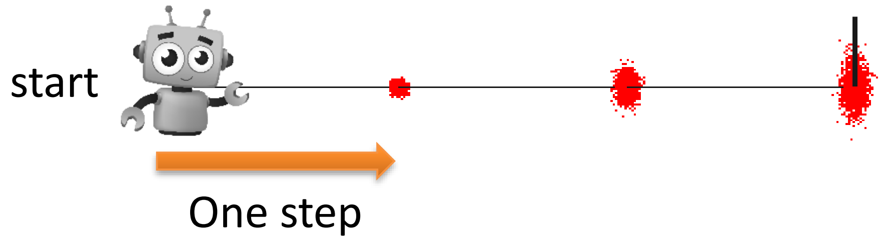
Filtering and State Estimation

Robotics – COMS4045

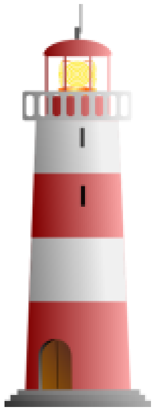
Benjamin Rosman

Some material taken from slides by: Benjamin Kuipers, Greg Welch and Gary Bishop, Wolfram Burgard, Paul Rybski, Pieter Abbeel

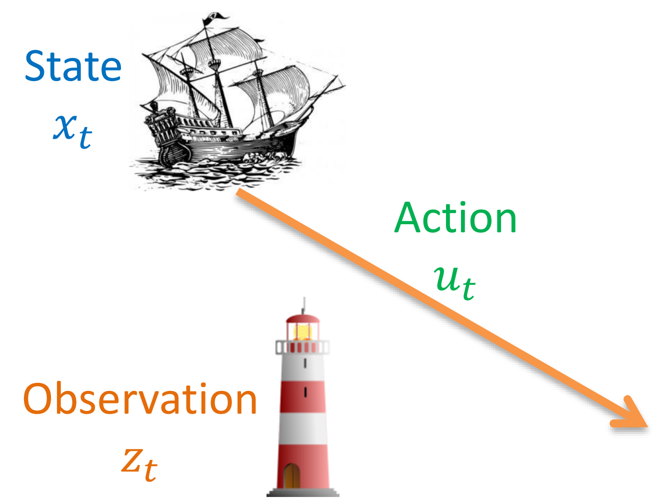
The robot localisation problem



Lost at Sea



Filtering



- Recall a dynamical system:
  - $x_{t+1} = Ax_t + Bu_t + noise$
  - $z_t = Hx_t + noise$

- State is noisy, not fully observable
- But:
  - Control is often a function of state
  - We need to know the state to make decisions!

- Filtering:
  - Estimating the true state of the system, given some (noisy) observations

Applications

- Why do we care about this?

- Tracking: known initial position
- Localisation: unknown initial position

- State-based control
- Smoothing, prediction
- Building towards SLAM

The Kalman filter

- Given a linear dynamical system with Gaussian noise, what is the best estimate of state x?

- **Kalman filter is a recursive approach to provide this estimate**
  - Recursive (update based on only most recent estimate)

- Require:
  - Known action and observation models
  - Noise models (but not noise itself)
  - Observations

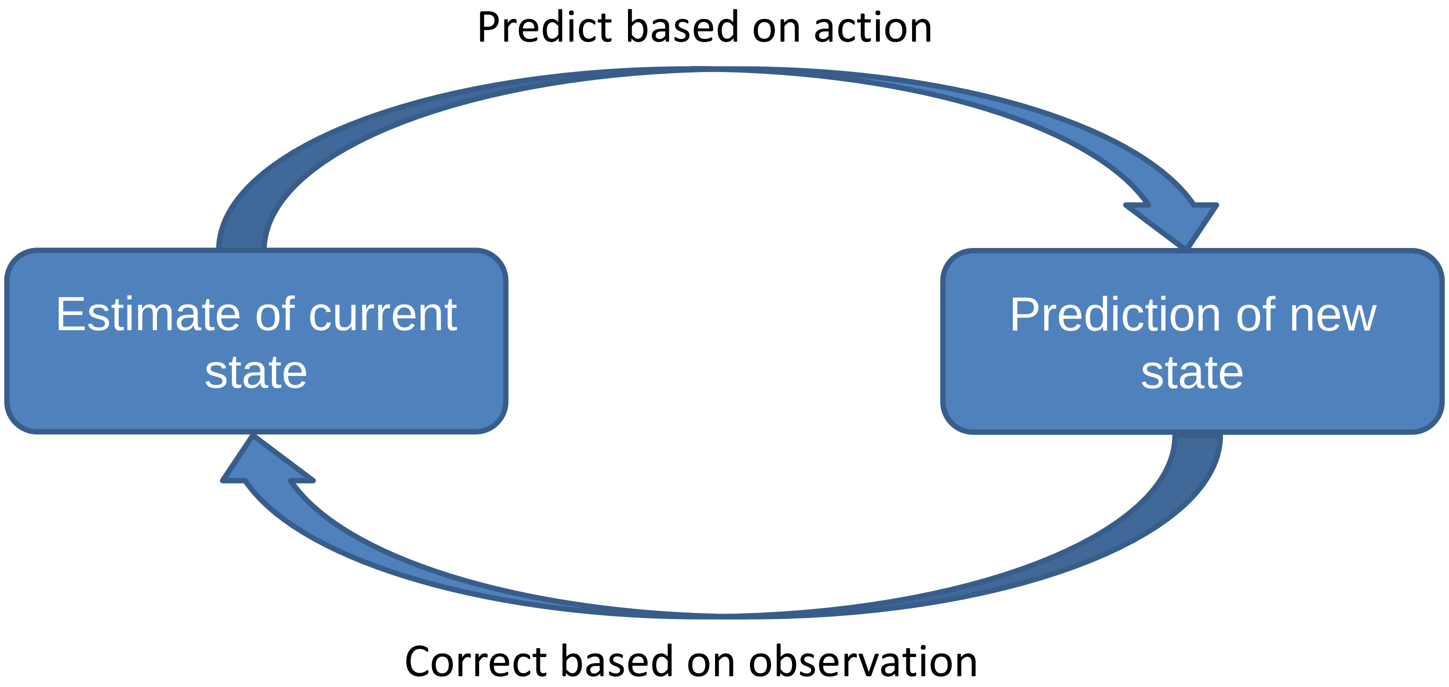
Problem Formulation

- Linear system
  - $x_t = Ax_{t-1} + Bu_{t-1} + \epsilon_t$ $\qquad \epsilon_t \sim N(0, Q)$
  - $z_t = Hx_t + \delta_t$ $\qquad \delta_t \sim N(0, R)$
- Gaussian noise in both models
  - Posterior belief is Gaussian (see later)
- Given:
  - $z_{t-1}, \dots, z_0$ and $u_{t-1}, \dots, u_0$
- Estimate:
  - State x_t
  - Error covariance P_t

Think of these as the mean and covariance of the Gaussian distribution over what we believe to be the true state

Gaussians stay Gaussian as long as all transformations are linear

Kalman filter overview



- Predictor-corrector algorithm
- Need a dynamical model, and a stream of observations
- At each step: state is weighted average between
  - Prediction from action model: $P(x_t|u_{t-1}, x_{t-1})$
  - Correction from sensor model: $P(z_t|x_t)$

KF: Predictor-Corrector

- **Predictor (time update):**
- Update expected value of x
  - $x_t^- = Ax_{t-1} + Bu_{t-1}$
- Update error covariance matrix
  - $P_t^- = AP_{t-1}A^T + Q$
- **Corrector (measurement update):**
- Update expected value
  - $x_t = x_t^- + K_t \underbrace{(z_t - Hx_t^-)}$
- Update error covariance matrix
  - $P_t = (I - K_t H)P_t^-$

Linear system:

$$x_t = Ax_{t-1} + Bu_{t-1} + \epsilon_t \quad \epsilon_t \sim N(0, Q)$$
$$z_t = Hx_t + \delta_t \quad \delta_t \sim N(0, R)$$

Optimal Kalman gain K_t is:
$K_t = P_t^- H^T (HP_t^- H^T + R)^{-1}$
Trade-off between confidence in estimates vs noise

Innovation term: difference between observation and expected observation

KF: Predictor-Corrector Intuitions

- **Predictor:**
- Update expected value of x
  - $x_t^- = Ax_{t-1} + Bu_{t-1}$
- Update error covariance matrix
  - $P_t^- = AP_{t-1}A^T + Q$
- **Corrector:**
- Update expected value
  - $x_t = x_t^- + K_t(z_t - Hx_t^-)$
- Update error covariance matrix
  - $P_t = (I - K_tH)P_t^-$

Update state estimate from system dynamics

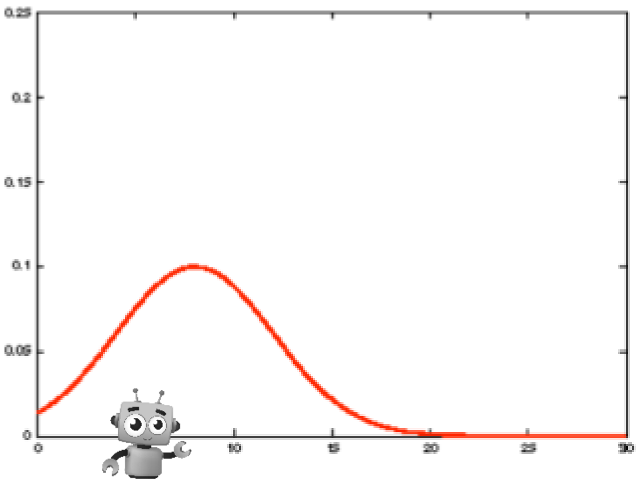
Uncertainty estimate grows from last step, by dynamics and noise

Correct based on expected observation error, and Kalman gain (based on uncertainty estimate and observation noise)

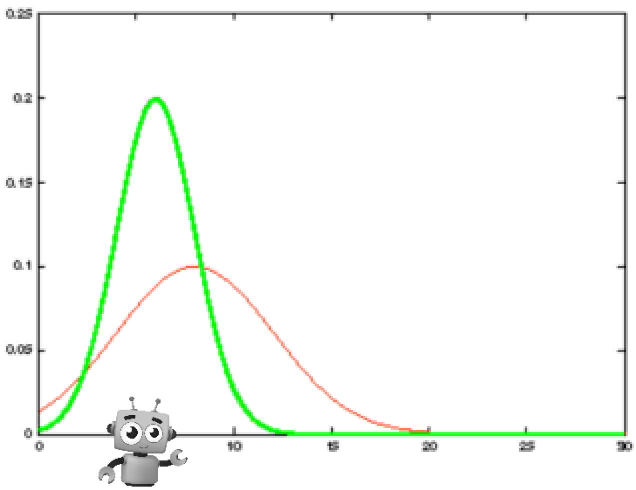
Uncertainty estimate shrinks based on relationship between state and observations, and Kalman gain

Example

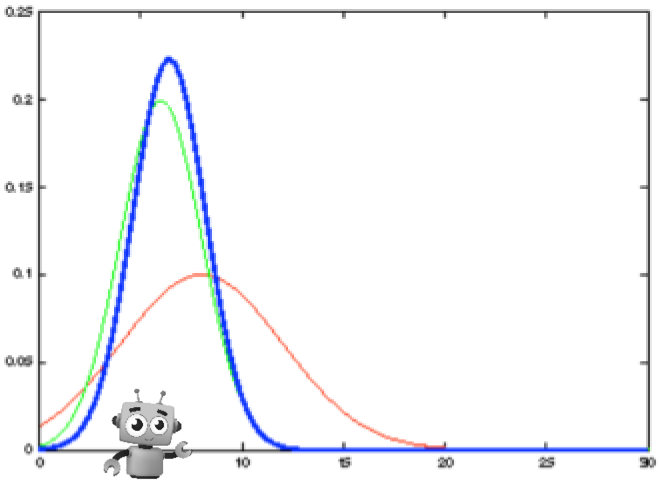
Initial belief



Observation

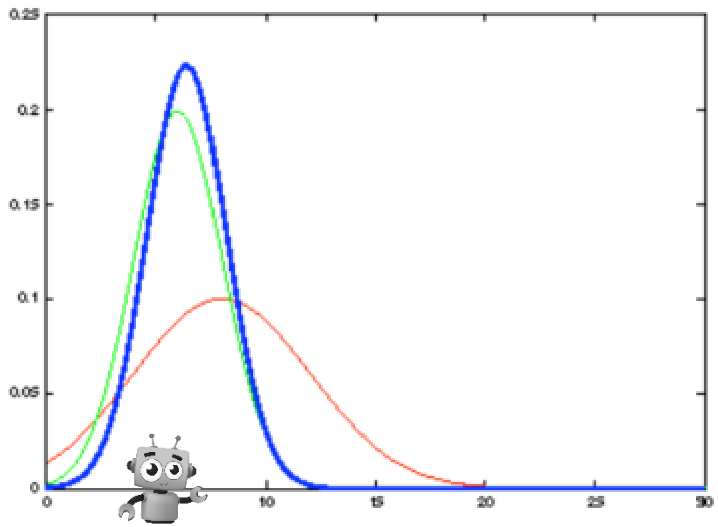


Updated belief

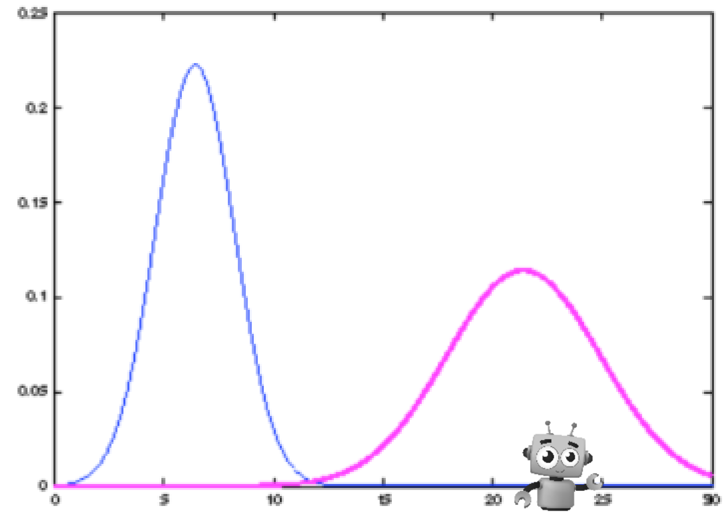


Example

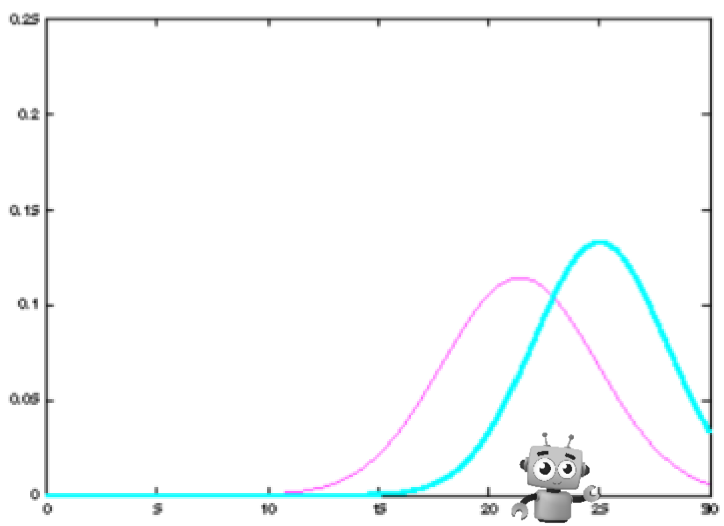
Prior belief



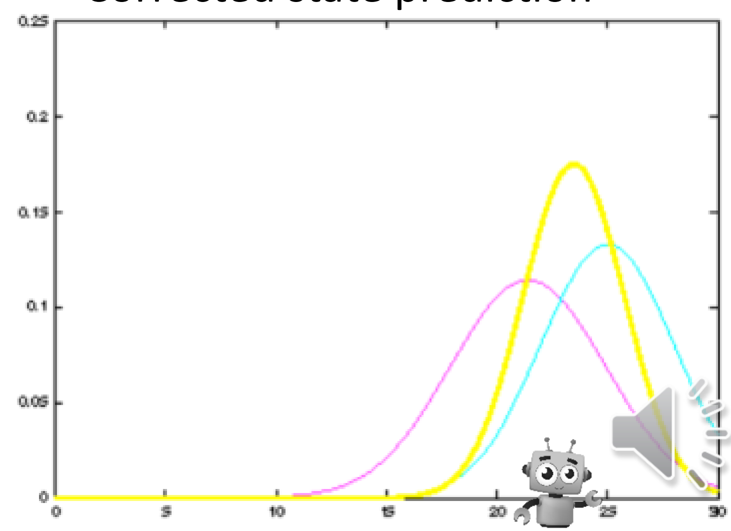
State prediction (from action)



Observation



Corrected state prediction



Performance

- Efficient: $O(t^{2.376} + n^2)$
- Closed-form solution
- Optimal for linear Gaussian systems
  - But most robotic systems are nonlinear
- Limitations:
  - Linear models
  - Unimodal Gaussian distributions

Extended Kalman Filter

- Non-linear system with additive noise?
- Extended Kalman filter
  - Linearise nonlinear system
  - Approximate with first-order Taylor series expansion at state estimators
  - Greater errors
- Still assume noise and error models are Gaussian

Linearising

- Non-linear system
  - $x_t = f(x_{t-1}, u_{t-1}) + \epsilon_t$ $\quad\quad \epsilon_t \sim N(0, Q)$
  - $z_t = h(x_t) + \delta_t$ $\quad\quad \delta_t \sim N(0, R)$
- Let A be the Jacobian of f with respect to x
  - $A_{ij} = \frac{\partial f_i}{\partial x_j}(x_{t-1}, u_{t-1})$
- Let H be the Jacobian of h with respect to x
  - $H_{ij} = \frac{\partial h_i}{\partial x_j}(x_t)$
- Update equations almost as before!

EKF Update Equations

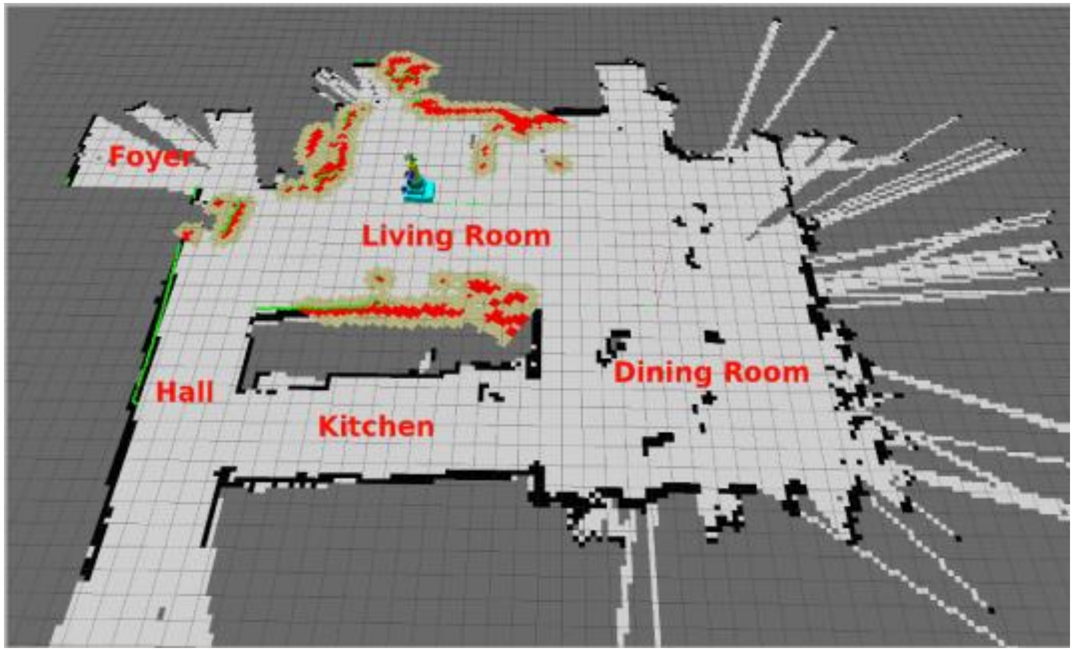
- Predictor step:
  - $x_t^- = f(x_{t-1}, u_{t-1})$
  - $P_t^- = AP_{t-1}A^T + Q$
- Kalman gain:
  - $K_t = P_t^- H^T (HP_t^- H^T + R)^{-1}$
- Corrector step:
  - $x_t = x_t^- + K_t\left(z_t - h(x_t^-)\right)$
  - $P_t = (I - K_t H)P_t^-$

Evaluate with true functions

Update with approximations

Localisation

- Localisation (where am I in the world?)
- Given a map, where is the robot?
  - Landmarks are known, robot's position is not
  - From sensor readings, infer most likely position

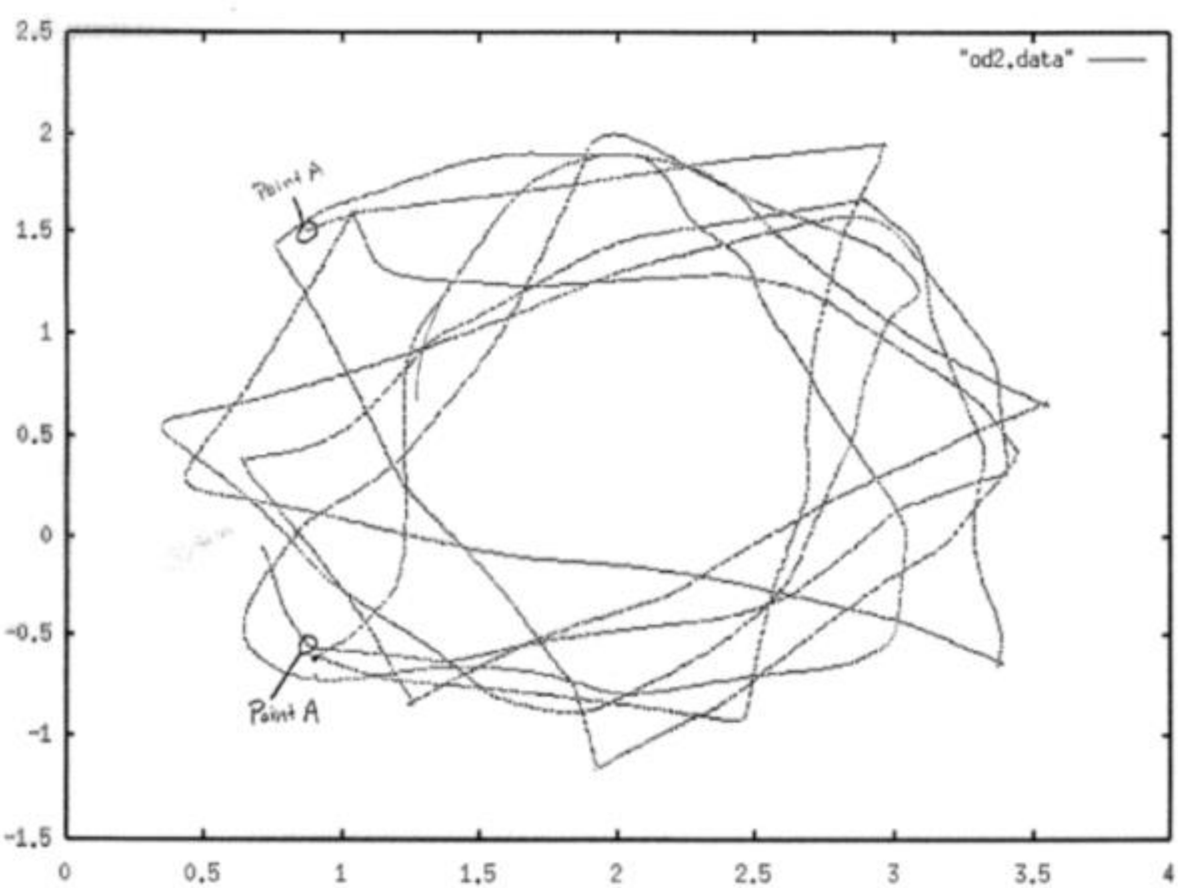


Localisation

- Localisation
    - Sense
    - Relate sensor reading to a world model
    - Update location relative to world model
- Assumes: perfect world model (see SLAM)
- Kalman filter!
- But: assumes Gaussian – no good in general
    - What if I don't know where I started?
    - Multimodal distribution

Localisation Errors

- Odometry-only tracking
  - Driving 6 times around a 2m x 3m area

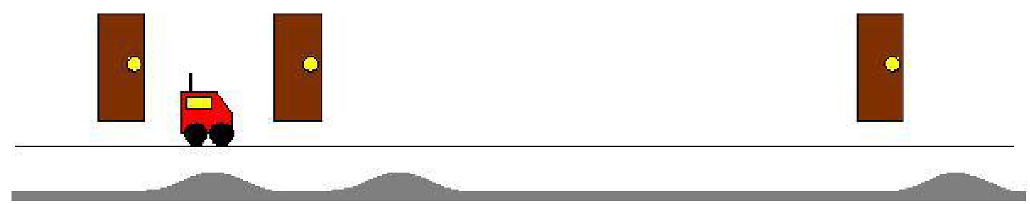


Localising with Doors

Initial state
detects nothing:

Moves and
detects landmark:

Moves and
detects nothing:



Moves and
detects landmark:

Recap: Probability

- Conditional probability:
  - $P(x, y) = P(x|y)P(y)$
- Marginalising:

Discrete	Continuous case
$\sum_y P(y) = 1$	$\int p(y)\, dy = 1$
$P(x) = \sum_y P(x, y)$	$p(x) = \int p(x, y)\, dy$
$P(x) = \sum_y P(x \mid y) P(y)$	$p(x) = \int p(x \mid y)\, p(y)\, dy$

Bayes' Rule

- From conditional probability:
  - $P(x, y) = P(x|y)P(y) = P(y|x)P(x)$
- Bayes' rule/theorem/law:

Rev. Thomas Bayes

Likelihood of data y under x

Prior probability of x

- $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$

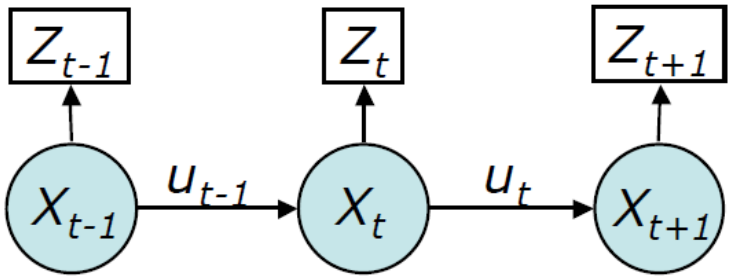
Probability of data y

Posterior probability of x,
having seen y

The Markov Assumption

- The Markov assumption:
  - Given the present, the future is independent of the past.

- Probabilistic graphical model of the problem
  - Edges denote dependence



- Given state x_t, observation z_t (and state x_{t+1}) is independent of the past
  - $P(z_t | x_t, x_1, u_1, z_1, \ldots, u_{t-1}) = P(z_t | x_t)$

The Bayes Filter

- In general, updating estimates of x may be arbitrary distributions over x
  - Posterior distribution over x = "belief"
  - Evaluate from every x_{t-1} to x_t
  - Computational issues
- $Bel(x_t) = \eta P(z_t|x_t) \int P(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$
- Kalman filter = special case
  - Bayes filter using Gaussians as beliefs
  - Reduces to matrix multiplications

Derivation

- Estimate state given data:
  - $Bel(x_t) = P(x_t | D_T)$
- Data D_T is a sequence of observations z_i and actions u_i
  - $Bel(x_t) = P(x_t | z_t, u_{t-1}, z_{t-1}, u_{t-2}, \dots, z_0)$
- Using Bayes rule
  - $Bel(x_t) = \frac{P(z_t | x_t, u_{t-1}, z_{t-1}, \dots, z_0) P(x_t | u_{t-1}, z_{t-1}, \dots, z_0)}{P(z_t | u_{t-1}, z_{t-1}, \dots, z_0)}$
- Denominator is a constant relative to x_t
  - $\eta = 1/P(z_t | u_{t-1}, z_{t-1}, \dots, z_0)$
  - $Bel(x_t) = \eta P(z_t | x_t, u_{t-1}, \dots, z_0) P(x_t | z_{t-1}, \dots, z_0)$

Derivation

- Markov assumption on first term
  - $Bel(x_t) = \eta P(z_t|x_t) P(x_t|u_{t-1}, \dots, z_0)$

- Expand the last term
  - $Bel(x_t) = \eta P(z_t|x_t) \int P(x_t|x_{t-1}, u_{t-1}, \dots, z_0) P(x_{t-1}|u_{t-1}, \dots, z_0) dx_{t-1}$

- Markov assumption on middle term
  - $Bel(x_t) = \eta P(z_t|x_t) \int P(x_t|x_{t-1}, u_{t-1}) P(x_{t-1}|u_{t-1}, \dots, z_0) dx_{t-1}$

- Substitute in the definition $Bel(x_{t-1})$
  - $Bel(x_t) = \eta P(z_t|x_t) \int P(x_t|x_{t-1}, u_{t-1}) Bel(x_{t-1}) dx_{t-1}$

The Iterative Filter

- Propagate the motion model
  - $Bel^{-}(x_t) = \int P(x_t | x_{t-1}, u_{t-1}) Bel(x_{t-1}) dx_{t-1}$
  - Compute current state estimate before taking sensor reading by integrating over all possible previous estimates and applying the motion model
  - (c.f. KF predictor step)

- Update the sensor model
  - $Bel(x_t) = \eta P(z_t | x_t) Bel^{-}(x_t)$
  - Compute current state estimate by weighting the current motion-based estimate by the prob of the sensor reading
  - (c.f. KF corrector step)

Bayes Filter Algorithm

Algorithm **Bayes_filter**(*Bel(x),d*):

η=0

If *d* is a perceptual data item *z* then

  For all *x* do

    $$Bel'(x) = P(z \mid x) Bel(x)$$

    $$\eta = \eta + Bel'(x)$$

  For all *x* do

    $$Bel'(x) = \eta^{-1} Bel'(x)$$

Else if *d* is an action data item *u* then

  For all *x* do

    $$Bel'(x) = \int P(x \mid u, x')\, Bel(x')\, dx'$$

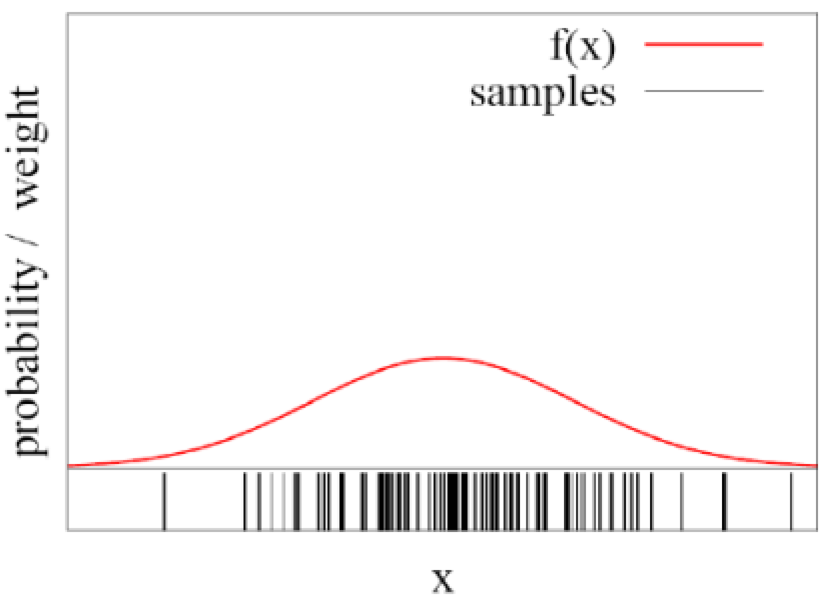
Return *Bel'(x)*

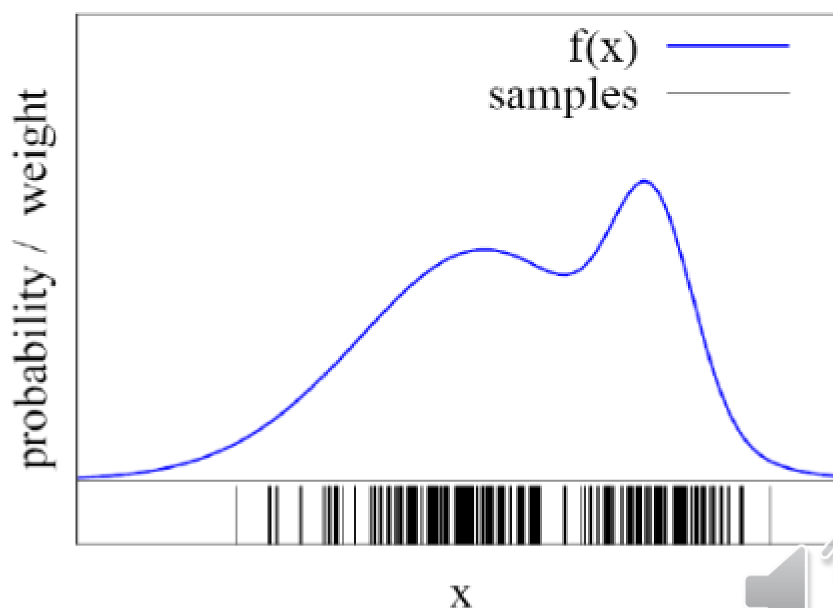
Particle Filters

- Computing arbitrary probability distributions over a continuous space is hard
- Instead:
  - Approximate the distribution
  - Simulate a concrete set of samples (particles, poses)
  - Efficient way to represent non-Gaussian distribution
- Use N random samples $x_t(i)$
  - Each has a weight initialised to $w_t(i) = 1/N$
- Belief = weighted set of samples $\{\langle x_t(i), w_t(i)\rangle\}$, $\Sigma_t\, w_t = 1$
- A Bayes filter with this representation is a **particle filter**

Approximating Functions with Samples

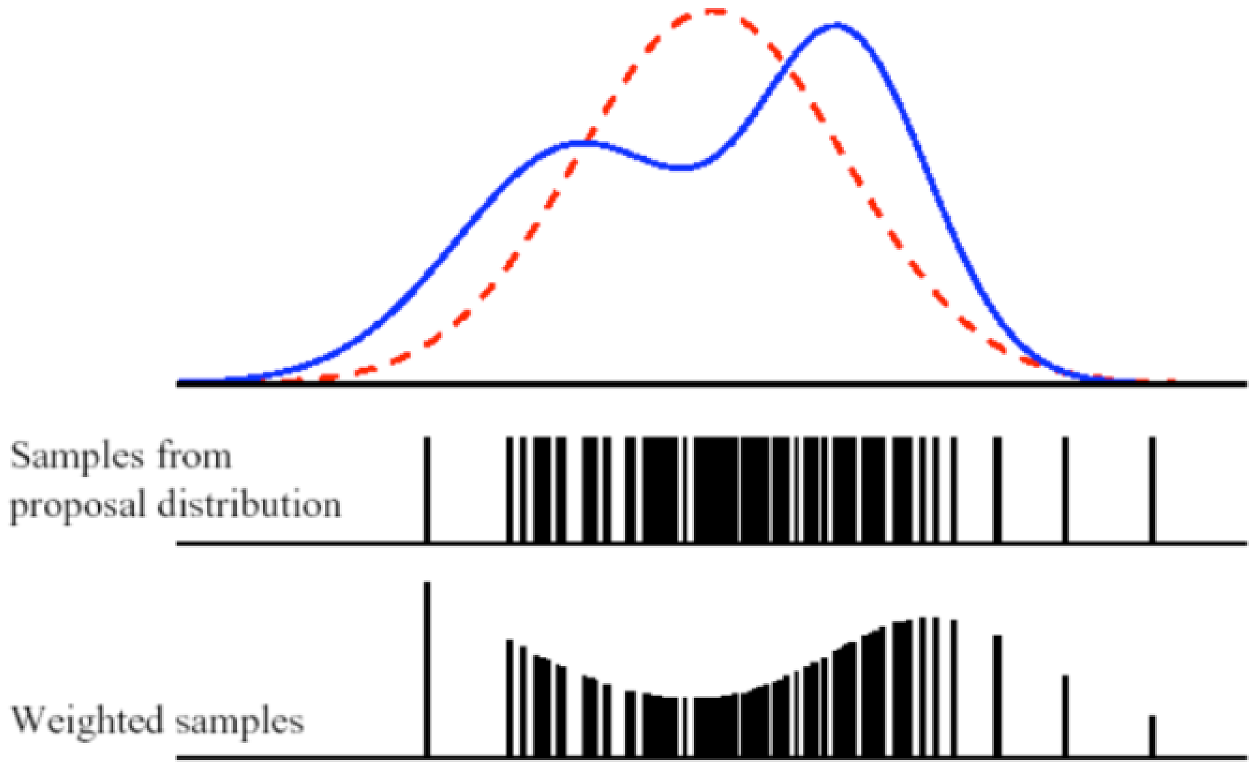
- Particle sets can approximate functions
- The more particles in an interval, the higher its probability
- How to draw samples?





Importance Sampling

- Sample from a proposal distribution $q(x)$
  - Correct (re-weight) to approximate a target distribution $p(x)$
  - Importance weights $w(x) = p(x)/q(x)$



The PF Algorithm

- Repeat to collect N samples
  - Draw sample x_{t-1} from distribution $Bel(x_{t-1})$, with likelihood given by its weight
  - Given the action u_{t-1} and action model distribution $P(x_t|u_{t-1}, x_{t-1})$, sample state x_t
  - Assign weight $P(z_t|x_t)$ to x_t
- Normalise weights
- Repeat for each time step

- $Bel(x_t) = \eta P(z_t|x_t) \int P(x_t|x_{t-1}, u_{t-1}) Bel(x_{t-1}) dx_{t-1}$

Visualisation of the PF

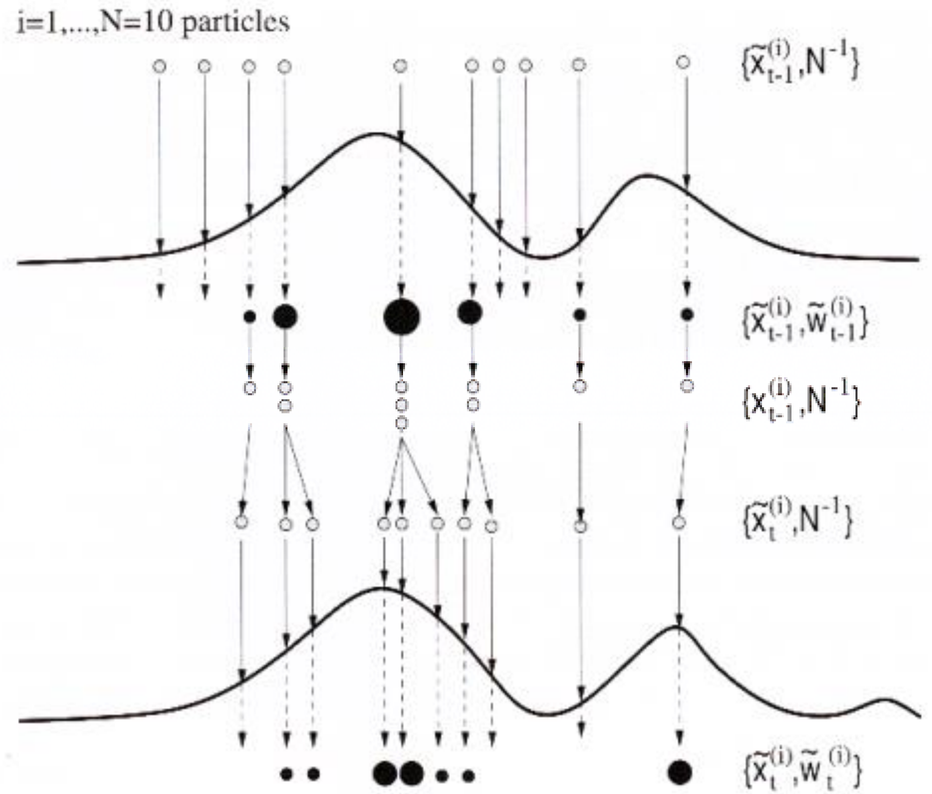
Unweighted measure

Compute weights

Resample

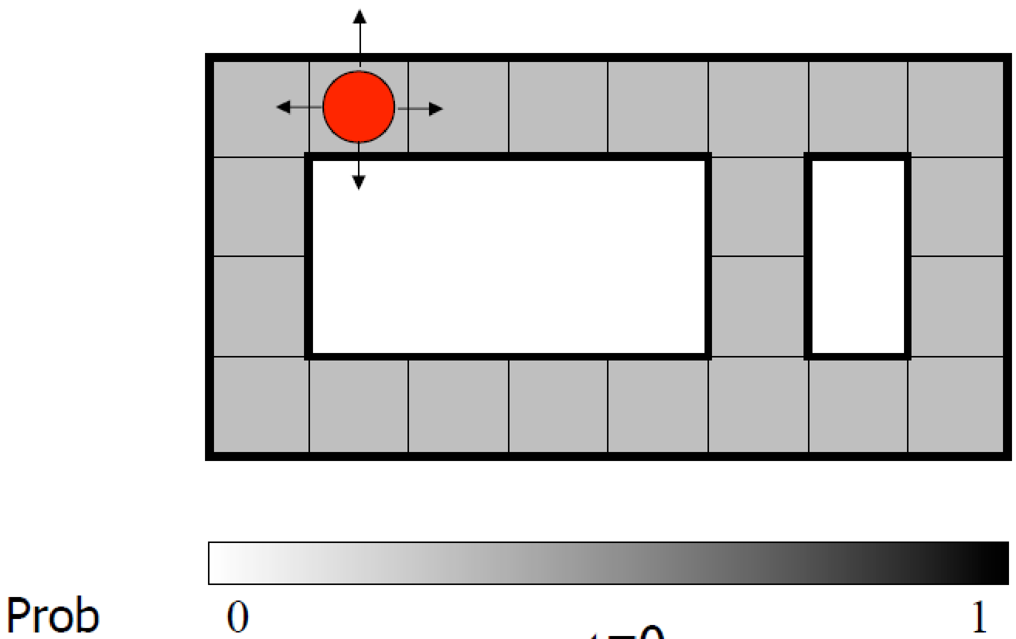
Move particles

Compute weights



Toy Example

Example from
Michael Pfeiffer



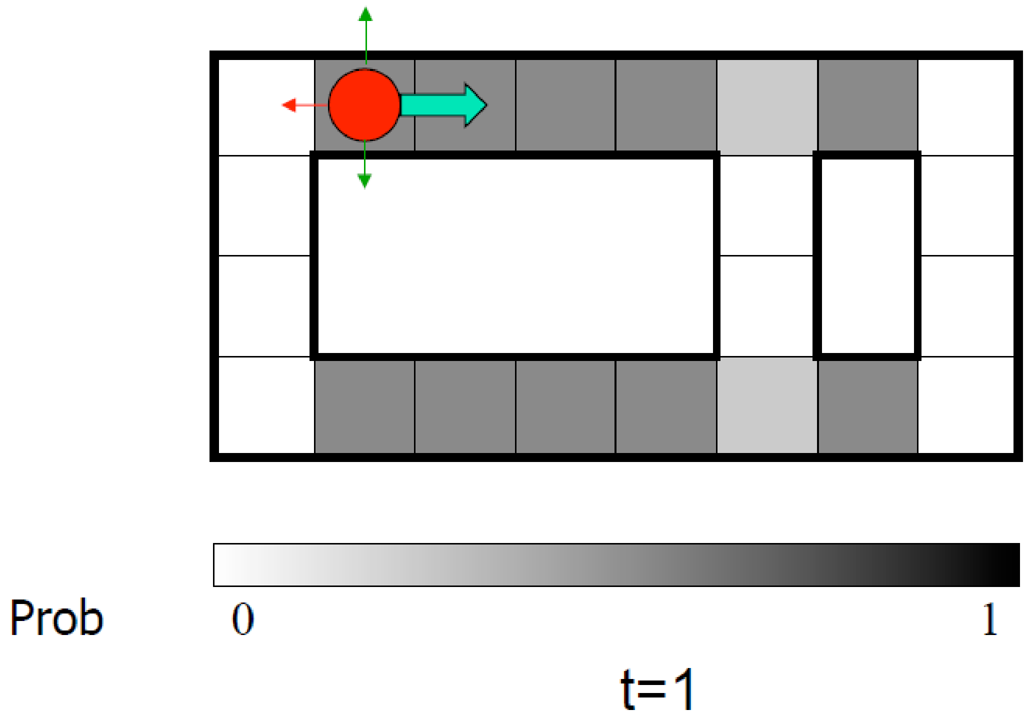
t=0

Sensor model: never more than 1 mistake

Know the heading (North, East, South or West)

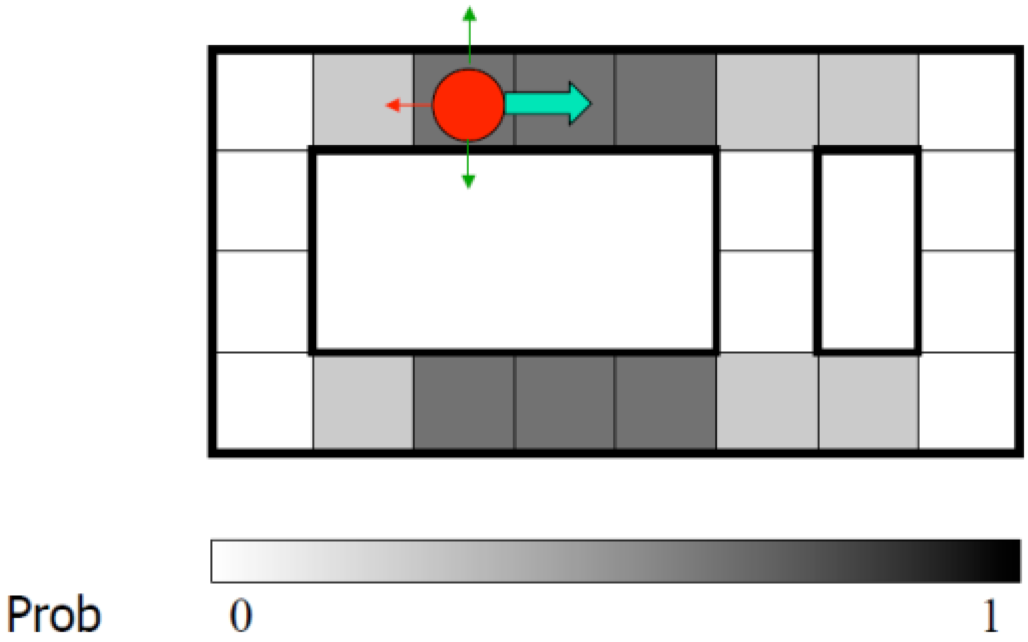
Motion model: may not execute action with small prob.

Toy Example



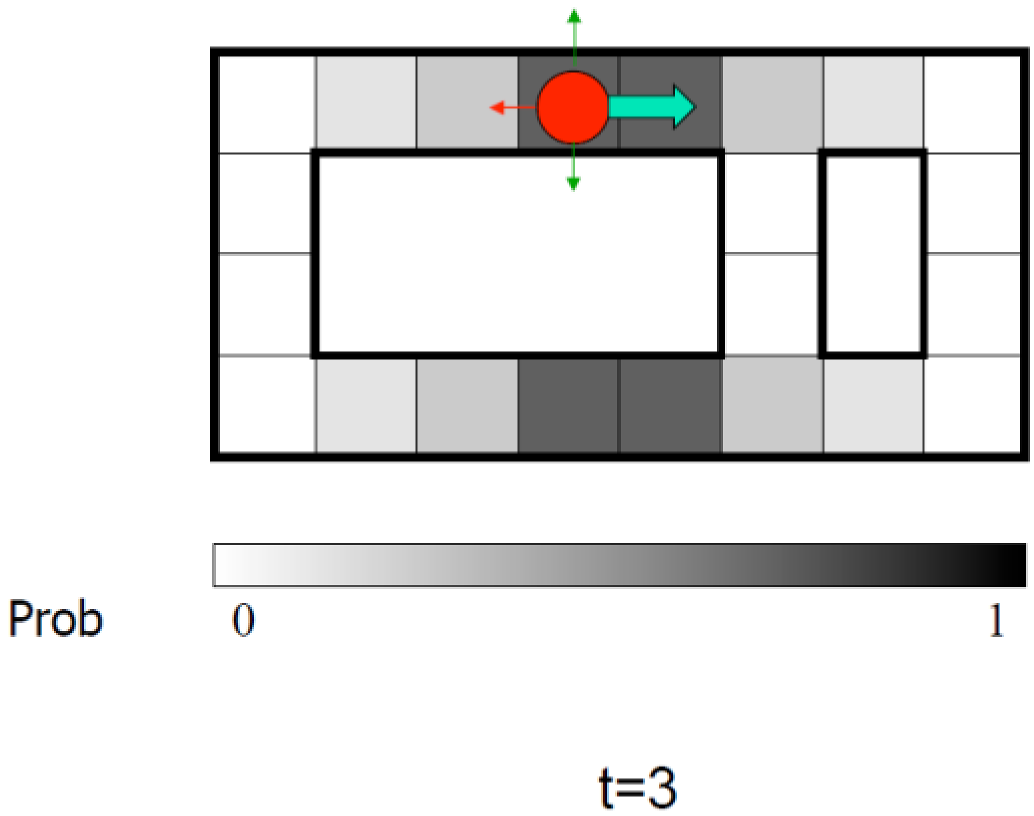
Lighter grey: was possible to get the reading, but less likely b/c required 1 mistake

Toy Example

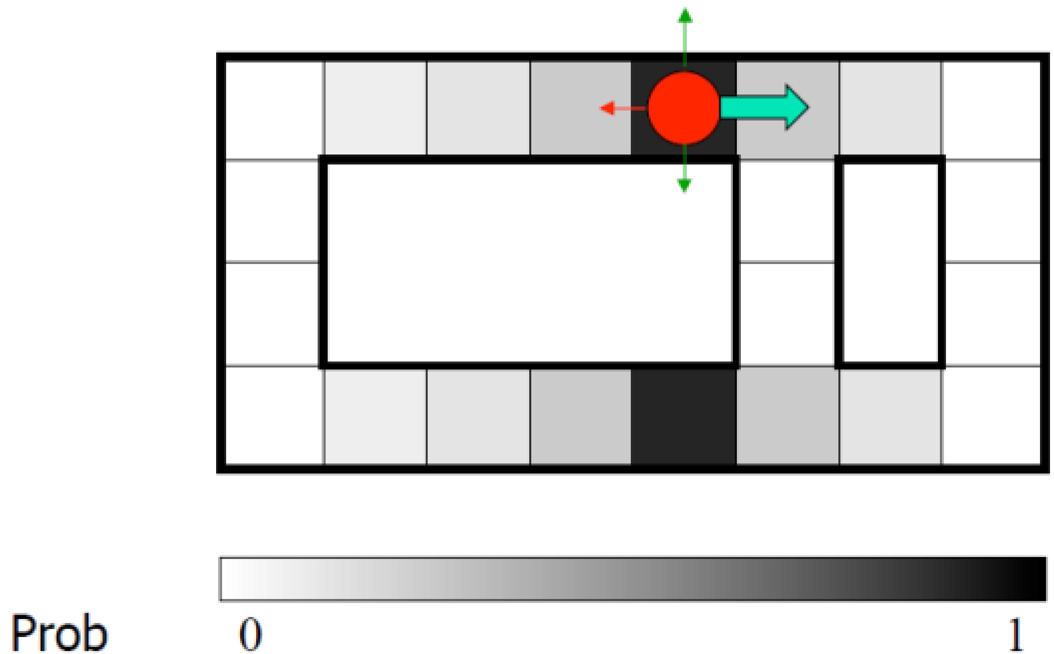


t=2

Toy Example

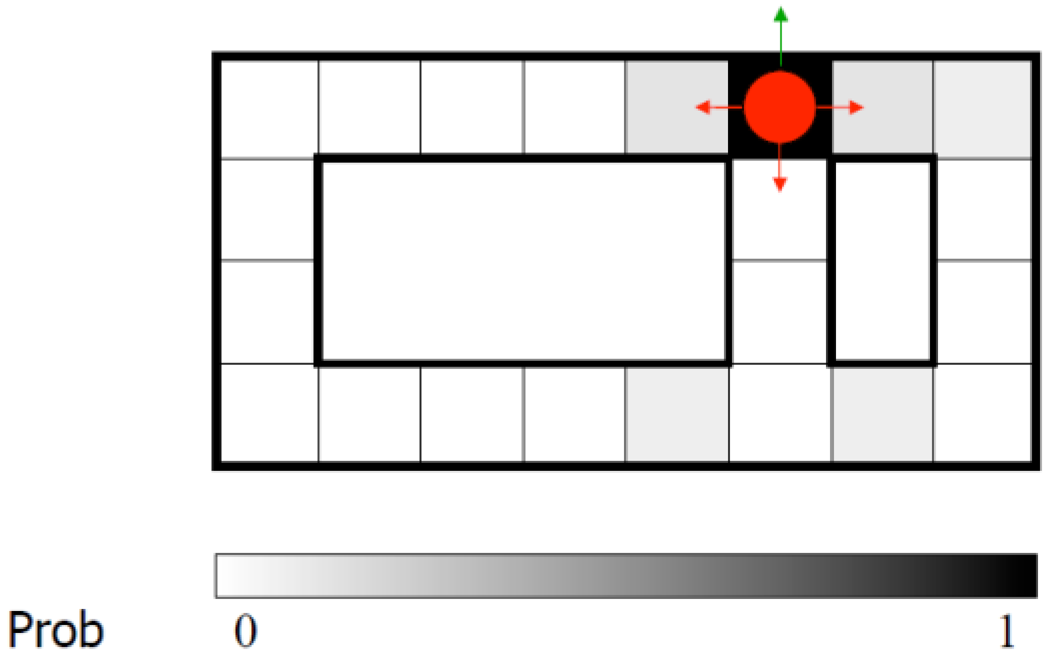


Toy Example



t=4

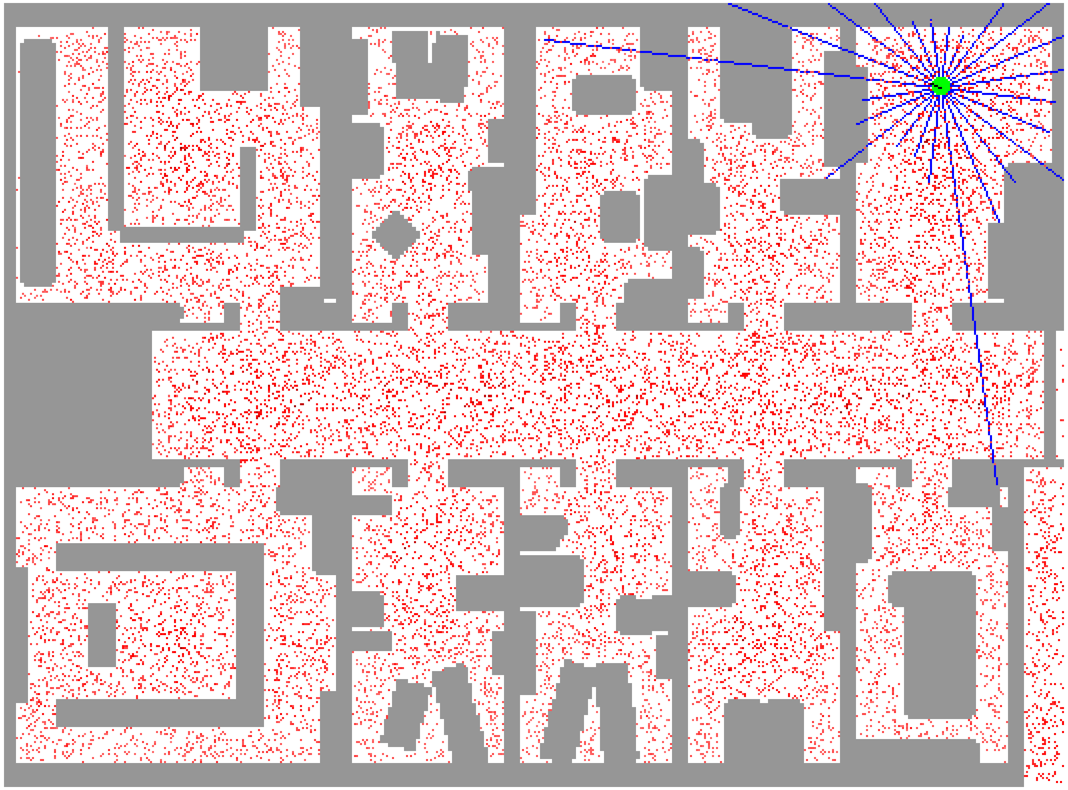
Toy Example



t=5

Exploring Robot Example

- Particle filtering



http://www.cs.washington.edu/robotics/mcl/