

Introduction to Mobile Robotics

Path Planning and Collision Avoidance

Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras



Motion Planning

Latombe (1991):

“...eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world.”

Goals:

- Collision-free trajectories.
- Robot should reach the goal location as fast as possible.

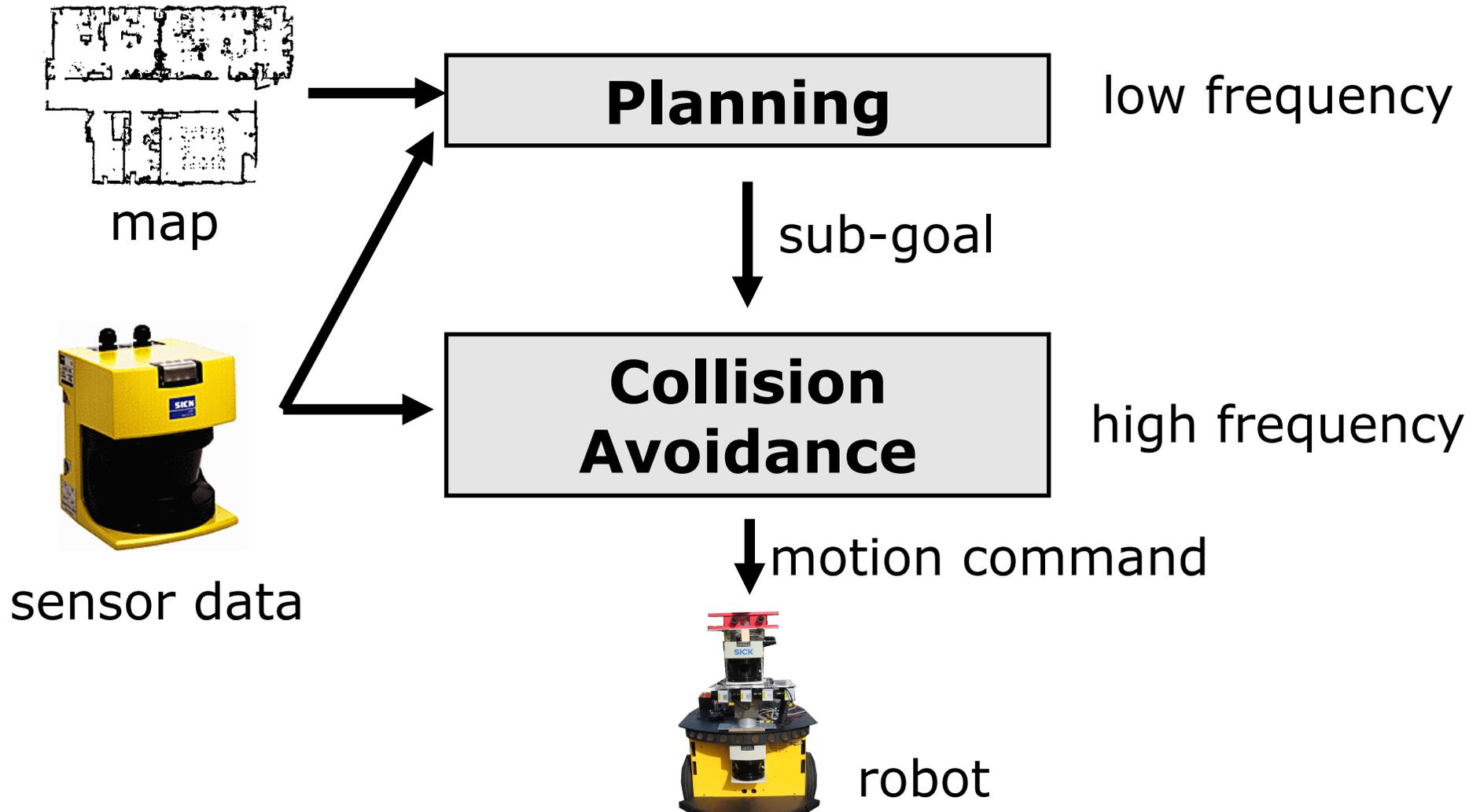
... in Dynamic Environments

- How to react to unforeseen obstacles?
 - efficiency
 - reliability
- Dynamic Window Approaches
[Simmons, 96], [Fox et al., 97], [Brock & Khatib, 99]
- Grid map based planning
[Konolige, 00]
- Nearness Diagram Navigation
[Minguez et al., 2001, 2002]
- Vector-Field-Histogram+
[Ulrich & Borenstein, 98]
- A*, D*, D* Lite, ARA*, ...

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



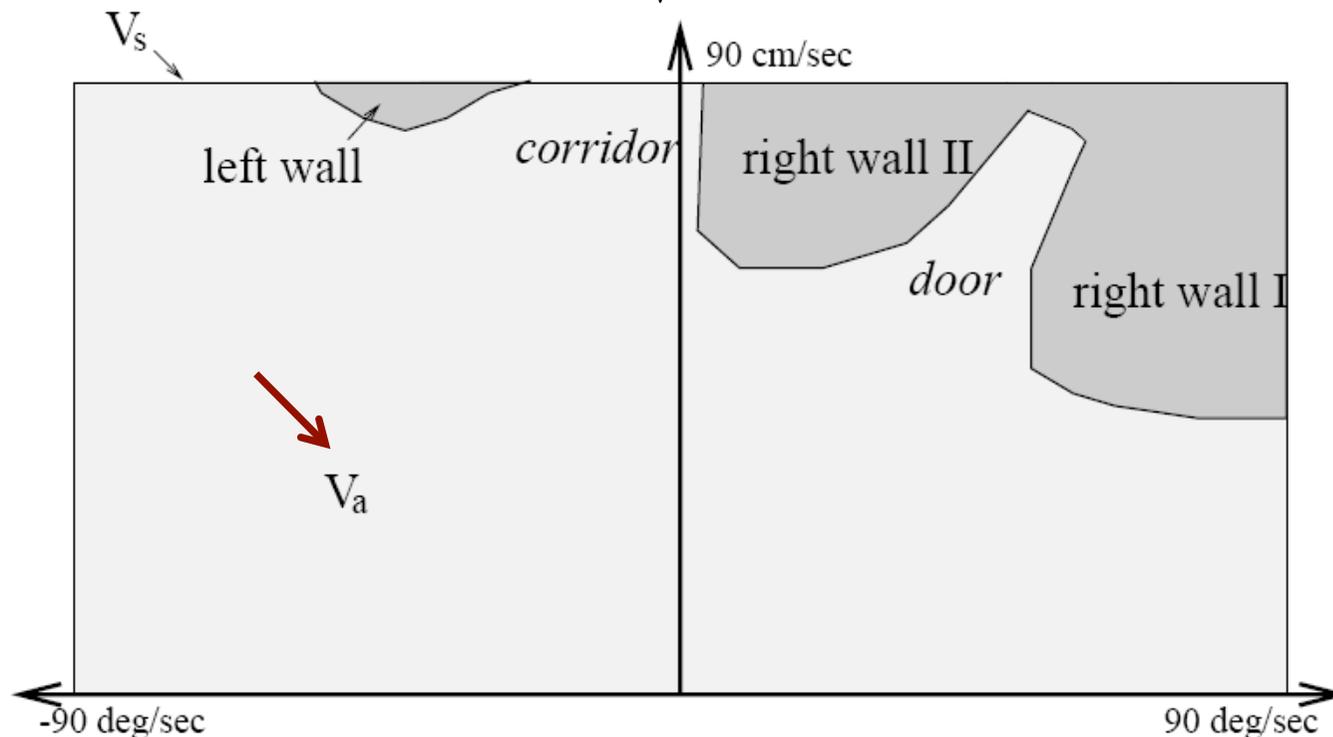
Dynamic Window Approach

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

Admissible Velocities

- Speeds are admissible if the robot would be able to stop before reaching the obstacle

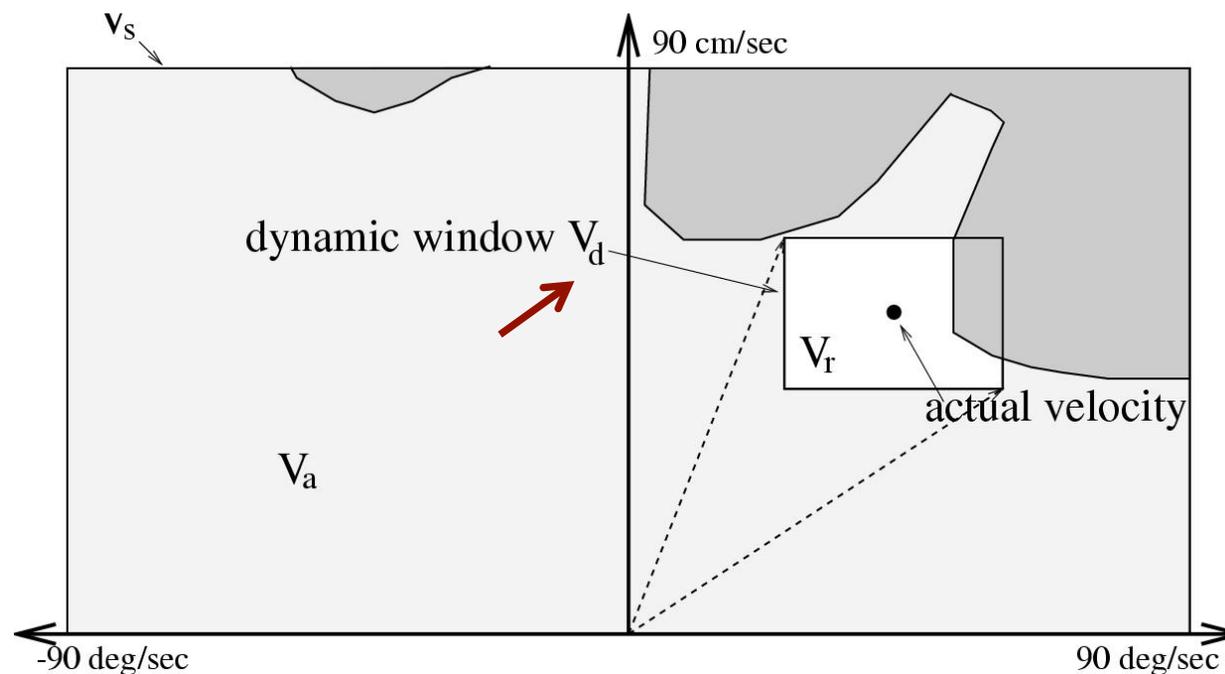
$$V_a = \{(v, \omega) \mid v \leq \sqrt{2 \text{dist}(v, \omega) a_{trans}} \wedge \omega \leq \sqrt{2 \text{dist}(v, \omega) a_{rot}}\}$$



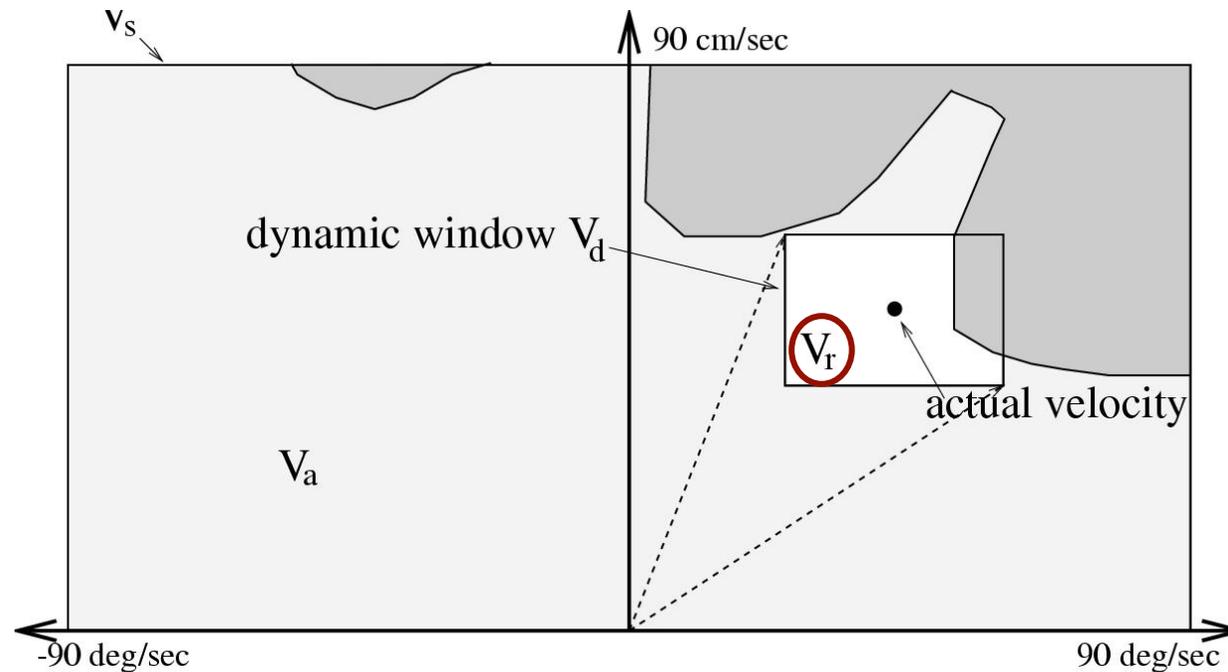
Reachable Velocities

- Speeds that are reachable by acceleration

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



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.

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

Dynamic Window Approach

- How to choose $\langle v, \omega \rangle$?
- Steering commands are chosen by a heuristic navigation function.
- This function tries to minimize the travel-time by:
“**driving fast** in the **right direction.**”

Dynamic Window Approach

- **Heuristic** navigation function.
- Planning restricted to $\langle x, y \rangle$ -space.
- No planning in the velocity 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.
- Planning restricted to $\langle x, y \rangle$ -space.
- No planning in the velocity 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.
- Planning restricted to $\langle x, y \rangle$ -space.
- No planning in the velocity 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.**

Dynamic Window Approach

- **Heuristic** navigation function.
- Planning restricted to $\langle x, y \rangle$ -space.
- No planning in the velocity 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.

Follows grid based path computed by A*.

Dynamic Window Approach

- **Heuristic** navigation function.
- Planning restricted to $\langle x, y \rangle$ -space.
- No planning in the velocity space.

Navigation Function:

Goal nearness.

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

Maximizes velocity.

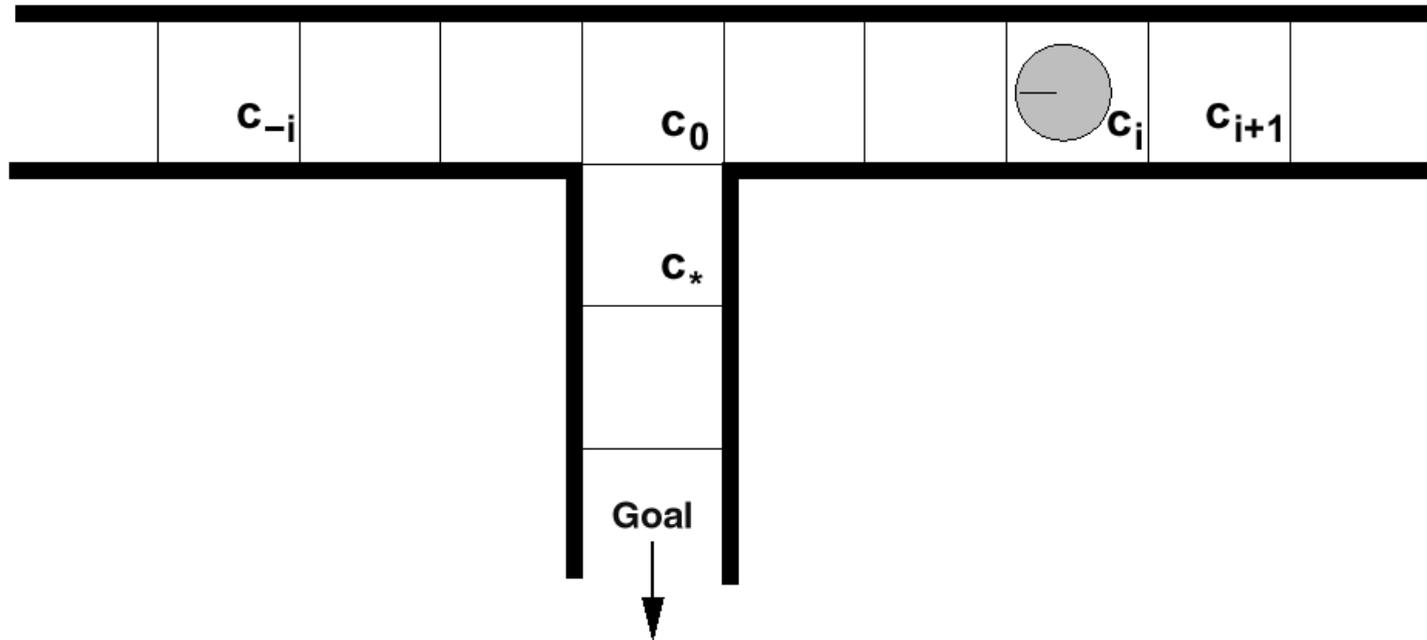
Considers cost to reach the goal.

Follows grid based path computed by A*.

Dynamic Window Approach

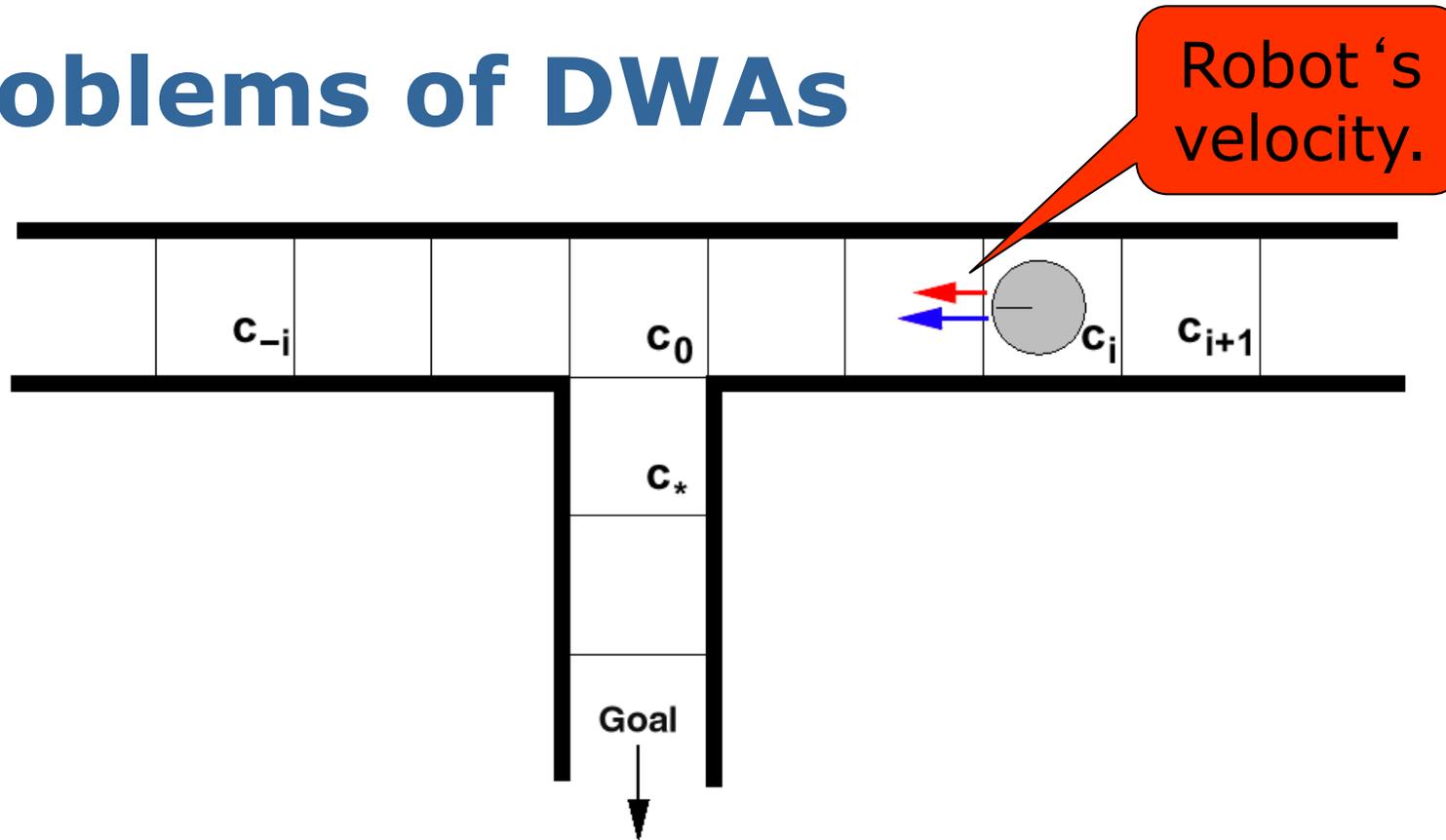
- Reacts quickly.
- Low CPU power requirements.
- Guides a robot on a collision-free path.
- Successfully used in a lot of real-world scenarios.
- Resulting trajectories sometimes sub-optimal.
- Local minima might prevent the robot from reaching the goal location.

Problems of DWAs



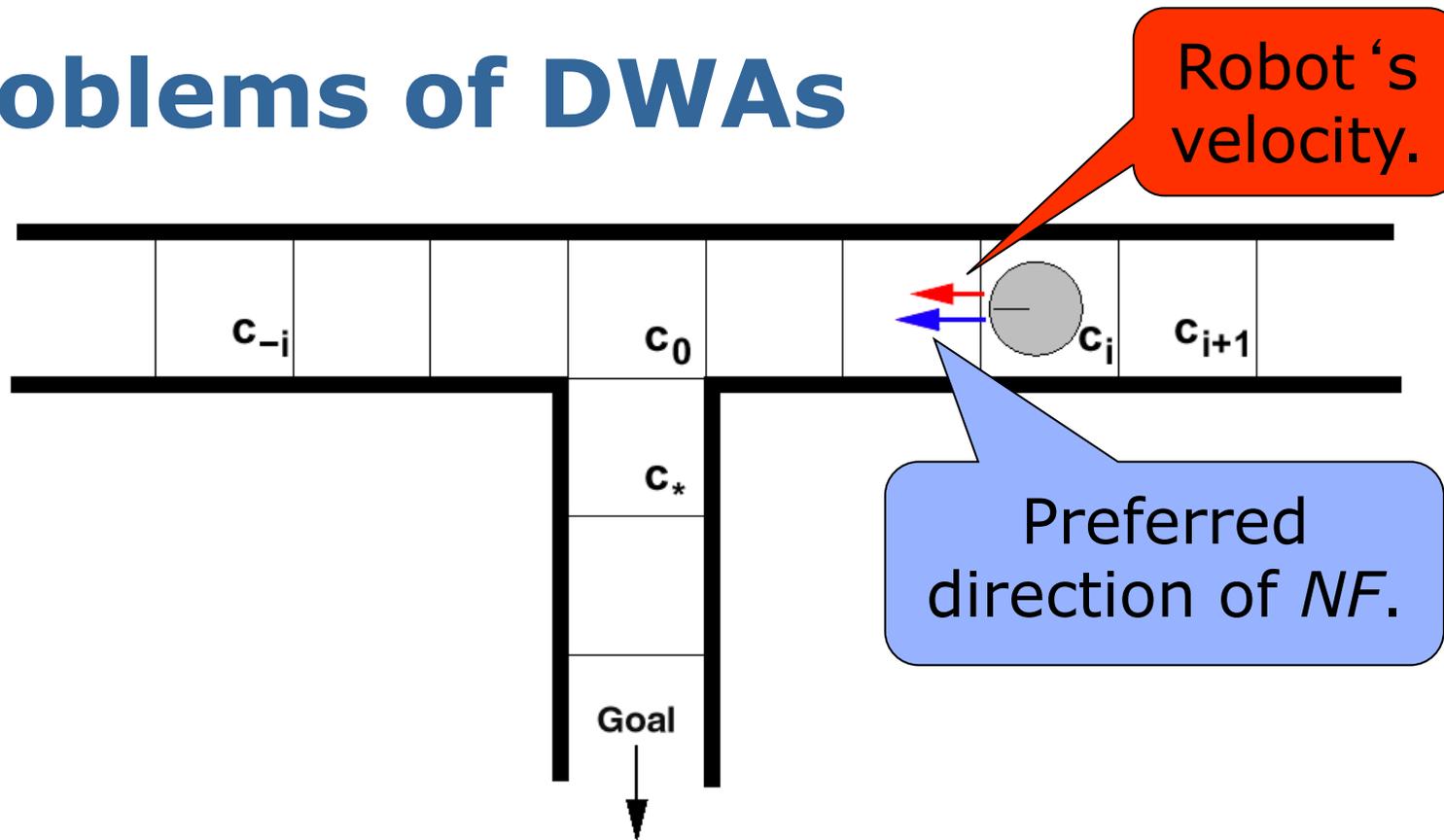
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



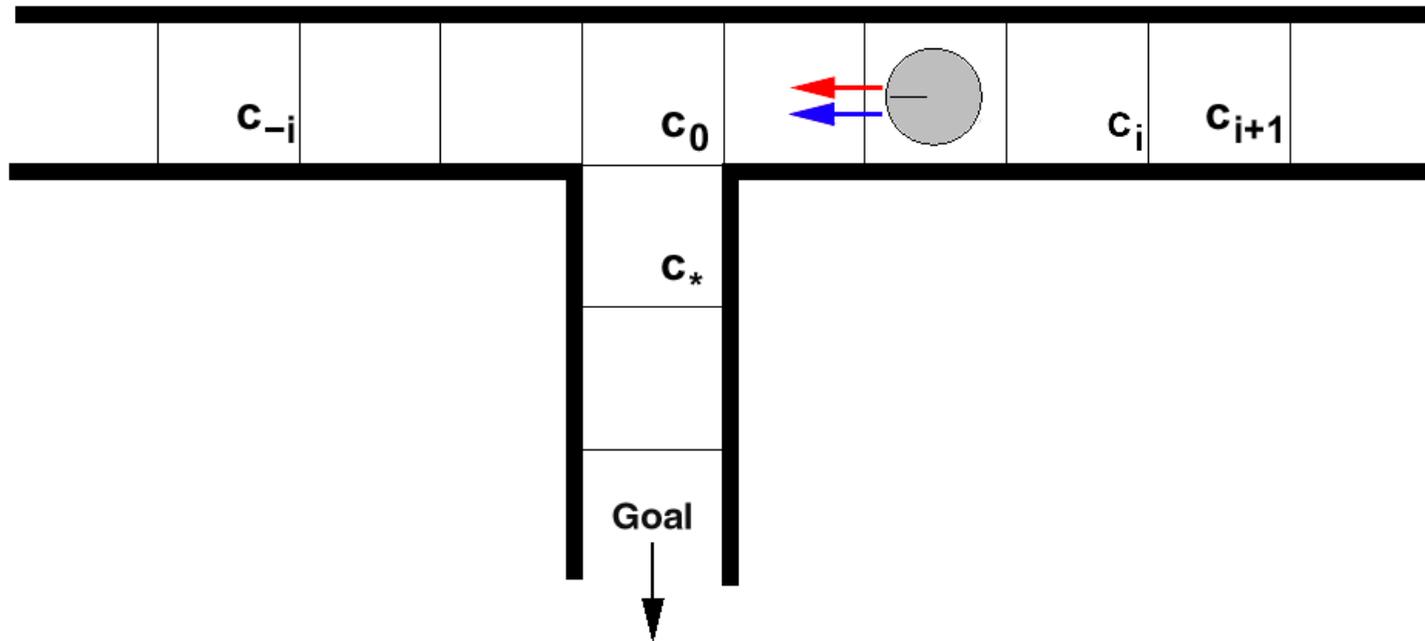
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



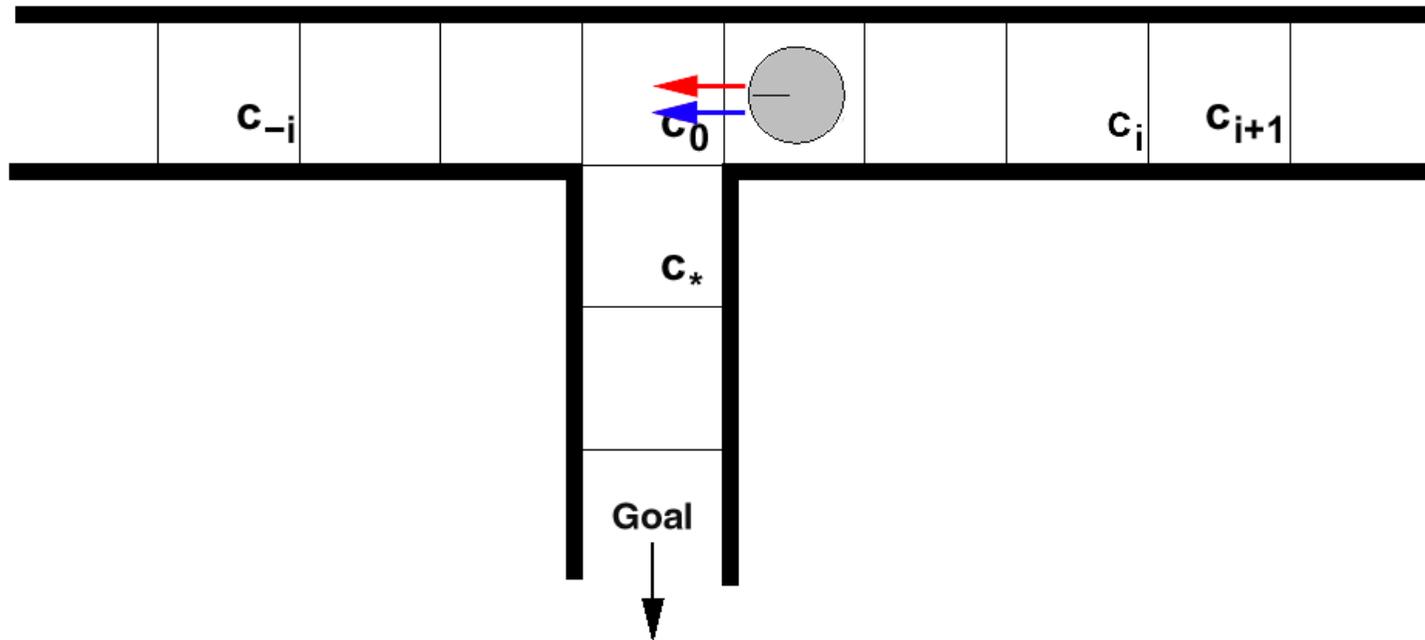
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



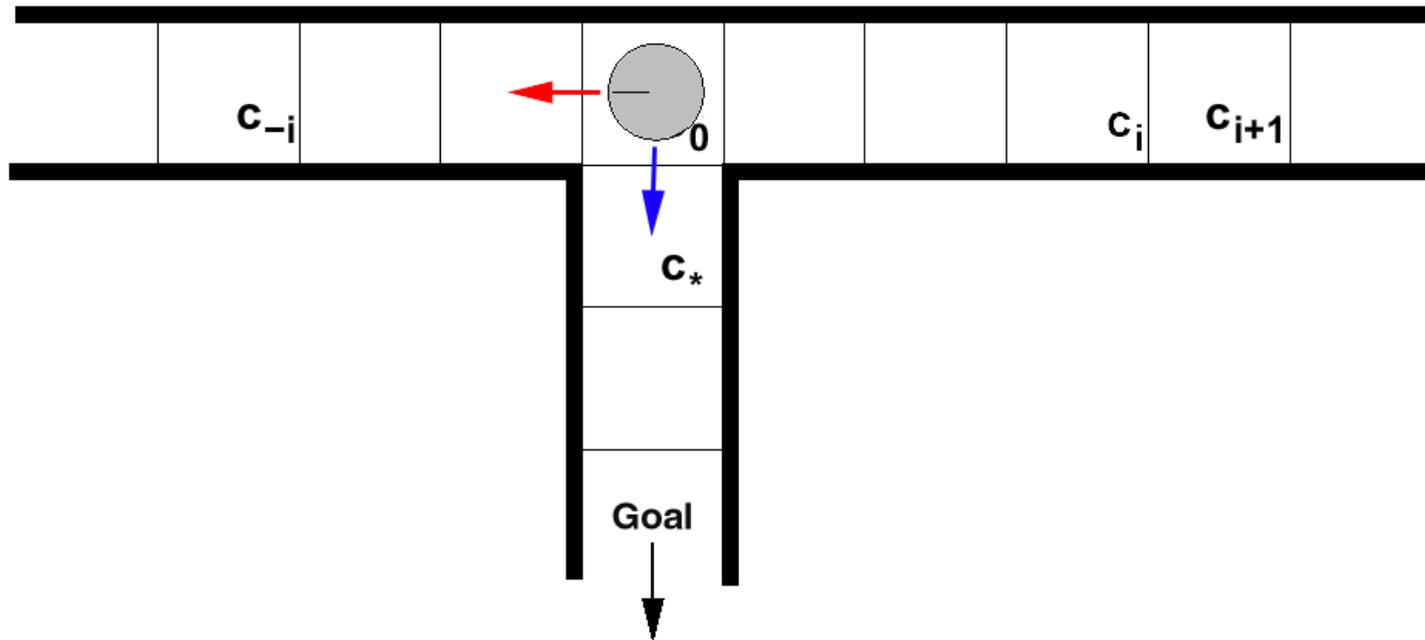
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



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

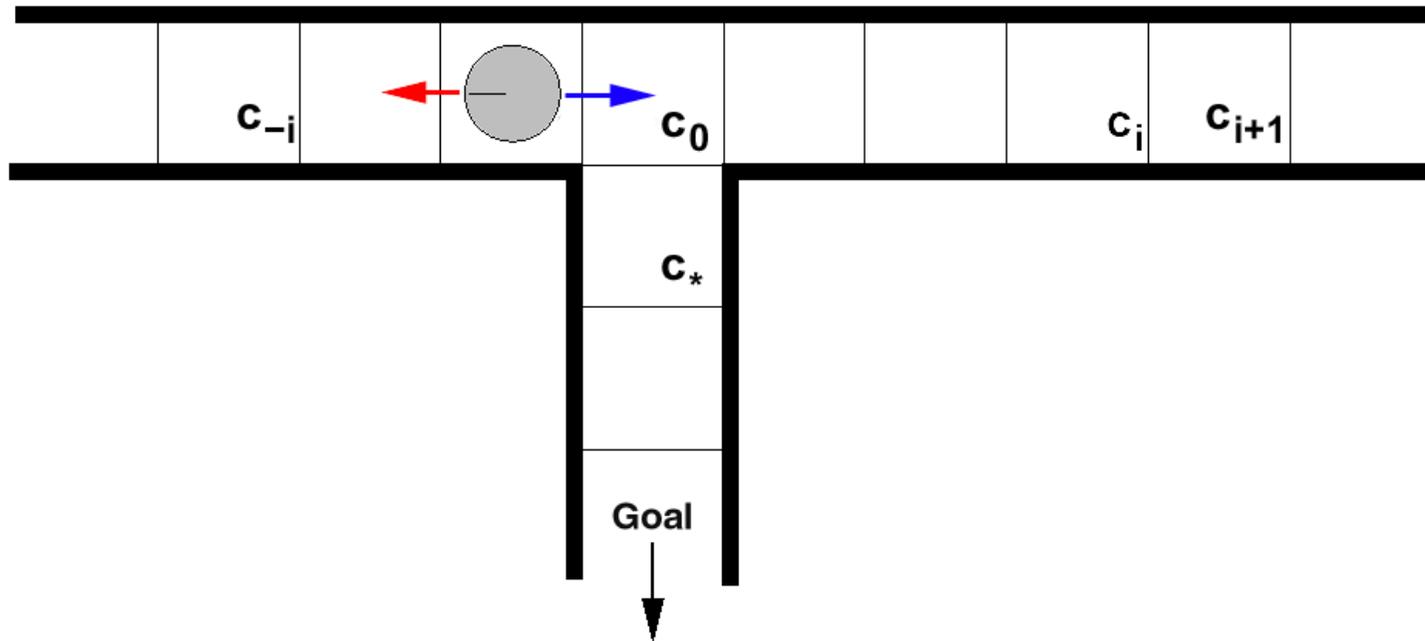
Problems of DWAs



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

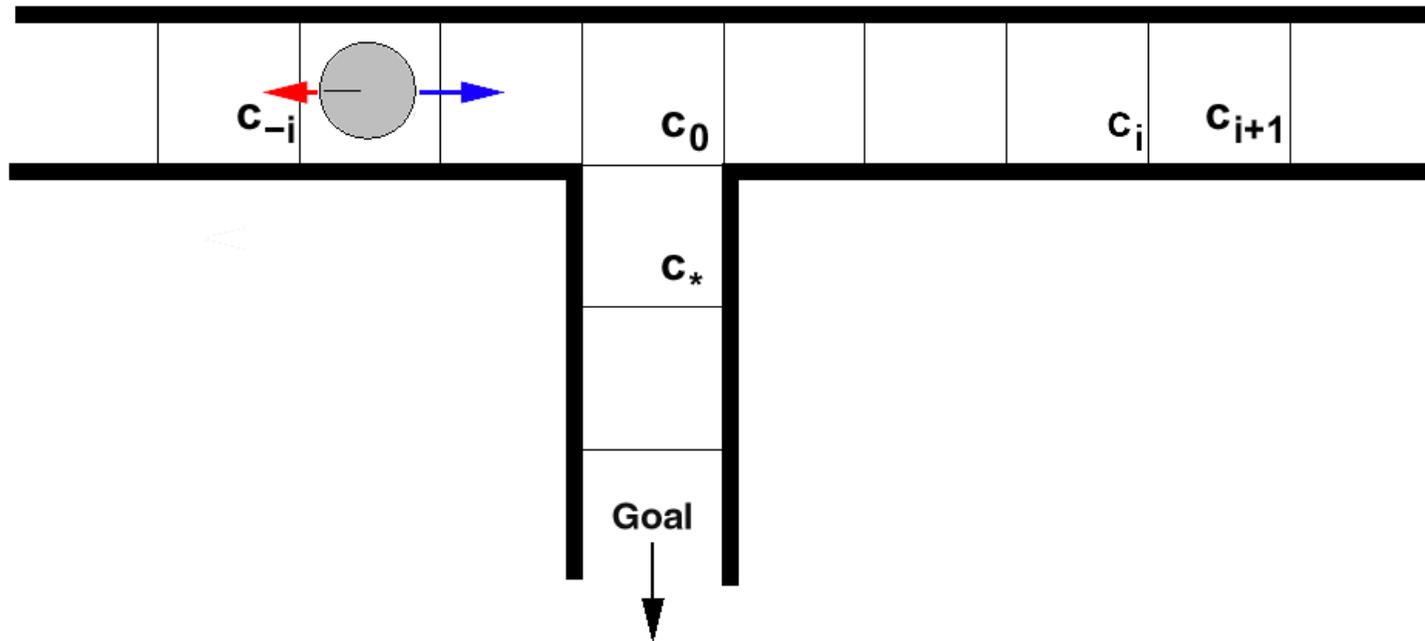
- The robot drives too fast at c_0 to enter corridor facing south.

Problems of DWAs



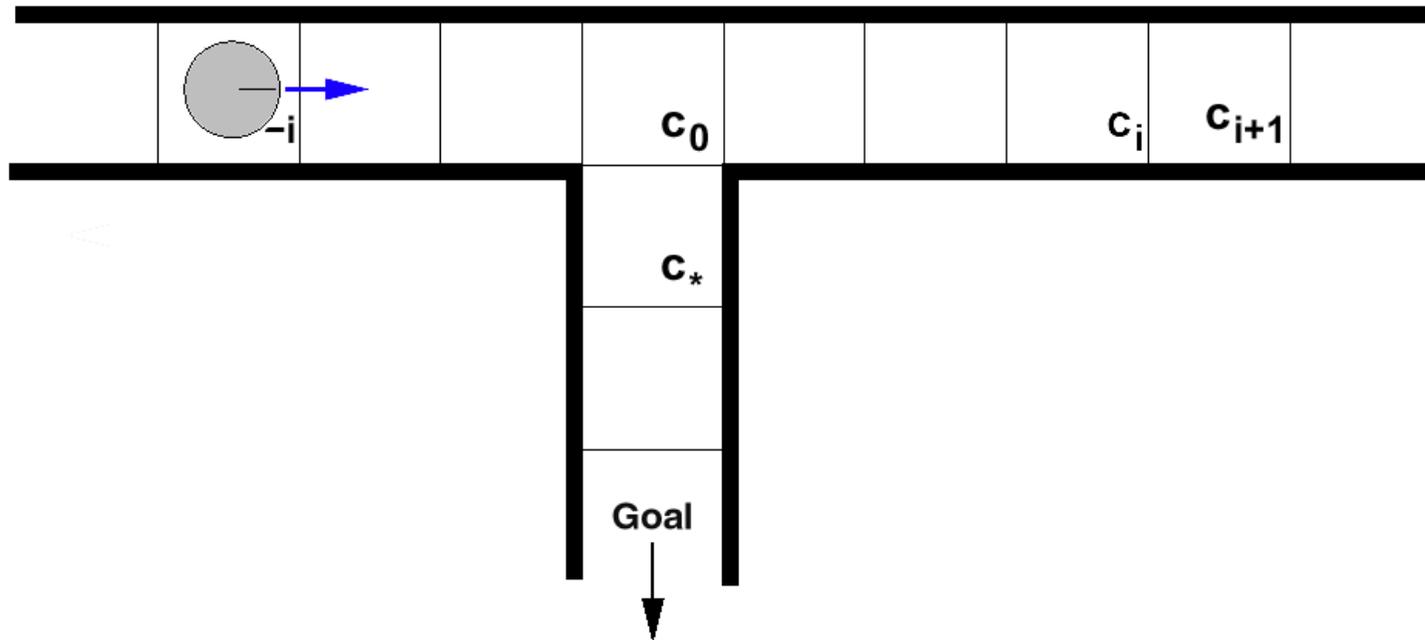
$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Problems of DWAs



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

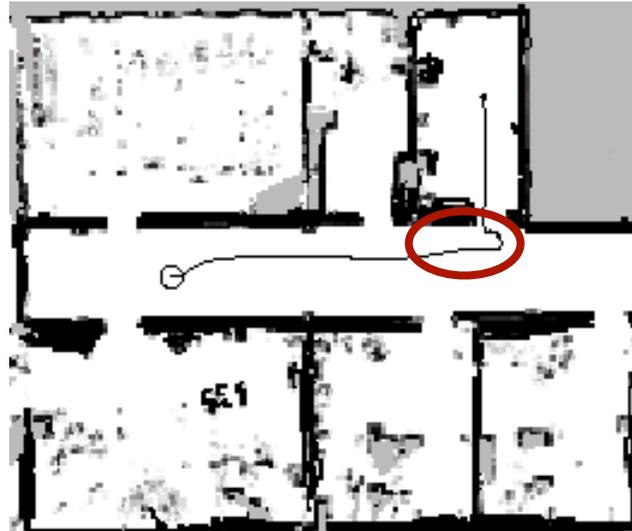
Problems of DWAs



- Same situation as in the beginning.
 - ➔ DWAs have problems to reach the goal.

Problems of DWAs

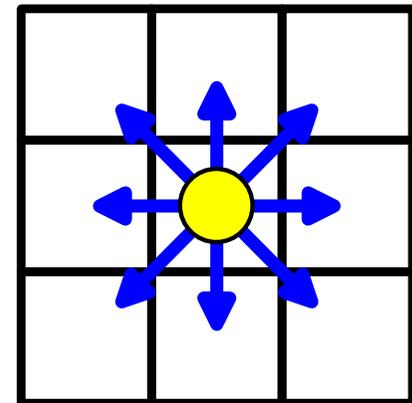
- Typical problem in a real world situation:



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

Robot Path Planning with A*

- 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



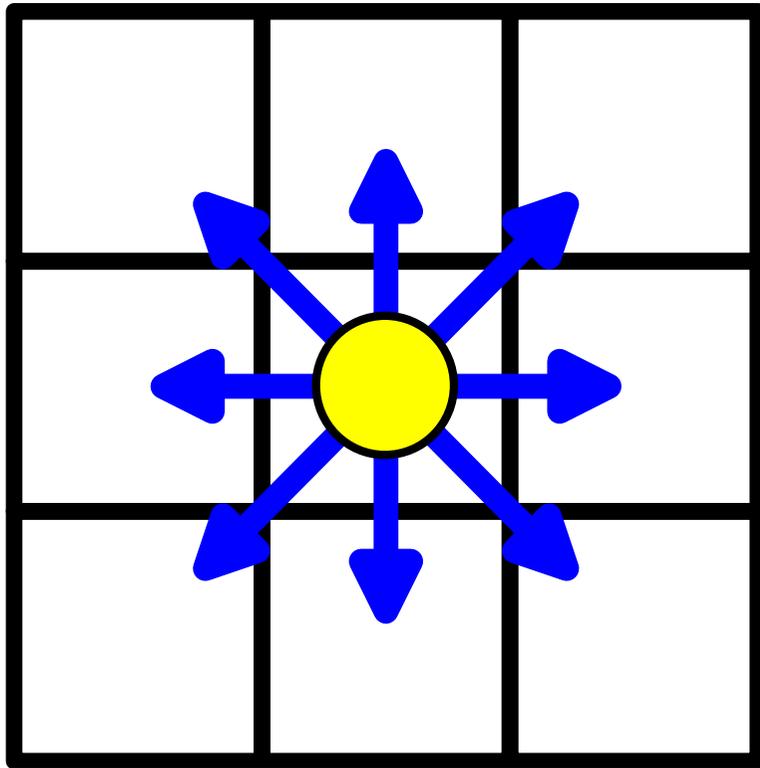
A*: Minimize the estimated path costs

- $g(n)$ = actual cost from the initial state to n .
- $h(n)$ = estimated cost from n to the next goal.
- $f(n) = g(n) + h(n)$, the estimated cost of the cheapest solution through n .
- Let $h^*(n)$ be the actual cost of the optimal path from n to the next goal.
- h is admissible if the following holds for all n :

$$h(n) \leq h^*(n)$$

- We require that for A*, h is admissible (the straight-line distance is admissible in the Euclidean Space).

Example: Path Planning for Robots in a Grid-World



Deterministic Value Iteration

- To compute the shortest path from every state to one goal state, use (deterministic) value iteration.
- Very similar to Dijkstra's Algorithm.
- Such a cost distribution is the optimal heuristic for A^* .



Typical Assumption in Robotics for A* Path Planning

- The robot is assumed to be localized.
- The robot computes its path based on an occupancy grid.
- The correct motion commands are executed.

Is this always true?

Problems

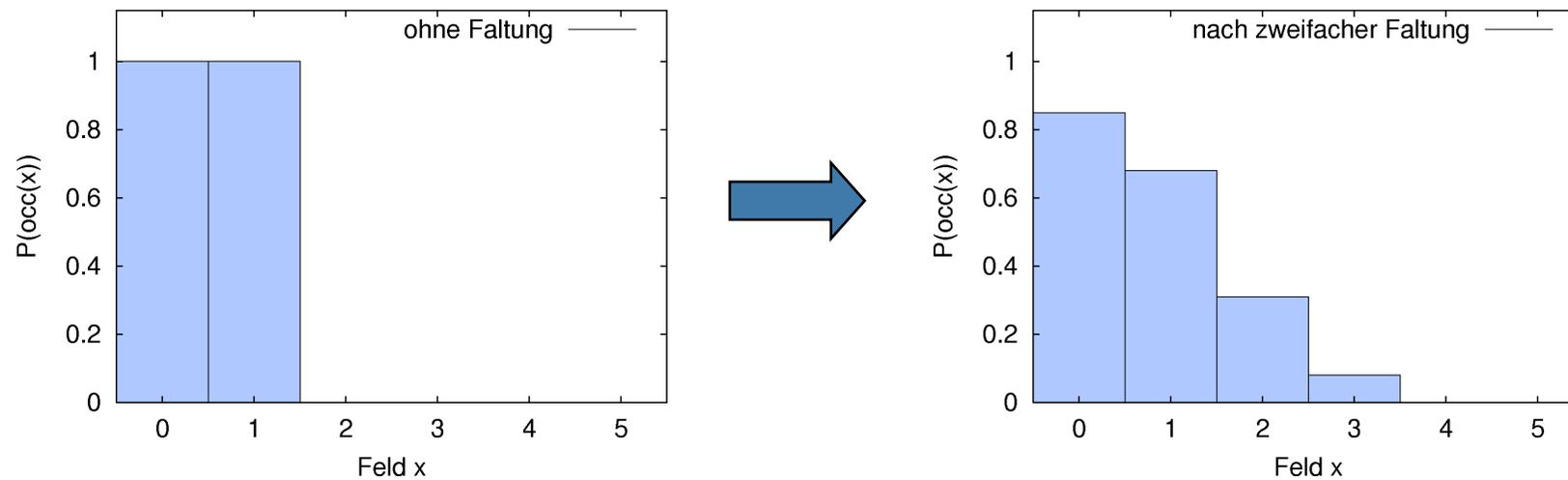
- What if the robot is slightly delocalized?
- Moving on the shortest path guides often the robot on a trajectory close to obstacles.
- Trajectory aligned to the grid structure.

Convolution of the Grid Map

- Convolution blurs the map.
- Obstacles are assumed to be bigger than in reality.
- Perform an A* search in such a convolved map.
- Robot increases **distance to obstacles** and moves on a **short path!**

Example: Map Convolution

- 1-d environment, cells c_0, \dots, c_5



- Cells before and after 2 convolution runs.

Convolution

- Consider an occupancy map. Then the convolution 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.
- “Gaussian blur”

A* in Convolved Maps

- The costs are a product of path length and occupancy probability of the cells.
- Cells with higher probability (e.g., caused by convolution) are avoided by the robot.
- Thus, it keeps distance to obstacles.
- This technique is **fast** and quite **reliable**.

5D-Planning – an Alternative to the Two-layered Architecture

- Plans in the full $\langle x, y, \theta, v, \omega \rangle$ -configuration space using A^* .
 - Considers the robot's kinematic constraints.
- Generates a sequence of steering commands to reach the goal location.
- Maximizes trade-off between driving time and distance to obstacles.

The Search Space (1)

- What is a state in this space?
 $\langle x, y, \theta, v, \omega \rangle =$ current position and speed of the robot
- How does a state transition look like?
 $\langle x_1, y_1, \theta_1, v_1, \omega_1 \rangle \longrightarrow \langle x_2, y_2, \theta_2, v_2, \omega_2 \rangle$
with motion command (v_2, ω_2) and
 $|v_1 - v_2| < a_v, |\omega_1 - \omega_2| < a_\omega$. Pose of the Robot is a result of 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 (.25 secs for online control).

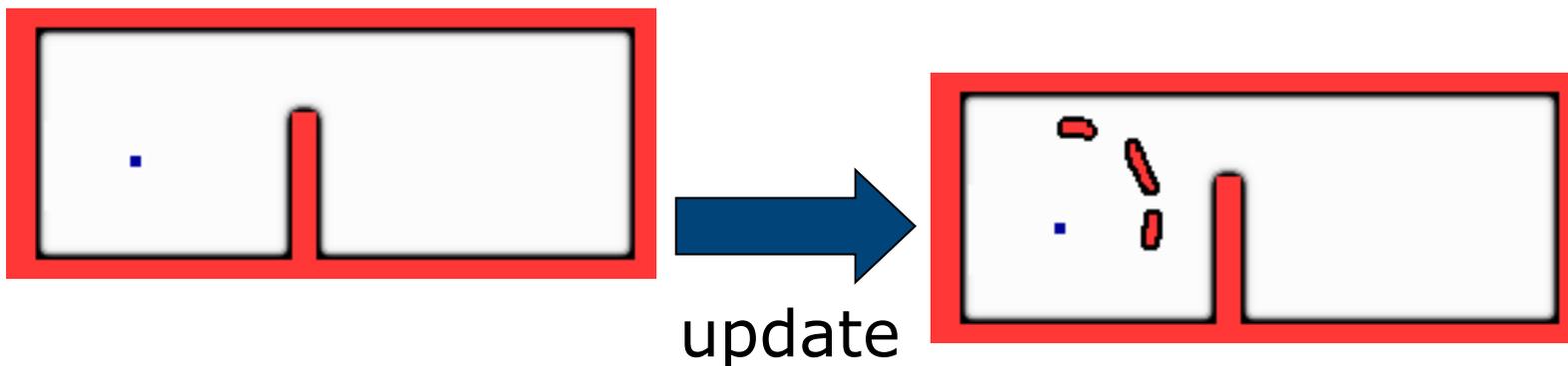
Solution: restrict the full search space.

The Main Steps of Our Algorithm

1. Update (static) grid map based on sensory input.
2. Use A^* to find a trajectory in the $\langle x, y \rangle$ -space using the updated grid map.
3. Determine a restricted 5d-configuration space based on step 2.
4. Find a trajectory by planning in the restricted $\langle x, y, \theta, v, \omega \rangle$ -space.

Updating the Grid Map

- The environment is represented as a 2d-occupancy grid map.
- Convolution of the map increases security distance.
- Detected obstacles are added.
- Cells discovered free are cleared.



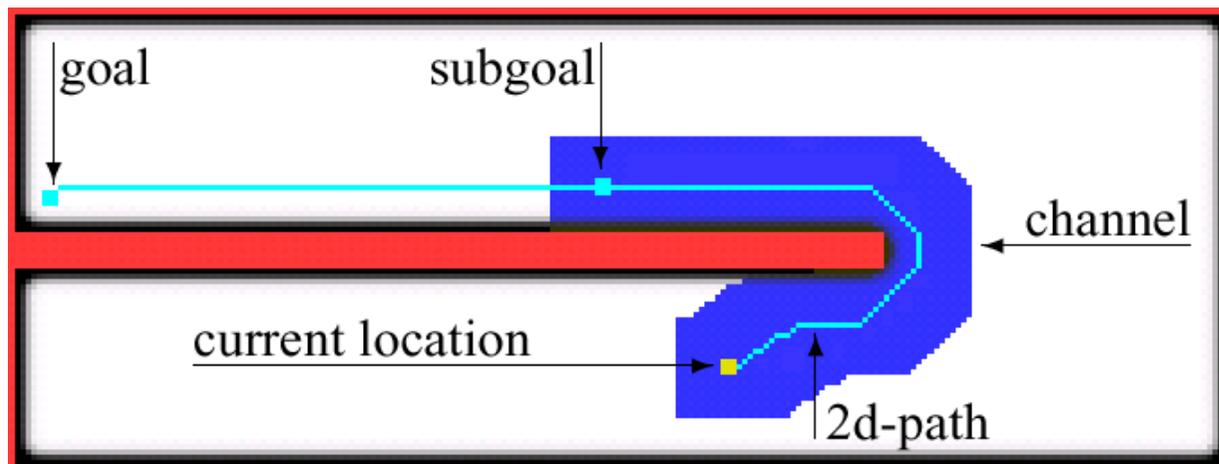
Restricting the Search Space

Assumption: the projection of the 5d-path onto the $\langle x, y \rangle$ -space lies close to the optimal 2d-path.

Therefore: construct a restricted search space (channel) based on the 2d-path.

Space Restriction

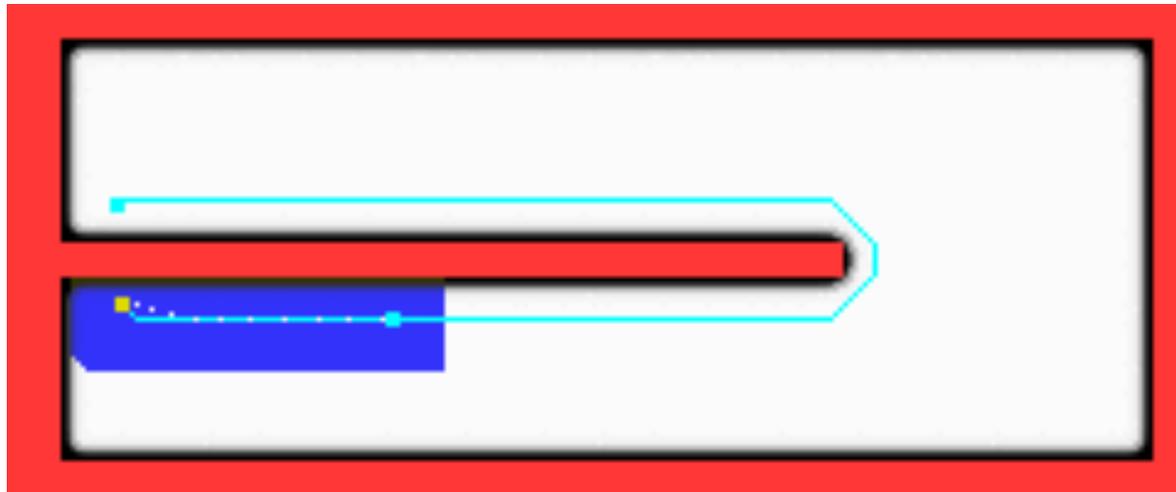
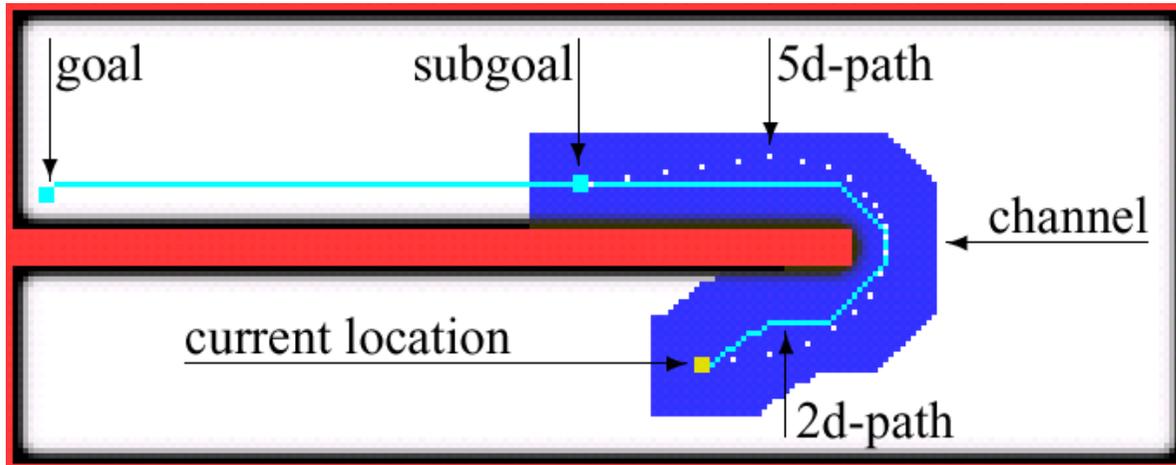
- Resulting search space = $\langle x, y, \theta, v, \omega \rangle$ with $(x, y) \in \text{channel}$.
- Choose a sub-goal lying on the 2d-path within the channel.



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.
- To estimate cell costs: perform a deterministic 2d-value iteration within the channel.

Examples



Timeouts

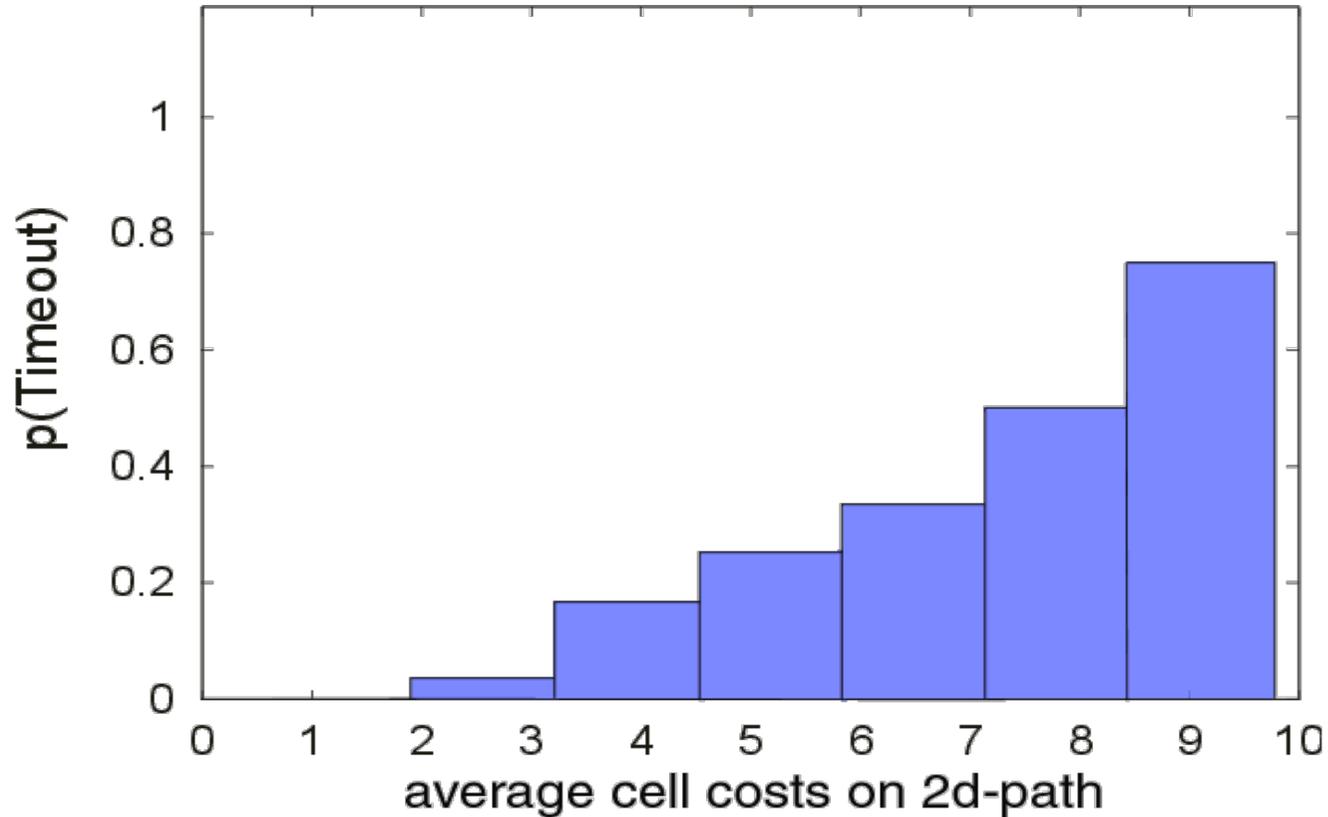
- Steering a robot online requires to set a new steering command every .25 secs.
- Abort search after .25 secs.

How to find an admissible steering command?

Alternative Steering Command

- Previous trajectory still admissible?
→ OK
- If not, drive on the 2d-path or use DWA to find new command.

Timeout Avoidance



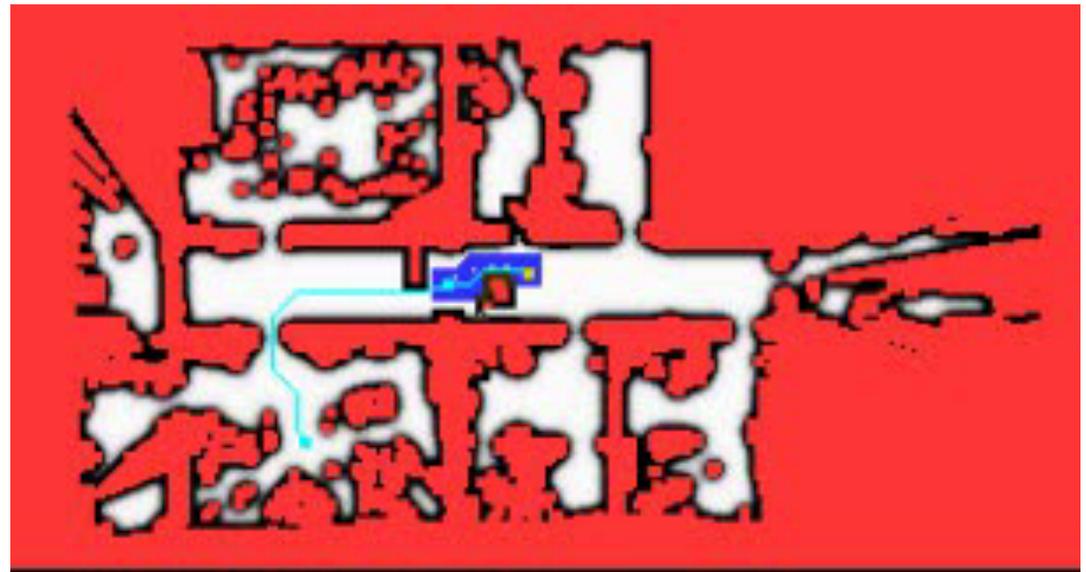
- ➔ Reduce the size of the channel if the 2d-path has high cost.

Example

start videos



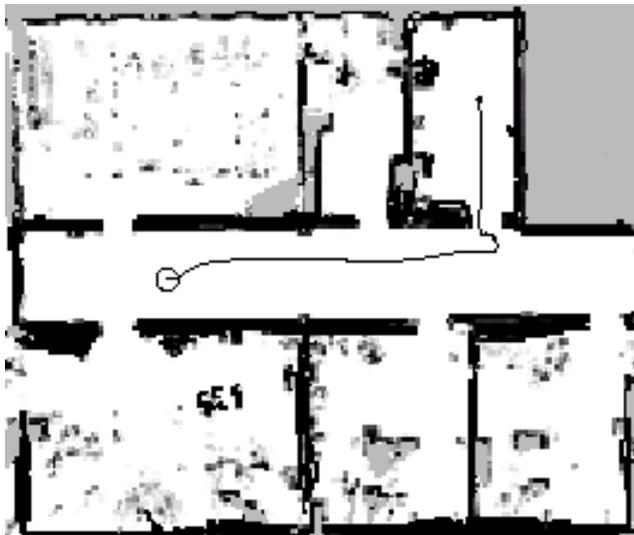
Robot Albert



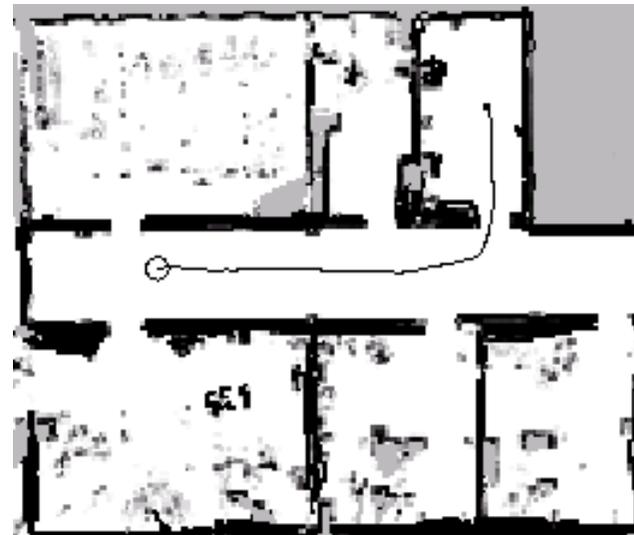
Planning state

Comparison to the DWA (1)

- DWAs often have problems entering narrow passages.



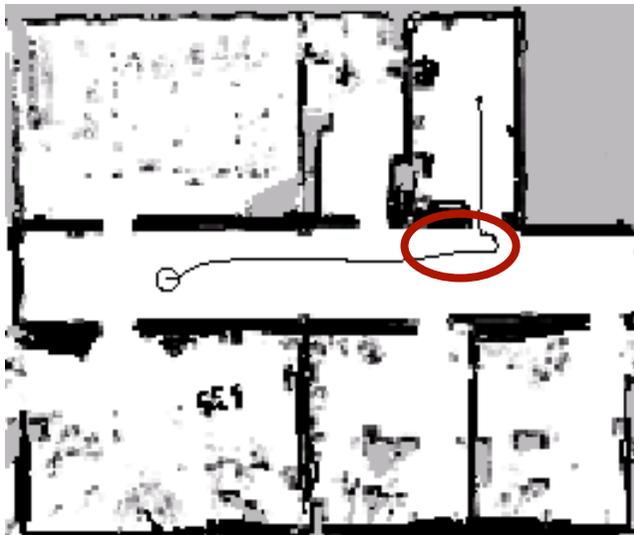
DWA planned path.



5D approach.

Comparison to the DWA (1)

- DWAs often have problems entering narrow passages.

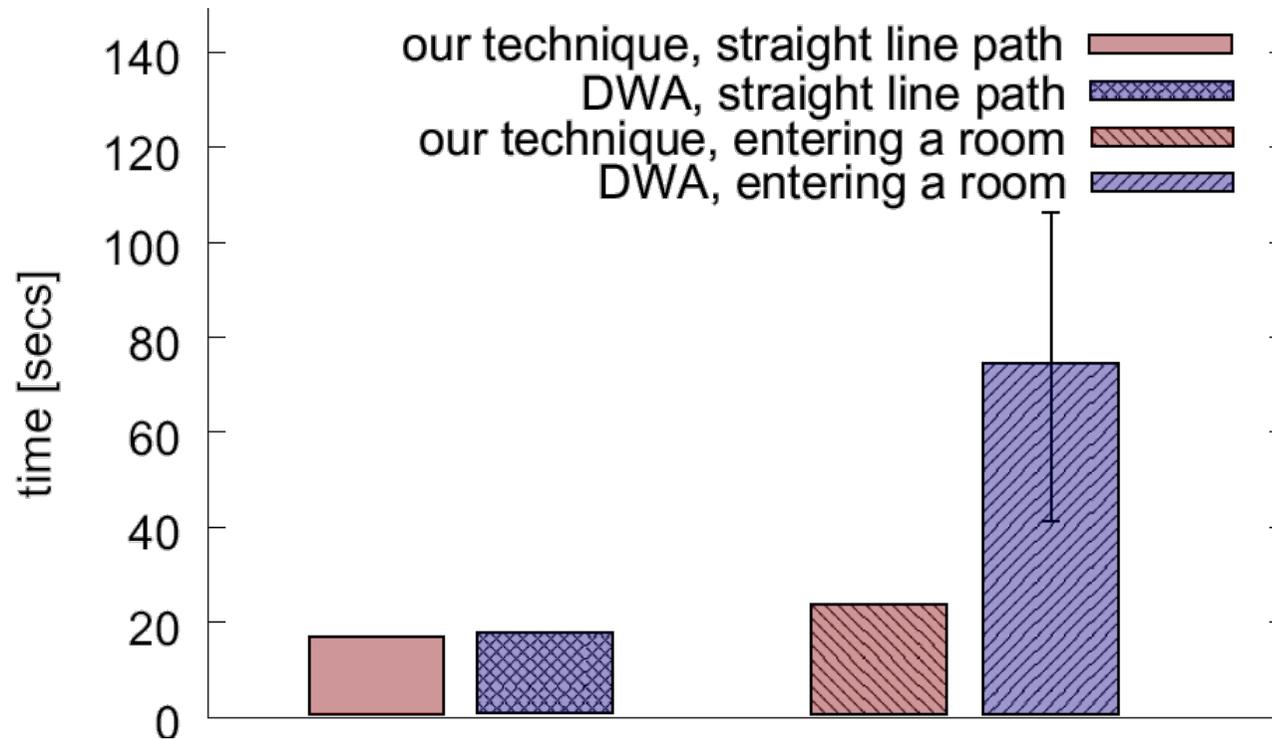


DWA planned path.



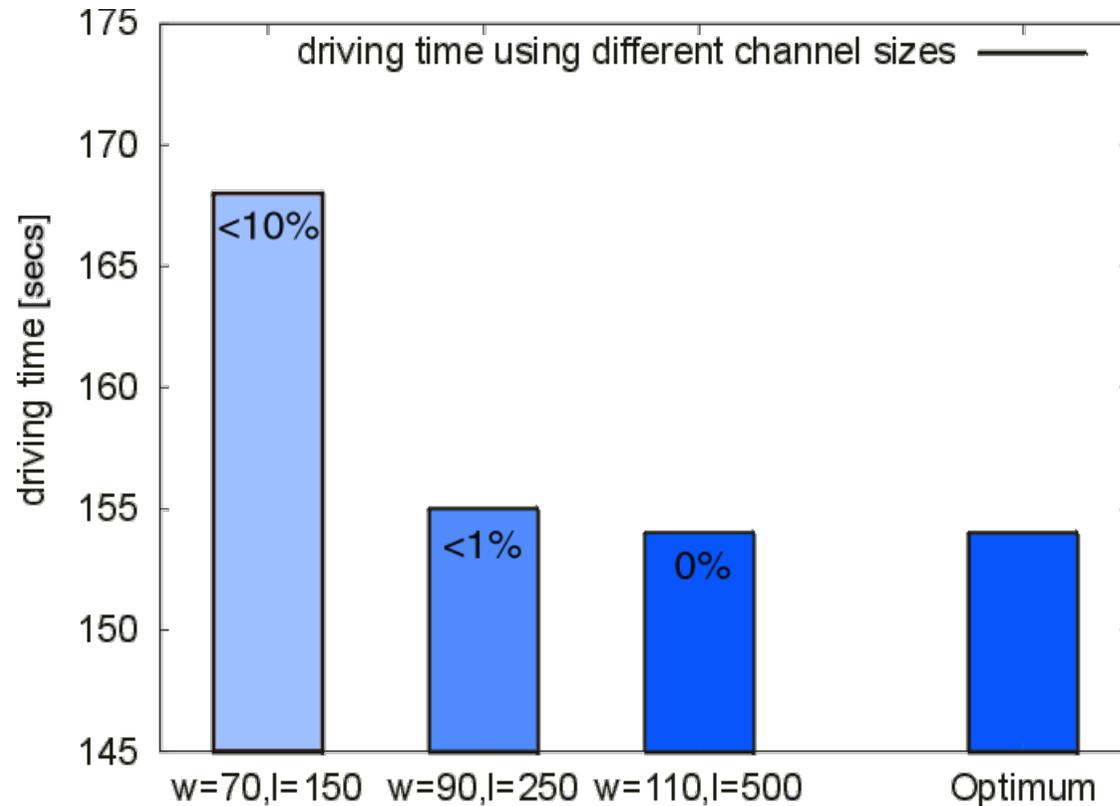
5D approach.

Comparison to the DWA (2)



The presented approach results in significantly 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 combined path planning & collision avoidance
- Approaches need to consider **robot's kinematic constraints** and **plans in the velocity space**.
- Combination of search and reactive techniques show **better results than the pure DWA** in a variety of situations.
- Using the 5D-approach the **quality of the trajectory scales** with the performance of the underlying hardware.
- The **resulting paths** are often **close to the optimal ones**.

What's Missing?

- More complex vehicles (e.g., cars).
- Moving obstacles, motion prediction.
- ...