Robotic 3D Vision

Lecture 3: Probabilistic State Estimation – Filtering

WS 2020/21

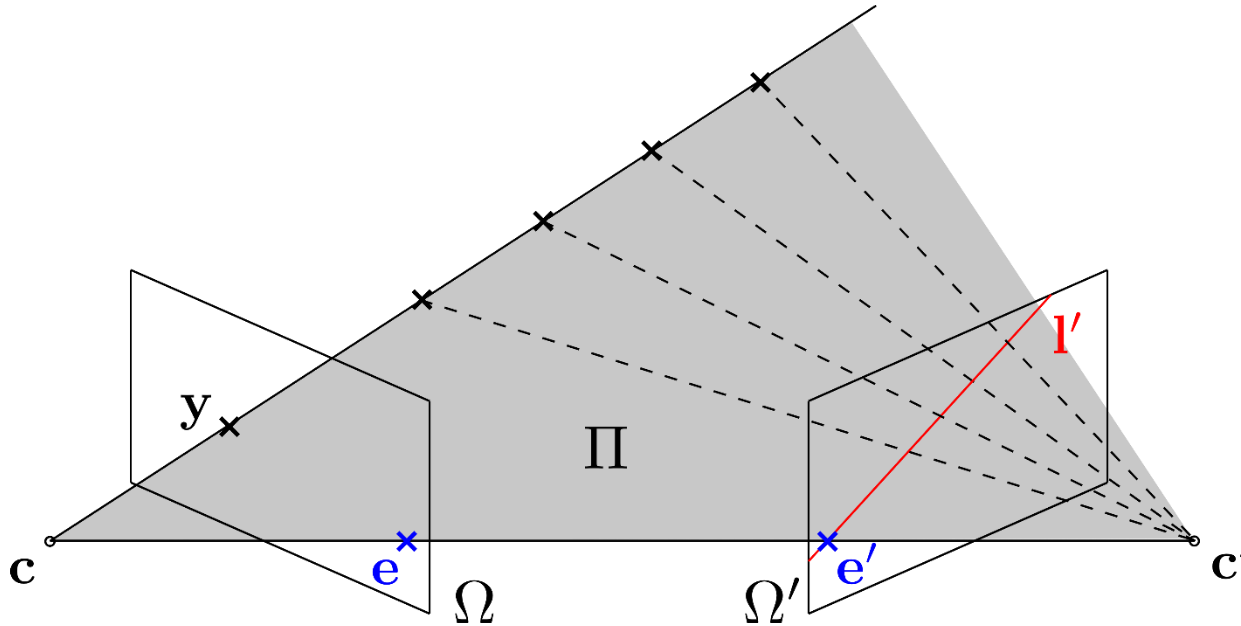
Dr. Niclas Zeller
Artisense GmbH

What We Will Cover Today

- Epipolar Geometry, Essential Matrix (leftover from last lecture)

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

Epipolar Geometry

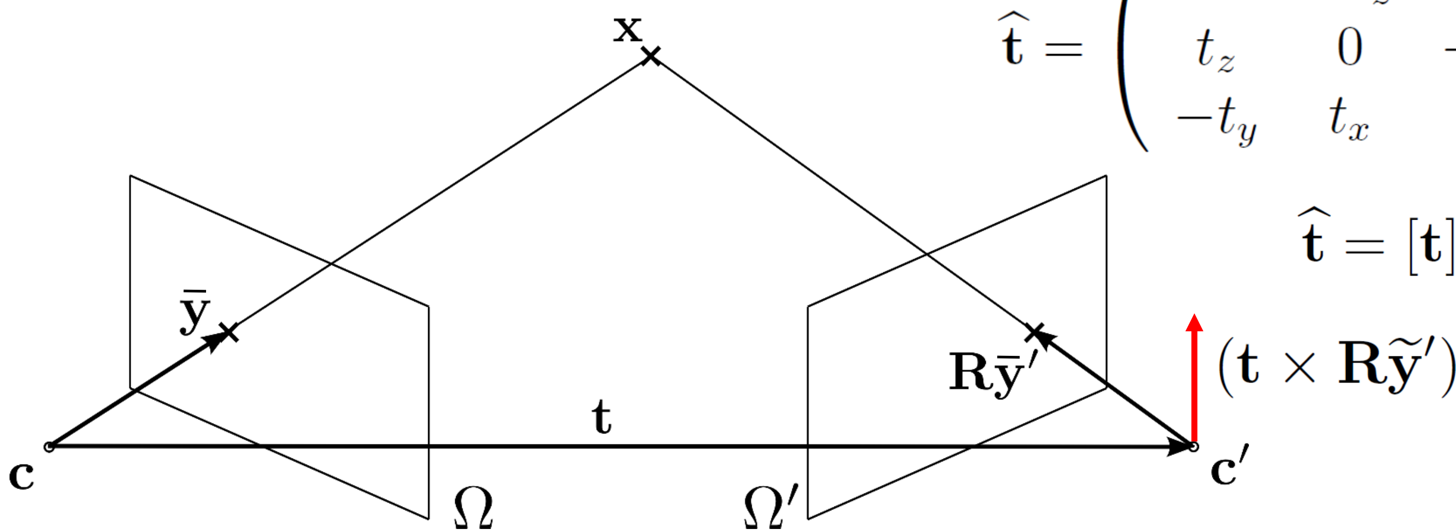


- Camera centers $\mathbf{c}$, $\mathbf{c}'$ and image point $\mathbf{y} \in \Omega$ span the epipolar plane Π
- The ray from camera center $\mathbf{c}$ through point $\mathbf{y}$ projects as the epipolar line $\mathbf{l}'$ in image plane Ω'
- The intersections of the line through the camera centers with the image planes are called epipoles $\mathbf{e}$, $\mathbf{e}'$

Essential Matrix

$$\hat{\mathbf{t}} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}$$

$$\hat{\mathbf{t}} = [\mathbf{t}]_\times$$

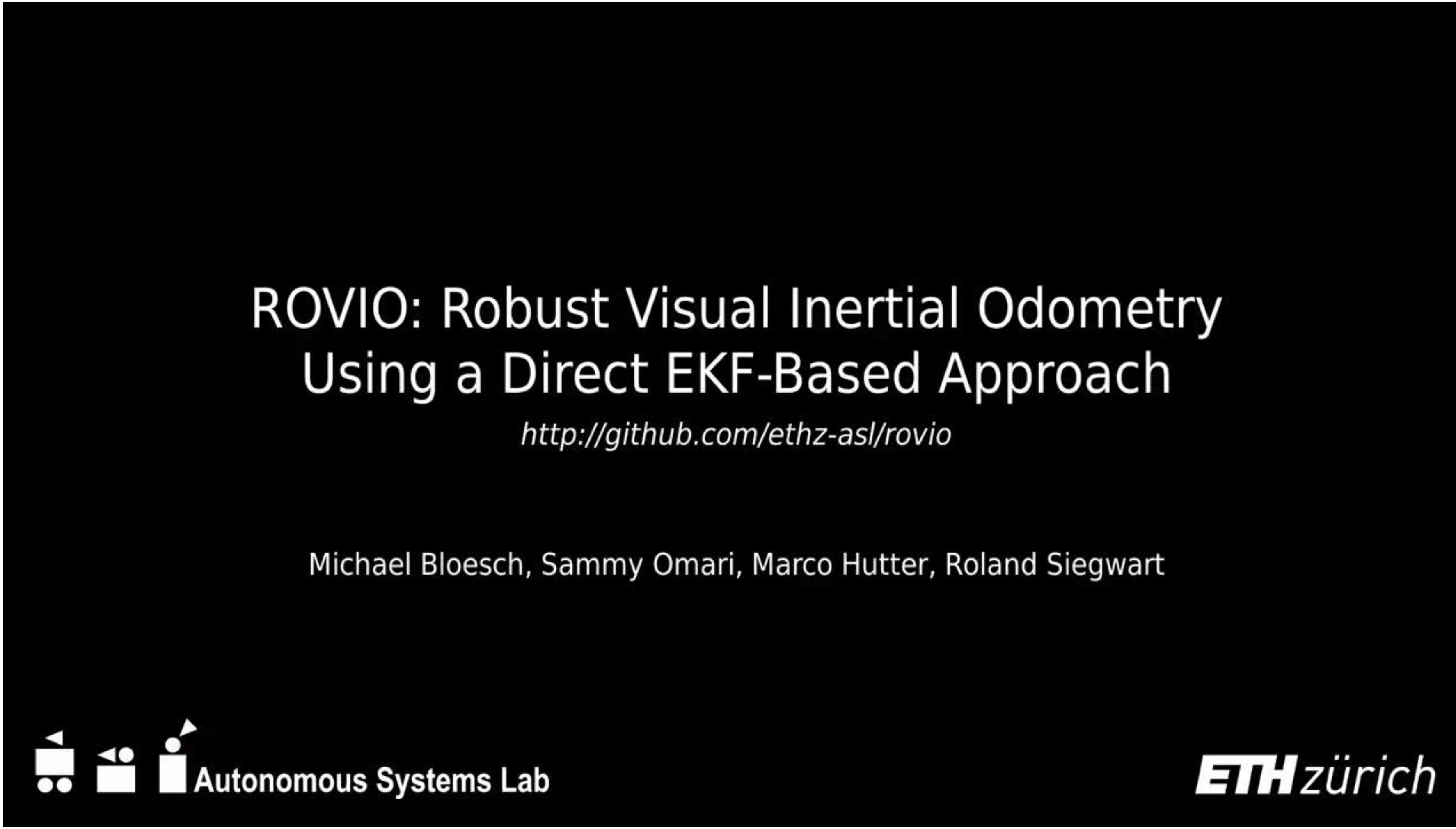


- The rays to the 3D point and the baseline $\mathbf{t}$ are coplanar

$$\tilde{\mathbf{y}}^\top (\mathbf{t} \times \mathbf{R}\tilde{\mathbf{y}}') = 0 \Leftrightarrow \tilde{\mathbf{y}}^\top \hat{\mathbf{t}}\mathbf{R}\tilde{\mathbf{y}}' = 0$$

- The essential matrix $\mathbf{E} := \hat{\mathbf{t}}\mathbf{R}$ captures the relative camera pose
- Each point correspondence provides an „epipolar constraint“
- 5 correspondences suffice to determine $\mathbf{E}$ (simpler: 8-point algorithm)

Probabilistic State Estimation

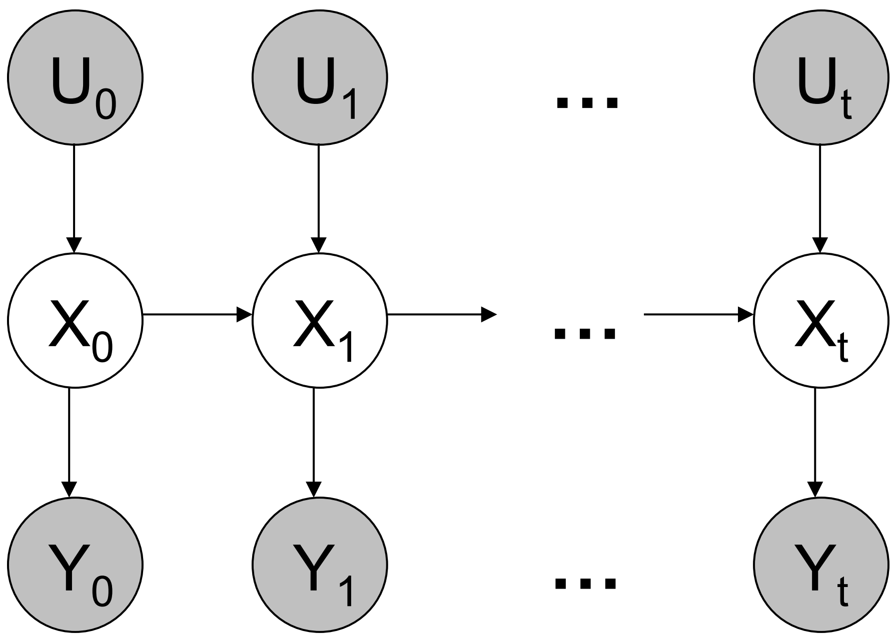


(Bloesch, Omari, Hutter, Siegwart, IROS 2015)

https://www.youtube.com/watch?v=ZMAISVy-6ao

Probabilistic State Estimation

- 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



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 Assumption

- Only the immediate past matters for a state transition

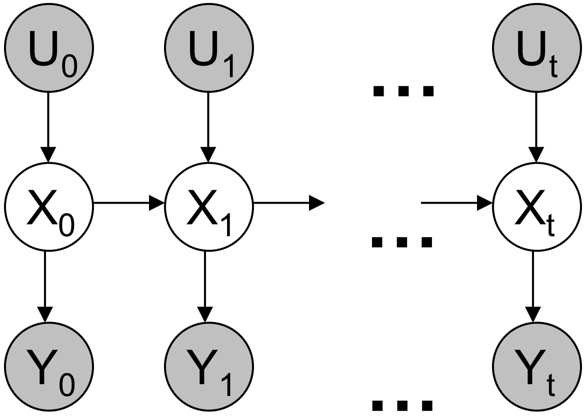
$$p\left(X_t \middle| X_{0:t-1}, U_{0:t}\right) = \boxed{p\left(X_t \middle| X_{t-1}, U_t\right)}$$

state transition model

- Observations depend only on the current state

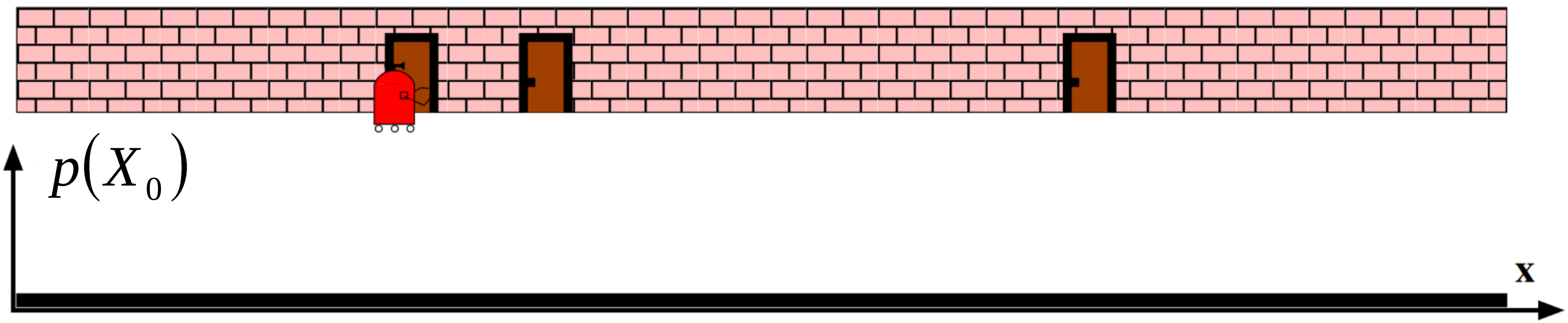
$$p\left(Y_t \middle| X_{0:t}, U_{0:t}, Y_{0:t-1}\right) = \boxed{p\left(Y_t \middle| X_t\right)}$$

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

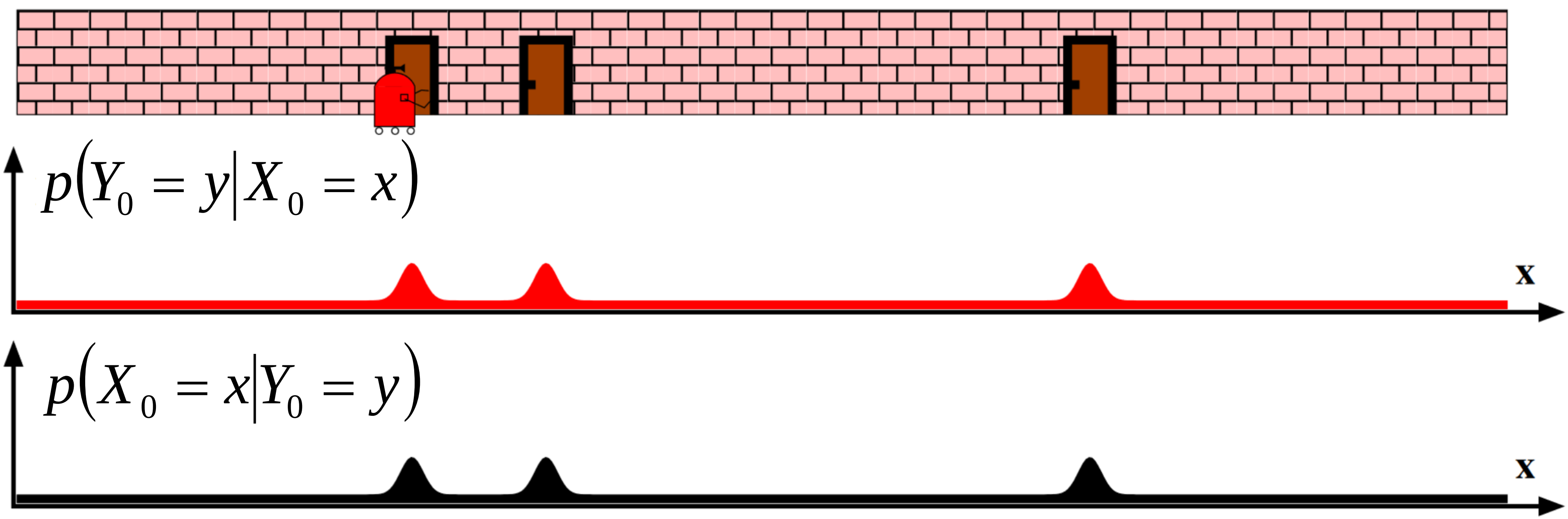


- Initially it knows nothing about its location: uniform $p(X_0)$

Image: 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

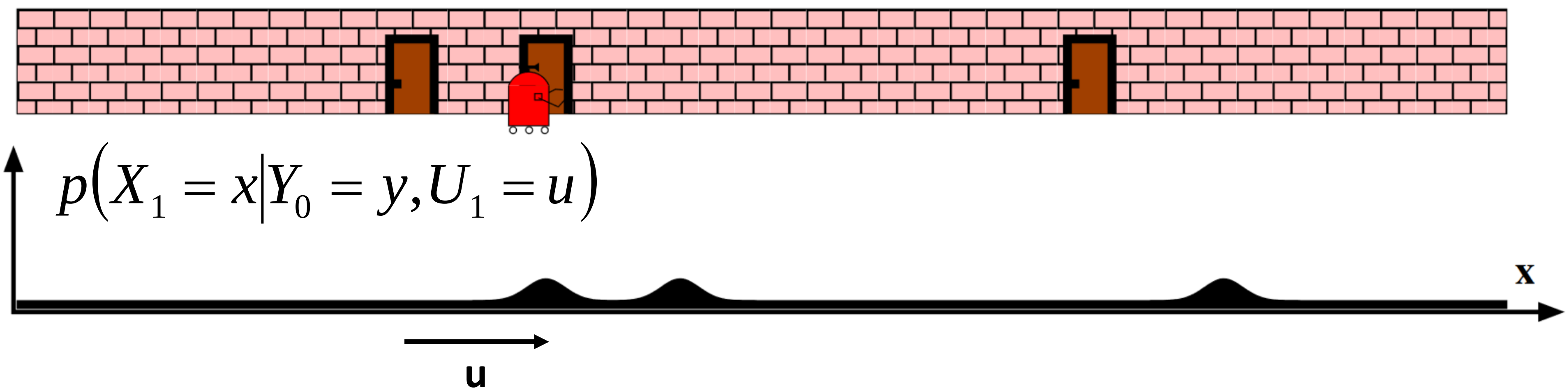


- Observation of door increases the likelihood of x at doors

Image: 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



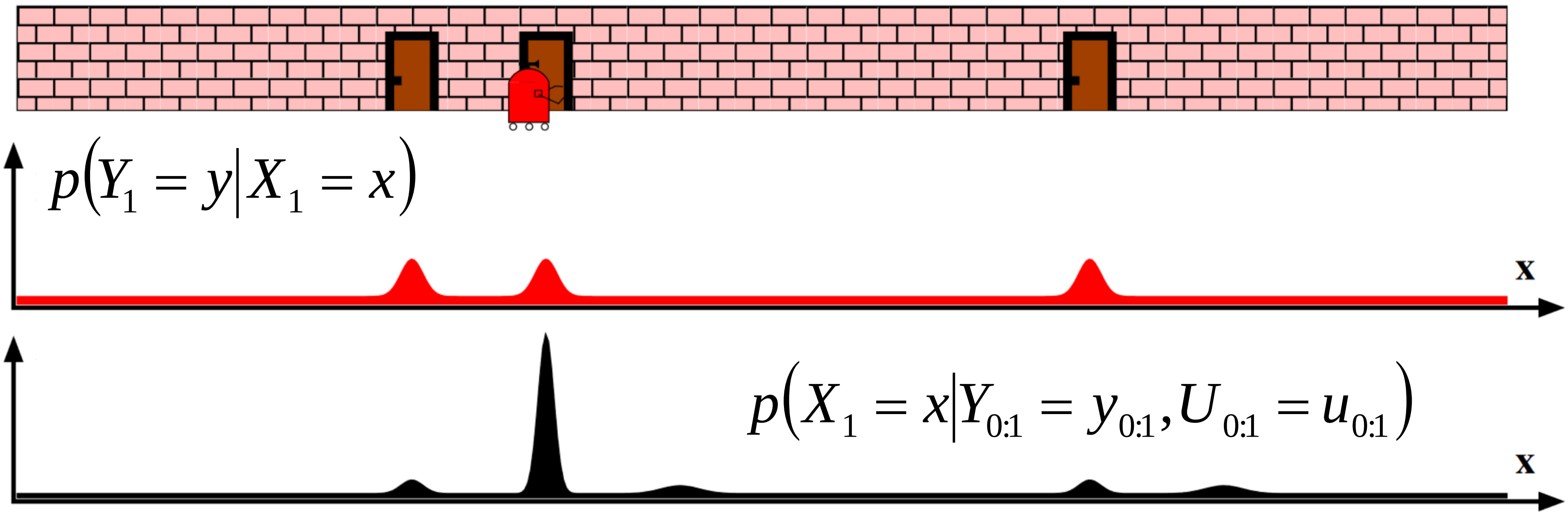
$$p\left(X_1 = x \middle| Y_0 = y, U_1 = u\right)$$

- Robot moves: state is propagated, uncertainty increases

Image: 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

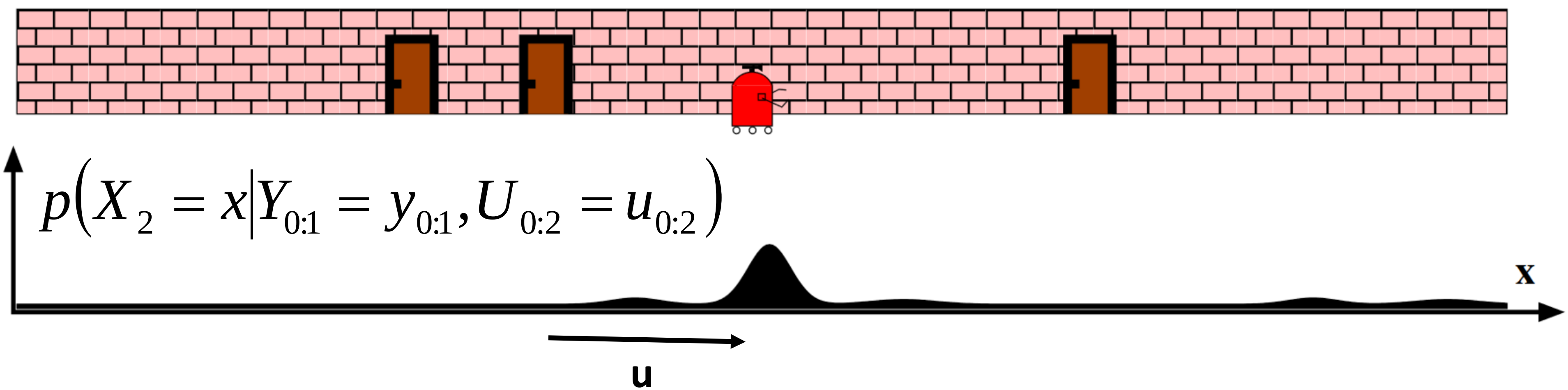


- Observation of door increases the likelihood of x at doors

Image: 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

Image: Thrun, Burgard, Fox, 2005

Bayes' Theorem

$$p(A,B) = p(A|B)p(B) = p(B|A)p(A)$$

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

$$p(A,B|C) = p(A|B,C)p(B|C) = p(B|A,C)p(A|C)$$

$$p(A|B,C) = \frac{p(B|A,C)p(A|C)}{p(B|C)}$$

Recursive State Estimation

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

$$p\left(X_t \middle| y_{0:t}, u_{0:t}\right)$$

$$= \frac{p(y_t \mid X_t, y_{0:t-1}, u_{0:t})\, p(X_t \mid y_{0:t-1}, u_{0:t})}{p(y_t \mid y_{0:t-1}, u_{0:t})}$$

Bayes' theorem

Markov assumption

$$= \frac{p(y_t \mid X_t)\, p(X_t \mid y_{0:t-1}, u_{0:t})}{p(y_t \mid y_{0:t-1}, u_{0:t})}$$

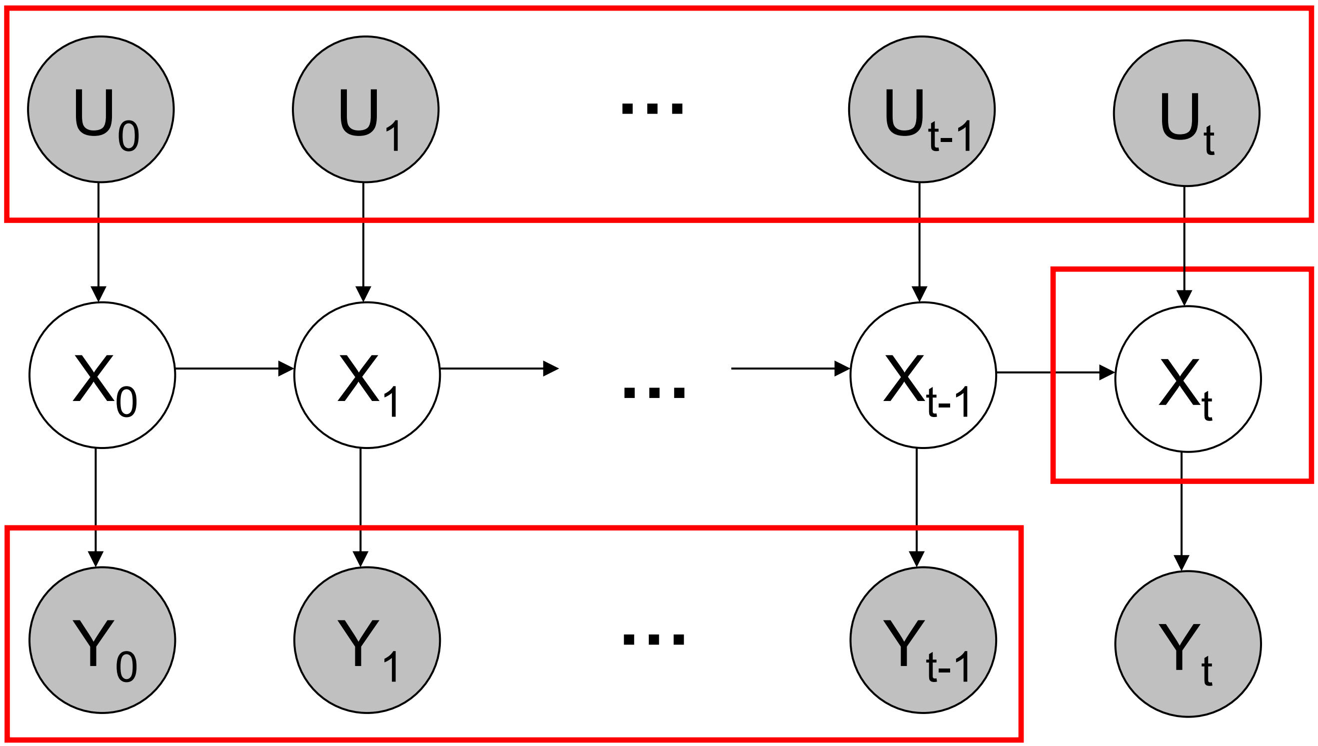
What does this term mean?

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

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

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

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

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

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

Markov assumption

Prediction and Correction

- Prediction:

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

- Correction:

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

Predict-Correct Cycle

- Prediction:

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

observation y_t

action u_t

- Correction:

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

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**

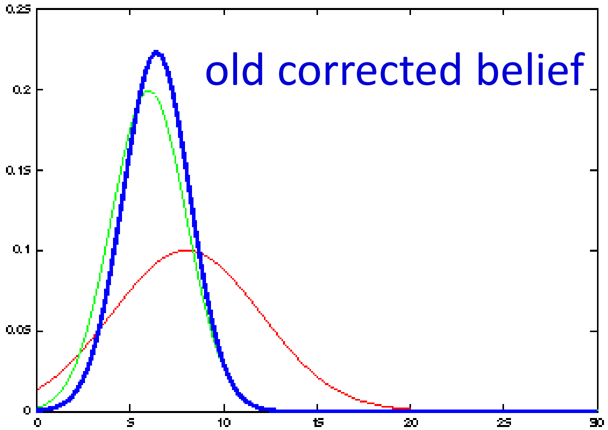
$$\boldsymbol{\mu}_t^+ = \boldsymbol{\mu}_t^- + \mathbf{K}_t (\mathbf{y}_t - \mathbf{C}_t \boldsymbol{\mu}_t^-)$$

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

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

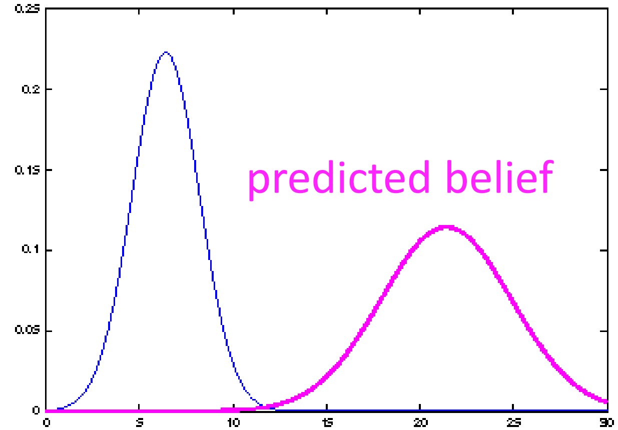


Image: Thrun, Burgard, Fox, 2005

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

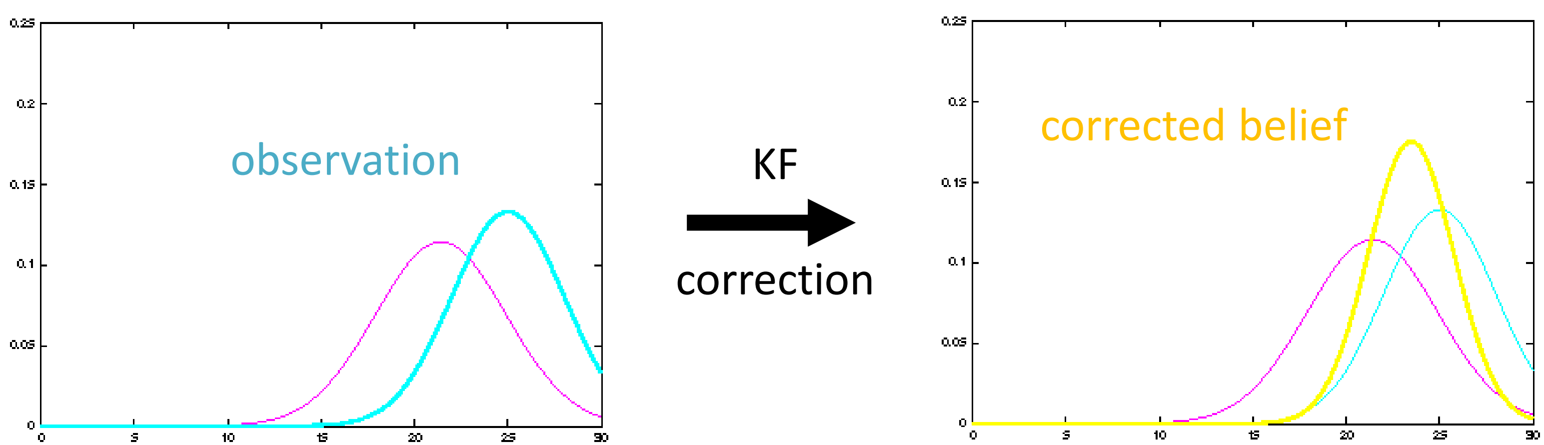
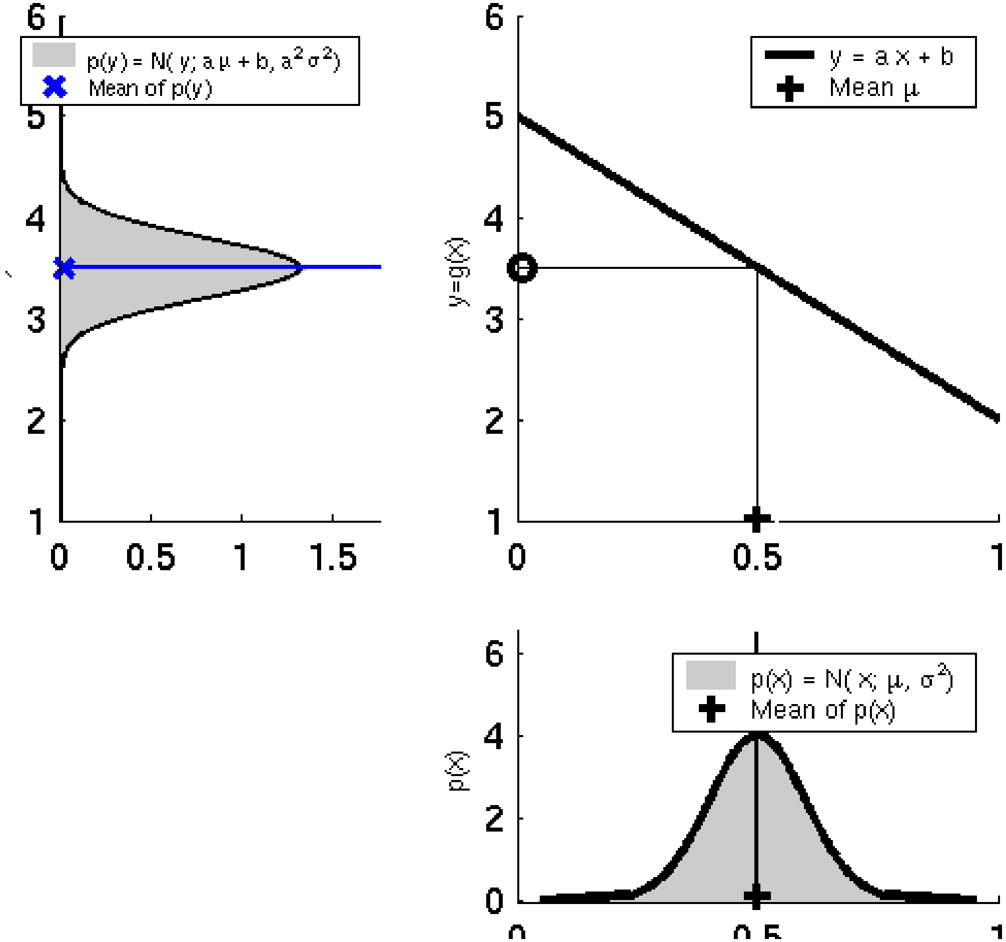


Image: Thrun, Burgard, Fox, 2005

Kalman Filter Properties

- Highly efficient: Polynomial in measurement dimensionality k and state dimensionality n
- Optimal solution for linear Gaussian systems!
- In robotic vision, most models are non-linear!

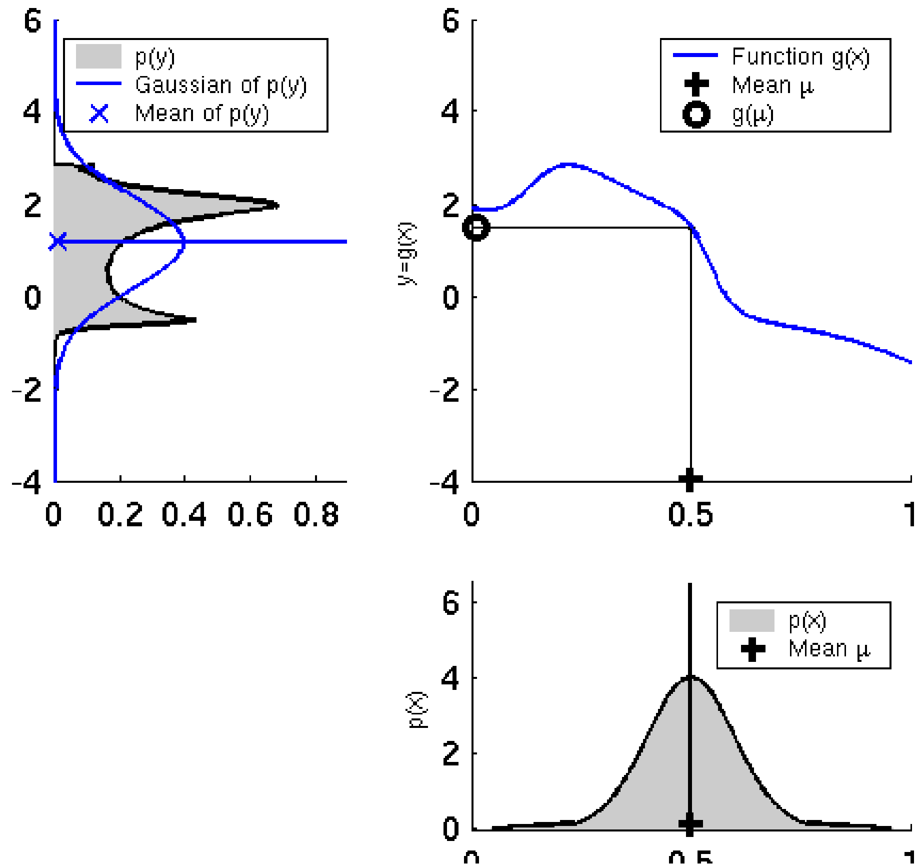
Gaussian Propagation for Linear Models



- Gaussians propagate exactly through a linear function

Image: Thrun, Burgard, Fox, 2005

Gaussian Propagation for Non-Linear Models



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

Image: Thrun, Burgard, Fox, 2005

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:

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

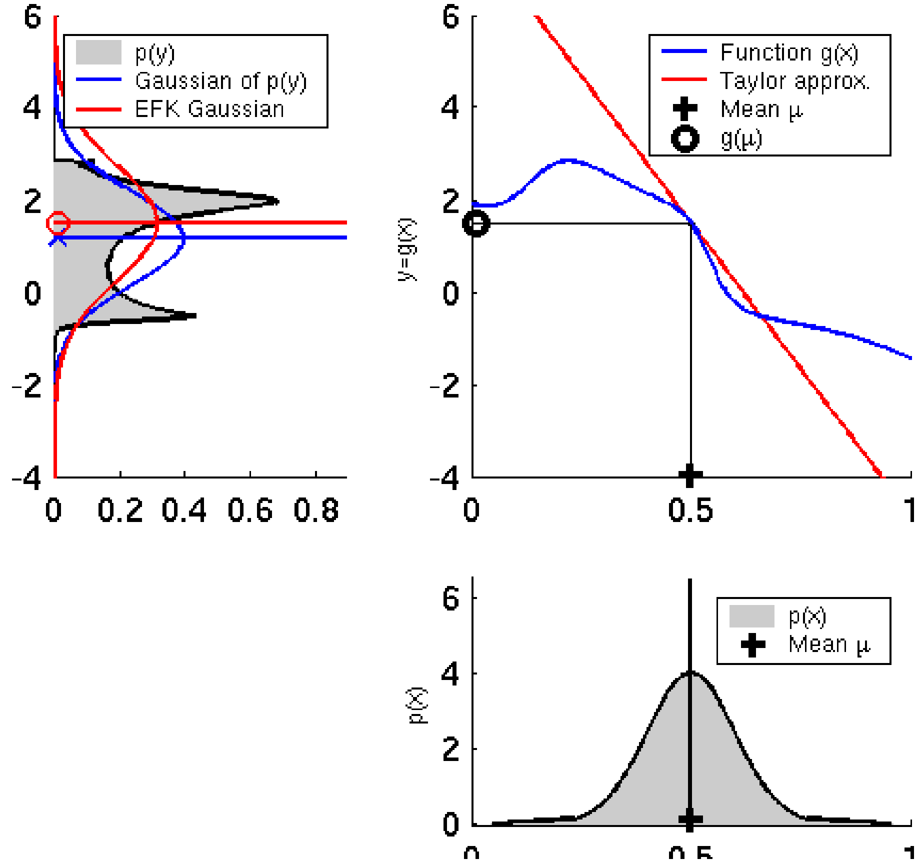
- How to cope with non-linear system?

- Idea: linearize the models in each time step

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

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

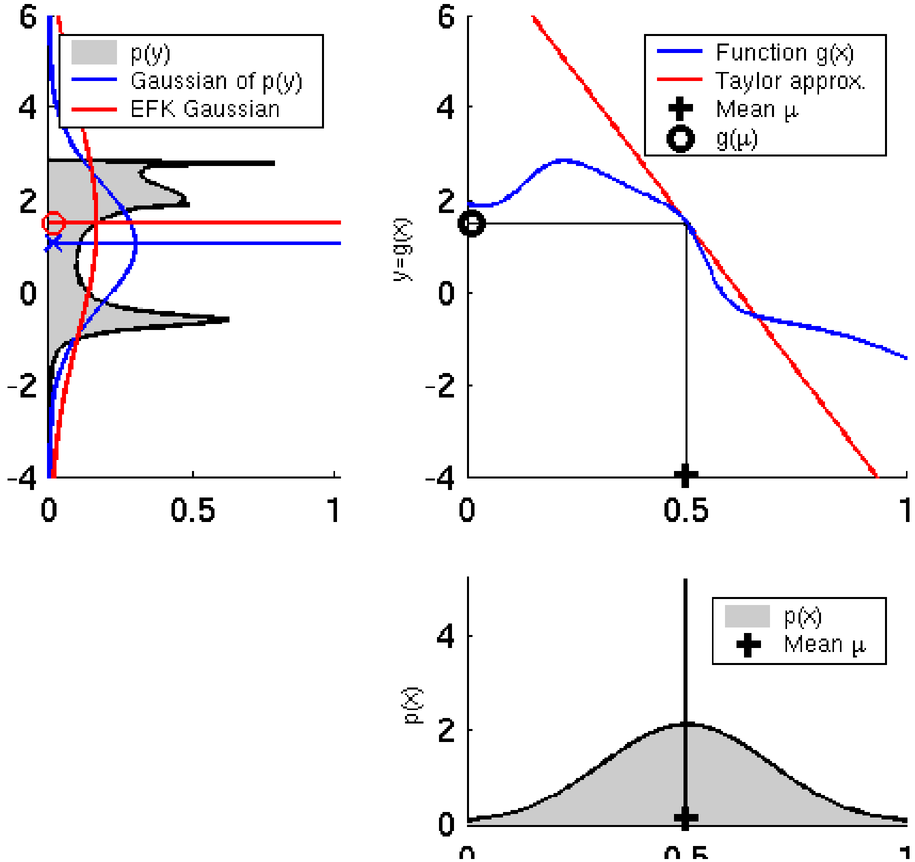
EKF Linearization



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

Image: Thrun, Burgard, Fox, 2005

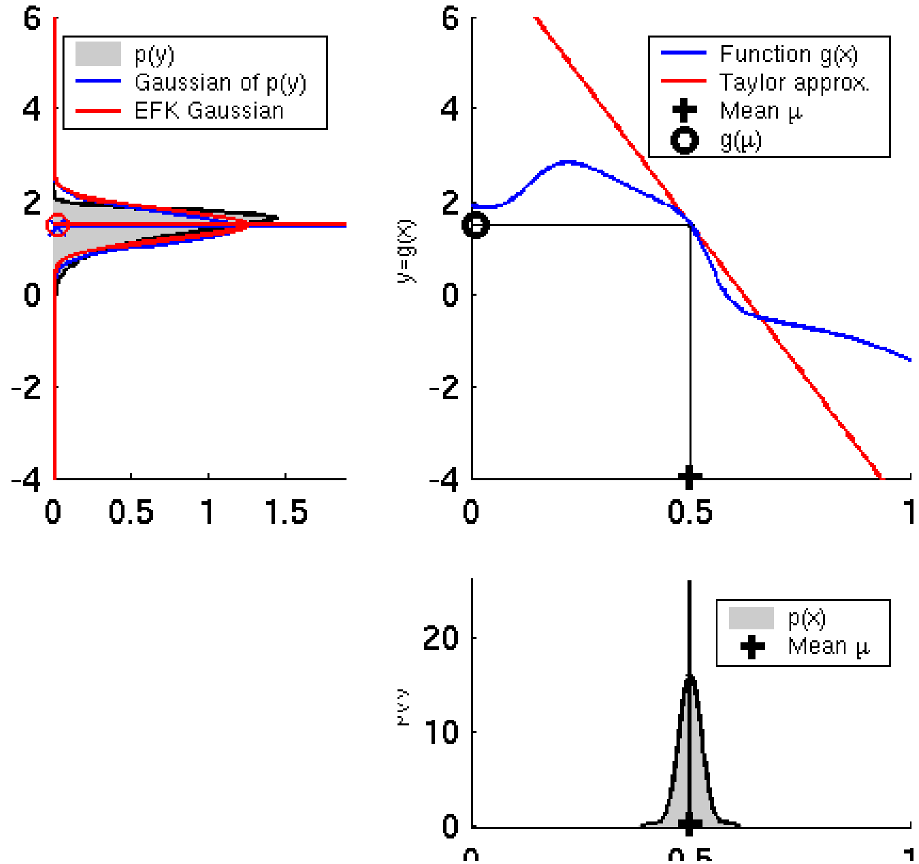
EKF Linearization



- The larger the uncertainty, the larger errors are introduced

Image: Thrun, Burgard, Fox, 2005

EKF Linearization



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

Image: Thrun, Burgard, Fox, 2005

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

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

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

Extended Kalman Filter Properties

- Still highly efficient: Polynomial in measurement dimensionality k and state dimensionality n
- 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)
    - Does not require to explicitly calculate Jacobians

What is a Particle Filter?

- Gaussians are restrictive for state and noise modelling
- Idea:
  - Find a nonparametric implementation for probabilistic state estimation
  - Representation of state estimate by random samples

What is a Particle Filter?

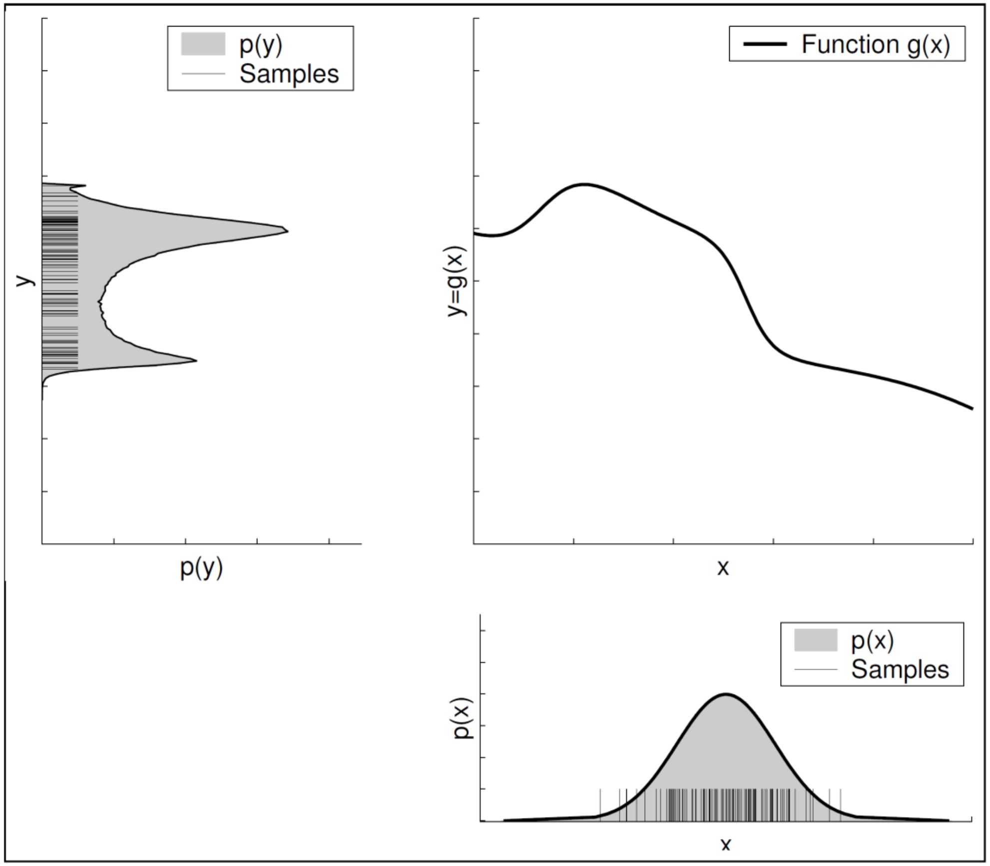
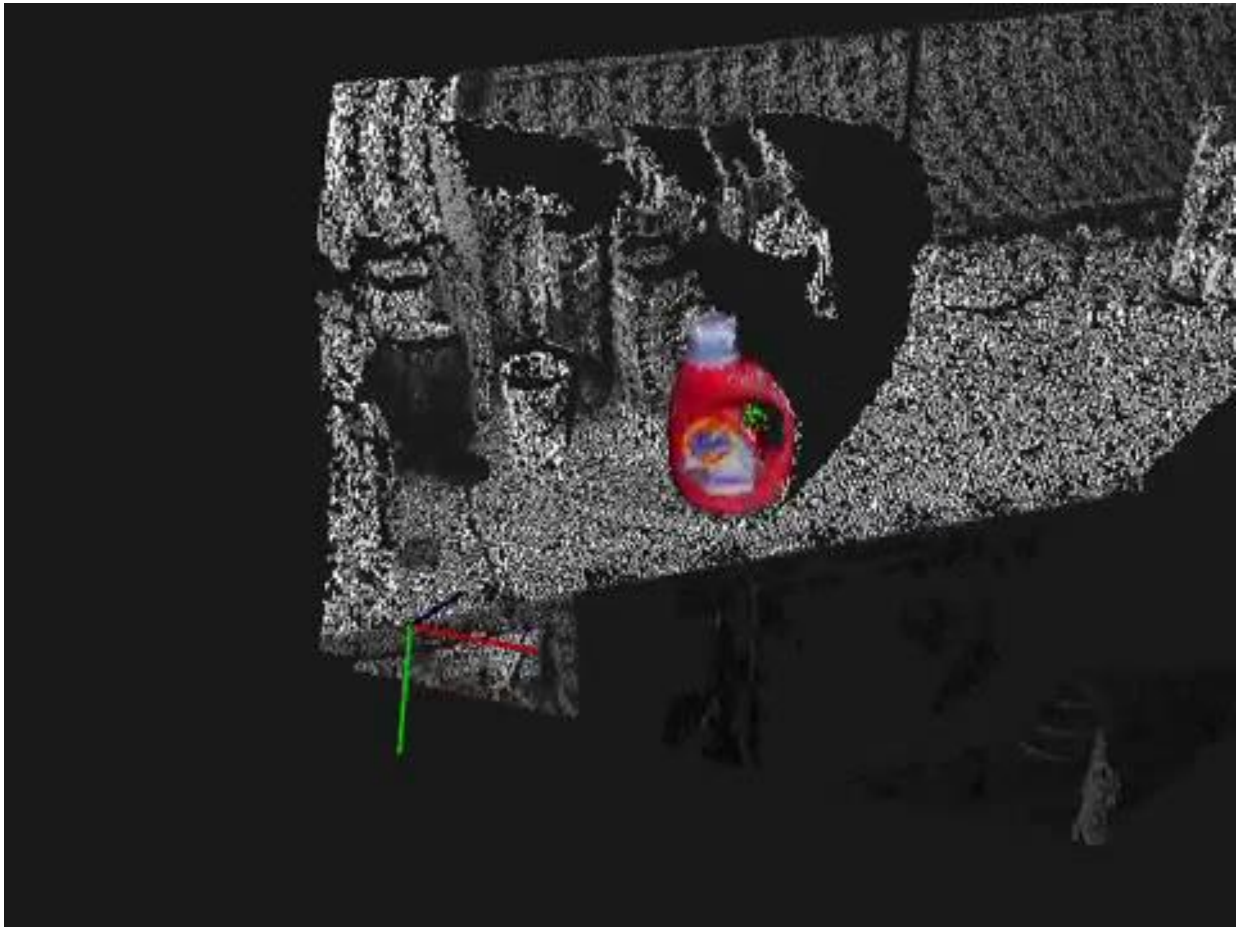


Image: Thrun, Burgard, Fox, 2005

What is a Particle Filter?



(Choi and Christensen, IROS 2013)

https://www.youtube.com/watch?v=ZwlX9CXs6fU&feature=emb_logo

Importance Sampling Concept

- Using particles we are able to handle nonlinearities
  - We able to perform prediction (without considering process noise)

$$p(X_t|y_{0:t-1})$$

- How can we incorporate a new measurement y_t?
  - How do we get to

$$p(X_t|y_{0:t})$$

- Weighting of particles respectively → importance sampling

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)

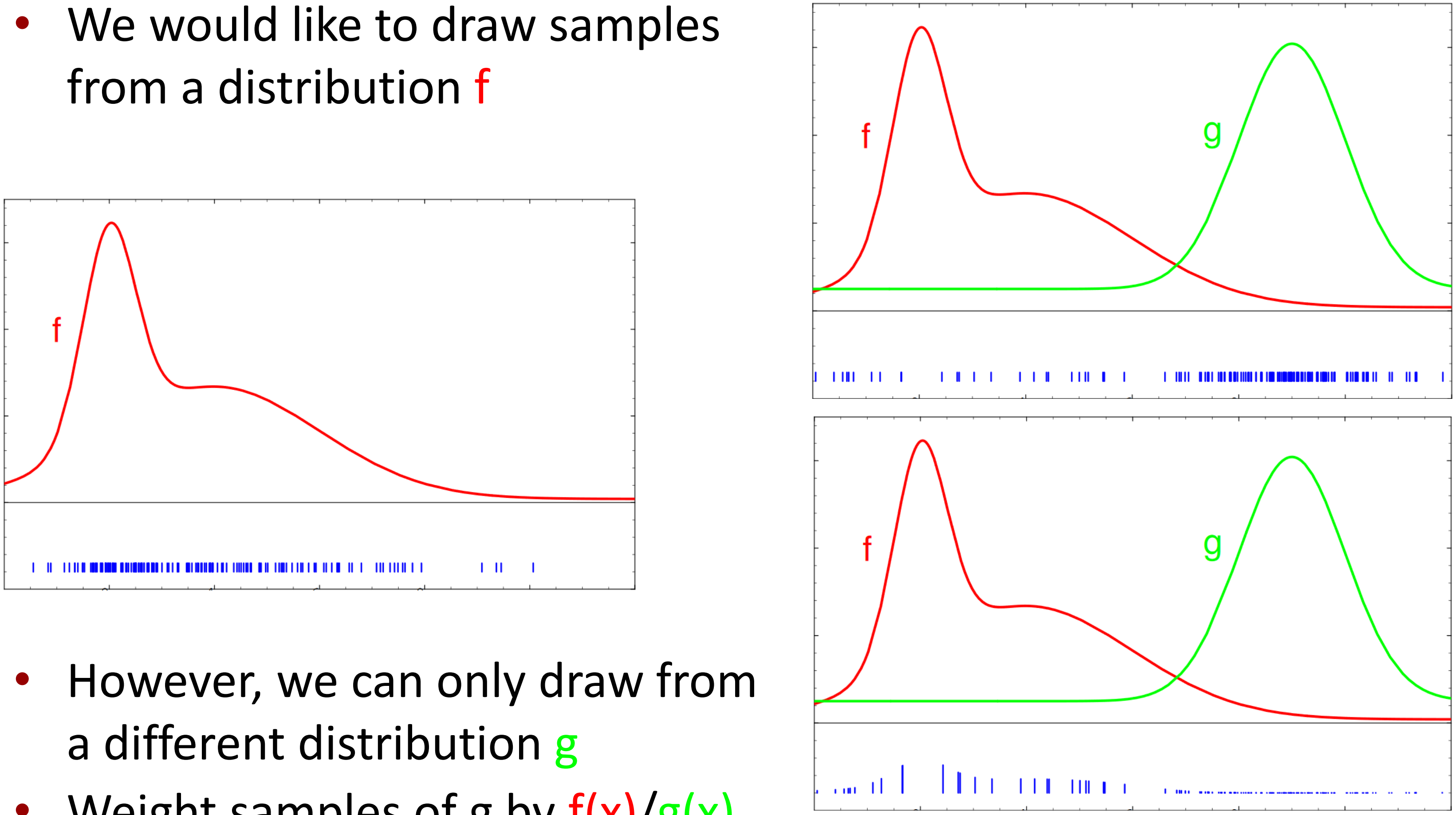


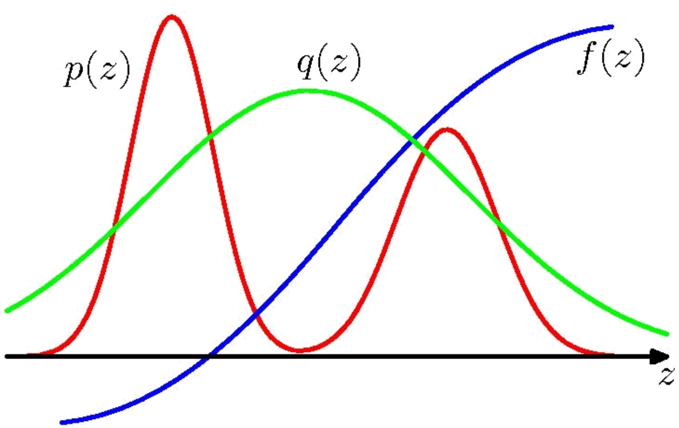
Image: Thrun, Burgard, Fox, 2005

Importance Sampling Concept

- 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 $q(\mathbf{z})$

$$\mathbb{E}_p\{f(Z)\} = \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz$$

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



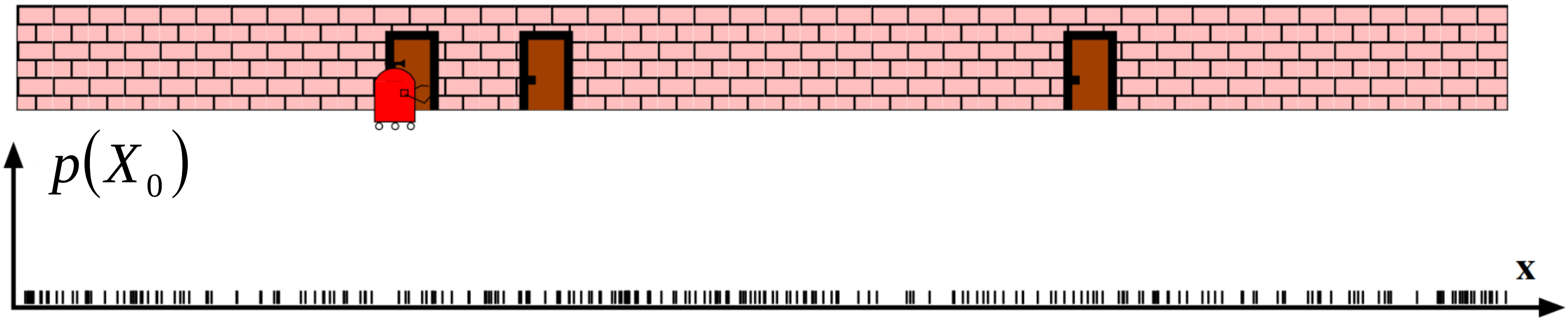
- With importance weights

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

Image: 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

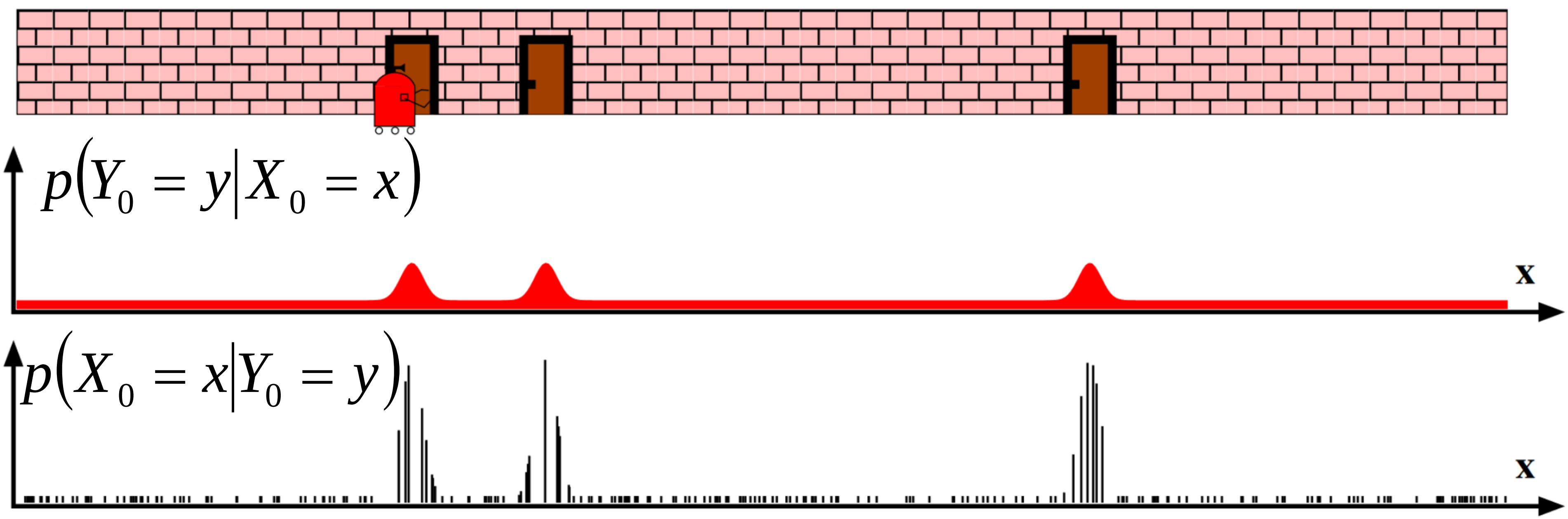


- Initially it knows nothing about its location: uniform $p(X_0)$

Image: 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

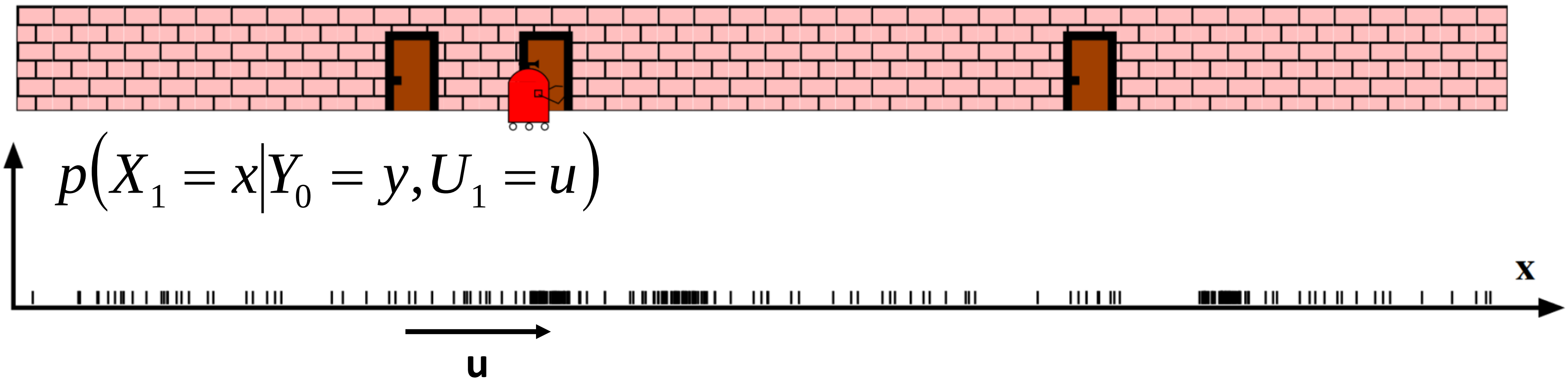


- Observation of door increases the likelihood of x at doors

Image: 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

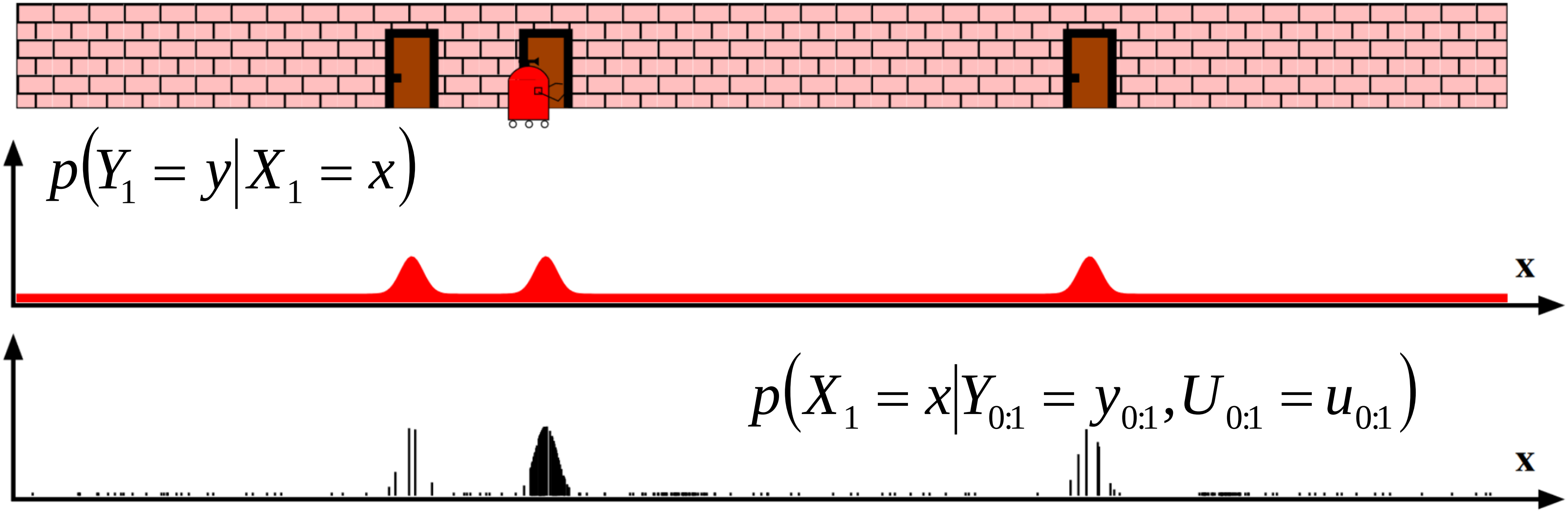


- Robot moves: state is propagated, uncertainty increases
- Samples are resampled and propagated

Image: 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

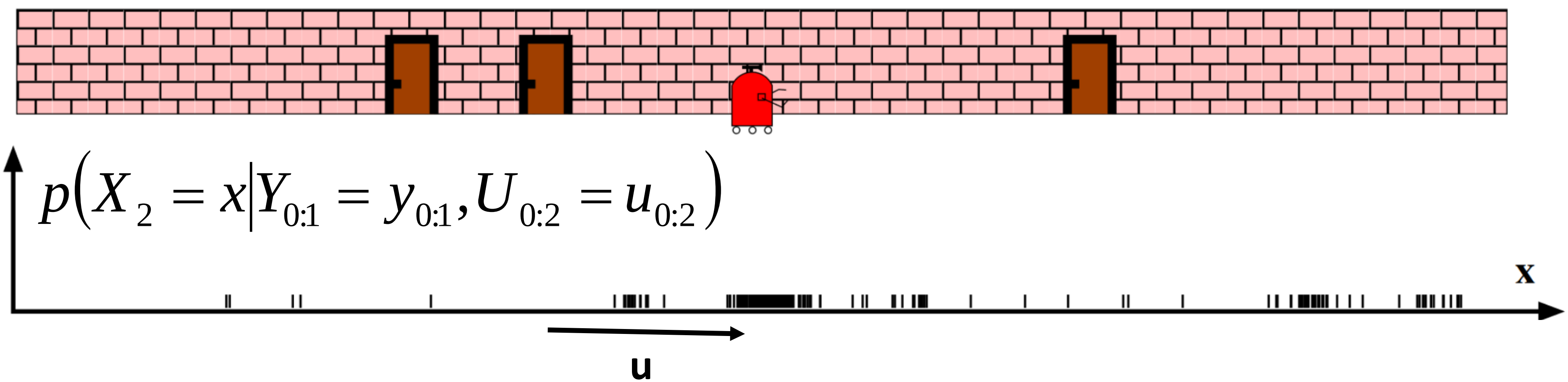


- Observation of door increases the likelihood of x at doors

Image: 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: Thrun, Burgard, Fox, 2005

Particle Filter (PF)

- Non-linear observation and state-transition distributions

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

- State estimate is represented as a set of weighted samples

$$\{\boldsymbol{x}_t^i, w_t^i\}_{i=1}^N$$

State hypothesis **Importance weight**

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

- The weighted samples a.k.a. particles are propagated and updated over time to approximate the posterior $p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t},\boldsymbol{u}_{1:t})$

Sequential Importance Sampling (SIS)

- Draw samples from a proposal distribution $q(\mathbf{x}_t | \cdots)$ given
  - previous samples $\mathbf{x}_{t-1}^i$
  - potentialy measurement $\mathbf{y}_t$ and action $\mathbf{u}_t$
- Update weights of particles

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

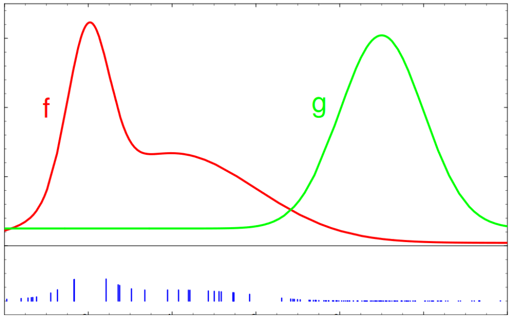
- Correction: $$w_t^i = w_{t-1}^i p(\mathbf{y}_t \mid \mathbf{x}_t^i)$$

- 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

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

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

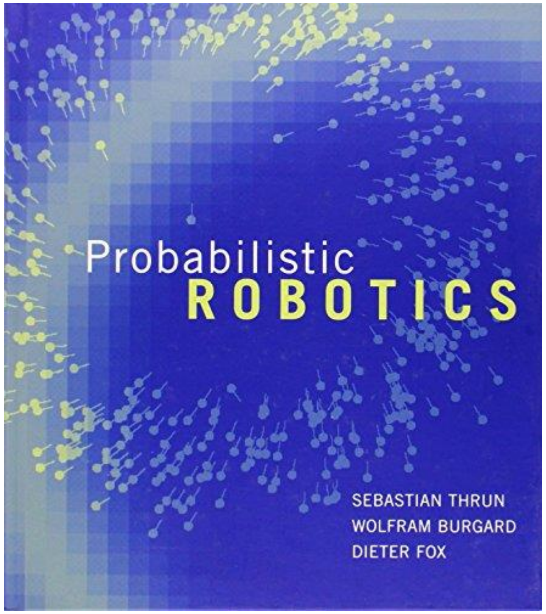
Lessons Learned Today

- State estimation can be modelled in a probabilistic framework
  - Simplification based on Markov assumption
  - 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 for the linear Gaussian case
  - 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 systems and noise models
  - PFs can represent arbitrary state distributions
  - PFs are based on importance sampling

Further Reading

- Probabilistic Robotics textbook



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

Thanks for your attention!

Slides Information

- These slides have been initially created by Jörg Stückler as part of the lecture “Robotic 3D Vision” in winter term 2017/18 at Technical University of Munich.
- The slides have been revised by myself (Niclas Zeller) for the same lecture held in winter term 2020/21
- Acknowledgement of all people that contributed images or video material has been tried (please kindly inform me if such an acknowledgement is missing so it can be added).