Filtering and Robot Localization

Robert Platt
Northeastern University

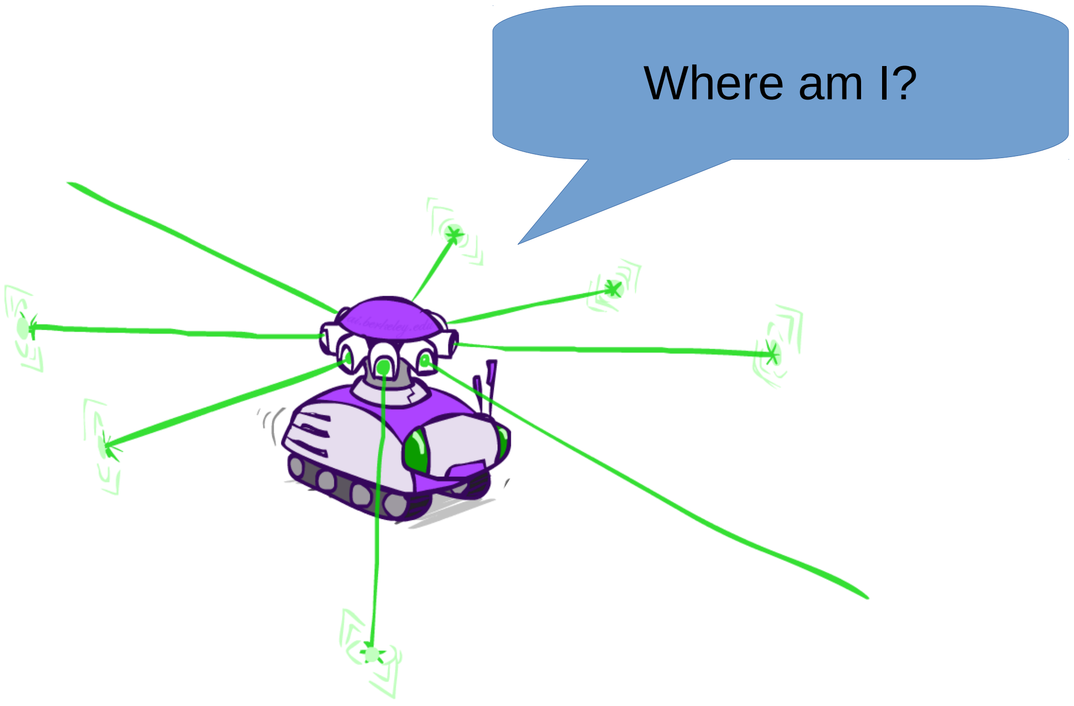
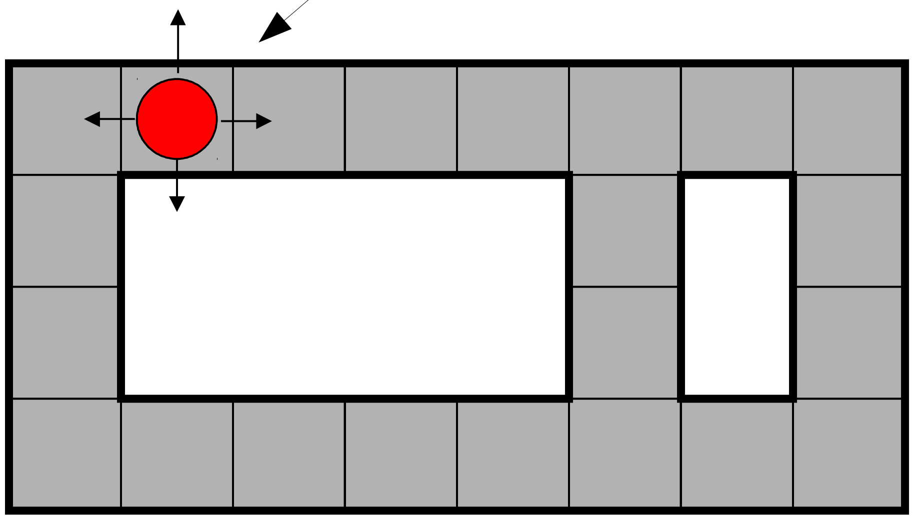
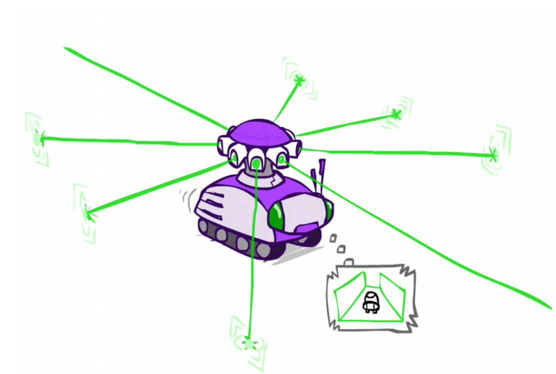


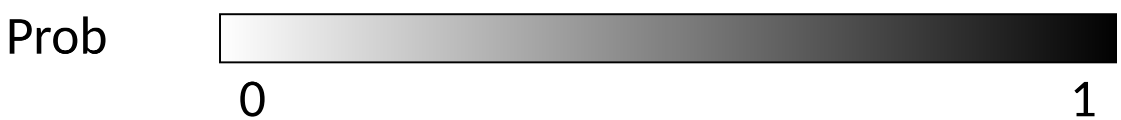
Image: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example

Robot is actually located here, but it doesn't know it.







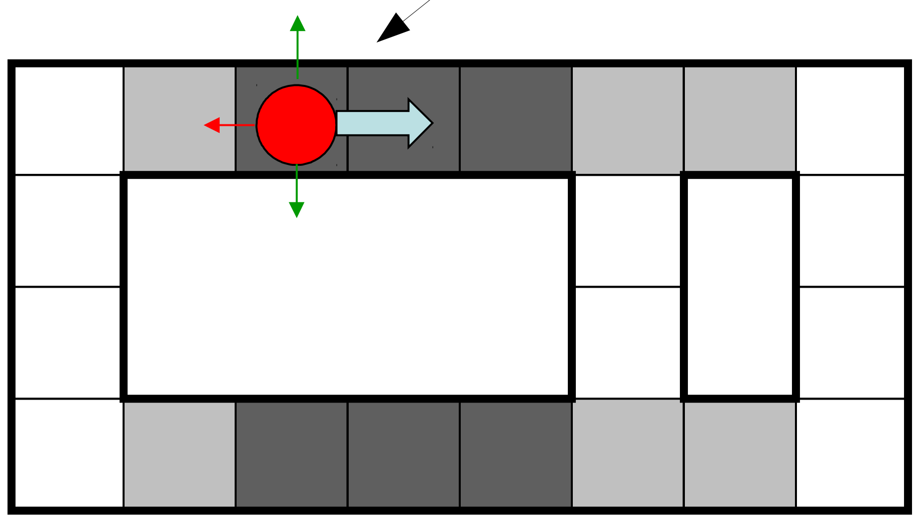
Gray level denotes estimated probability that robot is in that square

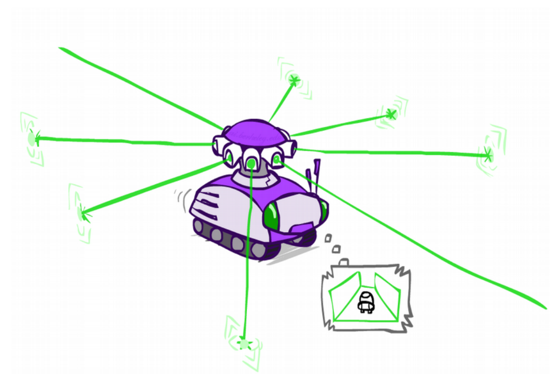
<u>Goal:</u> localize the robot based on sequential observations
– robot is given a map of the world; robot could be in any square
– initially, robot doesn't know which square it's in

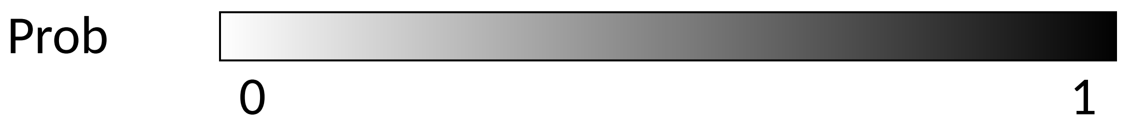
Image: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example

Robot perceives that there are walls above and below, but no walls either left or right





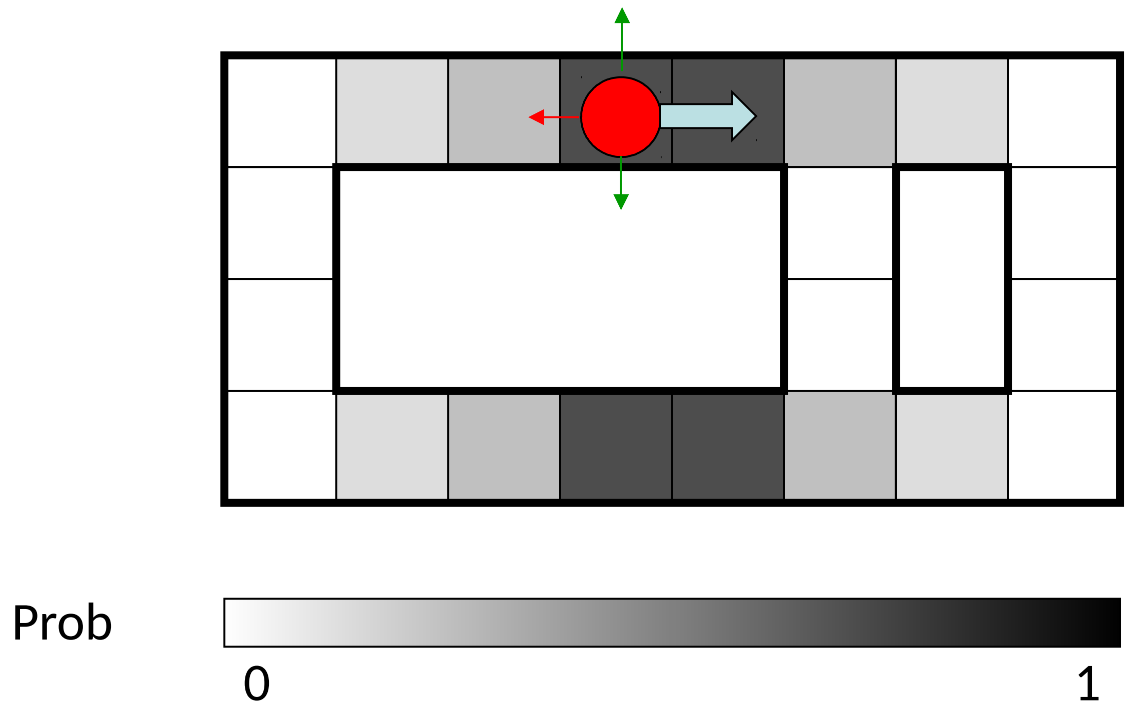


Gray level denotes estimated probability that robot is in that square

On each time step, the robot moves, and then observes the directions in which there are walls.
– observes a four-bit binary number
– observations are noisy: there is a small chance that each bit will be flipped.

Image: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example



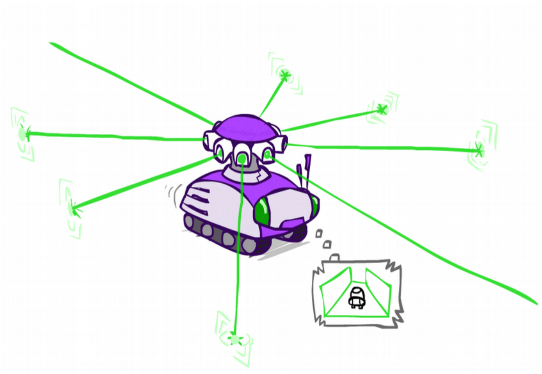
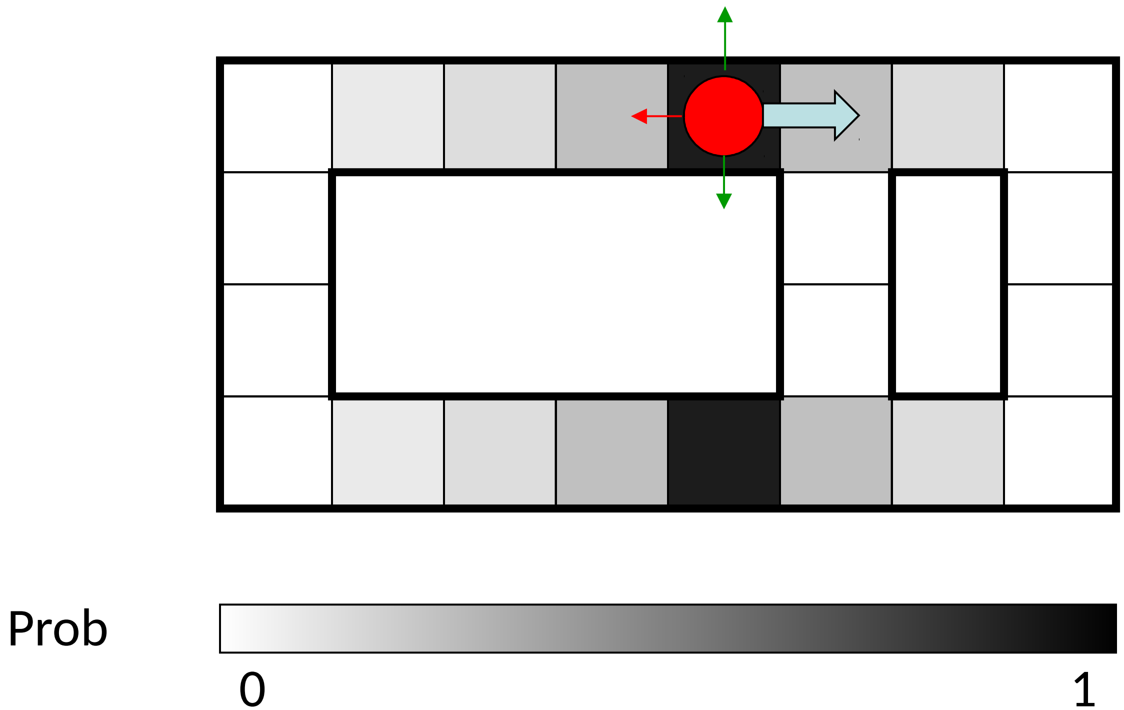


Image: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example



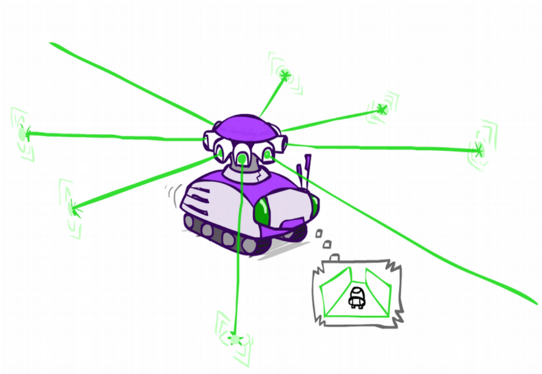
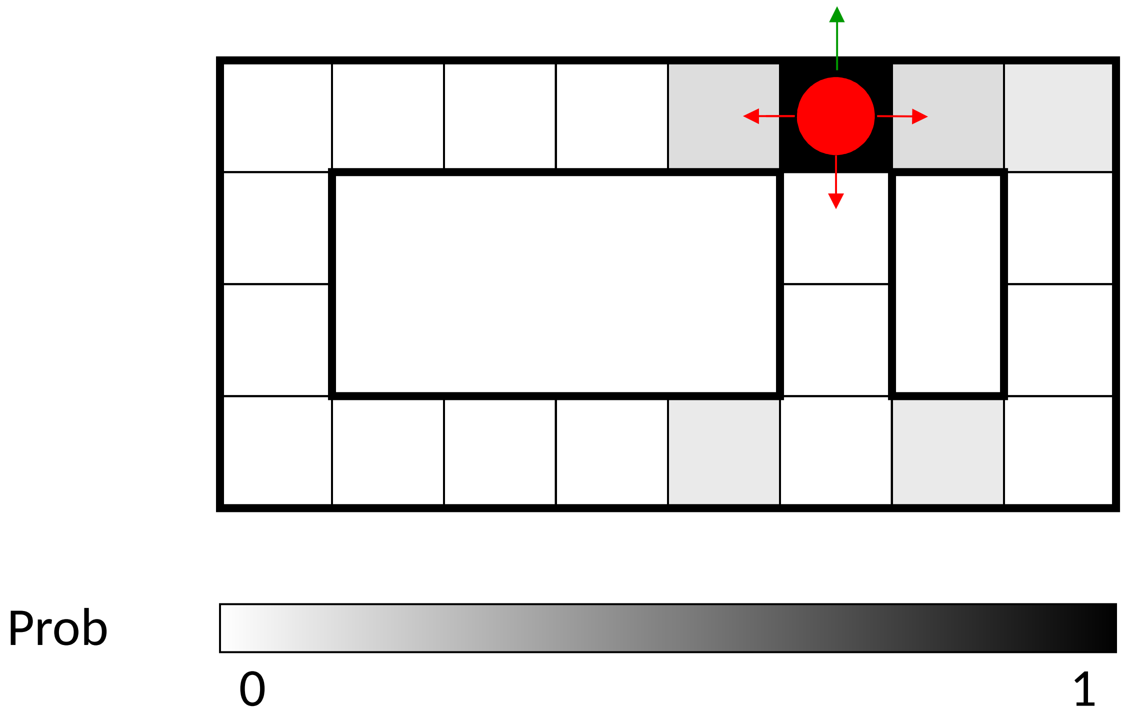


Image: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example



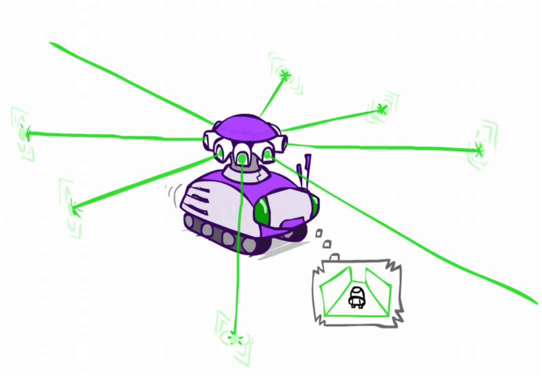
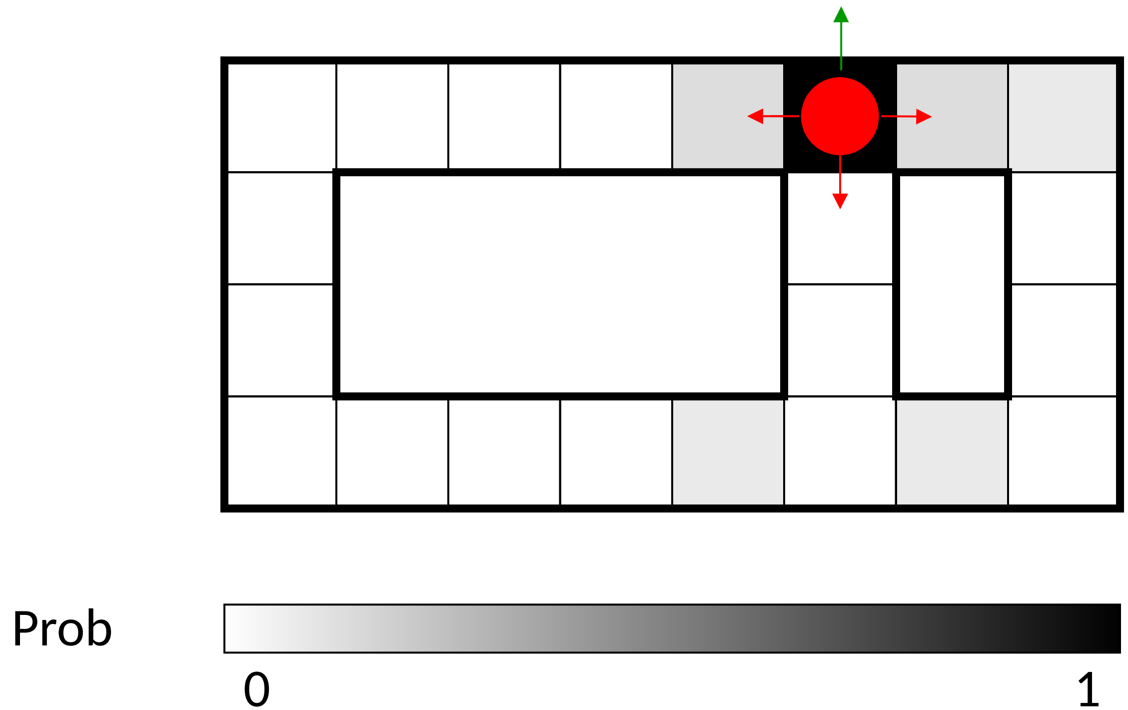
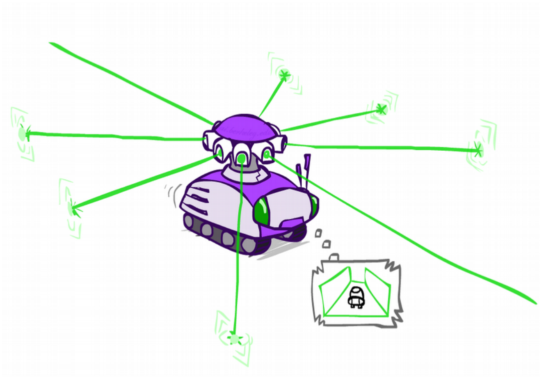


Image: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example

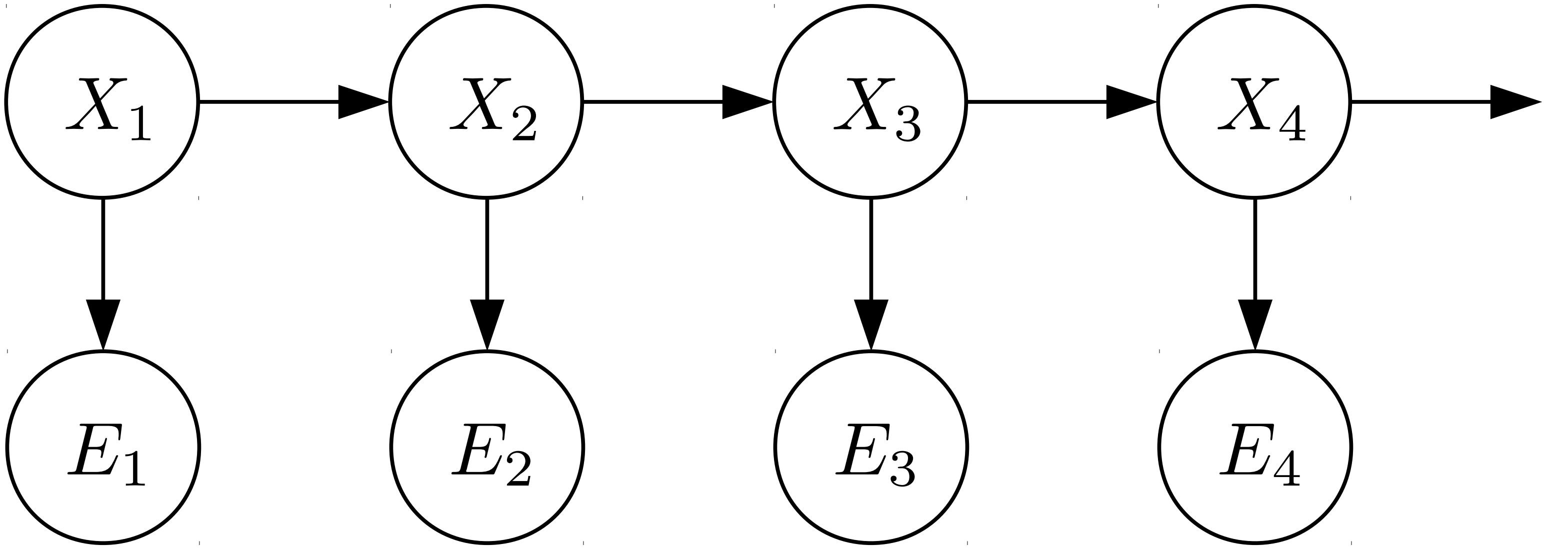




<u>Question</u>: how do we update this probability distribution from time *t* to *t+1*?

Image: Berkeley CS188 course notes (downloaded Summer 2015)

Hidden Markov Models (HMMs)

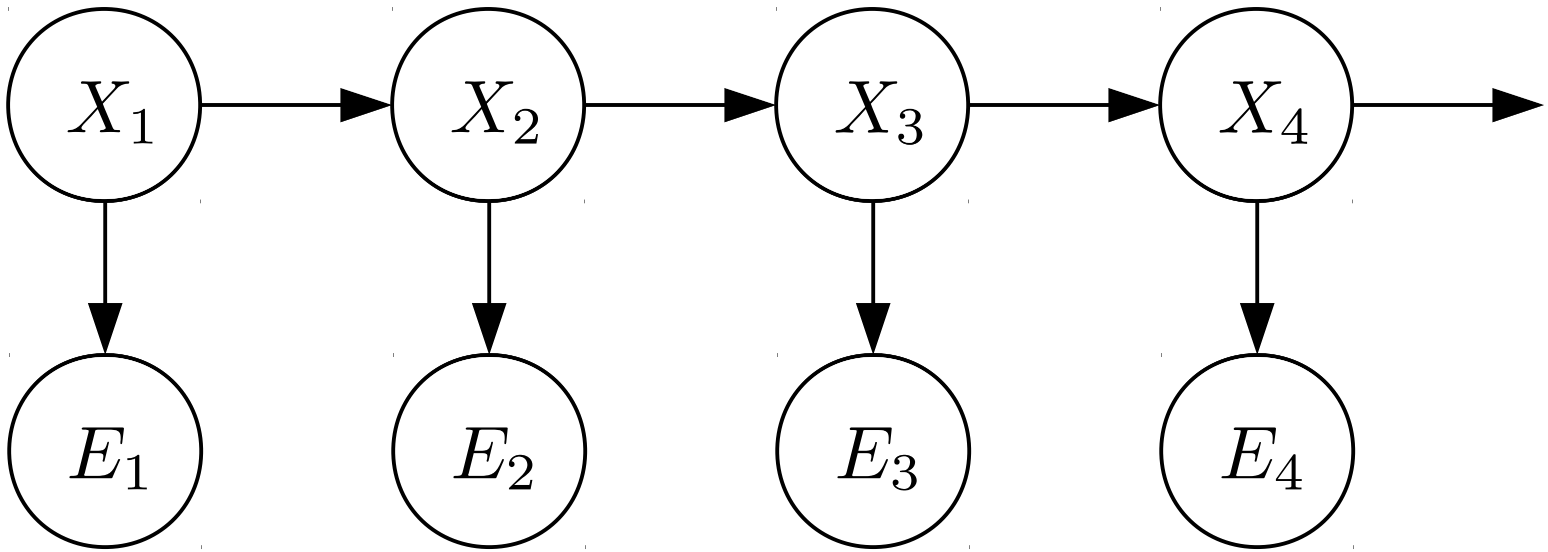


Called an "emission"

State, X_t , is assumed to be unobserved

However, you get to make one observation, E_t , on each timestep.

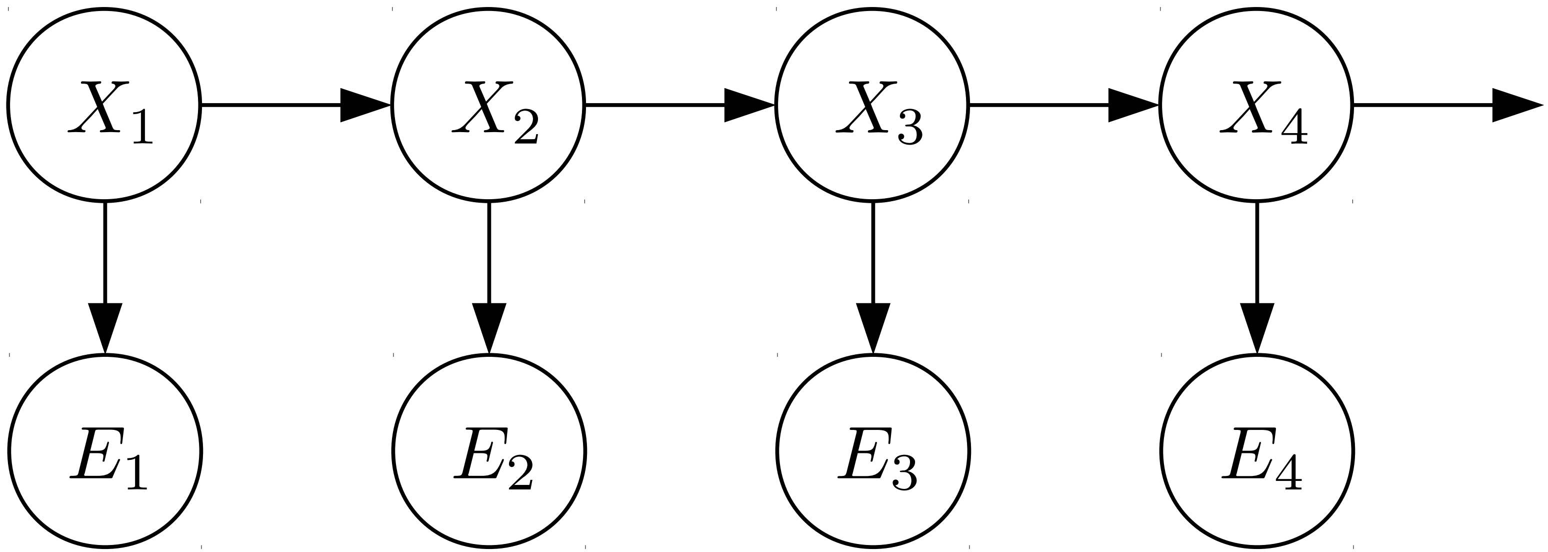
Hidden Markov Models (HMMs)



Process dynamics: $P(X_t|X_{t-1})$ ← How the system changes from one time step to the next

Observation dynamics: $P(E_t|X_t)$ ← What gets observed as a function of what state the system is in

Hidden Markov Models (HMMs)

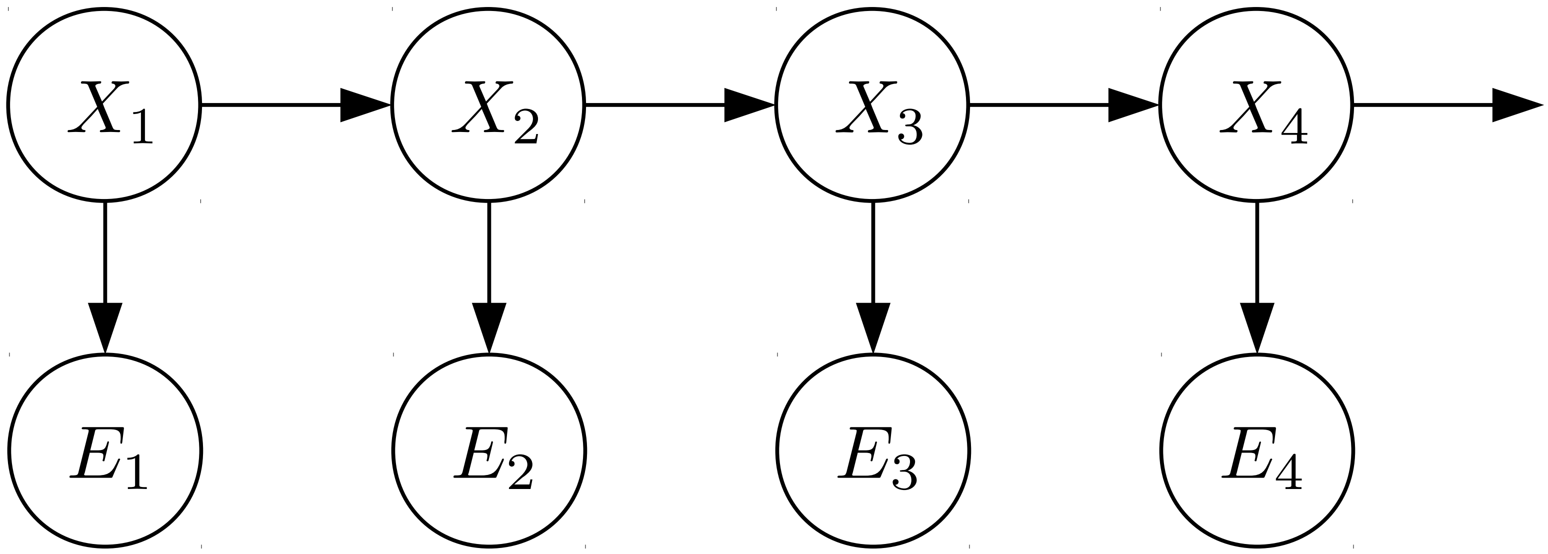


Process dynamics: $P(X_t|X_{t-1})$ ← How the system changes from one time step to the next

Observation dynamics: $P(E_t|X_t)$ ← What gets observed as a function of what state the system is in

Let's assume (for now) that these probability distributions are given to us.

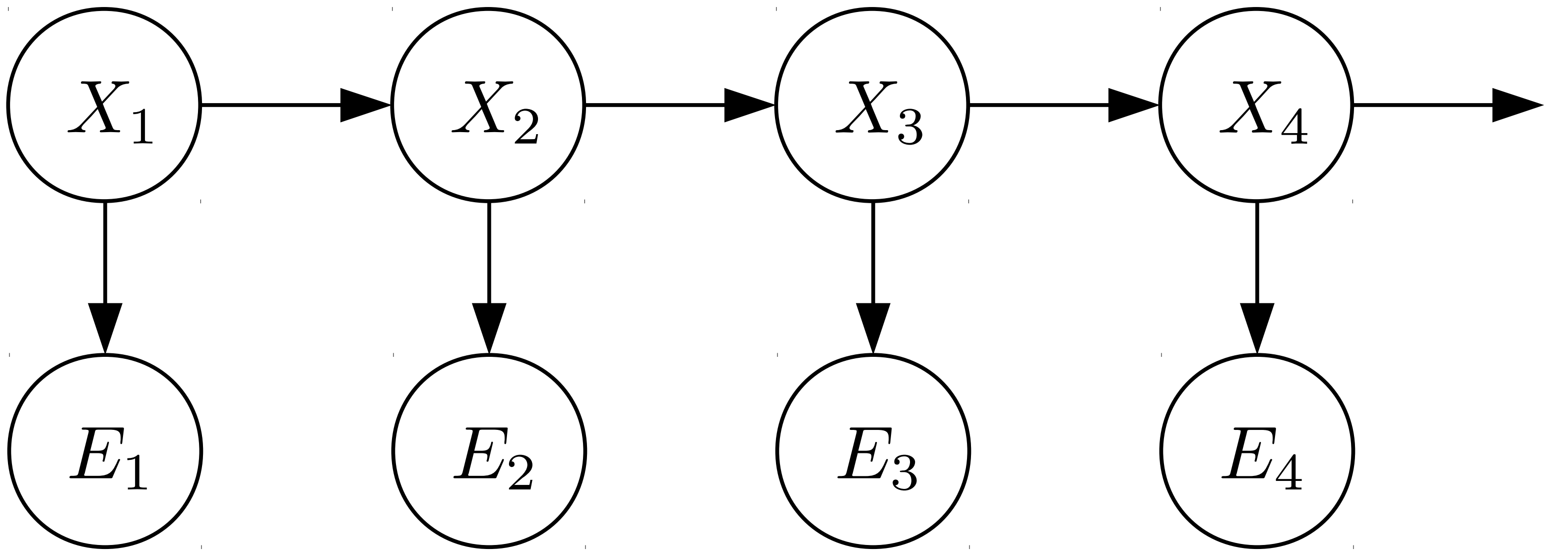
Hidden Markov Models (HMMs)



Process dynamics: $P(X_t|X_{t-1}) = P(X_t|X_{t-1},\ldots,X_1)$

Observation dynamics: $P(E_t|X_t) = P(E_t|X_t, X_{t-1},\ldots,X_1)$

Hidden Markov Models (HMMs)

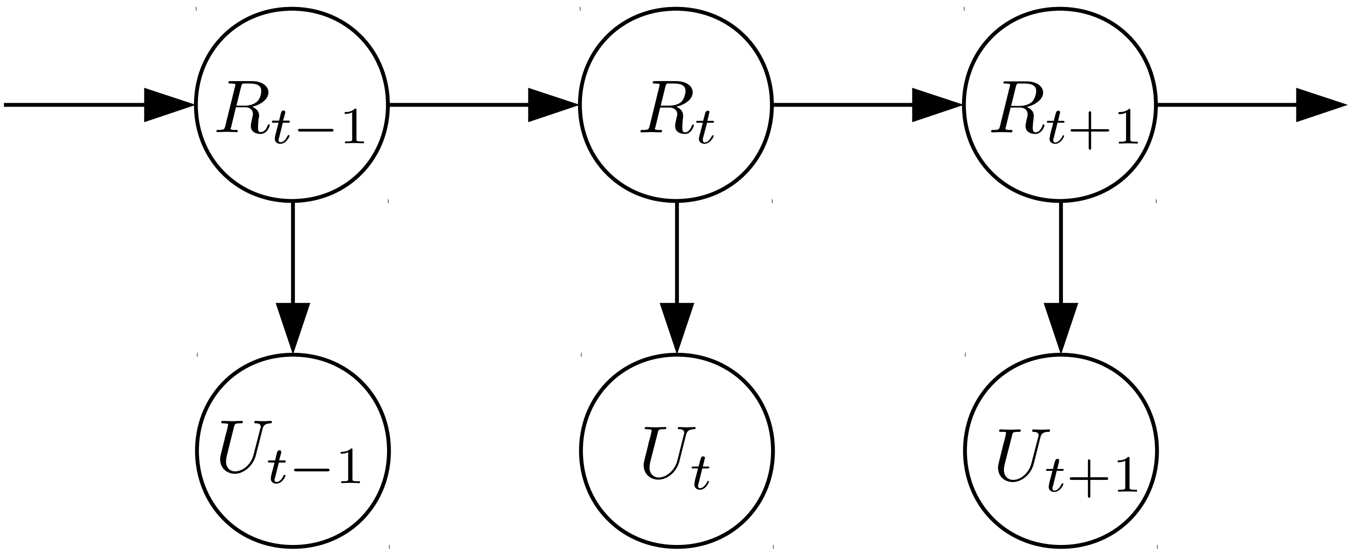


Process dynamics: $P(X_t|X_{t-1}) = P(X_t|X_{t-1},\ldots,X_1)$

Observation dynamics: $P(E_t|X_t) = P(E_t|X_t, X_{t-1},\ldots,X_1)$

Markov assumptions

HMM example

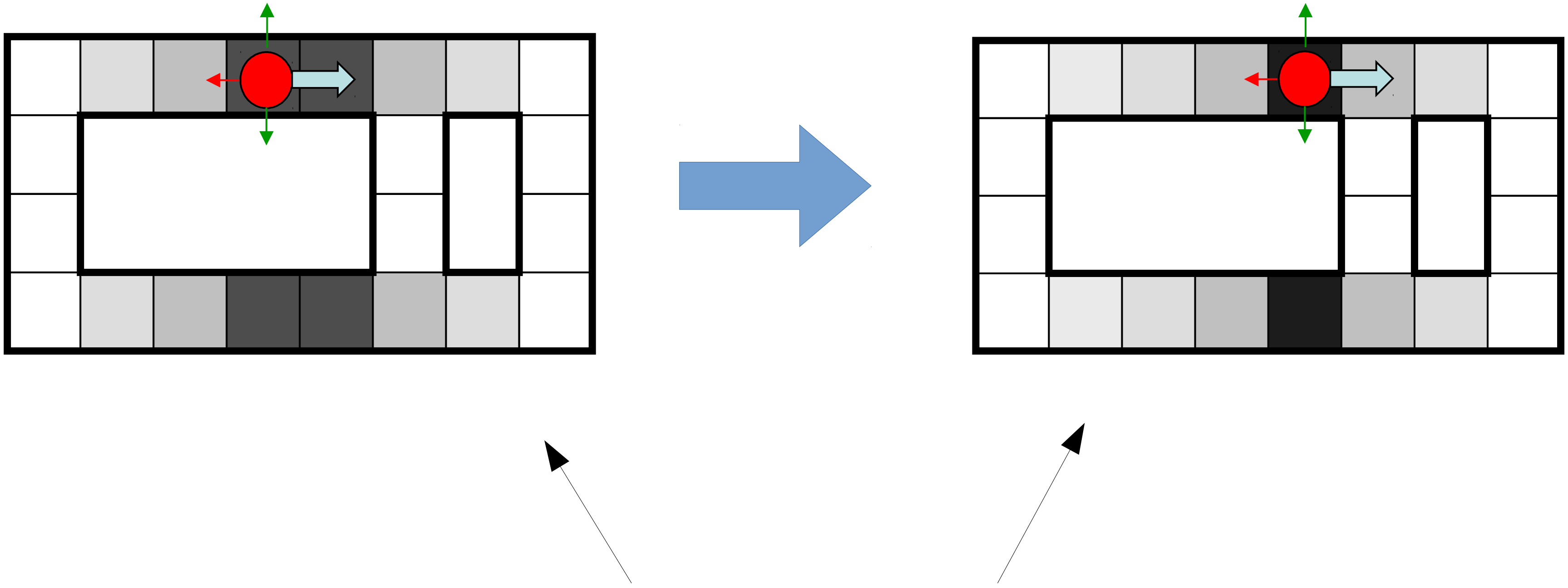


R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

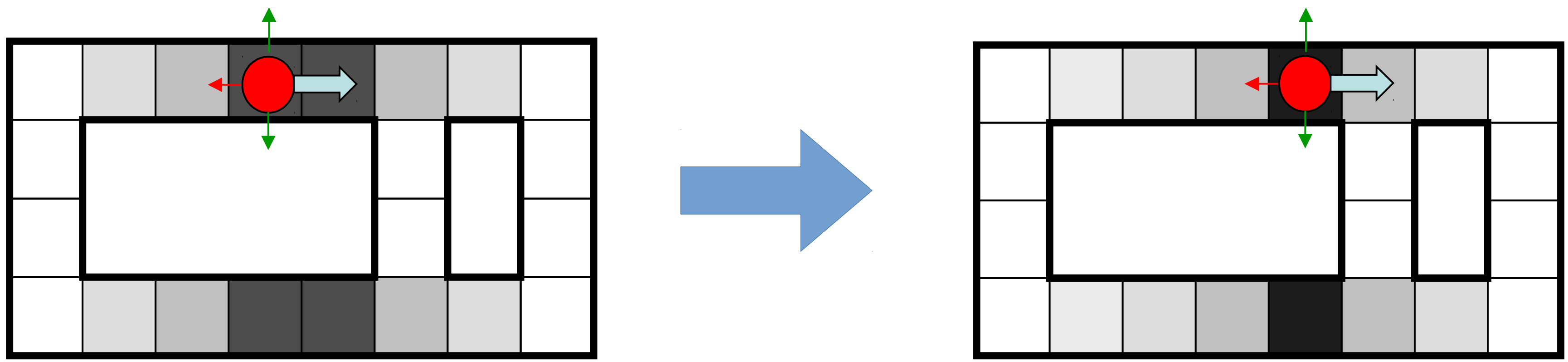
Images: Berkeley CS188 course notes (downloaded Summer 2015)

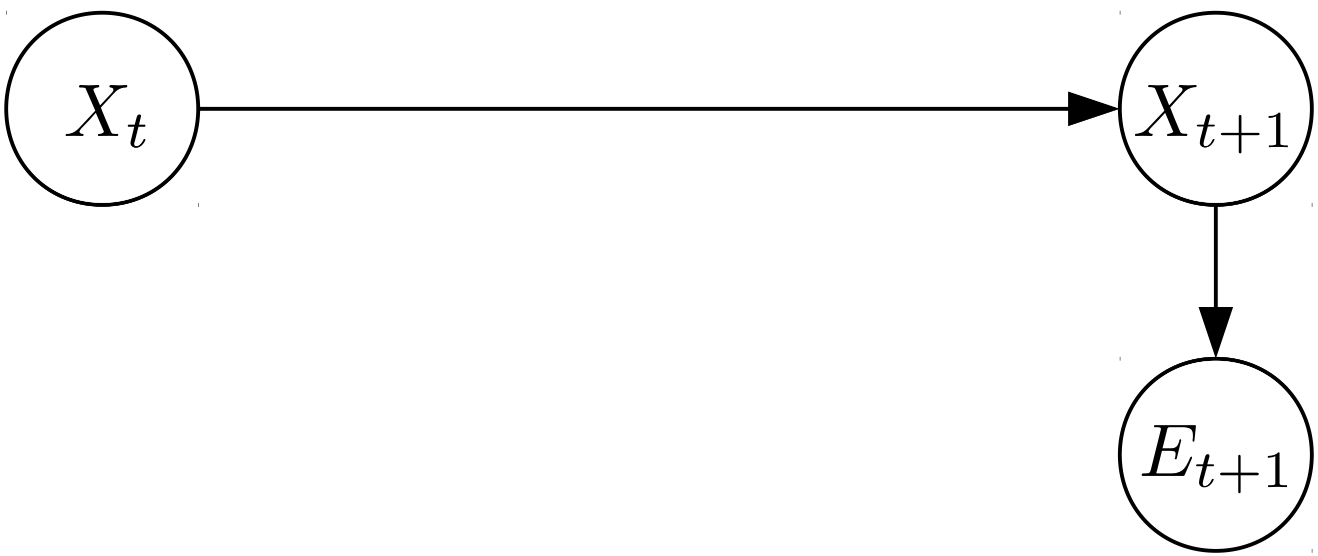
Bayes Filtering



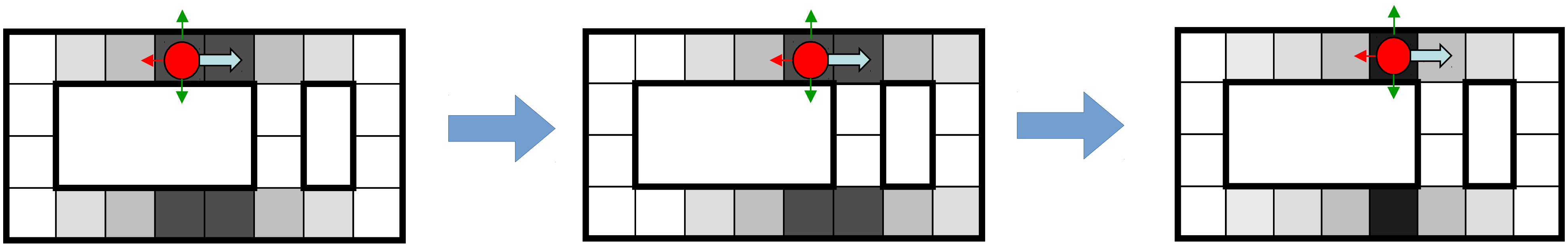
How do we go from this distribution to this distribution?

Bayes Filtering





Bayes Filtering



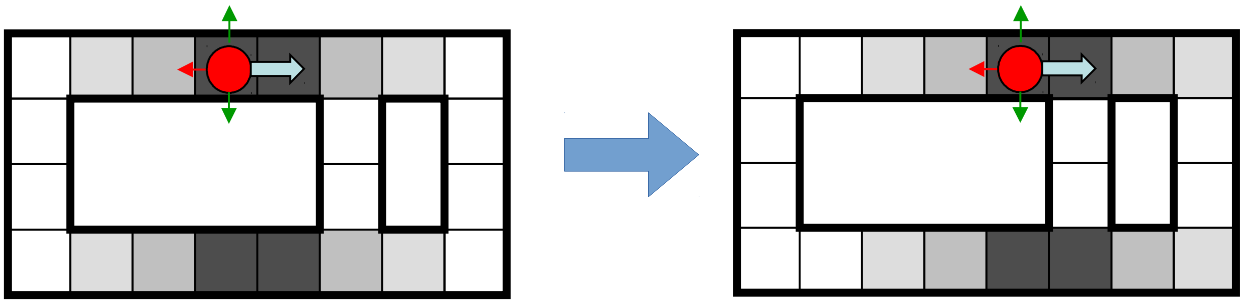
$B(X_t)$ — $B'(X_t)$ — $B(X_{t+1})$

$$P(X_t|E_{1:t}) \rightarrow P(X_{t+1}|E_{1:t}) \rightarrow P(X_{t+1}|E_{1:t+1})$$

Process update

Observation update

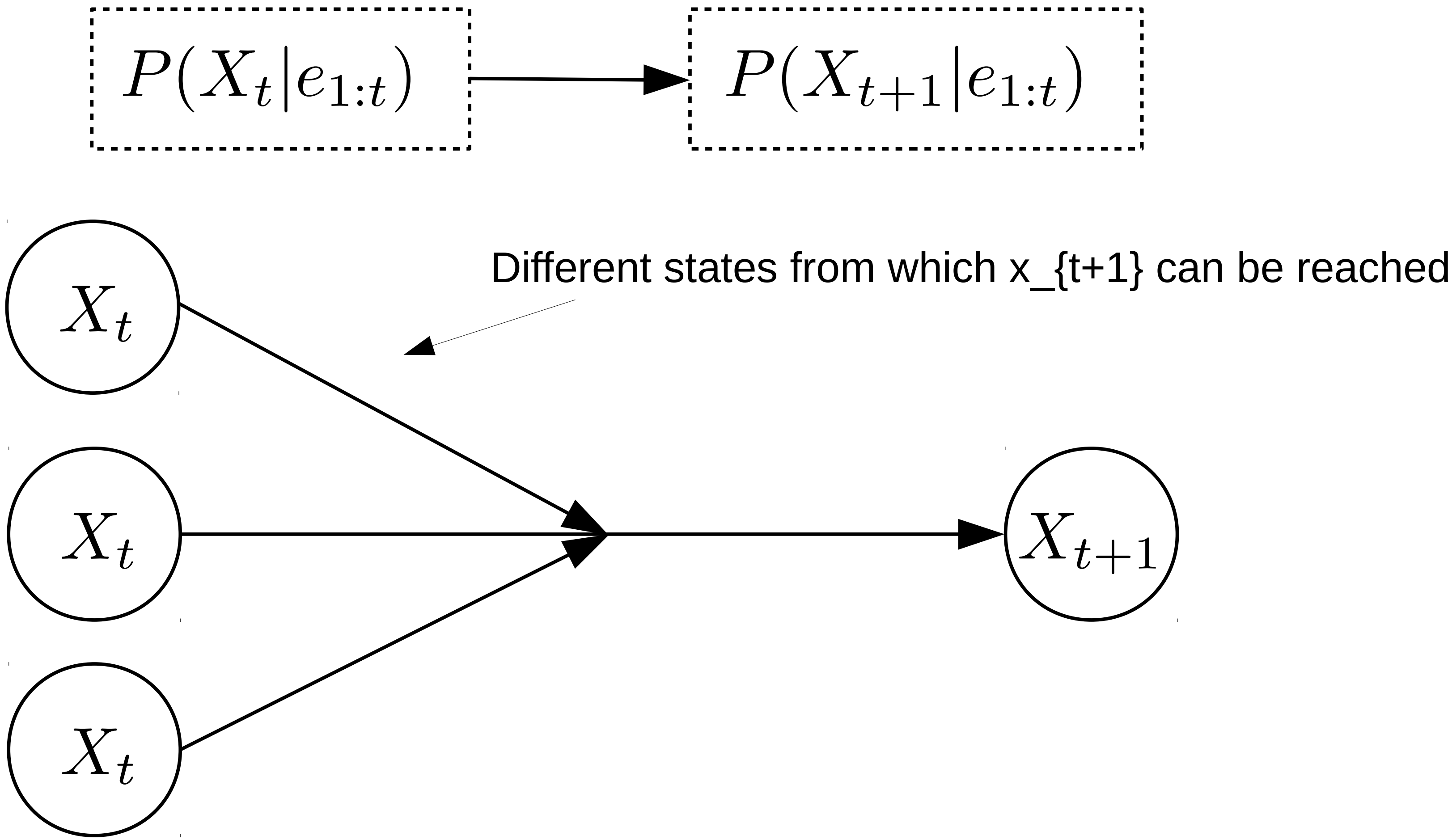
Process update



$B(X_t)$ $B'(X_t)$

$P(X_t|e_{1:t})$ → $P(X_{t+1}|e_{1:t})$

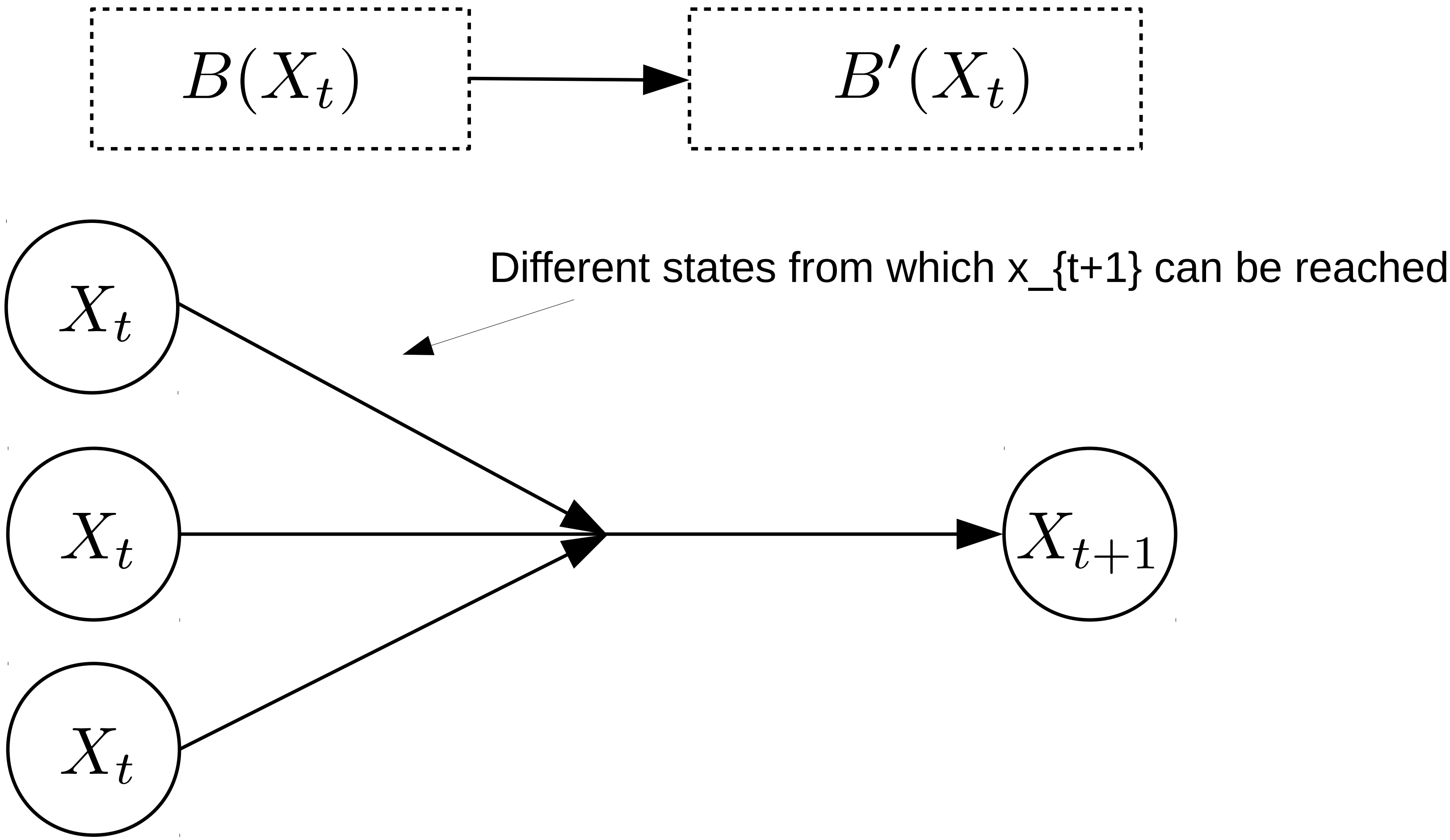
Process update



$$P(X_{t+1}|e_{1:t}) = \sum_{X_t} P(X_{t+1}|X_t, e_{1:t})P(X_t|e_{1:t})$$

Marginalize over
next states

Process update



$$B'(X_{t+1}) = \sum_{X_t} P(X_{t+1}|X_t, e_{1:t})B(X_t)$$

Marginalize over
next states

Process update

Before process update

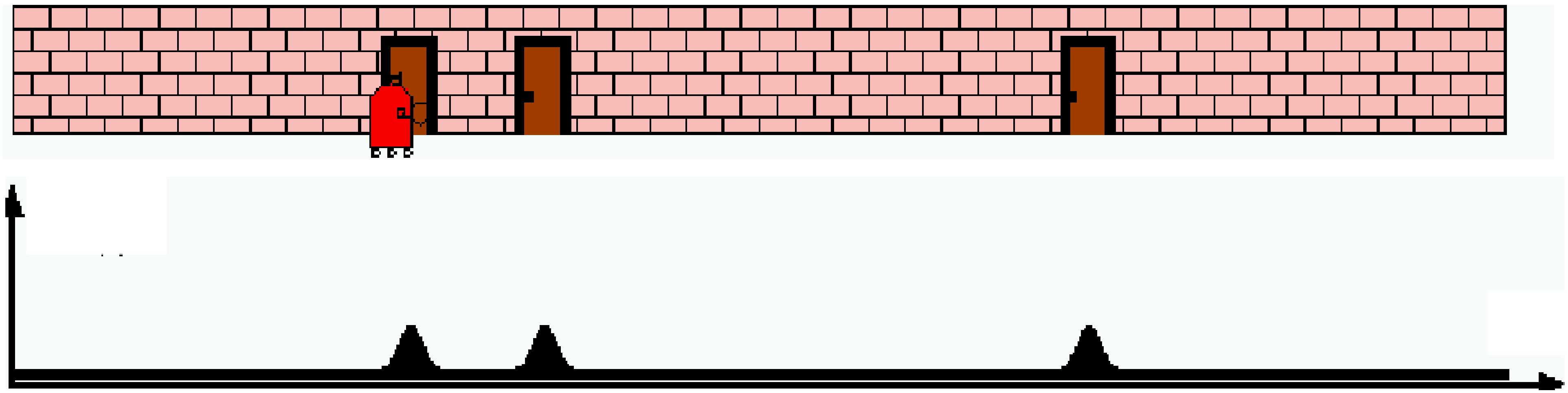
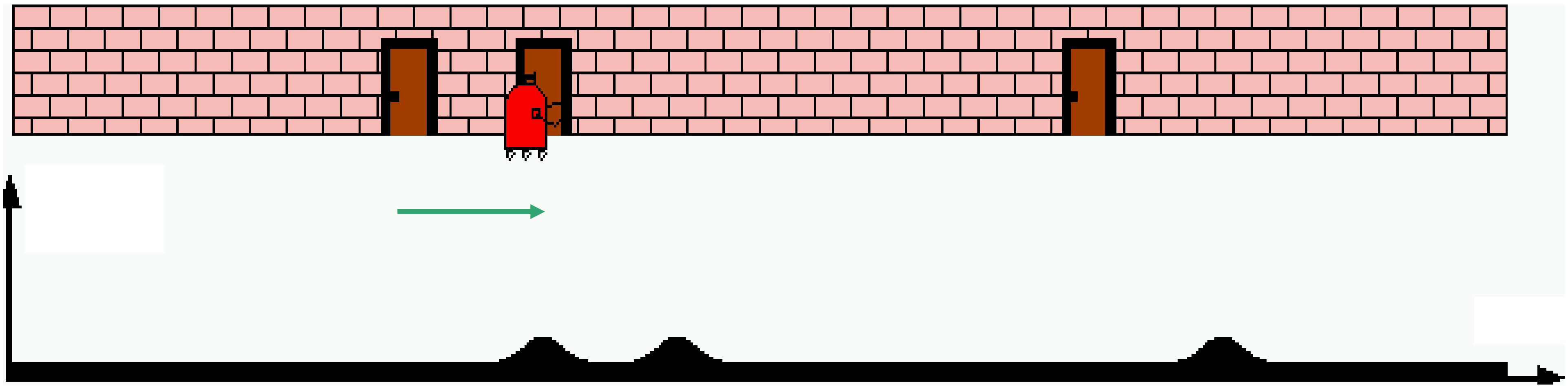


Image: Thrun, *Probabilistic Robotics*, 2006

Process update

After process update



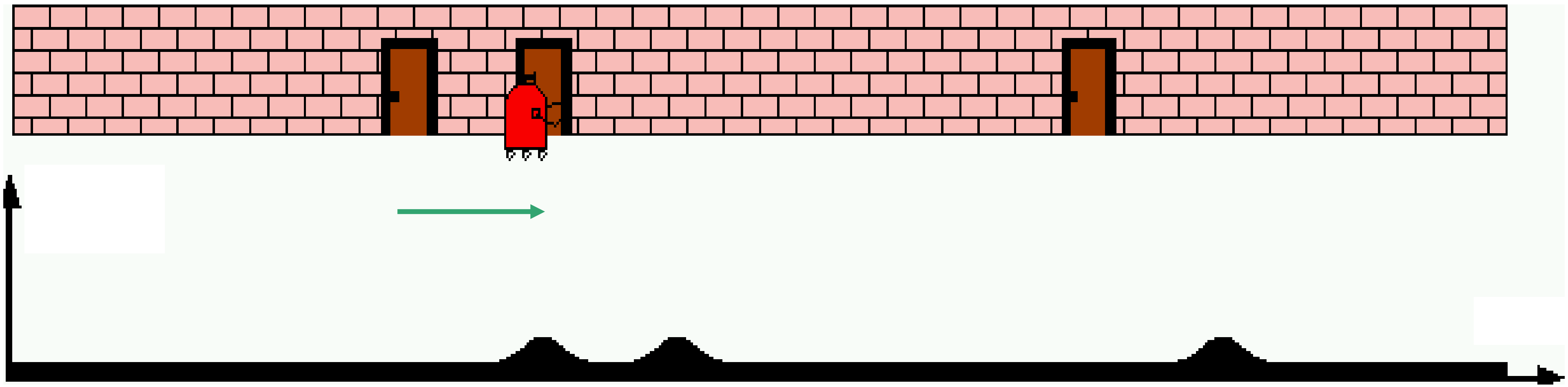
$$B'(X_{t+1}) = \sum_{X_t} P(X_{t+1}|X_t, e_{1:t})B(X_t)$$

This is a little like convolution...

Image: Thrun, *Probabilistic Robotics*, 2006

Process update

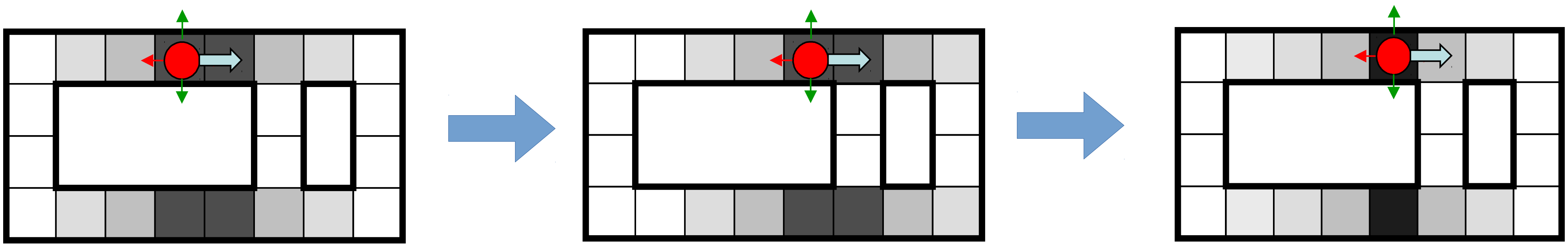
After process update



Each time you execute a process update, belief gets more disbursed
– *i.e.* Shannon entropy increases
– this makes sense: as you predict state further into the future, your uncertainty grows.

Image: Thrun, *Probabilistic Robotics*, 2006

Bayes Filtering



$B(X_t)$ $\quad$ $B'(X_t)$ $\quad$ $B(X_{t+1})$

$$P(X_t|E_{1:t}) \rightarrow P(X_{t+1}|E_{1:t}) \rightarrow P(X_{t+1}|E_{1:t+1})$$

Process update

Observation update

Observation update

$B'(X_t)$ $\qquad\qquad$ $B(X_{t+1})$

$$P(X_{t+1}|e_{1:t}) \longrightarrow P(X_{t+1}|e_{1:t+1})$$

$$P(X_{t+1}|e_{1:t+1}) = \eta P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t})$$

Observation update

$B'(X_t)$ $\qquad$ $B(X_{t+1})$

$P(X_{t+1}|e_{1:t}) \rightarrow P(X_{t+1}|e_{1:t+1})$

$$P(X_{t+1}|e_{1:t+1}) = \eta \underbrace{P(e_{t+1}|X_{t+1})} P(X_{t+1}|e_{1:t})$$

Probability of seeing observation e_{t+1} from state X_{t+1}

Observation update

$B'(X_t)$ $\qquad$ $B(X_{t+1})$

$P(X_{t+1}|e_{1:t}) \rightarrow P(X_{t+1}|e_{1:t+1})$

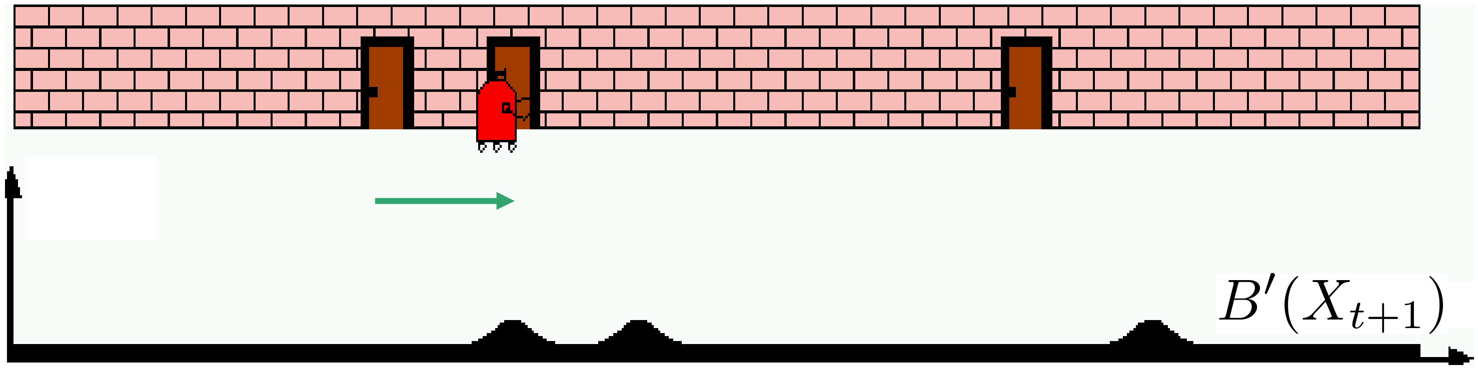
$$P(X_{t+1}|e_{1:t+1}) = \eta P(e_{t+1}|X_{t+1}) P(X_{t+1}|e_{1:t})$$

$$B(X_{t+1}) = \eta P(e_{t+1}|X_{t+1}) B'(X_{t+1})$$

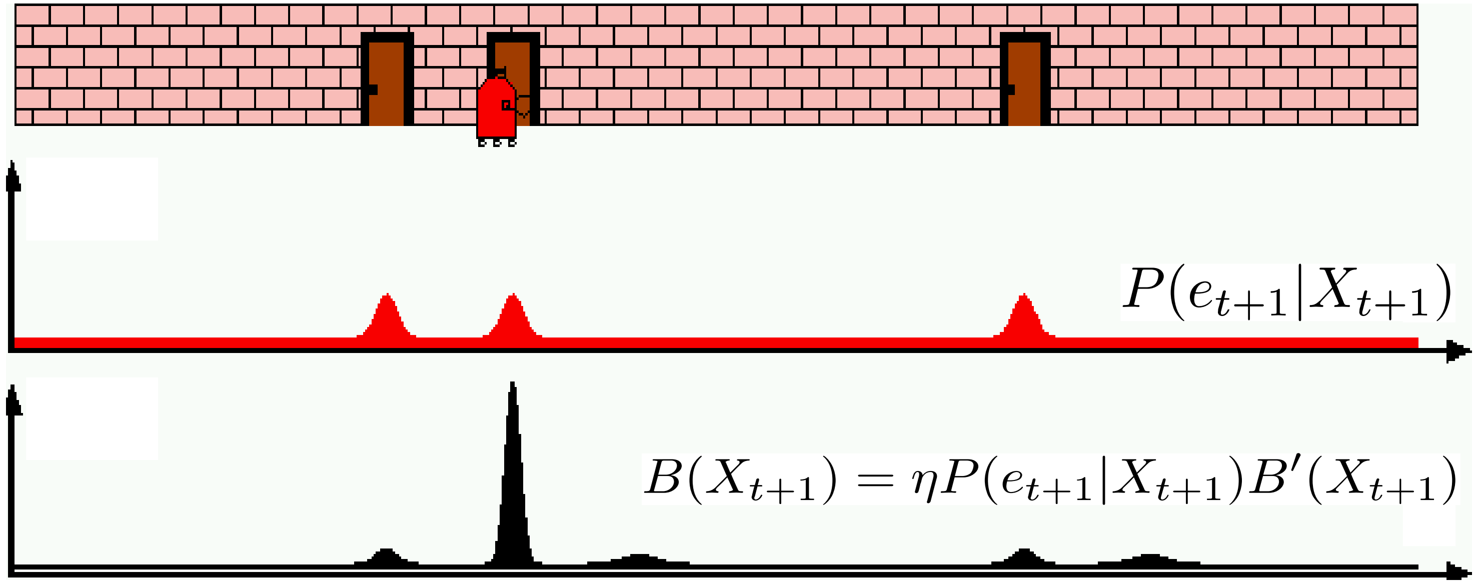
Where $\eta = \dfrac{1}{P(e_{t+1})}$ is a normalization factor

Observation update

Before observation update



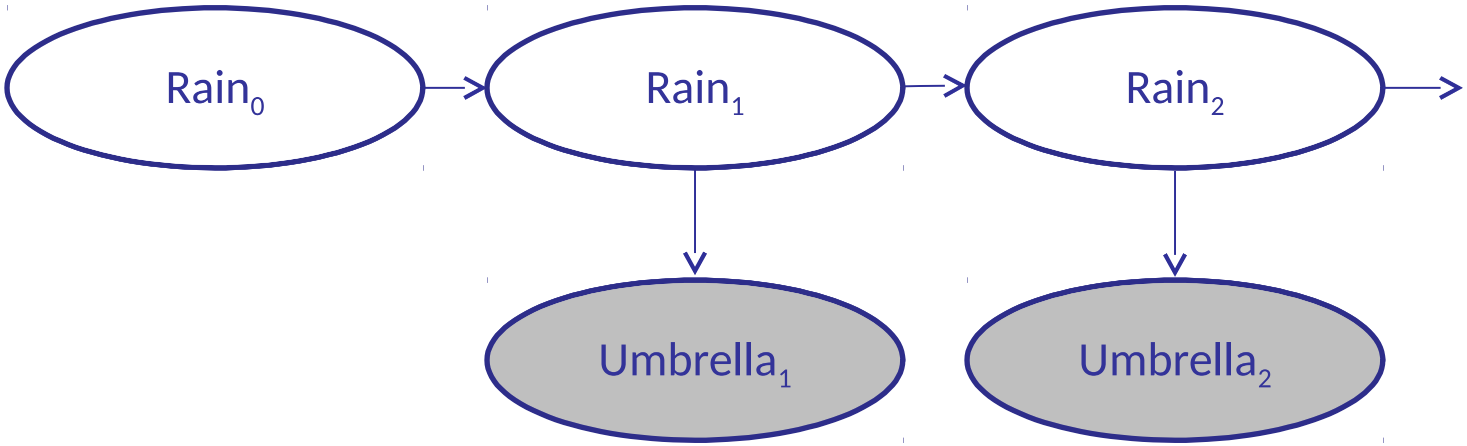
After observation update



Weather HMM example

B(+r) = 0.5
B(-r) = 0.5

R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7



R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

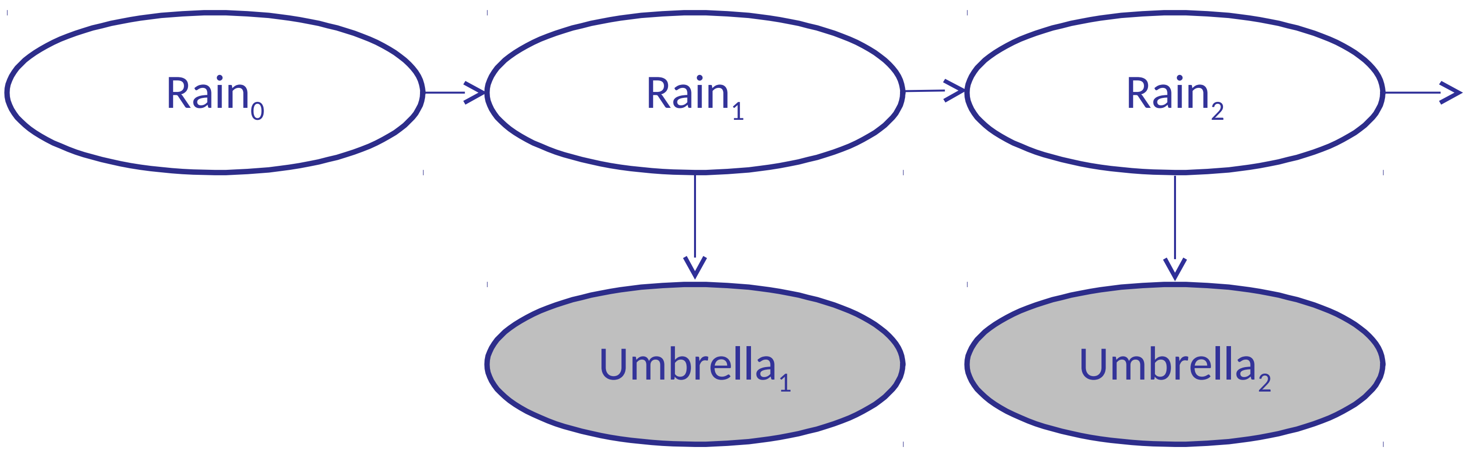
Slide: Berkeley CS188 course notes (downloaded Summer 2015)

Weather HMM example

$B'(+r) = 0.5$
$B'(-r) = 0.5$

R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

$B(+r) = 0.5$
$B(-r) = 0.5$



R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Slide: Berkeley CS188 course notes (downloaded Summer 2015)

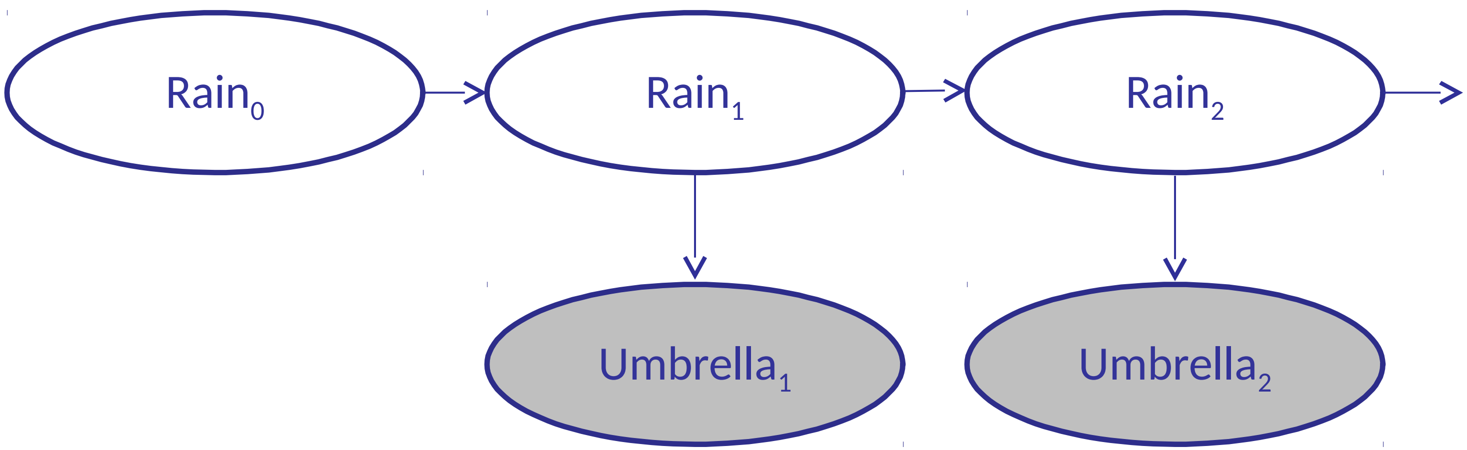
Weather HMM example

B(+r) = 0.5
B(-r) = 0.5

B'(+r) = 0.5
B'(-r) = 0.5

B(+r) = 0.818
B(-r) = 0.182

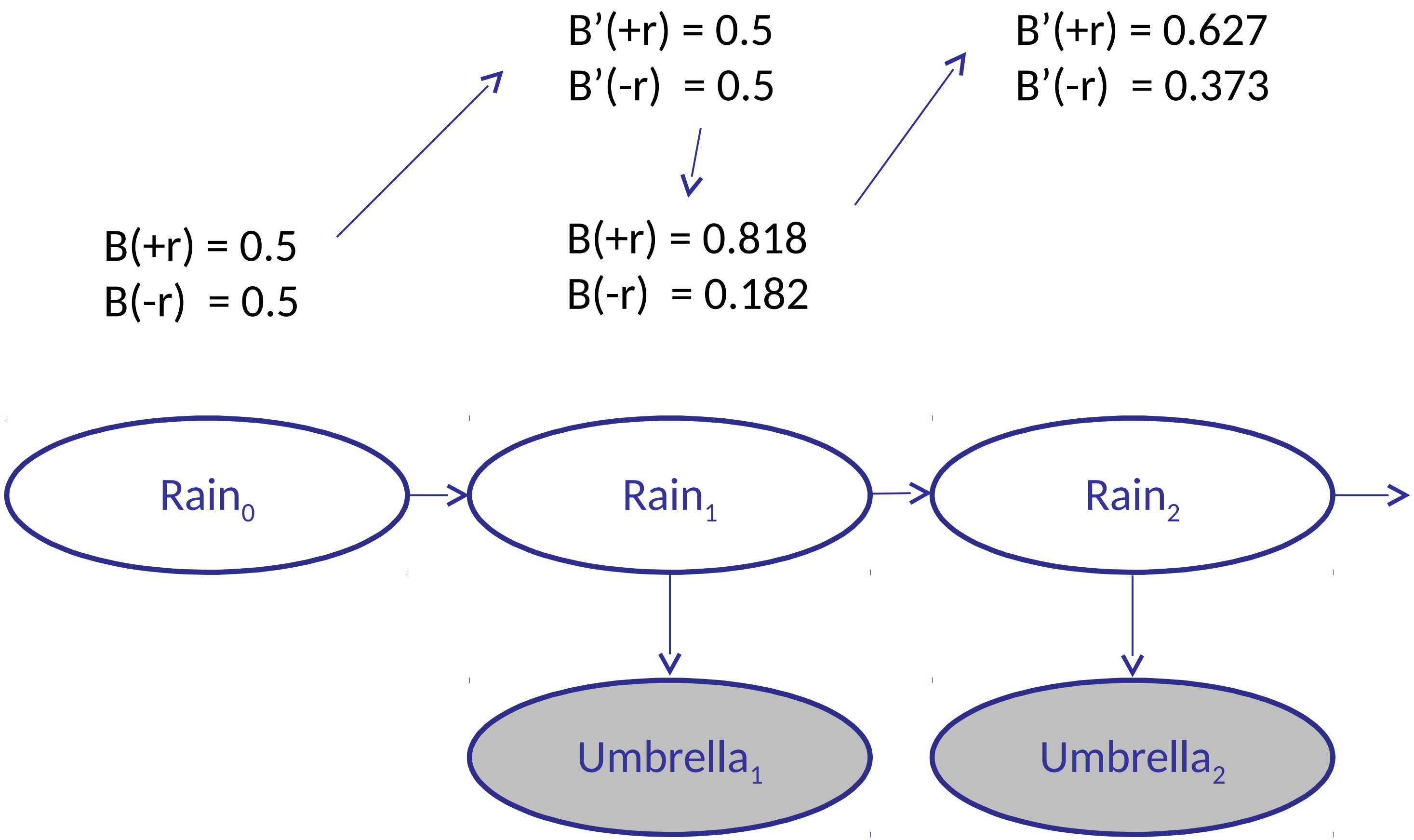
R_t	R_{t+1}	$P(R_{t+1}\|R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7



R_t	U_t	$P(U_t\|R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Slide: Berkeley CS188 course notes (downloaded Summer 2015)

Weather HMM example

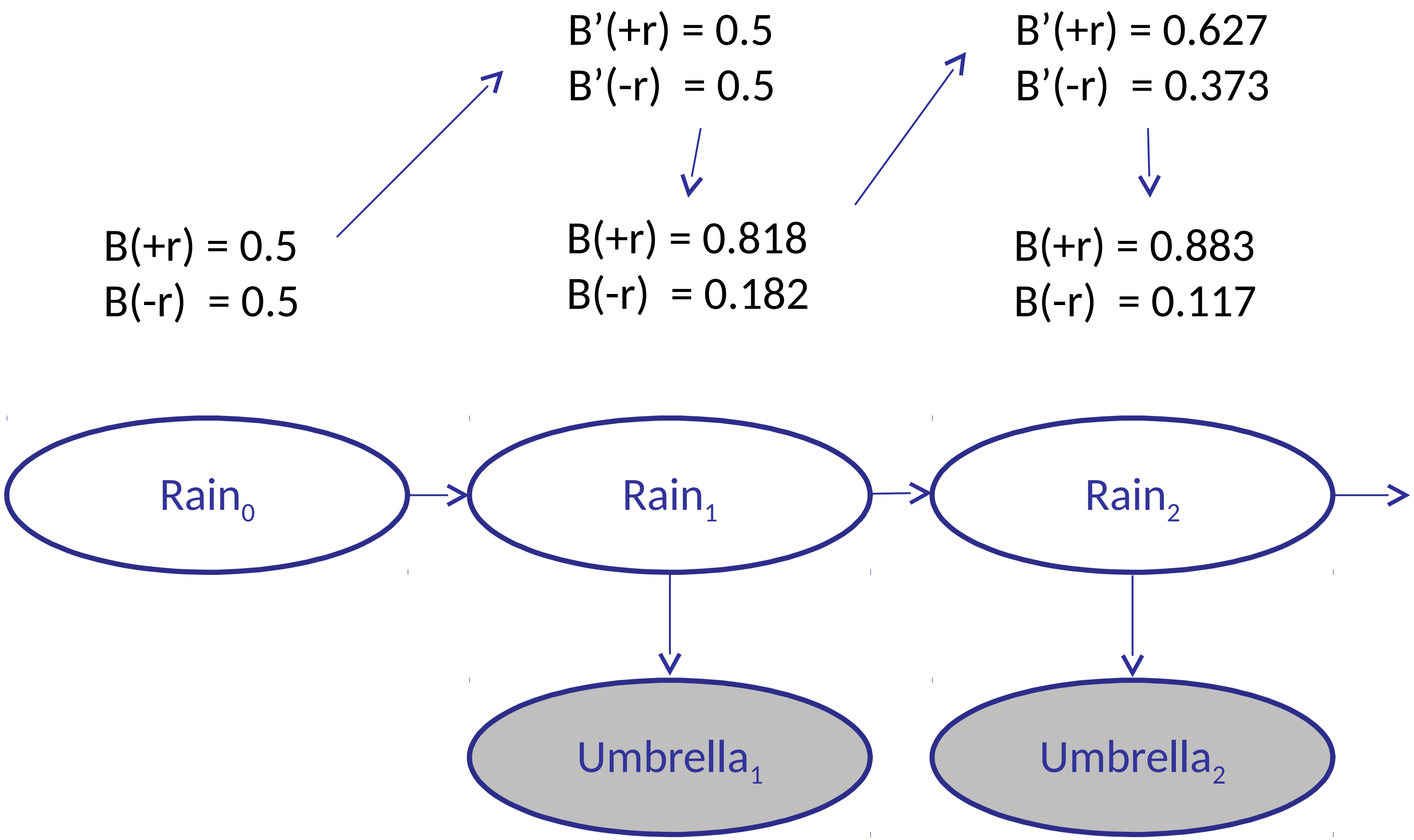


R_t	R_{t+1}	$P(R_{t+1}\|R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t\|R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Slide: Berkeley CS188 course notes (downloaded Summer 2015)

Weather HMM example

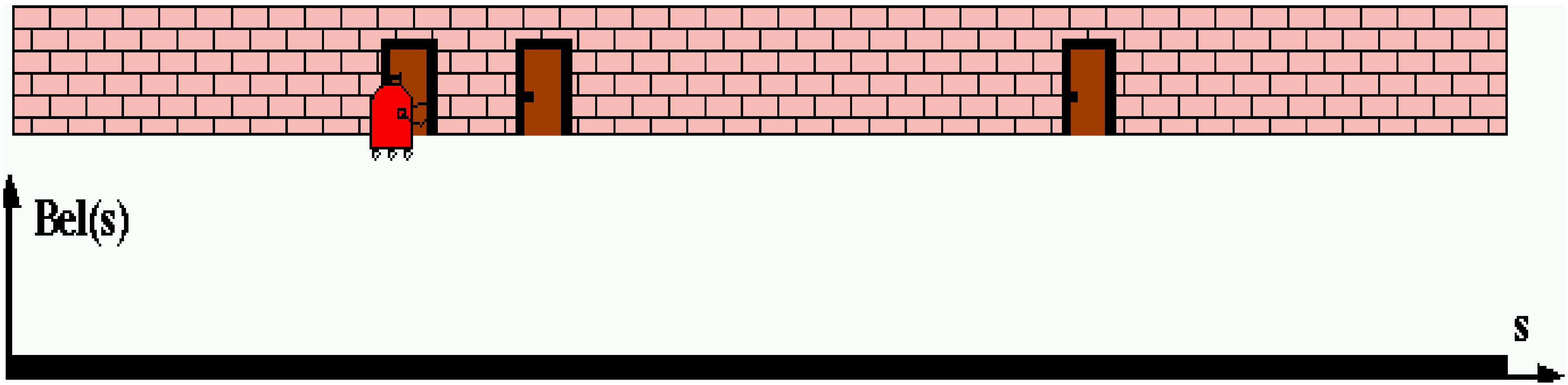


R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

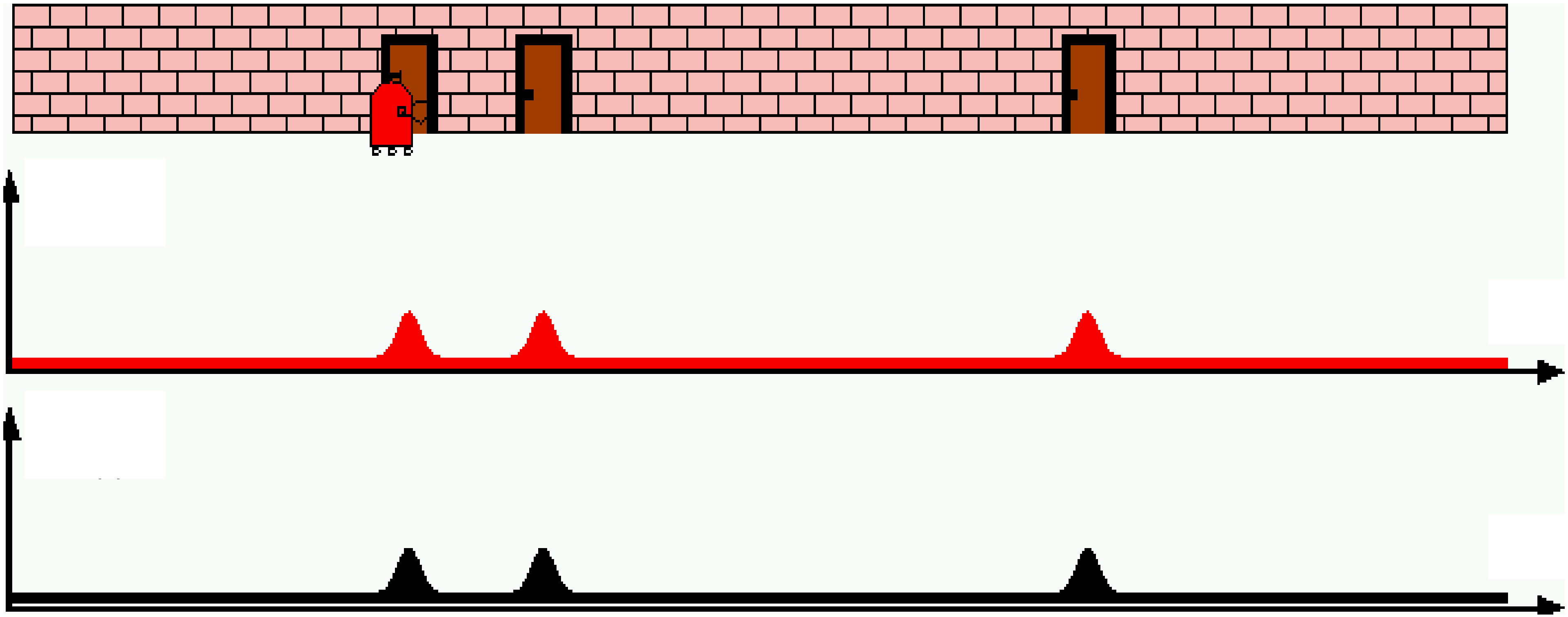
Slide: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example



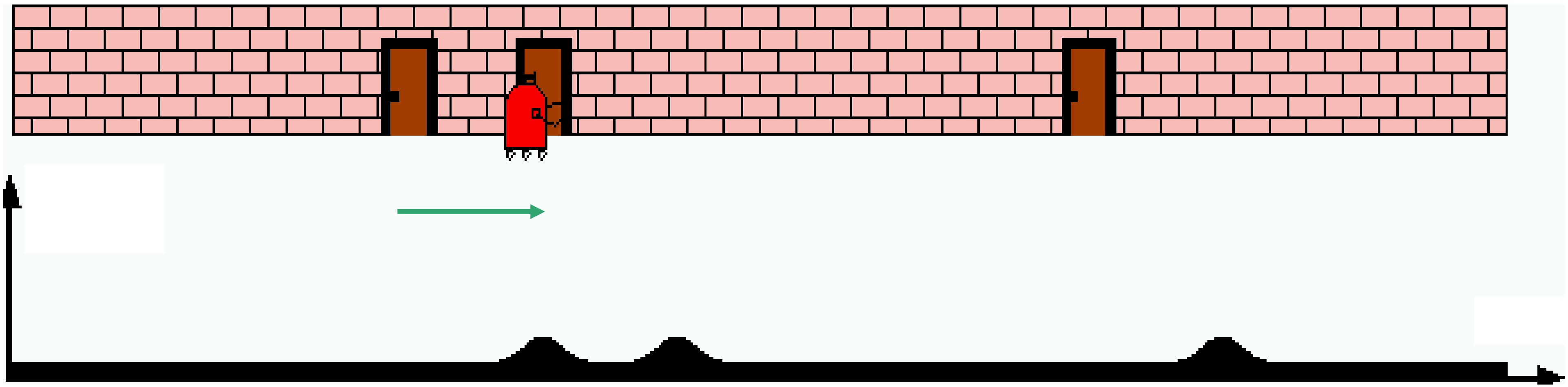
Slide: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example

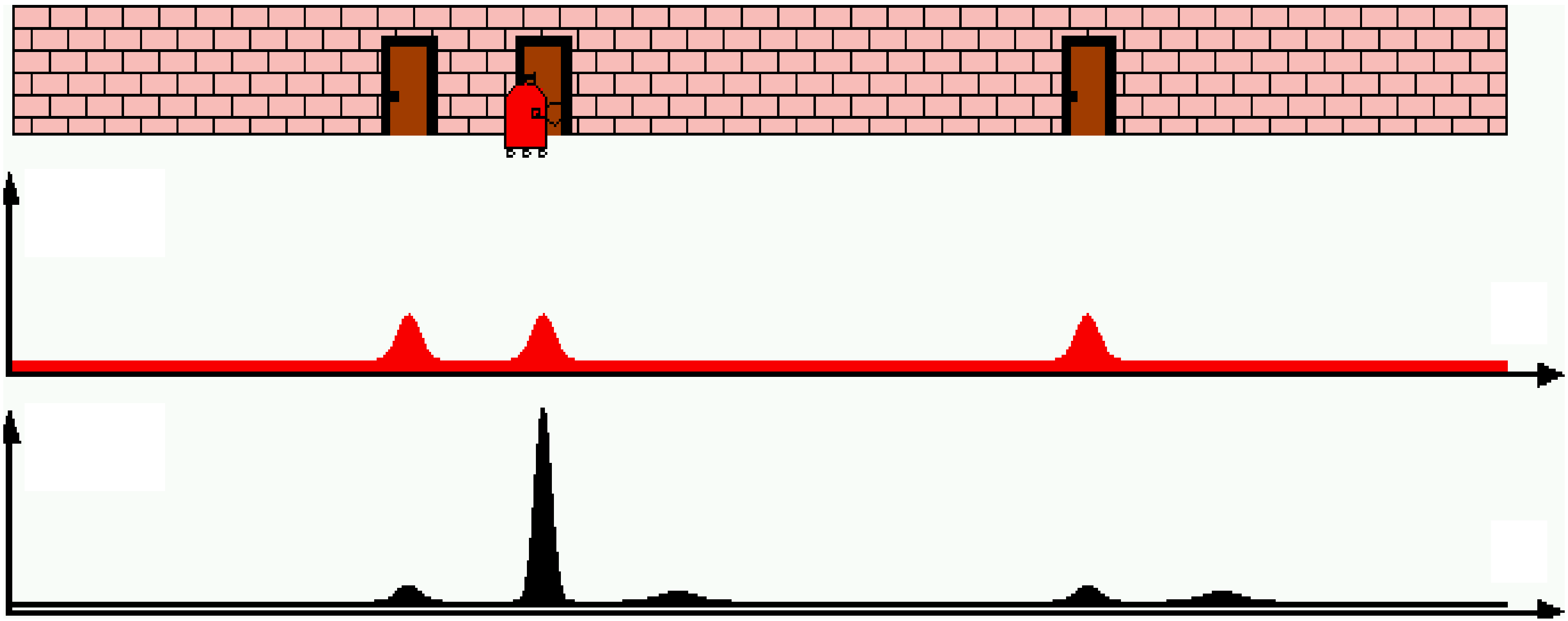


Slide: Berkeley CS188 course notes (downloaded Summer 2015)

Robot localization example

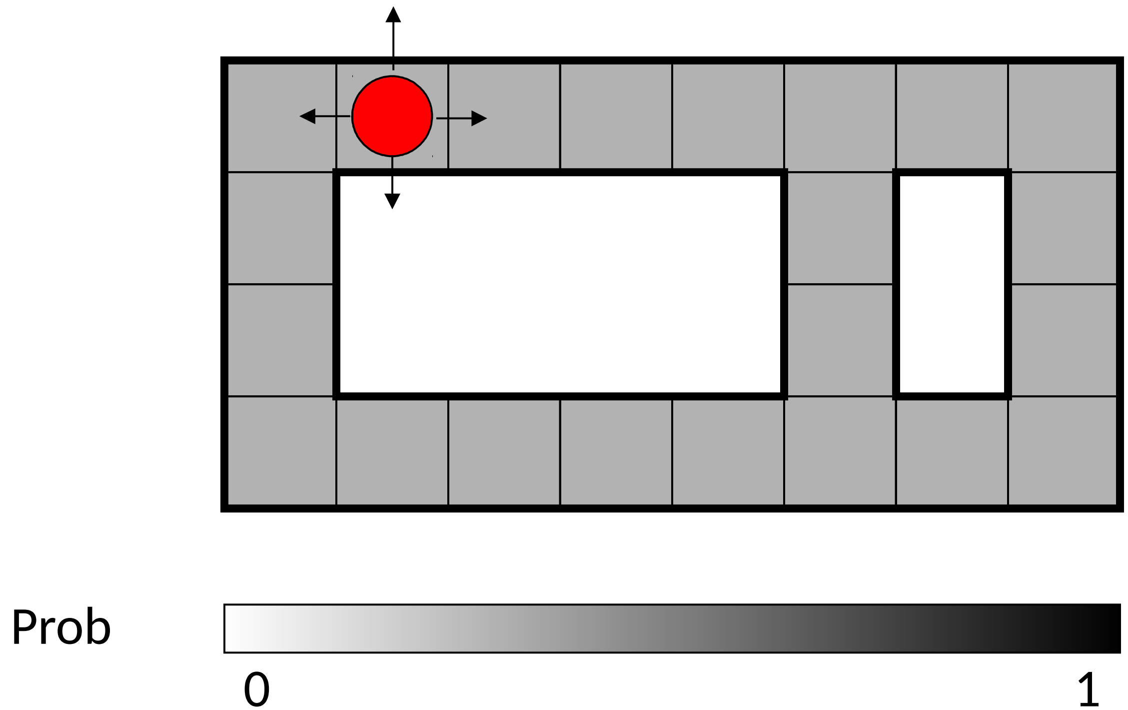


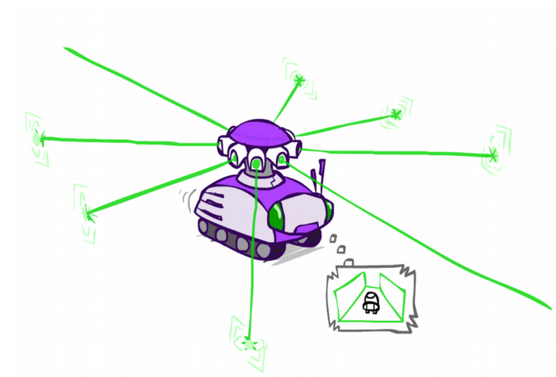
Robot localization example



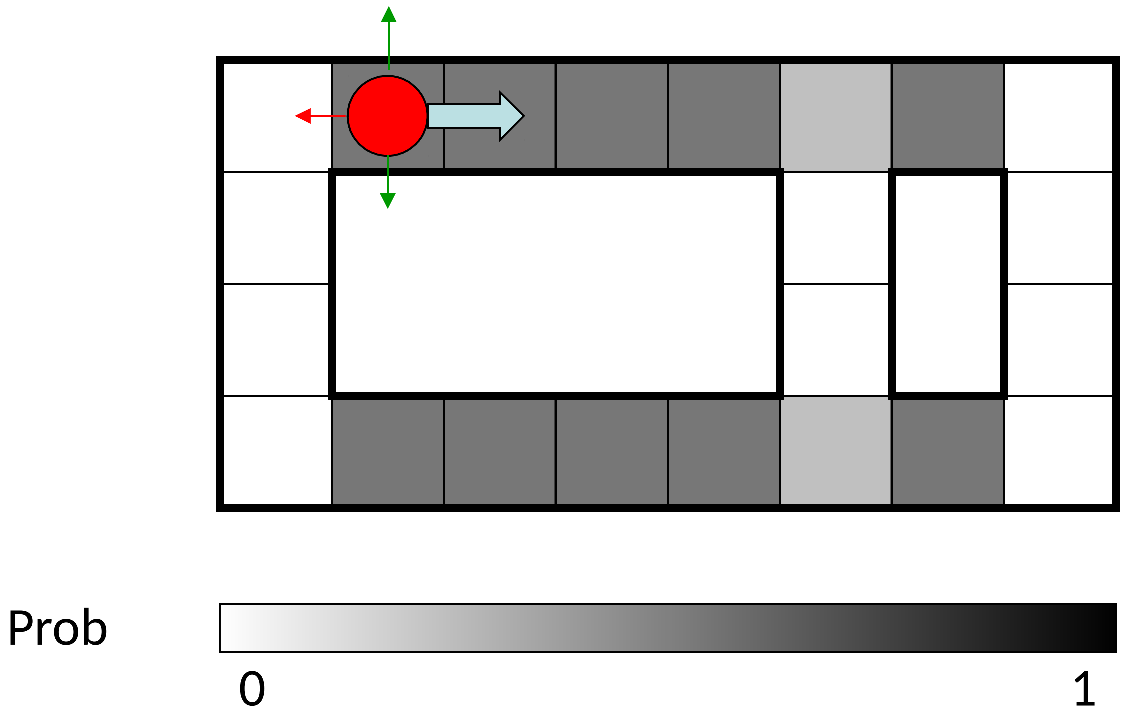
Slide: Berkeley CS188 course notes (downloaded Summer 2015)

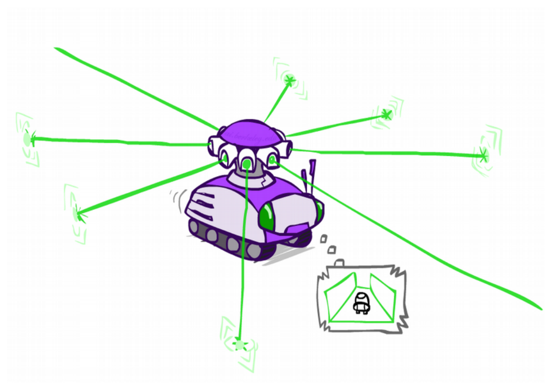
Robot localization example



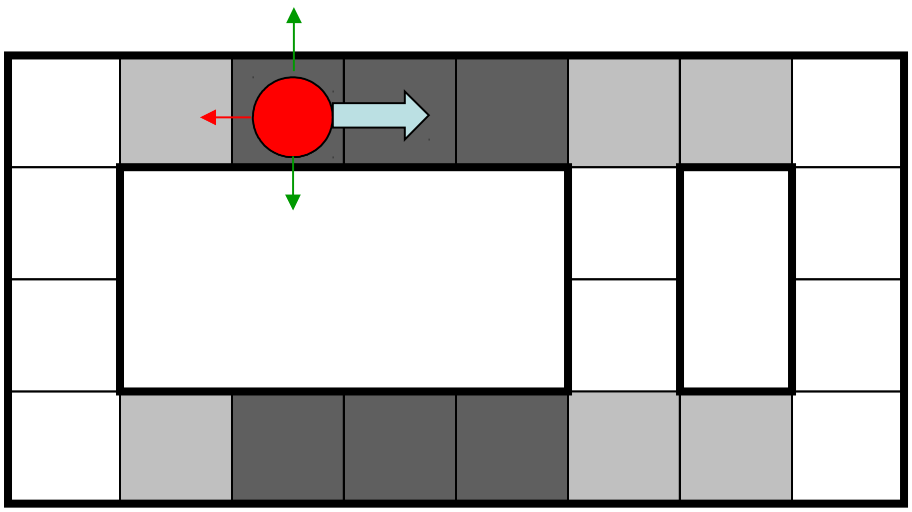


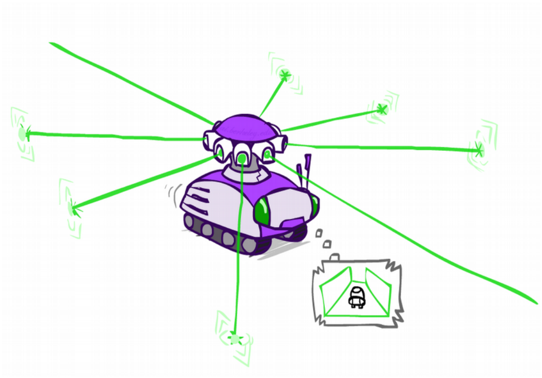
Robot localization example





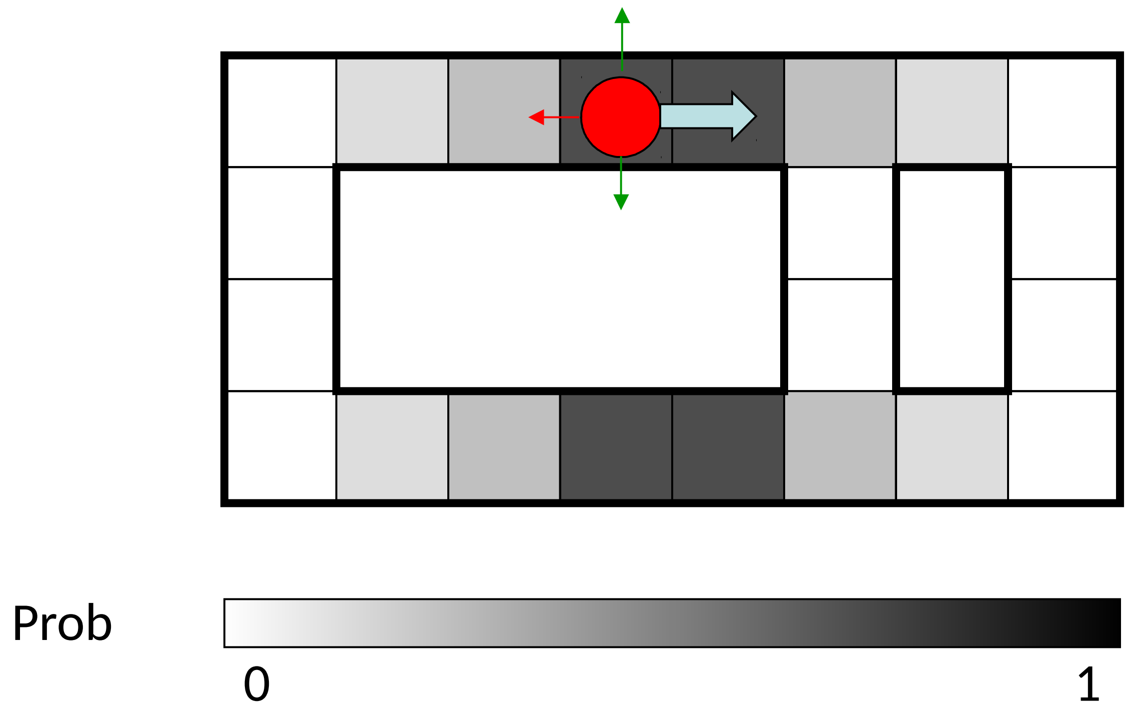
Robot localization example

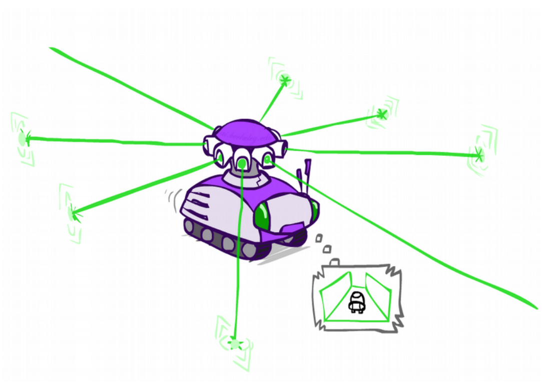




Slide: Berkeley CS188 course notes (downloaded Summer 2015)

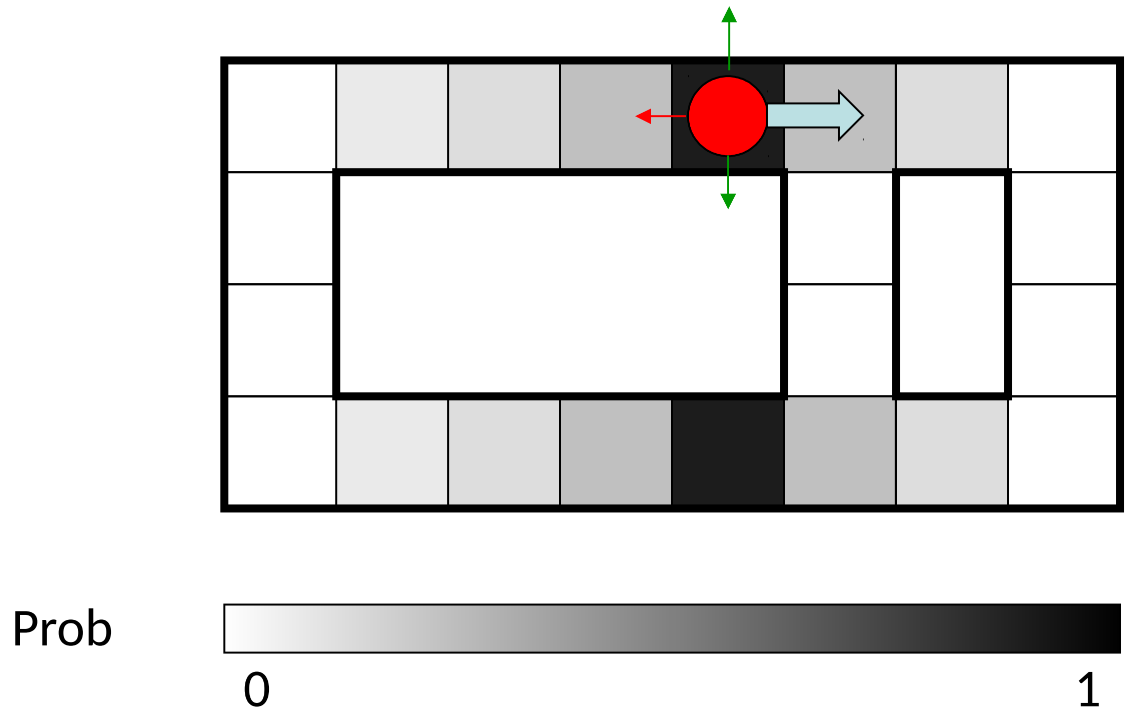
Robot localization example

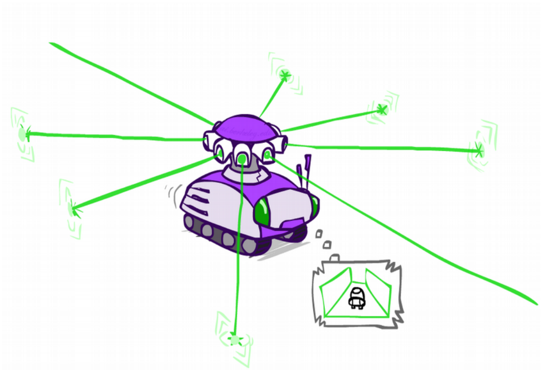




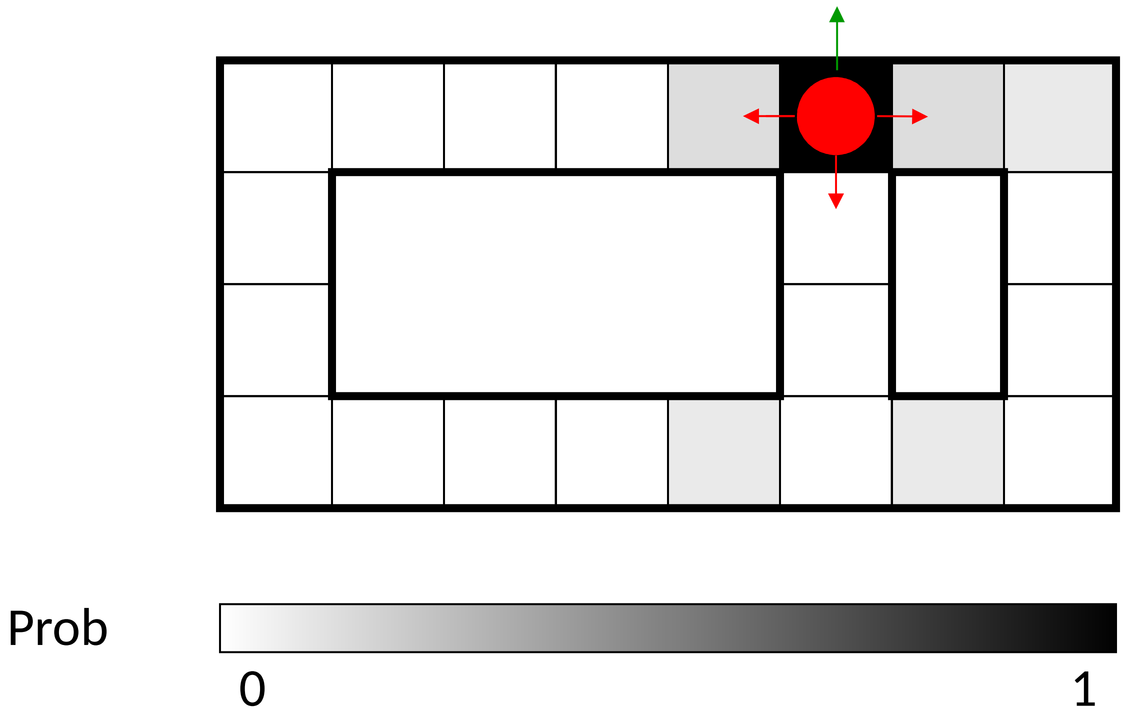
Slide: Berkeley CS188 course notes (downloaded Summer 2015)

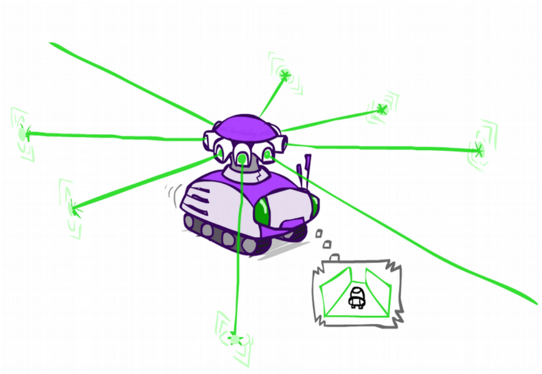
Robot localization example





Robot localization example



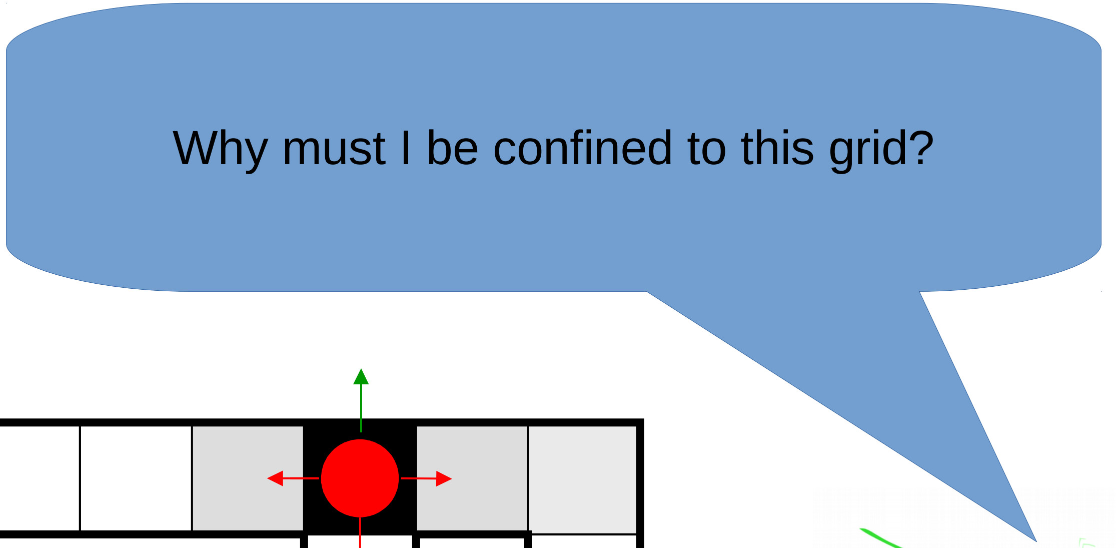


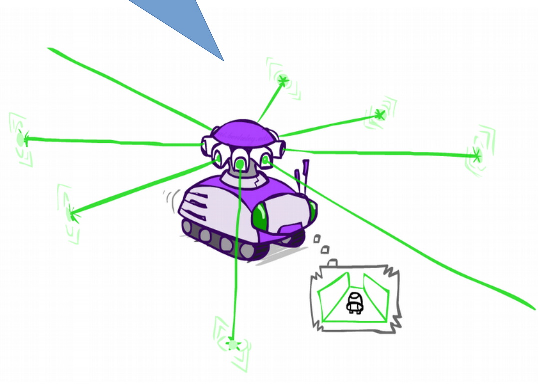
Applications of HMMs

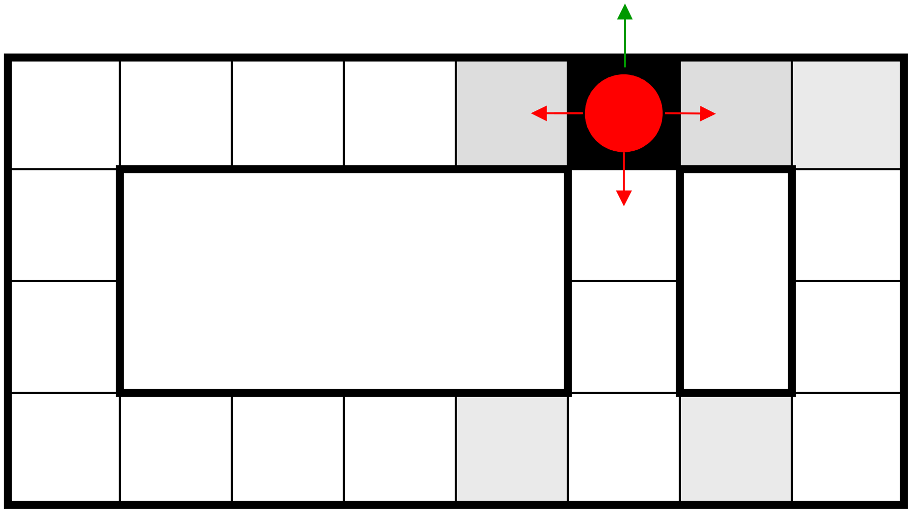
- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)
- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are translation options
- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)

Slide: Berkeley CS188 course notes (downloaded Summer 2015)

Particle Filter





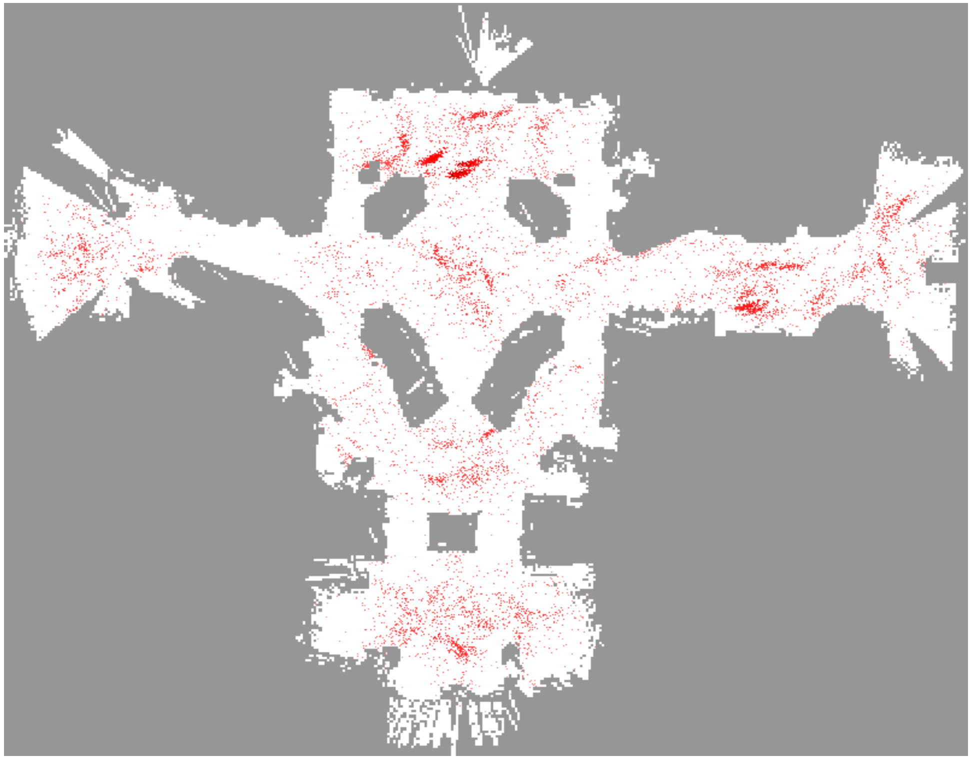


Standard Bayes filtering requires discretizing state space into grid cells

Can do Bayes filtering w/o discretizing?
– yes: particle filtering or Kalman filtering

Image: Berkeley CS188 course notes (downloaded Summer 2015)

Particle Filter



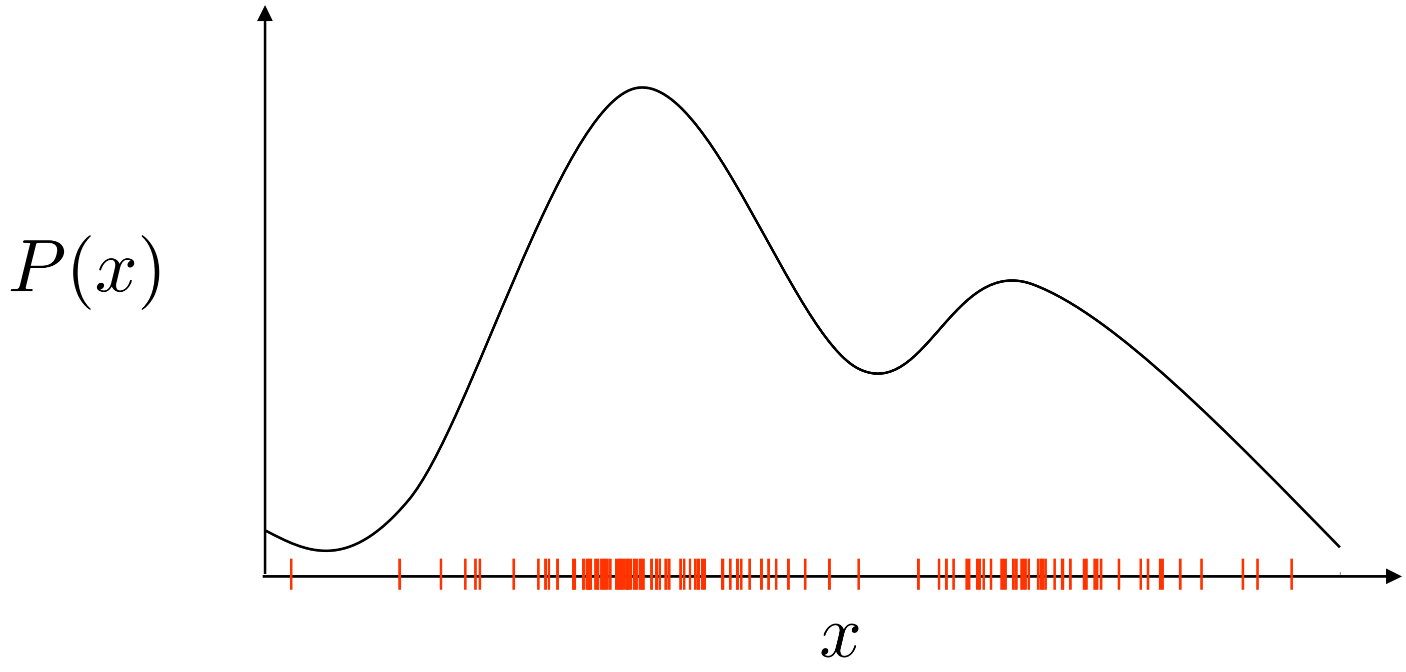
Sequential Bayes Filtering is great, but it's not great for continuous state spaces.
– you need to discretize the state space (e.g. a grid) in order to use Bayes filtering
  – but, doing filtering on a grid is not efficient...

Therefore:
  – particle filters
  – Kalman filters

} Two different ways of filtering in continuous state spaces

Image: Thrun CS223b Course Notes (downloaded Summer 2015)

Particle Filter



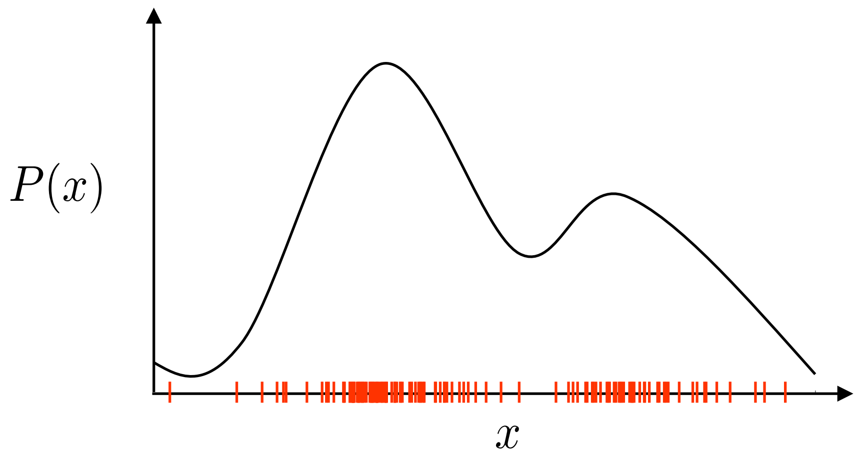
Key idea: represent a probability distribution as a finite set of points

– density of points encodes probability mass.

– particle filtering is an adaptation of Bayes filtering to this particle representation

Image: Thrun CS223b Course Notes (downloaded Summer 2015)

Monte Carlo Sampling



Suppose you are given an unknown probability distribution, $P(x)$

Suppose you can't evaluate the distribution analytically, <u>but you can draw samples from it</u>

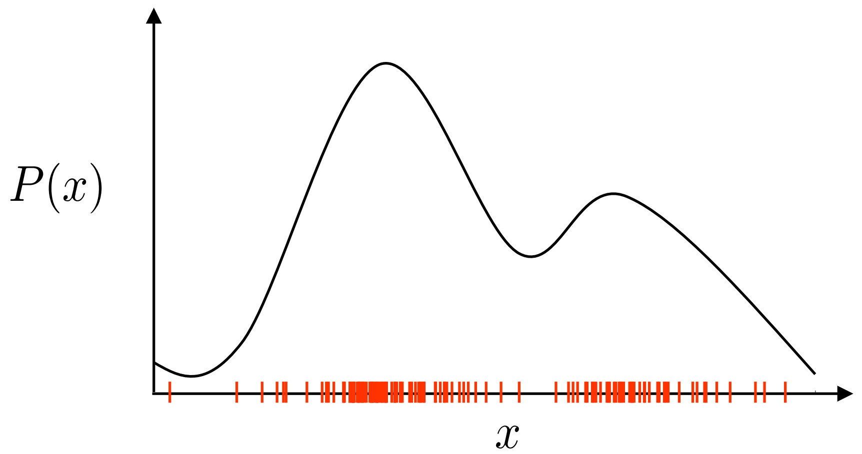
What can you do with this information?

$$E_{x \sim P(x)}(h(x)) = \int_x h(x) P(x)$$

$$\approx \frac{1}{k} \sum_{i=1}^{k} h(x^i)$$ where x^i are samples drawn from $P(x)$

Image: Thrun CS223b Course Notes (downloaded Summer 2015)

Importance Sampling



Suppose you are given an unknown probability distribution, $P(x)$

Suppose you can't evaluate the distribution analytically, ~~but you can draw samples from it~~

What can you do with this information?

Suppose you can't even sample from it?

Suppose that all you can do is evaluate the function at a given point?

Image: Thrun CS223b Course Notes (downloaded Summer 2015)

Importance Sampling

Question: how estimate expected values if cannot draw samples from f(x)
– suppose all we can do is evaluate f(x) at a given point...

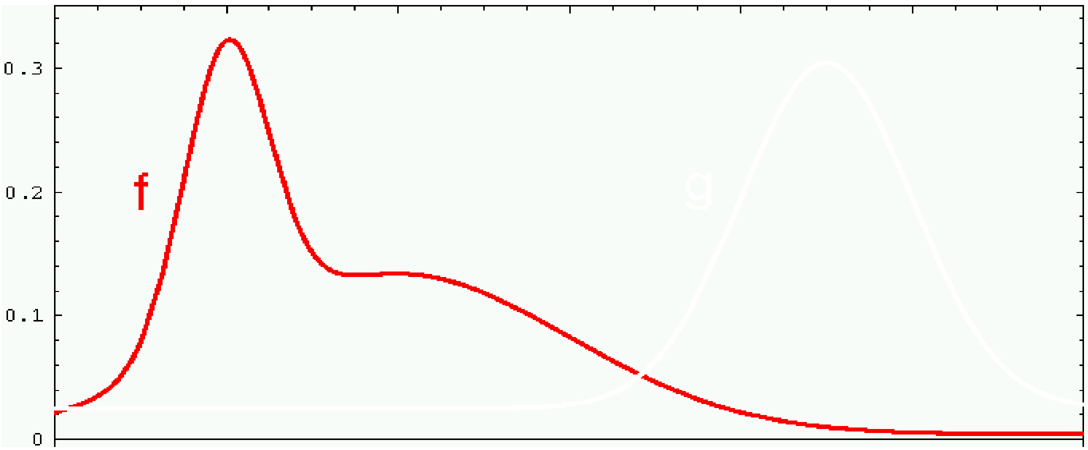
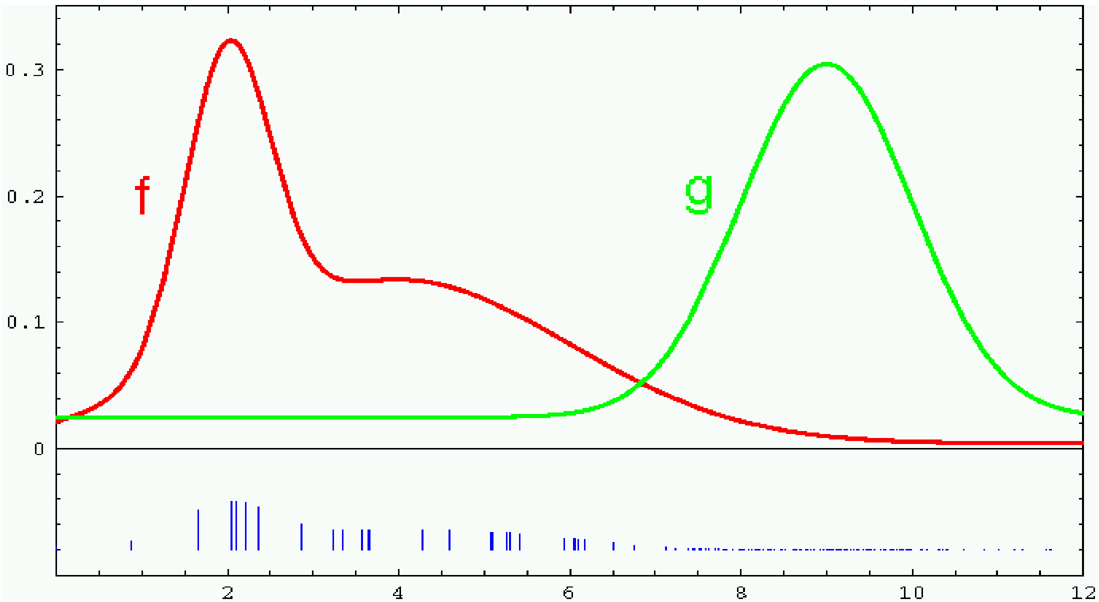


Image: Thrun CS223b Course Notes (downloaded Summer 2015)

Importance Sampling

Question: how estimate expected values if cannot draw samples from f(x)
– suppose all we can do is evaluate f(x) at a given point...

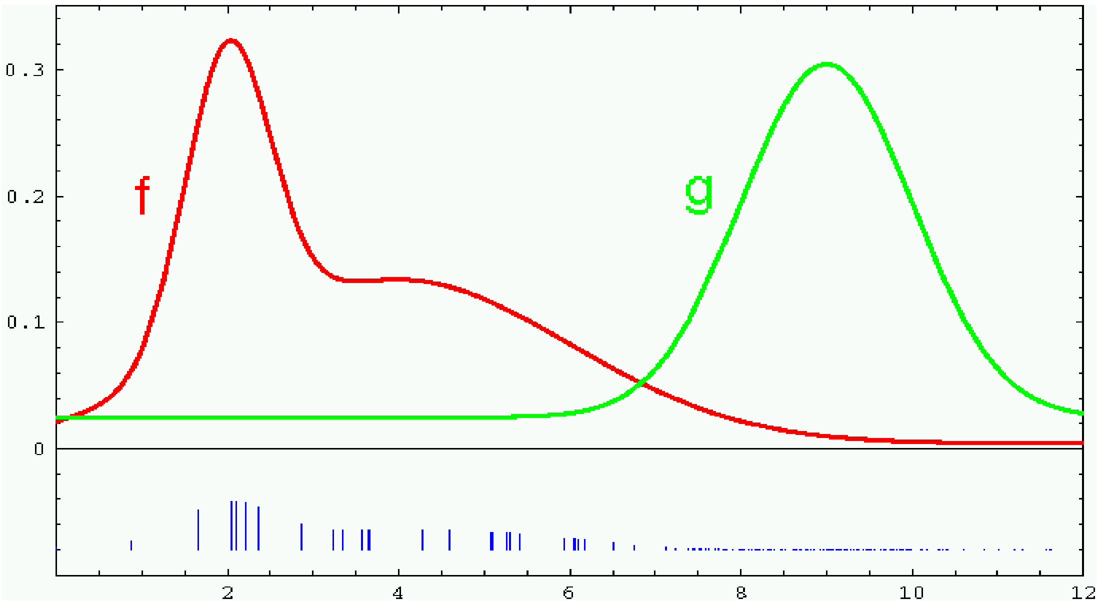


Answer: draw samples from a different distribution and weight them

Image: Thrun CS223b Course Notes (downloaded Summer 2015)

Importance Sampling

Question: how estimate expected values if cannot draw samples from f(x)
– suppose all we can do is evaluate f(x) at a given point...



Answer: draw samples from a different distribution and weight them

$$E_{x\sim f(x)}(h(x)) = \int_x h(x)\frac{f(x)}{g(x)}g(x)$$

$$\approx \frac{1}{k}\sum_{i=1}^{k} h(x^i)w_i$$

where x^i are samples drawn from $g(x)$ ← Proposal distribution

and $w_i = f(x^i)/g(x^i)$

Image: Thrun CS223b Course Notes (downloaded Summer 2015)

Particle Filter

Prior distribution

$$x_t^1, \ldots, x_t^n \qquad w_t^1, \ldots, w_t^n = 1$$

$B(X_t)$ $\quad P(X_t | E_{1:t})$

↓

$B'(X_t)$ $\quad P(X_{t+1} | E_{1:t})$

↓

$B(X_{t+1})$ $\quad P(X_{t+1} | E_{1:t+1})$

Particle Filter

$B(X_t)$ $P(X_t|E_{1:t})$

↓

$B'(X_t)$ $P(X_{t+1}|E_{1:t})$

↓

$B(X_{t+1})$ $P(X_{t+1}|E_{1:t+1})$

<u>Prior distribution</u>

$$x_t^1, \ldots, x_t^n \qquad w_t^1, \ldots, w_t^n = 1$$

<u>Process update</u>

$$\bar{x}_{t+1}^i \sim P(X_{t+1}|x_t^i, e_{1:t})$$

Particle Filter

$B(X_t)$ $\quad P(X_t|E_{1:t})$

↓

$B'(X_t)$ $\quad P(X_{t+1}|E_{1:t})$

↓

$B(X_{t+1})$ $\quad P(X_{t+1}|E_{1:t+1})$

<u>Prior distribution</u>

$$x_t^1, \ldots, x_t^n \qquad w_t^1, \ldots, w_t^n = 1$$

<u>Process update</u>

$$\bar{x}_{t+1}^i \sim P(X_{t+1}|x_t^i, e_{1:t})$$

<u>Observation update</u>

$$w_{t+1}^i = P(e_{t+1}|\bar{x}_{t+1}^i)w_t^i$$

Particle Filter

$B(X_t)$ $P(X_t|E_{1:t})$

↓

$B'(X_t)$ $P(X_{t+1}|E_{1:t})$

↓

$B(X_{t+1})$ $P(X_{t+1}|E_{1:t+1})$

Do this *n* times →

Prior distribution

$$x_t^1, \ldots, x_t^n \qquad w_t^1, \ldots, w_t^n = 1$$

Process update

$$\bar{x}_{t+1}^i \sim P(X_{t+1}|x_t^i, e_{1:t})$$

Observation update

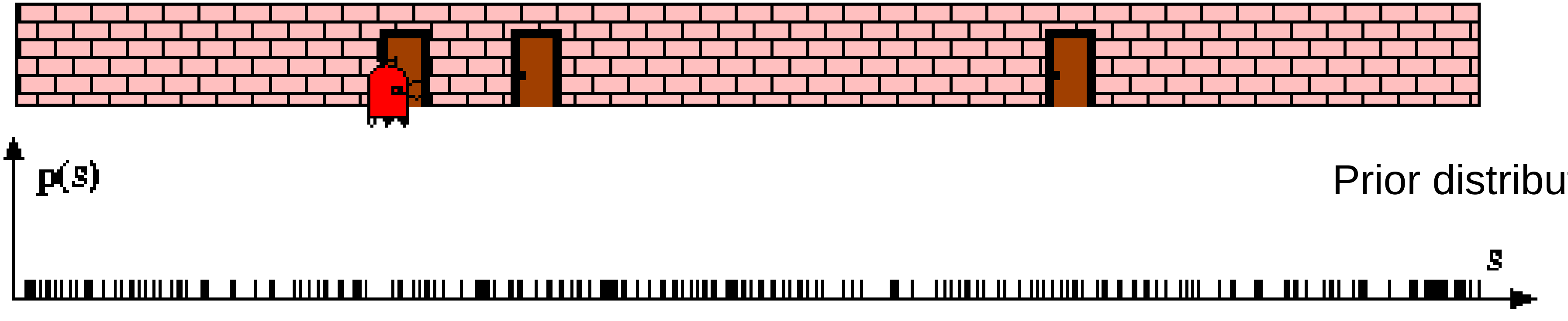
$$w_{t+1}^i = P(e_{t+1}|\bar{x}_{t+1}^i) w_t^i$$

Resample

$$X_{t+1} = \{\}$$

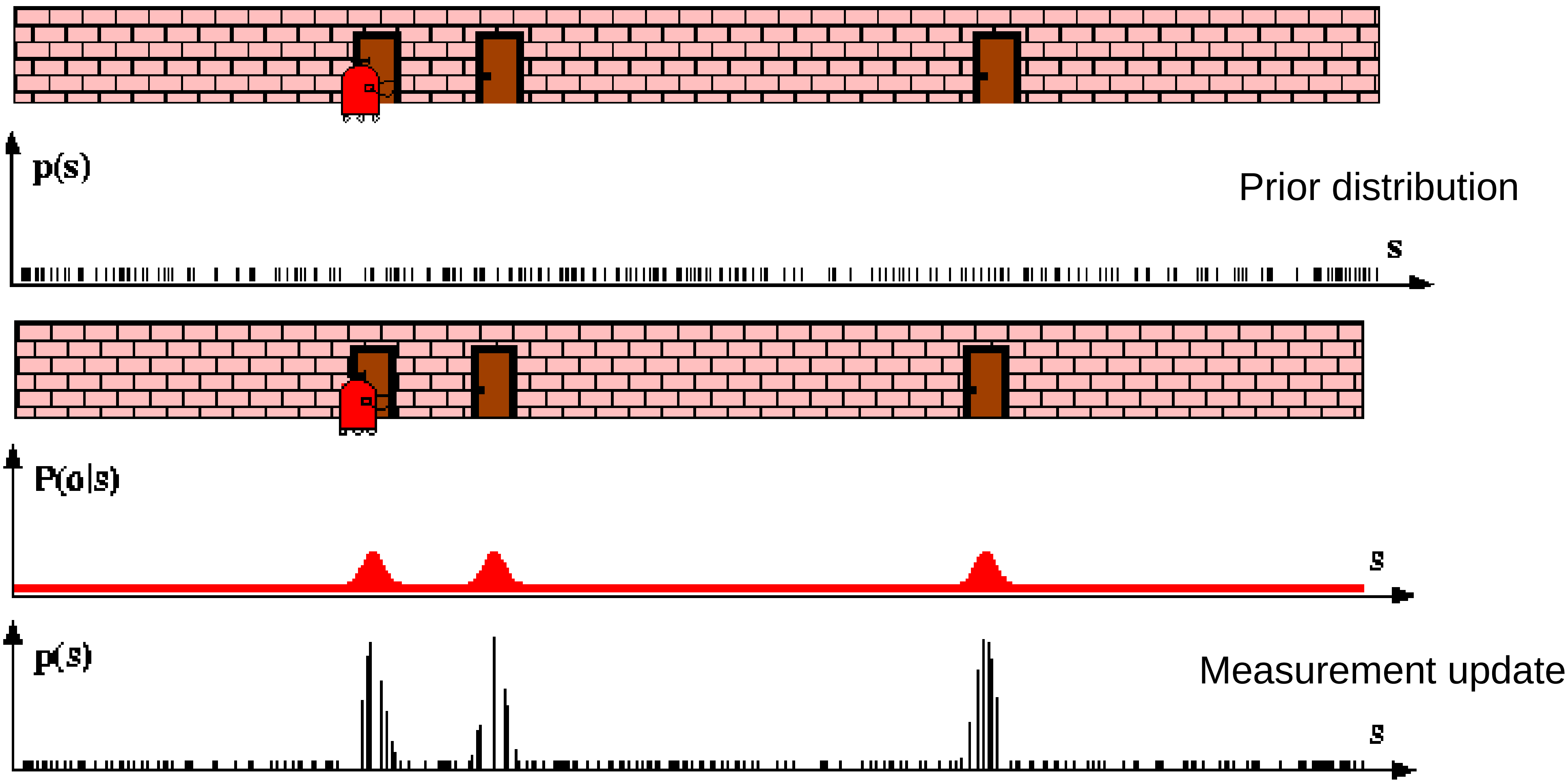
$$X_{t+1} = X_{t+1} \cup \bar{x}_{t+1}^i \text{ w/ prob } w_{t+1}^i$$

Particle Filter

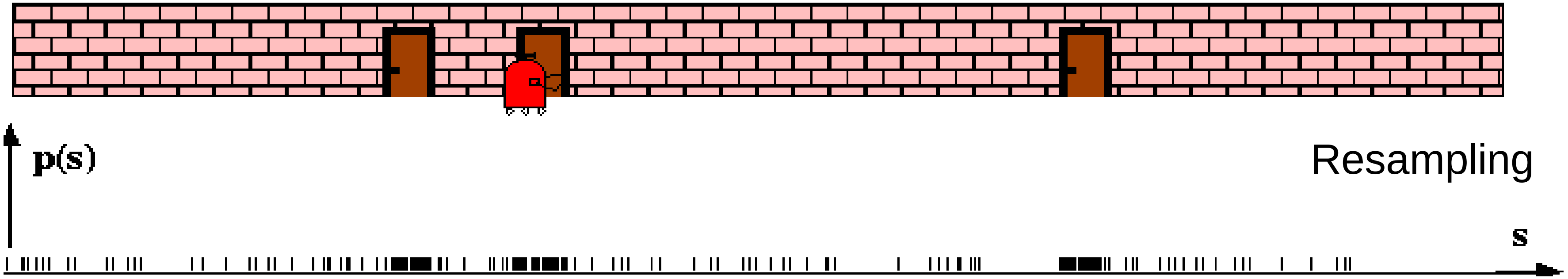


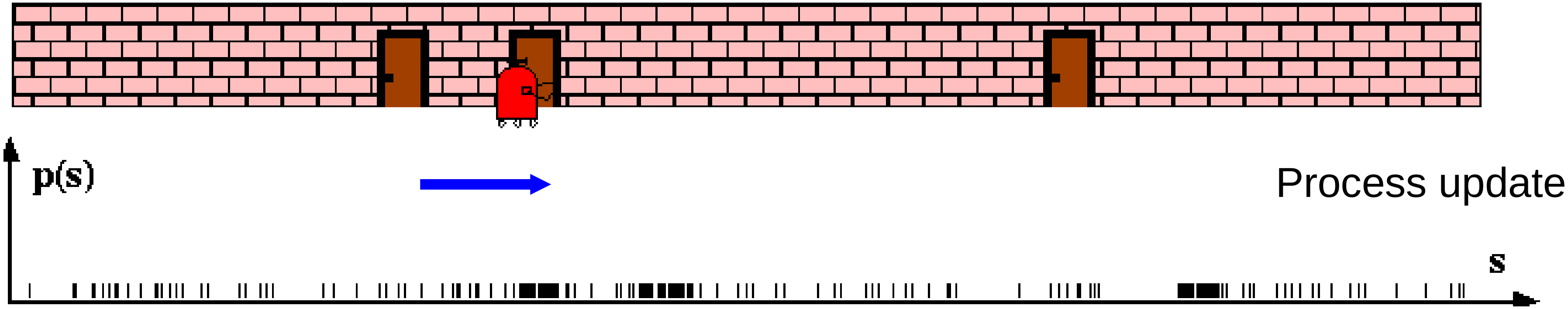
Prior distribution

Particle Filter

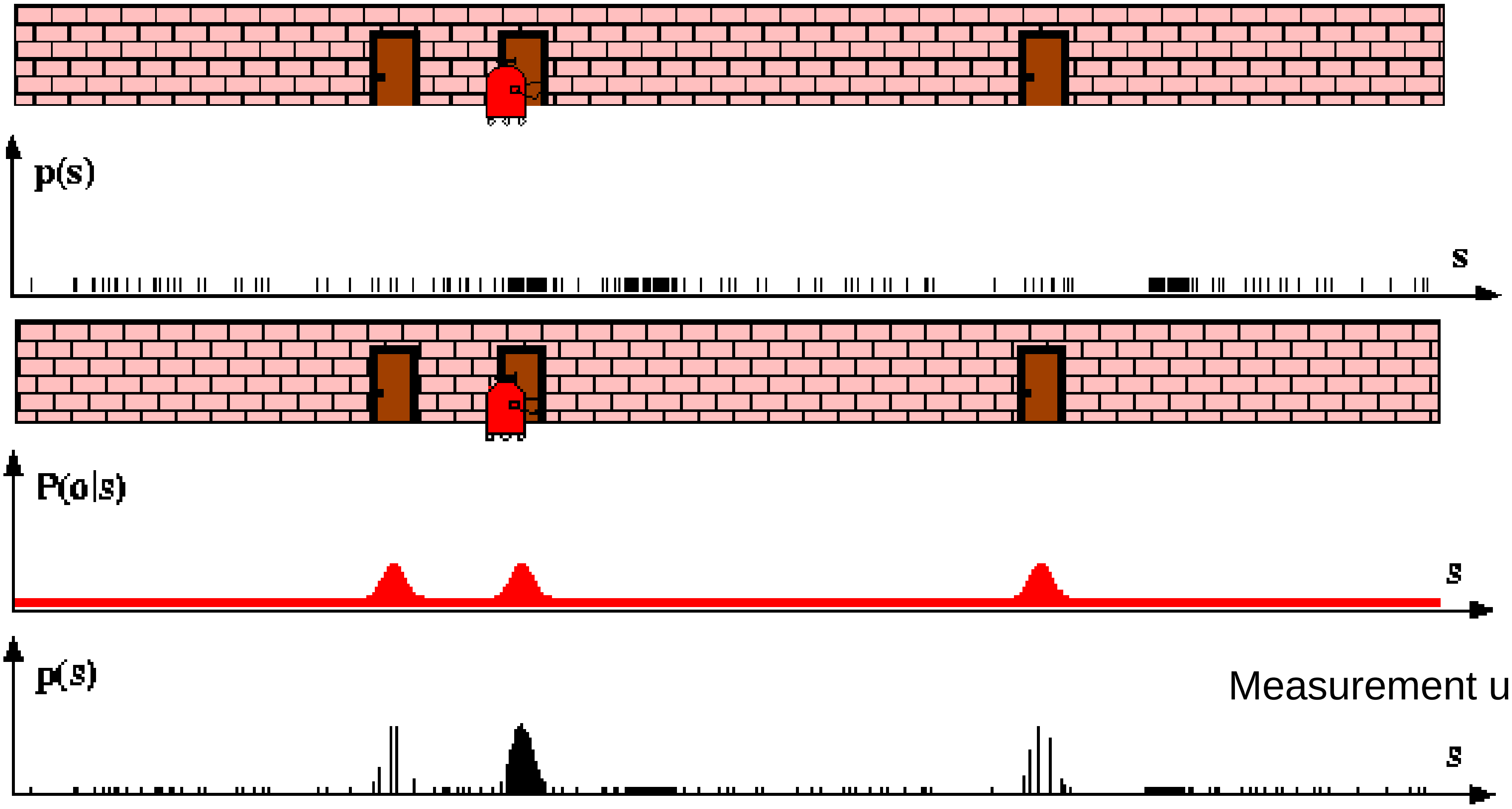


Particle Filter



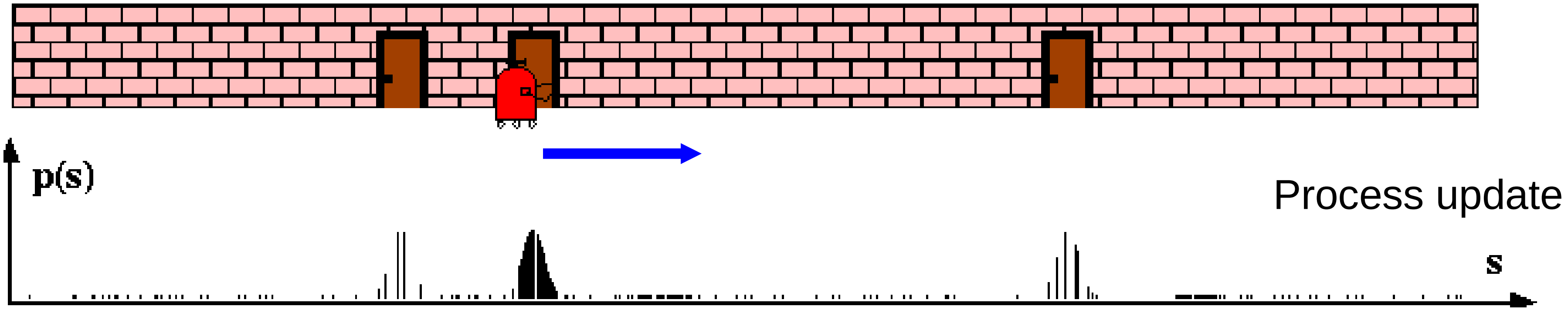


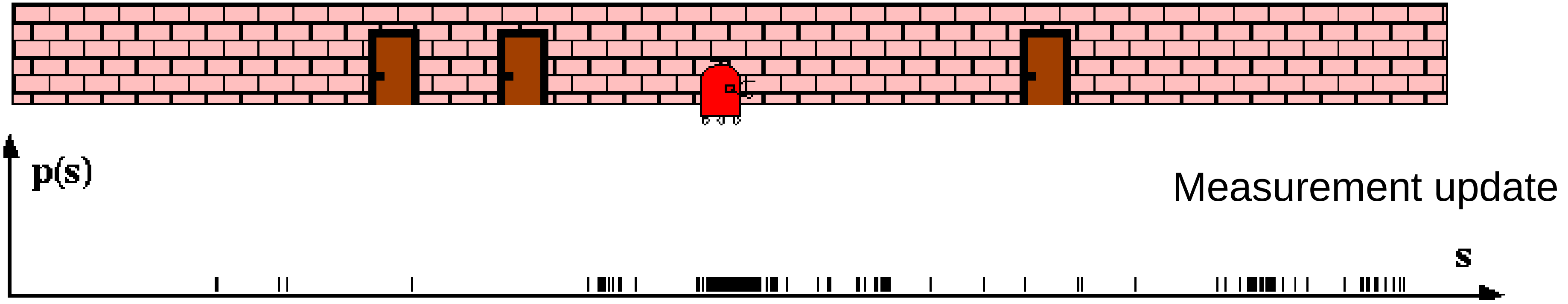
Particle Filter



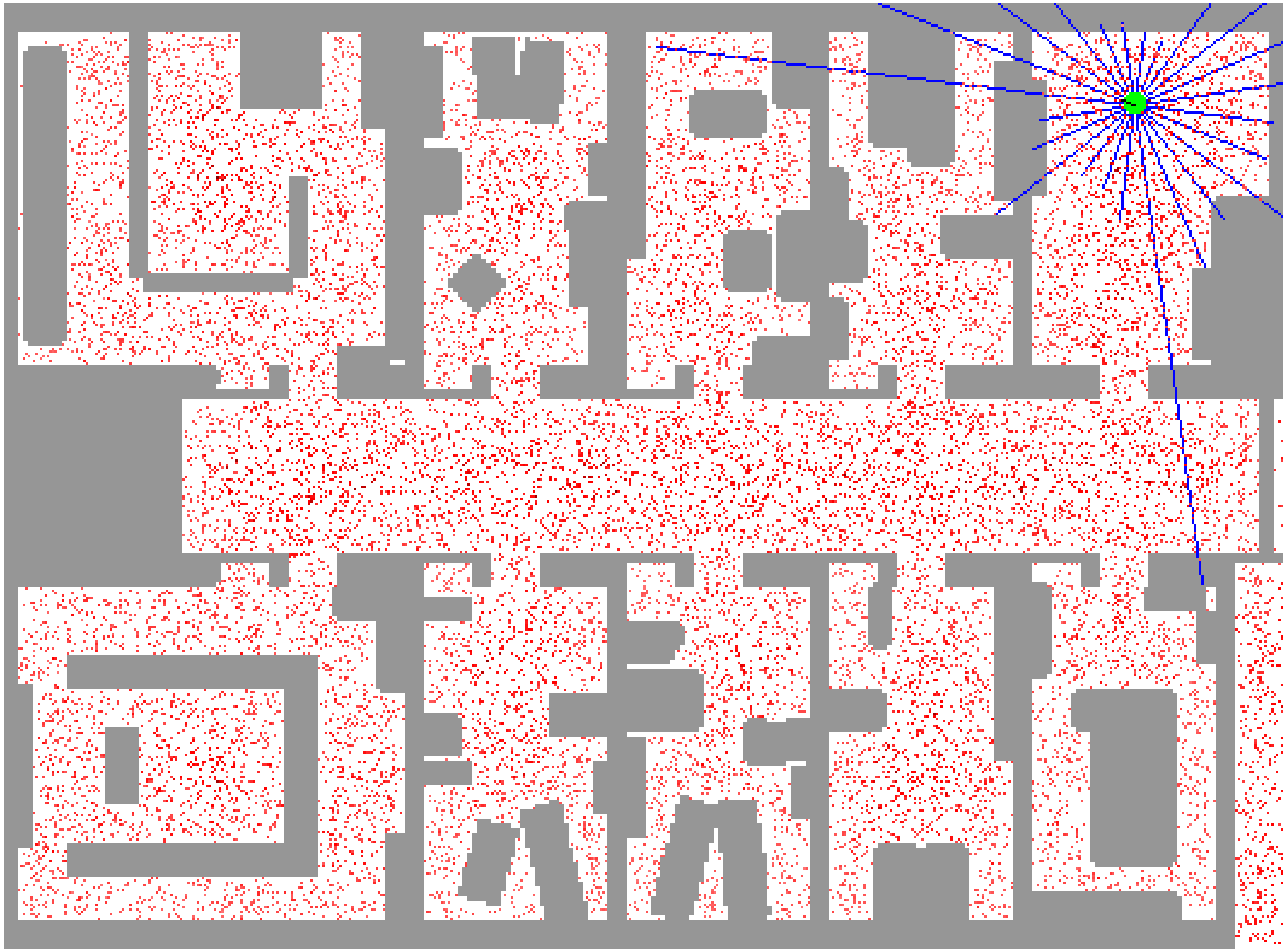
Measurement update

Particle Filter

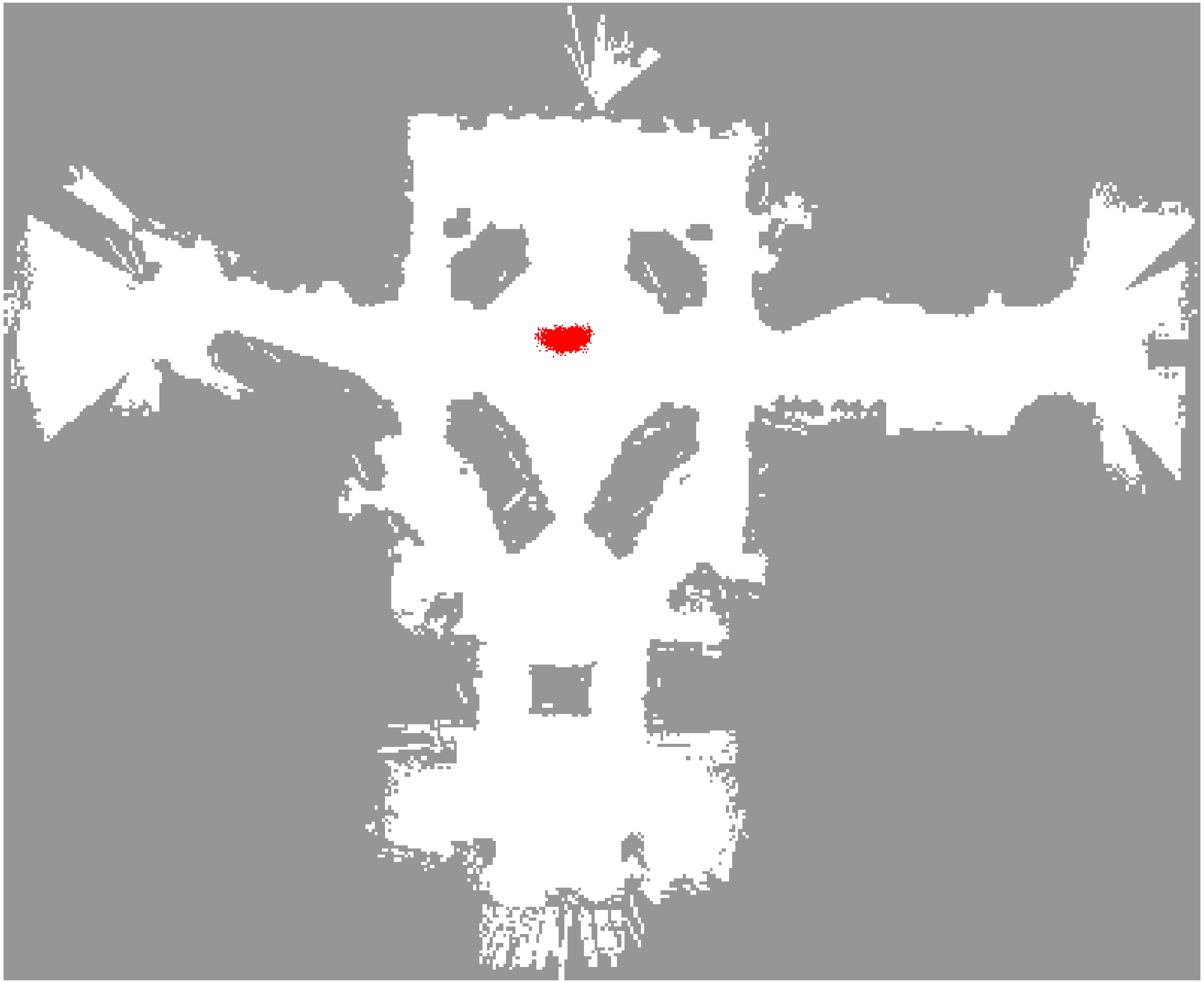




Particle Filter Example



Particle Filter Example



Particle Filtering

Pros:

– works in continuous spaces
– can represent multi-modal distributions

Cons:

– parameters to tune
– sample impoverishment

Sample Impoverishment

<u>Pros:</u>

– works in continuous spaces
– can represent multi-modal distributions

<u>Cons:</u>

– parameters to tune
– sample impoverishment

No particles nearby the true system state

Sample Impoverishment

If there aren't enough samples, then we might ``resample away'' the true state...

Prior distribution

$x_t^n \qquad w_t^1, \ldots, w_t^n = 1$

Process update

$$P(X_{t+1} | x_t^i, e_{1:t})$$

Observation update

$$w_{t+1}^i = P(e_{t+1} | \bar{x}_{t+1}^i) w_t^i$$

$B(X_{t+1})$ $P(X_{t+1} | E_{1:t+1})$

Do this *n* times

Resample

$$X_{t+1} = \{\}$$

$X_{t+1} = X_{t+1} \cup \bar{x}_{t+1}^i$ w/ prob w_{t+1}^i

Sample Impoverishment

Prior distribution

$x_t^n \qquad w_t^1, \dots, w_t^n = 1$

If there aren't enough samples, then we might ``resample away'' the true state...

One solution: add an additional k samples drawn completely at random

Process update

$P(X_{t+1} | x_t^i, e_{1:t})$

Observation update

$w_{t+1} = P(e_{t+1} | \bar{x}_{t+1}^i) w_t^i$

$B(X_{t+1})$ $P(X_{t+1} | E_{1:t+1})$

Resample

$X_{t+1} = \{\}$

Do this *n* times → $X_{t+1} = X_{t+1} \cup \bar{x}_{t+1}^i$ w/ prob w_{t+1}^i

Sample Impoverishment

Prior distribution

$x_t^n \qquad w_t^1, \ldots, w_t^n = 1$

If there aren't enough samples, then we might ``resample away'' the true state...

One solution: add an additional k samples drawn completely at random

BUT: there's always a chance that the true state won't be represented well by the particles...

Process update

$P(X_{t+1} | x_t^i, e_{1:t})$

Observation update

$w_{t+1} = P(e_{t+1} | \bar{x}_{t+1}^i) w_t^i$

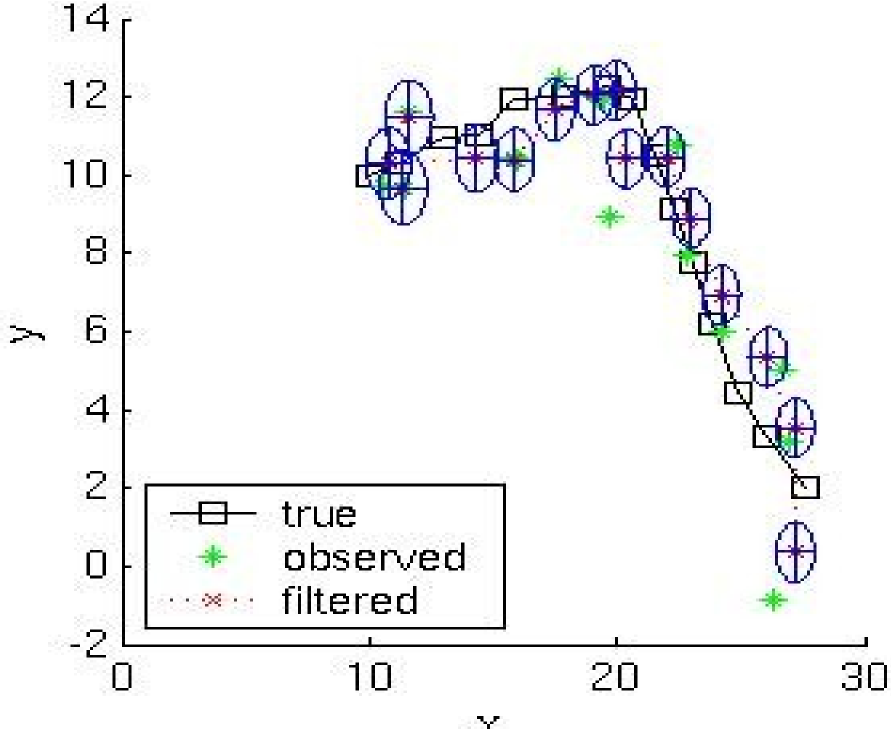
$B(X_{t+1}) \quad P(X_{t+1} | E_{1:t+1})$

Resample

$X_{t+1} = \{\}$

Do this n times → $X_{t+1} = X_{t+1} \cup \bar{x}_{t+1}^i$ w/ prob w_{t+1}^i

Kalman Filtering



Another way to adapt Sequential Bayes Filtering to continuous state spaces

– relies on representing the probability distribution as a Gaussian

– first developed in the early 1960s (before general Bayes filtering); used in Apollo program

Image: UBC, Kevin Murphy Matlab toolbox

Kalman Idea

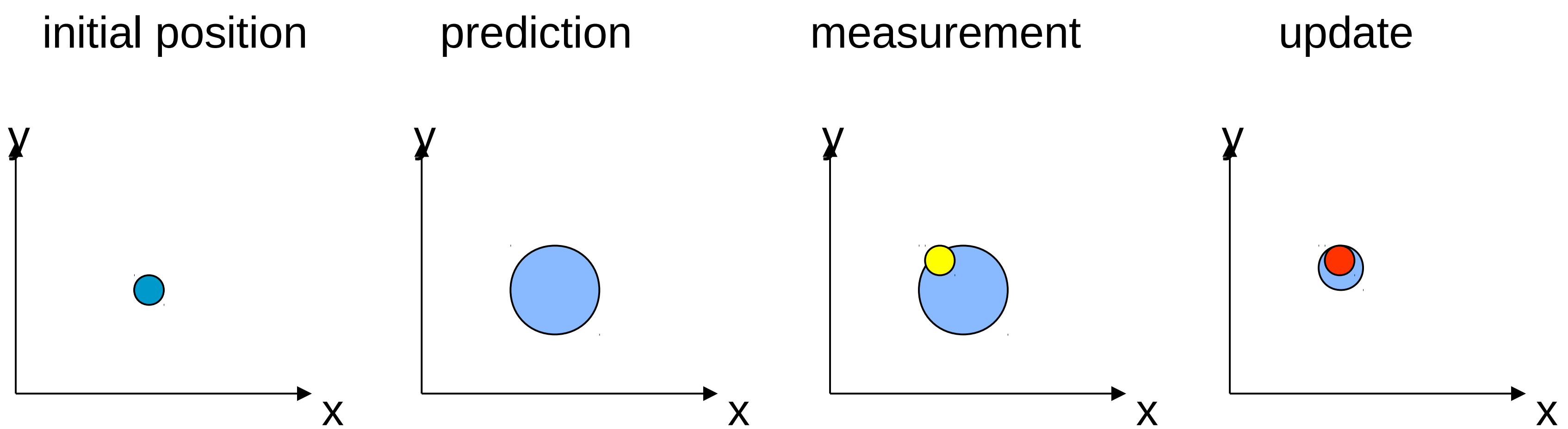


Image: Thrun *et al.*, *CS233B course notes*

Kalman Idea

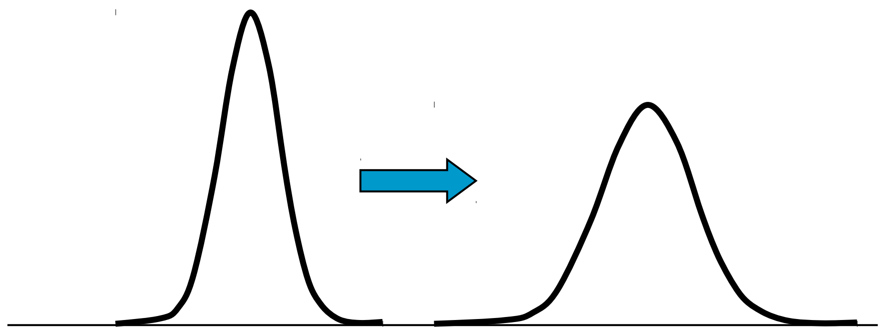


Image: Thrun *et al.*, *CS233B course notes*

$$P(x_{t+1}|z_{0:t}) = \sum_{x_t} P(x_{t+1}|x_t)P(x_t|z_{0:t})$$

$$P(x_{t+1}|z_{0:t+1}) = \eta P(z_{t+1}|x_{t+1})P(x_{t+1}|z_{0:t})$$

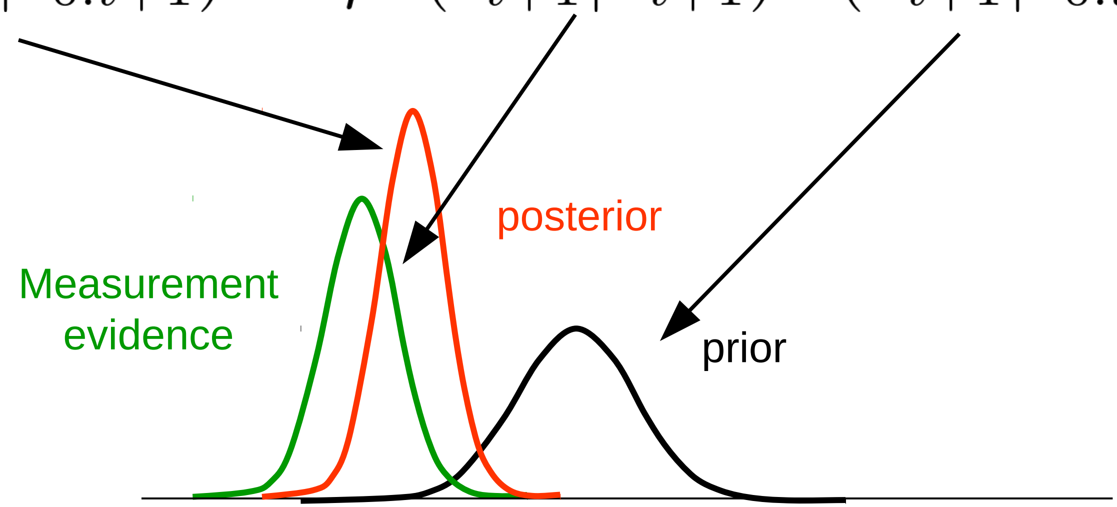


Image: Thrun *et al.*, *CS233B course notes*

Gaussians

- Univariate Gaussian:

$$P(x) = \eta e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

- Multivariate Gaussian:

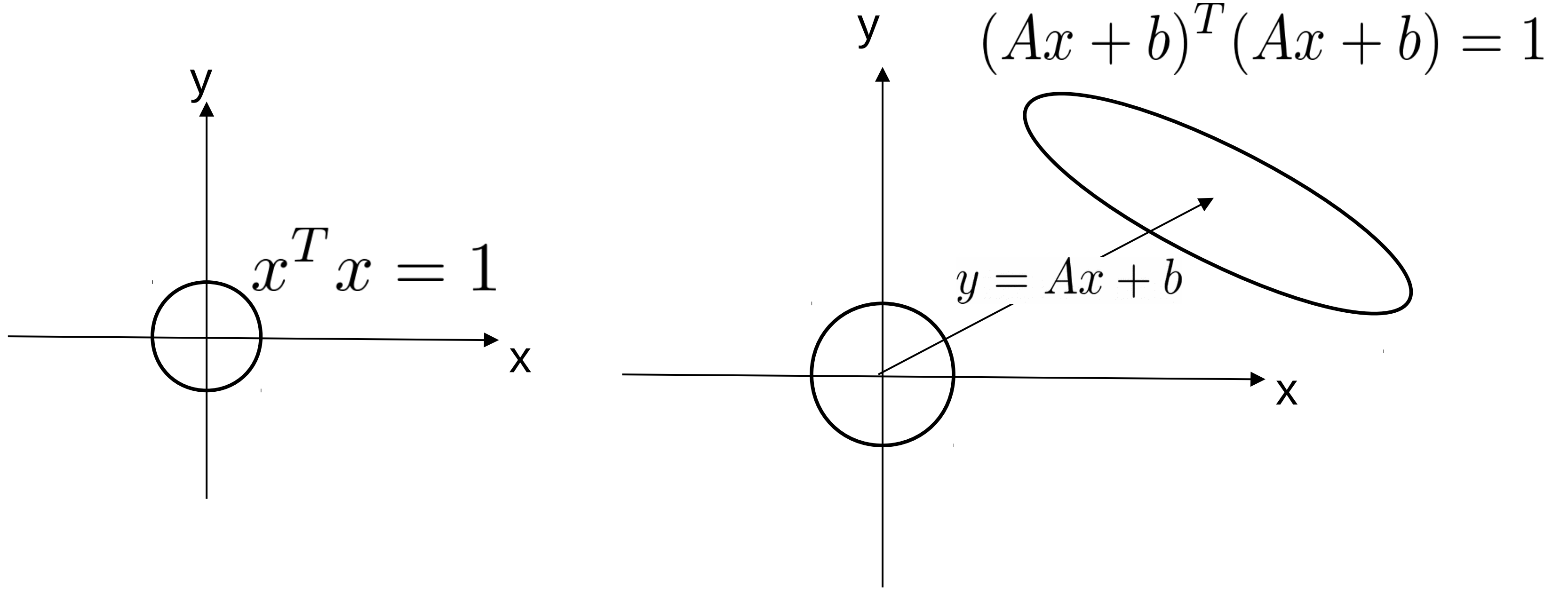
$$P(x) = \eta e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

$$P(x) = N(x;\mu,\Sigma)$$

Playing w/ Gaussians

- Suppose: $$P(x) = N(x; \mu, \Sigma)$$ $$y = Ax + b$$

- Calculate: $$P(y) = ?$$ $$P(y) = N(y; A\mu + b, A\Sigma A^T)$$



In fact

- Suppose: $P(x) = N(x; \mu, \Sigma)$

  $y = Ax + b$

- Then:

$$P\begin{pmatrix} x \\ y \end{pmatrix} = N\left[\begin{matrix} x \\ y \end{matrix} \ : \ \begin{matrix} \mu \\ A\mu + b \end{matrix}, \begin{pmatrix} \Sigma & \Sigma A^T \\ A\Sigma & A\Sigma A^T \end{pmatrix}\right]$$

Illustration

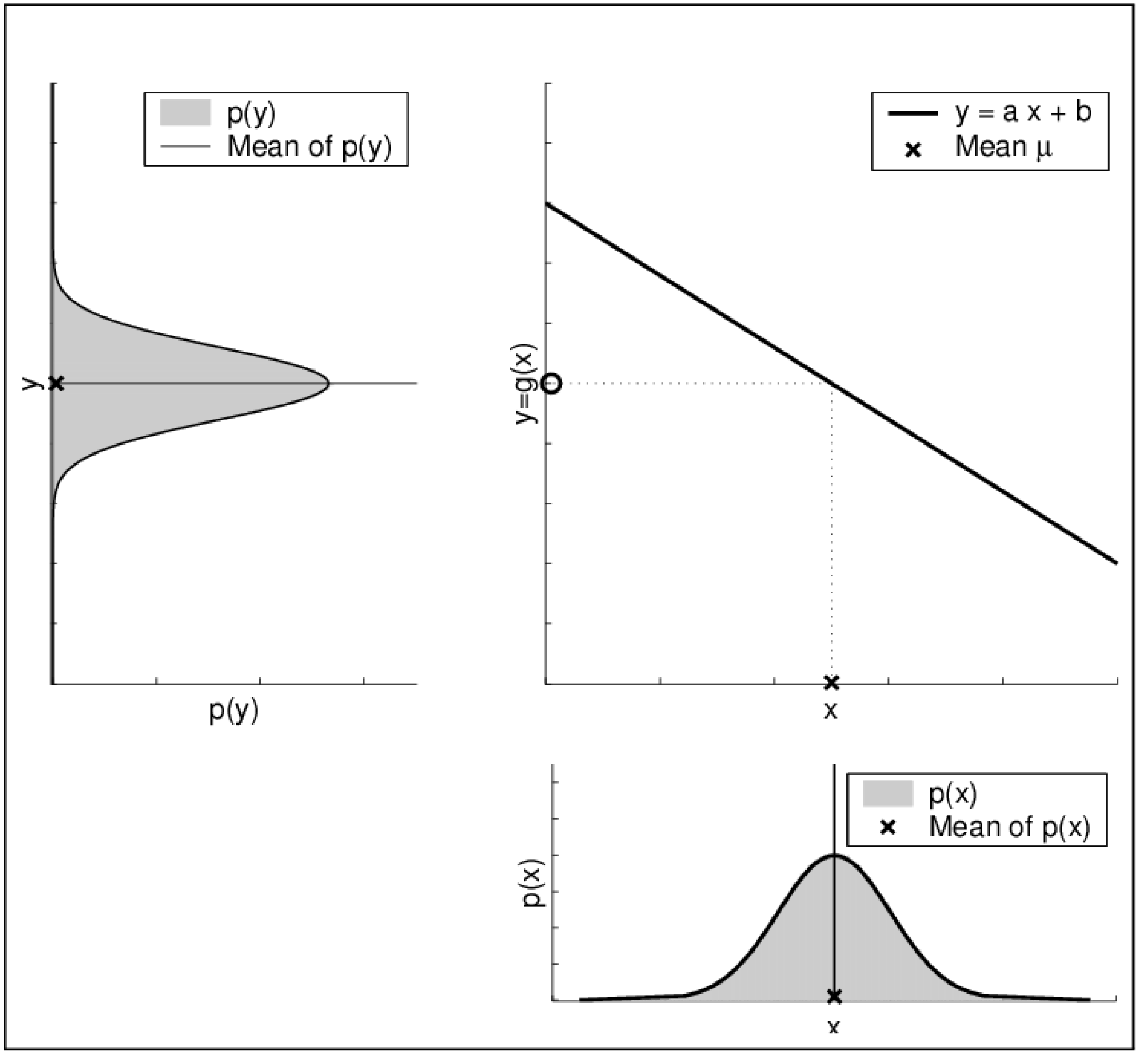


Image: Thrun *et al.*, *CS233B course notes*

And

Suppose: $P(x) = N(x; \mu, \Sigma)$

$$P(y|x) = N(y; Ax + b, R)$$

Then:

$$P\begin{pmatrix} x \\ y \end{pmatrix} = N\left[\begin{matrix} x \\ y \end{matrix} : \begin{matrix} \mu \\ A\mu + b \end{matrix}, \begin{pmatrix} \Sigma & \Sigma A^T \\ A\Sigma & A\Sigma A^T + R \end{pmatrix} \right]$$

$$P(y) = N(y; A\mu + b, A\Sigma A^T + R)$$

Marginal distribution

Does this remind us of anything?

Does this remind us of anything?

Process update (discrete):

Process update (continuous): $$P(x_{t+1}|z_{0:t}) = \int_{x_t} P(x_{t+1}|x_t)P(x_t|z_{0:t})$$

Does this remind us of anything?

Process update (discrete):

$$P(x_{t+1}|z_{0:t}) = \sum_{x_t} P(x_{t+1}|x_t)P(x_t|z_{0:t})$$

Process update (continuous):

$$N(x_{t+1}|Ax_t, Q)$$

transition dynamics

$$N(x_t|\mu_t, \Sigma_t)$$

prior

Does this remind us of anything?

Process update (discrete):

$$P(x_{t+1}|z_{0:t}) = \sum_{x_t} P(x_{t+1}|x_t)P(x_t|z_{0:t})$$

Process update (continuous):

$$N(x_{t+1}|Ax_t, Q)$$

transition dynamics

$$N(x_t|\mu_t, \Sigma_t)$$

prior

$$P(x_{t+1}|z_{0:t}) = \int_{x_t} N(x_{t+1}|Ax_t, Q)N(x_t; \mu_t, \Sigma_t)$$

$$P(x_{t+1}|z_{0:t}) = N(x_{t+1}|A\mu_t, A\Sigma_t A^T + Q)$$

Observation update

Observation update:

$$N(z_{t+1}|Cx_{t+1}, R) \qquad N(x_t|\mu'_t, \Sigma'_t)$$

Where: $\mu'_t = A\mu_t$

$$\Sigma'_t = A\Sigma_t A^T + Q$$

Observation update

Observation update:

$$N(x_t|\mu_t', \Sigma_t')$$

Where:

$$\Sigma_t' = A\Sigma_t A^T + Q$$

$$P(z_{t+1}, x_{t+1}|z_{0:t}) = \eta N(z_{t+1}|Cx_t, R)N(x_t; \mu_t', \Sigma_t')$$

Observation update

Observation update:

$$N(x_t | \mu'_t, \Sigma'_t)$$

Where:

$$\Sigma'_t = A \Sigma_t A^T + Q$$

$$P(z_{t+1}, x_{t+1} | z_{0:t}) = N \left[\begin{array}{c} x_{t+1} \\ z_{t+1} \end{array} : \begin{array}{c} \mu'_t \\ C\mu'_t \end{array}, \left(\begin{array}{cc} \Sigma'_t & \Sigma'_t C^T \\ C\Sigma'_t & C\Sigma'_t A^T + R \end{array} \right) \right]$$

Observation update

But we need: $P(x_{t+1}|z_{0:t+t}) = ?$

Another Gaussian identity...

Suppose: $$N\left[\begin{array}{ccc} x & : & a \\ y & & b \end{array}, \left(\begin{array}{cc} A & C \\ C^T & B \end{array}\right)\right]$$

Calculate: $$P(y|x) = ?$$

$$P(y|x) = N(y|b + C^T A^{-1}(x-a), B - C^T A^{-1} C)$$

Observation update

$$P(z_{t+1}, x_{t+1} | z_{0:t}) = N\left[\begin{array}{c} x_{t+1} \\ z_{t+1} \end{array} : \begin{array}{c} \mu_t' \\ C\mu_t' \end{array}, \left(\begin{array}{cc} \Sigma & \Sigma C^T \\ C\Sigma & C\Sigma A^T + R \end{array} \right) \right]$$

But we need: $P(x_{t+1} | z_{0:t+1}) = ?$

$$P(x_{t+1} | z_{0:t+1}) = N(x_{t+1}; \mu_{t+1}, \Sigma_{t+1})$$

$$\mu_{t+1} = \mu_t' + \Sigma_t' C^T (R + C\Sigma_t' C^T)^{-1} (z_{t+1} - C\mu_t')$$

$$\Sigma_{t+1} = \Sigma_t' - \Sigma_t' C^T (R + C\Sigma_t' C^T)^{-1} C\Sigma_t'$$

To summarize the Kalman filter

System: $$P(x_{t+1}|x_t) = N(x_{t+1}|Ax_t, Q)$$
$$P(z_{t+1}|x_{t+1}) = N(z_{t+1}|Cx_{t+1}, R)$$

Prior: μ_t

Σ_t

Process update: $\mu'_t = A\mu_t$

Measurement update:

Suppose there is an action term...

System: $$P(x_{t+1}|x_t) = N(x_{t+1}|Ax_t + u_t, Q)$$

Prior:

$$\Sigma_t$$

Process update: $$\mu'_t = A\mu_t + u_t$$
$$\Sigma'_t = A\Sigma_t A^T + Q$$

Measurement update:

To summarize the Kalman filter

Prior:

$$\Sigma_t$$

Process update: $$\mu'_t = A\mu_t$$

$$\Sigma'_t = A\Sigma_t A^T + Q$$

Measurement update:

This factor is often called the "Kalman gain"

Things to note about the Kalman filter

Process update: $\mu'_t = A\mu_t$

Measurement update:

– covariance update is independent of observation

– Kalman is only optimal for linear-Gaussian systems

– the distribution “stays” Gaussian through this update

– the error term can be thought of as the different between the observation and the prediction

Kalman in 1D

System:

$$P(x_{t+1}|x_t) = N(x_{t+1} : x_t + u_t, q)$$
$$P(z_{t+1}|x_{t+1}) = N(z_{t+1}|2x_{t+1}, r)$$

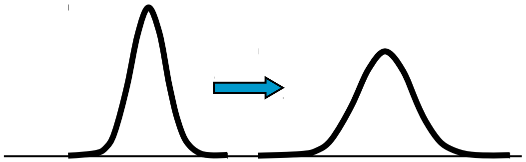


Image: Thrun *et al.*, *CS233B course notes*

Process update:

$$\bar{\mu}_t = \mu_t + u_t$$

$$\bar{\sigma}_t^2 = \sigma_t^2 + q$$

Measurement update:

$$\mu_{t+1} = \bar{\mu}_t + \frac{2\bar{\sigma}_t^2}{r + 4\bar{\sigma}_t^2}(z_{t+1} - \bar{\mu}_t)$$

$$\sigma_{t+1} = \bar{\sigma}_t^2 - \frac{4(\bar{\sigma}_t^2)^2}{r + 4\bar{\sigma}_t^2}$$

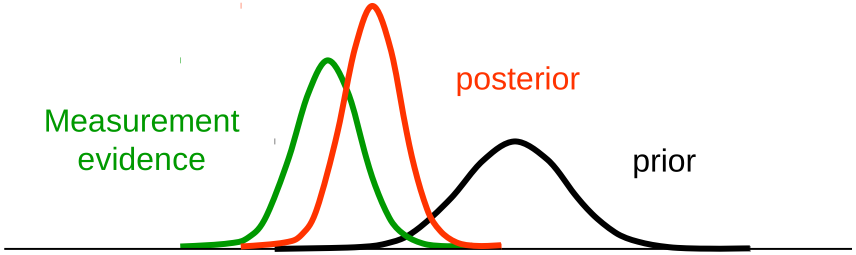


Image: Thrun *et al.*, *CS233B course notes*

Kalman Idea

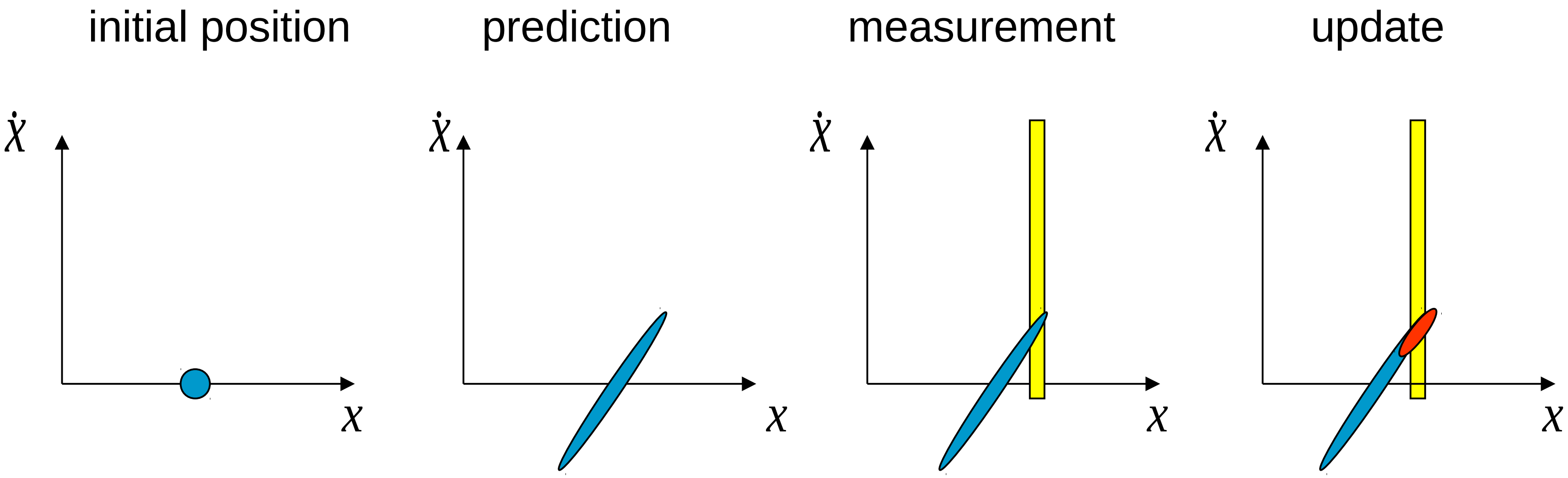


Image: Thrun *et al.*, *CS233B course notes*

Example: estimate velocity

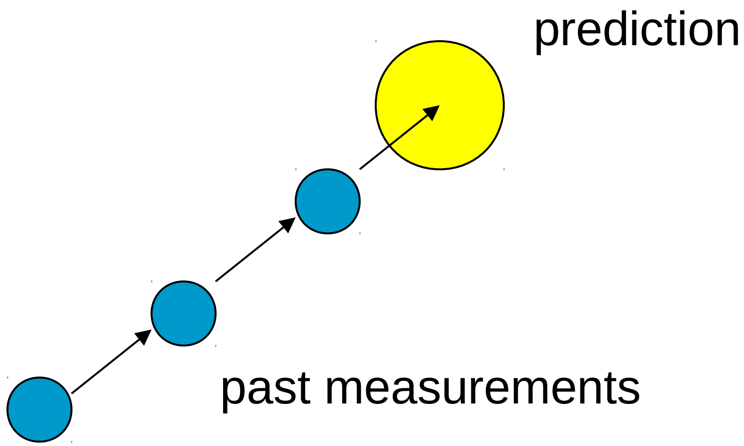


Image: Thrun *et al.*, *CS233B course notes*

Example: filling a tank

$$x = \begin{pmatrix} l \\ f \end{pmatrix}$$

← Level of tank

← Fill rate

$$l_{t+1} = l_t + f dt$$

Process: $$x_{t+1} = \begin{pmatrix} 1 & dt \\ 0 & 1 \end{pmatrix} x_t + q$$

Observation: $$z_{t+1} = \begin{pmatrix} 1 & 0 \end{pmatrix} x_{t+1} + r$$

Example: estimate velocity

$$x_{t+1} = Ax_t + w_t$$

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \dot{x}_{t+1} \\ \dot{y}_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{pmatrix} + \mathrm{w}_t$$

$$z_{t+1} = Cx_{t+1} + r_{t+1}$$

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \dot{x}_{t+1} \\ \dot{y}_{t+1} \end{pmatrix} + \mathrm{r}_{t+1}$$

But, my system is NON-LINEAR!

$$x_{t+1} = f(x_t, u_t)$$
$$\neq Ax_t + Bu_t$$

What should I do?

But, my system is NON-LINEAR!

$$x_{t+1} = f(x_t, u_t)$$

- What should I do?

Well, there are some options...

-

But, my system is NON-LINEAR!

$$x_{t+1} = f(x_t, u_t)$$

- What should I do?

Well, there are some options...

- But none of them are great.
-

But, my system is NON-LINEAR!

$$x_{t+1} = f(x_t, u_t)$$

- What should I do?

Well, there are some options...

But none of them are great.

Here's one: the Extended Kalman Filter

Extended Kalman filter

Take a Taylor expansion:

$$x_{t+1} = f(x_t, u_t)$$
$$\approx f(\mu_t, u_t) + A_t(x_t - \mu_t)$$

Where: $A_t = \frac{\partial f}{\partial x}(\mu_t, u_t)$

$$z_{t+1} = h(x_t)$$
$$\approx h(\mu_t) + C_t(x_t - \mu_t)$$

Where: $C_t = \frac{\partial h}{\partial x}(\mu_t)$

Extended Kalman filter

Take a Taylor expansion:

$$x_{t+1} = f(x_t, u_t)$$

Where: $$A_t = \frac{\partial f}{\partial x}(\mu_t, u_t)$$

$$z_{t+1} = h(x_t)$$

Where: $$C_t = \frac{\partial h}{\partial x}(\mu_t)$$

Then use the same equations...

To summarize the EKF

Prior:

$$\Sigma_t$$

Process update:

$$\mu'_t = f(\mu_t, u_t)$$

$$\Sigma'_t = A_t \Sigma_t A_t^T + Q$$

Measurement update:

$$\mu_{t+1} = \mu'_t + \Sigma'_t C^T (R + C \Sigma'_t C^T)^{-1} (z_{t+1} - h(\mu'_t))$$

Extended Kalman filter

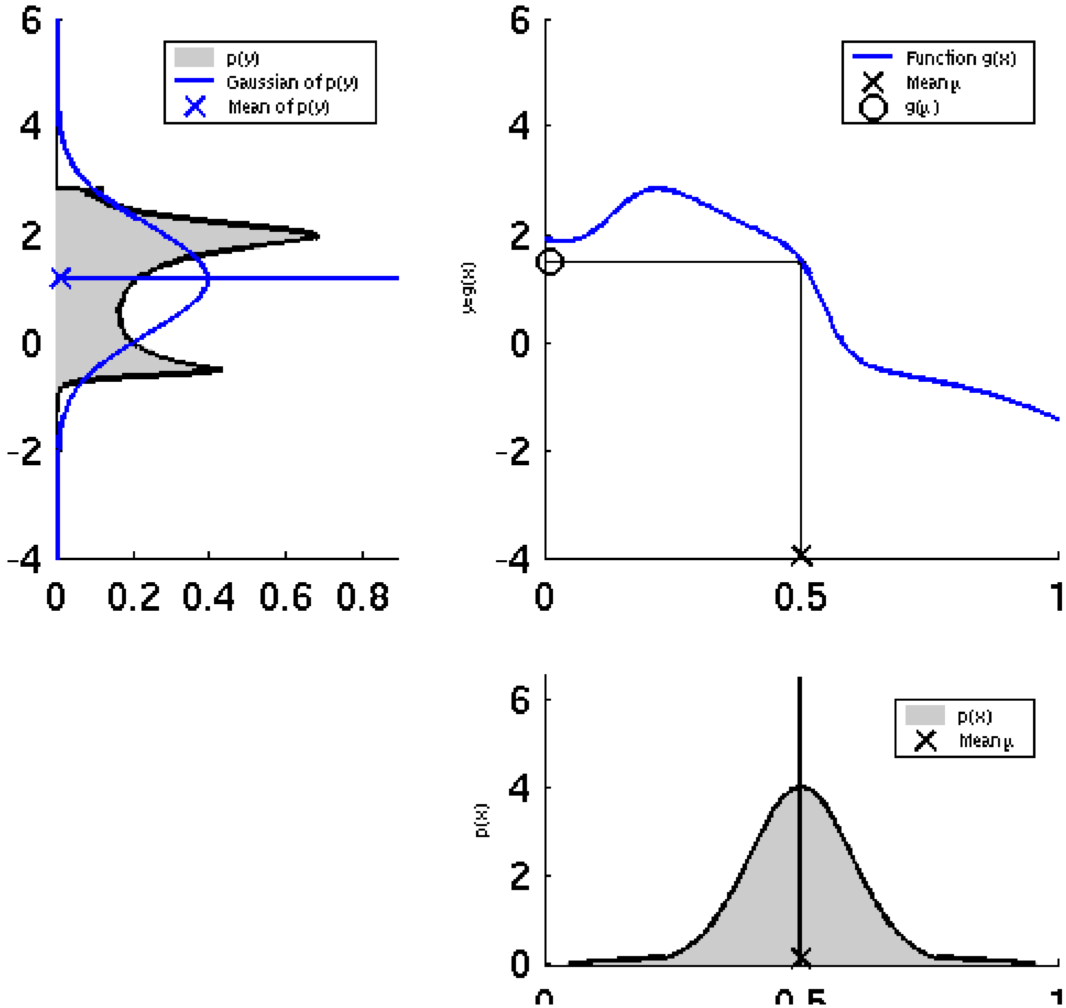


Image: Thrun *et al.*, *CS233B course notes*

Extended Kalman filter

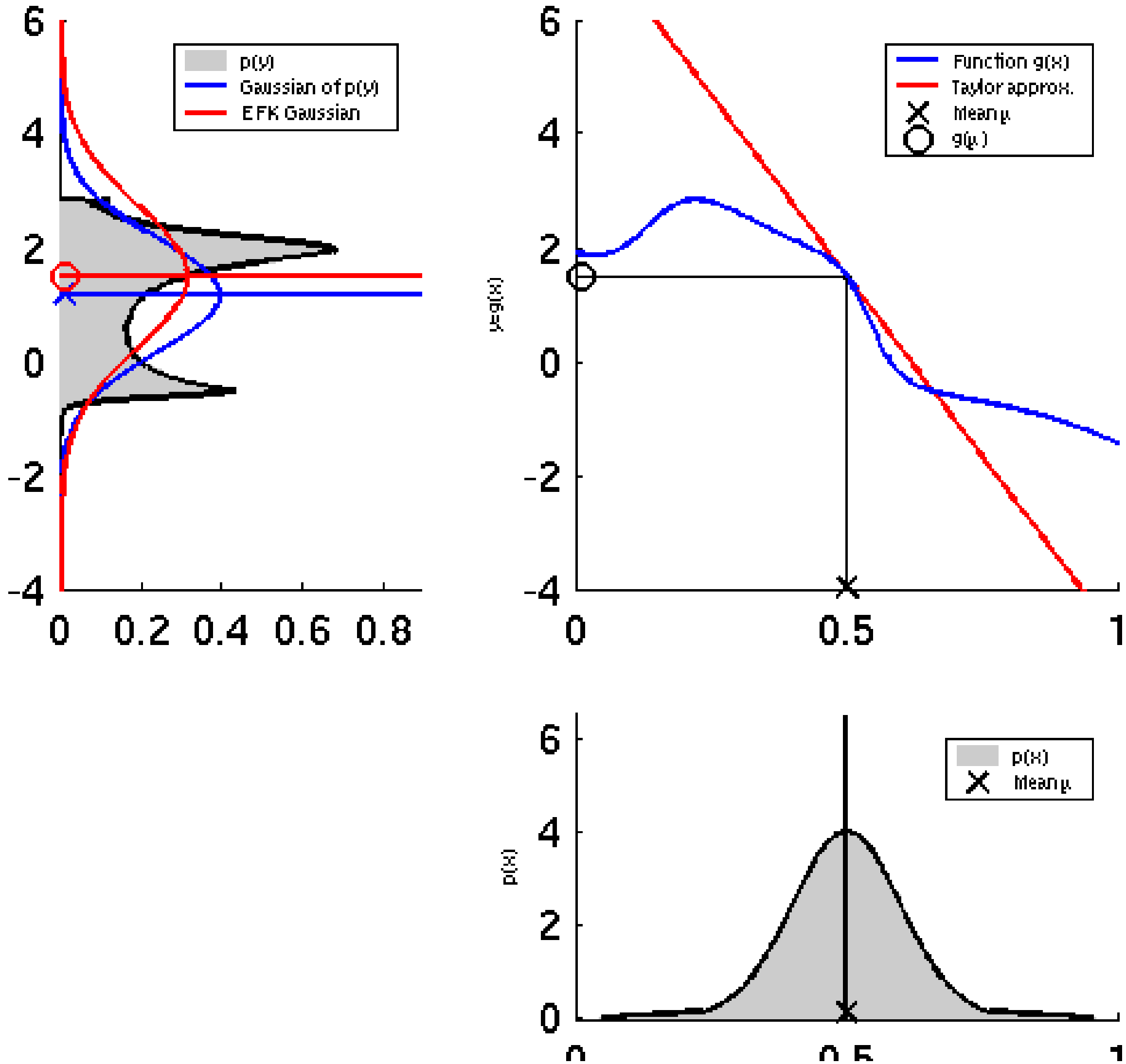
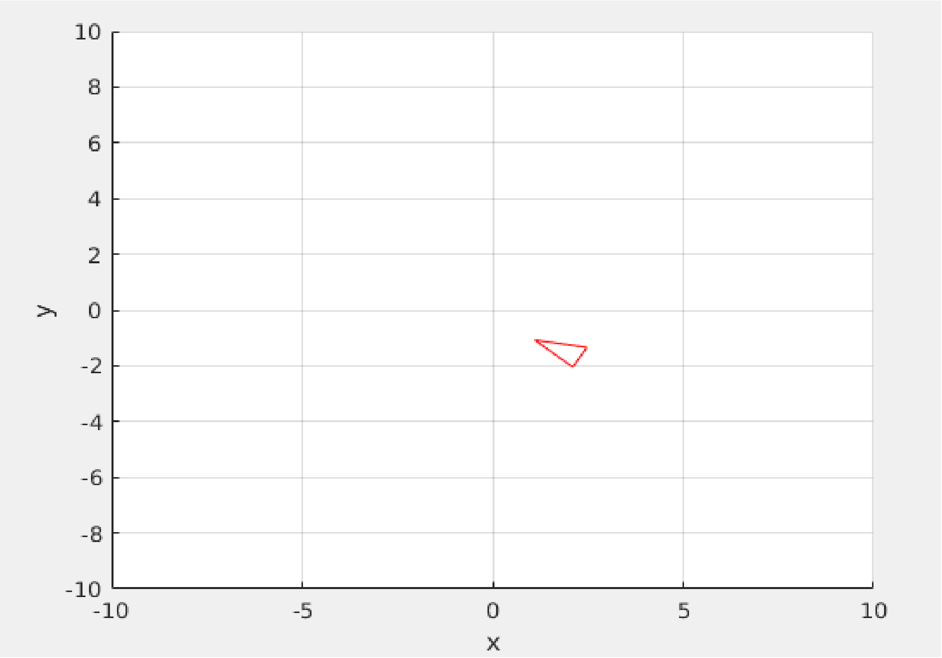


Image: Thrun *et al.*, *CS233B course notes*

EKF Mobile Robot Localization



Suppose we have a mobile robot wandering around in a 2-d world ...

noise

Process dynamics: $\boldsymbol{x}\langle k+1\rangle = \boldsymbol{f}\left(\boldsymbol{x}\langle k\rangle,\, \delta\langle k\rangle,\, \boldsymbol{v}\langle k\rangle\right)$

state

Odometry measurement

Process noise is assumed to be Gaussian: $\boldsymbol{v} = (\nu_d,\, \nu_\theta) \sim N(0,\, V)$

EKF Mobile Robot Localization

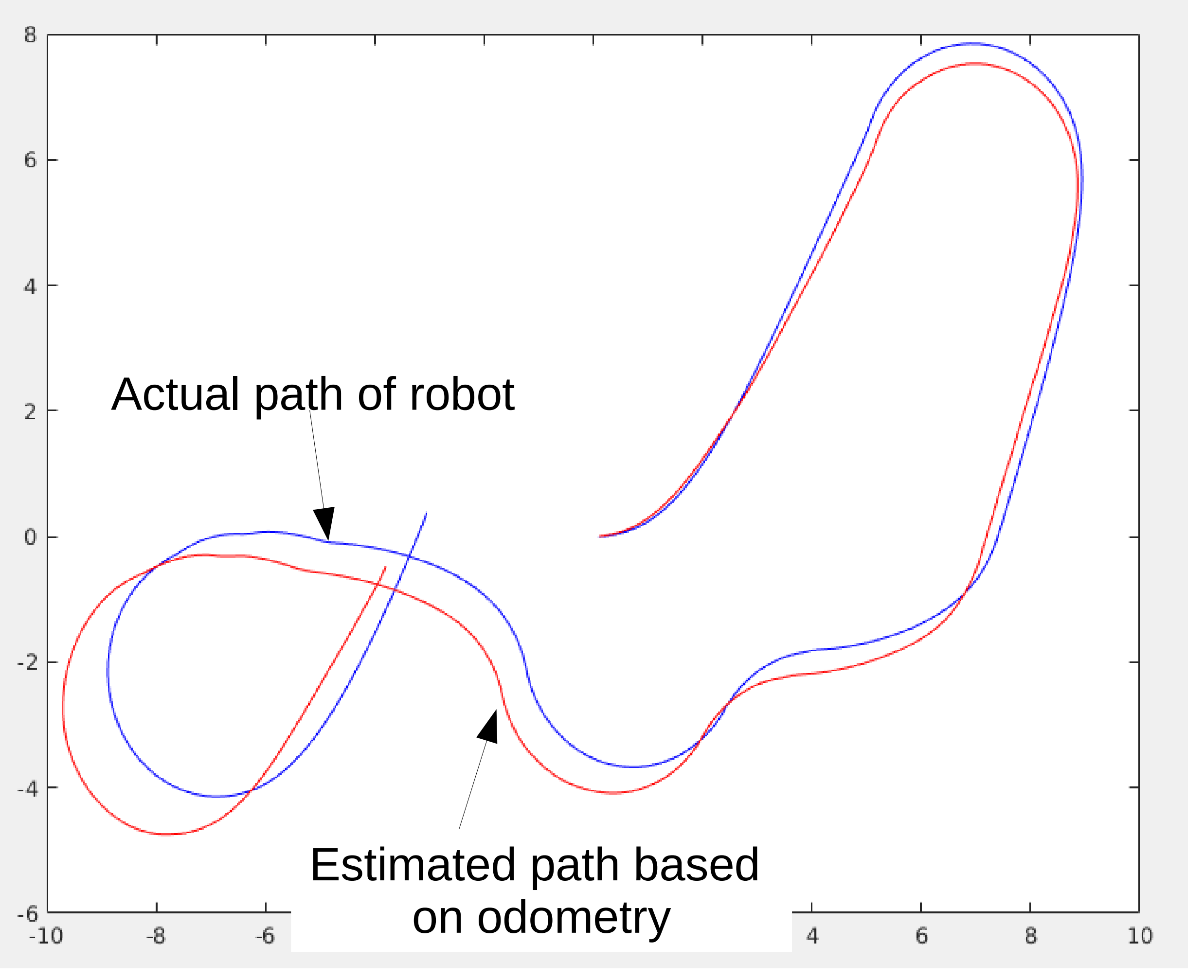
Process dynamics:

noise

$$\xi\langle k+1\rangle = \begin{pmatrix} x\langle k\rangle + (\delta_d\langle k\rangle + \nu_d)\cos(\theta\langle k\rangle + \delta_\theta + \nu_\theta) \\ y\langle k\rangle + (\delta_d\langle k\rangle + \nu_d)\sin(\theta\langle k\rangle + \delta_\theta + \nu_\theta) \\ \theta\langle k\rangle + \delta_\theta + \nu_\theta \end{pmatrix}$$

Odometry measurement

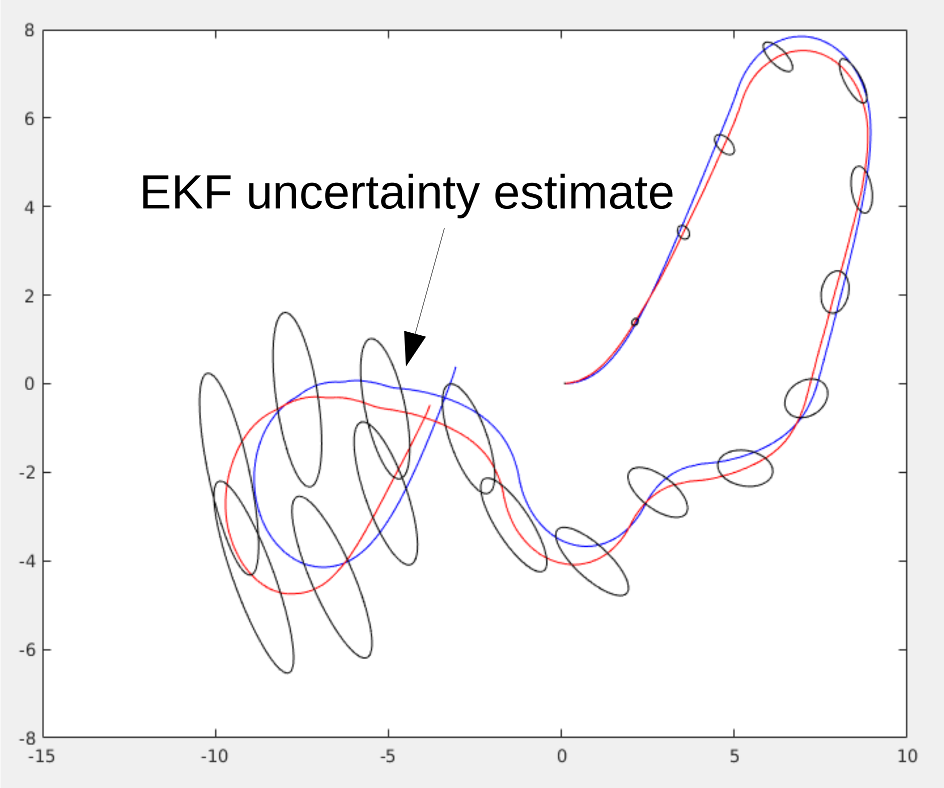
EKF Mobile Robot Localization



But, wheels slip – odometry is not always correct...

How do we localize? Extended Kalman Filter!

EKF Process Update



Dynamics:

Linearized dynamics:

$$\hat{\boldsymbol{x}}\langle k+1\rangle = \hat{\boldsymbol{x}}\langle k\rangle + \boldsymbol{F}_x\left(\boldsymbol{x}\langle k\rangle - \hat{\boldsymbol{x}}\langle k\rangle\right) + \boldsymbol{F}_v\boldsymbol{v}\langle k\rangle$$

Where:

$$\boldsymbol{F}_x = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{v}=0} = \begin{pmatrix} 1 & 0 & -\delta_d\langle k\rangle - \sin(\theta\langle k\rangle + \delta_\theta) \\ 0 & 1 & \delta_d\langle k\rangle\cos(\theta\langle k\rangle + \delta_\theta) \\ 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{F}_v = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}}\right|_{\boldsymbol{v}=0} = \begin{pmatrix} \cos(\theta\langle k\rangle + \delta_\theta) & -\delta_d\langle k\rangle\sin(\theta\langle k\rangle + \delta_\theta) \\ \sin(\theta\langle k\rangle + \delta_\theta) & \delta_d\langle k\rangle\cos(\theta\langle k\rangle + \delta_\theta) \\ 0 & 1 \end{pmatrix}$$

EKF Process Update

Dynamics:

Linearized dynamics:

$$\hat{\boldsymbol{x}}\langle k+1\rangle = \hat{\boldsymbol{x}}\langle k\rangle + \boldsymbol{F}_x\big(\boldsymbol{x}\langle k\rangle - \hat{\boldsymbol{x}}\langle k\rangle\big) + \boldsymbol{F}_v\boldsymbol{v}\langle k\rangle$$

EKF uncertainty estimate

Where:

$$\boldsymbol{F}_v = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}}\bigg|_{\boldsymbol{v}=0} = \begin{pmatrix} \cos(\theta\langle k\rangle + \delta_\theta) & -\delta_d\langle k\rangle\sin(\theta\langle k\rangle + \delta_\theta) \\ \sin(\theta\langle k\rangle + \delta_\theta) & \delta_d\langle k\rangle\cos(\theta\langle k\rangle + \delta_\theta) \\ 0 & 1 \end{pmatrix}$$

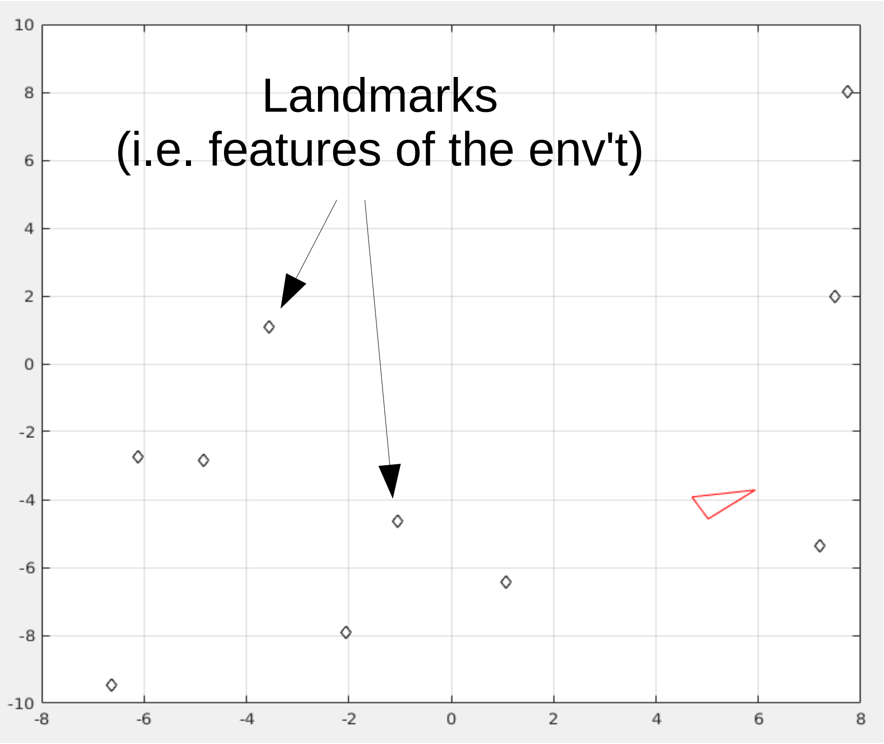
Process update:

$$\hat{\boldsymbol{x}}\langle k+1|k\rangle = \boldsymbol{f}\big(\hat{\boldsymbol{x}}\langle k\rangle,\, \delta\langle k\rangle,\, 0\big)$$

$$\hat{\boldsymbol{P}}\langle k+1|k\rangle = \boldsymbol{F}_x\langle k\rangle\hat{\boldsymbol{P}}\langle k|k\rangle\boldsymbol{F}_x\langle k\rangle^T + \boldsymbol{F}_v\langle k\rangle\hat{\boldsymbol{V}}\boldsymbol{F}_v\langle k\rangle^T$$

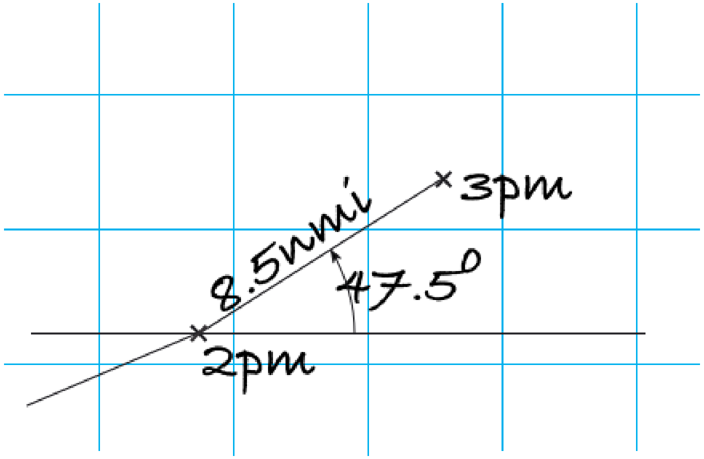
With no observations, uncertainty grows over time...

Observations



<u>Observations:</u>

– range and bearing of a landmark



Observations: $\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}_v, \boldsymbol{x}_f, \boldsymbol{w})$

$$\boldsymbol{z} = \begin{pmatrix} \sqrt{(y_i - y_v)^2 + (x_i - x_v)^2} \\ \tan^{-1}(y_i - y_v)/(x_i - x_v) - \theta_v \end{pmatrix} + \begin{pmatrix} w_r \\ w_\beta \end{pmatrix}$$

range

bearing

$$\begin{pmatrix} w_r \\ w_\beta \end{pmatrix} \sim N(0, W), \quad W = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\beta^2 \end{pmatrix}$$

Observations

Landmarks
(i.e. features of the env't)

Observations:

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}_v, \boldsymbol{x}_f, \boldsymbol{w})$$

$$\boldsymbol{z}\langle k \rangle = \hat{\boldsymbol{h}} + \boldsymbol{H}_x\left(\boldsymbol{x}\langle k \rangle - \hat{\boldsymbol{x}}\langle k \rangle\right) + \boldsymbol{H}_w \boldsymbol{w}\langle k \rangle$$

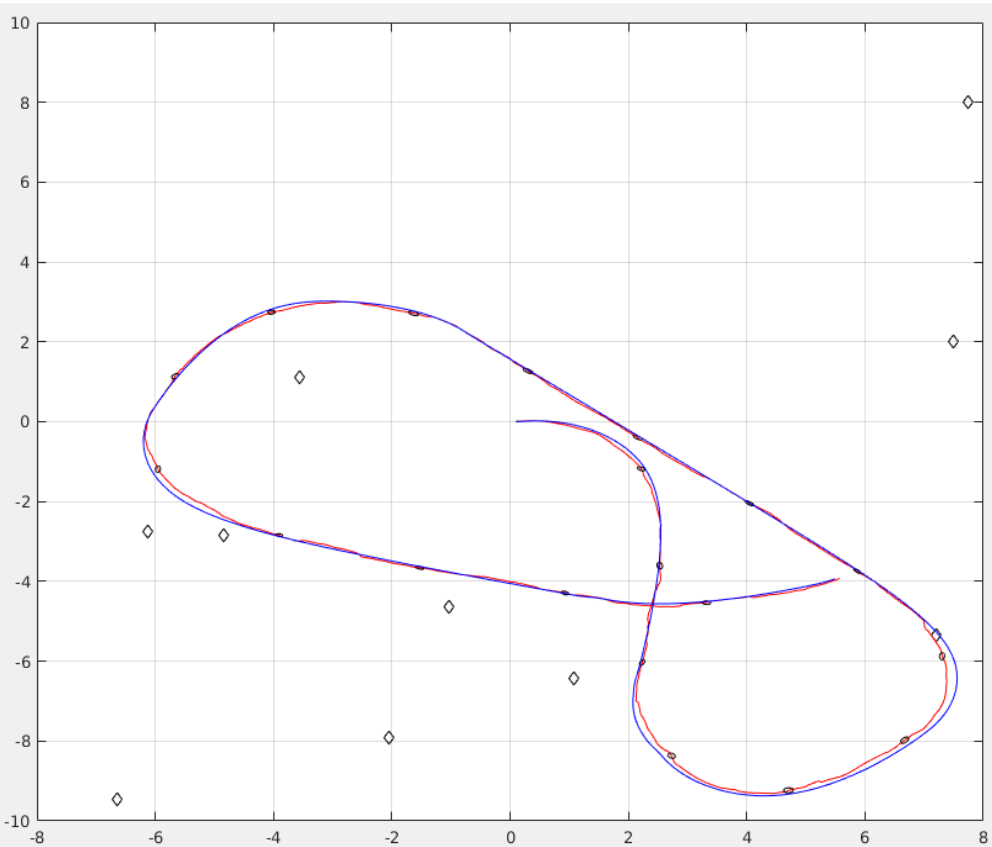
where:

$$\boldsymbol{H}_{x_i} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_v}\right|_{w=0} = \begin{pmatrix} -\dfrac{x_i - x_v\langle k \rangle}{r} & -\dfrac{y_i - y_v\langle k \rangle}{r} & 0 \\ \dfrac{x_i - x_v\langle k \rangle}{r^2} & -\dfrac{y_i - y_v\langle k \rangle}{r^2} & -1 \end{pmatrix}$$

$$\boldsymbol{H}_w = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{w}}\right|_{w=0} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} w_r \\ w_\beta \end{pmatrix} \sim N(0, \boldsymbol{W}) \qquad \boldsymbol{W} = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\beta^2 \end{pmatrix}$$

EKF Mobile Robot Localization



Process Update:

Observation Update:

$$\hat{\boldsymbol{x}}\langle k+1|k+1\rangle = \hat{\boldsymbol{x}}\langle k+1|k\rangle + \boldsymbol{K}\langle k+1\rangle \nu\langle k+1\rangle$$

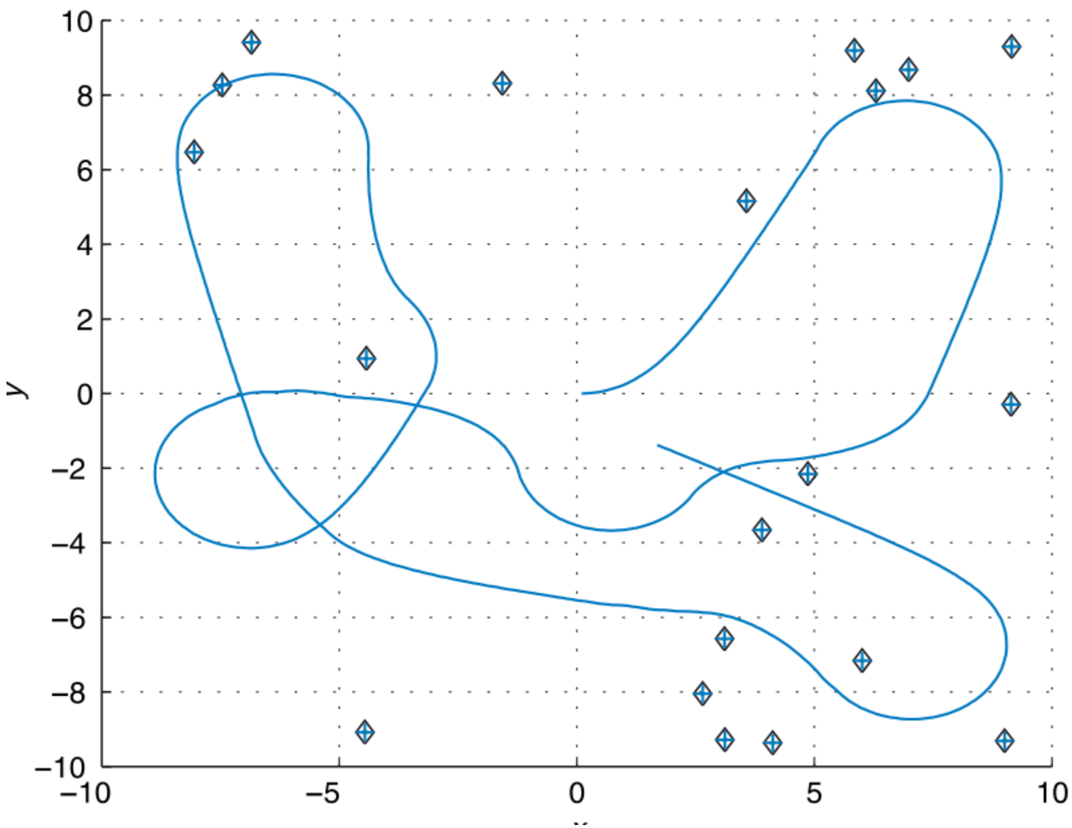
$$\hat{\boldsymbol{P}}\langle k+1|k+1\rangle = \hat{\boldsymbol{P}}\langle k+1|k\rangle \boldsymbol{F}_x\langle k\rangle^T - \boldsymbol{K}\langle k+1\rangle \boldsymbol{H}_x\langle k+1\rangle \hat{\boldsymbol{P}}\langle k+1|k\rangle$$

$$\boldsymbol{\nu}\langle k+1\rangle = \boldsymbol{z}\langle k+1\rangle - \boldsymbol{h}\left(\hat{x}(k+1|k), x_f, 0\right)$$

$$\boldsymbol{S}\langle k+1\rangle = \boldsymbol{H}_x\langle k+1\rangle \hat{\boldsymbol{P}}\langle k+1|k\rangle \boldsymbol{H}_x\langle k+1\rangle^T + \boldsymbol{H}_w\langle k+1\rangle \hat{\boldsymbol{W}}\langle k+1\rangle \boldsymbol{H}_w\langle k+1\rangle^T$$

$$\boldsymbol{K}\langle k+1\rangle = \hat{\boldsymbol{P}}\langle k+1|k\rangle \boldsymbol{H}_x\langle k+1\rangle^T \boldsymbol{S}\langle k+1\rangle^{-1}$$

Mapping using the EKF

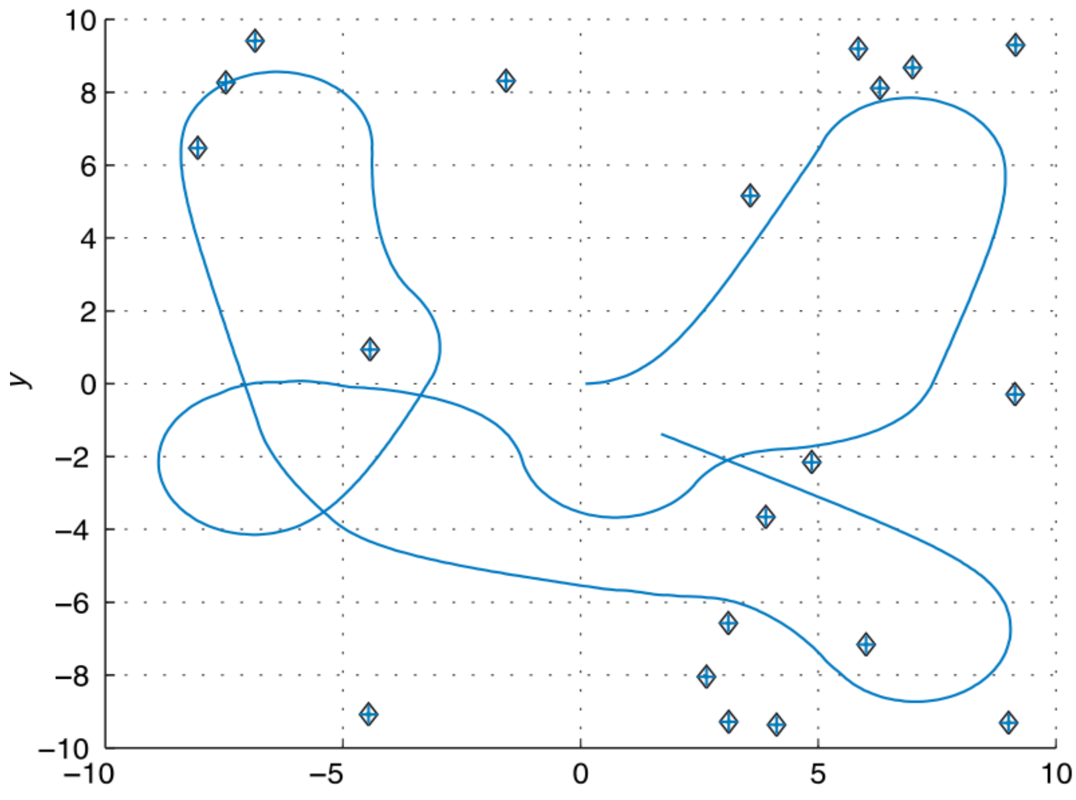


How do we use the EKF to estimate landmark positions?

State: $\hat{\boldsymbol{x}} = (x_1, y_1, x_2, y_2, \cdots x_M, y_M)^T$

Positions of each of the M landmarks (base frame)

Mapping using the EKF



<u>Process update (no new detections):</u>

$$\hat{\boldsymbol{x}}\langle k+1|k\rangle = \hat{\boldsymbol{x}}\langle k|k\rangle$$

$$\hat{\boldsymbol{P}}\langle k+1|k\rangle = \hat{\boldsymbol{P}}\langle k|k\rangle$$

<u>Process update (new detections):</u>

$$\boldsymbol{x}\langle k|k\rangle^{*} = \begin{pmatrix} \boldsymbol{x}\langle k|k\rangle \\ \boldsymbol{g}\left(\boldsymbol{x}_{v}\langle k|k\rangle, \boldsymbol{z}\langle k\rangle\right) \end{pmatrix}$$

$$\hat{\boldsymbol{P}}\langle k|k\rangle^{*} = \boldsymbol{Y}_{z} \begin{pmatrix} \hat{\boldsymbol{P}}\langle k|k\rangle & 0 \\ 0 & \hat{\boldsymbol{W}} \end{pmatrix} \boldsymbol{Y}_{z}^{T}$$

est position of new landmark

covariance of new landmark

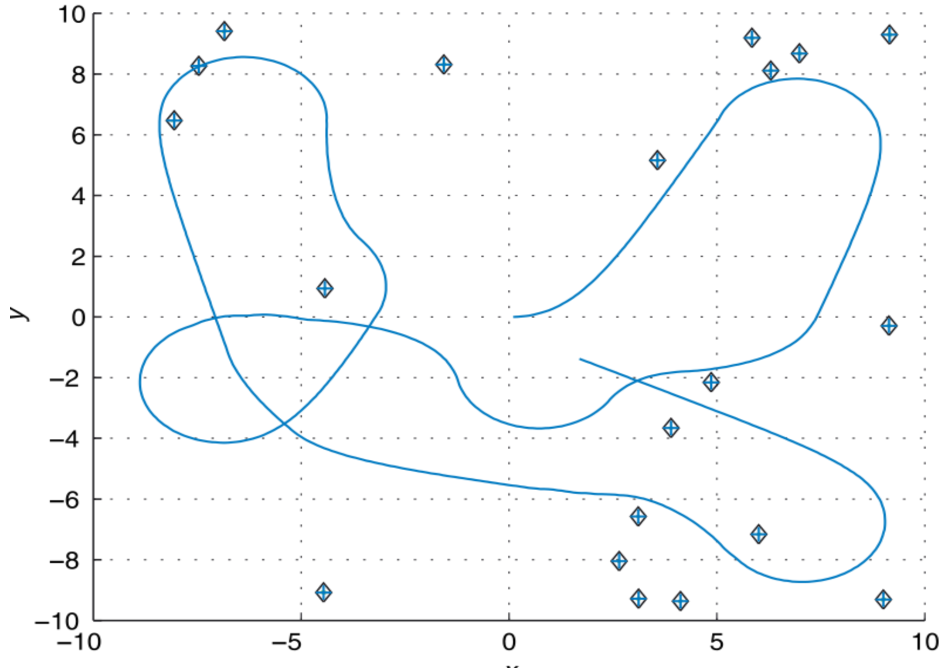
where:

$$\boldsymbol{Y}_{z} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}} = \begin{pmatrix} \boldsymbol{I}_{n\times n} & & \boldsymbol{0}_{n\times 2} \\ \boldsymbol{G}_{x} & \boldsymbol{0}_{2\times n-3} & \boldsymbol{G}_{z} \end{pmatrix}$$

$$\boldsymbol{G}_{x} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_{v}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{G}_{z} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}} = \begin{pmatrix} \cos(\theta_{v} + \theta_{z}) & -r_{z}\sin(\theta_{v} + \theta_{z}) \\ \sin(\theta_{v} + \theta_{z}) & r_{z}\cos(\theta_{v} + \theta_{z}) \end{pmatrix}$$

Mapping using the EKF

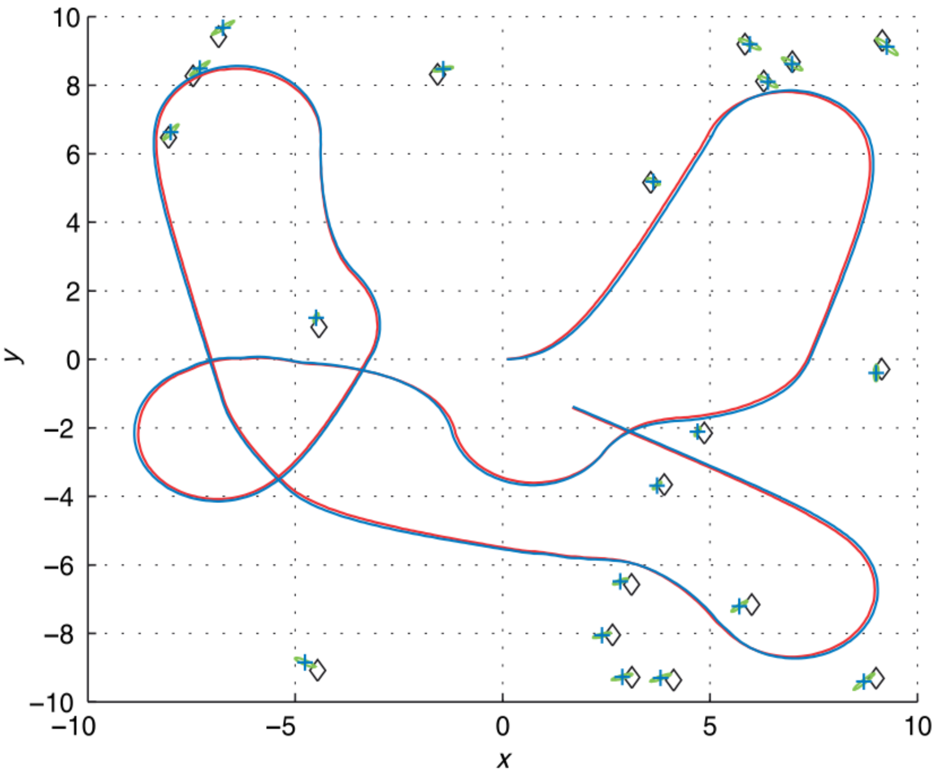


Observation update:

$$\boldsymbol{H}_x = (0 \cdots \boldsymbol{H}_{x_i} \cdots 0)$$

$$\boldsymbol{H}_{x_i} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_i} = \begin{pmatrix} \frac{x_i - x_v}{r} & \frac{y_i - y_v}{r} \\ -\frac{x_i - x_v}{r^2} & \frac{y_i - y_v}{r^2} \end{pmatrix}$$

SLAM using the EKF



Estimate both robot position and landmark positions:

$$\hat{\boldsymbol{x}} = (\underbrace{x_v, y_v, \theta_v}_{\text{robot position}}, \underbrace{x_1, y_1, x_2, y_2, \cdots x_M, y_M}_{\text{Landmark positions}})$$

robot position

Landmark positions

$$\hat{\boldsymbol{P}} = \begin{pmatrix} \hat{\boldsymbol{P}}_{vv} & \hat{\boldsymbol{P}}_{vm} \\ \hat{\boldsymbol{P}}_{vm}^T & \hat{\boldsymbol{P}}_{mm} \end{pmatrix}$$

Vehicle covariance

Vehicle/landmark covariance

Landmark covariance

SLAM using the EKF

Process update:

Vehicle portion of update

Map portion of update

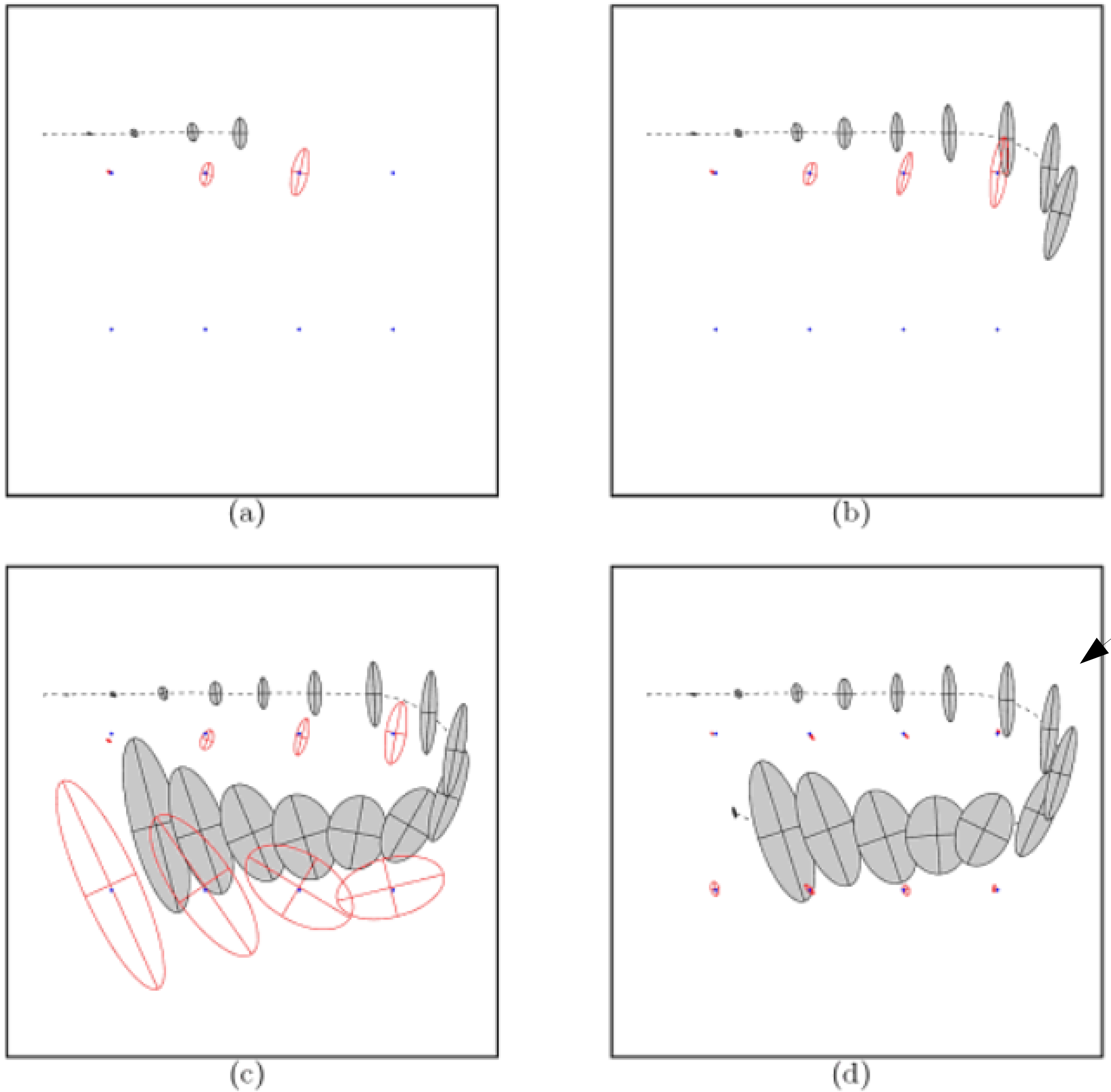
No new landmarks

$$G_x = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_v} = \begin{pmatrix} 1 & 0 & -r_z \sin(\theta_v + \theta_z) \\ 0 & 1 & r_z \cos(\theta_v + \theta_z) \end{pmatrix}$$

New landmarks

Same observation update, but using: $\boldsymbol{H}_x = (\boldsymbol{H}_{x_v} \cdots 0 \cdots \boldsymbol{H}_{x_i} \cdots 0)$

SLAM using the EKF



Landmark covariance drops significantly as soon as “loop closure” occurs.

Image: Thrun

SLAM using the EKF

$$\boldsymbol{H}_x = (\boldsymbol{H}_{x_v} \cdots 0 \cdots \boldsymbol{H}_{x_i} \cdots 0)$$

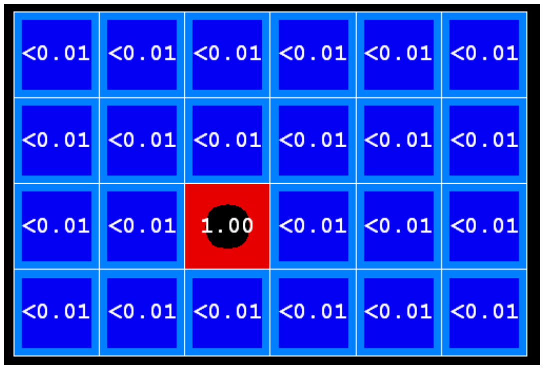
$$\boldsymbol{Y}_z = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}} = \begin{pmatrix} \boldsymbol{I}_{n\times n} & & \boldsymbol{0}_{n\times 2} \\ \boldsymbol{G}_x & \boldsymbol{0}_{2\times n-3} & \boldsymbol{G}_z \end{pmatrix}$$

$$\boldsymbol{G}_x = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_v} = \begin{pmatrix} 1 & 0 & -r_z \sin(\theta_v + \theta_z) \\ 0 & 1 & r_z \cos(\theta_v + \theta_z) \end{pmatrix}$$

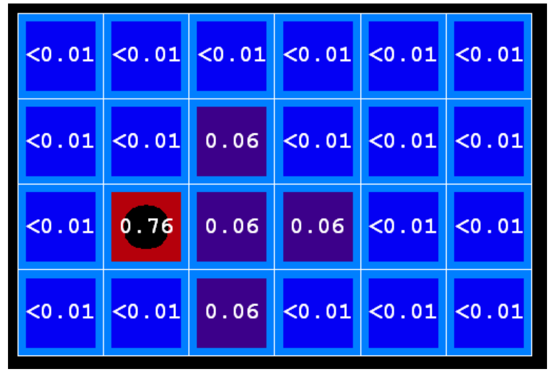
Mapping using the EKF

$$\boldsymbol{H}_x = (\boldsymbol{H}_{x_v} \cdots 0 \cdots \boldsymbol{H}_{x_i} \cdots 0)$$

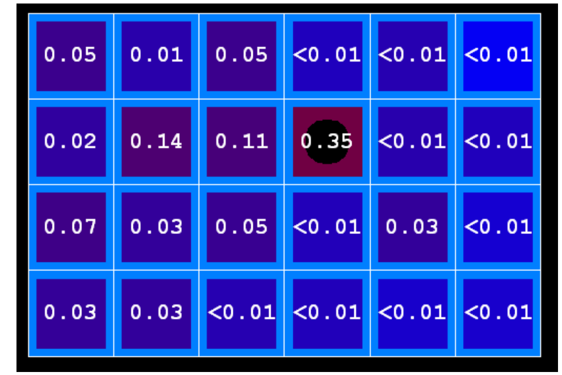
Process update



T = 1



T = 2



T = 5

Each time you execute a process update, belief gets more disbursed

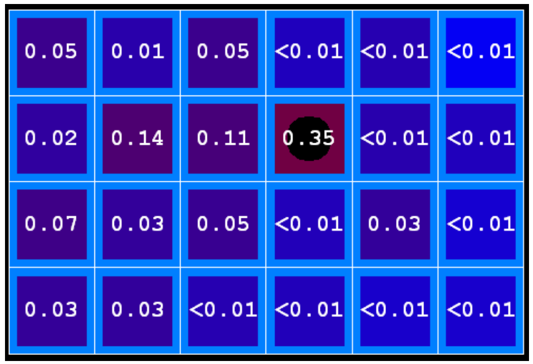
- *i.e.* Shannon entropy increases
- this makes sense: as you predict state further into the future, your uncertainty grows.

$$B'(X_{t+1}) = \sum_{X_t} P(X_{t+1}|X_t, e_{1:t}) B(X_t)$$

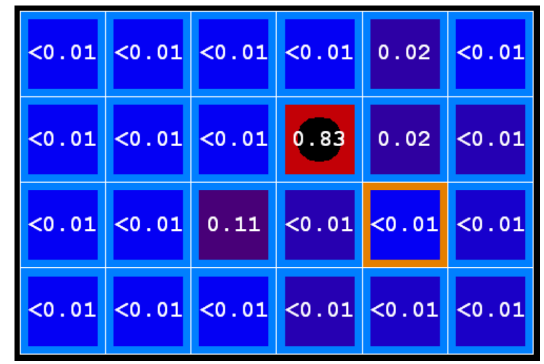
This is a little like convolution...

Images: Berkeley CS188 course notes (downloaded Summer 2015)

Observation update



Before observation



After observation

Process update *increases* uncertainty

Observation update *decreases* uncertainty
– observations give you more information

Images: Berkeley CS188 course notes (downloaded Summer 2015)