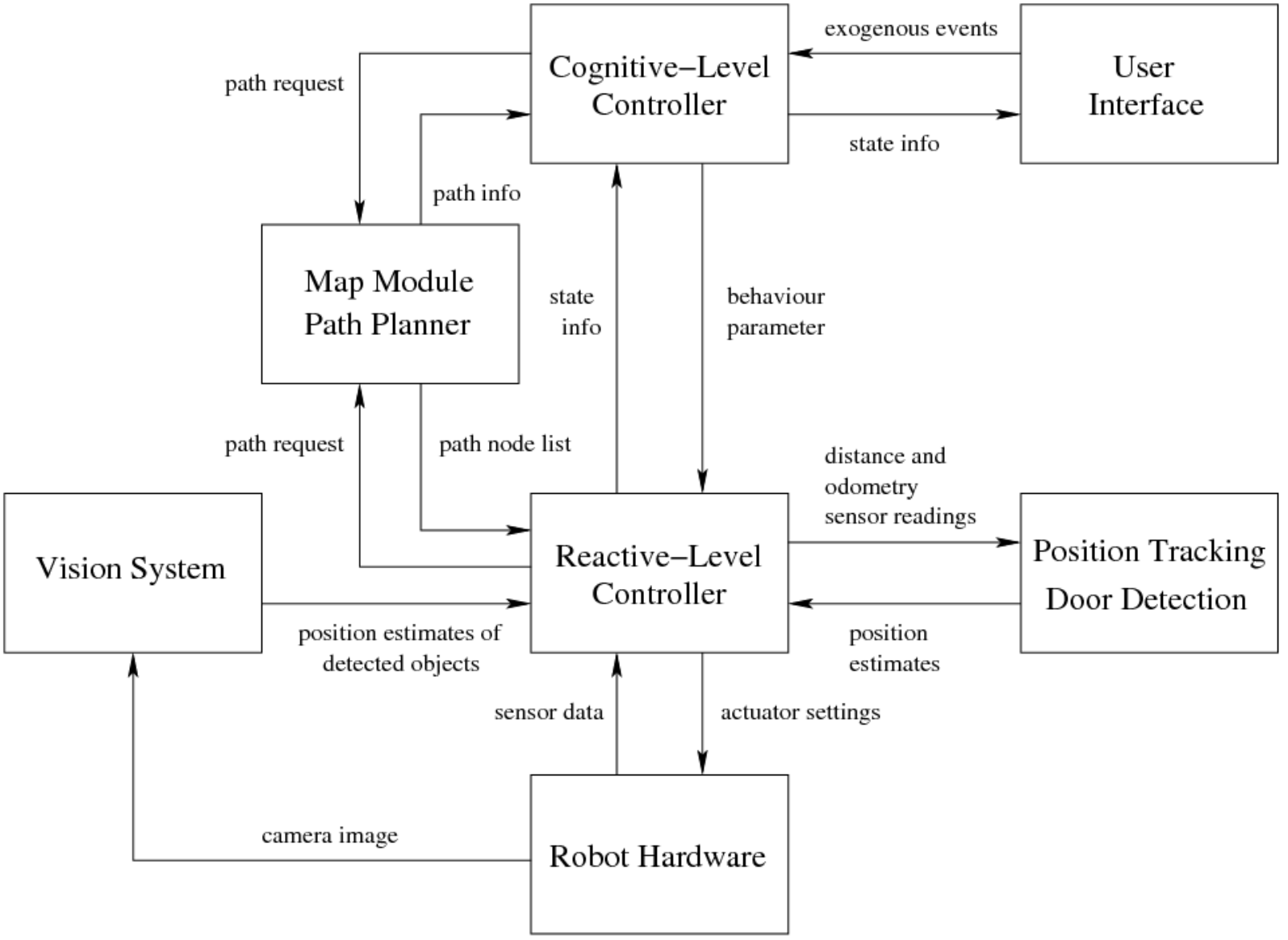
Robot Control Architectures

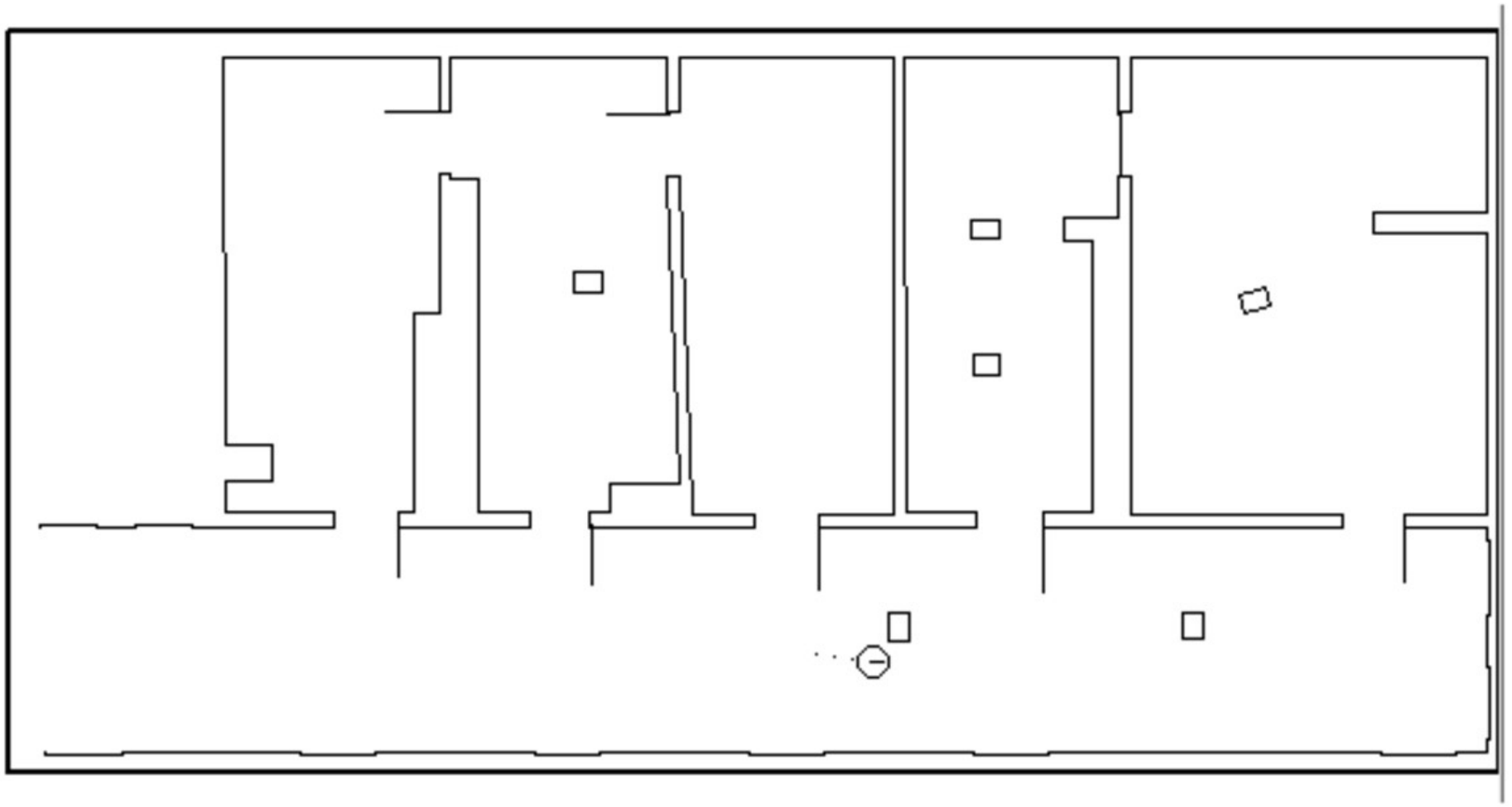


Low-Level Robotics

- D. Fox etal.
  “Markov localization for mobile robots in dynamic environments.”
  Journal of Artificial Intelligence Research 11:391-427, 1999. vol 1600. 1999.

- S. Thrun etal.
  “Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva.”
  International Journal of Robotics Research 19(11), 2000.

Maps: Known Obstacles



Localization

- Local localization methods: keeping track of position

  - Initial position must be known (approximately)

  - Cannot recover if track is lost

- Global localization methods, can deal with

  - Wake-up robot problem

  - Kidnapped robot problem

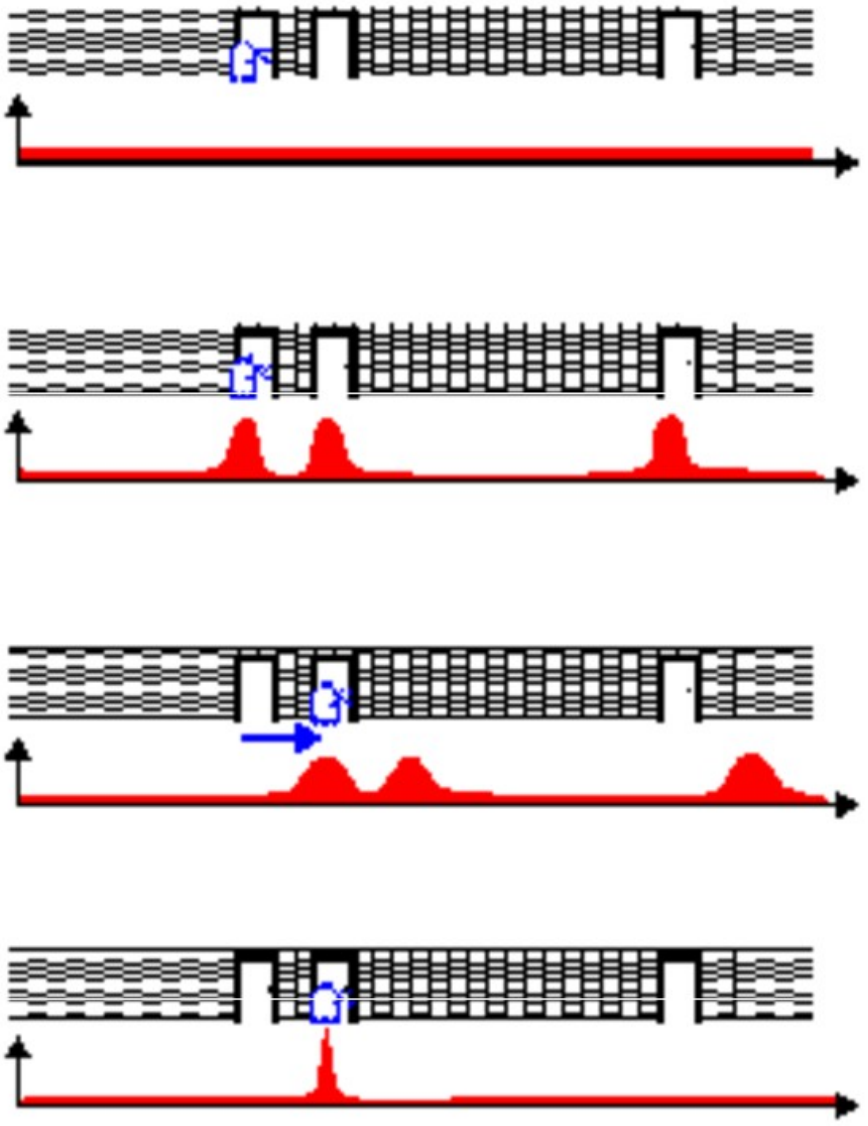
- Probabilistic position estimation

  - Uniform distribution: complete uncertainty

  - Multi-modal distribution: ambiguous situation

  - Unimodal distribution: high certainty about location

Probabilistic Position Estimation



Bayesian Localization Method

Robot perceives a stream of sensor measurements and odometry readings

$$\vec{d} = d_{0,} d_{1,} \ldots, d_T$$

Random variable L_t for robot's location $l = \langle x, y, \theta \rangle$

Localization means to estimate the posterior distribution

$$P(L_t = l | \vec{d})$$

Markov assumption

Given the robot's position, future measurements are independent of past measurements

$$P(d_{t+1} | L_t = l, d_{0,} \ldots, d_t) = P(d_{t+1} | L_t = l)$$

Case 1: Sensor Measurement $d_T = s_T$

Let $\vec{d} = d_{0,} d_{1,} \ldots, d_T$. According to Bayes' Rule the position estimation is

$$P(L_T = l | \vec{d}, s_T) = \frac{P(s_T | \vec{d}, L_T = l) \cdot P(L_T = l | \vec{d})}{P(s_T | \vec{d})}$$

By the Markov assumption,

$$P(L_T = l | \vec{d}, s_T) = \frac{P(s_T | L_T = l) \cdot P(L_T = l | \vec{d})}{P(s_T | \vec{d})}$$

Denominator is a normalization constant since it does not depend on L_T, hence,

$$P(L_T = l | \vec{d}, s_T) = \alpha_T \cdot P(s_T | L_T = l) \cdot P(L_T = l | \vec{d})$$

This allows to determine the position estimation recursively:

$$Bel(L_T = l) = \alpha_T \cdot P(s_T | l) \cdot Bel(L_{T-1} = l)$$

Case 2: Odometry Reading $d_T = a_T$

According to the Theorem of Total Probability the position estimation is

$$P(L_T = l|\vec{d}, a_T) = \int P(L_T = l|\vec{d}, L_{T-1} = l') \cdot P(L_{T-1} = l'|\vec{d}, a_T) dl'$$

By the Markov assumption,

$$P(L_T = l|\vec{d}, a_T) = \int P(L_T = l|a_T, L_{T-1} = l') \cdot P(L_{T-1} = l'|\vec{d}, a_T) dl'$$

Since α_T does not carry any information about the position L_{T-1},

$$P(L_T = l|\vec{d}, a_T) = \int P(L_T = l|a_T, L_{T-1} = l') \cdot P(L_{T-1} = l'|\vec{d}) dl'$$

This allows to determine the position estimation recursively:

$$Bel(L_T = l) = \int P(l|a_T, l') \cdot Bel(L_{T-1} = l') dl'$$

An Idealistic Iterative Localization Algorithm

Motion model $P(l|a,l')$

Perceptual model $P(s|l)$

- $P(L_0 = l)$ uniformly distributed if initial position is unknown
- $P(L_0 = l)$ Gaussian distribution if initial position approximately known

1. For each location l do $Bel(L_0 = l) := P(L_0 = l)$
2. If new input s_T is received then for each location l do

$$\widehat{Bel}(L_T = l) := P(s_T|l) \cdot Bel(L_{T-1} = l)$$

$$Bel = Normalize(\widehat{Bel})$$

3. If new input a_T is received then for each location l do

$$Bel(L_T = l) := \int P(l|a_T, l') \cdot Bel(L_{T-1} = l')\, dl'$$

4. Goto 2

State Space Representations

- Kalman filtering asumes unimodal Gaussian distribution

  - cannot represent ambiguous situations (multi-modal distribution)

- Topological representation based on landmarks

  - smal state space

  - requires environments with landmarks

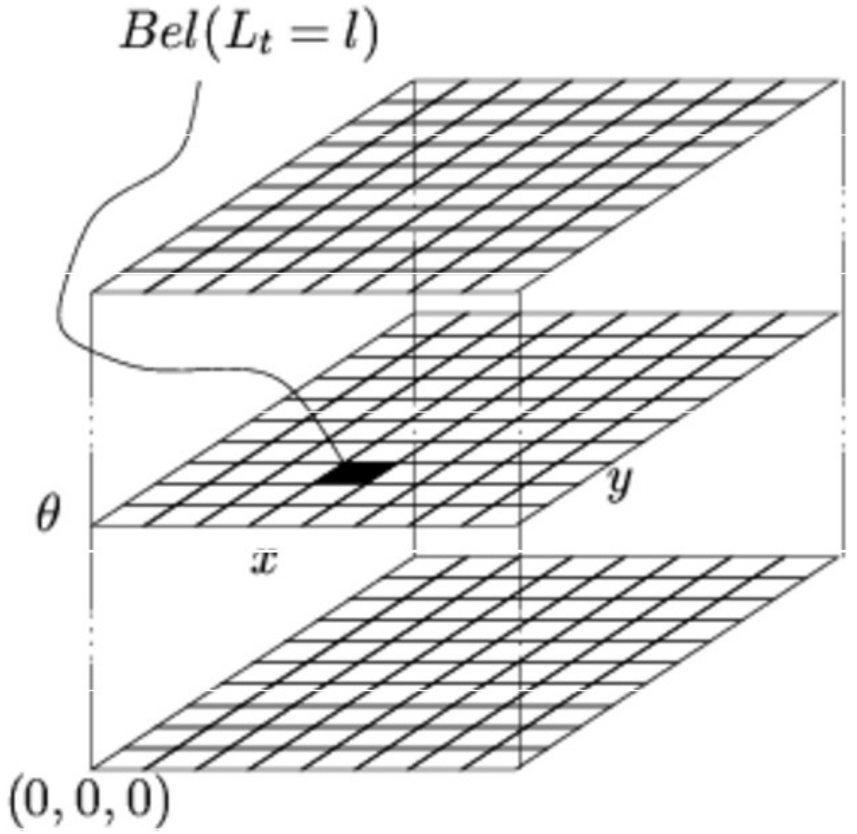
  - limited accuracy due to coarse resolution

- Grid-based representation

  - accuracy due to fine-grained discretization

  - huge state space

Grid-Based Representation



- Typical resolution: $\Delta x \times \Delta y = 15 \times 15$ cm^2, $\Delta\theta = 2^\circ$
- Size of state space in 30×30 m^2 environment: 7,200,000

Selective Update

Update only grid cells *l* with $Bel(L_t) > \varepsilon$ (typical threshold: $\varepsilon = 1\%$)

- Partition state space into segments $\pi_1, ..., \pi_n$

  Segment is passive if the probabilities of all locations are below the threshold.

- For all locations below the threshold, the update is appoximated by:

$$Bel(L_t = l) := a_t \cdot \tilde{P}(s_t) \cdot Bel(L_{t-1} = l)$$

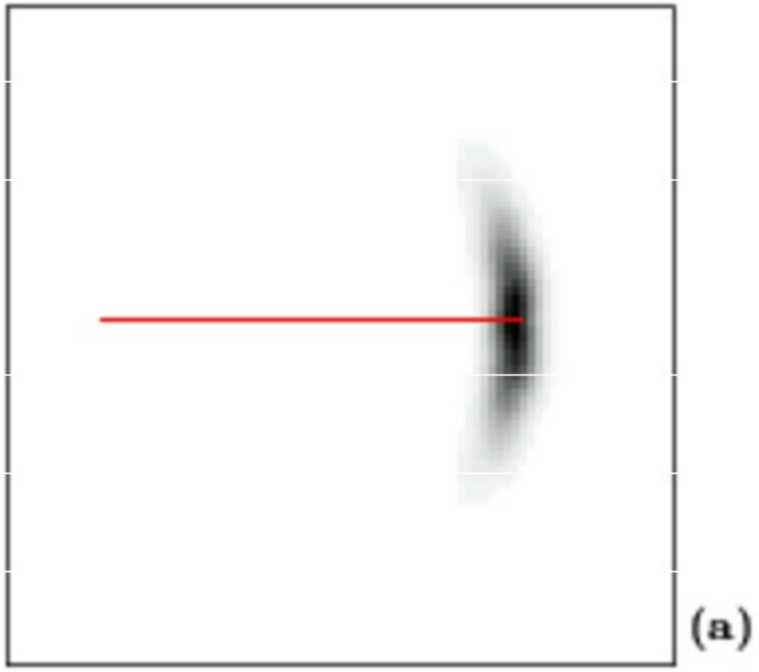
  where $\tilde{P}(s_t)$ is the a priori likelihood of measurement s_t

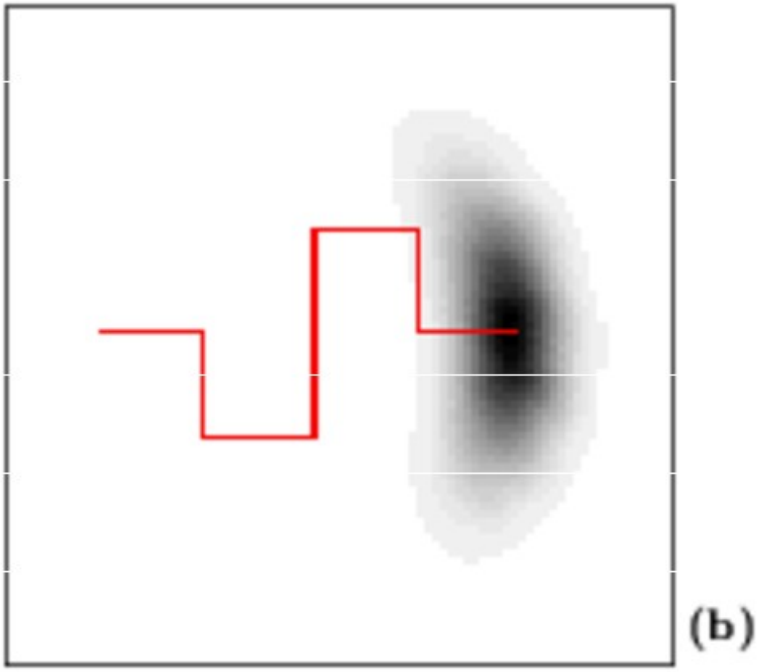
- For passive segments it suffices to accumulate $\tilde{P}(s_t)$ and the motion. These values are incorporated into the probabilities for individual cells only when a segment becomes active.

- To decide when a segment becomes active, store the maximal probability of all cells in it.

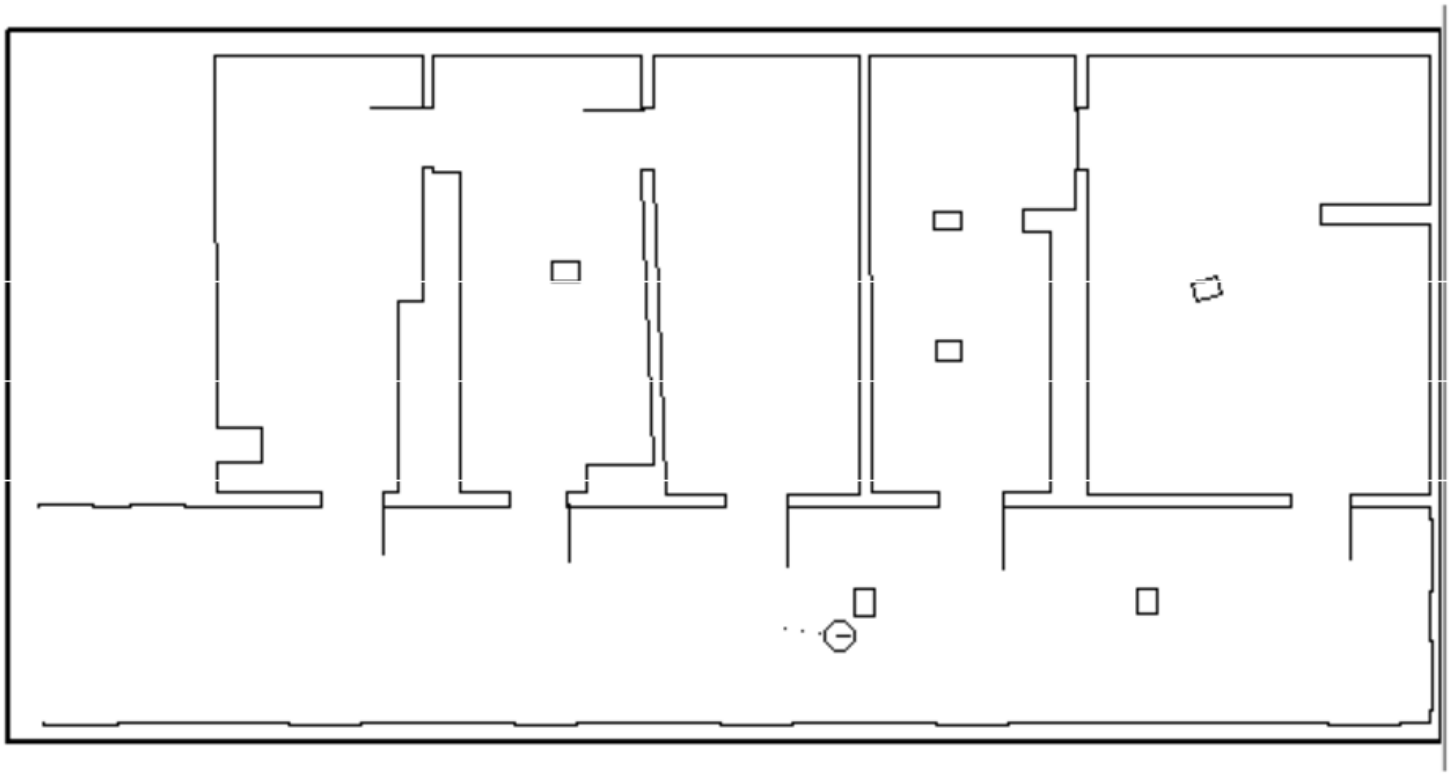
Motion Model $P(l|a, l')$

- Normally distributed errors in translation and rotation
- Two independent, zero-centered Gaussian distributions
- Variances are proportional to length of measured motion





Perceptual Model $P(s|l)$



- Proximity sensor (sonar or laser) measures distance to nearest obstacle
- Discretization of possible distances d_1, ..., d_n (typical values: Δd_i = 5 cm, d_n = 10 m)
- Perceptual model $P(d_i|l)$
- In dynamic environments there are known and unknown obstacles

Known Obstacles

- $P_m(d|l)$ probability of measuring d if beam reflected by closest (wrt. l) obstacle in the map
- o_l distance to this obstacle
- Gaussian distribution with mean o_l

$$P_m(d|l) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(d-o_l)^2}{2\sigma^2}}$$

- Standard deviation σ models uncertainty of measured distance, based on

  - granularity of state-space representation
  - accuracy of map
  - accuracy of sensor

Unknown Obstacles (People etc.)

- Unknown obstacles assumed to be equally distributed in the environment
- $P_u(d)$ probability of detecting unknown obstacle at distance d
- Geometric distribution

$$P_u(d_i) = \begin{cases} 0 & \text{if } i = 0 \\ c_r(1 - \sum_{j<i} P_u(d_j)) & \text{otherwise} \end{cases}$$

where

- $\sum_{j<i} P_u(d_j)$ probability that there is no closer unknown obstacle
- c_r probability that sensor is reflected by unknown obstacle at any range

Determining the Combined Probability $P(d_i|l)$

- Case (a): Sensor beam reflected by known obstacle at d_i, which requires that there is no closer unknown obstacle:

$$P_{(a)} = c_d P_m(d_i|l) \cdot \left(1 - \sum_{j<i} P_u(d_j)\right)$$

  where c_d probability that closest obstaclee in map is detected

- Case (b): Sensor beam reflected by unknown obstacle at d_i, which requires that there is no closer known or unknown obstacle:

$$P_{(b)} = c_r \cdot \left(1 - \sum_{j<i} P(d_j|l)\right)$$

- Combining the two cases:

$$P(d_i|l) = 1 - (1 - P_{(a)}) \cdot (1 - P_{(b)}) \qquad (i < n)$$

$$P(d_n|l) = 1 - \sum_{i<n} P(d_i|l) \qquad (\text{maximal sensor range } d_n)$$

Filtering Techniques for Crowded Environments

- Vicinity of unknown obstacles usually increases the belief of being close to known obstacles.
- Too many unknown obstacles lead to too many unexpected measurements.
- Non-Markovian environments:

  Sensor measurements depend not only on the current state (i.e., location)
- Filtering techniques to detect corrupted sensor readings (caused by dynamic obstacles)

Technique 1: Entropy Filter

- Idea: Filter out all measurements that increase uncertainty
- Entropy of belief over L defined as

$$H(L) = -\sum_l Bel(L=l) \cdot \log Bel(L=l)$$

(measure of uncertainty)

- Change of entropy given sensor reading s:

$$\Delta H(L|s) = H(L|s) - H(L)$$

- All sensor measurements with $\Delta H(L|s) > 0$ are filtered out
- Entropy filter works well if robot focuses on correct hypothesis.
- Entropy filter may fail if robot's belief is incorrect.

Technique 2: Distance Filter

- Idea: Filter out all distance measurements that are shorter than expected
- $P_{short}(d_i|l)$ probability that measured distance d_i shorter than known obstacle

$$P_{\text{short}}(d_i|l) = \sum_{j>i} P_m(d_j|l)$$

(i.e., probability that the expected (according to map) measurement is longer than d_i)

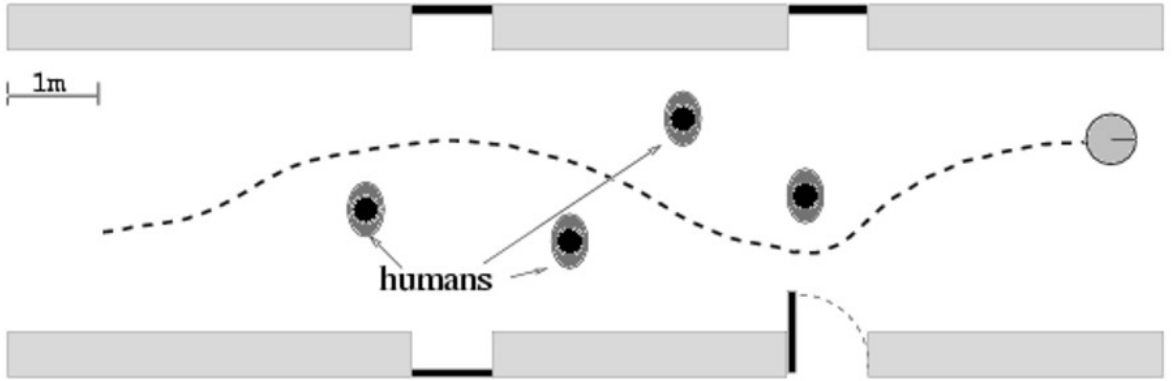
- Averaging over all possible positions:

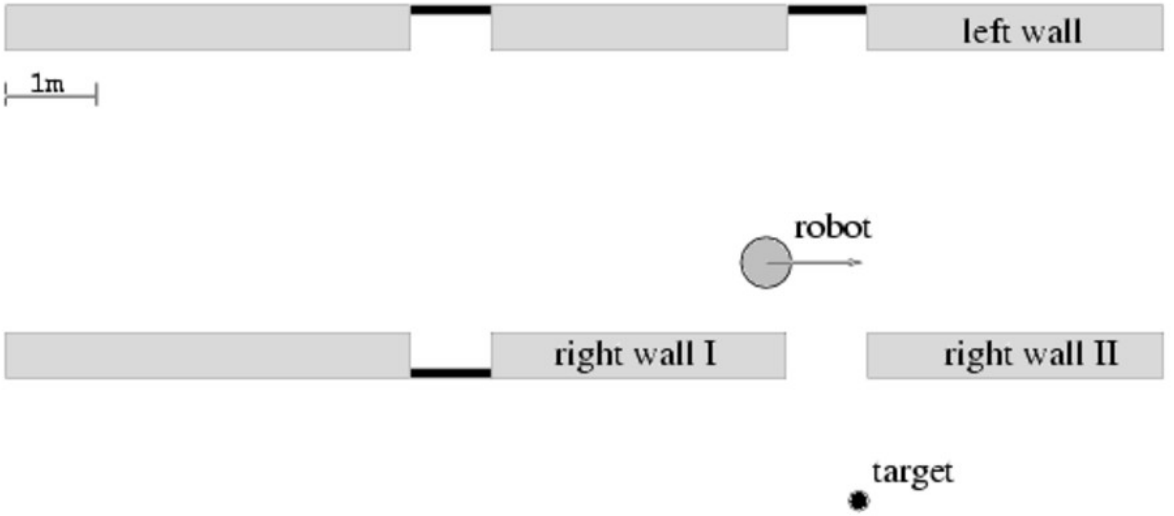
$$P_{\text{short}}(d_i) = \sum_{l} P_{\text{short}}(d_i|l) \cdot Bel(L=l)$$

- All sensor measurements d_i with $P_{short}(d_i) > \gamma$ are filtered out.

(typical threshold: γ = 99%)

Navigation


1m
humans


left wall
1m
robot
right wall I
right wall II
target

Motion Equations

- $x(t)$, $y(t)$, $\theta(t)$ location coordinates
- $v(t)$ translational velocity
- $w(t)$ rotational velocity

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot \cos\theta(t)\, dt$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot \sin\theta(t)\, dt$$

Acceleration (1)

- Translational and rotational velocities achieved by acceleration
- Acceleration commands given at times t_0, t_1, ..., t_n with $\Delta t := t_{i+1} - t_i$
- Acceleration $\dot{\upsilon}_i$ and $\dot{\omega}_i$ constant in time interval $[t_i, t_{i+1}]$
- In $[t_i, t_{i+1}]$, translational and rotational velocity are approximated by constant υ_i and ω_i

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v(t) \cdot \cos(\theta(t_i) + \omega_i \cdot (t - t_i))\, dt$$

$$y(t_n) = y(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v(t) \cdot \sin(\theta(t_i) + \omega_i \cdot (t - t_i))\, dt$$

Acceleration (2)

- Solving the integrals yields

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} F_x^i \quad \text{and} \quad y(t_n) = y(t_0) + \sum_{i=0}^{n-1} F_y^i$$

where

$$F_x^i = \begin{cases} \frac{v_i}{\omega_i} \cdot (\sin\theta(t_i) - \sin(\theta(t_i) + \omega_i \cdot \Delta t)) & \text{if } \omega_i \neq 0 \\ v_i \cdot \cos\theta(t_i) \cdot \Delta t & \text{if } \omega_i = 0 \end{cases}$$

$$F_y^i = \begin{cases} -\frac{v_i}{\omega_i} \cdot (\cos\theta(t_i) - \cos(\theta(t_i) + \omega_i \cdot \Delta t)) & \text{if } \omega_i \neq 0 \\ v_i \cdot \sin\theta(t_i) \cdot \Delta t & \text{if } \omega_i = 0 \end{cases}$$

- In case $\omega_i = 0$ the robot follows a straight line
- In case $\omega_i \neq 0$ the robot's trajectory describes a circle with radius $\frac{v_i}{\omega_i}$

Search Space of Possible Velocities

- Circular trajectories

  Two-dimensional search space of translational and rotational velocities (υ, ω)

- Admissible velocities

  Pair (υ, ω) admissible if robot is able to stop before reaching nearest obstacle

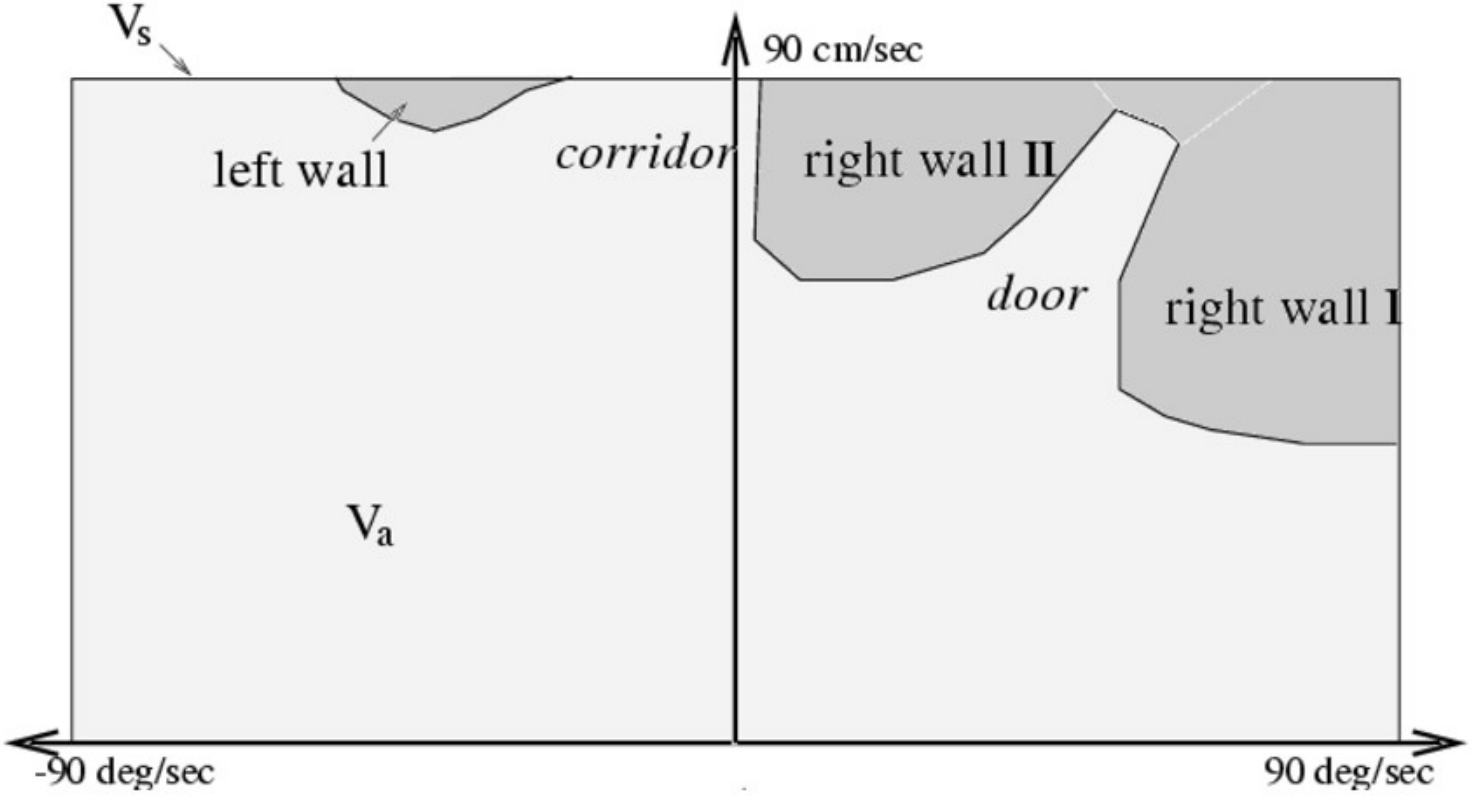
- Dynamic window

  Consider only admissible velocities that can be exerted within Δt

Admissible Velocities

- *dist*(υ, ω) distance to closest obstacle on trajectory (υ, ω)
- $\dot{\upsilon}_b, \dot{\omega}_b$ acceleration for breakage
- Admissible velocities

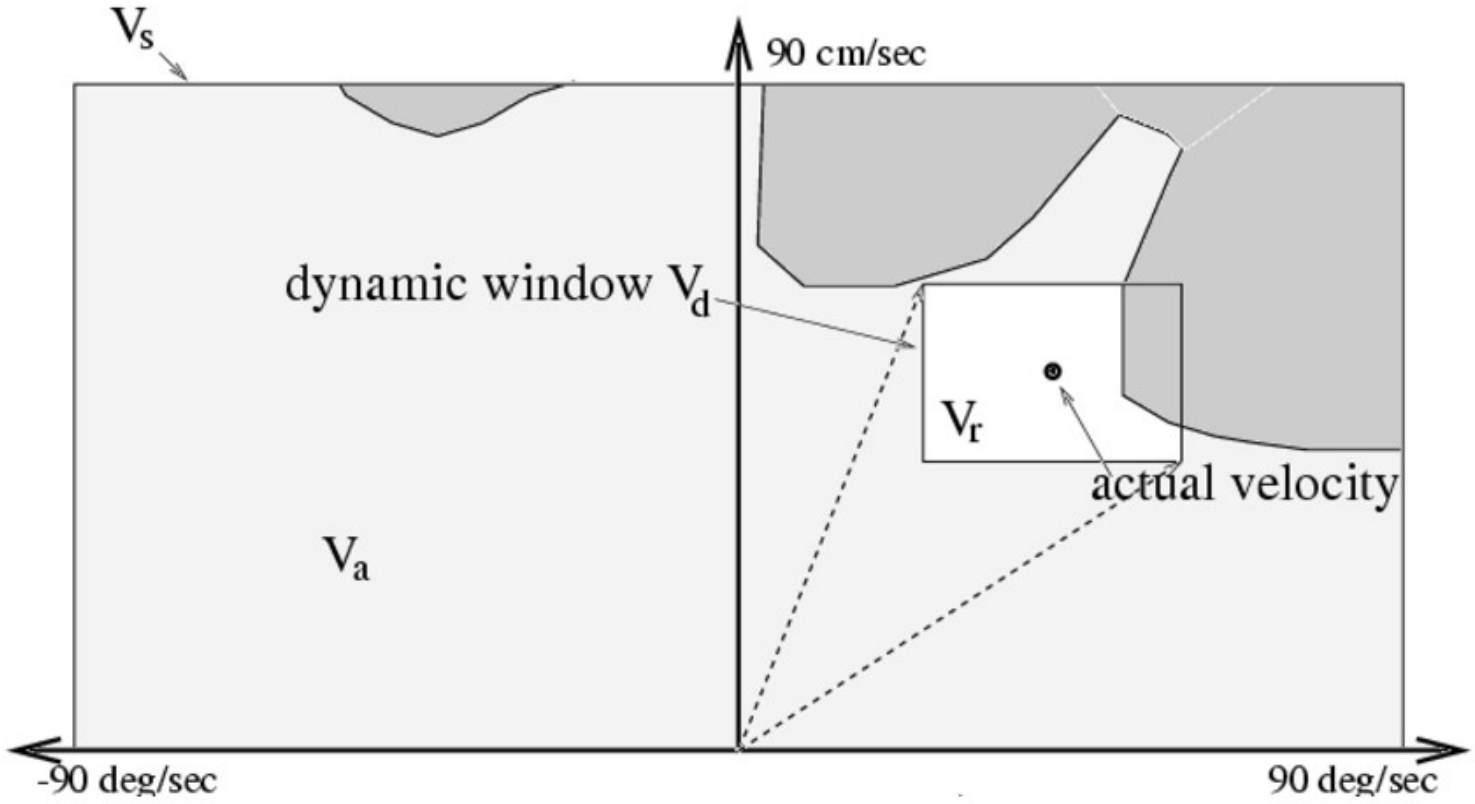
$$\{(\upsilon,\omega): \upsilon \le \sqrt{2 \cdot dist(\upsilon,\omega) \cdot \dot{\upsilon}_b},\ \ \omega \le \sqrt{2 \cdot dist(\upsilon,\omega) \cdot \dot{\omega}_b}\}$$



Dynamic Window

- $\dot{v}, \dot{\omega}$ accelerations that can be exerted
- v_a, ω_a actual translational and rotational velocity
- Dynamic window

$$\{(v,\omega): v \in [v_a - \dot{v}\cdot\Delta t, v_a + \dot{v}\cdot\Delta t],\ \omega \in [\omega_a - \dot{\omega}\cdot\Delta t, \omega_a + \dot{\omega}\cdot\Delta t]\}$$



Choosing the Optimal Velocity

Maximize the objective function

$$G(\upsilon, \omega) = \alpha \cdot heading(\upsilon, \omega) + \beta \cdot dist(\upsilon, \omega) + \gamma \cdot vel(\upsilon, \omega)$$

- Target heading

  Function *heading* is a measure of how directly the robot moves towards target location

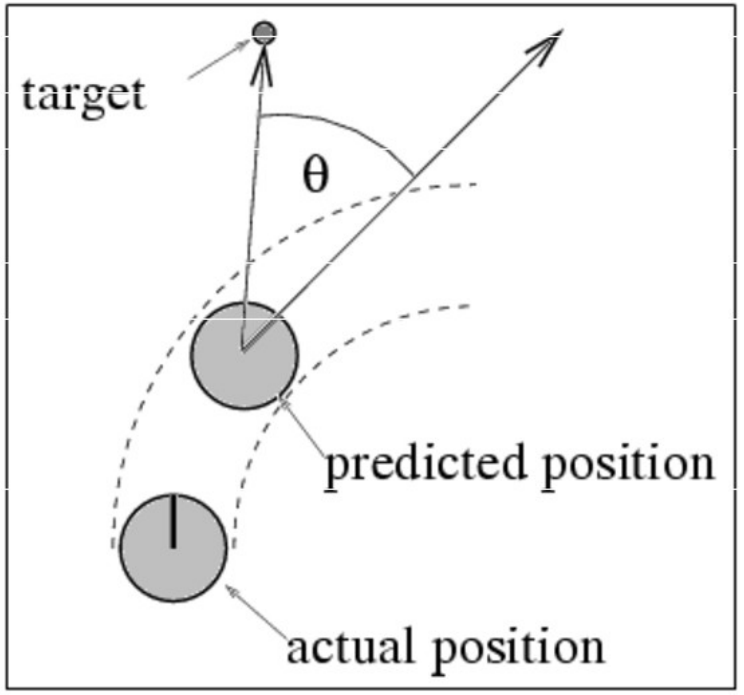
- Clearance

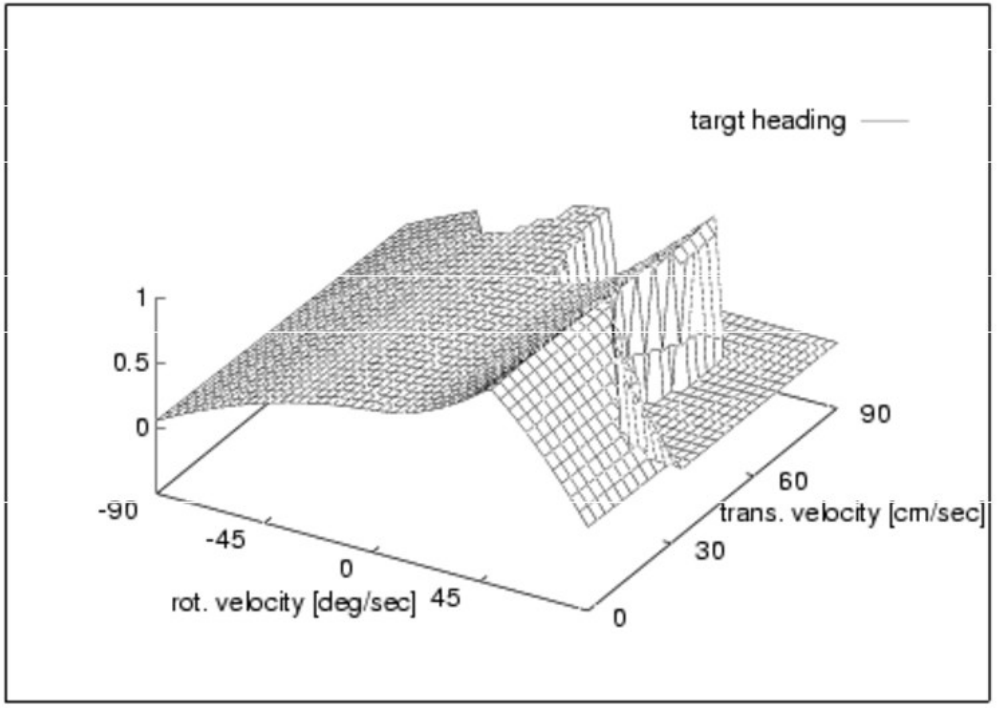
  Function *dist* is the distance of the robot to nearest obstacle

- Velocity

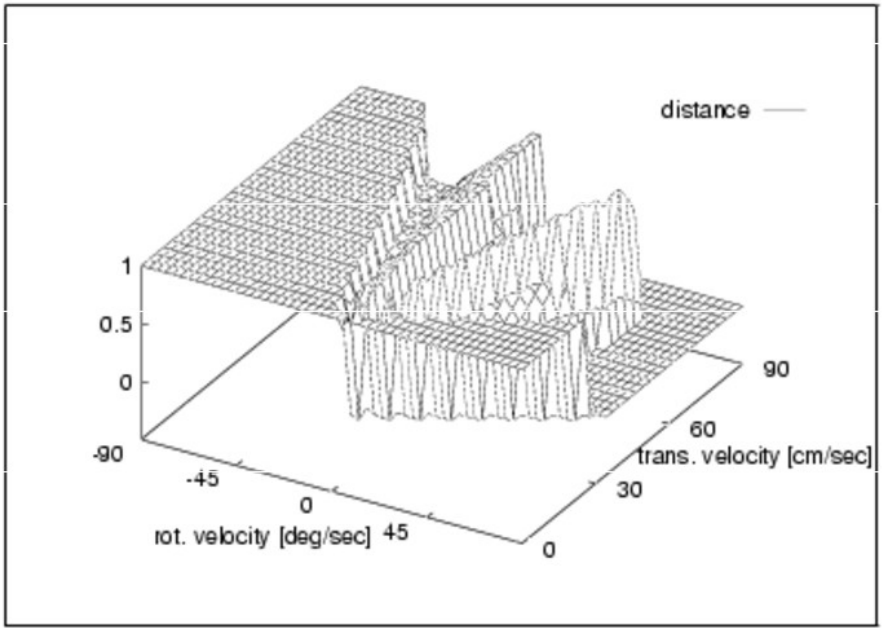
  Function *vel* is the forward velocity of the robot

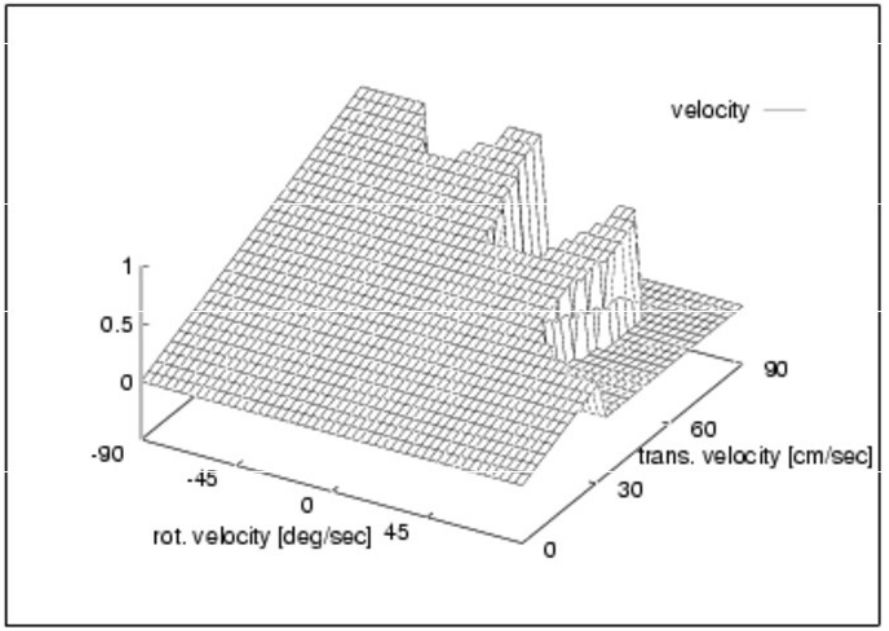
Target Heading



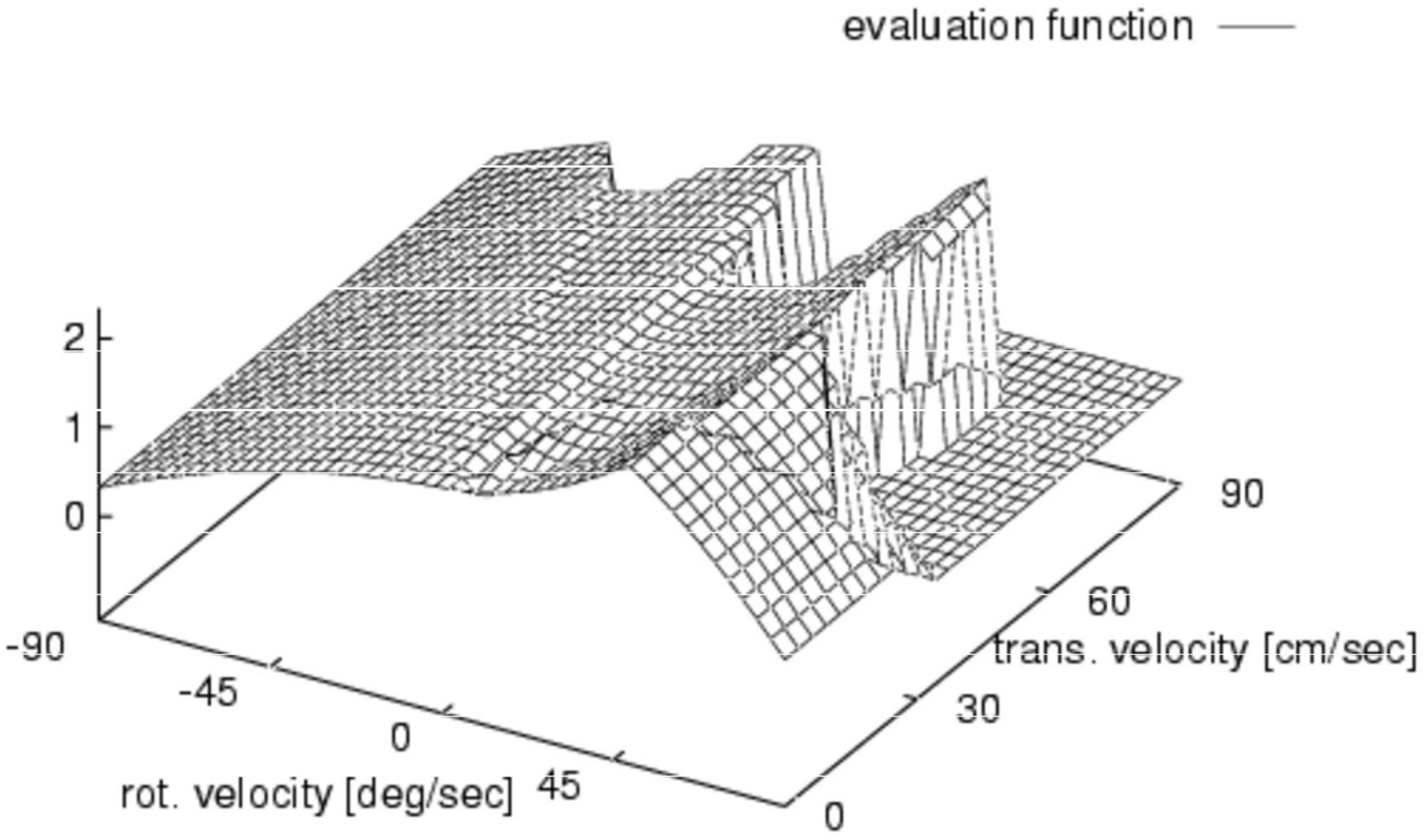


Clearance and Velocity

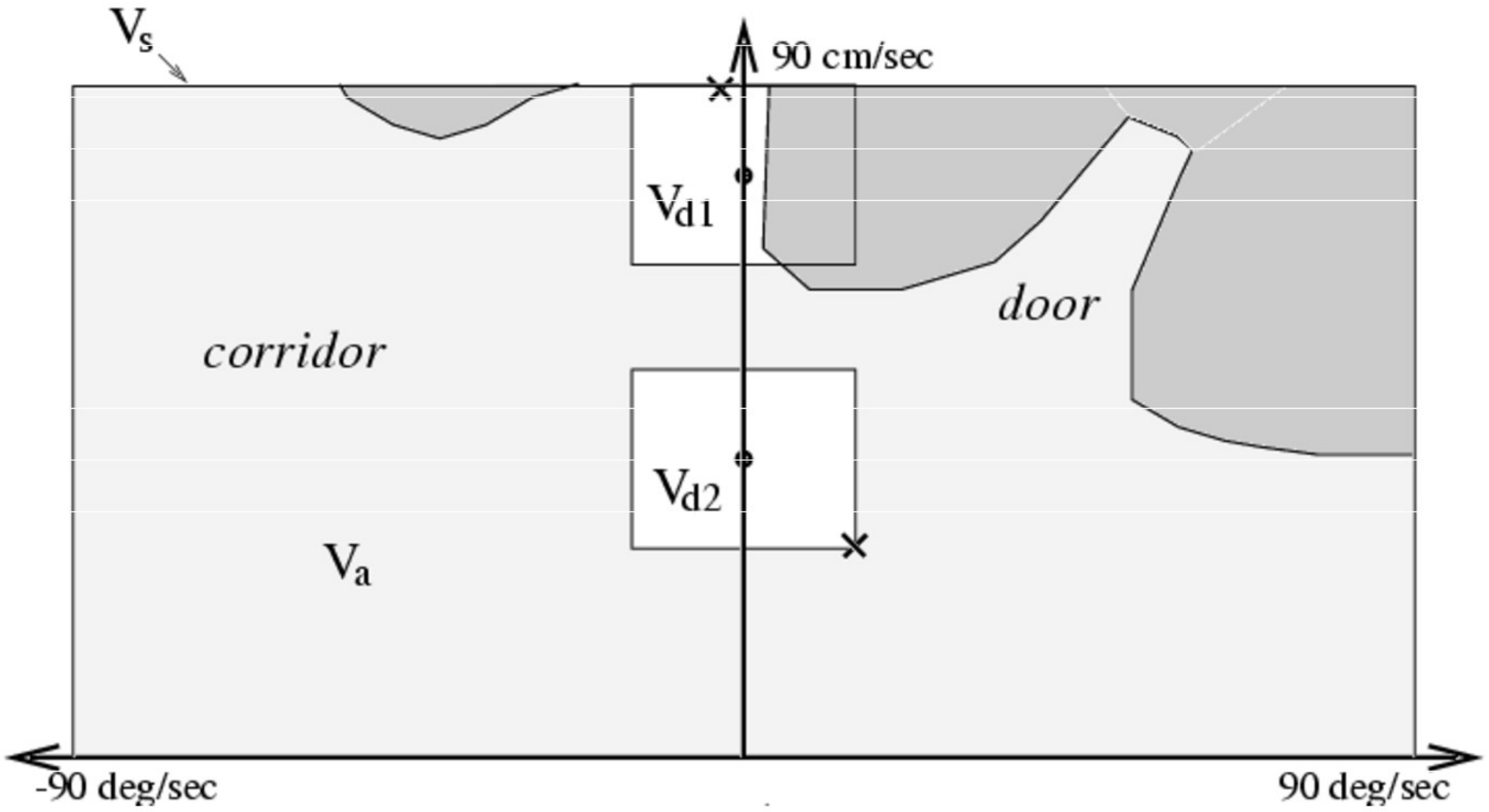




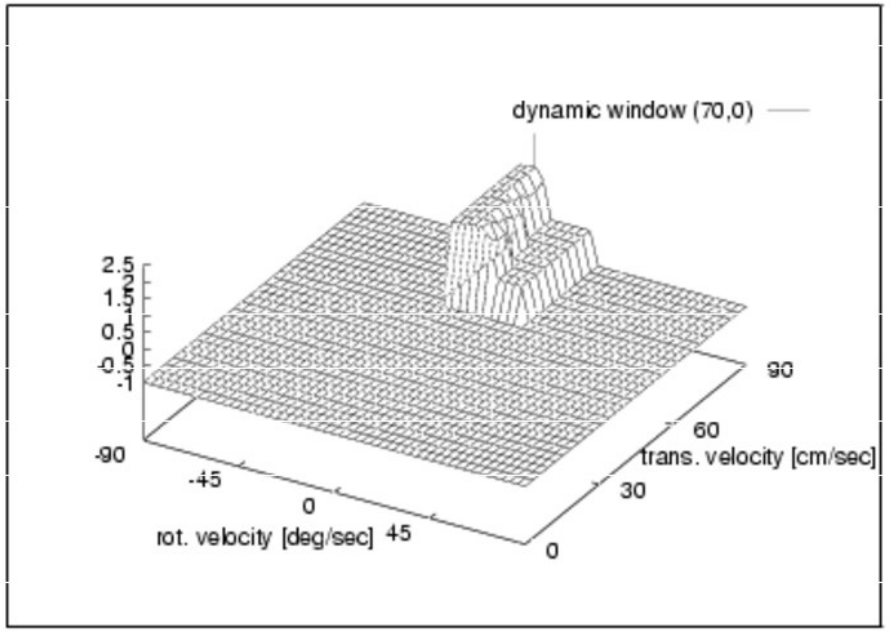
Resulting Objective Function (α = 2, β = γ = 0.2)

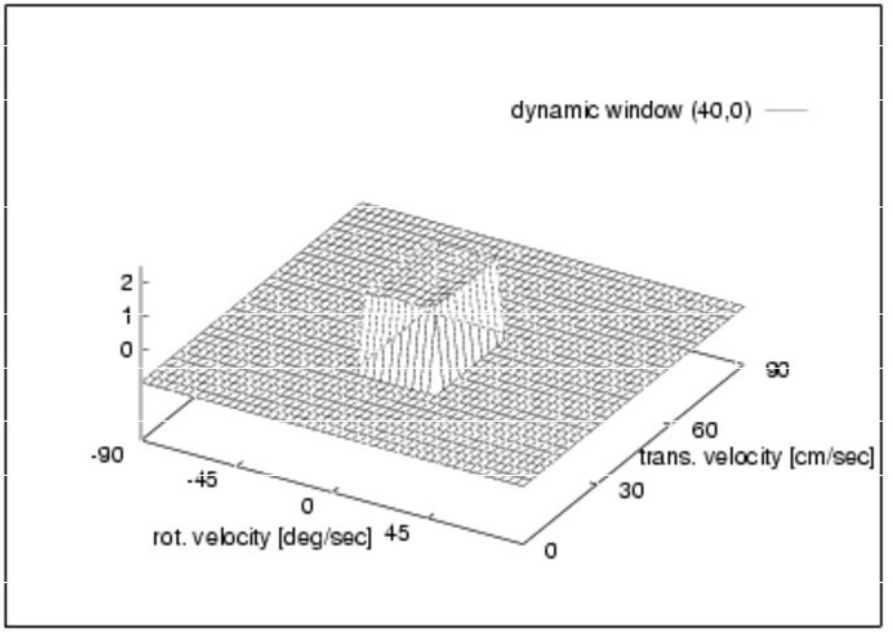


Examples for Different Current Velocities



Examples for Different Current Velocities: Objective Function





Invisible Obstacles

- Obstacles, such as forward escalators, may be invisible
- Invisible obstacles are marked in the map
- Requires to interleave obstacle avoidance with localization

~> In addition to the physical measurements of obstacles at a certain distance, a virtual range measurement is determined from the map.

Virtual Measurements of Invisible Obstacles

- $d_i^{\alpha}(l)$ distance to nearest (wrt. l) invisible obstacle in direction α

$$P(d_i^{\alpha}) = \sum_l P(d_i^{\alpha} | l) \cdot P(l)$$

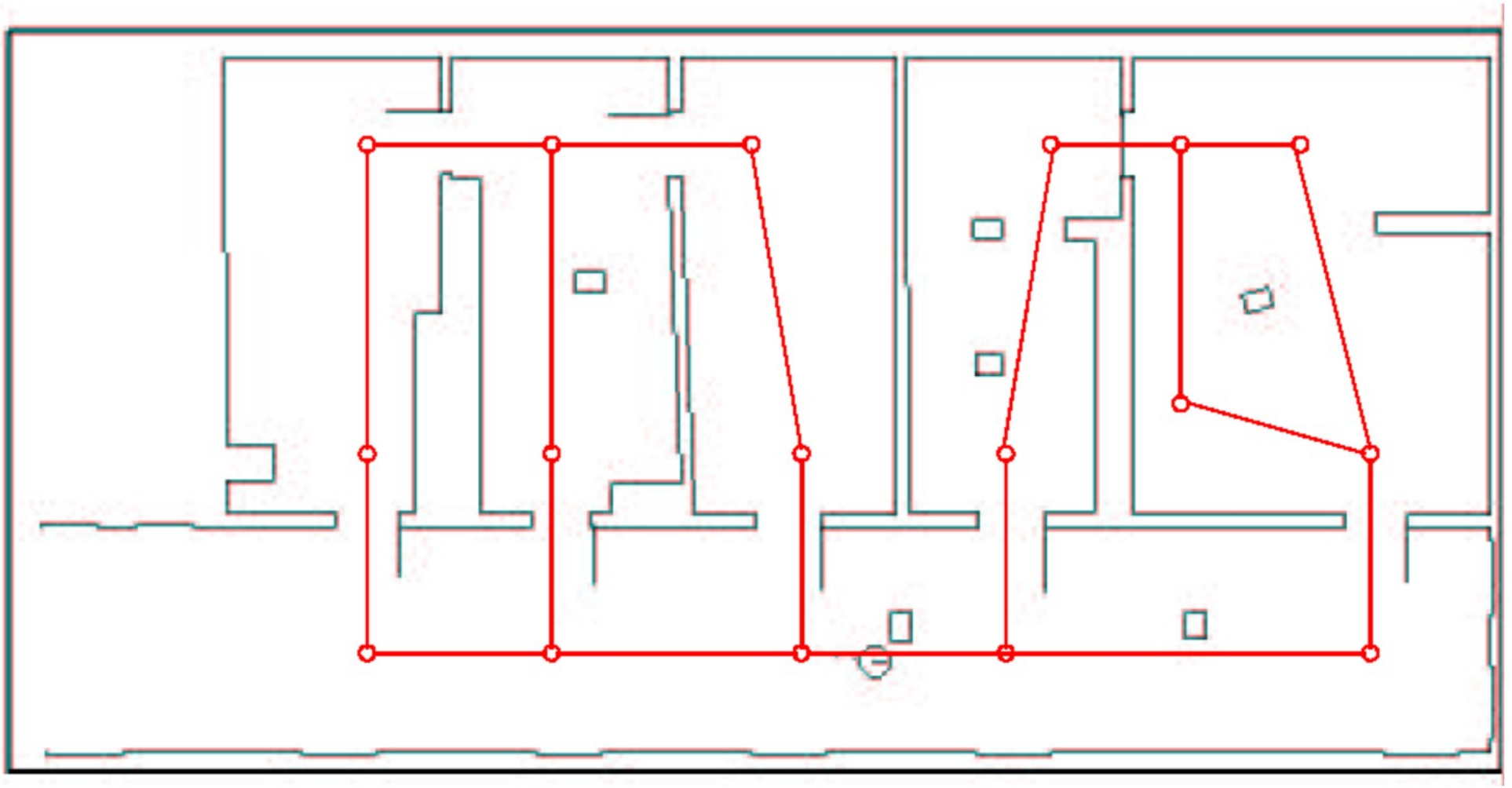
- $P_{\text{long}}(d_i^{\alpha})$ probability that virtual sensor measures distance larger than d_i in direction α

$$P_{\text{long}}(d_i^{\alpha}) = \sum_{j>i} P(d_j^{\alpha})$$

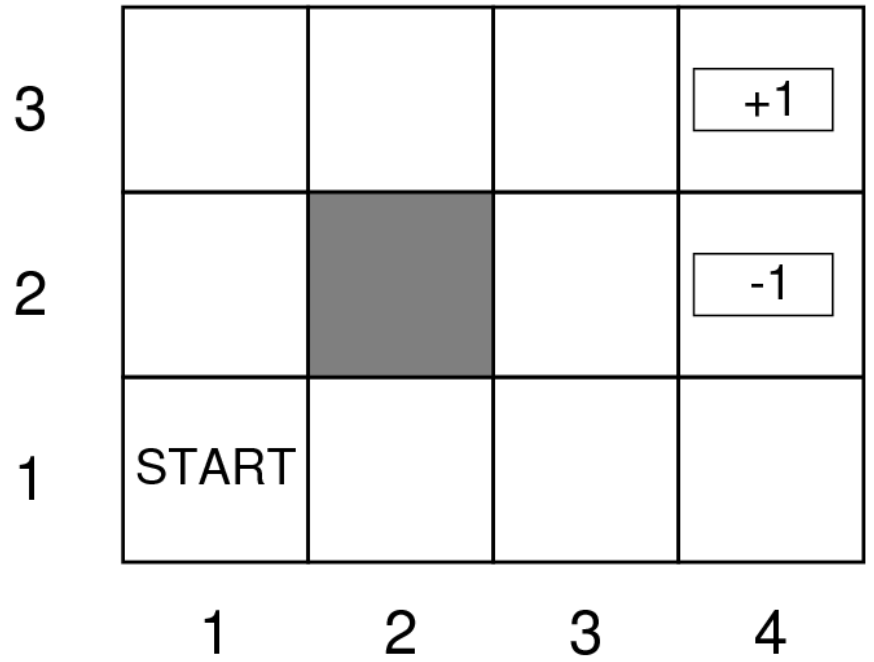
- Virtual measurement $(d^{\alpha})^*$:= largest distance such that with probability 99% the nearest invisible obstacle is a distance larger than d^*:

$$(d^{\alpha})^* = \max_i \{ d_i^{\alpha} | P_{\text{long}}(d_i^{\alpha}) \geq 0.99 \}$$

Path Planning



Path Planning as Markov Decision Problem (MDP)



- Going in one of the four direction succeeds with probability 0.8
- With probability 0.1 + 0.1 robot goes +90∞ or –90∞ instead
- Robot ends up at same location if move would hit a wall
- Every move “rewarded” by –0.04

Optimal Policy

- $P(l'|a,l)$ probability of reaching location l' by action a in location l
- $R(l)$ reward of location l
- $U(l)$ utility of location l

$$U(l) = R(l) + \max_a \sum_{l'} P(l'|a,l) \cdot U(l')$$

- $policy^*(l)$ optimal policy (mapping from locations to actions)

$$policy^*(l) = \arg \max_a \sum_{l'} P(l'|a,l) \cdot U(l')$$

Value Iteration

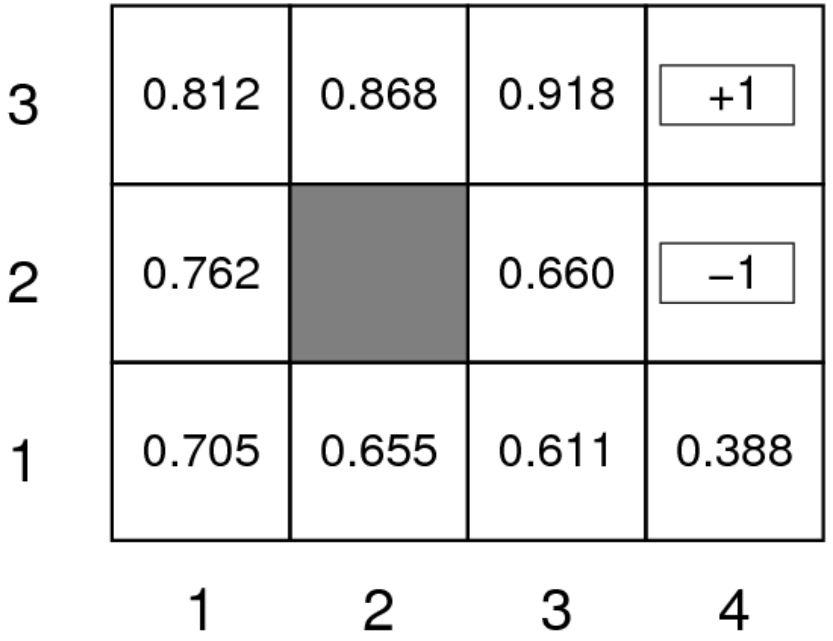
- Initialize utility with $U_0(l) := R(l)$
- Iteration using the following equation converges to stable values (under certain conditions)

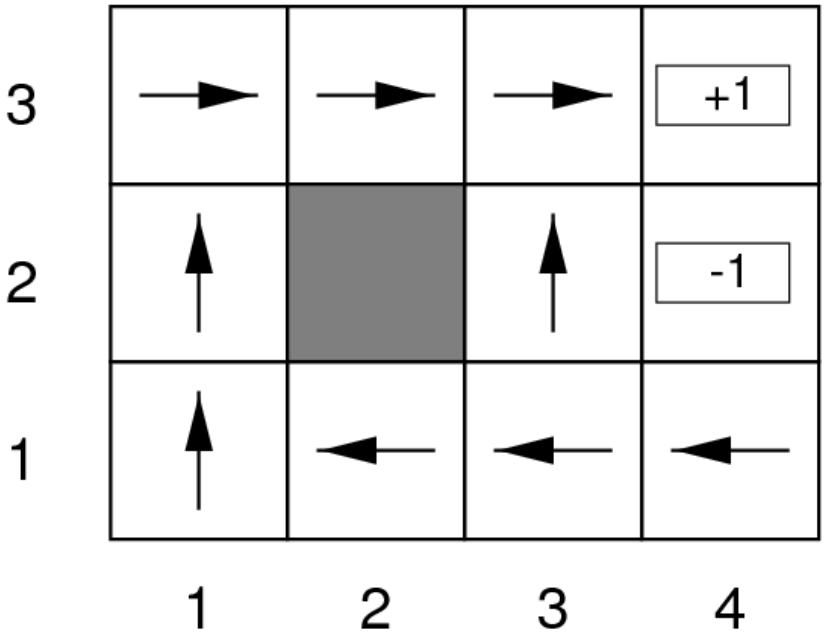
$$U_{i+1}(l) = R(l) + \max_a \sum_{l'} P(l' | a, l) \cdot U_i(l')$$

3	0.04	0.04	0.04	+1
2	0.04		0.04	−1
1	0.04	0.04	0.04	0.04
	1	2	3	4

3	0.812	0.868	0.918	+1
2	0.762		0.660	−1
1	0.705	0.655	0.611	0.388
	1	2	3	4

Optimal Policy





Partially Observable Markov Decision Problem (POMDP)

- POMDPs are MDPs with probability distributions over locations
- $U(b)$ utility of belief state b, i.e., $Bel(L = l)$ for possible locations l
- $R(b)$ reward of belief state b

$$R(b) = \int R(l) \cdot Bel(L = l)\, dl$$

- Value iteration

$$U_{i+1}(b) = \max_a \int (R(b') + U_i(b')) \cdot P(b' | a, b)\, db'$$

- Optimal policy

$$policy^*(b) = \arg \max_a \int U(b') \cdot P(b' | a, b)\, db'$$