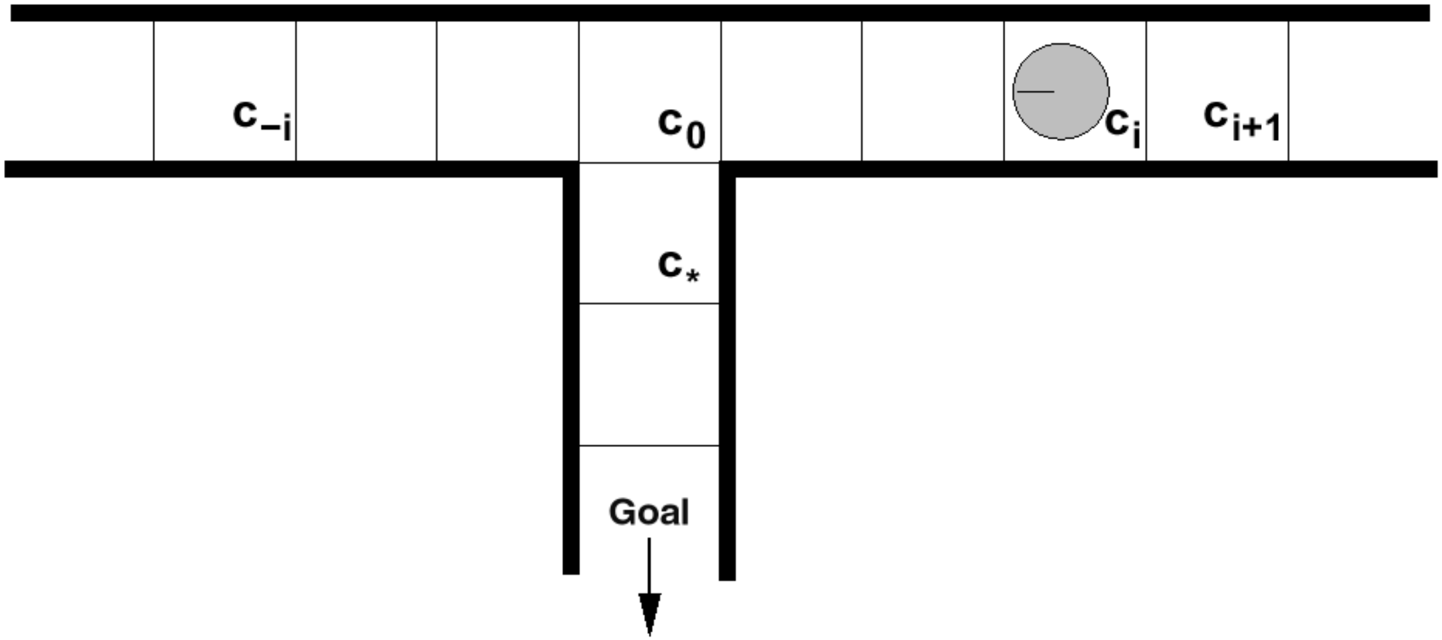
Considers cost to reach the goal and alignment

Considers cost change

Dynamic Window Approach

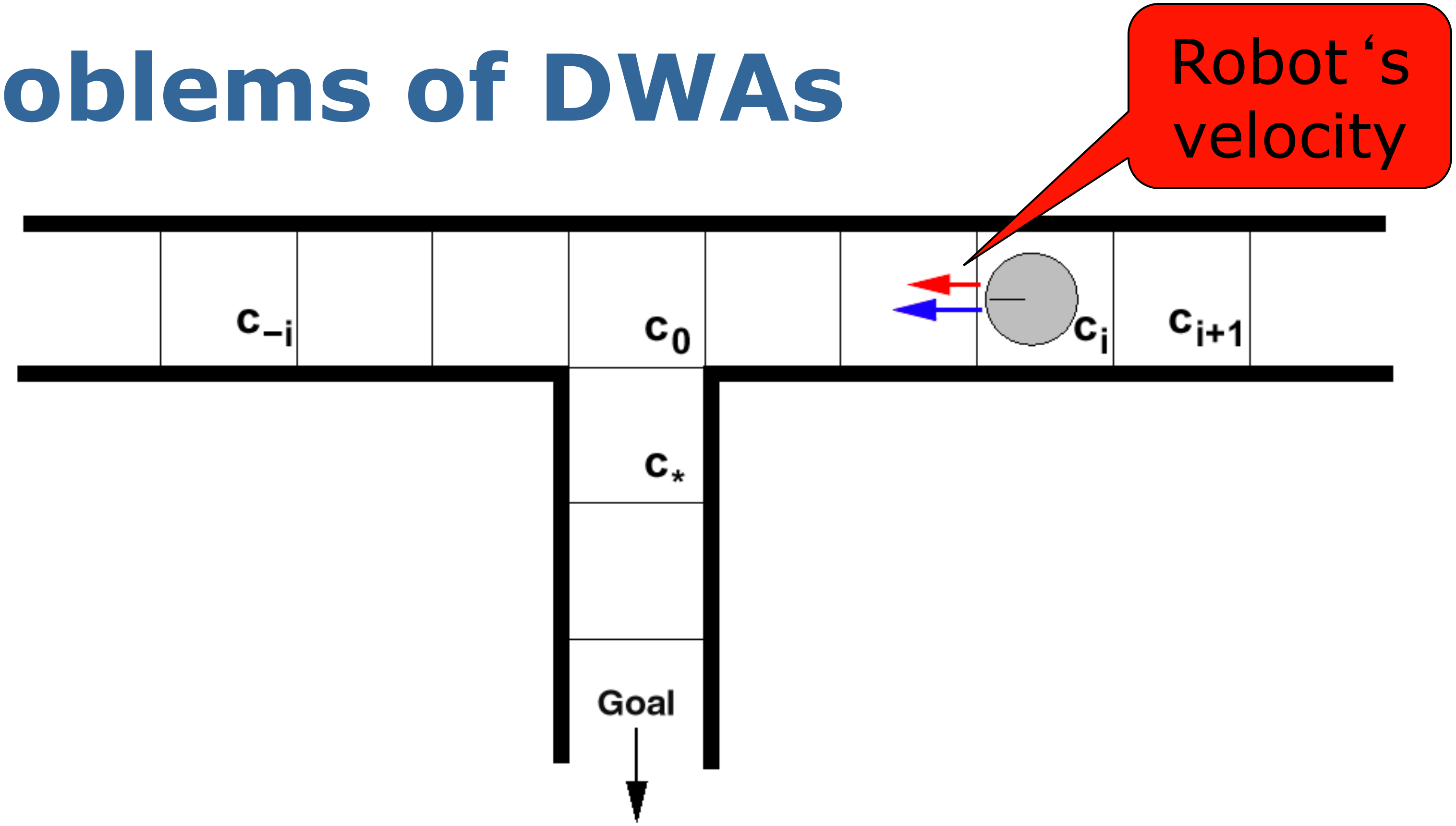
- Enables quick reaction
- Low CPU power requirements
- Guides the robot on a collision-free path
- Successfully used in a lot of real-world systems
- Resulting trajectories sometimes sub-optimal
- Local minima might prevent the robot from reaching the goal location

Problems of DWAs



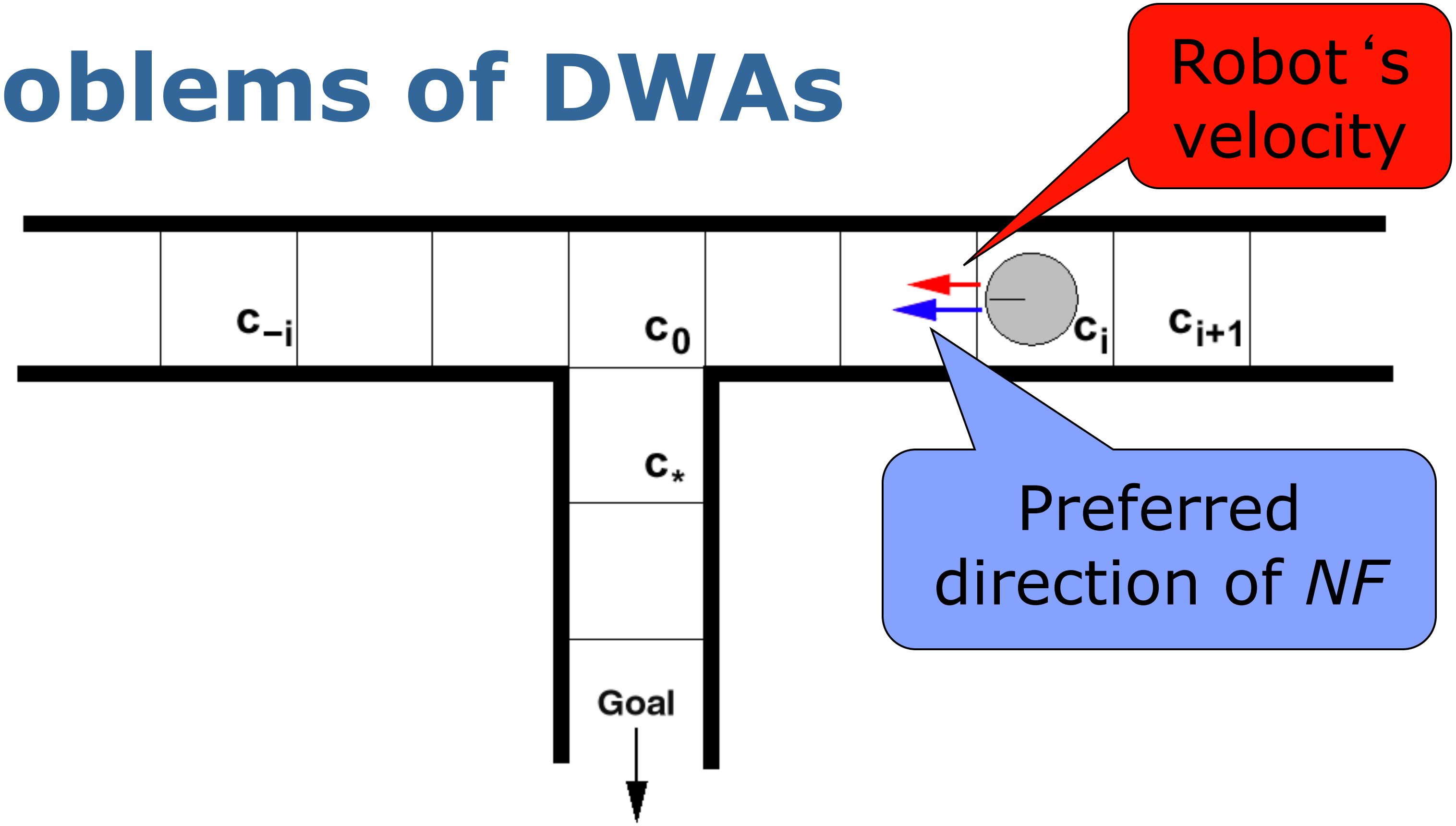
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



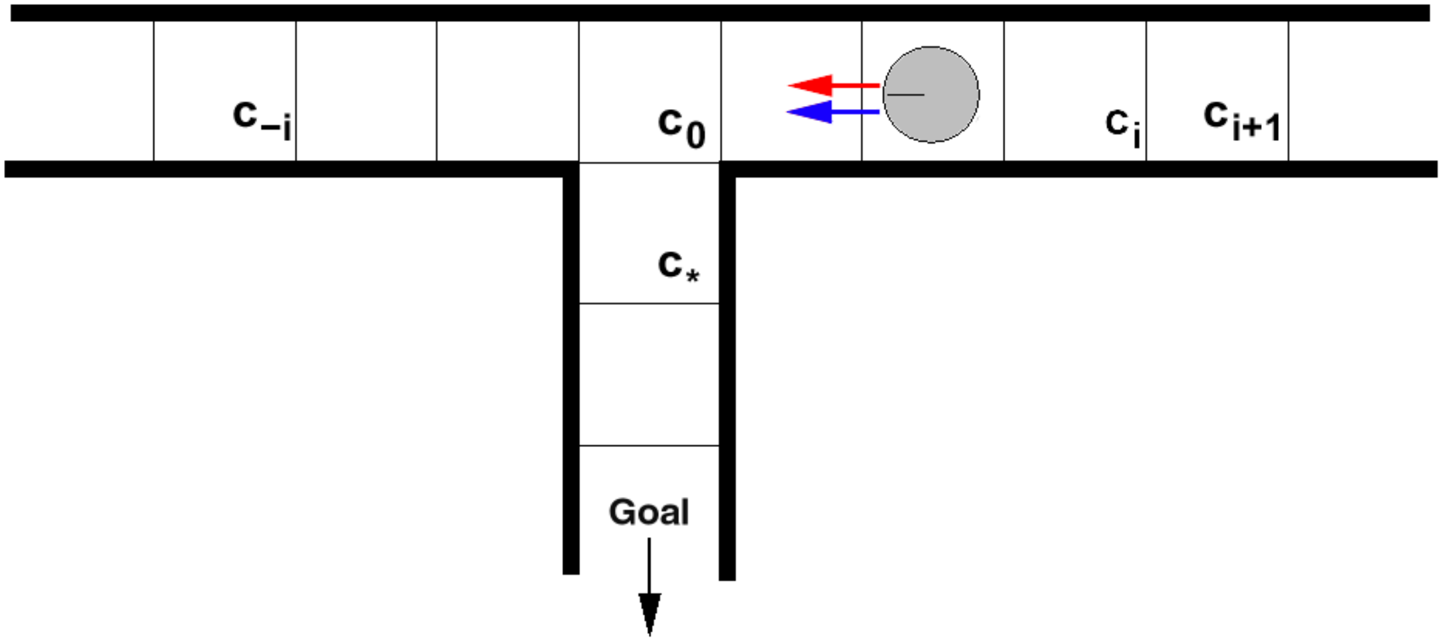
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



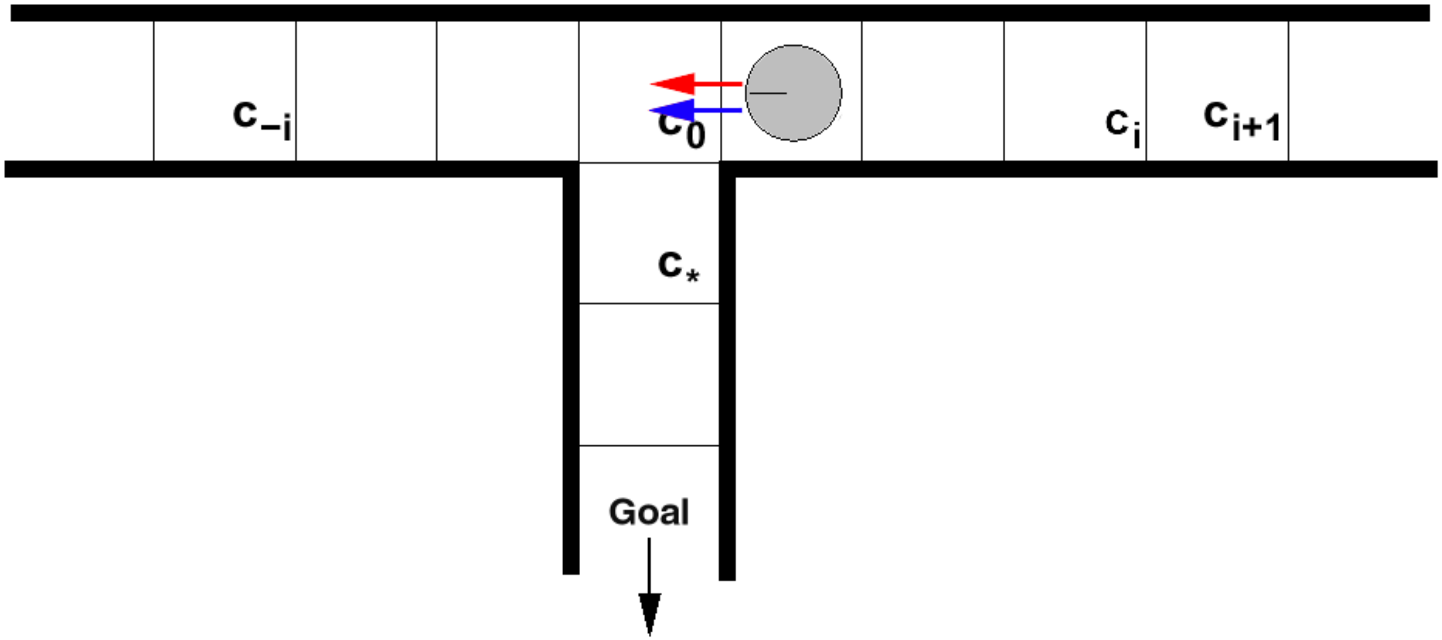
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



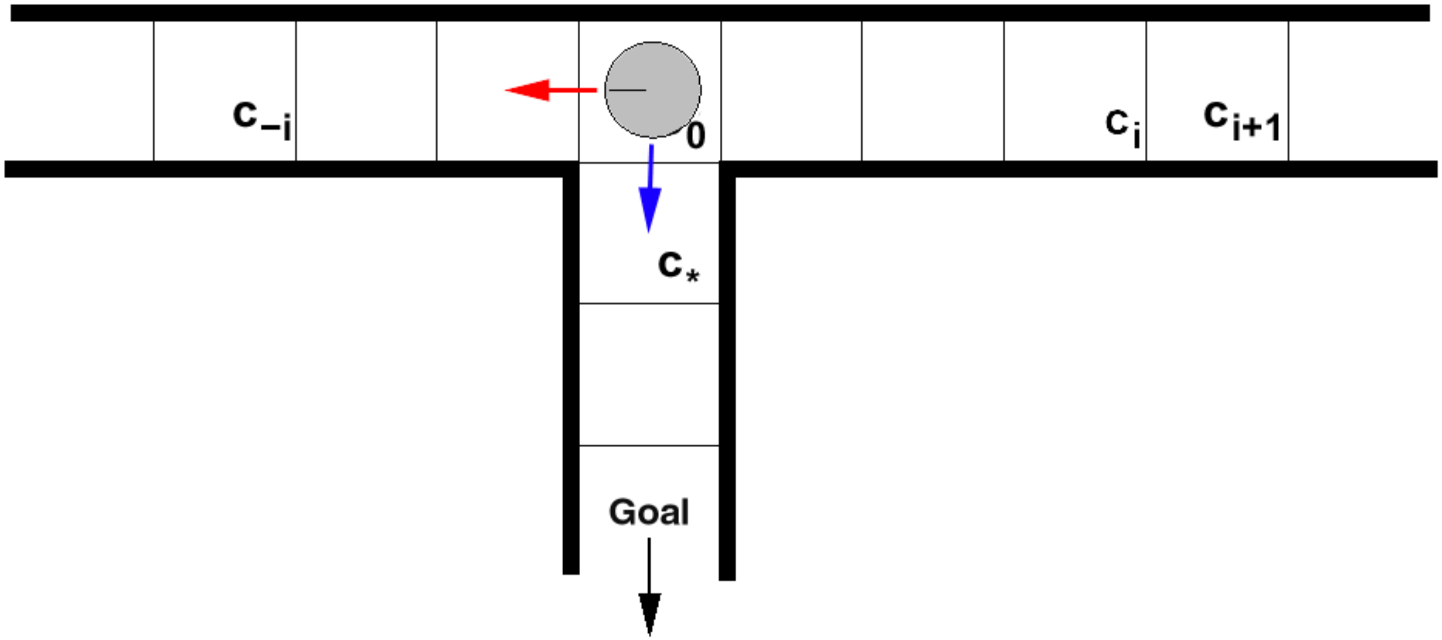
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



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

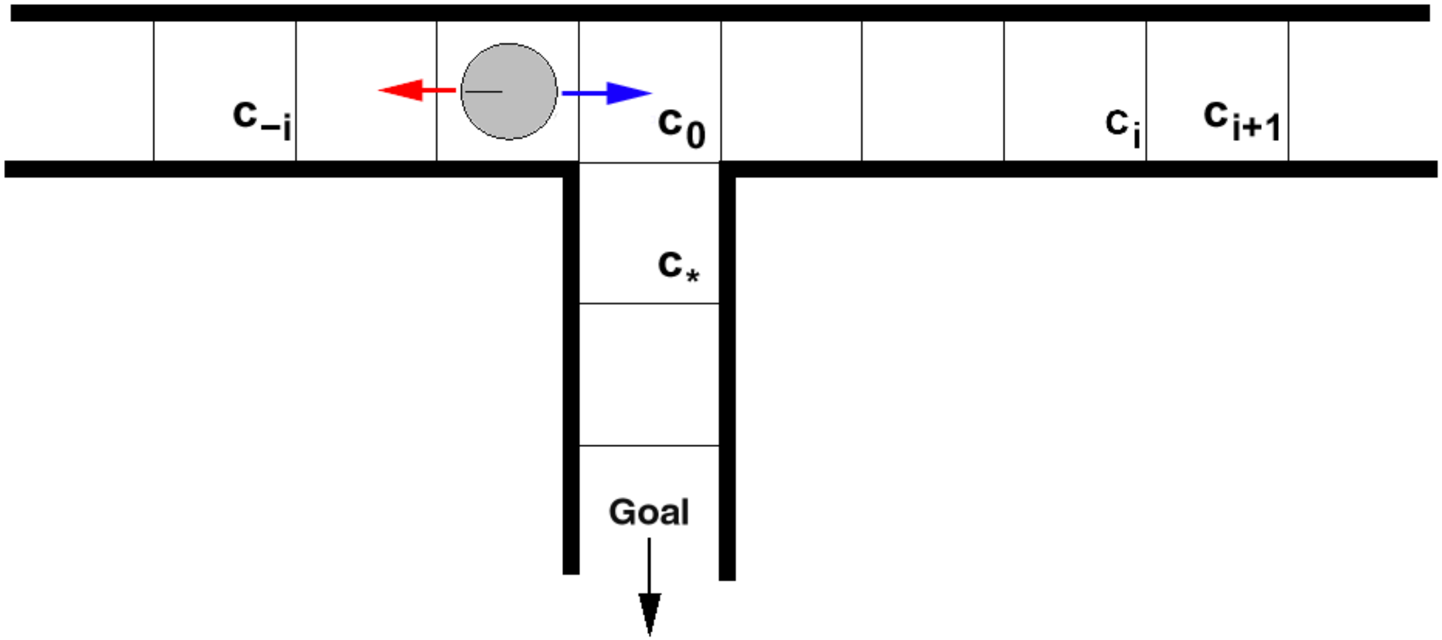
Problems of DWAs



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

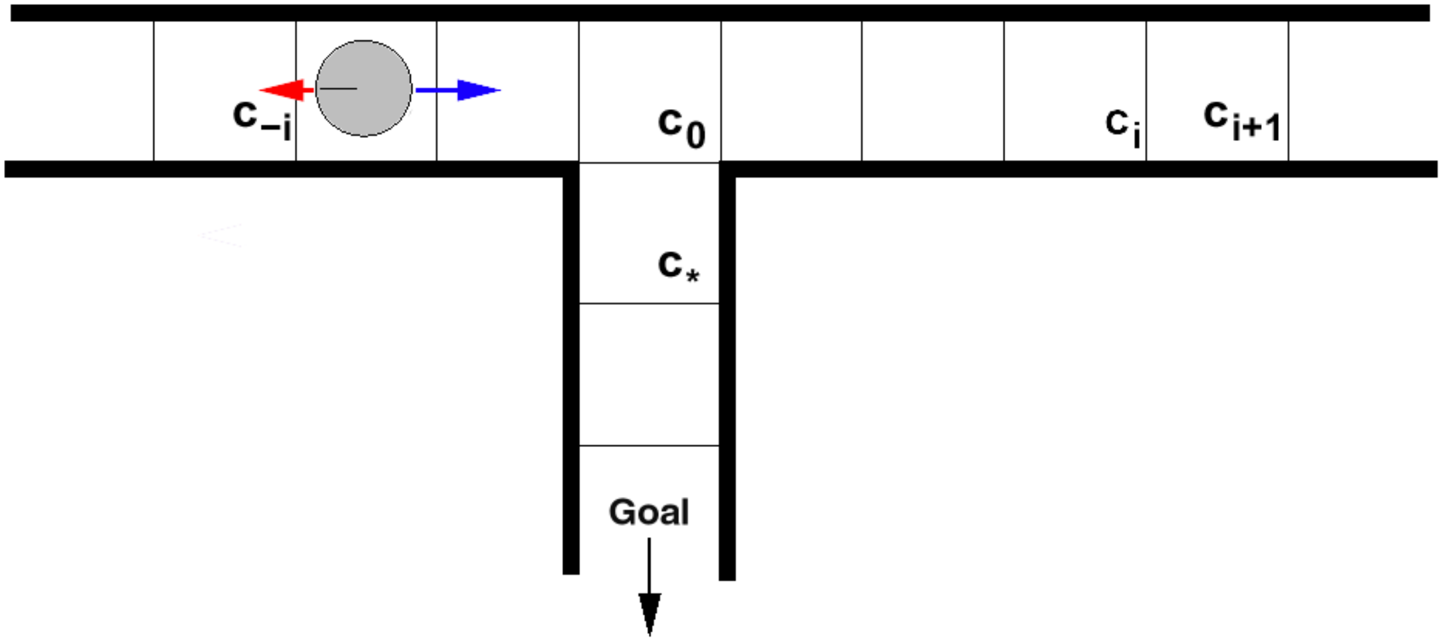
The robot drives too fast at c_0 to enter the corridor facing south

Problems of DWAs



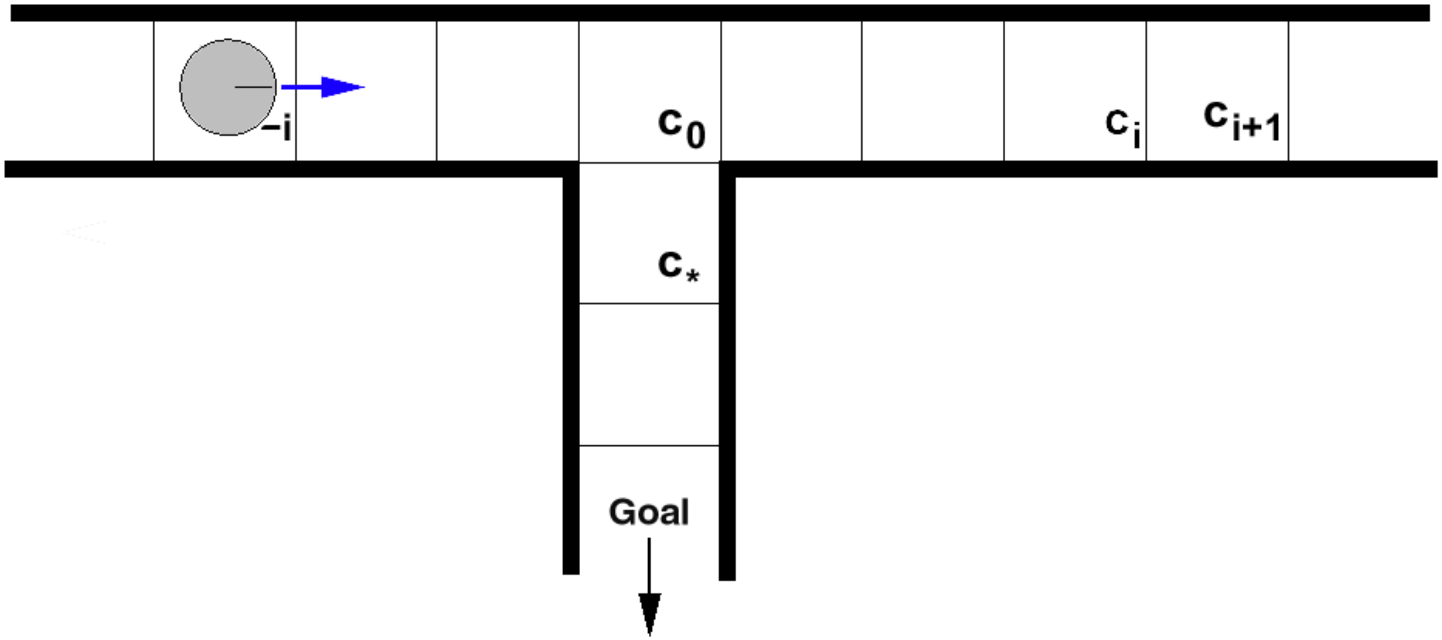
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



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

Problems of DWAs



- Same situation as in the beginning!
- DWAs might not be able to reach the goal location

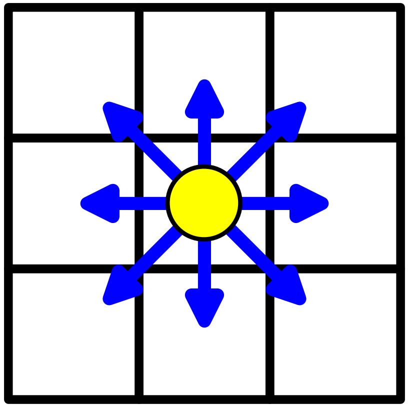
Problems of DWAs

- Typical problem in a real world situation:

- Robot does not slow down early enough to enter the doorway

Robot Path Planning with A*

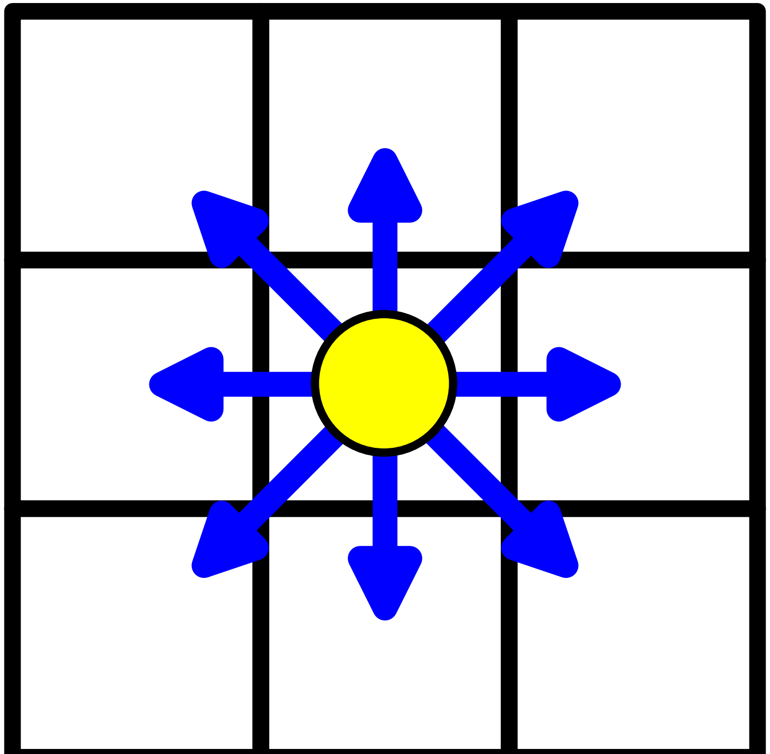
- Find the shortest 2D path in a given static map of the environment with A*
- 2D grid map (=states)
- 8-connected neighborhood (=actions)
- Consider move cost and occupancy value (see exercise)

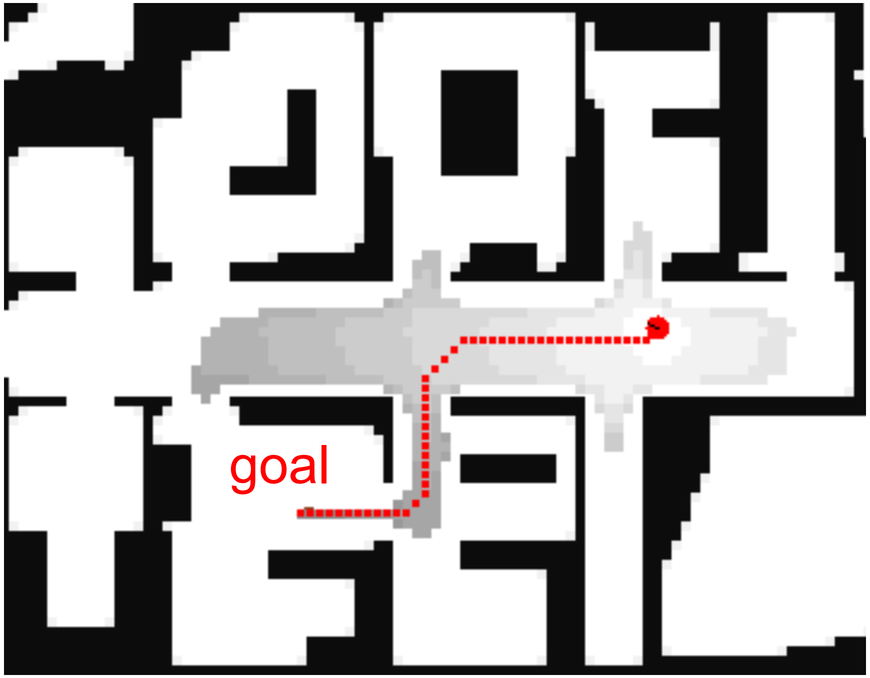


Reminder: A* (AI Lecture)

- $g(n)$: actual cost from the initial state to n
- $h(n)$: estimated cost from n to the goal
- $f(n) = g(n) + h(n)$: estimated cost of the cheapest solution through n
- Let h*(n) be the actual cost of the optimal path from n to the goal
- h is admissible if it holds for all n:
  $h(n) \leq h^*(n)$
- A* yields the optimal path if h is admissible
  (the straight-line distance in the Euclidean space is admissible)

Example: Path Planning with A* on a Grid Map





Deterministic Value Iteration

- To compute the shortest path from every state to one goal state, use value iteration (similar to Dijkstra's Algorithm)
- Yields the optimal heuristics for A*, which is needed for re-planning, e.g., in case of non-static obstacles, also for localization errors

Problems when Using A* on Grid Maps

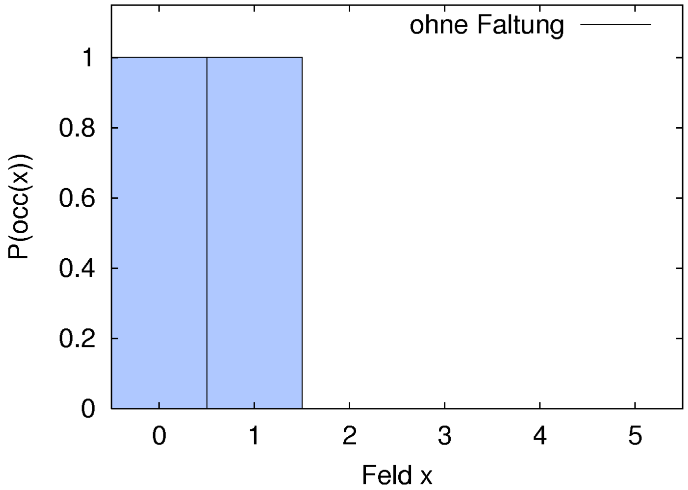
- Moving on the shortest path often guides the robot on a trajectory close to obstacles
- What if the robot is slightly delocalized?
- What if commands are only inaccurately executed?

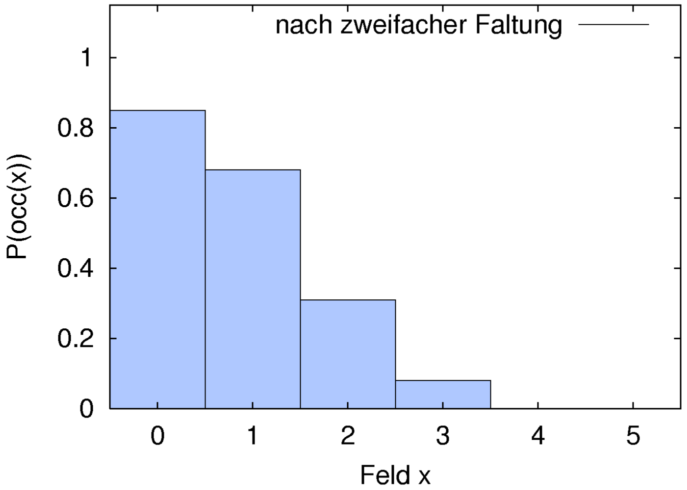
Convolution of the Grid Map

- Convolution, e.g, with a Gaussian kernel to "blur" the map
- Obstacles are assumed to be bigger than in reality
- Perform an A* search on such a convolved map
- As a result, the robot increases its distance to obstacles and moves on a short path

Example: Map Convolution

- 1D environment, cells $c_0, \ldots, c_5$





- Cells before and after 2 convolution runs

Convolution

- The convolution of an occupancy map is defined as:

$$P(occ_{x_i,y}) = \frac{1}{4} \cdot P(occ_{x_{i-1},y}) + \frac{1}{2} \cdot P(occ_{x_i,y}) + \frac{1}{4} \cdot P(occ_{x_{i+1},y})$$

$$P(occ_{x_0,y}) = \frac{2}{3} \cdot P(occ_{x_0,y}) + \frac{1}{3} \cdot P(occ_{x_1,y})$$

$$P(occ_{x_{n-1},y}) = \frac{1}{3} \cdot P(occ_{x_{n-2},y}) + \frac{2}{3} \cdot P(occ_{x_{n-1},y})$$

- This is done for each row and each column of the map
- Named "Gaussian blur"

A* in Convolved Maps

- Cells with higher occupancy value are avoided by the robot
- Thus, the robot keeps distance to obstacles
- Fast and quite effective for path planning in real-world environments

5D Planning – An Alternative to the Two-Layered Architecture

- **A* search in the full 5D <x,y,θ,v,ω>-configuration space**
- Considers the robot's kinematic constraints directly
- Generates a sequence of steering commands to reach the goal location
- Considers driving time and distance to obstacles in the cost function

The Search Space (1)

- What is a state in this space?
  $<x,y,\theta,v,\omega>$ = current pose and velocities of the robot

- How does a state transition look like?
  $<x_1,y_1,\theta_1,v_1,\omega_1> \Rightarrow <x_2,y_2,\theta_2,v_2,\omega_2>$

  with motion command (v_2,ω_2) and
  $|v_1-v_2| < a_v$, $|\omega_1-\omega_2| < a_\omega$

- The robot's pose is computed based on the motion equations

The Search Space (2)

Idea: Search in the discretized $\langle x,y,\theta,v,\omega\rangle$-space

Problem: The search space is too huge to be explored within the time constraints (5+ Hz for online motion planning)

Solution: Restrict the full search space

Main Steps of 5D Path Planning

1. Update the grid map based on sensory input

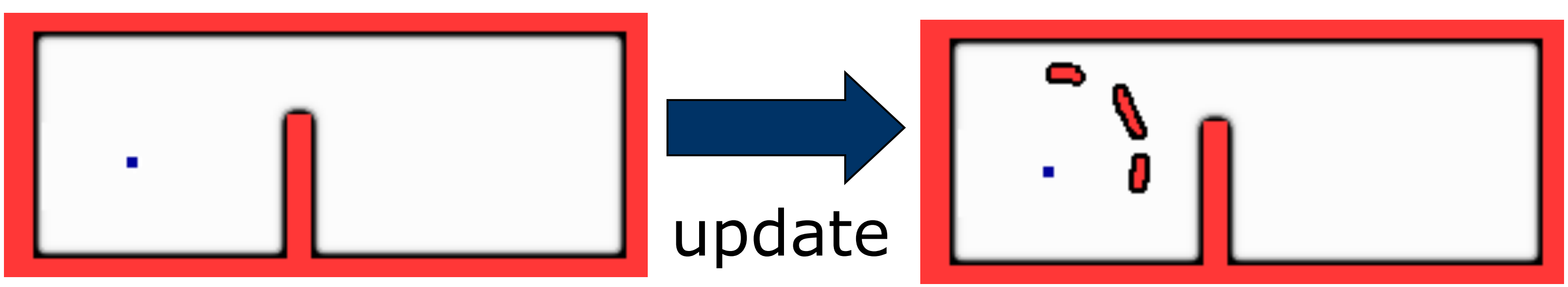
2. Use A^* to find a path in the $<x,y>$-space using the updated grid map

3. Determine a restricted 5D configuration space based on step 2

4. Find a trajectory by planning in this restricted $<x,y,\theta,v,\omega>$-space

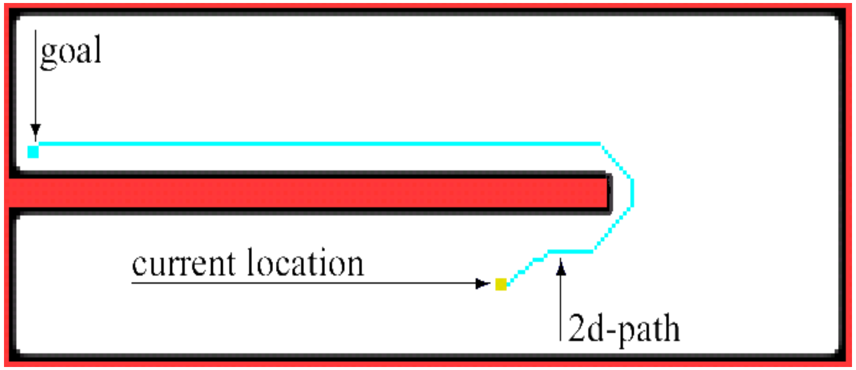
Updating the Grid Map

- Add newly detected obstacles
- Convolve the map to increase security distance



Find a Path in the 2D Space

- Use A^* to search for the optimal path in the 2D grid map
- Use heuristics based on a deterministic value iteration within the static map



Restricting the Search Space

Assumption: The projection of the optimal 5D path onto the <x,y>-space lies close to the optimal 2D path

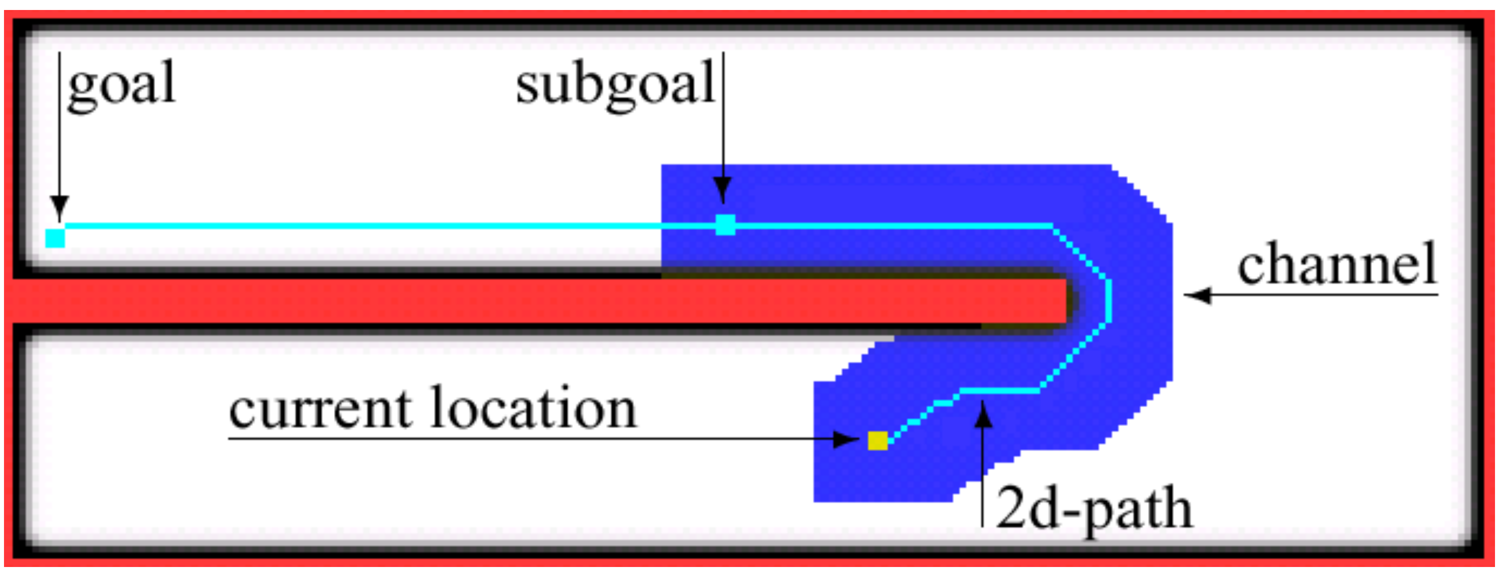
Restricting the Search Space

Assumption: The projection of the optimal 5D path onto the <x,y>-space lies close to the optimal 2D path

Idea: Construct a restricted search space ("channel") based on the 2D path

Space Restriction

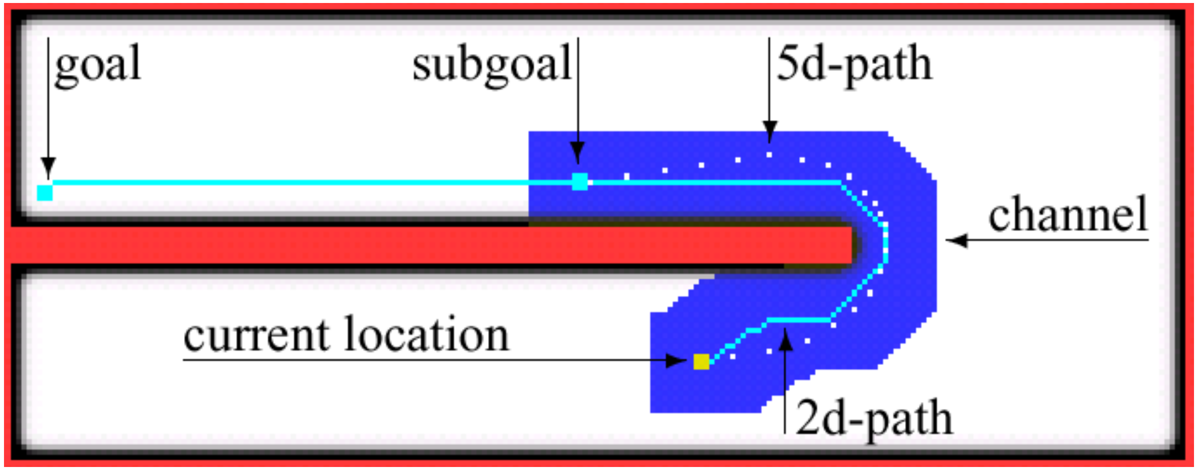
- Resulting search space:
  $<x, y, \theta, v, \omega>$ with $(x,y) \in$ channel
- Choose a sub-goal lying on the 2D path within the channel

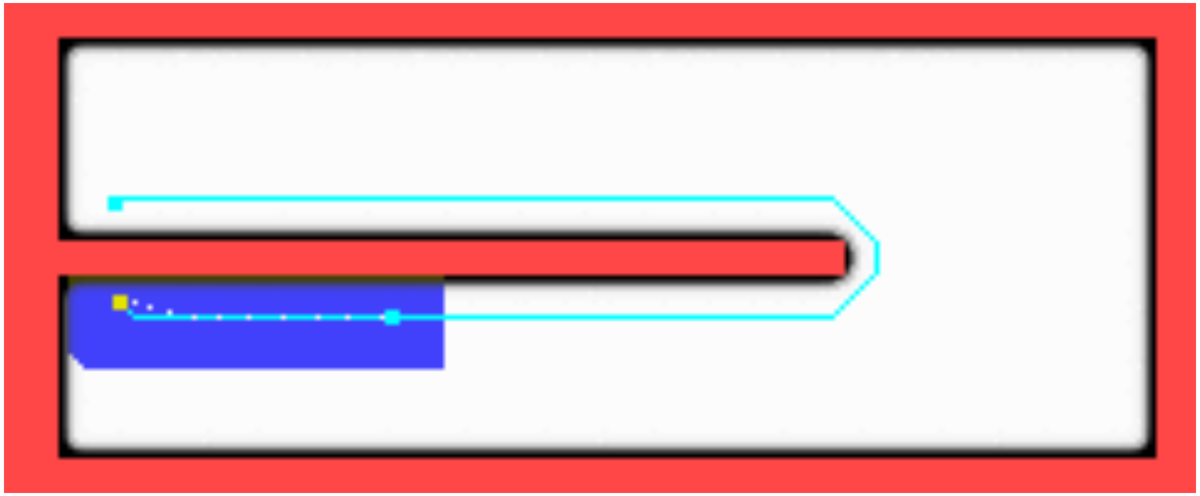


Find a Path in the 5D Space

- Use A^* in the restricted 5D space to find a sequence of steering commands to reach the sub-goal
- Heuristics to estimate cell costs: deterministic 2D value iteration within the channel

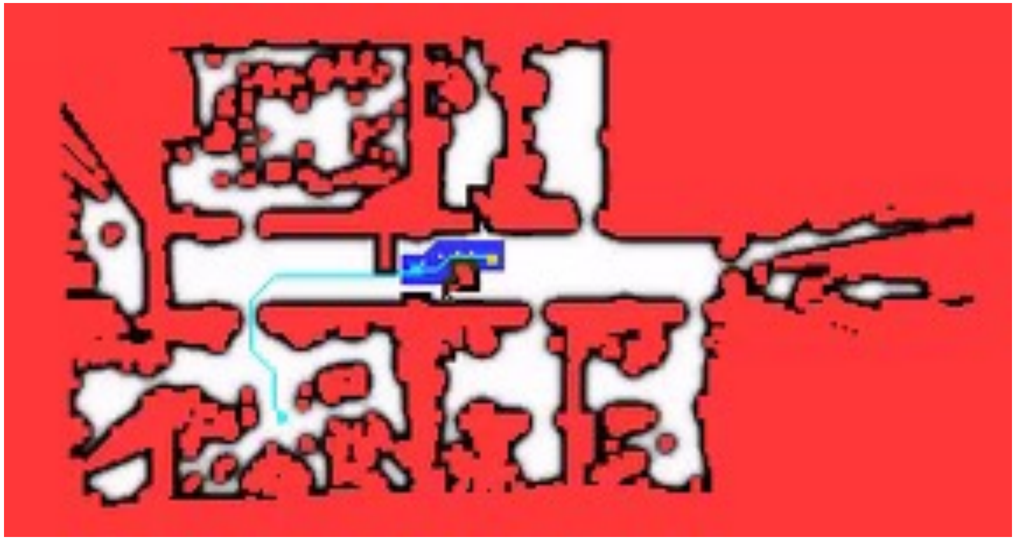
Examples





Example

real robot



planning state

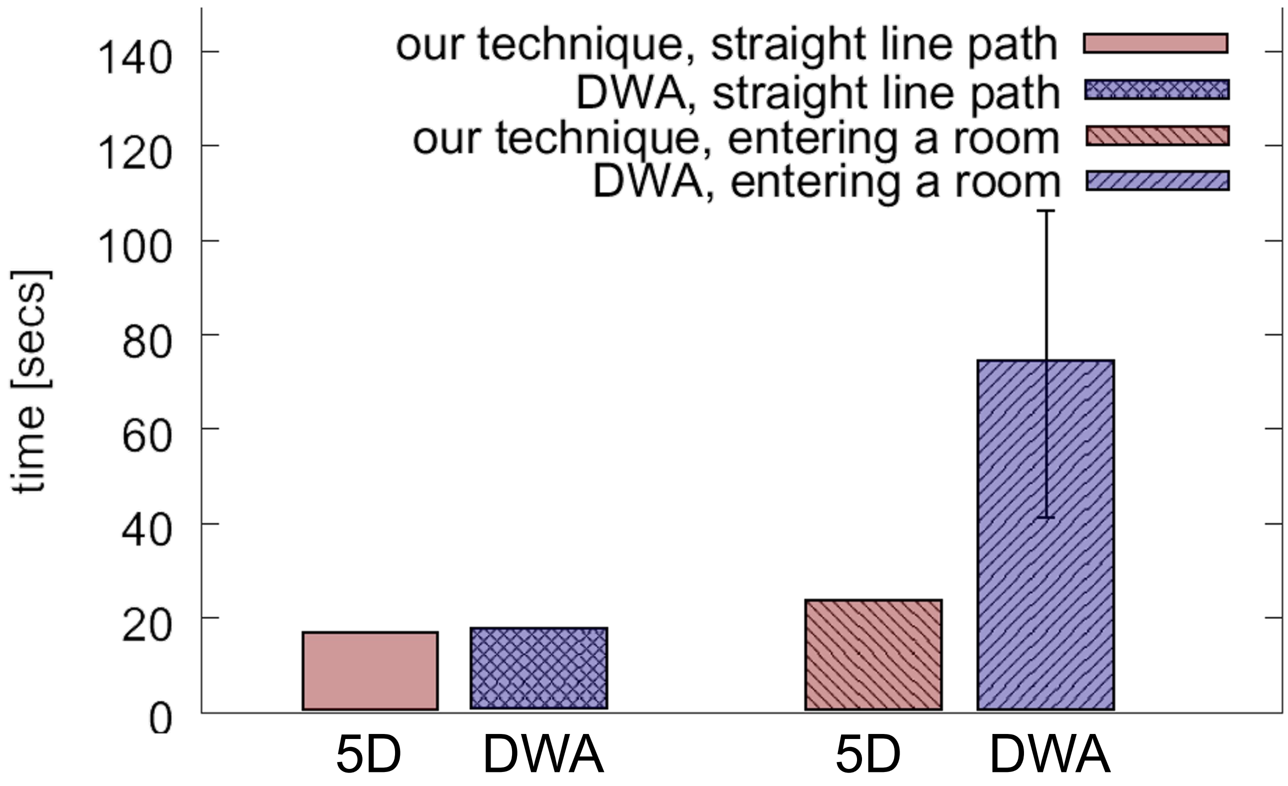
Comparison to the DWA (1)

DWAs often have problems entering narrow passages

DWA planned path

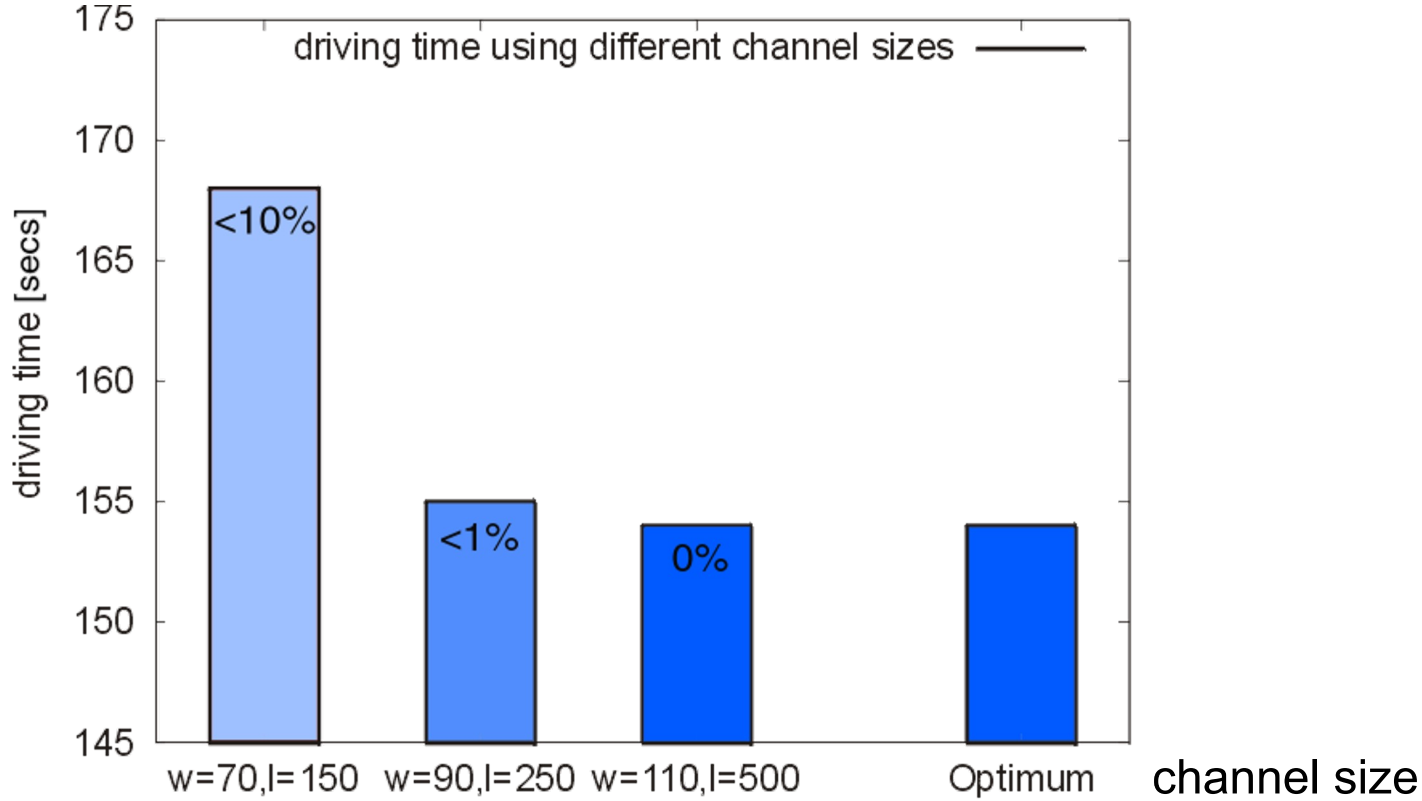
5D approach

Comparison to the DWA (2)



The presented approach results insignificantly faster motion when driving through narrow passages!

Comparison to the Optimum



Channel: with length=5m, width=1.1m

Resulting actions are close to the optimal solution

Summary

- Robust navigation requires path planning and collision avoidance
- Collision avoidance considers the robot's kinematic constraints and generates velocities
- Planning in the 5D space shows better results than the pure DWA in a variety of situations
- Using the 5D approach the quality of the trajectory scales with the computational resources available
- Still DWA often used in practice

Other Types of Robots

- For example, humanoid robots or flying vehicles
- Geometric descriptions of the robots and appropriate world representations needed
- Often, path planning based on A* in a specific action space

Literature

- The Dynamic Window Approach to Collision Avoidance
  Fox, Burgard, and Thrun, 1997
- High-Speed Navigation Using the Global Dynamic Window Approach
  Brock and Khatib, 1999
- An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments
  Stachniss and Burgard, 2002

Acknowledgment

- These slides have been created by Wolfram Burgard, Dieter Fox, Cyrill Stachniss, and Maren Bennewitz