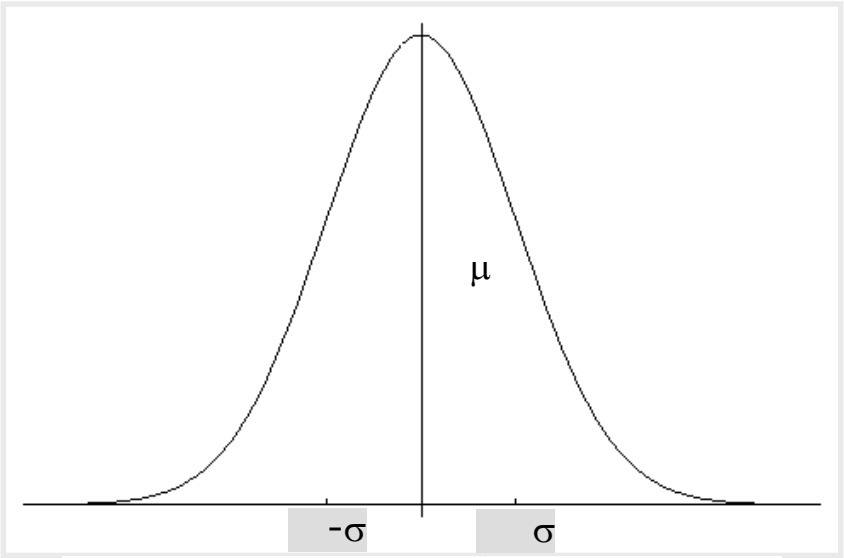
Gaussian (or Normal) Distribution

Univariate

$$p(x) \sim N(\mu, \sigma^2)$$

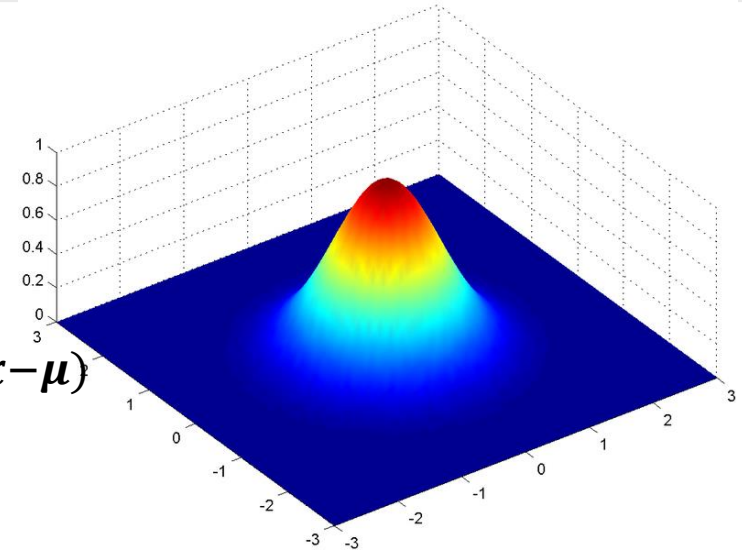
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1(x-\mu)^2}{2 \quad \sigma^2}}$$



Multivariate

$$p(\boldsymbol{x}) \sim N(\boldsymbol{\mu}, \Sigma)$$

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}$$



Properties of Gaussians

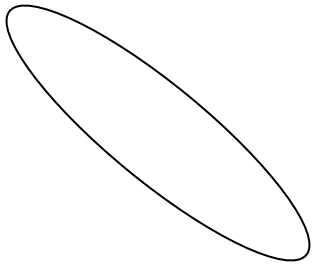
$$\left.\begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow Y \sim N(a\mu + b, a^2\sigma^2)$$

$$\left.\begin{array}{l} X_1 \sim N\left(\mu_1, \sigma_1^{\ 2}\right) \\ X_2 \sim N\left(\mu_2, \sigma_2^{\ 2}\right) \end{array}\right\} \Rightarrow X_1 + X_2 \sim N(\mu_1 + \mu_2, \sigma_1^{\ 2} + \sigma_2^{\ 2})$$

- We stay in the "Gaussian world" as long as we start with Gaussians and perform only linear transformations.
- Same holds for multivariate Gaussians

Covariance, covariance, covariance

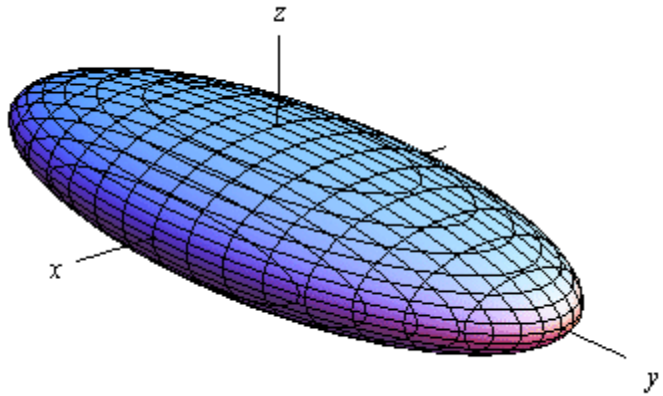
- Know what's a covariance matrix
  - What it does, what it means, why it matters, how you can compute one, how you can visualize them



$$\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{i=0}^{N} \boldsymbol{x}_i$$

$$\boldsymbol{e}_i = \boldsymbol{x}_i - \overline{\boldsymbol{x}}$$

$$\Sigma = \mathrm{E}[\boldsymbol{e}_i \boldsymbol{e}_i^T]$$



Kalman Filter

- Recursive solution for discrete linear filtering problems
  - A state $\boldsymbol{x} \in R^n$
  - A measurement $\boldsymbol{z} \in R^m$
  - Discrete (i.e. for time t = 1, 2, 3, …)
  - Recursive process (i.e. $\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1})$)
  - Linear system (i.e $\boldsymbol{x}_t = A\,\boldsymbol{x}_{t-1}$)

- The system is defined by:

1) ***linear*** process model

$$\boldsymbol{x}_t = A\,\boldsymbol{x}_{t-1} + B\,\boldsymbol{u}_{t-1} + \boldsymbol{w}_{t-1}$$

state transition; control Input (optional); Gaussian white noise

How a state transitions into another state

2) ***linear*** measurement model

$$\boldsymbol{z}_t = H\,\boldsymbol{x}_t + \boldsymbol{v}_t$$

observation model; Gaussian white noise

How a state relates to a measurement

Kalman Filter

- Recipe:
  1. Start with an initial guess

$$\hat{\boldsymbol{x}}_0, \Sigma_0$$

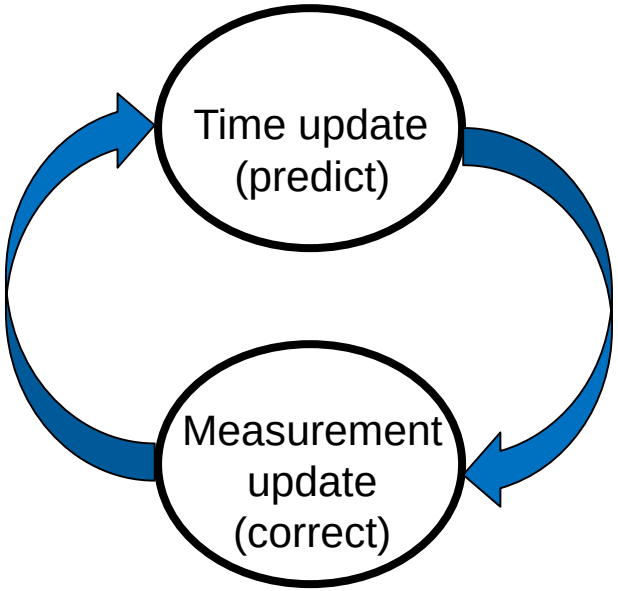
  2. Compute prior (prediction)

$$\hat{\boldsymbol{x}}'_t = A\hat{\boldsymbol{x}}_{t-1} + B\boldsymbol{u}_{t-1}$$
$$\Sigma'_t = A\Sigma_{t-1}A^T + Q$$

  3. Compute posterior (correction)

$$K_t = \Sigma'_t H^T (H\Sigma'_t H^T + R)^{-1}$$
$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}'_t + K_t(\boldsymbol{z}_t - H\hat{\boldsymbol{x}}'_t)$$
$$\Sigma_t = (1 - K_t H)\Sigma'_t$$

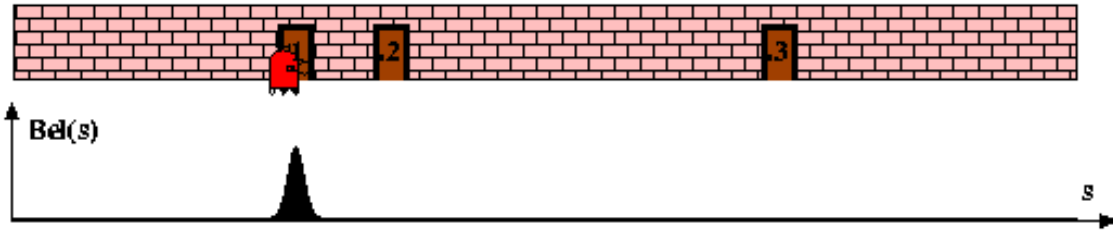
REPEAT



Minimizes the sum of the expected errors: $\sum e_i^2$

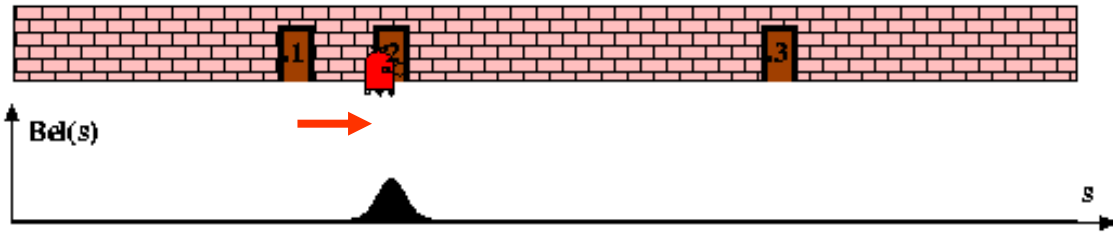
Initial

$\hat{\boldsymbol{x}}_0, \Sigma_0$



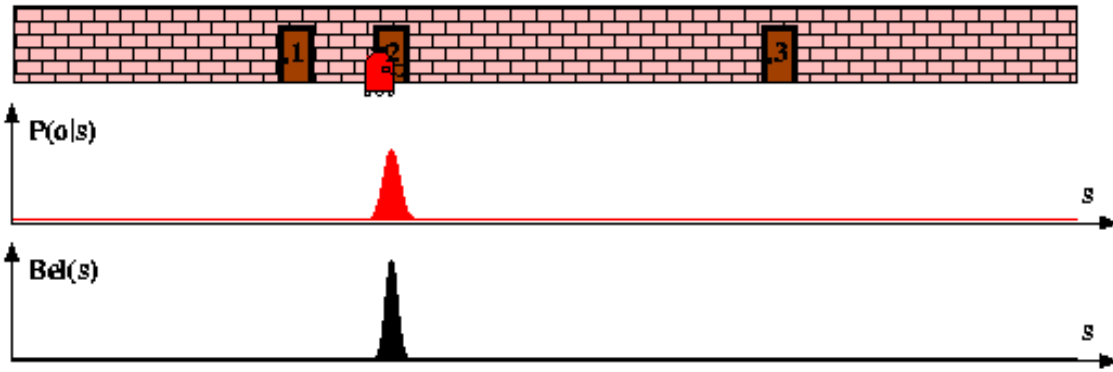
Predict $\hat{\boldsymbol{x}}_1'$ from $\hat{\boldsymbol{x}}_0$ and $\boldsymbol{u}_0$

$$\hat{\boldsymbol{x}}_1' = A\hat{\boldsymbol{x}}_0 + B\boldsymbol{u}_0$$
$$\Sigma_1' = A\Sigma_0 A^T + Q$$



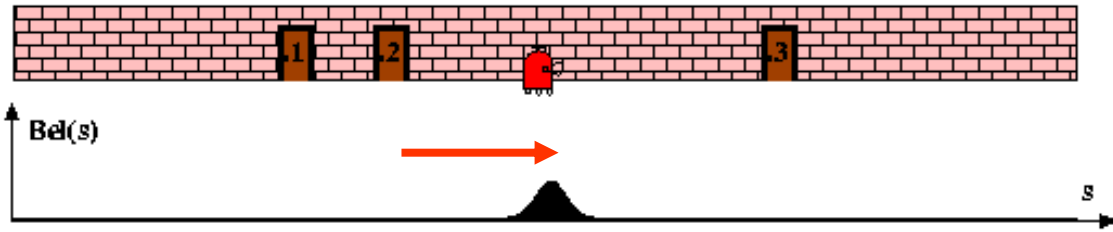
Correction using $\boldsymbol{z}_1$

$$K_1 = \Sigma_1' H^T (H\Sigma_1' H^T + R)^{-1}$$
$$\hat{\boldsymbol{x}}_1 = \hat{\boldsymbol{x}}_1' + K_1(\boldsymbol{z}_1 - H\hat{\boldsymbol{x}}_1')$$
$$\Sigma_1 = (1 - K_1 H)\Sigma_1'$$



Predict $\hat{\boldsymbol{x}}_2'$ from $\hat{\boldsymbol{x}}_1$ and $\boldsymbol{u}_1$

$$\hat{\boldsymbol{x}}_2' = A\hat{\boldsymbol{x}}_1 + B\boldsymbol{u}_1$$
$$\Sigma_2' = A\Sigma_1 A^T + Q$$



Kalman Filter Limitations

- Assumptions:
  - Linear process model $\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + \boldsymbol{w}_t$
  - Linear observation model $\boldsymbol{z}_t = H\boldsymbol{x}_t + \boldsymbol{v}_t$
  - White Gaussian noise $N(0, \Sigma)$

- What can we do if system is not linear?
  - Non-linear state dynamics $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_t)$
  - Non-linear observations $\boldsymbol{z}_t = h(\boldsymbol{x}_t, \boldsymbol{v}_t)$

Extended Kalman Filter

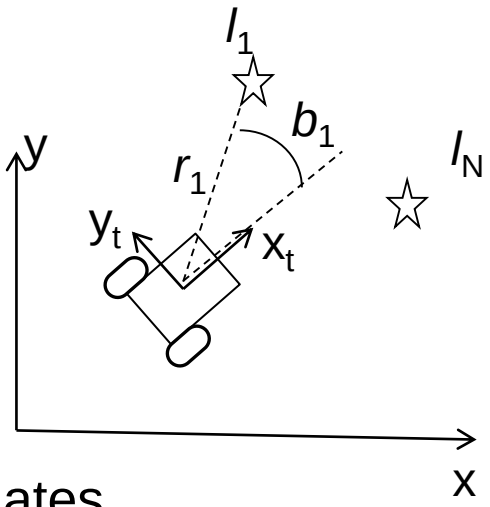
- Kalman Filter Recipe:
  - Given
    $\hat{x}_0, \Sigma_0$
  - Prediction
    $$\hat{\boldsymbol{x}}'_t = A\hat{\boldsymbol{x}}_{t-1} + B\boldsymbol{u}_{t-1}$$
    $$\Sigma'_t = A\Sigma_{t-1}A^T + Q$$
  - Measurement correction
    $$K_t = \Sigma'_t H^T (H\Sigma'_t H^T + R)^{-1}$$
    $$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}'_t + K(\boldsymbol{z}_t - H\hat{\boldsymbol{x}}'_t)$$
    $$\Sigma_t = (I - K_t H)\Sigma'_t$$

- Extended Kalman Filter Recipe:
  - Given
    $\hat{x}_0, \Sigma_0$
  - Prediction
    $$\hat{\boldsymbol{x}}'_t = \boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_{t-1}, 0)$$
    $$\Sigma'_t = A_t\Sigma_{t-1}A_t^T + W_t Q W_t^T$$
  - Measurement correction
    $$K_t = \Sigma'_t H_t^T (H_t\Sigma'_t H_t^T + V_t R V_t^T)^{-1}$$
    $$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}'_t + K_t(\boldsymbol{z}_t - \boldsymbol{h}(\hat{\boldsymbol{x}}'_t, 0))$$
    $$\Sigma_t = (I - K_t H_t)\Sigma'_t$$

EKF for Range-Bearing Localization

- State $\boldsymbol{s}_t = [x_t \quad y_t \quad \theta_t]^T$ 2D position and orientation
- Input $\boldsymbol{u}_t = [v_t \quad \omega_t]^T$ linear and angular velocity
- Process model

$$\boldsymbol{f}(\boldsymbol{s}_t, \boldsymbol{u}_t, \boldsymbol{w}_t) = \begin{bmatrix} x_{t-1} + (\Delta t) v_{t-1} \cos(\theta_{t-1}) \\ y_{t-1} + (\Delta t) v_{t-1} \sin(\theta_{t-1}) \\ \theta_{t-1} + (\Delta t) \omega_{t-1} \end{bmatrix} + \begin{bmatrix} w_{x_t} \\ w_y \\ w_{\theta_t} \end{bmatrix}$$



- Given a map, the robot sees N landmarks with coordinates

$$\boldsymbol{l}_1 = [x_{l_1} \quad y_{l_1}]^T, \cdots, \boldsymbol{l}_N = [x_{l_N} \quad y_{l_N}]^T$$

The observation model is

$$\boldsymbol{z}_t = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{s}_t, \boldsymbol{v}_1) \\ \vdots \\ \boldsymbol{h}_N(\boldsymbol{s}_t, \boldsymbol{v}_N) \end{bmatrix} \quad \boldsymbol{h}_i(\boldsymbol{s}_t, \boldsymbol{v}_t) = \begin{bmatrix} \sqrt{(x_t - x_{l_i})^2 + (y_t - y_{l_i})^2} \\ \tan^{-1}\left(\frac{y_t - y_{l_i}}{x_t - x_{l_i}}\right) - \theta_t \end{bmatrix} + \begin{bmatrix} v_r \\ v_b \end{bmatrix}$$

Kalman Filters and SLAM

- Localization: state is the location of the robot
- Mapping: state is the location of 2D landmarks
- SLAM: state combines both
- If the state is $\boldsymbol{s}_t = \begin{bmatrix} x_t & y_t & \theta_t & l_{1_t}^T & \cdots & l_{N_t}^T \end{bmatrix}^T$
  then we can write a linear observation system
  - note that if we don't have some fixed landmarks, our system is *unobservable* (we can't fully determine all unknown quantities)
- Covariance Σ is represented by

http://ais.informatik.uni-freiburg.de

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\ \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\ \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} & \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\ \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta} & \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} \end{bmatrix}$$

EKF Range Bearing SLAM
Prior State Estimation

- State $\boldsymbol{s}_t = [x_t \quad y_t \quad \theta_t \quad \boldsymbol{l}_1^T \quad \cdots \quad \boldsymbol{l}_N^T]$ position/orientation of robot and landmarks coordinates
- Input $\boldsymbol{u}_t = [v_t \quad \omega_t]^T$ forward and angular velocity
- The process model for localization is

$$\boldsymbol{s}_t' = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta t v_{t-1} \cos(\theta_{t-1}) \\ \Delta t v_{t-1} \sin(\theta_{t-1}) \\ \Delta t \omega_{t-1} \end{bmatrix}$$

This model is augmented for 2N+3 dimensions to accommodate landmarks. This results in the process equation

$$\begin{bmatrix} x_t' \\ y_t' \\ \theta_t' \\ \boldsymbol{l}_{1_t}' \\ \vdots \\ \boldsymbol{l}_{N_t}' \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \\ \boldsymbol{l}_{1_{t-1}} \\ \vdots \\ \boldsymbol{l}_{N_{t-1}} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta t v_{t-1} \cos(\theta_{t-1}) \\ \Delta t v_{t-1} \sin(\theta_{t-1}) \\ \Delta t \omega_{t-1} \end{bmatrix}$$

Landmarks don't depend on external input

EKF Range Bearing SLAM
Prior Covariance Update

We assume static landmarks. Therefore, the function **f**(***s***,***u***,***w***) only affects the robot's location and not the landmarks.

$$\Sigma'_t = A_t \Sigma_{t-1} A_t^T + W_t Q W_t^T$$

Jacobian of the process model

$$A = \begin{bmatrix} \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial \theta} & & \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial \theta} & 0 & \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \theta} & & \\ 0 & & & I & \end{bmatrix}$$

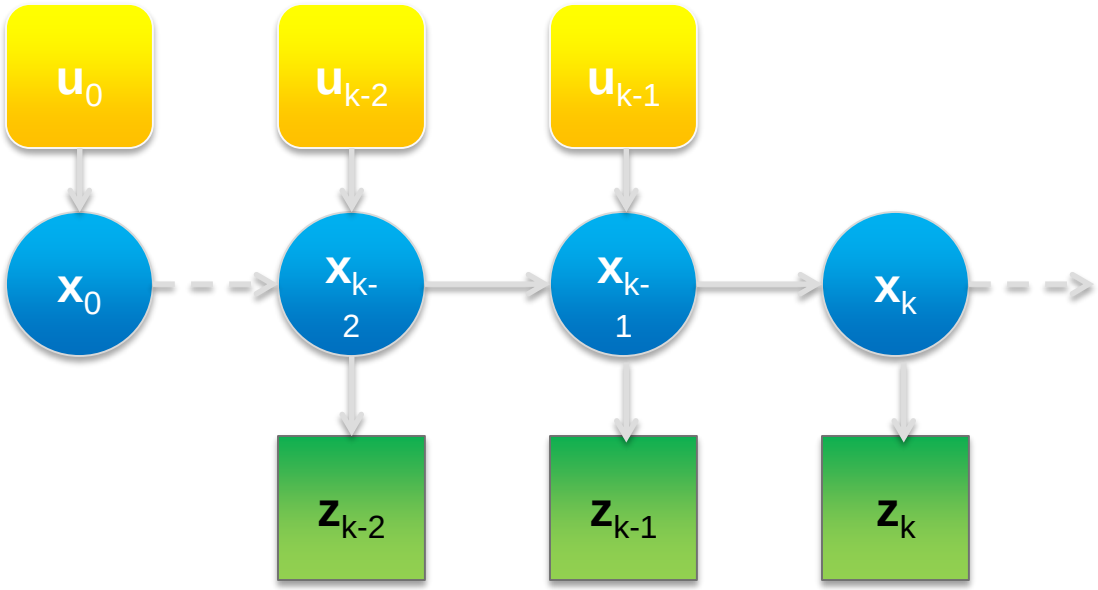
Jacobian of the robot motion

The motion of the robot does not affect the coordinates of the landmarks

2Nx2N identity

Bayes Filter

- Given a sequence of measurements $\mathbf{z}_1, \ldots, \mathbf{z}_k$
- Given a sequence of commands $\mathbf{u}_0, \ldots, \mathbf{u}_{k-1}$
- Given a sensor model $P(\mathbf{z}_k \mid \mathbf{x}_k)$
- Given a dynamic model $P(\mathbf{x}_k \mid \mathbf{x}_{k-1})$
- Given a prior probability $P(\mathbf{x}_0)$
- Find $P(\mathbf{x}_k \mid \mathbf{z}_{1:k}, \mathbf{u}_{0:k-1})$



Bayesian Localization

- Recall Bayes Theorem:

$$P(x\,|z) = \frac{P(z|x)P(x)}{P(z)}$$

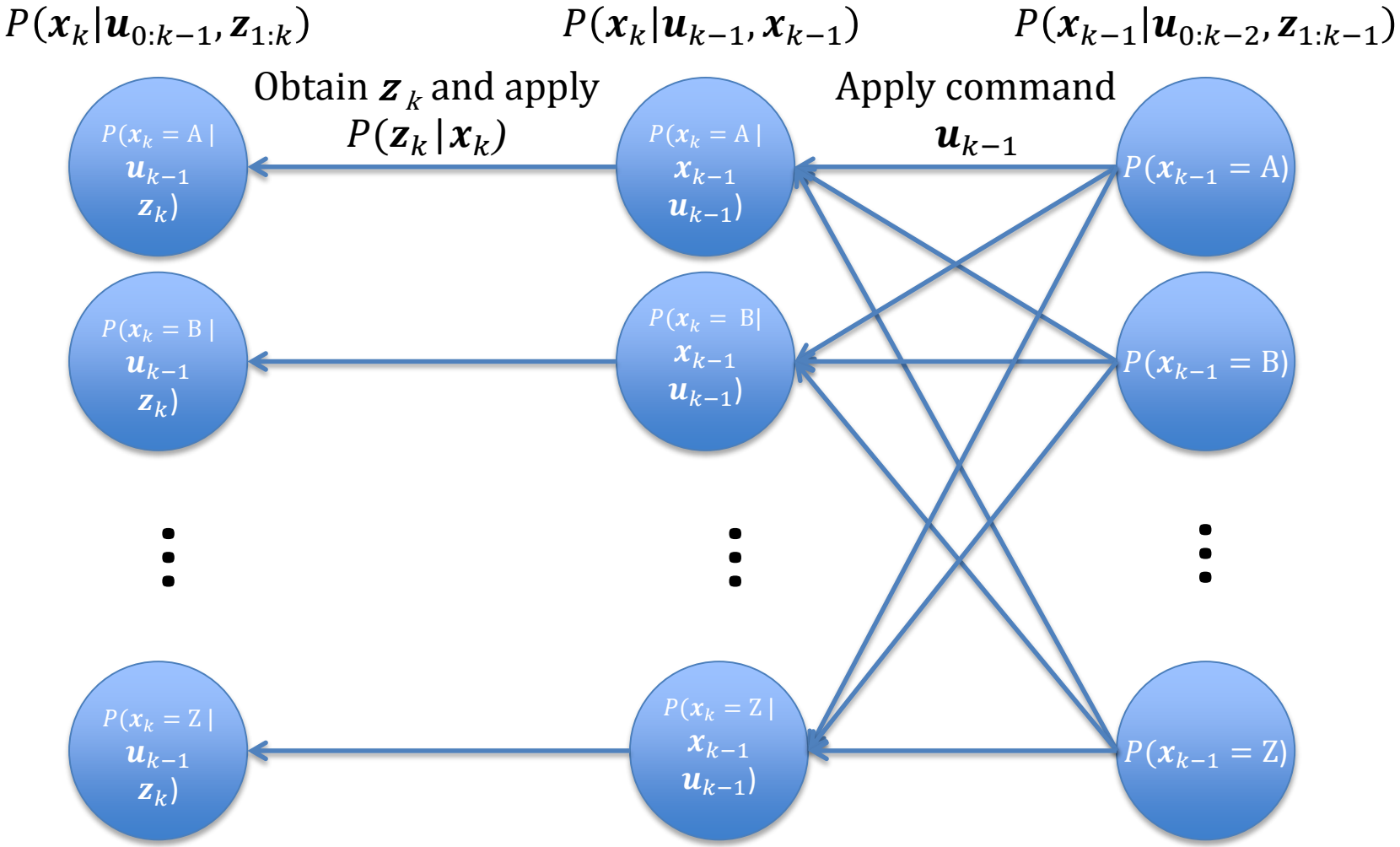
- Also remember conditional independence
- Think of $\mathbf{x}$ as the state of the robot and $\mathbf{z}$ as the data we know

$$P(\boldsymbol{x}_k|\boldsymbol{u}_{0:k-1}, \boldsymbol{z}_{1:k}) \longrightarrow \textit{Posterior Probability Distribution}$$

$$P(\boldsymbol{x}_k|\boldsymbol{u}_{0:k-1}, \boldsymbol{z}_{1:k}) = \frac{P(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{u}_{0:k-1}, \boldsymbol{z}_{1:k-1})P(\boldsymbol{x}_k|\boldsymbol{u}_{0:k-1}, \boldsymbol{z}_{1:k-1})}{P(\boldsymbol{z}_k|\boldsymbol{u}_{0:k-1}, \boldsymbol{z}_{1:k-1})}$$

$$= \underbrace{\eta_k P(\boldsymbol{z}_k|\boldsymbol{x}_k)}_{\textbf{observation}} \int_{\boldsymbol{x}_{k-1}} \underbrace{P(\boldsymbol{x}_k|\boldsymbol{u}_{k-1}, \boldsymbol{x}_{k-1})}_{\textbf{state prediction}} \underbrace{P(\boldsymbol{x}_{k-1}|\boldsymbol{u}_{0:k-2}, \boldsymbol{z}_{1:k-1})}_{\textbf{recursion}} d\boldsymbol{x}_{k-1}$$

Bayes Filter Recap



Predict Motion (Prior Distribution)

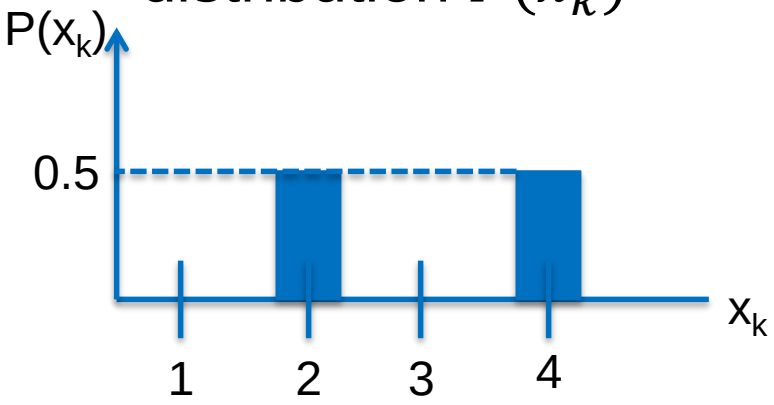
- Suppose we have $P(\boldsymbol{x}_k)$
- We have $P(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k)$
- Put together

$$P(\boldsymbol{x}_{k+1}) = \int_{\boldsymbol{x}_k} P(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k) P(\boldsymbol{x}_k) d\boldsymbol{x}_k$$

What is the probability distribution for $\boldsymbol{x}_{k+1}$ given the command $\boldsymbol{u}_k$ and all the previous states $\boldsymbol{x}_k$?

System Model

Prior probability distribution $P(x_k)$



State space X = { 1, 2, 3, 4}

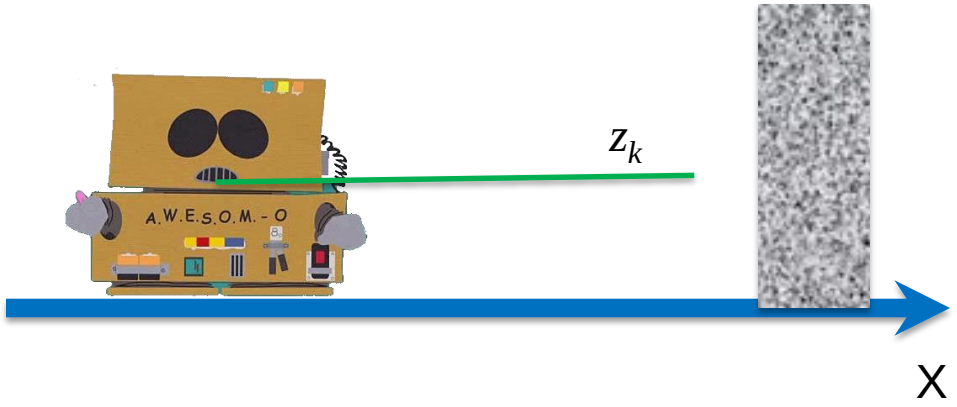
$P(x_{k+1} \mid x_k, u_k)$	$X_{k+1}=1$	$X_{k+1}=2$	$X_{k+1}=3$	$X_{k+1}=4$
$X_k=1$	0.25	0.5	0.25	0
$X_k=2$	0	0.25	0.5	0.25
$X_k=3$	0	0	0.25	0.75
$X_k=4$	0	0	0	1

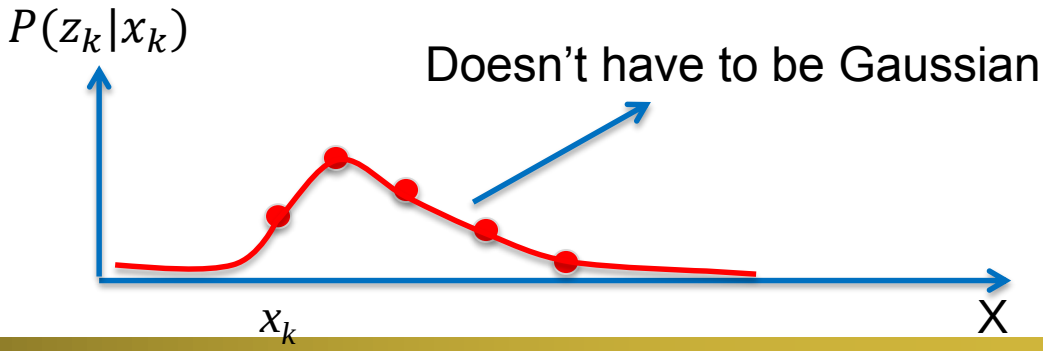
Transition matrix: The probability P($j|i$) of moving from i to j is given by $P_{i,j}$. Each row must sum to 1.

Compute $P(x_{k+1}) = \int_{x_k} P(x_{k+1}|x_k, u_k) dx_k$

Observation Model

- How likely is the measurement $\boldsymbol{z}_k$ given a state $\boldsymbol{x}_k$?
- The measurement $\boldsymbol{z}_k$ is what we get. So we have to stick by it
- However, not all states will likely produce the measurement
- We're not interested in finding the most likely state. *We want the whole distribution*





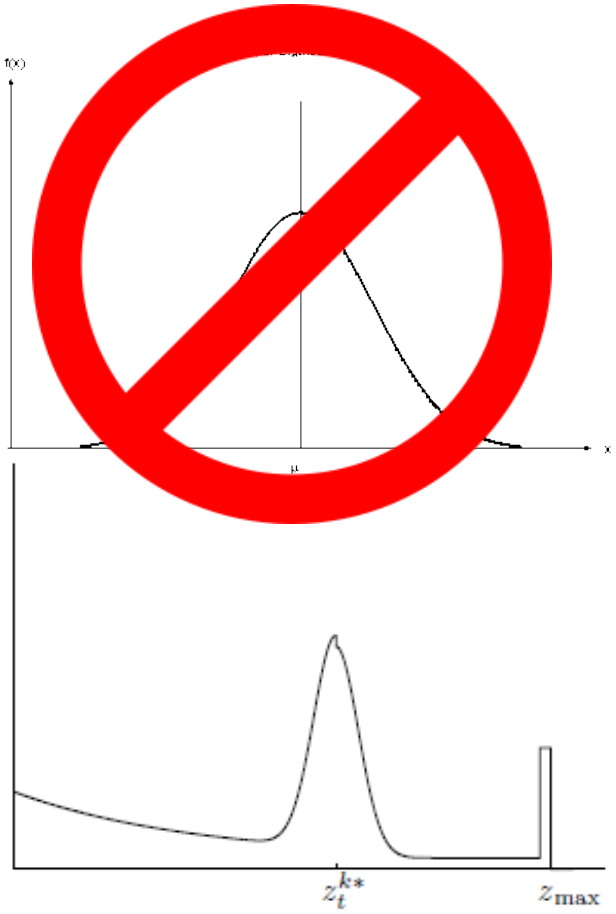
Observation Model
Beam Model

- Mixing all these cases together we get

$$P(z_k|\boldsymbol{x}_k, m) = \begin{bmatrix} w_{\text{hit}} \\ w_{\text{short}} \\ w_{\text{max}} \\ w_{\text{rand}} \end{bmatrix}^T \begin{bmatrix} P_{\text{hit}}(z_k|\boldsymbol{x}_k, m) \\ P_{\text{short}}(z_k|\boldsymbol{x}_k, m) \\ P_{\text{max}}(z_k|\boldsymbol{x}_k, m) \\ P_{\text{rand}}(z_k|\boldsymbol{x}_k, m) \end{bmatrix}$$

where the weights are parameters

$$w_{\text{hit}} + w_{\text{short}} + w_{\text{max}} + w_{\text{rand}} = 1$$

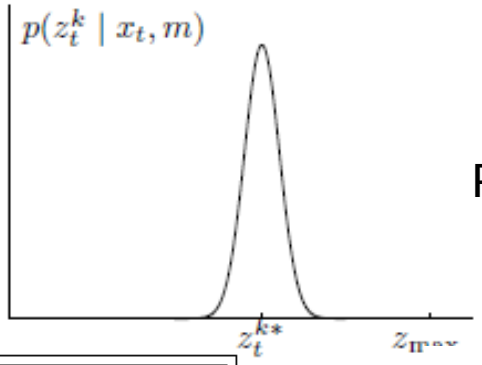


Probabilistic Robotic

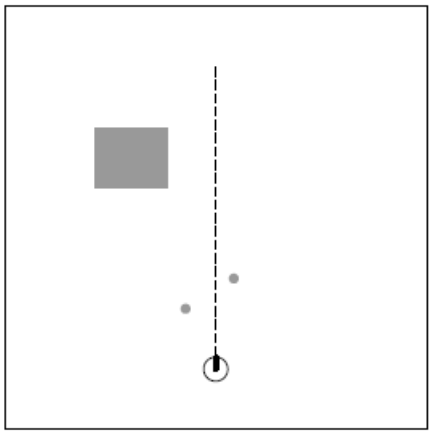
Likelihood Field Model

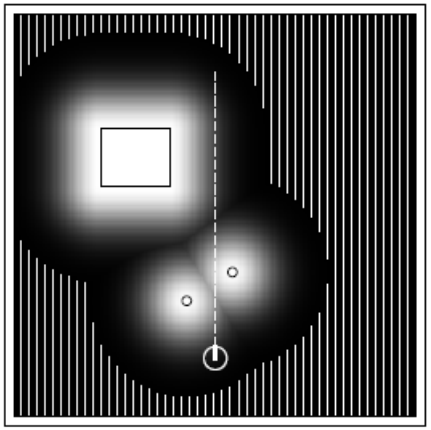
$$P_{\text{hit}}(z_k|\boldsymbol{x}_k, m) = \varepsilon_{\sigma_{hit}^2}(d^2)$$

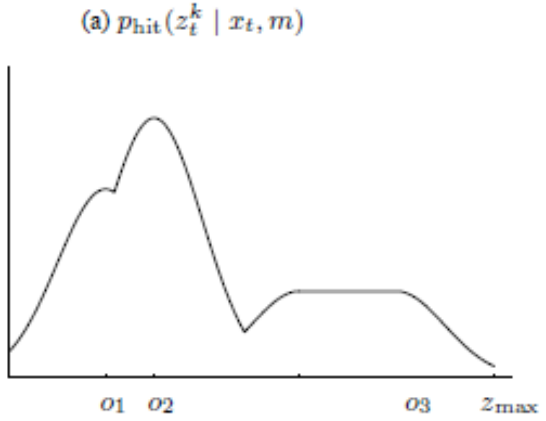
(a) Gaussian distribution p_{hit}



Probabilistic Robotic







Discrete Bayes Filter Algorithm

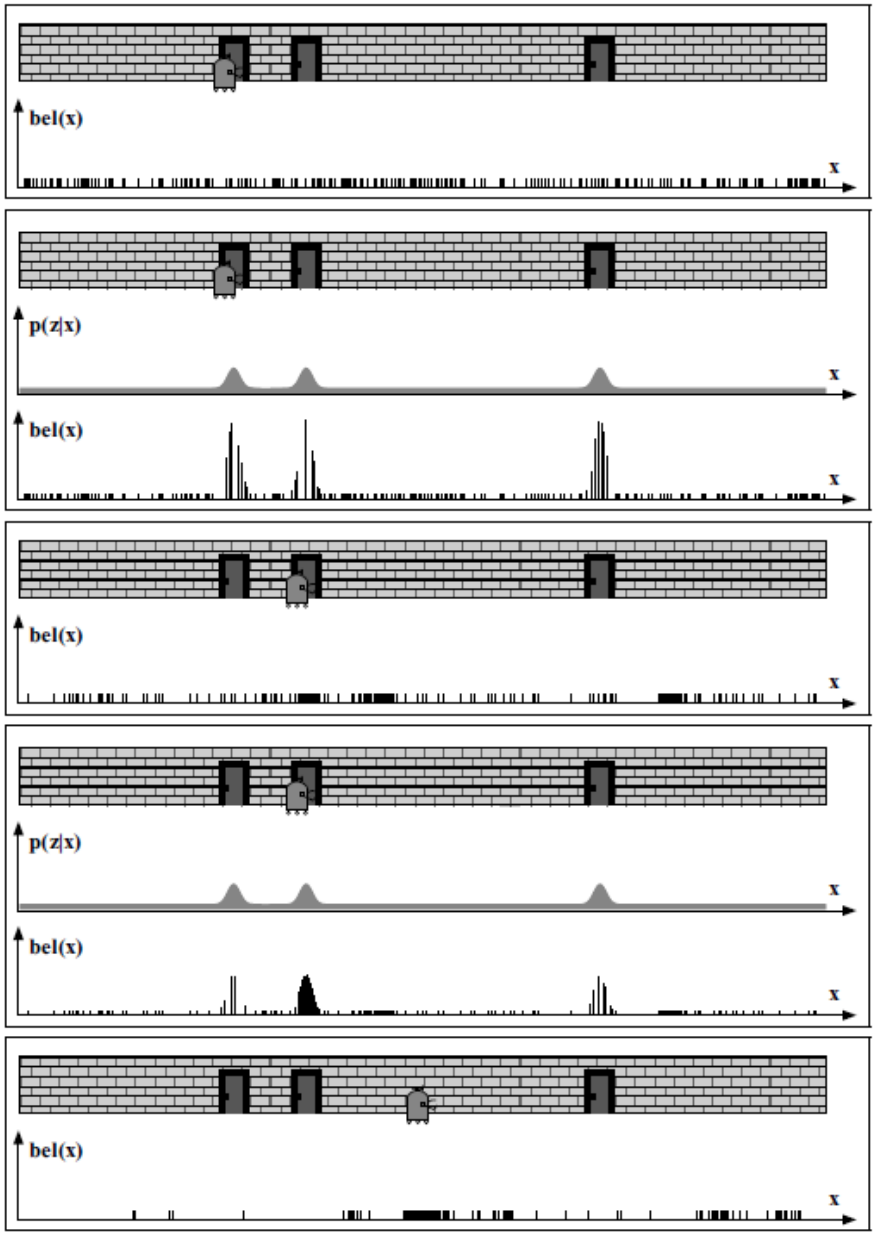
Algorithm **Discrete_Bayes_filter**($u_{0:k-1}$, $y_{1:k}$, $P(x_0)$)

1. $P(x) = P(x_0)$ (if you don't know: uniform distribution)
2. **for** i=1:k
3. **for all** states $x \in X$
4. $P'(x) = \sum_{x' \in X} P(x \mid u, x') P(x')$ Prediction given prior dist. and command
5. **end for**
6. η=0 Normalization constant
7. **for all** states $x \in X$
8. $P(x) = P(z \mid x) P'(x)$ Update using measurement
9. $\eta = \eta + P(x)$
10. **end for**
11. **for all** states $x \in X$
12. $P(x) = P(x) / \eta$ Normalize to 1
13. **end for**
14. **end for**

Particle Filter

- Computing $P(\boldsymbol{z}_k|\boldsymbol{x}_k, m)$ and $P(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k)$ is not easy
  - In practice, it is never directly computable
  - Need to propagate an entire conditional distribution, not just one state like we did with Kalman,
- Represent probability distribution by random samples
- Estimation of non-Gaussian, nonlinear processes

Particle Filter



Particle Filter Algorithm

Algorithm **Particle_filter**($u_{0:k-1}$, $y_{1:k}$, $P(x_0)$, set of N samples $\mathcal{M} = \{x_j, w_j\}$)

1. **for** $i=1:k$
2. **for** $j=1:N$
3. compute a new state x by sampling according to $P(x \mid u_{i-1}, x_j)$
4. $x_j = x$
5. **end**
6. $\eta=0$
7. **for** j=1:N
8. $w_j = P(z_i \mid x_j)$
9. η += w_j
10. **end**
11. **for** j=1:N
12. $w_j = w_j / \eta$
13. **end**
14. resample ($\mathcal{M}$) according to weights w_j
15. **end**