Mobile Robot Systems

Lecture 5: Localization

Dr. Amanda Prorok

asp45@cam.ac.uk

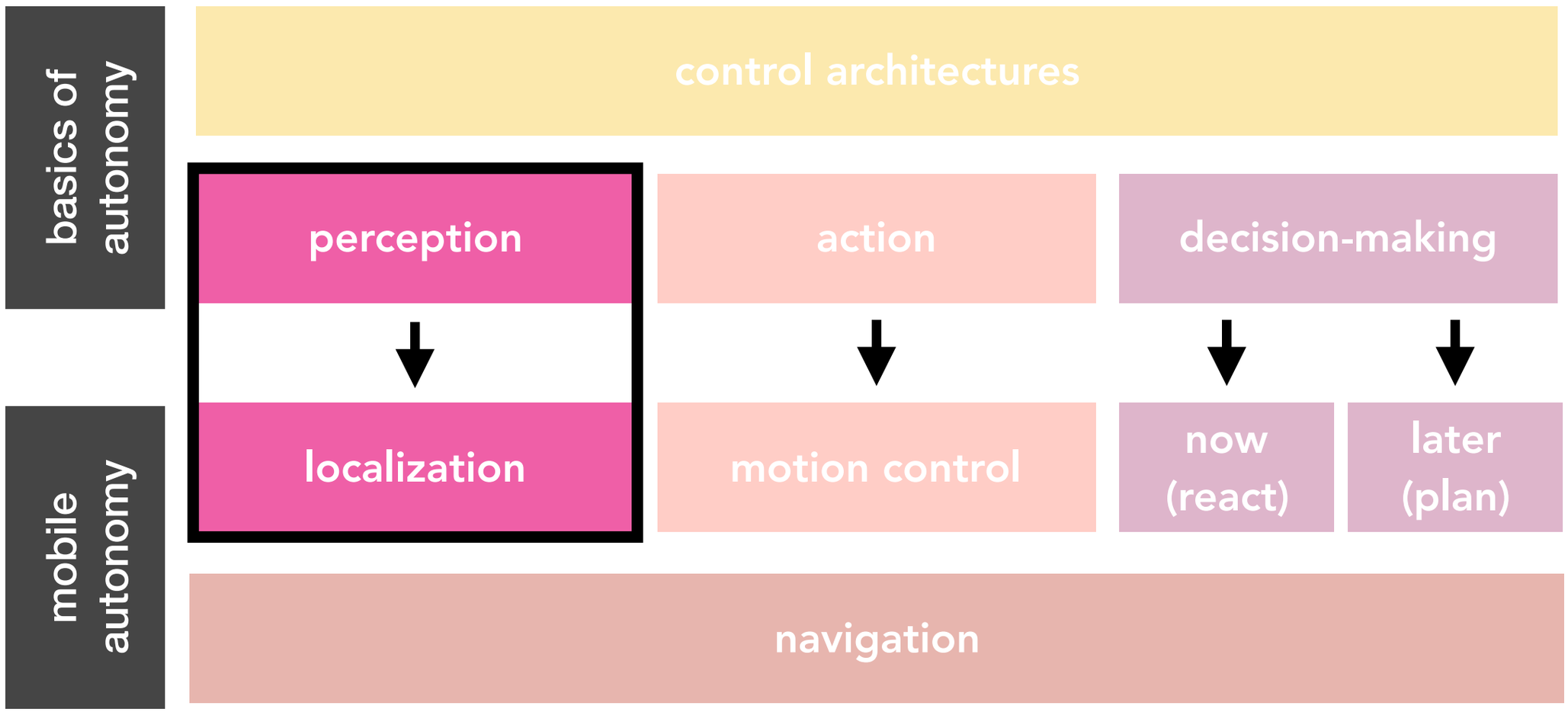
www.proroklab.org

In this Lecture

- Probabilistic localization
- Bayes (rule and algorithm)
- Grid localization
- Filters
  - Particle filter
  - Kalman Filter
- Map representations

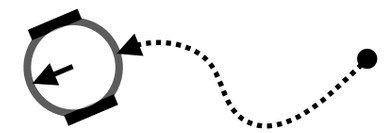
- Credits:
  - Examples from Probabilistic Robotics; Thrun et al.

Architecture of an Autonomous Mobile Robot

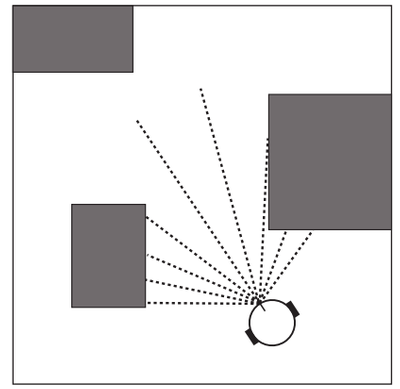


Taxonomy of Localization Problems

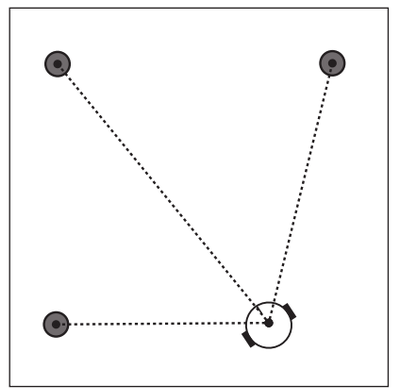
- Dead-reckoning (position tracking): initial position known
  - Blindly update pose based on differential movements
- Global localization: initial position unknown
  - Map-based (with landmarks)
    - Sensors: laser, camera, proximity
      - Method: map-matching techniques (various)
  - Beacon-based (with active infrastructure)
    - Bluetooth, WiFi, GPS (outdoors only), etc.
    - Method: trilateration, fingerprinting, proximity
- Global localization and position tracking combined.



odometry-based



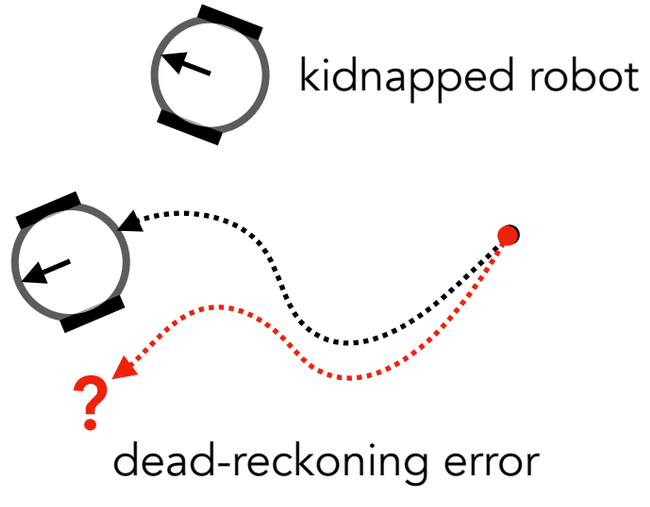
map-based

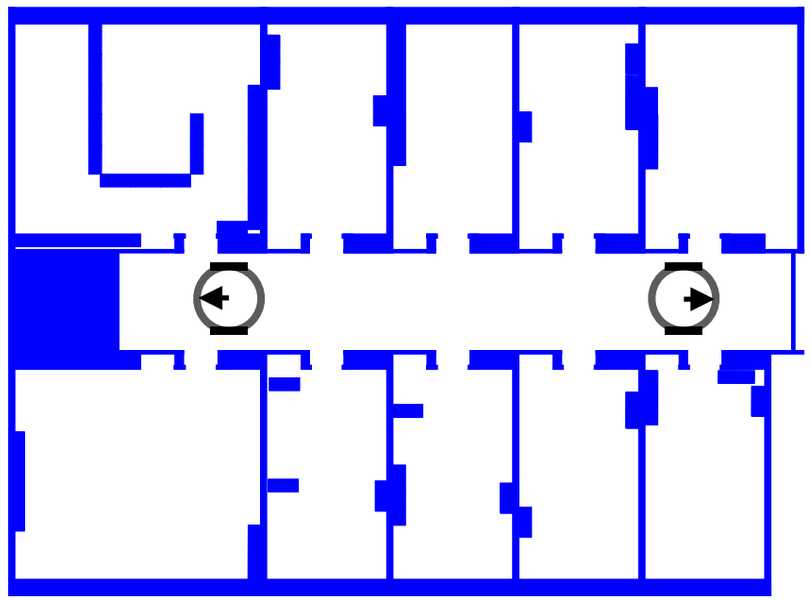


beacon-based

Challenges

- Dead-reckoning
  - wheel slip, slack in actuation mechanism
  - runaway errors
- Global localization
  - random errors and failures
  - non-Gaussian sensor noise
  - unavailability of sensor (GPS-denial)
  - map ambiguity
  - dynamic environments
  - kidnapped robot problem

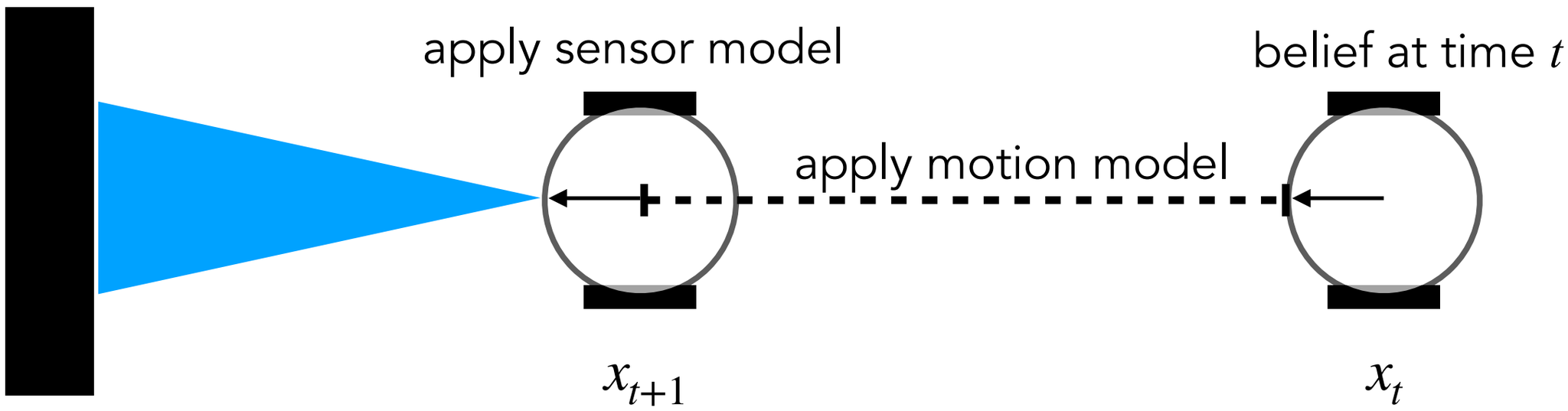




ambiguity (multiple local maxima)

Probabilistic Localization

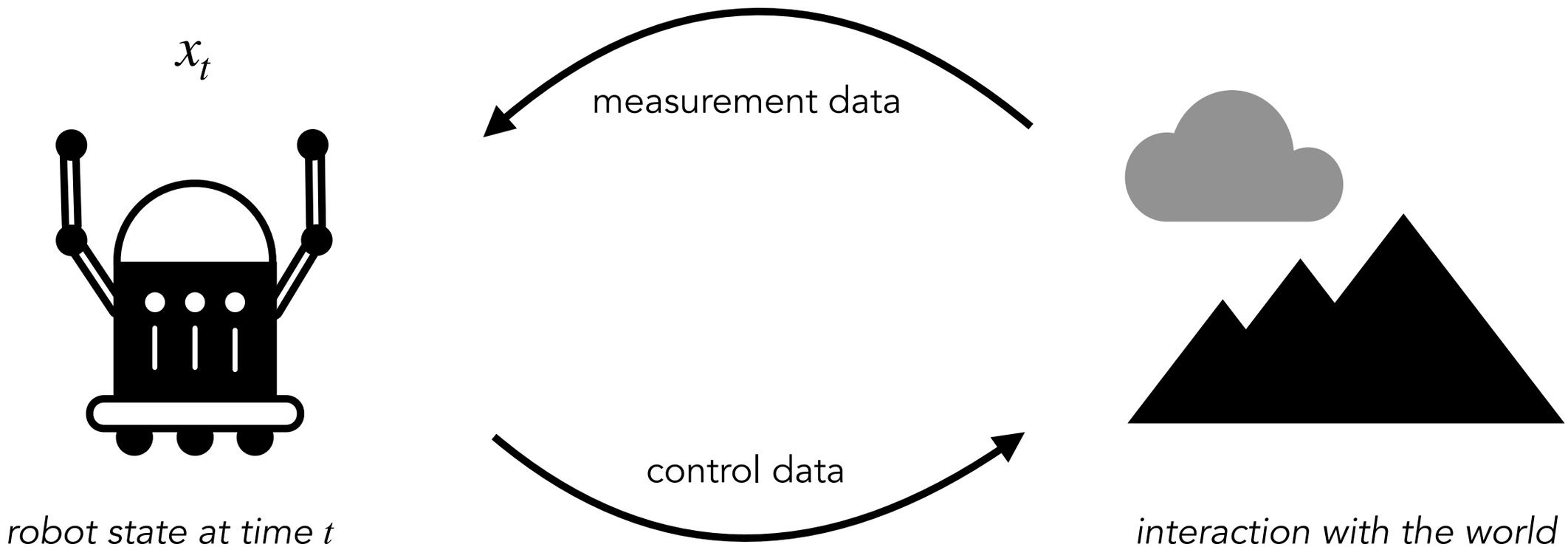
- In robotics, we deal with localization probabilistically.
- Three key components:
  1. a robot's belief of where it is (its state)
  2. a robot's motion model
  3. a robot's sensor (observation) model



Sensor and Control Data

- An autonomous robot interacting with the world

$$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \ldots, z_{t_2}$$



$$u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2}$$

we will treat odometry readings as our control data!

Bayes Rule in Robotics

- Let's assume x is the robot state, and z is measurement data.

  prior probability distribution: $p(x)$

  posterior probability distribution: $p(x \mid z)$

- Estimate robot state using a 'generative model' $p(z \mid x)$, which describes how a state variable ***causes*** sensor measurements z.

Bayes rule (discrete version):

$$p(x \mid z) = \frac{p(z \mid x)\ p(x)}{p(z)} = \frac{p(z \mid x)\ p(x)}{\sum_{x'} p(z \mid x')\ p(x')}$$

theorem of total probability

$$p(x \mid z) = \eta\ p(z \mid x)\ p(x)$$

normalizes density

denominator does not depend on x

- We can now update a robot's state estimate based on sensor measurements and a prior belief.

Probabilistic Generative Laws

- Robot state x_t is *'complete'* meaning that no extra knowledge of past events can help us better predict the future.
- Robot state x_t is *'generated'* from a probability distribution:

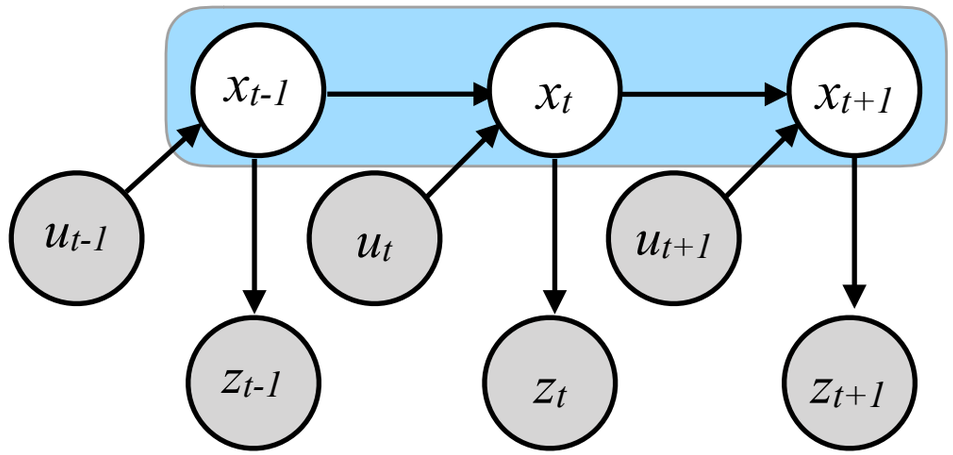
$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) \quad \xrightarrow{\text{the state is 'complete'}} \quad p(x_t \mid x_{t-1}, u_t)$$

state transition probability

$$p(z_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) \quad \xrightarrow{\text{the state is 'complete'}} \quad p(z_t \mid x_t)$$

measurement probability

Dynamic Bayes graph:



The Belief

- A robot's true state cannot be measured directly, it has to be inferred.
- Probabilistic robotics represents a ***belief*** (or state of knowledge) through conditional probability distributions.
- A belief distribution is a posterior over state variables conditioned on available data (i.e., *after* incorporating measurement data):

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$$

- Note: the posterior (belief) *before* incorporating measurement data is denoted as:

$$\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t})$$

Bayes Filter

For all (x_t) do:

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1}) dx_{t-1}$$

apply motion model to previous belief

$$bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$$

apply measurement update to current belief

return: $bel(x_t)$

- (Recursively) calculates the posterior over the state x_t, conditioned on measurement and control data.
- Requires definition of 3 probability distributions:

  initial belief: $p(x_0)$

  measurement probability: $p(z_t \mid x_t)$

  state transition probability: $p(x_t \mid u_t, x_{t-1})$

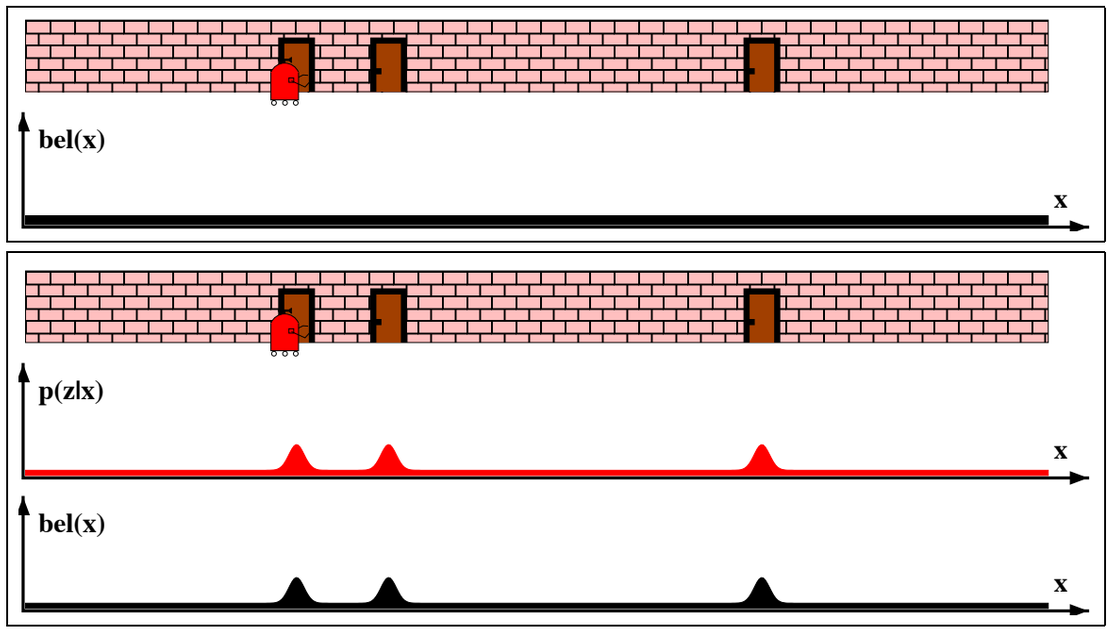
The Markov Assumption

- Plays a fundamental role in probabilistic robotics (and is equivalent to our assumption of a 'complete' state):

  "Past and future data are independent if one knows the current state x_t"

- Violations happen when:
  - there are unmodeled dynamics (e.g., dynamic obstacles)
  - probabilistic models are inaccurate
  - we have approximation errors
- Still: it is an indispensable assumption, since computations would otherwise become intractable.

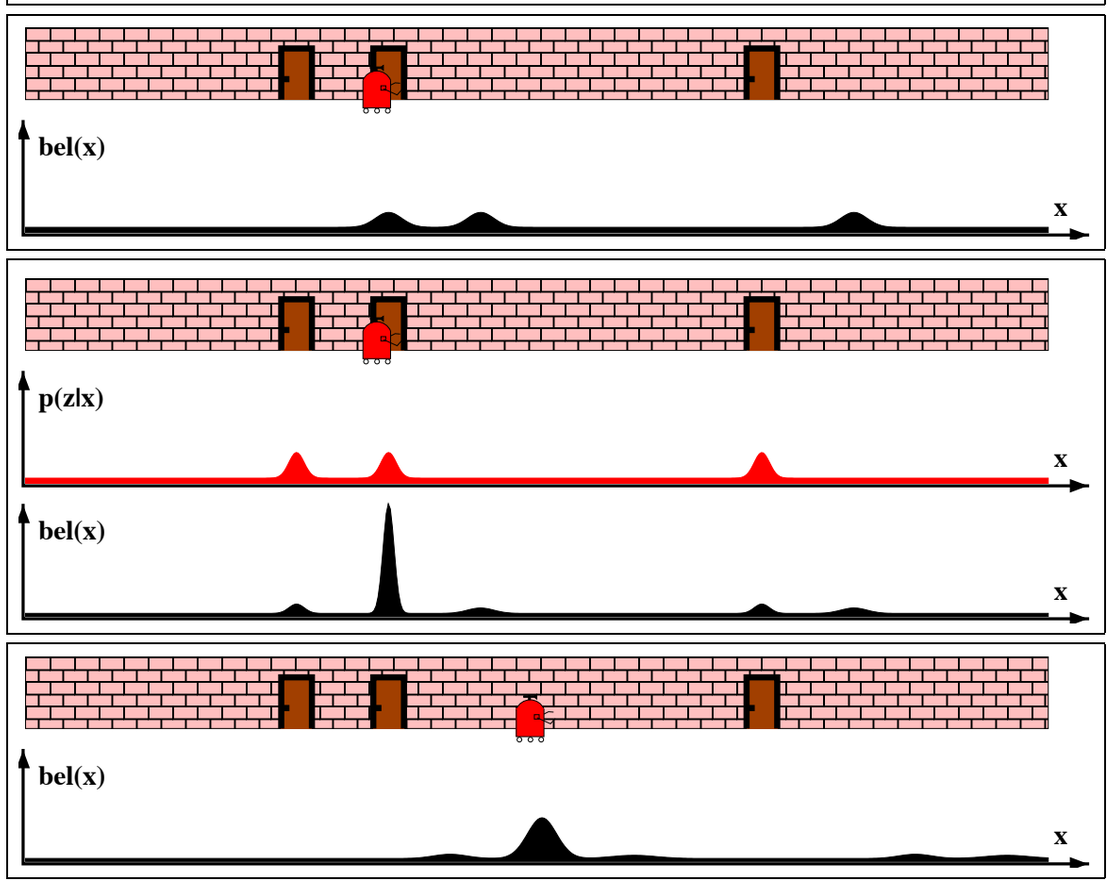
Markov Localization



posterior, before measurement update

measurement probability distribution given *x*

posterior, after measurement update



apply motion model to obtain new posterior

posterior, after measurement update

apply motion model to obtain new posterior

* image credit: Probabilistic Robotics; Thrun et al.

Discrete Bayes Filter

- Otherwise known as a **histogram filter**:

For all k do:

$$\bar{p}_{k,t} = \sum_i p(X_t = x_k \,|\, u_t, X_{t-1} = x_i)\; p_{i,t-1}$$

$$p_{k,t} = \eta\; p(z_t \,|\, X_t = x_k)\; \bar{p}_{k,t}$$

return: $\{p_{k,t}\}$

- Recall the Bayes filter:

For all (x_t) do:

$$\overline{bel}(x_t) = \int p(x_t \,|\, u_t, x_{t-1})\; bel(x_{t-1}) dx_{t-1}$$

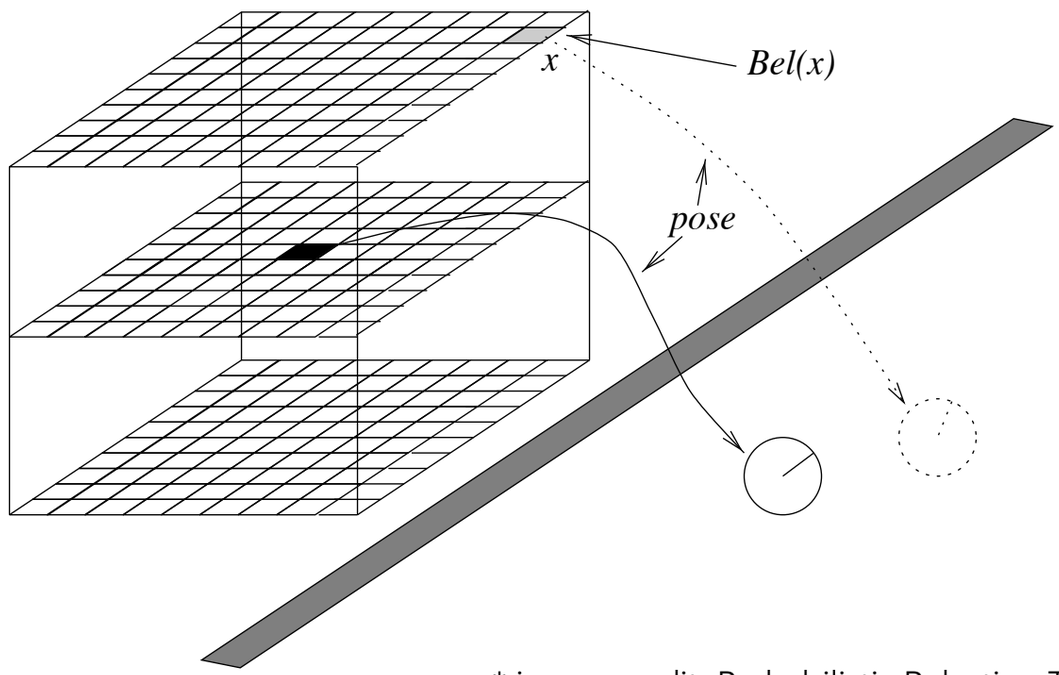
$$bel(x_t) = \eta\; p(z_t \,|\, x_t)\; \overline{bel}(x_t)$$

return: $bel(x_t)$

Grid Localization

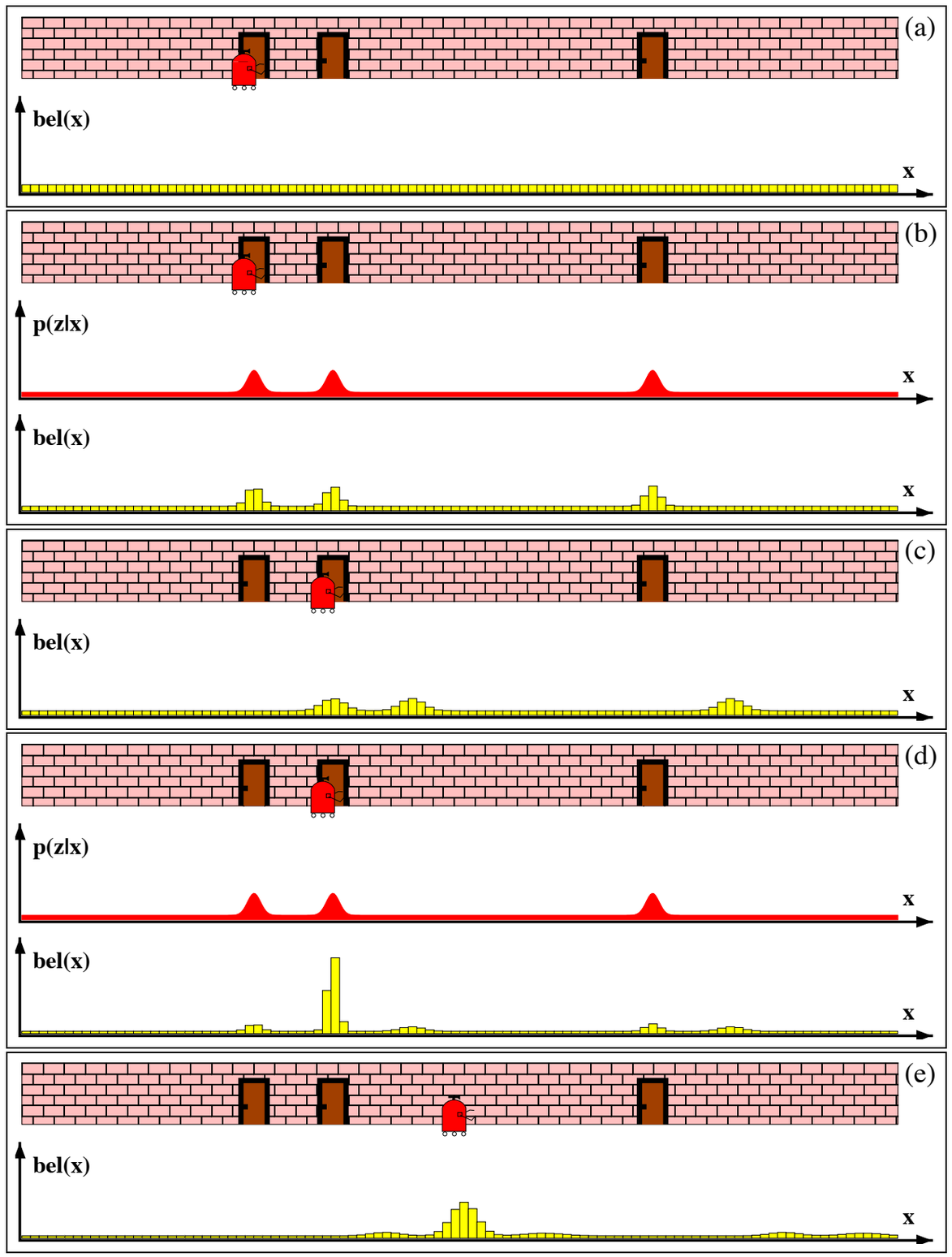
- Application of the Bayes filter using a discrete representation (finite state space).
  - Approximate the posterior using a histogram filter
  - Use grid decomposition of pose space
  - The belief is a collection of discrete probability values (that is normalized over the collection) :

$$bel(x_t) = \{p_{k,t}\}$$



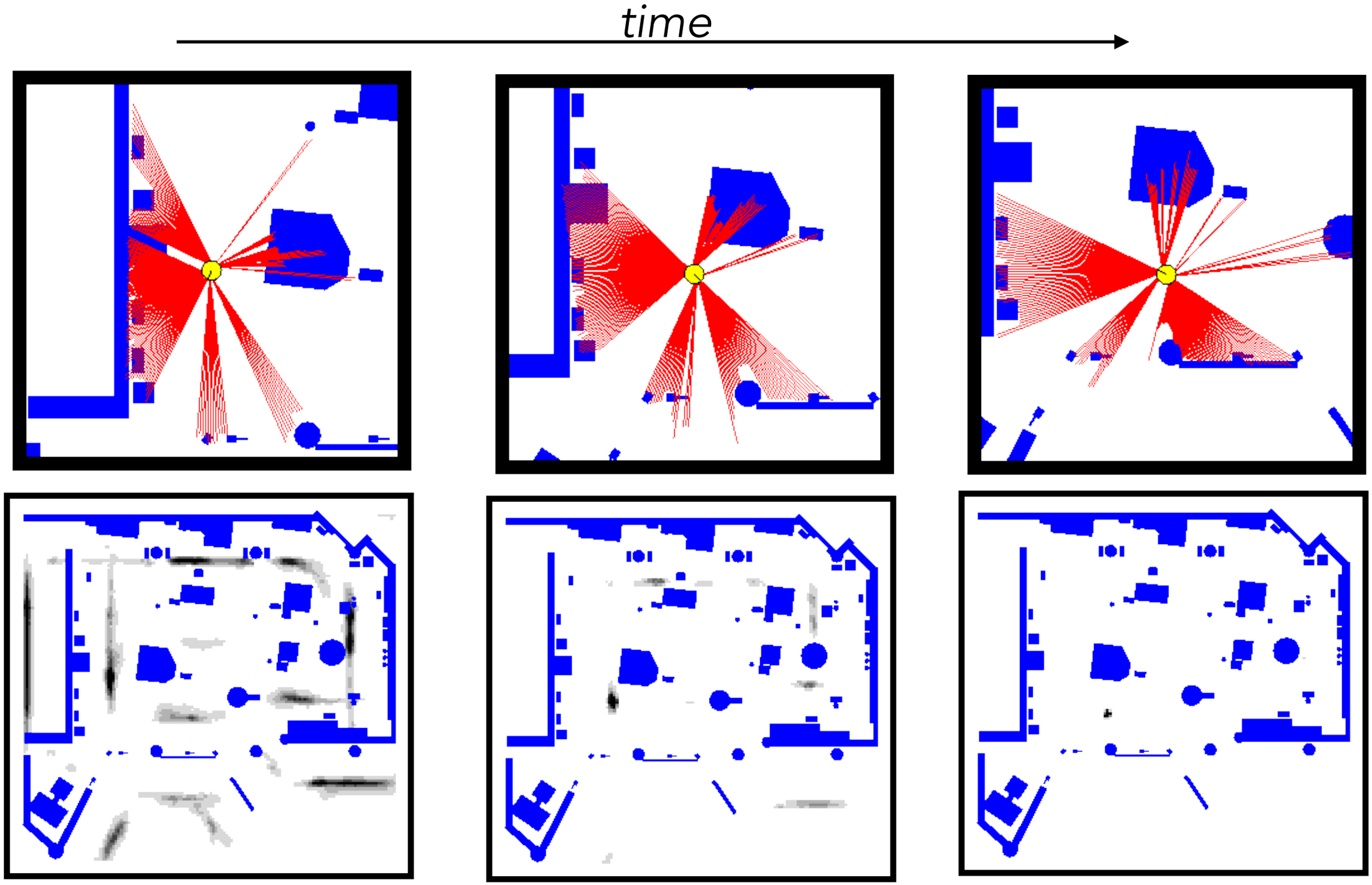
* image credit: Probabilistic Robotics; Thrun et al.

Grid Localization



* image credit: Probabilistic Robotics; Thrun et al.

Grid Localization



* image credit: Burgard, course notes.

Particle Filter

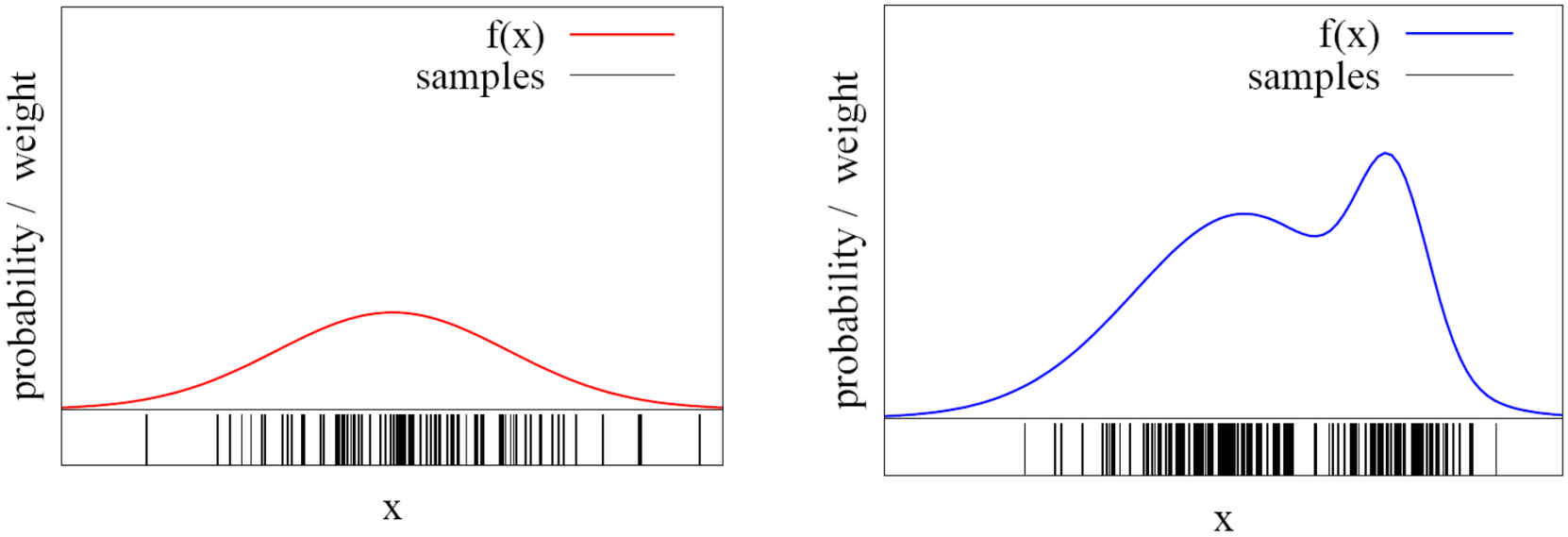
- Nonparametric implementation of the Bayes filter
- Similar to histogram filter, approximate posterior with finite number of parameters.
- Key idea: distribution is represented by a **set of samples** drawn from this distribution
  - Belief $bel(x_t)$ is represented by a set of random state samples drawn from this posterior. A 'particle' represents a state hypothesis.
  - Filter: *'survival of the fittest particles'!*
  - Very efficient for representing non-Gaussian distributions.
- Particles are denoted:

$$\mathcal{X}_t := \{\langle x^{[1]}, w^{[1]}\rangle, \langle x^{[2]}, w^{[2]}\rangle, \ldots, \langle x^{[M]}, w^{[M]}\rangle\}$$

state hypothesis — importance factor (weight)

Particle Filter

- The samples represent the posterior: $$p(x) = \sum_{i=1}^{M} w^{[i]} \, \delta_{x^{[i]}}(x)$$

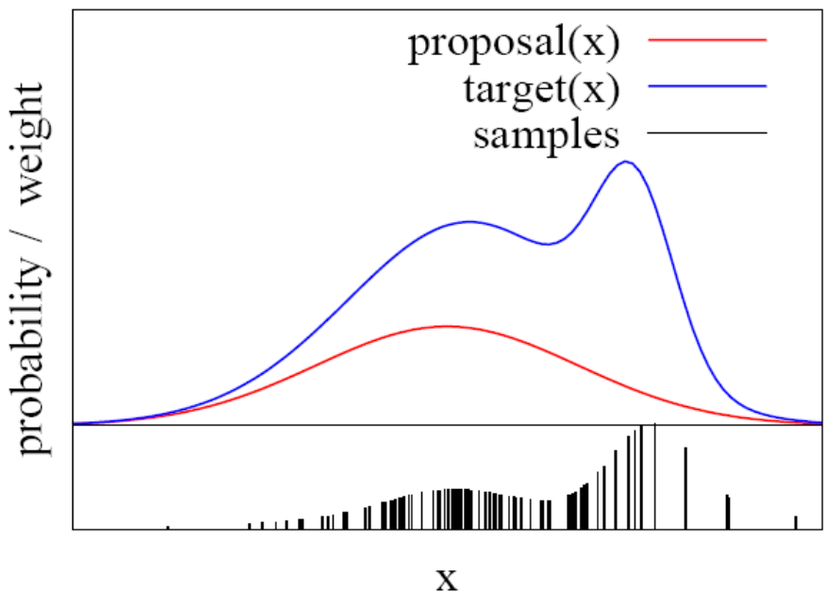


- The more particles fall into an interval, the higher its probability density.
- The more particles we use, the better the estimate!

Particle Filter

- Question: How are we to obtain samples from a new distribution?
- **Importance sampling** allows us to use samples from a **proposal** distribution g to generate new samples from a **target** distribution f
- Account for difference between the distributions by weighting particles according to quotient:

$$w^{[i]} = \frac{f(x^{[i]})}{g(x^{[i]})}$$



Particle Filter

- Three key steps (iterate over):
  1. Sample particles from the proposal distribution
  2. Compute the importance factors
  3. Re-sample the particles: replace unlikely samples with more likely ones.

- Applied to localization:
  1. Proposal distribution is given by motion model:

$$x_t^{[i]} \sim p(x_t \mid x_{t-1}, u_t)$$

  2. Particles are weighted by the measurement model:

$$w_t^{[i]} \sim p(z_t \mid x_t) \quad \propto \frac{target}{proposal}$$

  3. Particles are re-sampled

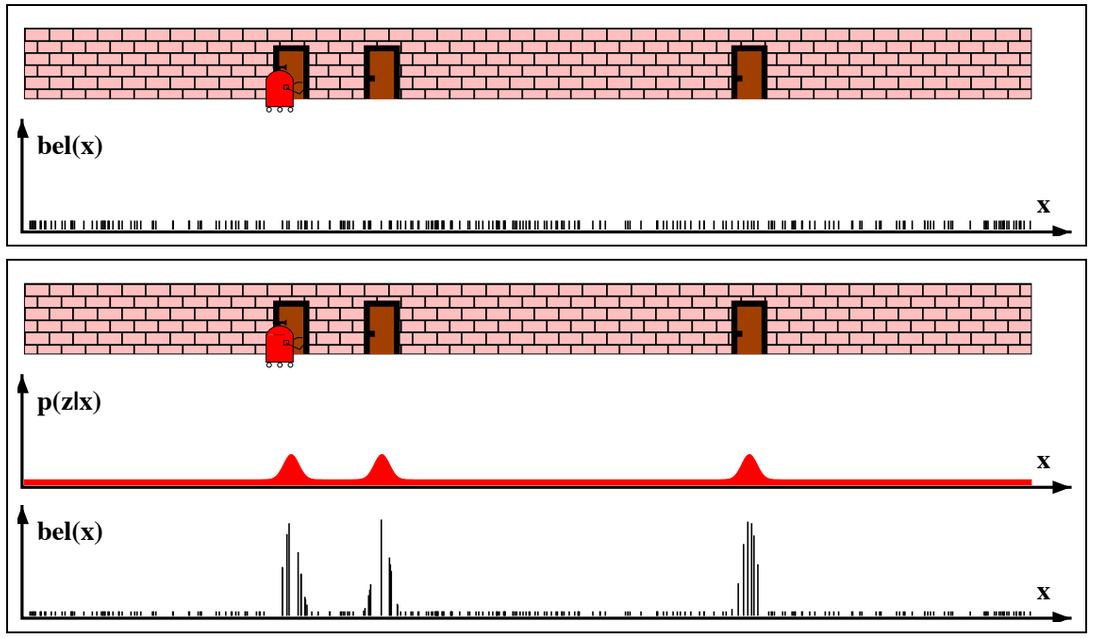
Monte Carlo Localization

- Application of the particle filter to the localization problem.

1: **Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

* see Probabilistic Robotics book for implementation of re-sampling algorithms for line 9.

Monte Carlo Localization



* image credit: Probabilistic Robotics; Thrun et al.

Kalman Filter

- An implementation of the Bayes filter with Gaussians
- Developed by Swerling (1958) and Kalman (1960)
- Technique for predicting and filtering linear Gaussian systems
  - Discrete time
  - Underlying concept: linear transformations conserve Gaussians
  - It is **optimal** (in a least-squares sense)

- Belief in continuous (potentially multi-variate) space:

The first moment μ_t describes the belief, and the second moment Σ_t describes its uncertainty

State transition model $$x_t = A_t x_{t-1} + B u_t + \epsilon_t$$

Measurement model $$z_t = C_t x_t + \delta_t$$

Kalman Filter

- Components:

For state dimensionality *n*, measurement dimensionality *k*, control dimensionality *l*

A_t (n x n) matrix that describes how the state evolves from *t-1* to *t* without controls or noise.

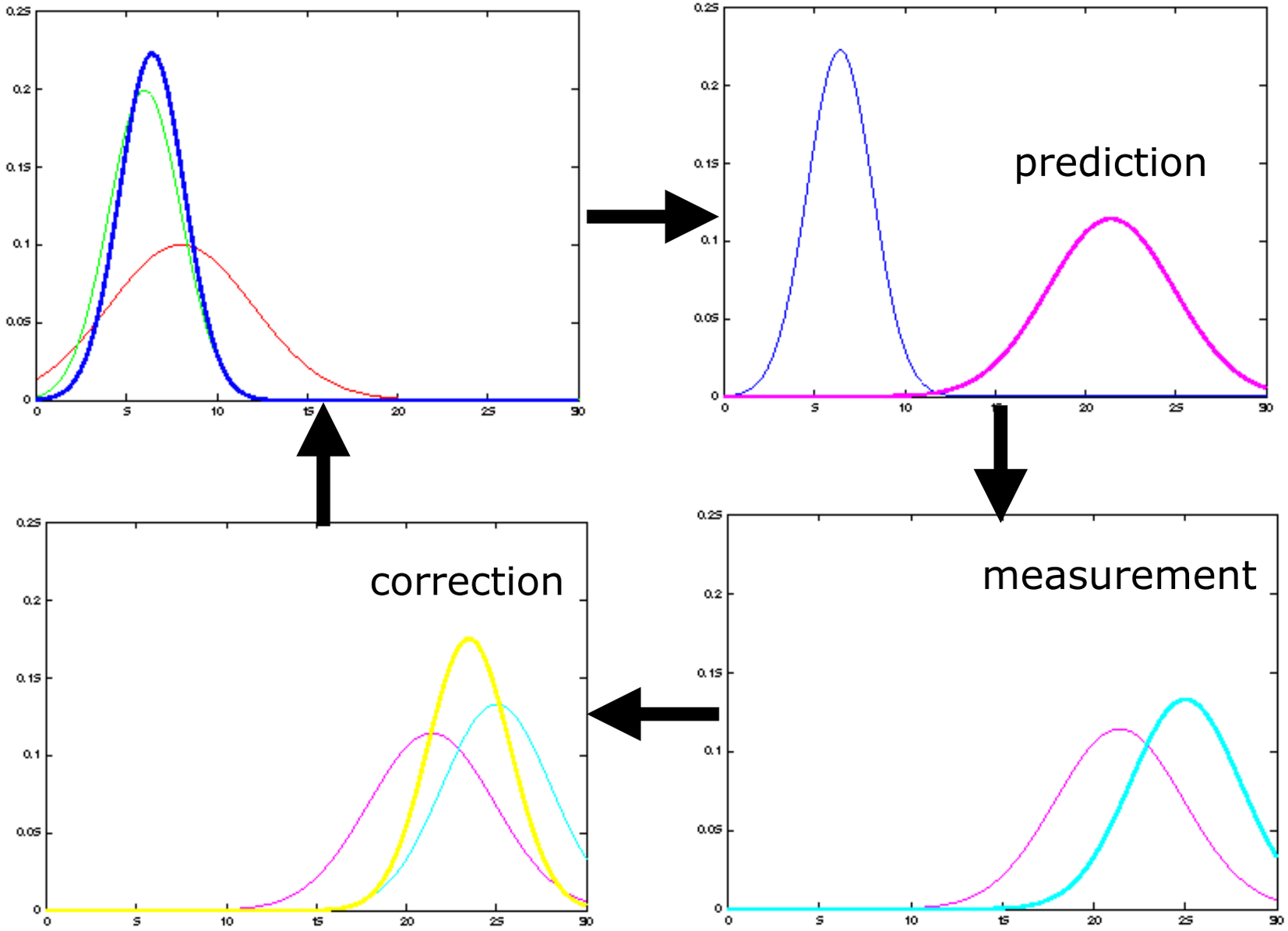
B_t (n x l) matrix that describes how the control u_t changes the state from *t -1* to *t*.

C_t (k x n) matrix that describes how to map the state x_t to an observation z_t

ϵ_t Random variables representing the process noise and measurement noise. These are assumed to be independent and δ_t normally distributed with covariance R_t and Q_t, respectively.

Kalman Filter

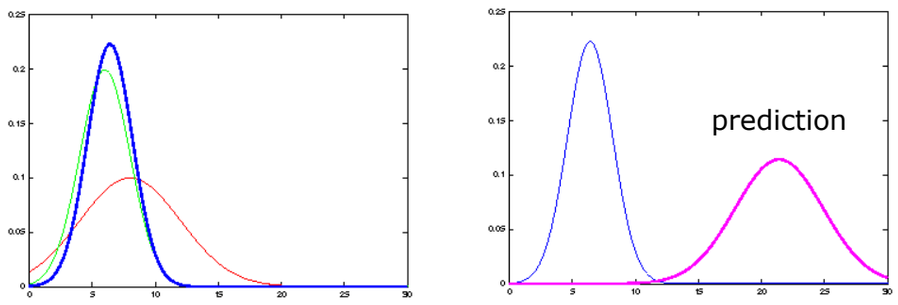
- The prediction-correction cycle



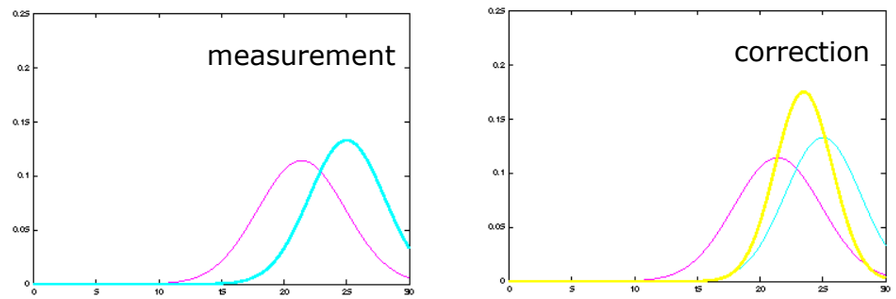
Kalman Filter

- The prediction-correction cycle

prediction step



correction step



$$\overline{bel}(x_t) \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R_t \end{cases}$$

- Compute linear state transition
- Update the covariance (it increases); the process noise follows a Normal distribution

$$bel(x_t) \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases}$$

- Compute the residual (differencing the predicted measurement and obtained measurement)
- Update covariance: matrix *K* specifies Kalman gain, tells us how much we believe the prediction vs how much we believe in the measurement.

Kalman Filter

Input: $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$

$$\bar{\mu}_t = A_t \mu_{t-1} + B u_t$$

$$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R_t$$

prediction

$$K_t = \overline{\Sigma}_t C_t (C_t \overline{\Sigma}_t C_t^\top + Q_t)^{-1}$$

Kalman gain

$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

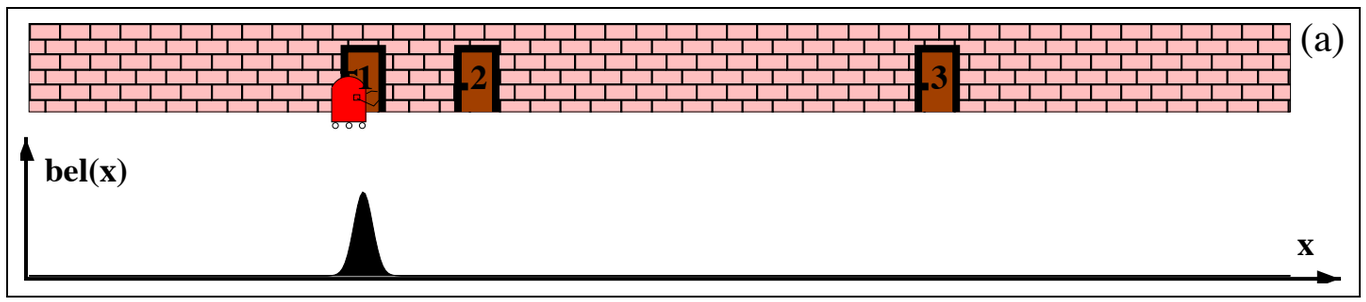
$$\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$$

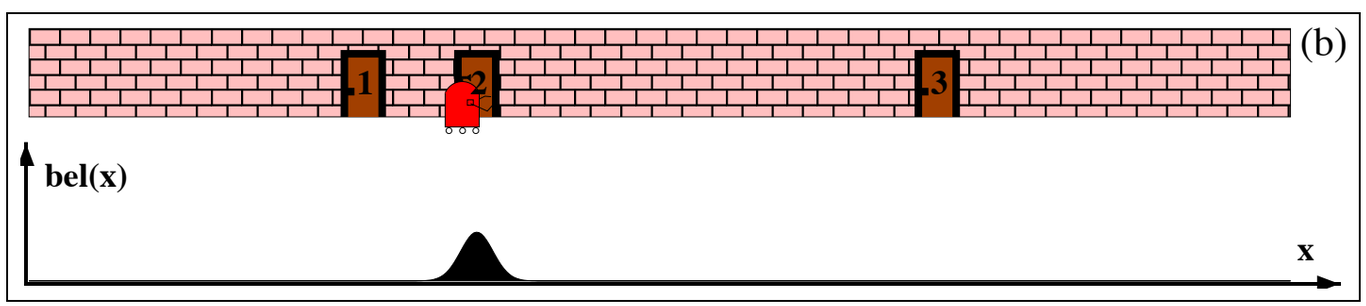
correction

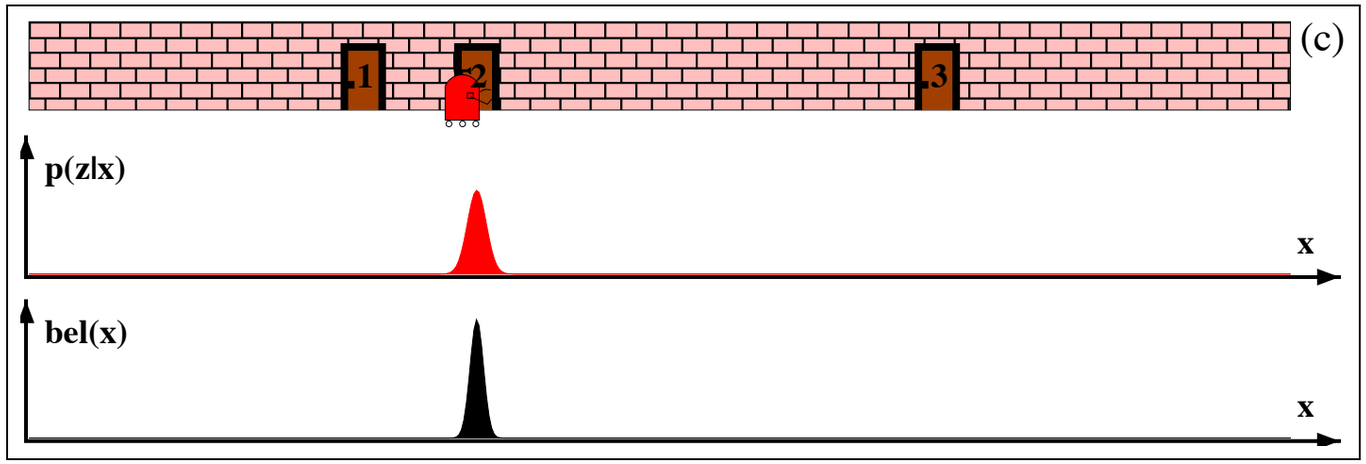
Output: μ_t, Σ_t

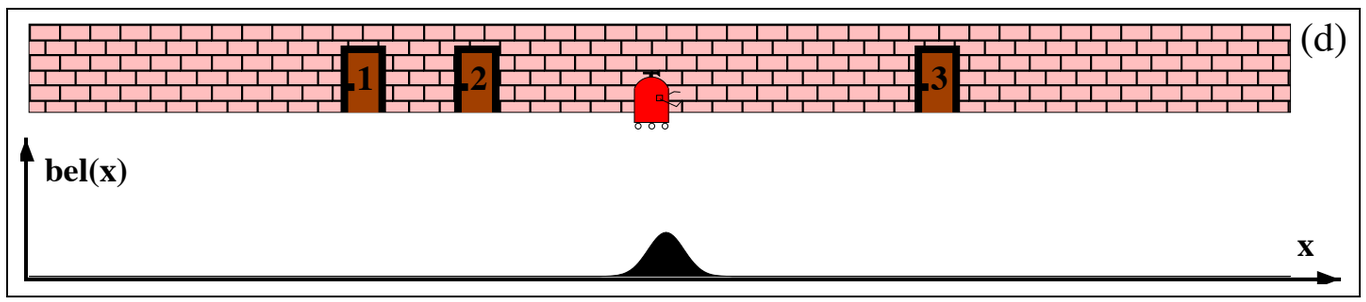
- Efficient: polynomial in the measurement dimensionality *k* and state dimensionality *n*
- Optimal for linear Gaussian systems
- Only models unimodal beliefs

Kalman Filter





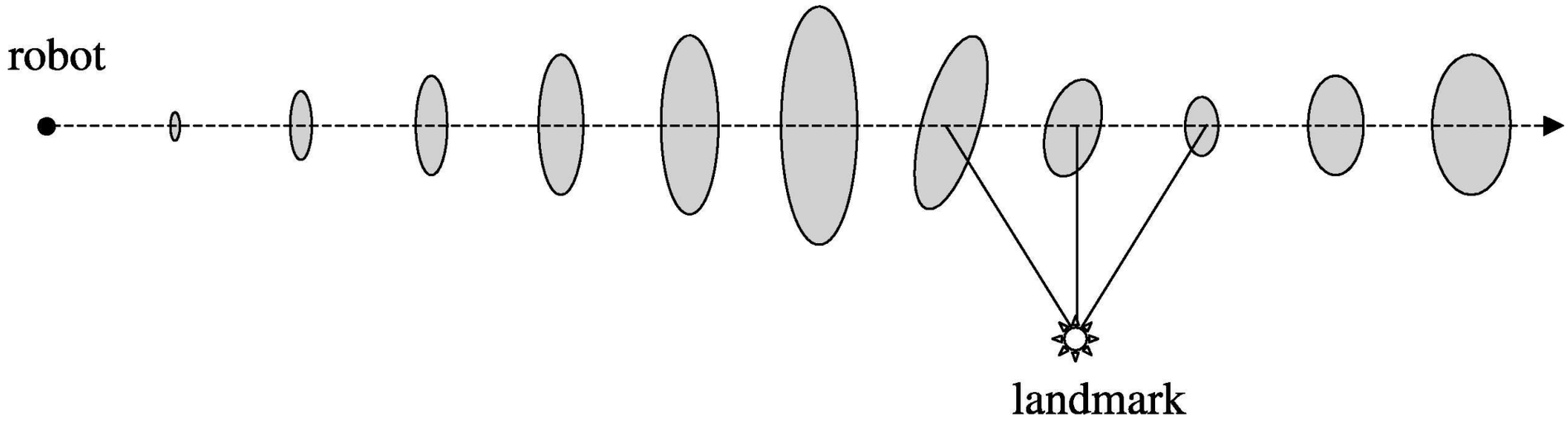




* image credit: Probabilistic Robotics; Thrun et al.

Kalman Filter

- Example: evolution of covariance as robot navigates; localization based on odometry and exteroceptive sensors capable of detecting a landmark.



Comparison

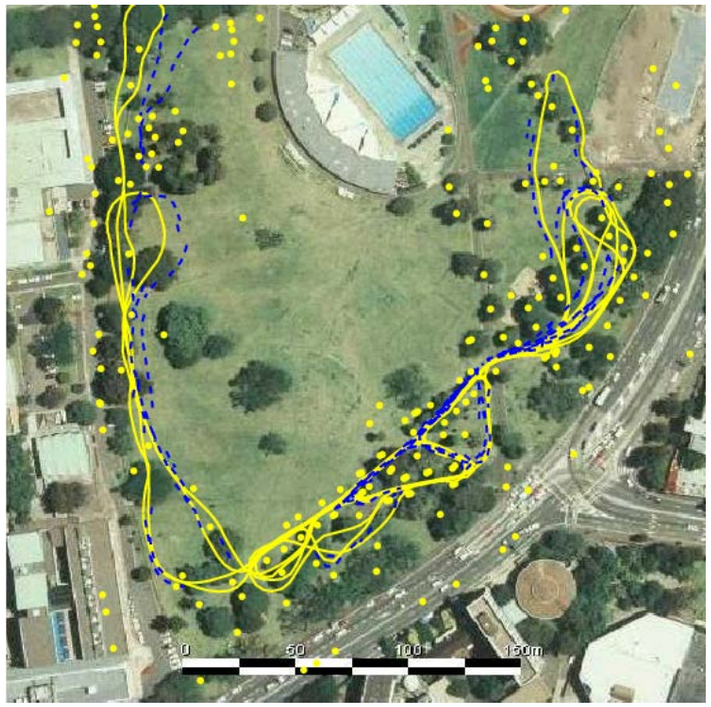
	Grid Localization	**Monte Carlo Localization**	**(Extended) Kalman Filter**
Measurements	raw measurements	raw measurements	landmarks
Measurement noise	any	any	Gaussian
Posterior	any	any	Gaussian
Efficiency (memory)	- -	- -	+ +
Efficiency (time)	- -	- -	+ +
Robustness	+ +	+ +	-
Resolution	+	+	+ +
Ease of implementation	-	+ +	+
Unknown initial pose	possible	possible	not possible

Map Representation

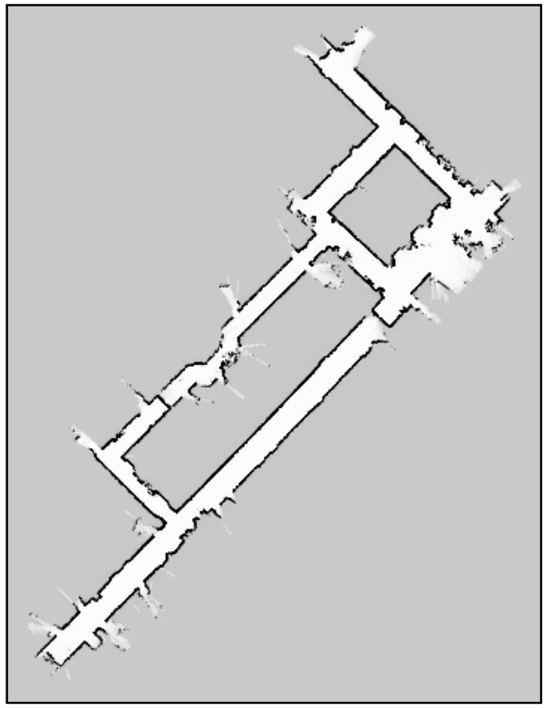
- Measurement models implicitly assume knowledge of a map:

  $p(z_t \mid x_t)$ is actually: $p(z_t \mid x_t, m)$

- Map representations:



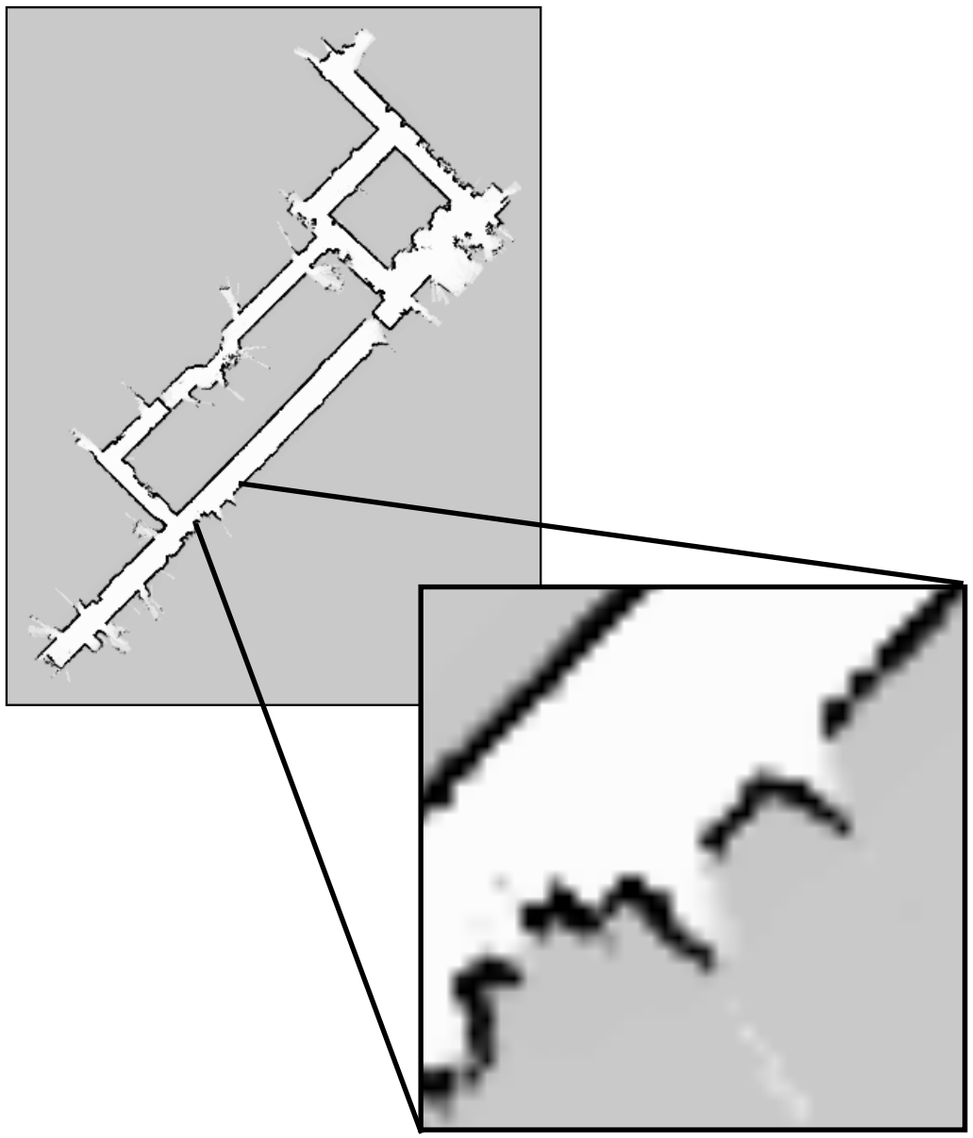
feature map



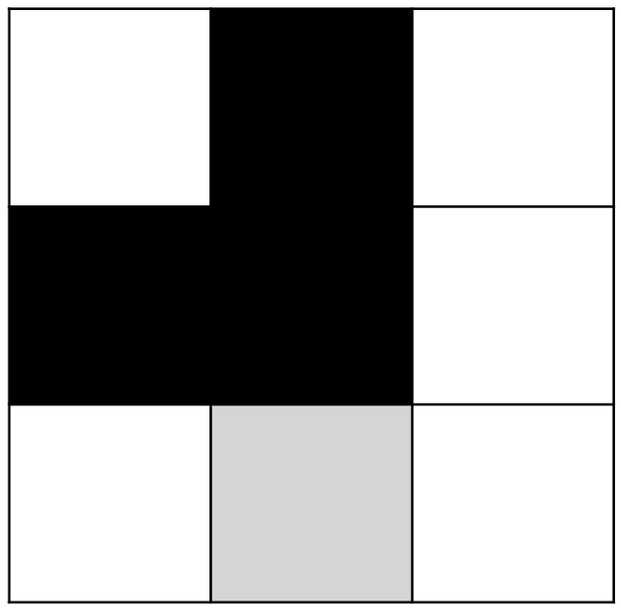
volumetric map (e.g., grid map)

* image credits: E. Nebot and W. Burgard.

Grid Maps

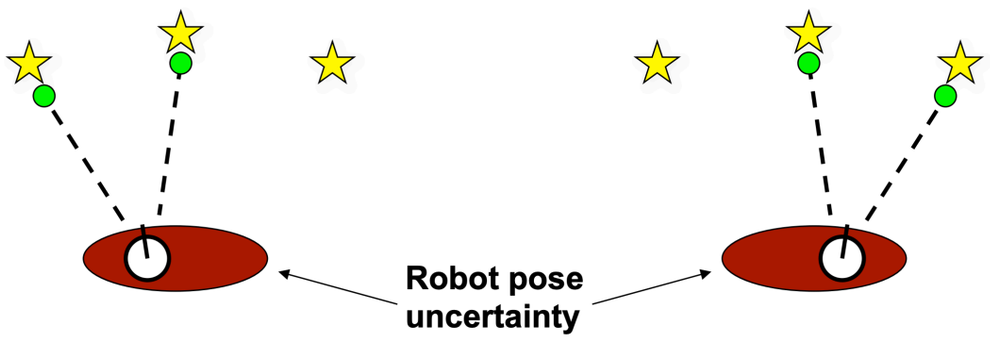


- World discretized into cells
- Each cell is either occupied or free space
- Non-parametric model (space intensive)
- Probabilistic model for mapping purposes



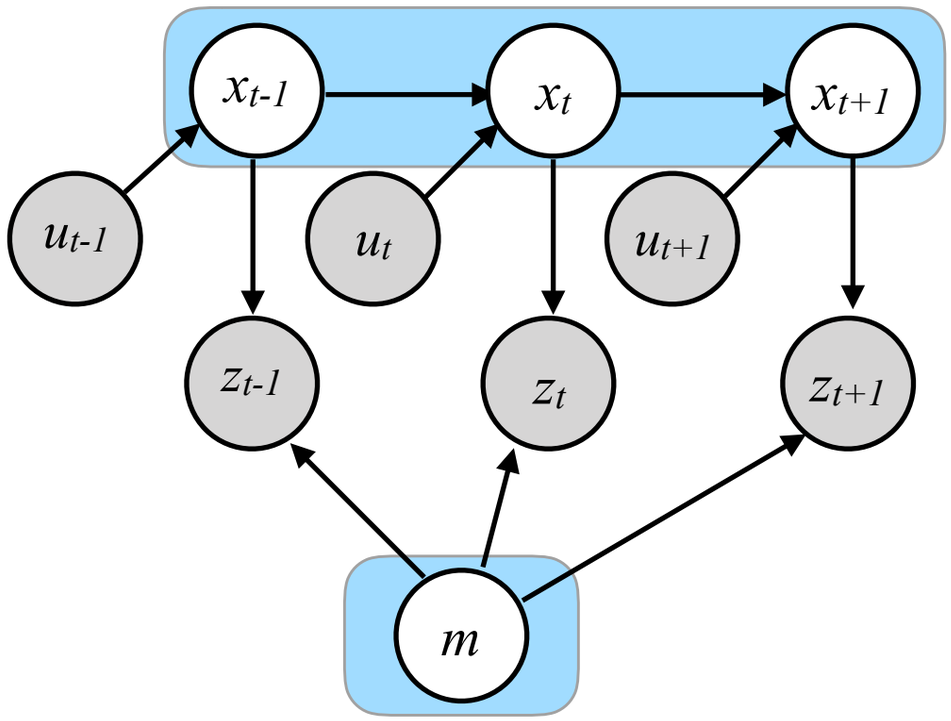
What is SLAM?

- **S**imultaneous **L**ocalization **a**nd **M**apping
- Hard problem - why?
  - Chicken-or-egg problem: a map is needed for localization; a pose is needed for mapping.
  - Errors in robot pose and map are correlated.
  - Data association problem



  - Uncertainties collapse after *loop-closure* (recognition of an already mapped area)

graphical model of full SLAM



Further Reading

Books that cover fundamental concepts:

- Probabilistic Robotics, S. Thrun et al., 2006