اصول علم ربات – اسلاید شانزدهم

Fundamentals of Robotics – Slide 16

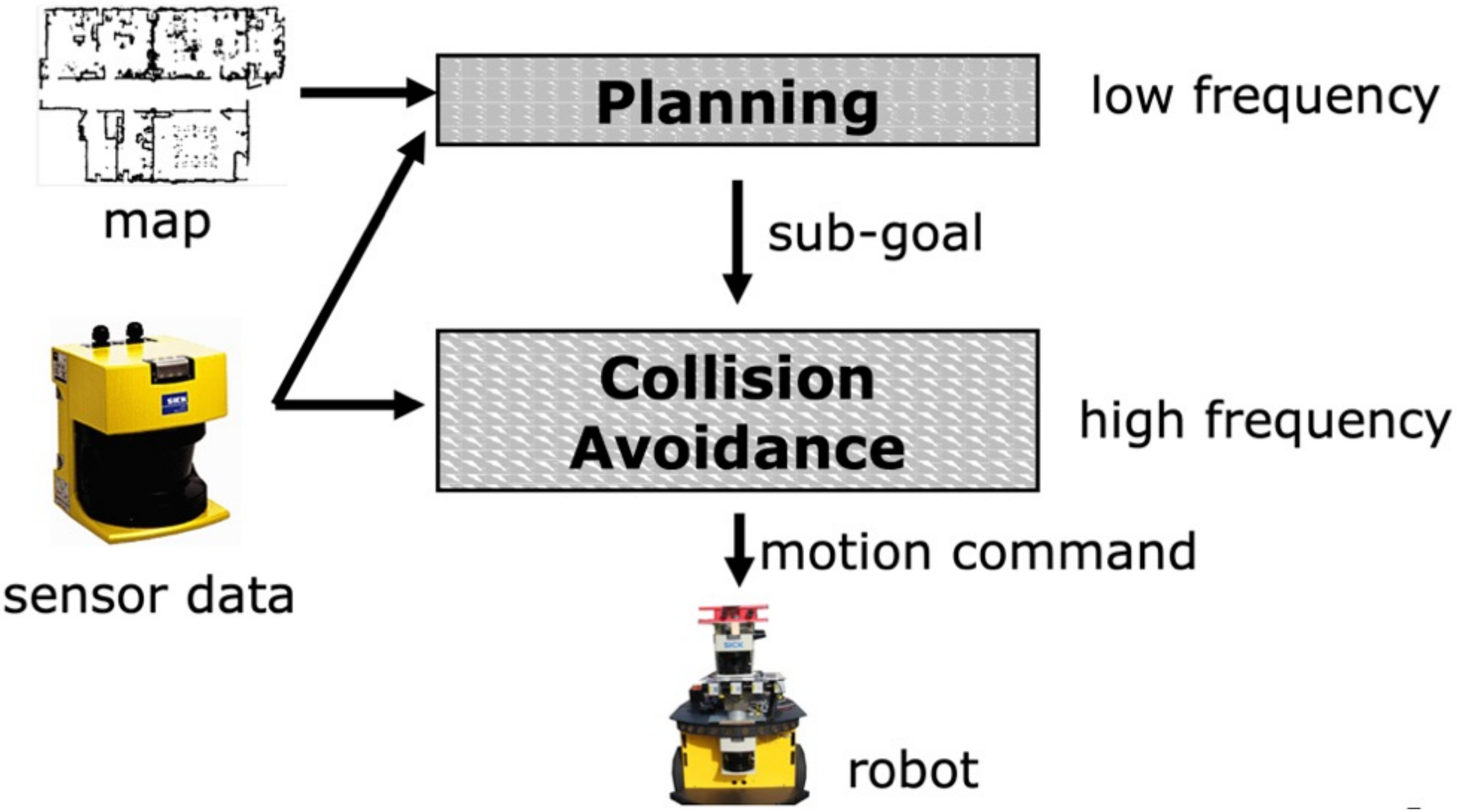
Navigation in Presence of Obstacles
Maps, Local Maps

دکتر مهدی جوانمردی

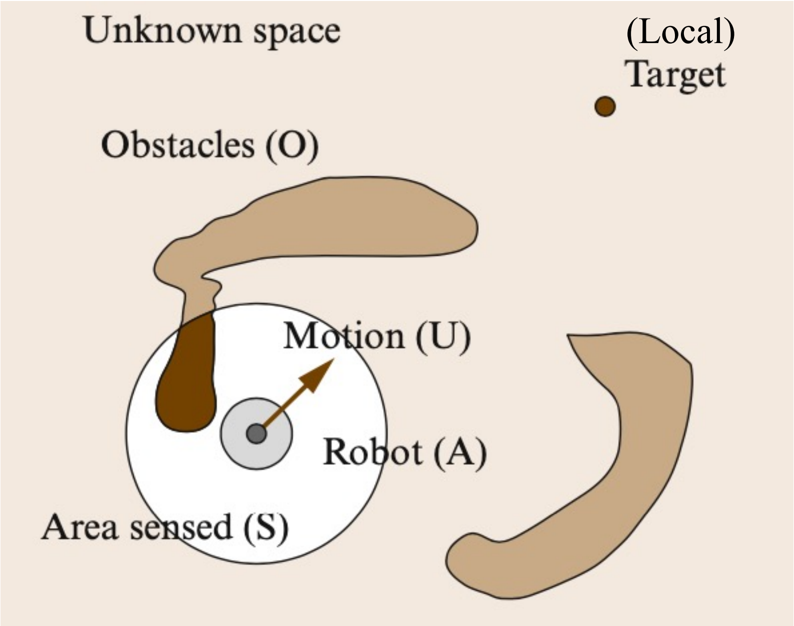
زمستان ۱۴۰۰ – بهار ۱۴۰۱

[slides adapted from Gianni Di Caro, @CMU with permission]

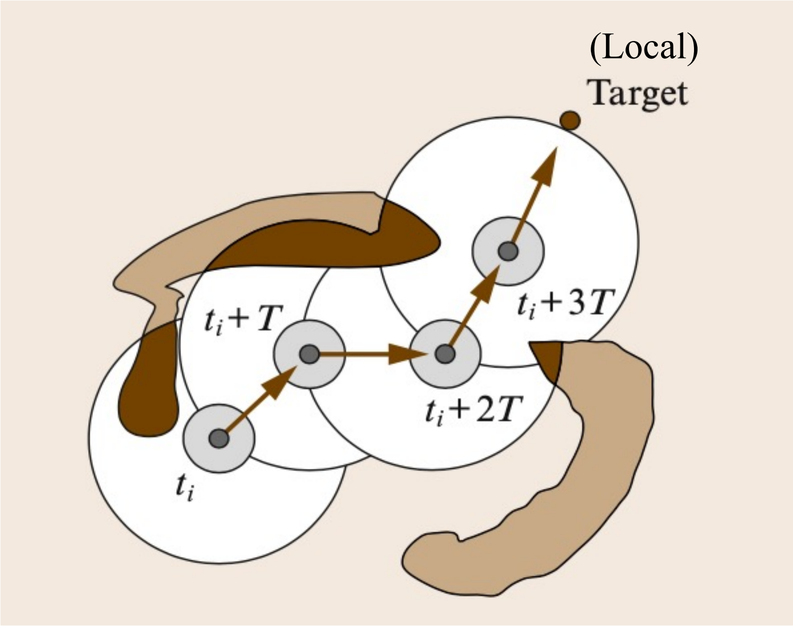
Standard two-layered architecture for map-based planning & navigation



Obstacle Avoidance / Local motion planning



- **Local target**: next waypoint, pose in a plan
- Use sensors to acquire information about surrounding environment
- Plan (or re-plan) the path in real-time avoiding obstacles
- Aim to reach target as fast and as reliably possible



❖ **Obstacle Avoidance / Local Planner** problem: computing a motion control that avoid collisions with the obstacles as observed by sensors, whilst driving the robot towards the target location.

➢ Result of applying this technique **online**, at each sample time, is a sequence of motions that drive the vehicle free of collisions to the target

Taxonomy of obstacle avoidance algorithms

- Methods that compute the motion in one step and that do it in more than one

- $Sensors \rightarrow Motion$: One-step methods directly reduce the sensor information to motion control
  - Various **heuristics**, e.g., Bug algorithms (reactive algorithms)
  - Use **physical analogies** assimilate the obstacle avoidance to a known physical problem, e.g., Potential field method

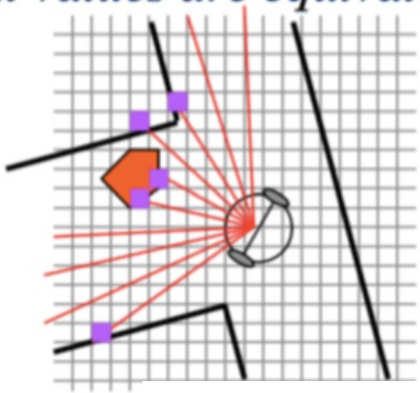
- $Sensors \rightarrow Intermediate\ Information\ Building \rightarrow Motion$: Methods with more than one step compute some intermediate information, which is processed next to obtain the motion.

  ✓ The methods of *subset of controls* compute an intermediate set of motion controls, and next choose one of them (the *best*) as a solution.

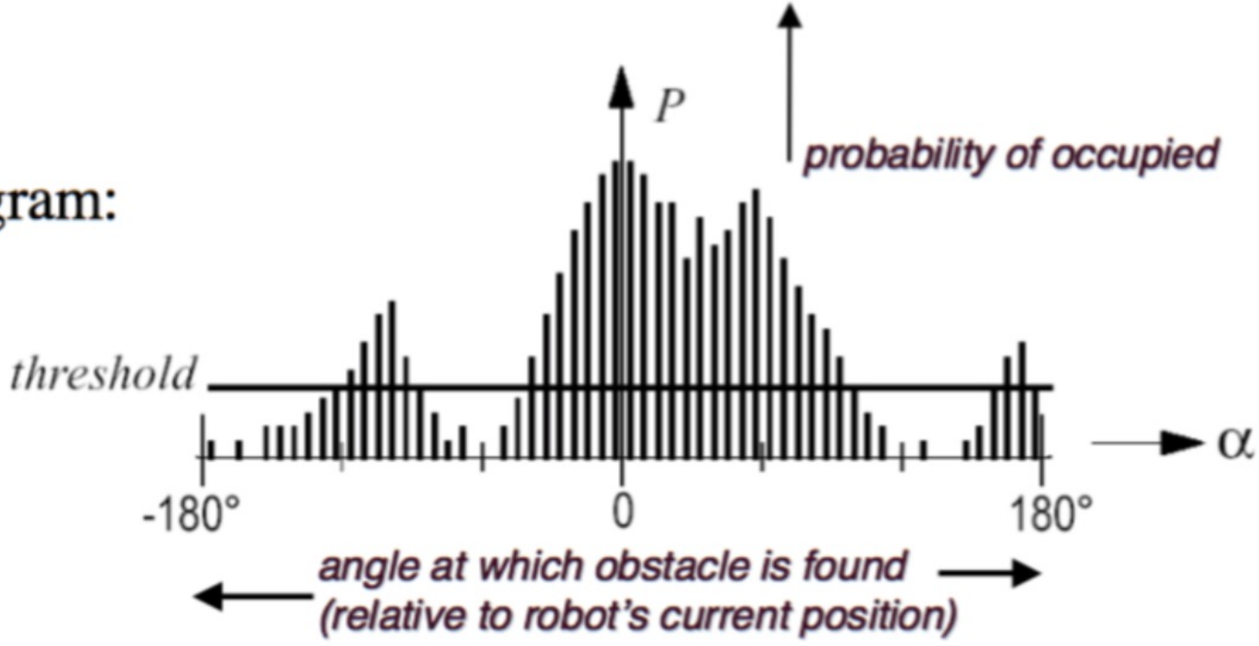
  - Subset of motion directions, e.g., Vector Field Histogram
  - Subset of velocity controls, e.g., Dynamic Window Adaptation

Vector Field Histogram

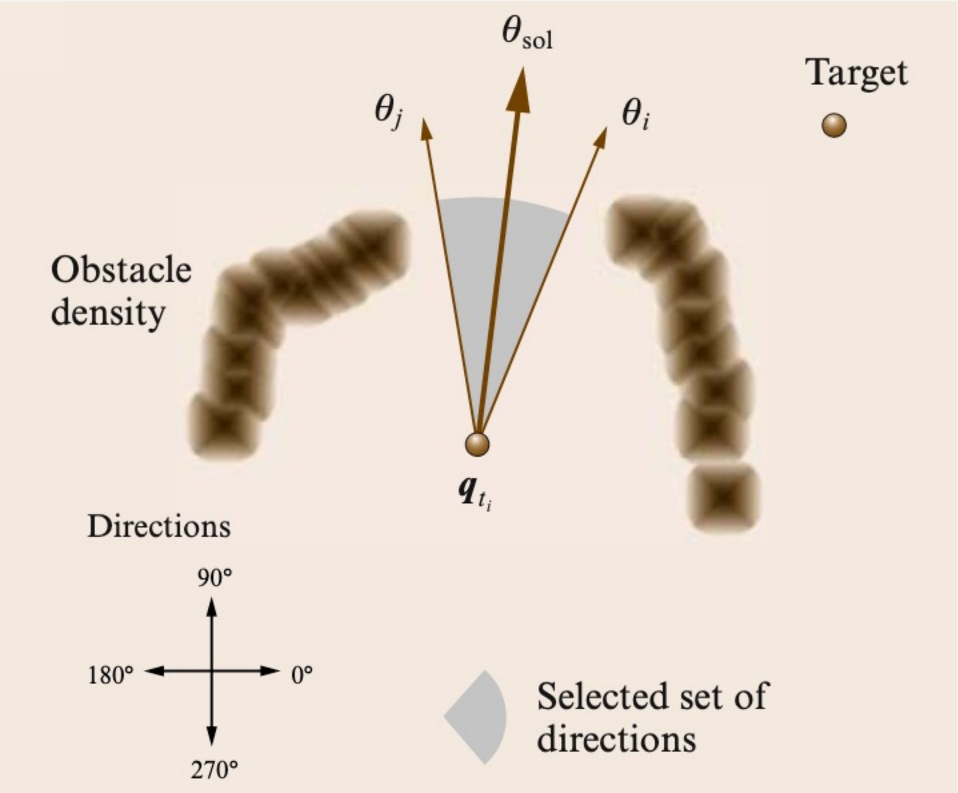
- Environment represented in a grid (2 DOF)   *Koren & Borenstein, ICRA 1990*
  - *cell values are equivalent to the probability that there is an obstacle*



- Generate polar histogram:



Vector Field Histogram



- Space is divided into sectors $k = 1, \ldots, N$ from robot location.
- Using sensor data, a polar histogram H is constructed around the robot, where each component represents the obstacle polar density in the corresponding sector.

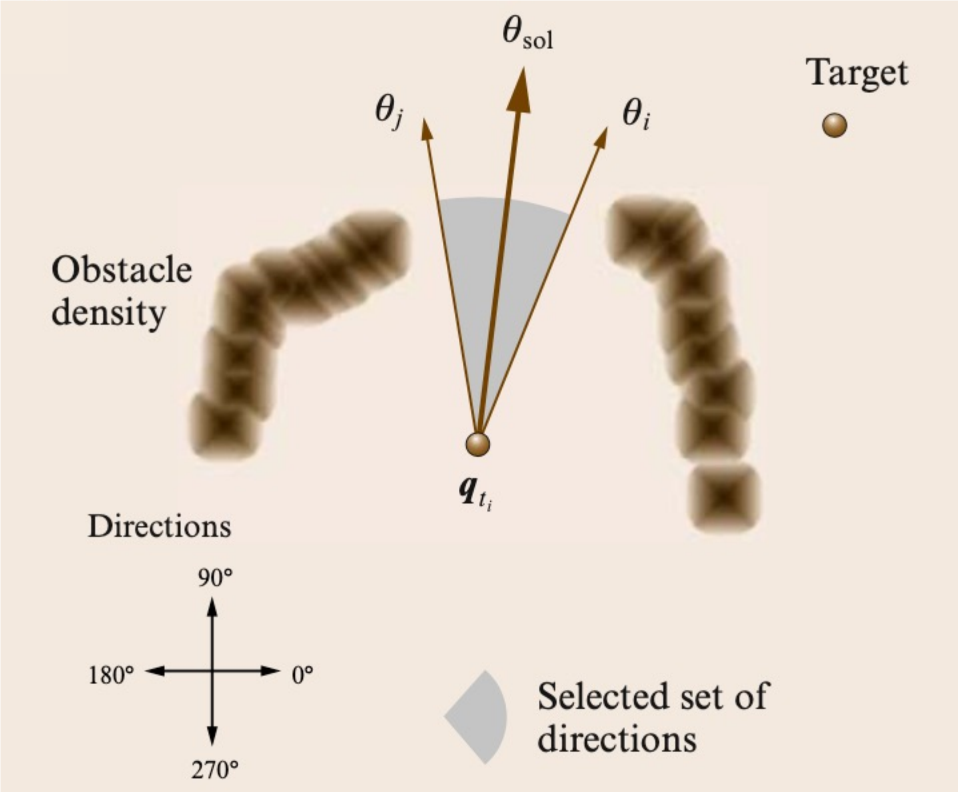
  - Function mapping observed obstacle distribution in sector k to a density value $h^k(\boldsymbol{q}_{t_i})$ in the histogram representation:

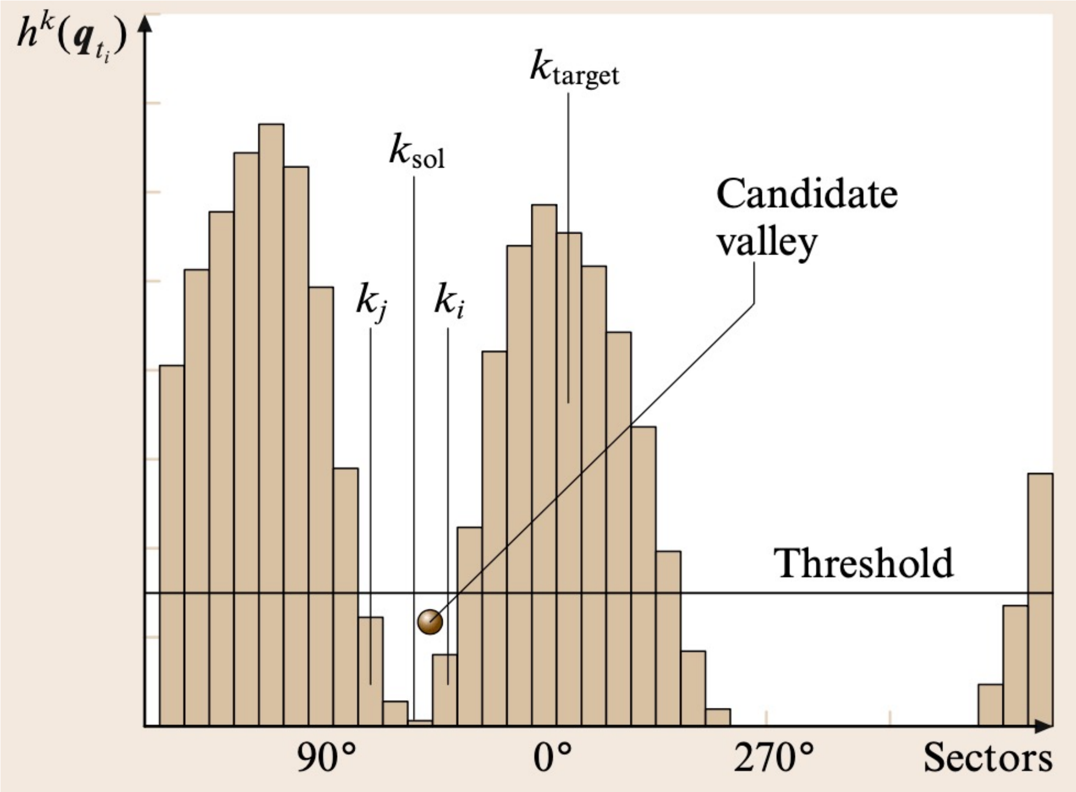
$$h^k(\boldsymbol{q}_{t_i}) = \int_{\Omega_k} P(\boldsymbol{p})^n \left(1 - \frac{d(\boldsymbol{q}_{t_i}, \boldsymbol{p})}{d_{\max}}\right)^r \mathrm{d}\boldsymbol{p}$$

where Ω_k is the set of points $\boldsymbol{p}$ falling within a certain maximal distance from the robot

$h^k(\boldsymbol{q}_{t_i}) \propto$ probability that a point is occupied by an obstacle $\times$ factor that increases as distance to point decreases

Vector Field Histogram: Step 1, select candidate directions



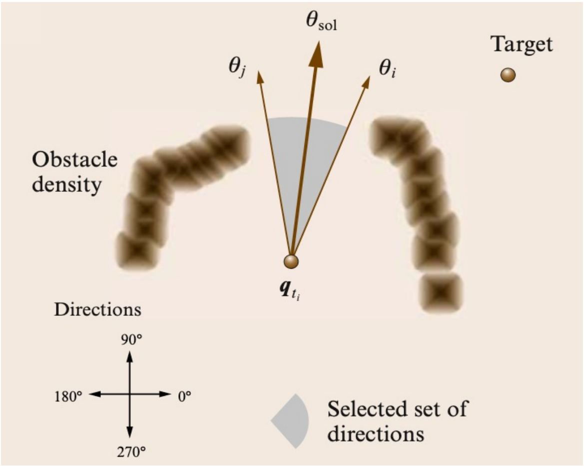


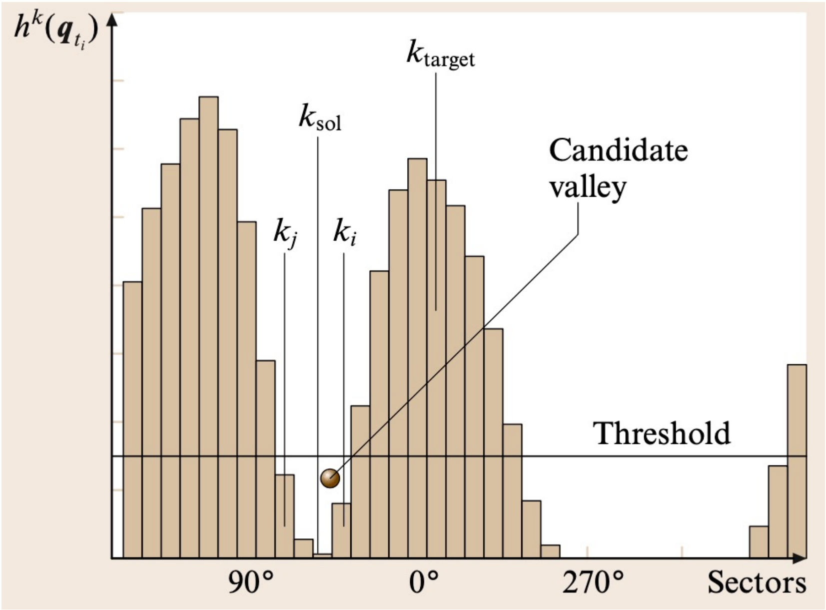
$$h^k(\boldsymbol{q}_{t_i}) = \int_{\Omega_k} P(\boldsymbol{p})^n \left(1 - \frac{d(\boldsymbol{q}_{t_i}, \boldsymbol{p})}{d_{\max}}\right)^r \mathrm{d}\boldsymbol{p}$$

✓ **Set of candidate directions**: set of adjacent components with lower density than a *given threshold*, and close to the component that contains the target direction

- Candidate *valleys*

Vector Field Histogram: Step 2, select motion

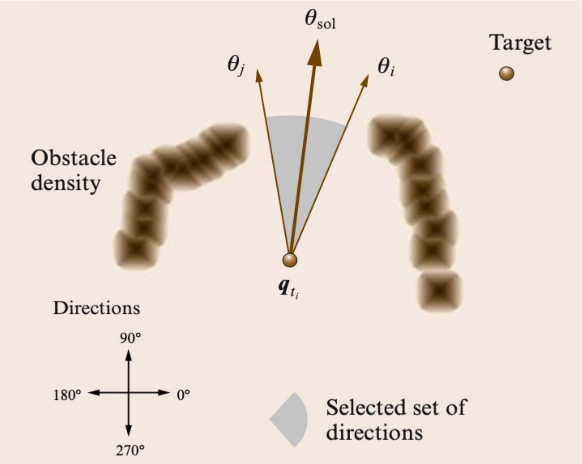


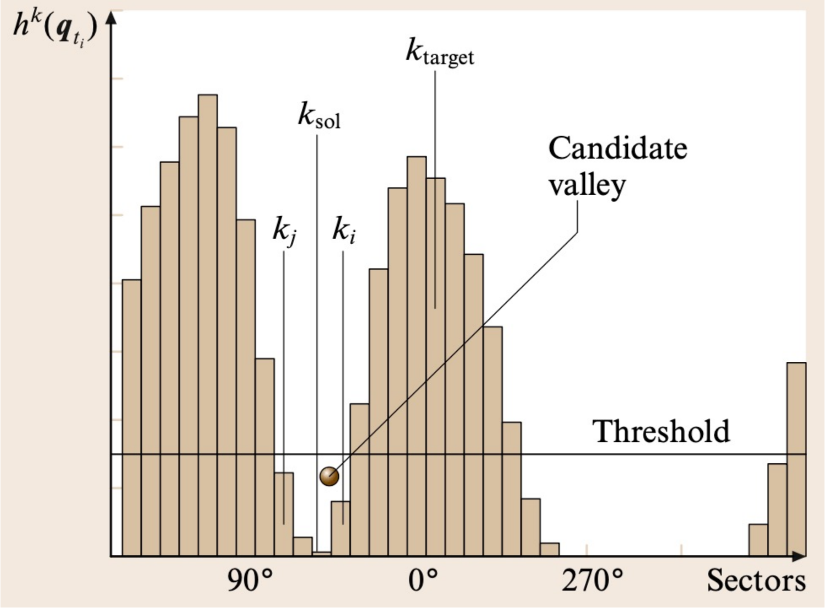


- ✓ **Set of candidate directions**: set of adjacent components with lower density than a given threshold, and close to the component that contains the target direction
- ➢ Select the *best* direction (i.e., sector) k_{sol}
  - Heuristic based on three cases

- ✓ **Case 1**: goal sector in the selected valley → $k_{sol} = k_{target}$ where k_{target} is the sector that contains the goal location
- ✓ **Case 2**: goal sector not in the selected valley and the number of sectors in the valley is greater than a threshold m (e.g., $m = 8$ → valley of ≈ 45^o → large valley) → $k_{sol} = k_{closer} \pm \frac{m}{2}$ where k_{closer} is the sector of the valley closer to k_{target}
- ✓ **Case 3**: goal sector not in the selected valley and number of sectors in the valley is lower than m (i.e., a narrow valley) → $k_{sol} = \frac{k_i+k_j}{2}$ where k_i and k_j are the extremal sectors of the valley

Vector Field Histogram: Step 2, select motion





✓ **Case 3**: goal sector not in the selected valley and number of sectors in the valley is lower than $m = 8$ (i.e., a narrow valley) $\rightarrow k_{sol} = \frac{k_i+k_j}{2}$ where k_i and k_j are the extreme sectors of the valley.

- The result is a sector k_{sol} whose bisector angle value is the direction solution for the direction to move θ_{sol}
- The linear velocity v is set inversely proportional to the distance to the closest obstacle.

✓ The **control** is $u_t = (v_{sol}, \theta_{sol}) \rightarrow (v_{sol}, \omega_{sol} = \dot{\theta}_{sol})$

Dynamic Window Adaptation (DWA): velocity space

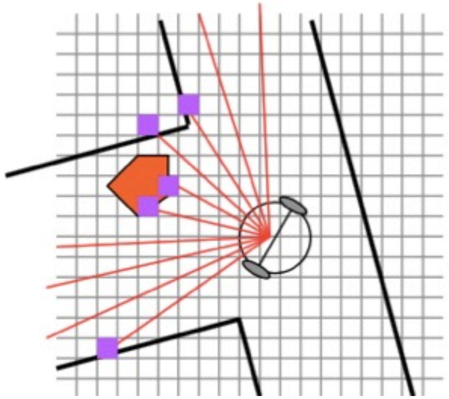
Basic ideas:

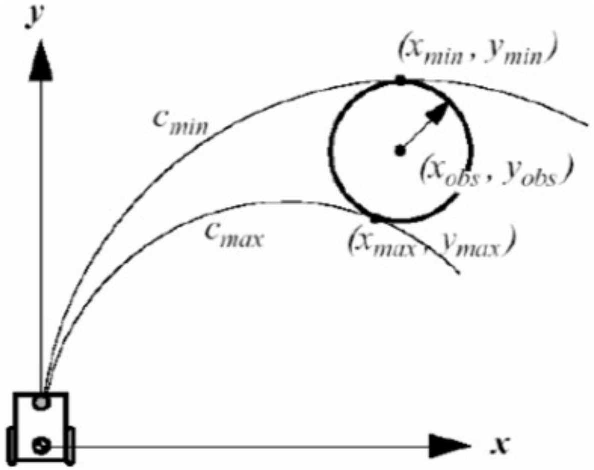
- Robot instantaneously moves over **circular trajectories**
- Radius is defined by $c = \frac{\omega}{v}$
- What are the **velocities** that determine obstacle-free and short circular trajectories (toward target)?
- → Work in **velocity space!**

Obstacle Avoidance: Basic Curvature Velocity Methods (CVM)

Simmons et al.

- Adding *physical constraints* from the robot and the environment on the *velocity space* (v, ω) of the robot
  - *Assumption that robot is traveling on arcs* ($c = \omega / v$)
  - *Constraints:* $-v_{max} < v < v_{max}$ $\quad -\omega_{max} < \omega < \omega_{max}$
  - *Obstacle constraints: Obstacles are transformed in velocity space*
  - *Objective function used to select the optimal speed*



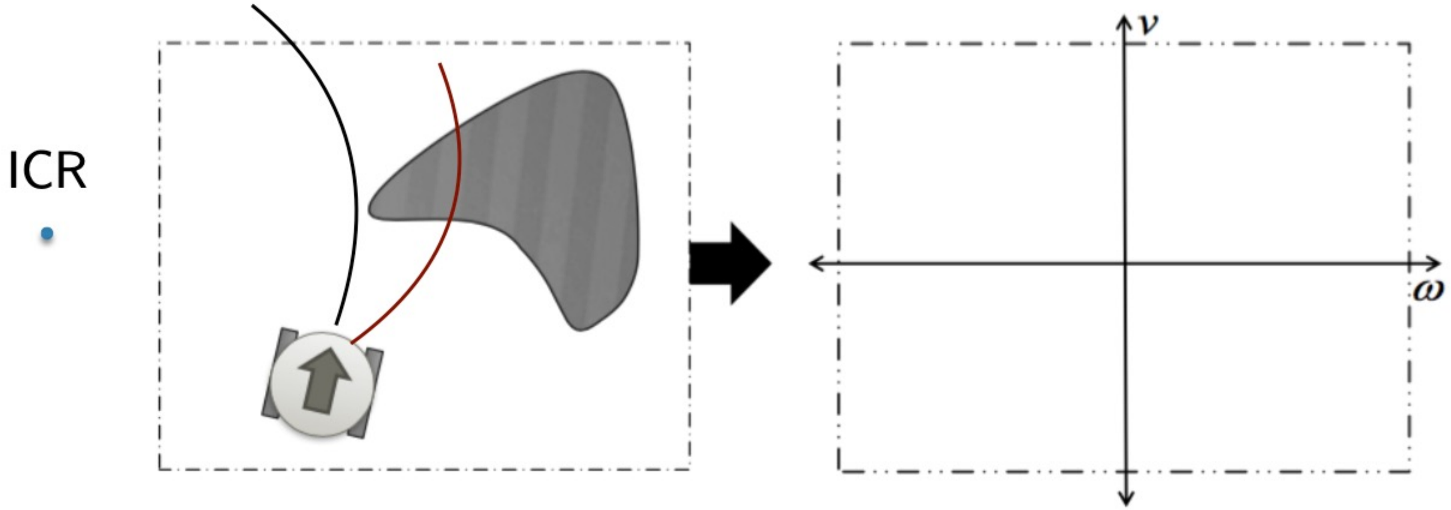


Dynamic Window Adaptation (DWA): velocity space

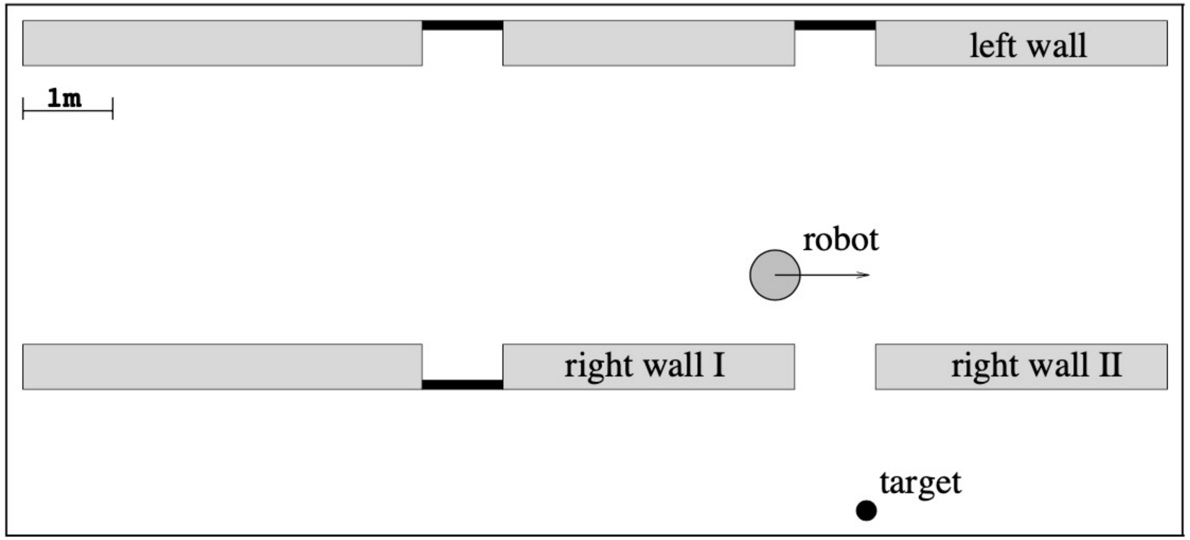
Dynamic Window Approach (DWA, 1987)

- Robot is assumed to instantaneously move on circular arcs (v, ω)
- 2D evidence grid is transformed into (v, ω) input-space based on robot deceleration capabilities / kino-dynamics, leading to V_o
- Static window V_s constrains velocities
- Dynamic window V_d accounts for vehicle dynamics
- Selection of (v, ω)-pair within $V_r = V_o \cap V_s \cap V_d$ maximizing objective containing heading, distance to goal and velocity terms

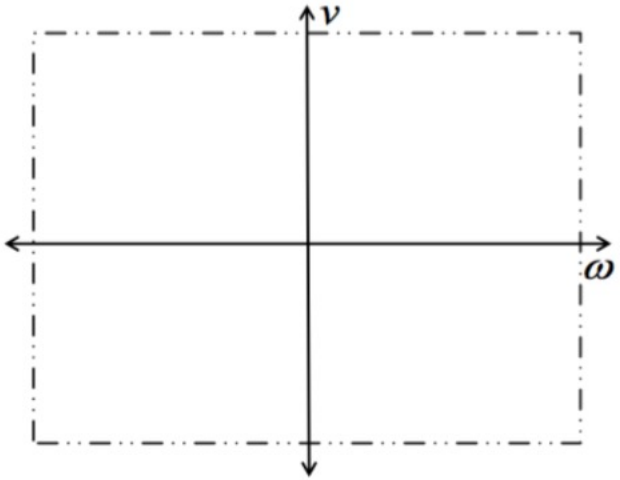
D. Fox, W. Burgard, S. Thrun, The Dynamic Window Approach to Collision Avoidance, IEEE Robotics & Automation Magazine 4(1):23 - 33 · April 1997



Dynamic Window Adaptation (DWA): Admissible velocities



V_s= Space of possible velocities for the robot



- Robot move with a curvature defined by (v, ω)
- $d(v, \omega)$ = closest distance to an obstacle on the corresponding curvature
- **Admissible velocity** (v, ω): the robot can stop before hitting the obstacle
- $\dot{v}_b, \dot{\omega}_b$ maximum accelerations ($\pm$) available for **breakage**

Admissible velocities V_a:

$$V_a = \left\{(v, \omega) \mid v \leq \sqrt{2 \cdot \mathrm{dist}(v, \omega) \cdot \dot{v}_b} \;\wedge\; \omega \leq \sqrt{2 \cdot \mathrm{dist}(v, \omega) \cdot \dot{\omega}_b}\right\}$$

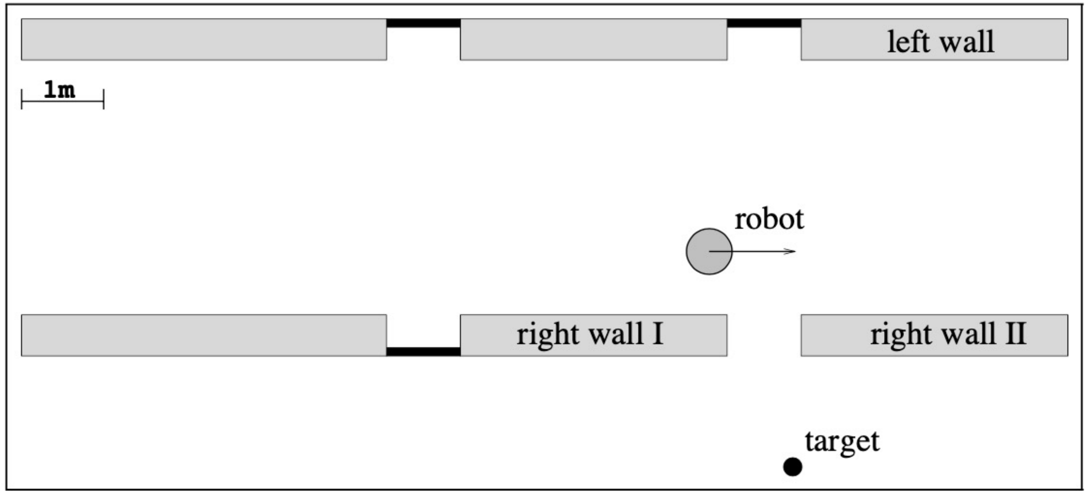
Dynamic Window Adaptation (DWA): Admissible velocities

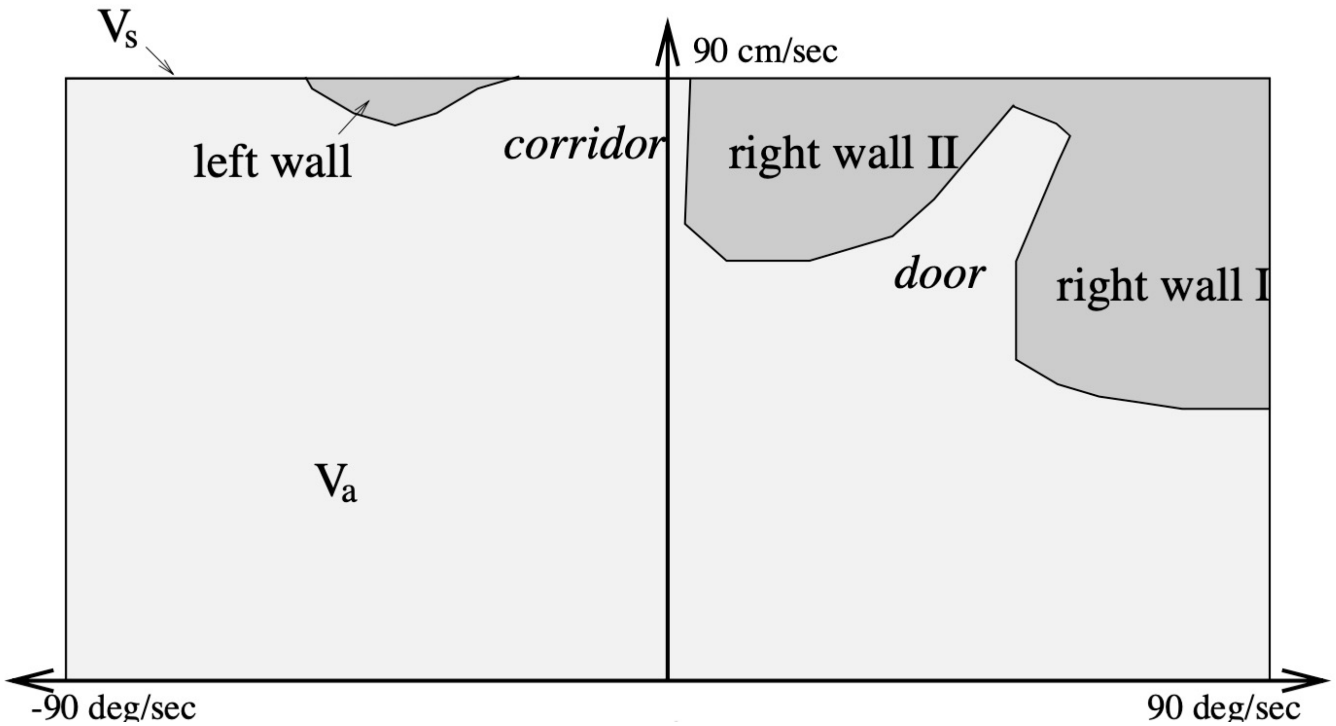
Admissible velocities V_a: $$V_a = \left\{(v,\omega) \mid v \le \sqrt{2 \cdot \mathrm{dist}(v,\omega) \cdot \dot{v}_b} \wedge \omega \le \sqrt{2 \cdot \mathrm{dist}(v,\omega) \cdot \dot{\omega}_b}\right\}$$

From kinematics:

- Assuming a constant acceleration, distance d traveled in time interval t (from $t = 0$) is $d = \frac{v_f - v_i}{2} t$ where v_i is the initial velocity, v_f is the final velocity, and $\frac{v_f - v_i}{2}$ is the average velocity
- It is also true that $v_f = v_i + at$, where a is the (constant) velocity in the interval
- Substituting v_f in $d = \frac{v_f - v_i}{2} t$ and making a few additional operations:
$$v_f = v_i^2 + 2ad$$
- In our case, v_f must be 0
- $0 = v_i^2 + 2ad$
- $v_a = \sqrt{2da}$

DWA: Admissible velocities



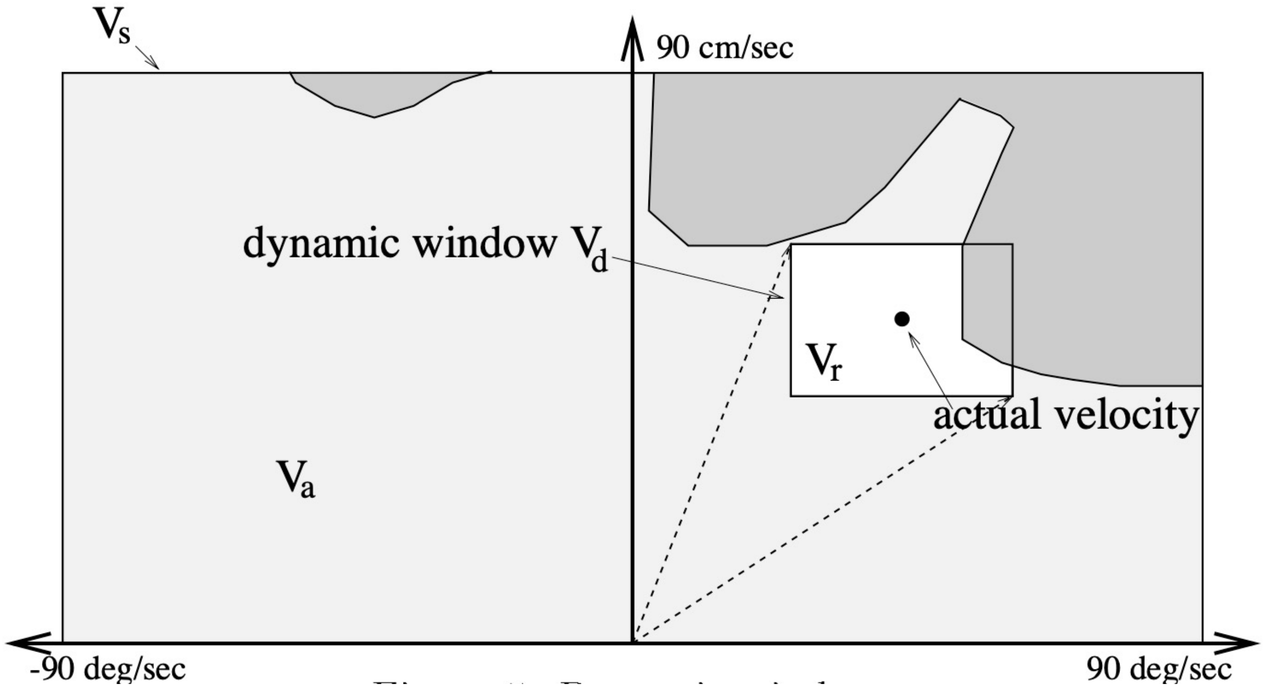


Example 1 *Again consider the example given in Figure 2. Figure 4 shows the velocities admissible in this situation given the accelerations $\dot{v}_b = 50$ cm/sec^2 and $\dot{\omega}_b = 60$ deg/sec^2. The non-admissible velocities are denoted by the dark shaded areas. For example all velocities in area* right wall II *would cause a sharp turn to the right and thus cause the robot to collide with the right wall in the example situation. The non-admissible areas are extracted from real world proximity information; in this special case this information was obtained from sonar sensors (see Section 5).*

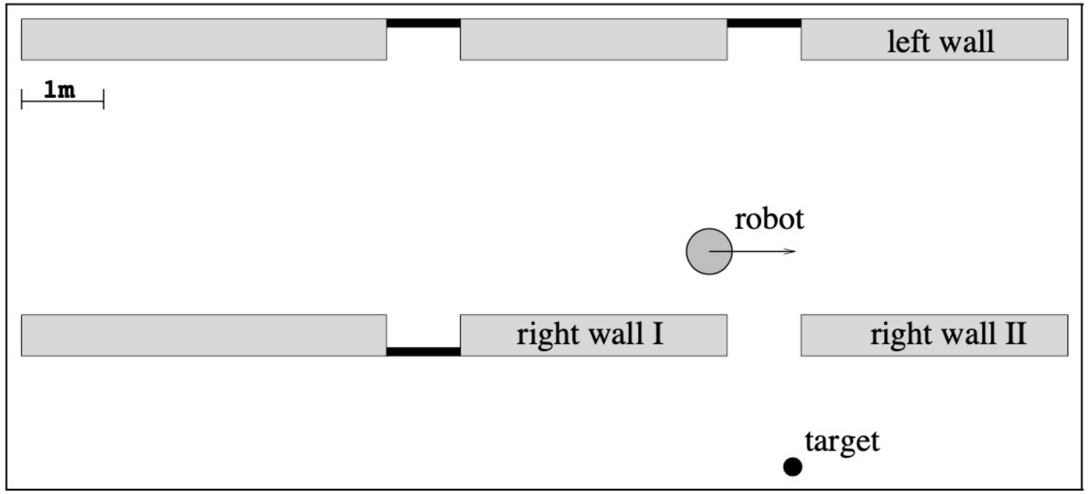
DWA: Dynamic window

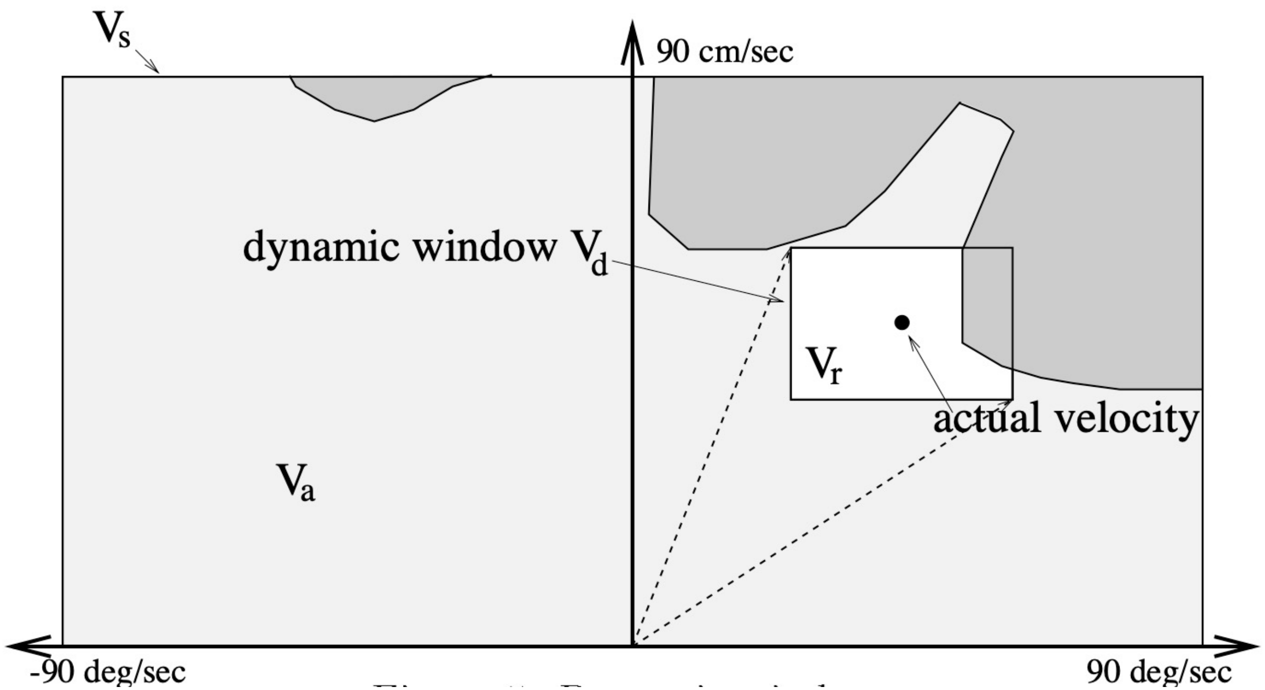
- Not all velocities can be reached, we can restrict to what velocities can be reached in the next time window
  - $t =$ Time interval during which the accelerations $\dot{v}, \dot{\omega}$ will be applied
  - (v_a, ω_a) = actual velocity

Dynamic window velocities, V_d: $$V_d = \{(v, \omega) \mid v \epsilon [v_a - \dot{v} \cdot t, v_a + \dot{v} \cdot t] \wedge \omega \epsilon [\omega_a - \dot{\omega} \cdot t, \omega_a + \dot{\omega} \cdot t]\}$$



DWA: Dynamic window





Example 2 *An exemplary dynamic window obtained in the situation shown in Figure 2 given accelerations of 50 cm/sec² and 60 deg/sec² and a time interval of 0.25 sec is shown in Figure 5. The two dotted arrows pointing to the corners of the rectangle denote the most extreme curvatures that can be reached.*

DWA: Set of velocities and Objective function

Set of velocities: $V_r = V_s \cap V_a \cap V_d$

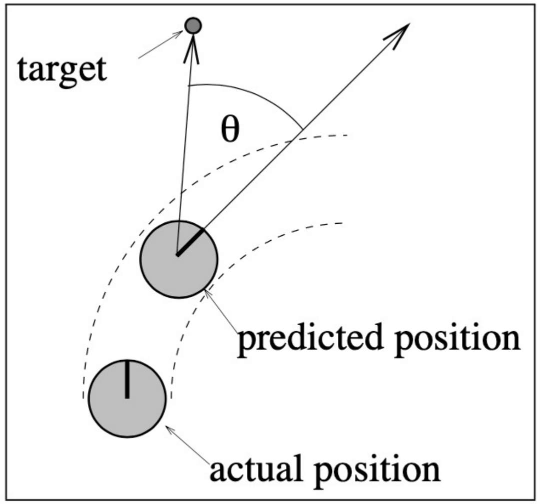
Best velocities: max over the cost function

$$G(v,\omega) = \sigma(\alpha \cdot \text{heading}(v,\omega) + \beta \cdot \text{dist}(v,\omega) + \gamma \cdot \text{velocity}(v,\omega))$$

Measure alignment with target: get to goal!

Measure Clearance: distance to closest obstacle on circular path → avoid obstacles

Robot velocity: move fast!



Use kinematics equations!

DWA: In practice

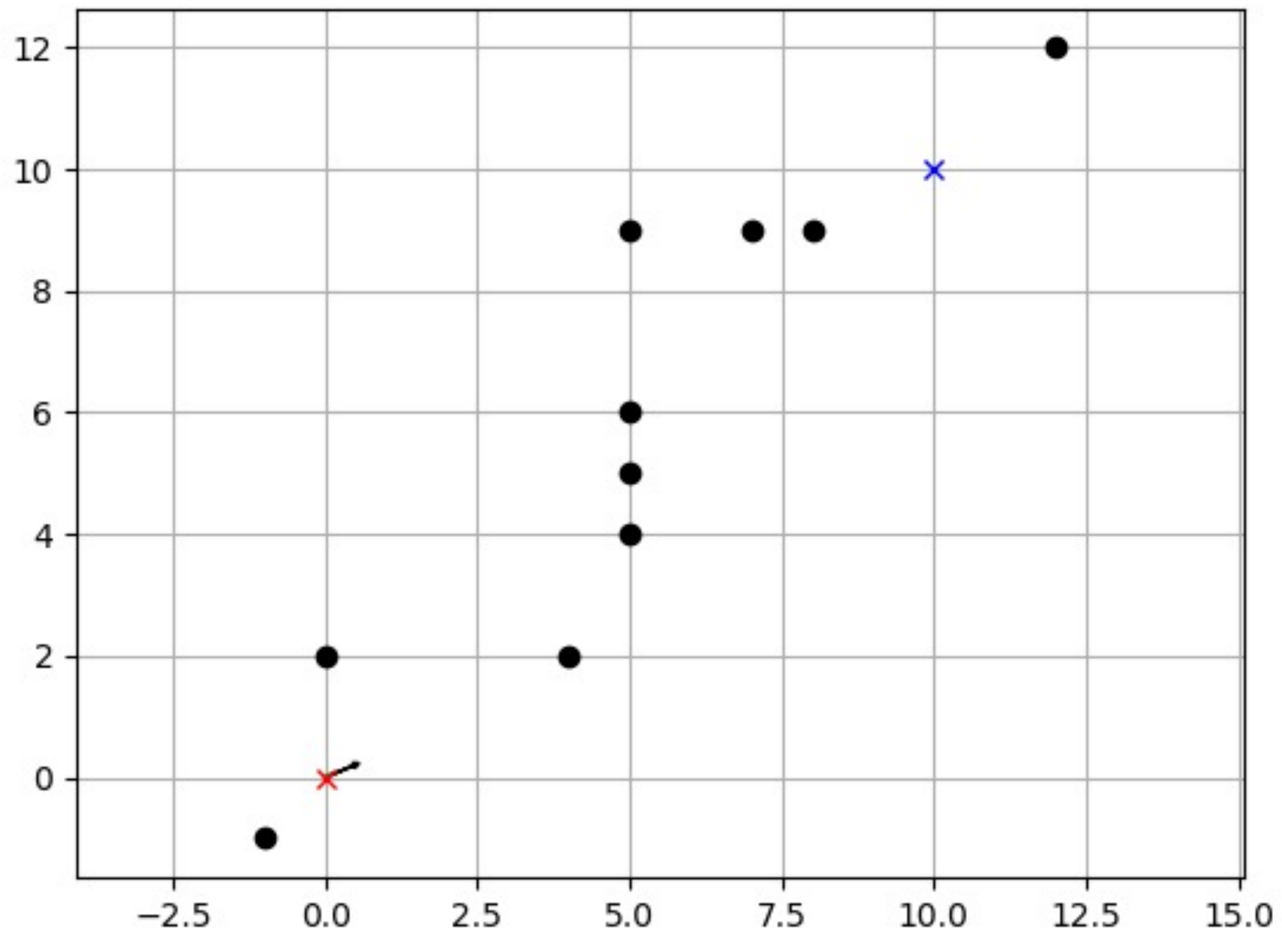
$$V_r = V_s \cap V_a \cap V_d$$

$$G(v,\omega) = \sigma(\alpha \cdot \mathrm{heading}(v,\omega) + \beta \cdot \mathrm{dist}(v,\omega) + \gamma \cdot \mathrm{velocity}(v,\omega))$$

Algorithm 1 DWA pseudocode

```
function DWA(robotPose, robotGoal, robotModel)
    laserscan ← readScanner()
    (v_allowable, w_allowable) ← generateWindow(robotVW, robotModel)
    for (each v in v_allowable) do
        for (each w in w_allowable) do
            dist ← findDist(v, w, laserscan, robotModel)
            breakDist ← calculateBreakingDistance(v)
            if (dist > breakDist) then
                cost ← costFunction
                if (cost > optimal) then
                    best_v ← v
                    best_w ← w
                    optimal ← cost
    return best_v, best_w
```

DWA in action



DWA in action: Problems with narrow passage, dependence on V_d

Read more on the reference paper!

D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance", *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23-33, March 1997.