Robotic 3D Vision

Lecture 3: Probabilistic State Estimation – Filtering

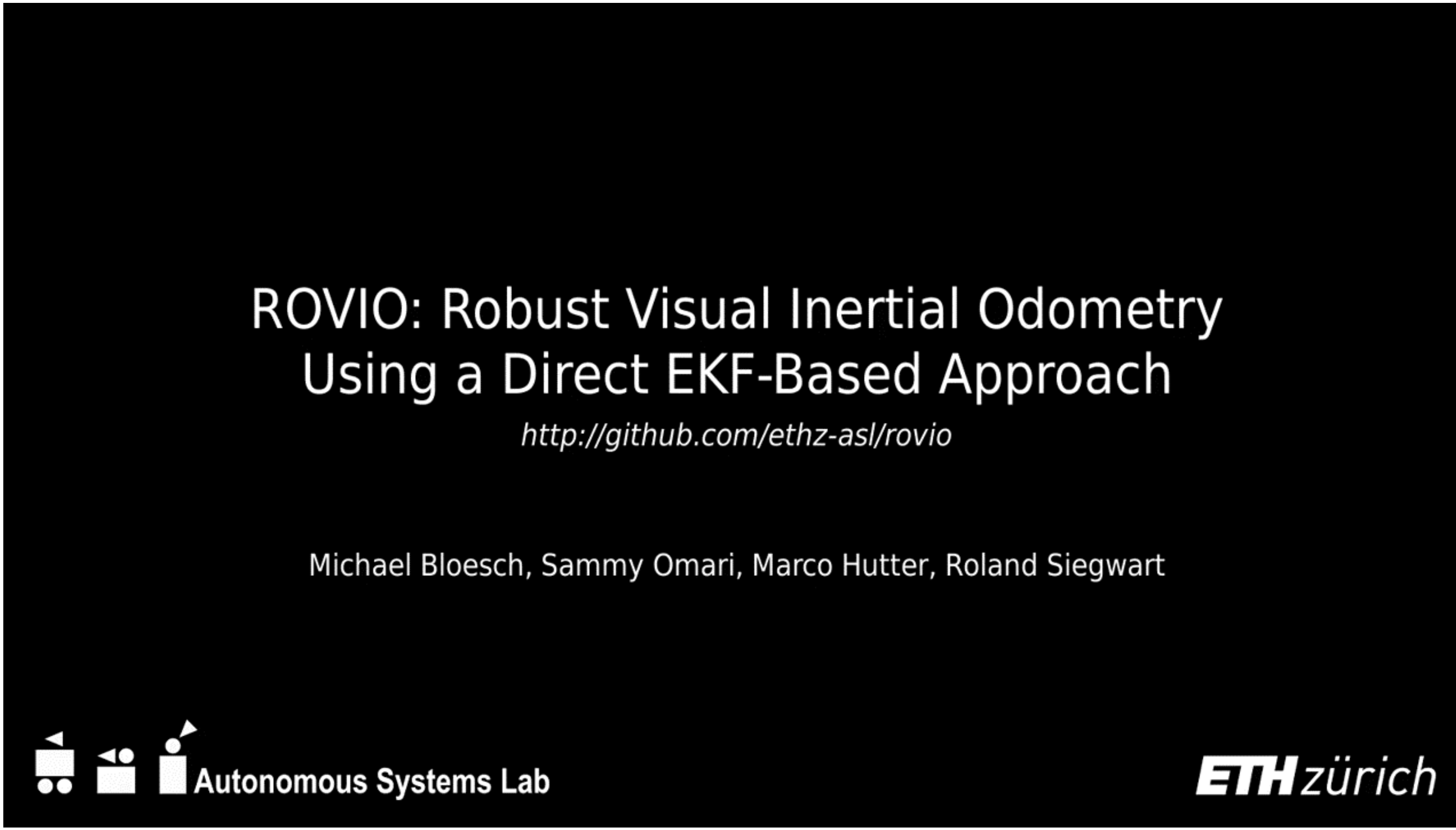
Prof. Dr. Jörg Stückler
Computer Vision Group, TU Munich
http://vision.in.tum.de

What We Will Cover Today

- Probabilistic modelling of state estimation problems
- Bayesian Filtering
- Kalman Filter
- Extended Kalman Filter
- Particle Filter

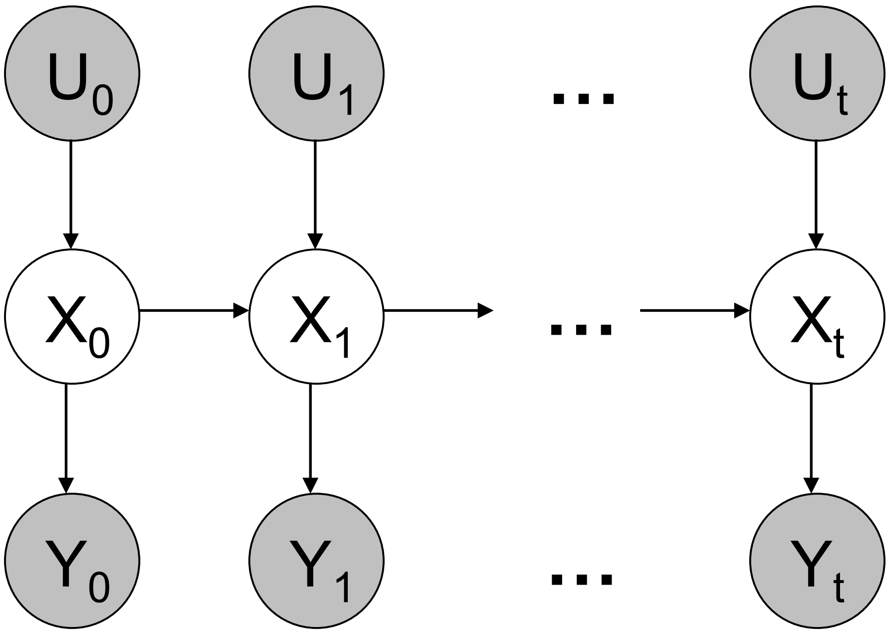
Why Probabilistic State Estimation?

Why Probabilistic State Estimation?



Probabilistic Model of Time-Sequential Processes

- Hidden state X gives rise to noisy observations Y
- At each time t,
  - the state changes stochastically from X_{t-1} to X_t
  - state change depends on action U_t
  - we get a new observation Y_t



Why Probabilistic State Estimation?

- Probabilistic modelling accounts for uncertainties
- State estimation: Inference in probabilistic model
- Cope with noisy state transitions and observations
- Maintain uncertainty in the state estimate
- Principled approaches to update the state estimate distribution based on probability theory

Recursive Bayesian Filtering

- Our goal: recursively estimate probability distribution of state X_t given all observations seen so far and previous estimate for X_{t-1}

- We assume
  - Knowledge about probability distribution of observations

$$p\left(Y_t \middle| X_{0:t}, U_{0:t}, Y_{0:t-1}\right)$$

  - Knowledge about probabilistic dynamics of state transitions

$$p\left(X_t \middle| X_{0:t-1}, U_{0:t}\right)$$

  - Estimate of initial state $p\left(X_0\right)$

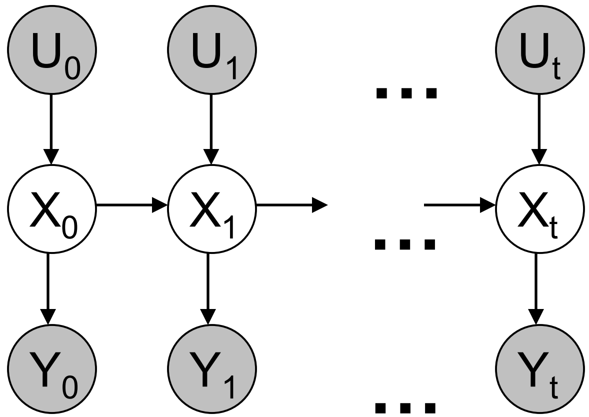
Markov Assumptions

- Only the immediate past matters for a state transition

$$p\left(X_t \mid X_{0:t-1}, U_{0:t}\right) = \boxed{p\left(X_t \mid X_{t-1}, U_t\right)}$$ **state transition model**

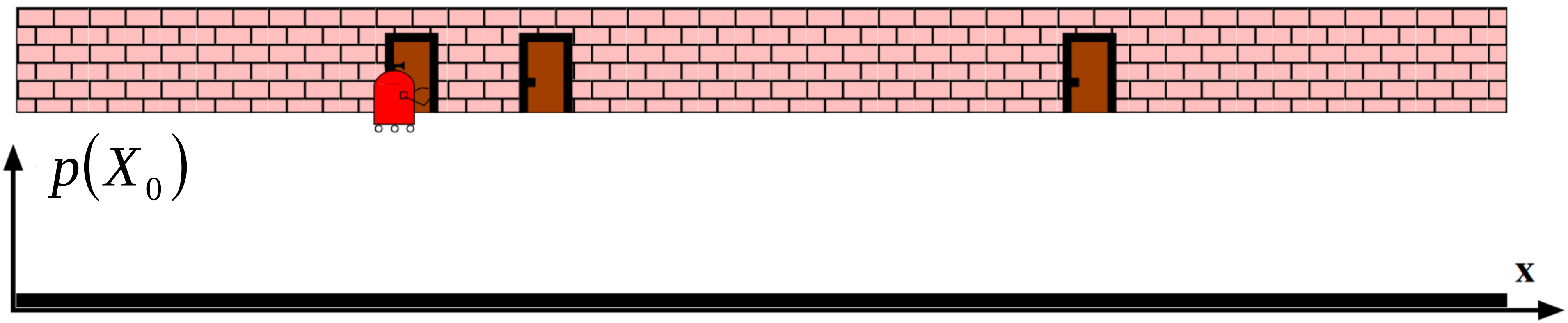
- Observations depend only on the current state

$$p\left(Y_t \mid X_{0:t}, U_{0:t}, Y_{0:t-1}\right) = \boxed{p\left(Y_t \mid X_t\right)}$$ **observation model**



The Door-Sensing Robot

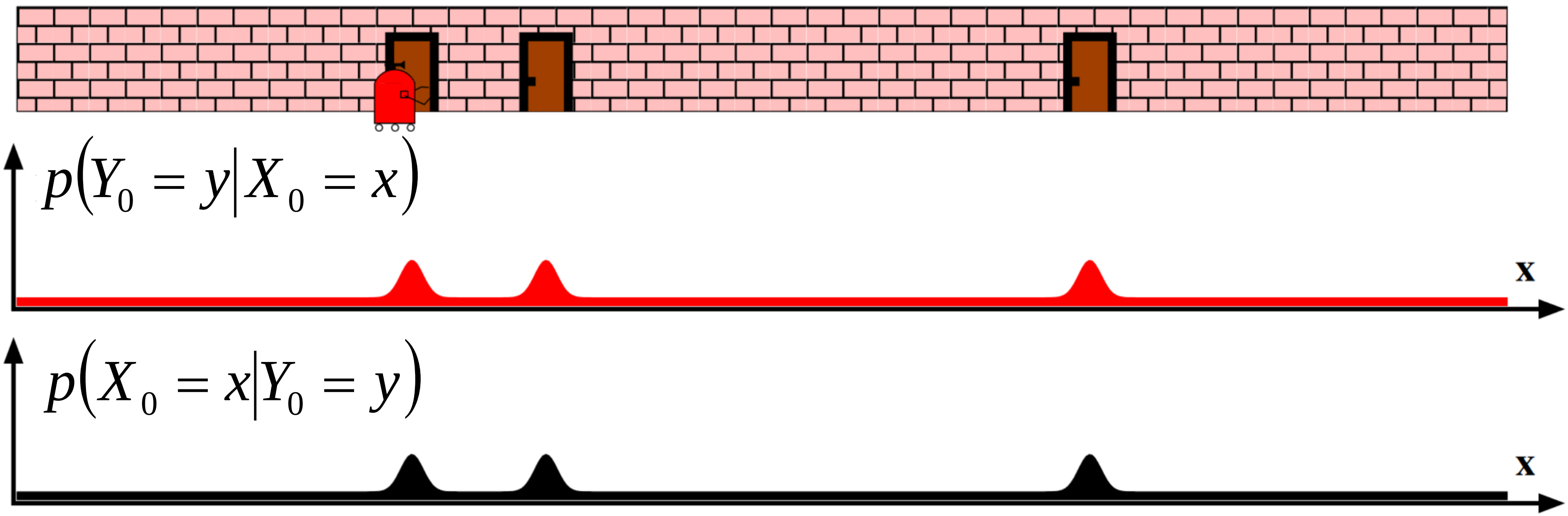
- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door



- Initially it knows nothing about its location: uniform $p(X_0)$

The Door-Sensing Robot

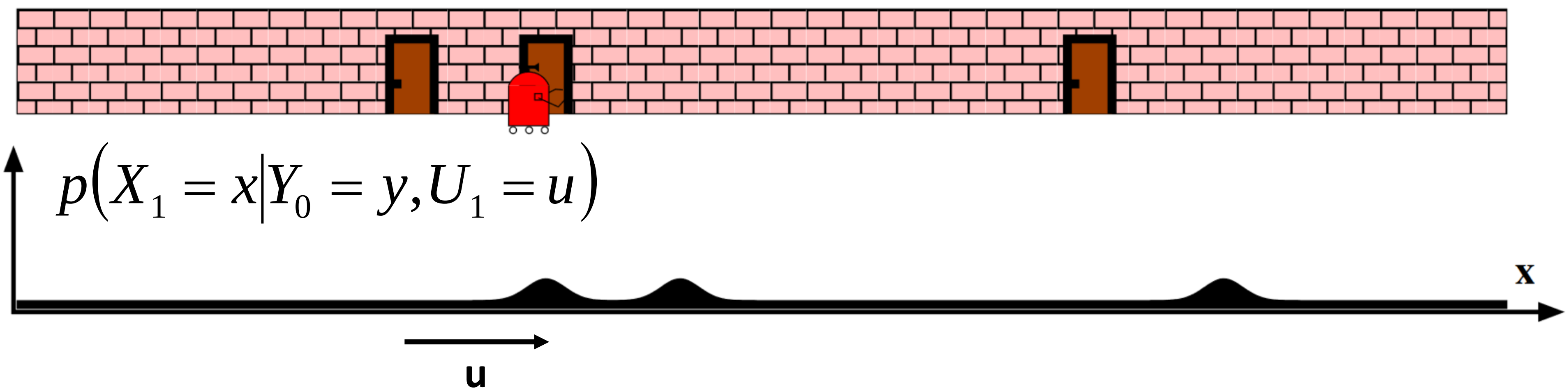
- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door



- Observation of door increases the likelihood of x at doors

The Door-Sensing Robot

- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door

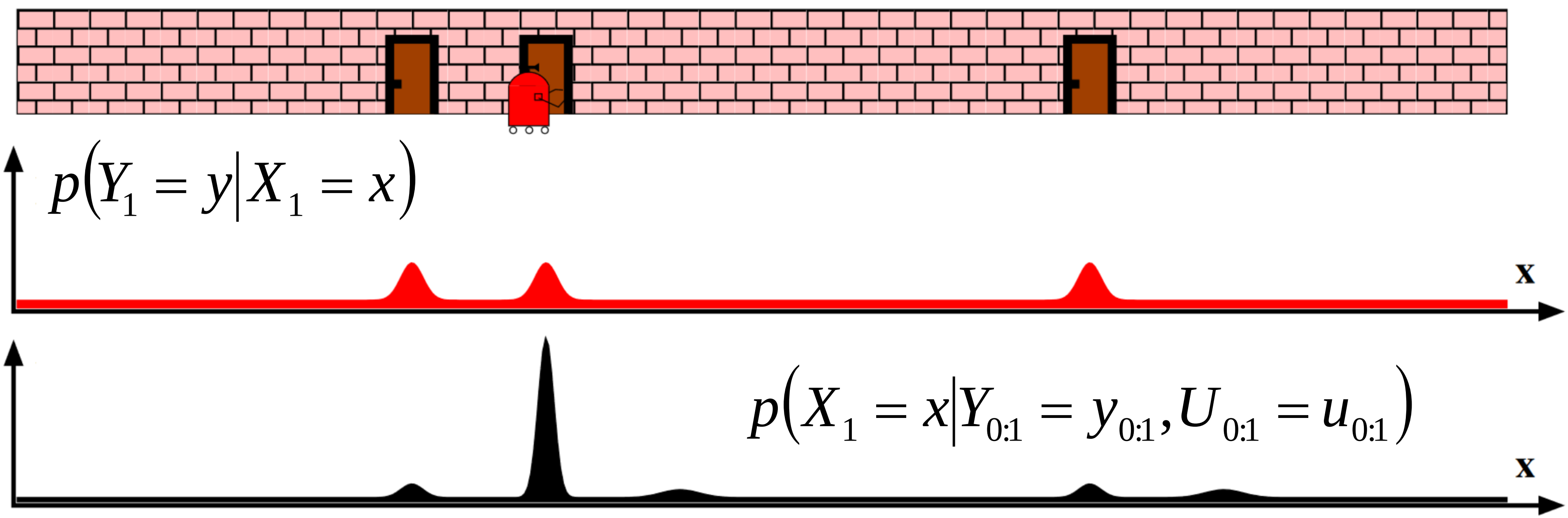


$$p\left(X_1 = x \middle| Y_0 = y, U_1 = u\right)$$

- Robot moves: state is propagated, uncertainty increases

The Door-Sensing Robot

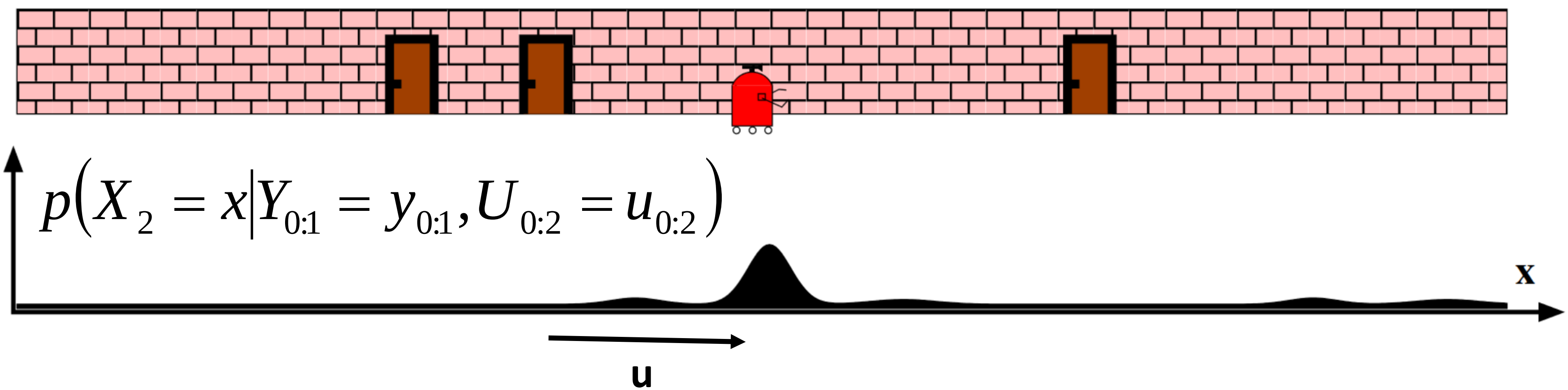
- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door



- Observation of door increases the likelihood of x at doors

The Door-Sensing Robot

- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door



- Robot moves: state is propagated, uncertainty increases

Base Case

- Assume we have initial prior that predicts state in absence of any evidence: $p\left(X_0\right)$
- At the first frame, correct this given the value of $Y_0=y_0$

$$p(X_0 \mid Y_0 = y_0) = \frac{p(y_0 \mid X_0)\, p(X_0)}{p(y_0)} \propto p(y_0 \mid X_0)\, p(X_0)$$

Posterior prob. of state given observation

Likelihood of observation

Prior of the state

Recursive State Estimation

- How to obtain $p\left(X_t \middle| y_{0:t}, u_{0:t}\right)$ from $p\left(X_{t-1} \middle| y_{0:t-1}, u_{0:t-1}\right)$?

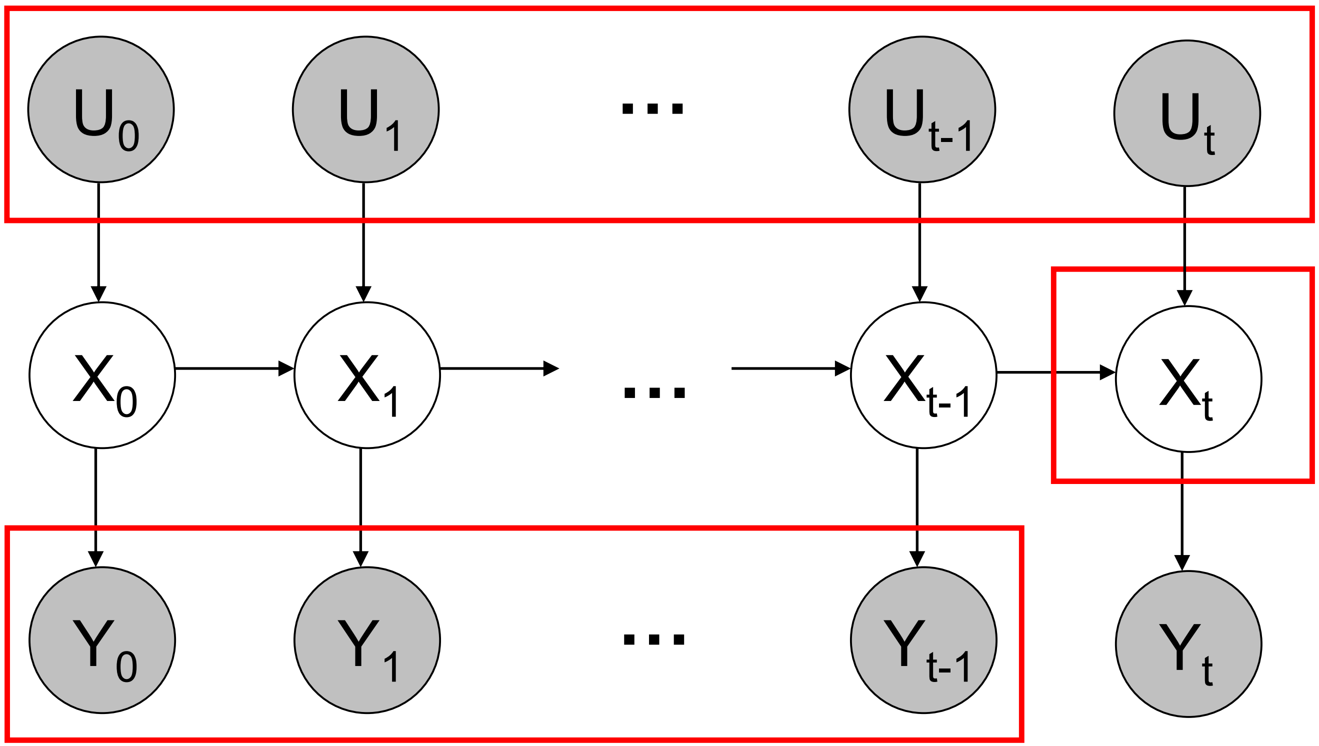
$$
\begin{aligned}
& p\left(X_t \middle| y_{0:t}, u_{0:t}\right) \\
& \quad = \frac{p(y_t \mid X_t, y_{0:t-1}, u_{0:t})\, p(X_t \mid y_{0:t-1}, u_{0:t})}{p(y_t \mid y_{0:t-1}, u_{0:t})} \\
& \quad = \frac{p(y_t \mid X_t)\, p(X_t \mid y_{0:t-1}, u_{0:t})}{p(y_t \mid y_{0:t-1}, u_{0:t})} \\
& \quad = \frac{p(y_t \mid X_t)\, \boxed{p(X_t \mid y_{0:t-1}, u_{0:t})}}{\int p(y_t \mid X_t)\, p(X_t \mid y_{0:t-1}, u_{0:t})\, dX_t}
\end{aligned}
$$

What does this term mean?

Marginalizing over X_t

Recursive State Estimation

- How to obtain $p\left(X_t \middle| y_{0:t-1}, u_{0:t}\right)$?
- Intuition: If we knew $p\left(X_{t-1} \middle| y_{0:t-1}, u_{0:t-1}\right)$, the state transition model should tell us how to propagate the state estimate



Recursive State Estimation

- How to obtain $p\left(X_t \middle| y_{0:t-1}, u_{0:t}\right)$?

$$
\begin{aligned}
& p\left(X_t \middle| y_{0:t-1}, u_{0:t}\right) \\
& \quad = \int p\left(X_t, X_{t-1} \middle| y_{0:t-1}, u_{0:t}\right) dX_{t-1} \\
& \quad = \int p\left(X_t \mid X_{t-1}, y_{0:t-1}, u_{0:t}\right) p\left(X_{t-1} \mid y_{0:t-1}, u_{0:t}\right) dX_{t-1} \\
& \quad = \int p\left(X_t \mid X_{t-1}, u_t\right) p\left(X_{t-1} \mid y_{0:t-1}, u_{0:t-1}\right) dX_{t-1}
\end{aligned}
$$

Markov assumption

Prediction and Correction

- Prediction:

$$p(X_t \mid y_{0:t-1}, u_{0:t}) = \int \underbrace{p(X_t \mid X_{t-1}, u_t)}_{\text{state transition model}} \underbrace{p(X_{t-1} \mid y_{0:t-1}, u_{0:t-1})}_{\text{corrected estimate from previous step}} dX_{t-1}$$

- Correction:

$$p(X_t \mid y_{0:t}, u_{0:t}) = \frac{\overbrace{p(y_t \mid X_t)}^{\text{observation model}} \overbrace{p(X_t \mid y_{0:t-1}, u_{0:t})}^{\text{predicted estimate}}}{\int p(y_t \mid X_t) p(X_t \mid y_{0:t-1}, u_{0:t}) dX_t}$$

Predict-Correct Cycle

- Prediction:

$$p\left(X_t \mid y_{0:t-1}, u_{0:t}\right) = \int p\left(X_t \mid X_{t-1}, u_t\right) p\left(X_{t-1} \mid y_{0:t-1}, u_{0:t-1}\right) dX_{t-1}$$

observation

y_t

action

u_t

- Correction:

$$p\left(X_t \middle| y_{0:t}, u_{0:t}\right) = \frac{p\left(y_t \mid X_t\right) p\left(X_t \mid y_{0:t-1}, u_{0:t}\right)}{\int p\left(y_t \mid X_t\right) p\left(X_t \mid y_{0:t-1}, u_{0:t}\right) dX_t}$$

Kalman Filter

- Kalman filters (KFs) instantiate recursive Bayesian filtering for a specific class of state transition and observation models

  - Linear state transition model with Gaussian noise:

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{d_t})$$

  - Linear observation model with Gaussian noise:

$$\mathbf{z}_t = \mathbf{C}_t \mathbf{x}_t + \boldsymbol{\delta} \qquad \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{m_t})$$

  - Gaussian initial state estimate: $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$

Kalman Filter Prediction & Correction

- Efficient closed-form correction and prediction steps which involve manipulation of Gaussians
- The state estimate can be represented as a Gaussian distribution

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

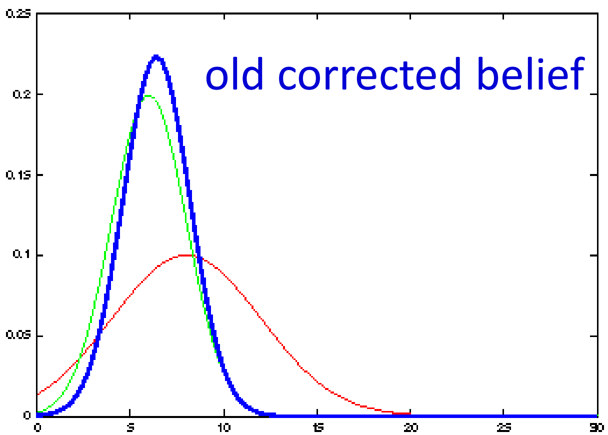
- Prediction: $$\boldsymbol{\mu}_t^- = \mathbf{A}_t \boldsymbol{\mu}_{t-1}^+ + \mathbf{B}_t \mathbf{u}_t$$ $$\boldsymbol{\Sigma}_t^- = \mathbf{A}_t \boldsymbol{\Sigma}_{t-1}^+ \mathbf{A}_t^\top + \boldsymbol{\Sigma}_{d_t}$$

- Correction: $$\mathbf{K}_t = \boldsymbol{\Sigma}_t^- \mathbf{C}_t^\top \left(\mathbf{C}_t \boldsymbol{\Sigma}_t^- \mathbf{C}_t^\top + \boldsymbol{\Sigma}_{m_t} \right)^{-1}$$ **Kalman gain** $$\boldsymbol{\mu}_t^+ = \boldsymbol{\mu}_t^- + \mathbf{K}_t (\mathbf{y}_t - \mathbf{C}_t \boldsymbol{\mu}_t^-)$$ $$\boldsymbol{\Sigma}_t^+ = (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t) \boldsymbol{\Sigma}_t^-$$

Kalman Filter 1D Example

- Let's make a 1D example
- Prediction: $\mu_t^- = a_t \mu_{t-1}^+ + b_t u_t$ **shifted mean**

$$(\sigma_t^-)^2 = a_t^2 (\sigma_{t-1}^+)^2 + \sigma_{d_t}^2$$ **scaled variance + noise**



KF

prediction

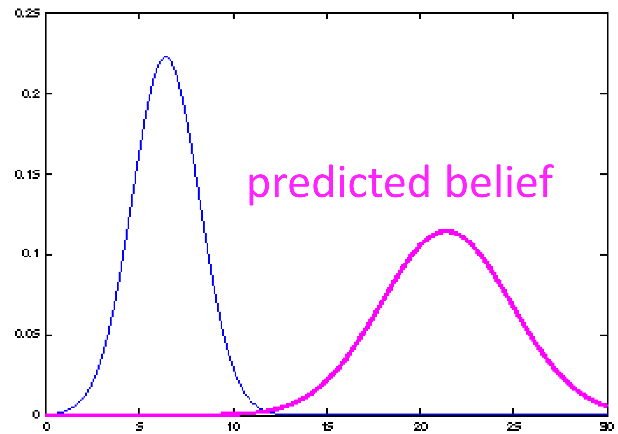


Image courtesy: Thrun, Burgard, Fox 2005

Kalman Filter 1D Example

- Let's make a 1D example
- Correction:

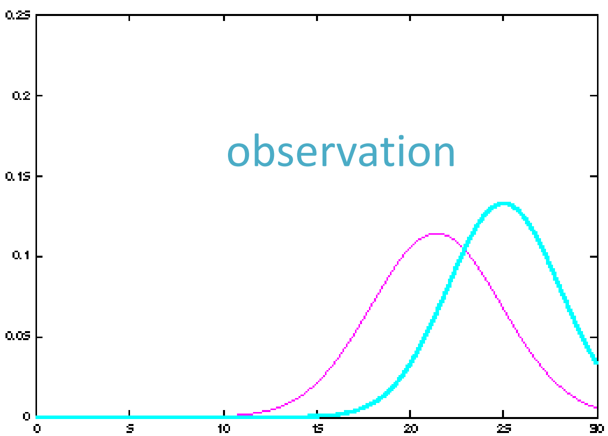
$$k_t = \frac{c_t(\sigma_t^-)^2}{c_t^2(\sigma_t^-)^2 + \sigma_{m_t}^2}$$

weighted mean

$$\mu_t^+ = \mu_t^- + k_t(y_t - c_t\mu_t^-) = \frac{\sigma_{m_t}^2 \mu_t^- + c_t^2(\sigma_t^-)^2 y_t}{\sigma_{m_t}^2 + c_t^2(\sigma_t^-)^2}$$

$$(\sigma_t^+)^2 = (\sigma_t^-)^2 - k_t c_t (\sigma_t^-)^2 = \frac{\sigma_{m_t}^2 (\sigma_t^-)^2}{\sigma_{m_t}^2 + c_t^2(\sigma_t^-)^2}$$

obs. noise determines update strength



KF

correction

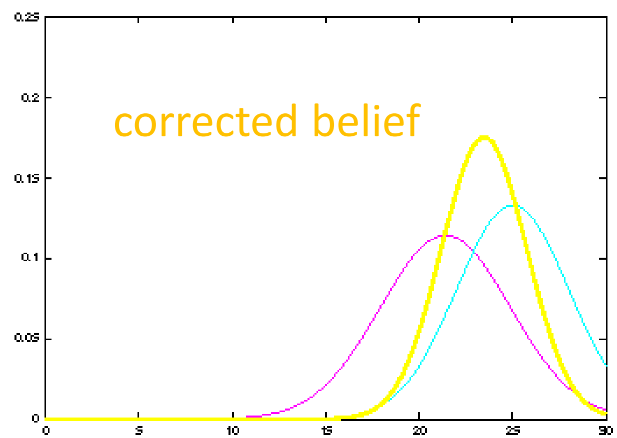


Image courtesy: Thrun, Burgard, Fox 2005

Kalman Filter Properties

- Highly efficient: Polynomial in measurement dimensionality *k* and state dimensionality *n*: $O(k^{2.376} + n^2)$
- Optimal for linear Gaussian systems!
- In robotic vision, most models are non-linear!

Extended Kalman Filter (EKF)

- Non-linear state-transition model with Gaussian noise:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{d_t})$$

- Non-linear observation model with Gaussian noise:

$$\mathbf{y}_t = h(\mathbf{x}_t) + \boldsymbol{\delta}_t \qquad \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{m_t})$$

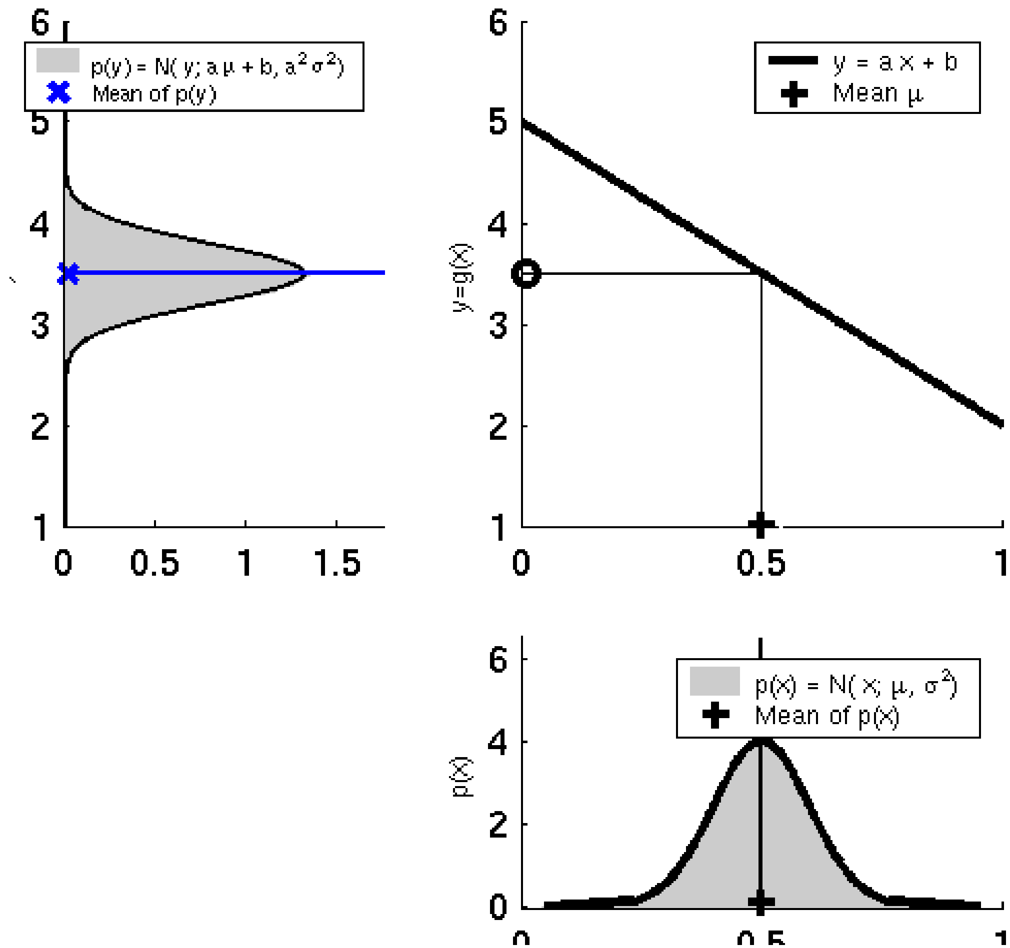
- How to cope with non-linear system?

- Idea: linearize the models in each time step

$$\Rightarrow \quad \mathbf{x}_t \approx g(\mathbf{x}^0_{t-1}, \mathbf{u}_t) + \nabla g(\mathbf{x}, \mathbf{u}_t)\big|_{\mathbf{x}=\mathbf{x}^0_{t-1}} \left(\mathbf{x}_{t-1} - \mathbf{x}^0_{t-1}\right) + \boldsymbol{\epsilon}_t$$

$$\Rightarrow \quad \mathbf{y}_t \approx h(\mathbf{x}^0_t) + \nabla h(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}^0_t} \left(\mathbf{x}_t - \mathbf{x}^0_t\right) + \boldsymbol{\delta}_t$$

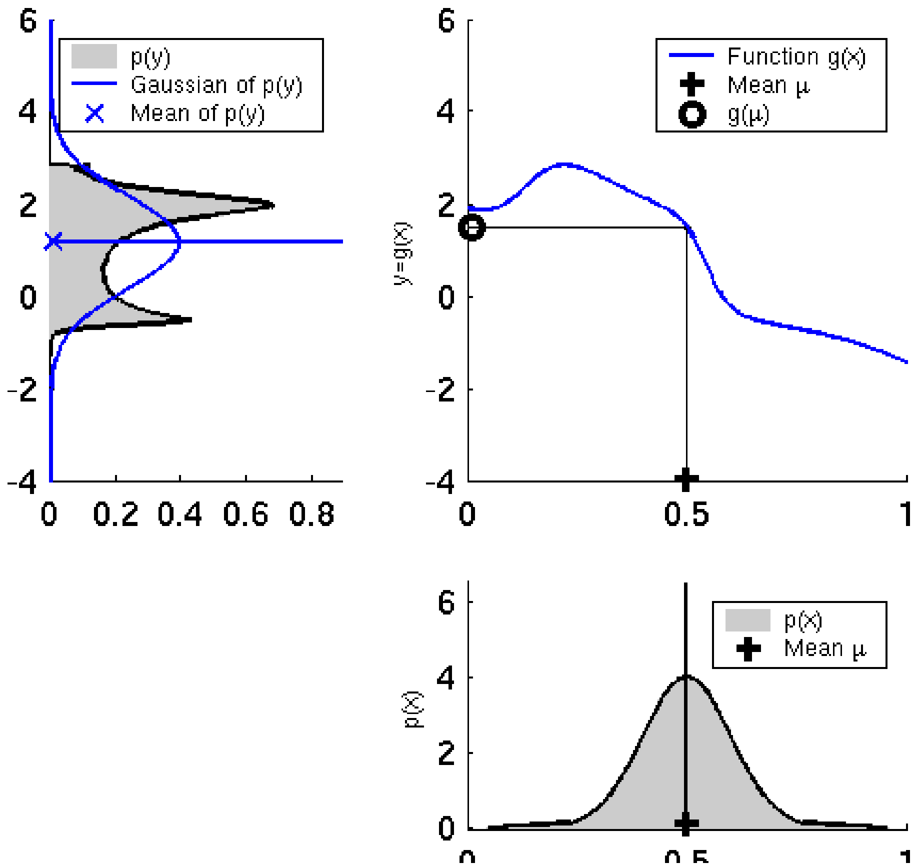
Gaussian Propagation for Linear Models



- Gaussians propagate exactly through a linear function

Image courtesy: Thrun, Burgard, Fox 2002

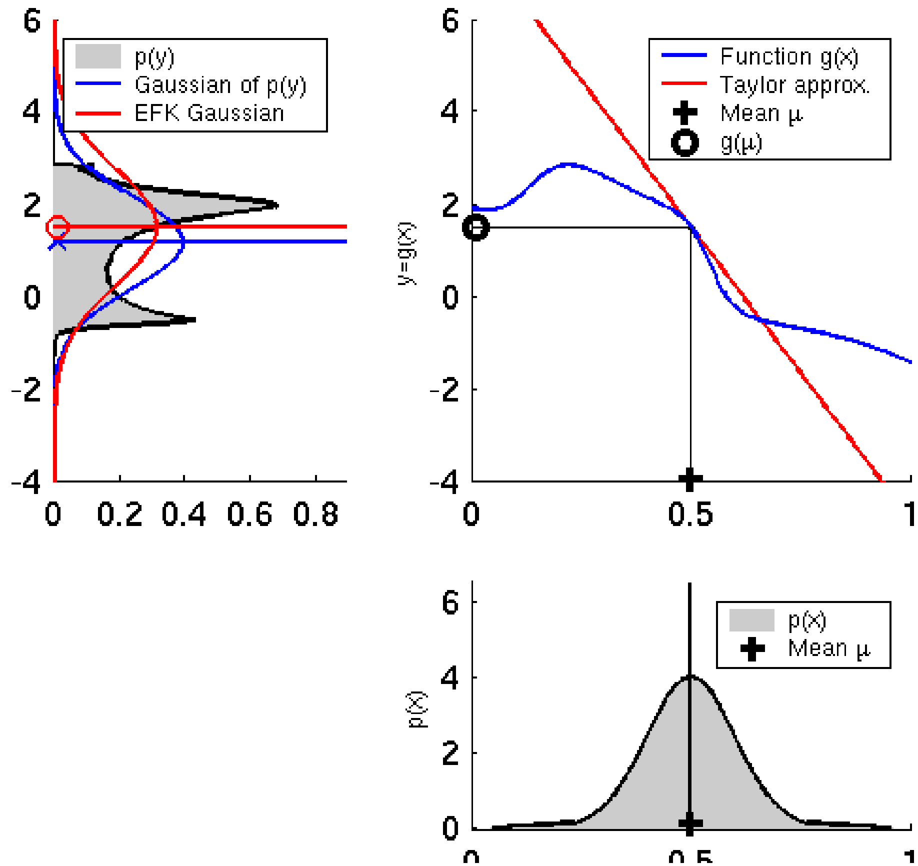
Gaussian Propagation for Non-Linear Models



- Gaussian state can be coarse approximation in non-linear system

Image courtesy: Thrun, Burgard, Fox 2002

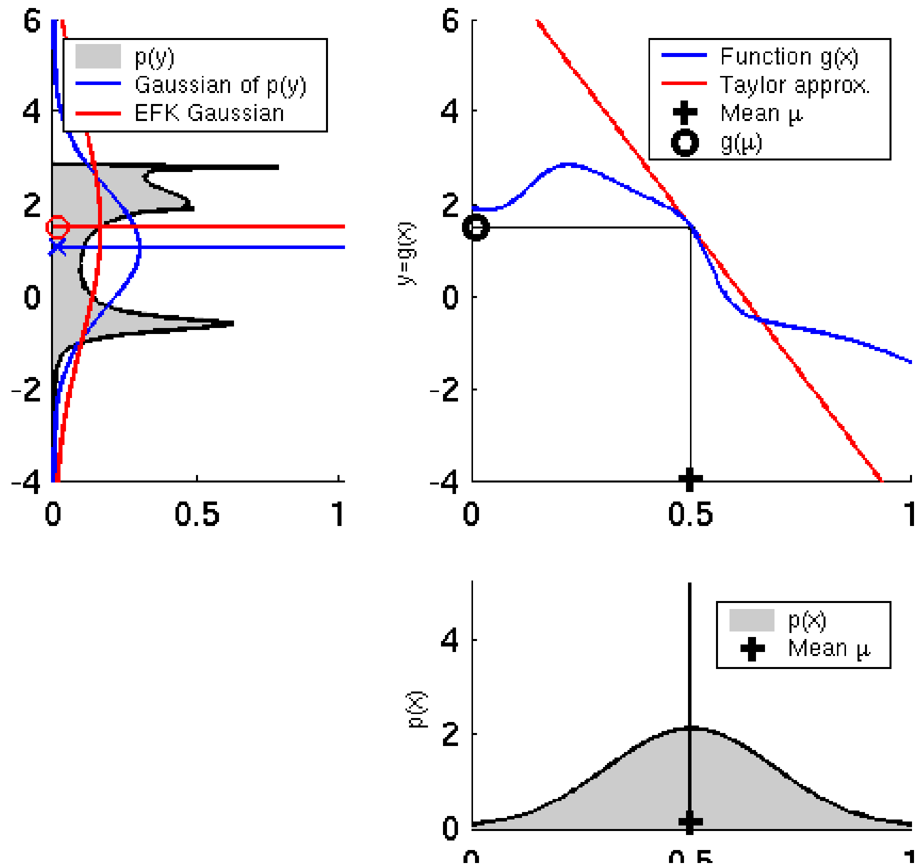
EKF Linearization



- Gaussian propagation through non-linear function can introduce bias from best approximating Gaussian

Image courtesy: Thrun, Burgard, Fox 2002

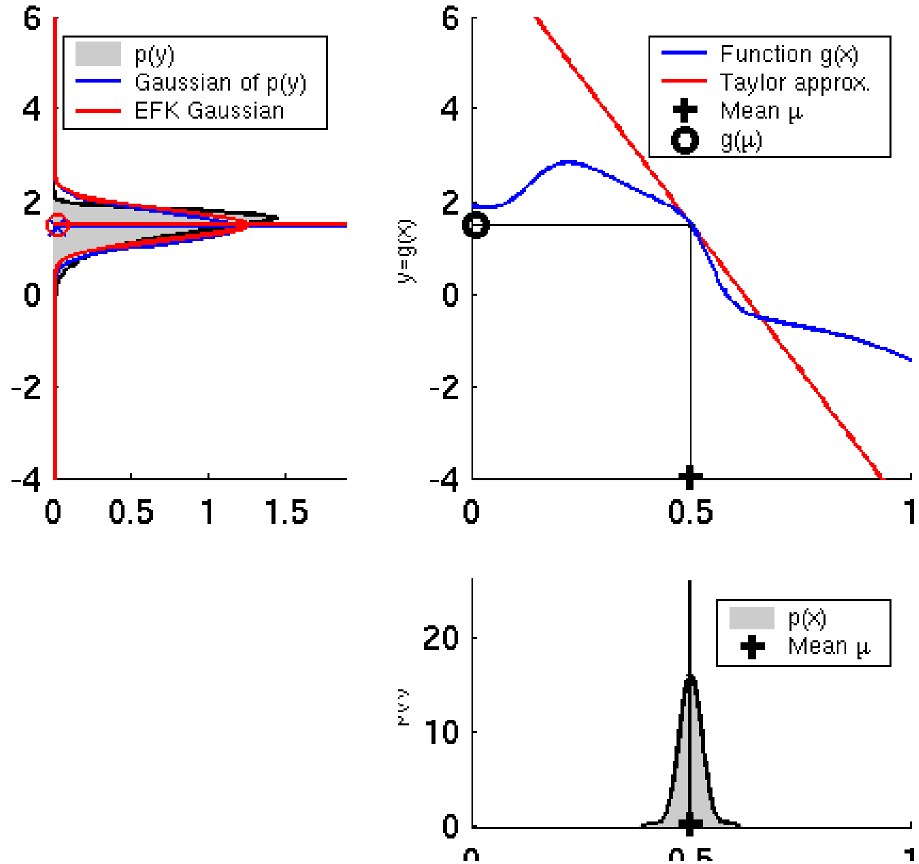
EKF Linearization



- The larger the uncertainty, the larger errors are introduced

Image courtesy: Thrun, Burgard, Fox 2002

EKF Linearization



- Good approximation when propagated probability mass covers a local regime that is close to linear

Image courtesy: Thrun, Burgard, Fox 2002

EKF Prediction & Correction

- Efficient approximate correction and prediction steps which involve manipulation of Gaussians and linearization
- The state estimate can be represented as a Gaussian distribution

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

- Prediction: $\boldsymbol{\mu}_t^- = g(\boldsymbol{\mu}_{t-1}^+, \mathbf{u}_t)$

$$\boldsymbol{\Sigma}_t^- = \mathbf{G}_t \boldsymbol{\Sigma}_{t-1}^+ \mathbf{G}_t^\top + \boldsymbol{\Sigma}_{d_t} \qquad \mathbf{G}_t := \nabla g(\mathbf{x}, \mathbf{u}_t)\big|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}^+}$$

- Correction: $\mathbf{K}_t = \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top \left(\mathbf{H}_t \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top + \boldsymbol{\Sigma}_{m_t}\right)^{-1}$

$$\boldsymbol{\mu}_t^+ = \boldsymbol{\mu}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - h(\boldsymbol{\mu}_t^-)\right) \qquad \mathbf{H}_t := \nabla h(\mathbf{x})\big|_{\mathbf{x}=\boldsymbol{\mu}_t^-}$$

$$\boldsymbol{\Sigma}_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \boldsymbol{\Sigma}_t^-$$

Extended Kalman Filter Properties

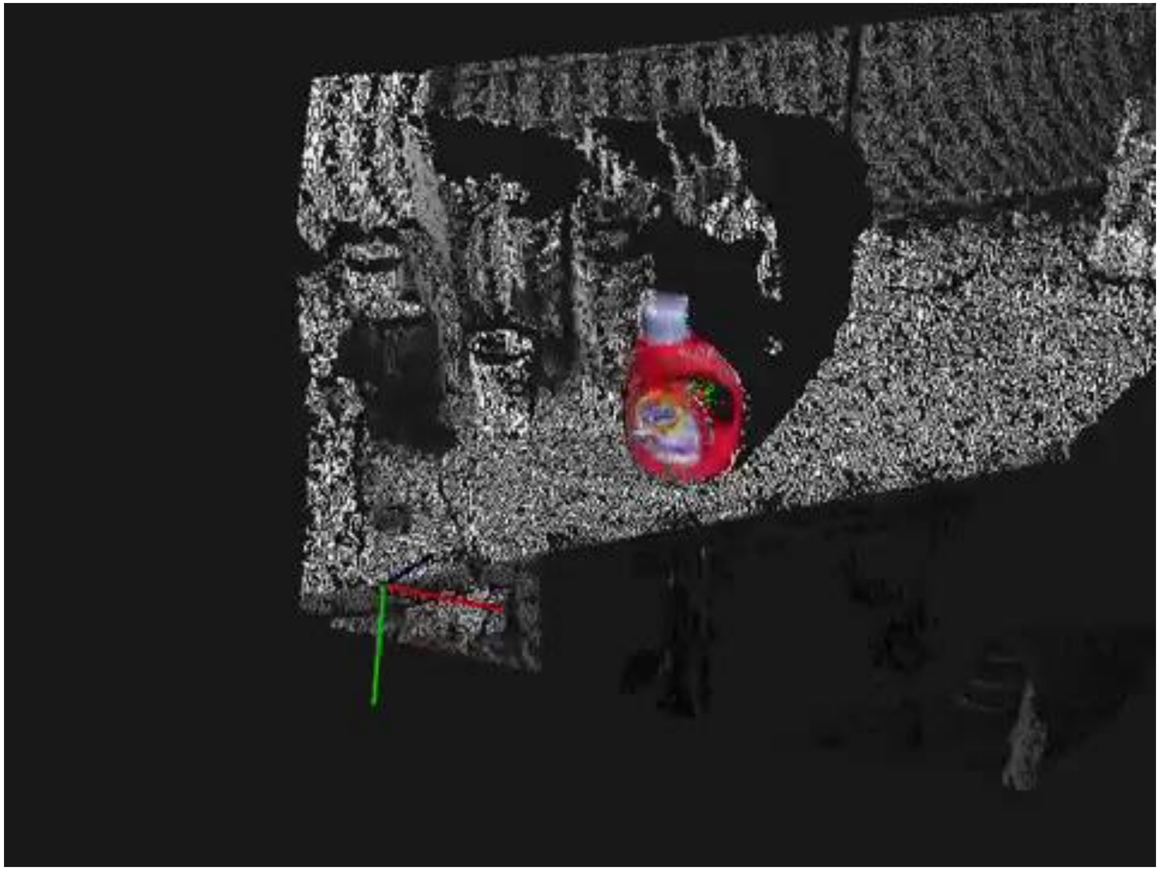
- Still highly efficient: Polynomial in measurement dimensionality k and state dimensionality n: $O(k^{2.376} + n^2)$

- No optimality guarantees!

- Linearization can be problematic for highly non-linear models
  - Different variant: Unscented Kalman Filter (UKF)
  - Idea: propagate samples through non-linearity and recover a better Gaussian approximation (second-order approximation)

What is a Particle Filter?

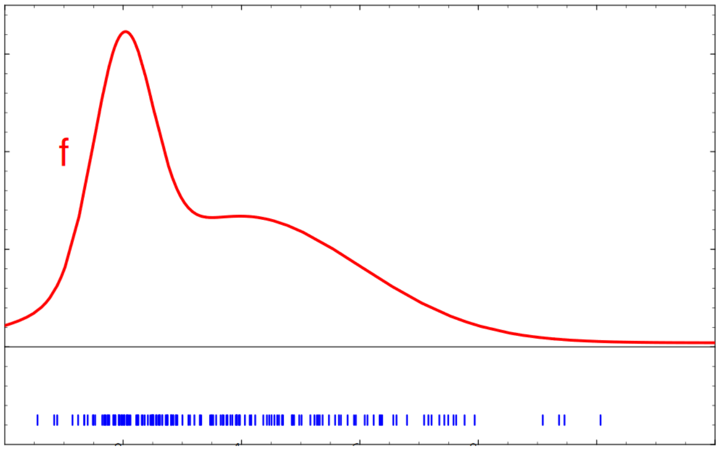
- Gaussians are restrictive for state and noise modelling
- Idea: represent the state estimate by random samples

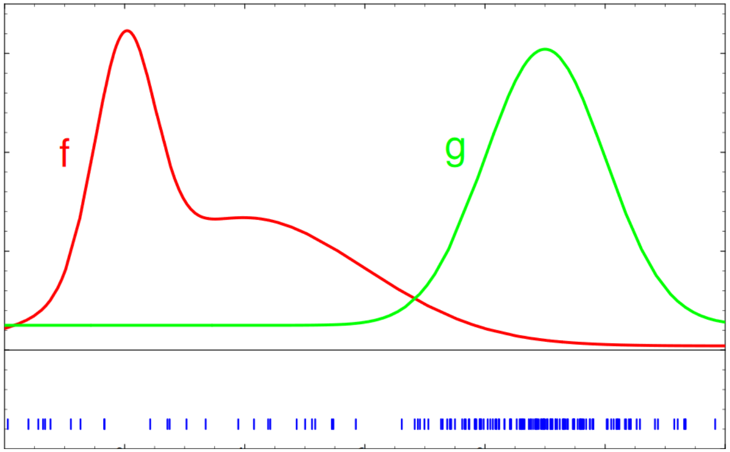


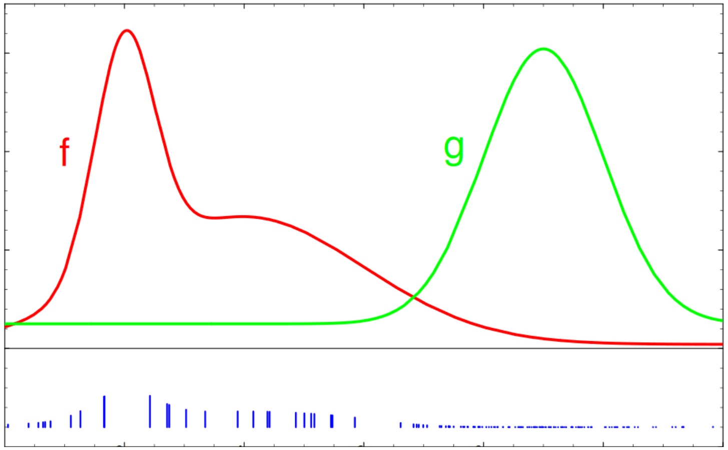
Video: Choi et al., 2013

Importance Sampling Concept

- A key concept in particle filters is importance sampling
  - We would like to draw samples from a distribution f







  - However, we can only draw from a different distribution g
  - Weight samples of g by f(x)/g(x)

Image courtesy: Thrun, Burgard, Fox 2005

Importance Sampling

- Objective: Evaluate expectation of a function $f(\mathbf{z})$ w.r.t. a probability function $p(\mathbf{z})$
- Use a proposal distribution $q(\mathbf{z})$ from which it is easy to draw samples and which is close in shape to $p(\mathbf{z})$
- Approximate expectation by a finite sum over samples from $q(\mathbf{z})$

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})d\mathbf{z}$$

$$\simeq \frac{1}{L}\sum_{l=1}^{L}\frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}f(\mathbf{z}^{(l)})$$

- With importance weights

$$w_l = \frac{p(\mathbf{z}^l)}{q(\mathbf{z}^l)}$$

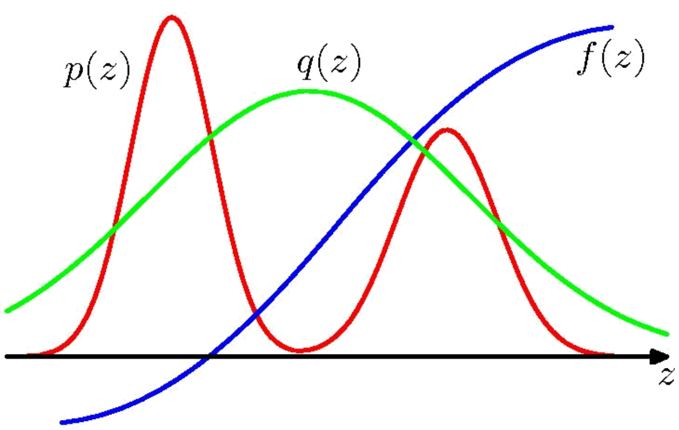
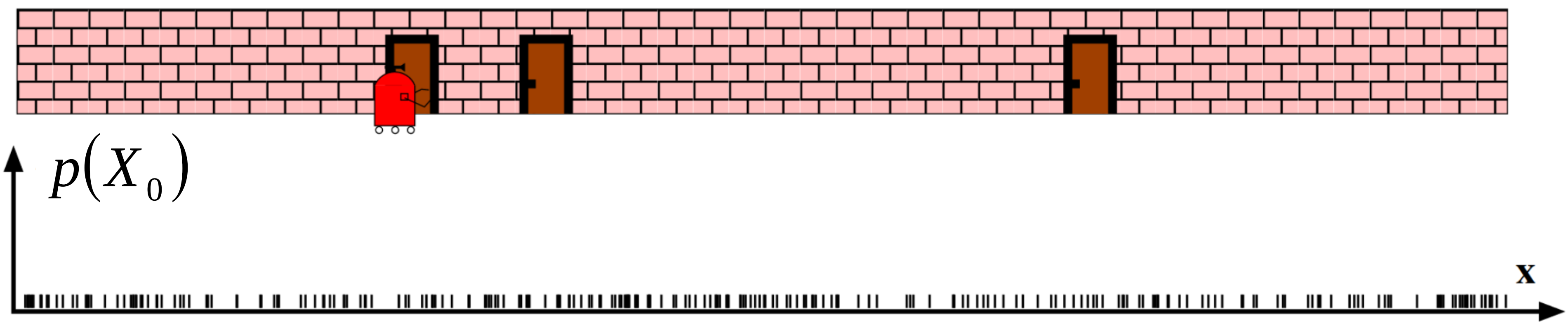


Image courtesy: Bishop 2006

The Door-Sensing Robot Resampled

- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door

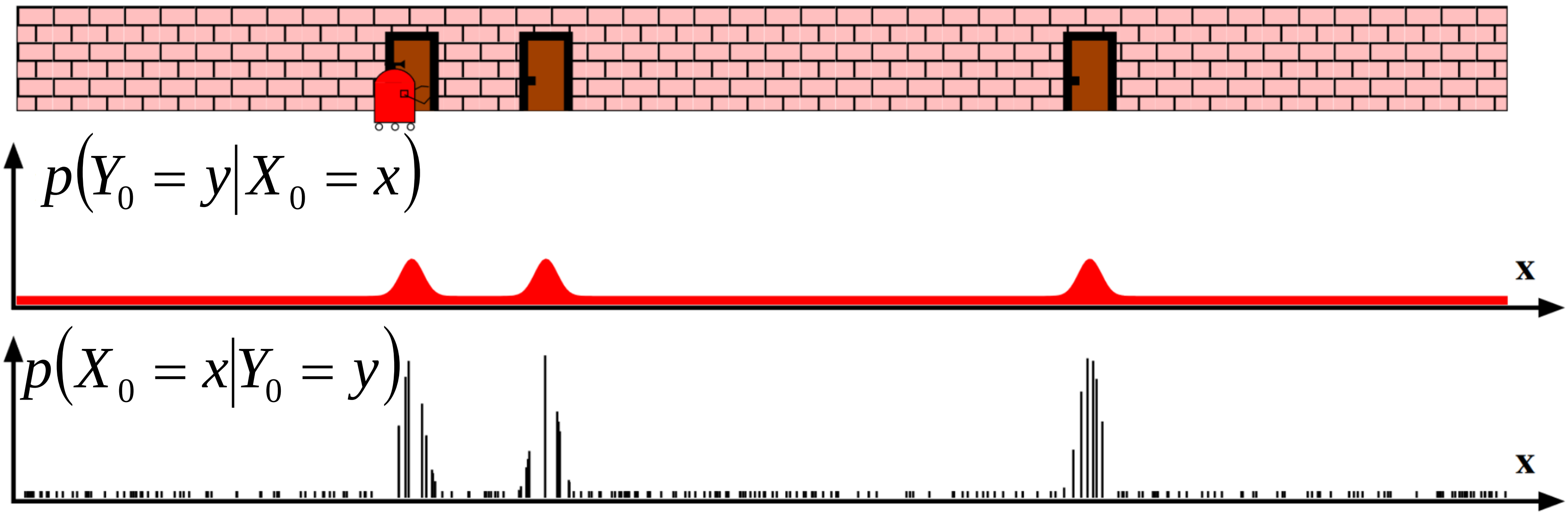


- Initially it knows nothing about its location: uniform $p(X_0)$

Image courtesy: Thrun, Burgard, Fox 2005

The Door-Sensing Robot

- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door

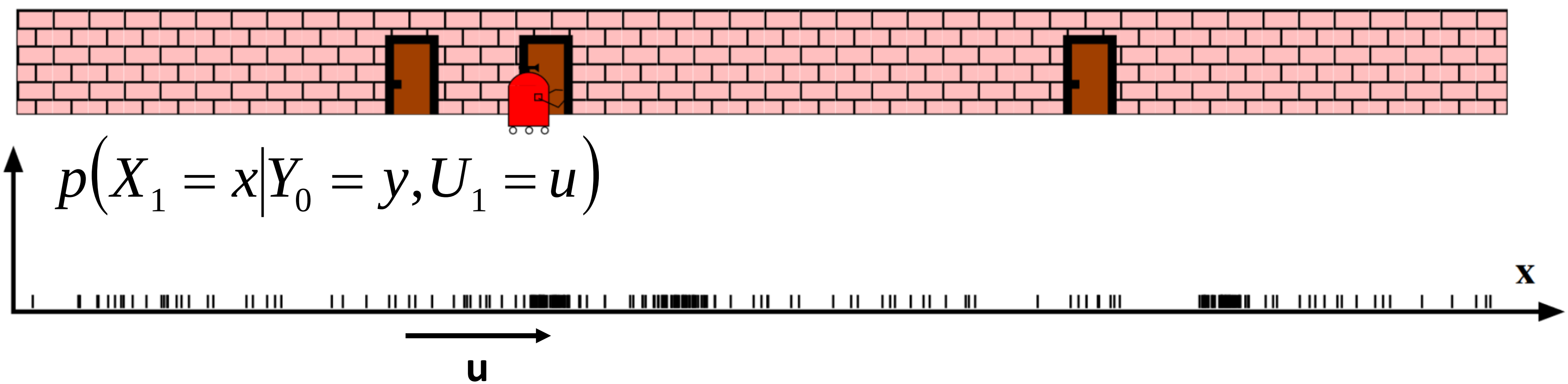


- Observation of door increases the likelihood of x at doors

Image courtesy: Thrun, Burgard, Fox 2005

The Door-Sensing Robot

- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door



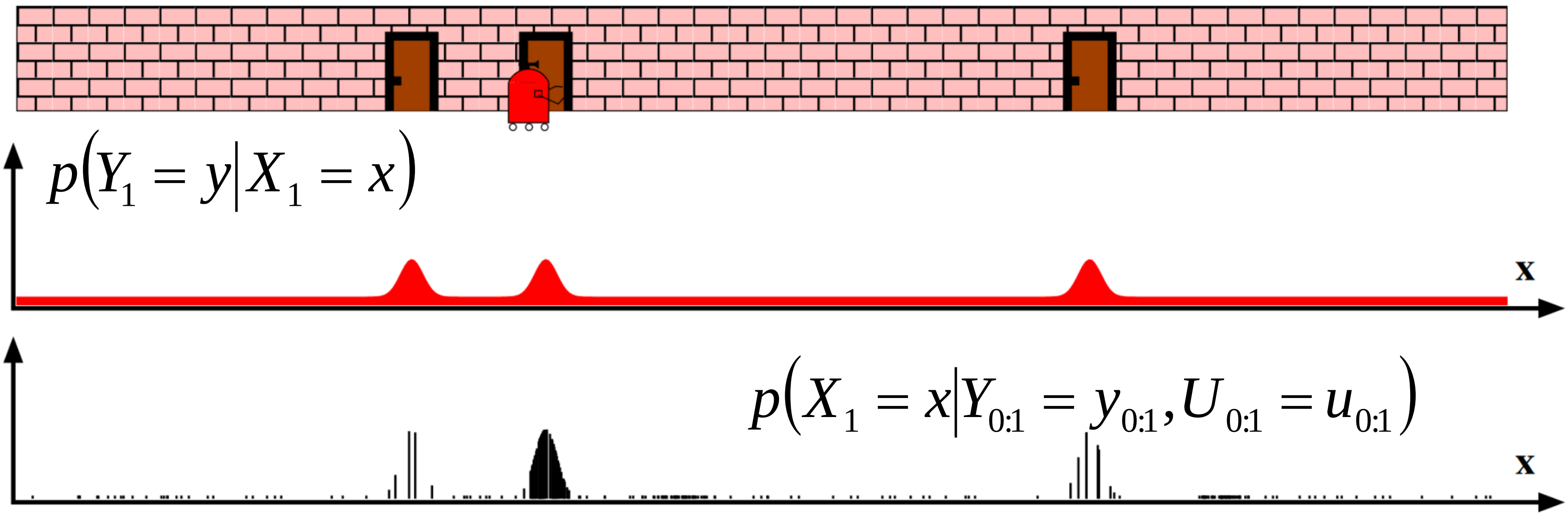
$$p\left(X_1 = x \middle| Y_0 = y, U_1 = u\right)$$

- Robot moves: state is propagated, uncertainty increases
- Samples are resampled and propagated

Image courtesy: Thrun, Burgard, Fox 2005

The Door-Sensing Robot

- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door

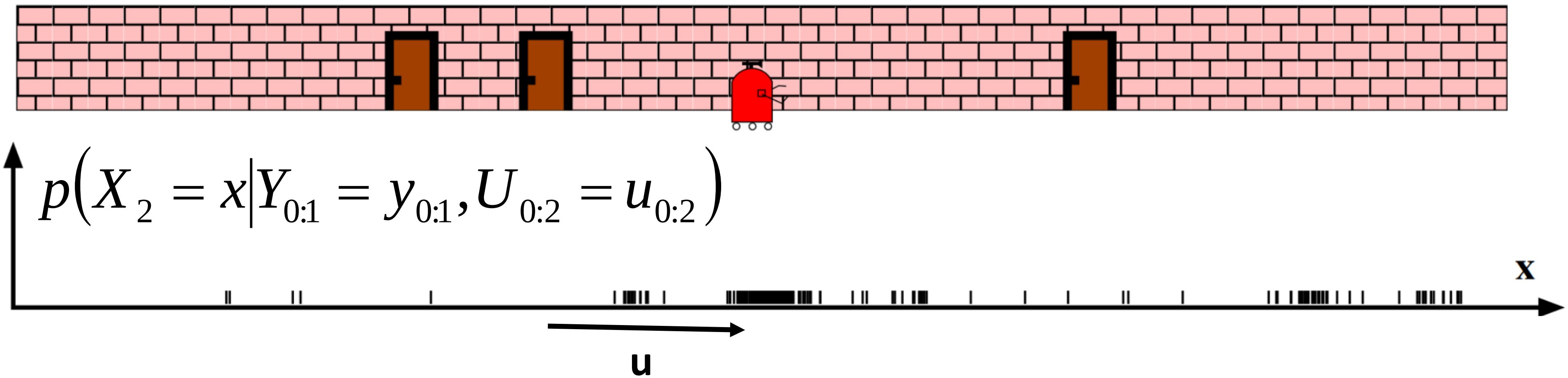


- Observation of door increases the likelihood of x at doors

Image courtesy: Thrun, Burgard, Fox 2005

The Door-Sensing Robot

- Our robot wants to localize itself along the corridor
- It can detect when it is in front of a door



$$p\left(X_2 = x \middle| Y_{0:1} = y_{0:1}, U_{0:2} = u_{0:2}\right)$$

- Robot moves: state is propagated, uncertainty increases
- Samples are resampled and propagated

Image courtesy: Thrun, Burgard, Fox 2005

Particle Filter (PF)

- Non-linear observation and state-transition distributions

$$p(\mathbf{y}_t \mid \mathbf{x}_t) \qquad p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$$

- State estimate (full posterior!) represented as a set of weighted samples

$$\left\{ \mathbf{x}_{0:t}^i, w_t^i \right\}_{i=1}^N$$

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{0:t}, \mathbf{u}_{1:t}) \approx \sum_{i=1}^N w_t^i \delta_{\mathbf{x}_{0:t}^i}(\mathbf{x}_{0:t})$$

- The weighted samples a.k.a. particles are propagated and updated over time to approximate the full posterior

Sequential Importance Sampling (SIS)

$$\mathbb{E}(f(X_{0:t})) = \int_{X_{0:t}} f(\mathbf{x}_{0:t})\, p(\mathbf{x}_{0:t} \mid \mathbf{y}_{0:t}, \mathbf{u}_{1:t})\, d\mathbf{x}_{0:t}$$
$$= \int_{X_{0:t}} f(\mathbf{x}_{0:t})\, \frac{p(\mathbf{x}_{0:t} | \mathbf{y}_{0:t}, \mathbf{u}_{1:t})}{q(\mathbf{x}_{0:t} | \mathbf{y}_{0:t}, \mathbf{u}_{1:t})}\, q(\mathbf{x}_{0:t} \mid \mathbf{y}_{0:t}, \mathbf{u}_{1:t})\, d\mathbf{x}_{0:t}$$

- Sequential update:
  - Particle update: $\mathbf{x}_t^i \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^i, \mathbf{y}_t, \mathbf{u}_t)$
  - Weight update: $w_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t | \mathbf{x}_t^i)\, p(\mathbf{x}_t^i | \mathbf{x}_{t-1}^i, \mathbf{u}_t)}{q(\mathbf{x}_t | \mathbf{x}_{t-1}^i, \mathbf{y}_t, \mathbf{u}_t)}$

SIS Algorithm

- At each time step t:

$$\eta = 0$$

$$\textbf{for} \quad i = 1{:}N$$

$$\mathbf{x}_t^i \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^i, \mathbf{y}_t, \mathbf{u}_t)$$

$$w_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t|\mathbf{x}_t^i)\, p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i,\mathbf{u}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i,\mathbf{y}_t,\mathbf{u}_t)}$$

$$\eta = \eta + w_t^i$$

$$\textbf{end}$$

$$\textbf{for} \quad i = 1{:}N$$

$$w_t^i = w_t^i / \eta$$

$$\textbf{end}$$

Choice of Proposal Distribution

- If we choose the state transition model as proposal distribution, we obtain prediction and correction steps

- Prediction: $\mathbf{x}_t^i \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^i, \mathbf{u}_t)$

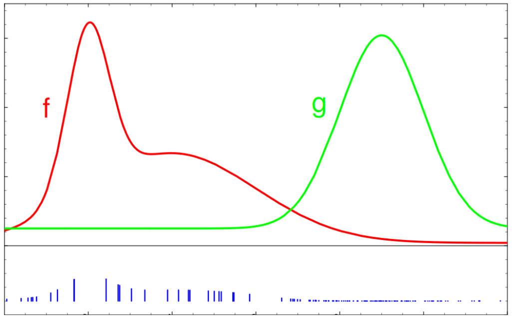
- Correction: $w_t^i = w_{t-1}^i p(\mathbf{y}_t \mid \mathbf{x}_t^i)$

- There can be better choices for the proposal distribution which take the current observation into account!

Sequential Importance Resampling (SIR)

- We propagate samples according to the proposal distribution

- Since the proposal distribution mismatches the target distribution, samples with high accumulated weight can get sparse



- Idea: resample the particles with replacement according to their weight (and reset to equal weights afterwards)

- Choose when to resample according to effective sample size

$$N_{\mathit{eff}} = \frac{1}{\sum_{i=1}^{N} (w_t^i)^2}$$

Particle Filter Properties

- Particle filters can handle arbitrary non-linear observation and state-transition distributions

- Easy to implement and to parallelize

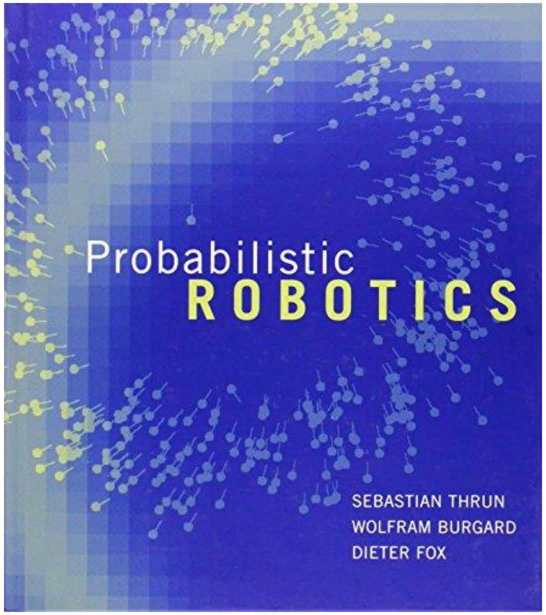
- Caveat: curse of dimensionality. In the worst case, number of samples to approximate the state distribution grows exponentially with number of dimensions

Lessons Learned Today

- State estimation can be modelled in a probabilistic framework
  - Graphical model describes stochastic independence relations between random variables
  - Probabilistic state transition and observation models
- Recursive Bayesian estimation of the state distribution
  - Kalman Filter for linear models with Gaussian noise + Gaussian state estimate
  - KF is efficient and optimal
  - Extended Kalman filter approximate inference for non-linear system
  - EKF has no optimality guarantees, quality depends on linear approximation
  - Particle filters can handle arbitrary non-linear and noise models
  - PFs can represent arbitrary state distributions, but curse of dimensionality!
  - PFs based on importance sampling

Further Reading

- Probabilistic Robotics textbook



Probabilistic Robotics,
S. Thrun, W. Burgard, D. Fox,
MIT Press, 2005

Thanks for your attention!