Estimation and Filtering

Stefano Carpin

Department of Computer Science and Engineering
School of Engineering
University of California, Merced
https://sites.ucmerced.edu/scarpin
https//robotics.ucmerced.edu

Estimation problem

- the process of determining a quantity from measurements that are *incomplete* and/or *inaccurate*.
- quantity being estimated can be a scalar or a vector
- Example: determine latitude and longitude from a set of readings coming from satellites (GPS)
- estimation often involves multiple heterogeneous measurements
  - sensor fusion

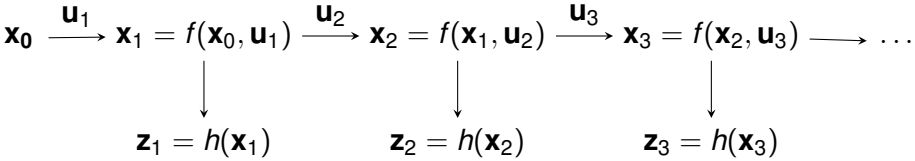
Estimation problem in robotics

- should consider both sensor readings and actions
- sensor readings decrease uncertainty, actions increase uncertainty
- classic estimation problem in robotics: localization
  - where is the robot? (e.g., x, y, ϑ)

Filtering

- an estimation problem where the quantity being estimated changes overtime
  - E.g., the state of a dynamical system, like the pose of a robot
  - more complicated than classic estimation
- solution must be updated over time
- algorithms solving this type of problems are typically called filters

Estimation problem in robotics



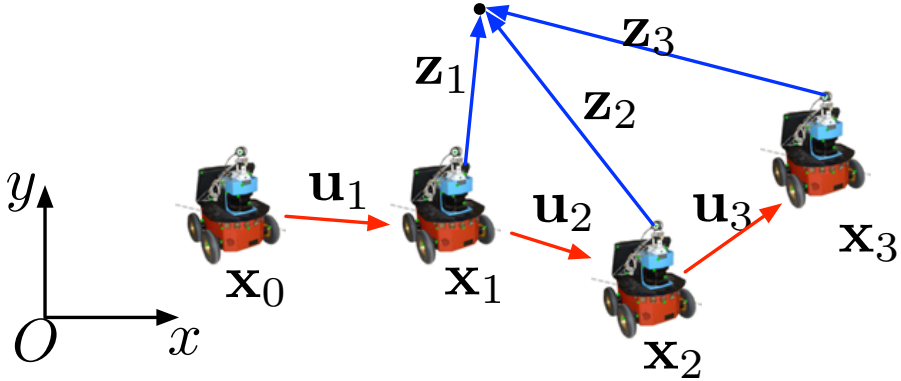
Estimation problem in robotics

$$\mathbf{x_0} \xrightarrow{\mathbf{u}_1} \mathbf{x}_1 = f(\mathbf{x}_0, \mathbf{u}_1, \zeta_1) \xrightarrow{\mathbf{u}_2} \mathbf{x}_2 = f(\mathbf{x}_1, \mathbf{u}_2, \zeta_1) \xrightarrow{\mathbf{u}_3} \mathbf{x}_3 = f(\mathbf{x}_2, \mathbf{u}_3, \zeta_3) \longrightarrow \dots$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

$$\mathbf{z}_1 = h(\mathbf{x}_1, \psi_1) \qquad \mathbf{z}_2 = h(\mathbf{x}_2, \psi_2) \qquad \mathbf{z}_3 = h(\mathbf{x}_3, \psi_3)$$

Estimation problem in robotics



Estimation

- We seek an algorithm computing the function

$$\hat{\mathbf{x}}_t = g(\mathbf{x}_0, \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_t, \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_t) \tag{1}$$

- $\hat{\mathbf{x}}_t$ is the estimate for x at time t.
- Note the dependency on time because $\mathbf{x}$ changes overt time, and so does the estimate.

Classical Vs Bayesian Approach to Estimation

- Classical Approach: quantity being estimated is an unknown constant
- Bayesian Approach: quantity being estimated is assumed to be random variable from a set Θ of possible values
  - a prior over Θ is assumed, and the objective is to produce a posterior

We will embrace the Bayesian method, i.e., our objective is to produce a posterior over Θ

Estimation error

- Difference between our estimate end the quantity being estimated:

$$\mathbf{e}_t = \hat{\mathbf{x}}_t - \mathbf{x}_t. \tag{2}$$

- both $\mathbf{x}$ and $\hat{\mathbf{x}}$ are random variables and so is the error.
- Estimate error should be *small*.
- Ideally we want to minimize this error metric:

$$\mathbb{E}_{\mathbf{x}_t, \hat{\mathbf{x}}_t}[||\mathbf{e}_t||^2] = \mathbb{E}_{\mathbf{x}_t, \hat{\mathbf{x}}_t}[||\hat{\mathbf{x}}_t - \mathbf{x}_t||^2] \tag{3}$$

- However, $\mathbf{x}_t$ is not known...
  - intuition: minimize difference between sensor readings (known) and *predicted* sensor reading (that can be computed)

Recursive estimation

Eq. (1) has a potential problem:

- As more and more data is collected over time, the estimation algorithm would depend on more and more inputs, and this could slow it down if they have to be all taken into account

Recursive estimation:

$$\hat{\mathbf{x}}_t = g(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_t, \mathbf{z}_t). \tag{4}$$

New estimate depends just on previous estimate and latest input and sensor reading.

Parametric Vs Non-Parametric Estimation

In Bayesian estimation $\mathbf{x}_t$ is a random variable. The estimation algorithm g computes a posterior PDF $\hat{\mathbf{x}}$.

- Parametric Estimation: $\hat{\mathbf{x}}$ is a random variable with a PDF determined by a finite number of parameters (e.g., Gaussian). Estimation algorithm determines the parameters (e.g., Kalman Filter)
- Non-parametric Estimation: $\hat{\mathbf{x}}$ is a random variable whose PDF is not described in closed form by a finite set of parameters. Estimation algorithm produces an approximation (e.g., particle filter)

Discrete estimation algorithms

Objective: estimate PMF of a variable with a finite alphabet $x_1, \ldots, x_n$, i.e., we want:

$$P(X = x_1) \qquad P(X = x_2) \qquad \ldots \qquad P(X = x_n)$$

Example: we may discretize the set of possible locations of a robot and we want to assign a probability to each location.

Bayes estimation

All algorithms we will see rely on Bayes rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Recall its variants, in particular those with background knowledge (i.e., conditioning on multiple events.) We assume to know $P(B|A)$ and $P(A)$, but we will not need $P(B)$.

Bayes rule in action

$$P(X = x_1|Z = z) = \frac{P(Z = z|X = x_1)P(X = x_1)}{P(Z = z)}$$

$$P(X = x_2|Z = z) = \frac{P(Z = z|X = x_2)P(X = x_2)}{P(Z = z)}$$

$$\dots$$

$$P(X = x_n|Z = z) = \frac{P(Z = z|X = x_n)P(X = x_n)}{P(Z = z)}$$

Denominator $P(Z = z)$ is the same in all expressions, and $\sum_{i=1}^{n} P(X = x_i|Z = z) = 1$.

Bayes rule in action

$$\sum_{i=1}^{n} P(X = x_i|Z = z) = \sum_{i=1}^{n} \frac{P(Z = z|X = x_i)P(X = x_i)}{P(Z = z)}$$
$$= \frac{1}{P(Z = z)} \sum_{i=1}^{n} P(Z = z|X = x_i)P(X = x_i) = 1$$

Hence:

$$P(Z = z) = \sum_{i=1}^{n} P(Z = z|X = x_i)P(X = x_i)$$

Hence we write

$$P(x_i|z) = \eta P(z|x_i)P(x_i)$$

because the normalization factor η is a constant that can be computed separately and can be assumed known.

Integrating multiple sensor readings

Assume we now have two sensor readings z_1, z_2. We assume they are independent, i.e., $P(z_1|z_2) = P(z_1)$ and $P(z_2|z_1) = P(z_2)$.

$$P(x_i|z_1, z_2) = \frac{P(z_1|x_i, z_2)P(x_i|z_2)}{P(z_1|z_2)} = \frac{P(z_1|x_i)P(x_i|z_2)}{P(z_1)}.$$

Apply Bayes rule again to $P(x_i|z_2)$:

$$P(x_i|z_1, z_2) = \frac{P(z_1|x_i)P(x_i|z_2)}{P(z_1)} = \frac{P(z_1|x_i)}{P(z_1)}\frac{P(z_2|x_i)P(x_i)}{P(z_2)} = \frac{P(z_1|x_i)}{P(z_1)}\frac{P(z_2|x_i)}{P(z_2)}P(x_i)$$

Estimation with multiple readings

$$P(x_i|z_1, z_2 \ldots, z_n) = \frac{P(z_1|x_i)}{P(z_1)} \frac{P(z_2|x_i)}{P(z_2)} \cdots \frac{P(z_n|x_i)}{P(z_n)} P(x_i)$$
$$= \left(\prod_{j=1}^{n} \frac{P(z_j|x_i)}{P(z_j)} \right) P(x_i) = \eta \left(\prod_{j=1}^{n} P(z_j|x_i) \right) P(x_i) \quad (5)$$

Each sensor reading corresponds to a scaling of the prior. Therefore the order in which they are considered does not matter.

Example: localization

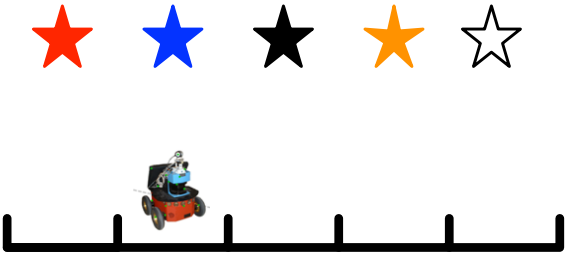


Figure: Example 6.4 in the lecture notes

Sensor readings: black, red, black, black, black
Nonuniform prior

Posterior propagation over time

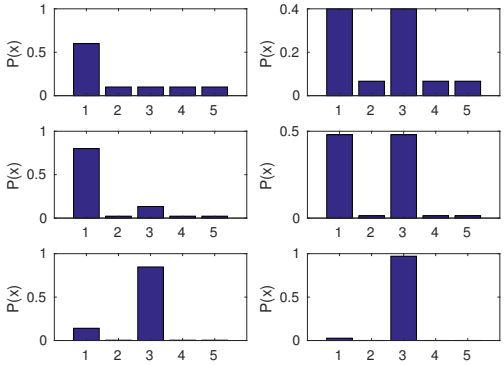


Figure: Prior (top left) and posterior (remaining diagrams) after the various sensor readings are sequentially considered (see lecture notes for numerical details).

Sensor model and motion model

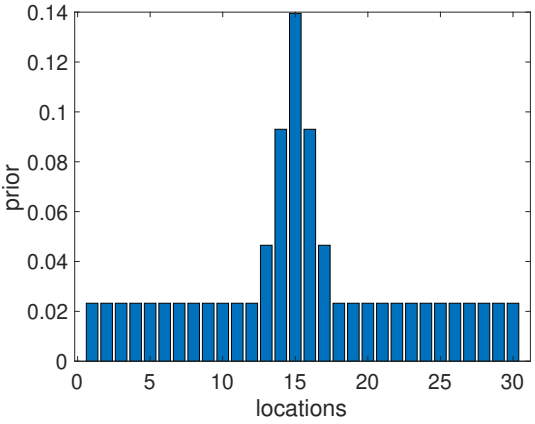
- $P(z|x)$ is the probabilistic version of the state observation equation $z = h(x)$. This is called sensor model.
- Similarly we can consider the probabilistic version of the state transition equation $x_t = f(x_{t-1}, u_t)$. This will be $P(x'|, x, u)$ and it will be called motion model.

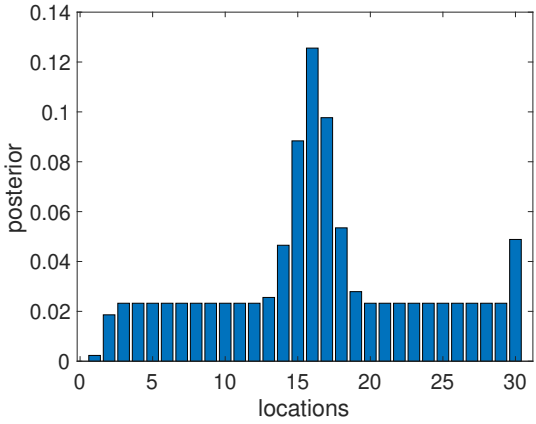
Estimation with inputs

$$P(x|u) = \sum_{i=1}^{n} P(x|y_i, u)P(y_i).$$

Where we used the total probability theorem.
Note that we assume to know the robot motion model and the prior.

Uncertainty Propagation





Estimation with multiple inputs

$$P(x|u_1, u_2) = \sum_{i=1}^{n} P(x|y_i, u_1, u_2) P(y_i|u_1).$$

Assume the motion model has the *Markov* property, i.e., the next state depends only on the current state and current input, but not the past. Therefore:

$$P(x|u_1, u_2) = \sum_{i=1}^{n} P(x|y_i, u_2) P(y_i|u_1).$$

Further expand last term:

$$P(x|u_1, u_2) = \sum_{i=1}^{n} \left[P(x|y_i, u_2) \sum_{j=1}^{n} P(y_i|y_j, u_1) P(y_j) \right].$$

Estimation with multiple inputs

If we now swap the order of the inputs we get the following expression:

$$P(x|u_2, u_1) = \sum_{i=1}^{n} \left[P(x|y_i, u_1) \sum_{j=1}^{n} P(y_i|y_j, u_2) P(y_j) \right]$$

Therefore, for the case of inputs *the order matters*.

Recursive Discrete Bayes Filter

Objective: determine

$$P(x_t|x_0, u_1, \ldots, u_t, z_1, \ldots, z_t).$$

Recall the formulation of our estimation problem:

$$\hat{x}_t = P(x_t|x_0, u_1, \ldots, u_t, z_1, \ldots, z_t).$$

Discrete Bayes Filter

$$
\begin{aligned}
\hat{x}_t &= P(x_t | x_0, u_1, \ldots, u_t, z_1, \ldots, z_t) \\
&= \frac{P(z_t | x_0, u_1, \ldots, u_t, z_1, \ldots, z_{t-1}, x_t) P(x_t | x_0, u_1, \ldots, u_t, z_1, \ldots, z_{t-1})}{P(z_t | x_0, u_1, \ldots, u_t, z_1, \ldots, z_{t-1})} \\
&= \frac{P(z_t | x_t) P(x_t | x_0, u_1, \ldots, u_t, z_1, \ldots, z_{t-1})}{P(z_t)} \\
&= \frac{P(z_t | x_t) \sum_{x_{t-1}} P(x_t | x_0, u_1, \ldots, u_t, z_1, \ldots, z_{t-1}, x_{t-1}) P(x_{t-1} | x_0, u_1, \ldots, u_t, z_1, \ldots, z_{t-1})}{P(z_t)}
\end{aligned}
$$

Discrete Bayes Filter

$$
\begin{aligned}
&= \frac{P(z_t|x_t)\sum_{x_{t-1}} P(x_t|x_{t-1},u_t)P(x_{t-1}|x_0,u_1,\ldots,u_t,z_1,\ldots,z_{t-1})}{P(z_t)} \\
&= \frac{P(z_t|x_t)\sum_{x_{t-1}} P(x_t|x_{t-1},u_t)P(x_{t-1}|x_0,u_1,\ldots,u_{t-1},z_1,\ldots,z_{t-1})}{P(z_t)} \\
&= \frac{P(z_t|x_t)\sum_{x_{t-1}} P(x_t|x_{t-1},u_t)\hat{x}_{t-1}}{P(z_t)} \\
&= \eta P(z_t|x_t)\sum_{x_{t-1}} P(x_t|x_{t-1},u_t)\hat{x}_{t-1}
\end{aligned}
$$

Discrete Bayes Filter

- $\sum_{x_{t-1}} P(x_t|x_{t-1}, u_t)\hat{x}_{t-1}$ is the prediction step and corresponds to an increase in uncertainty.
- $\eta P(z_t|x_t)$ is the correction step and corresponds to a decrease in uncertainty.
- prediction and correction are two pervasive concepts in all filtering algorithms we will see

Discrete Bayes Filter Algorithm

Data: $\hat{x}_{t-1}$, z_t, u_t
Result: Estimate $\hat{x}_t$
for $i \leftarrow 1$ **to** n **do**
  $x_i' \leftarrow \sum_{x_{t-1}} P(x_i'|x_{t-1}, u_t)\hat{x}_{t-1}$;
for $i \leftarrow 1$ **to** n **do**
  $x_i'' \leftarrow P(z_t|x_i)x_i'$;
$\frac{1}{\eta} \leftarrow \sum_{i=1}^{n} x_i''$;
for $i \leftarrow 1$ **to** n **do**
  $\hat{x}_i \leftarrow \eta x_i''$;

Algorithm 1: Discrete Bayes Filter

Probabilistic Motion Models

Data: $x_{t+1} = (x', y', \vartheta')$, $x_t = (x, y, \vartheta)$, $u_t = (v_t, v_r)$
Result: $P(x_{t+1}|x_t, u_t)$
1 $u' = (v_t', v_r') \leftarrow \text{InverseModel}(x_{t+1}, x_t)$;
2 $\delta\vartheta \leftarrow \frac{\vartheta' - \vartheta}{\Delta t} - v_r'$;
3 **return** $f_G(v_t - v_t', \alpha_1 v_t^2 + \alpha_2 v_r^2) \cdot f_G(v_r - v_r', \alpha_3 v_t^2 + \alpha_4 v_r^2) \cdot f_G(\delta\vartheta, \alpha_5 v_t^2 + \alpha_6 v_r^2)$;

Algorithm 2: Probabilistic Motion Model

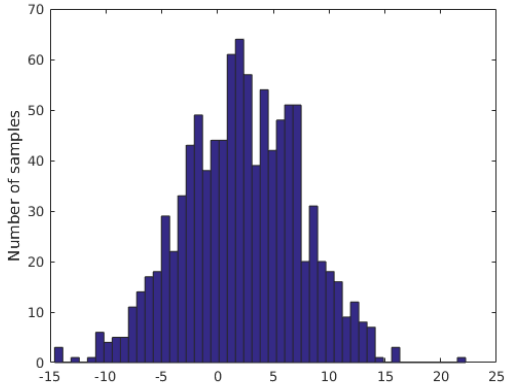
Discrete Bayes Filter Tradeoffs

- performance quadratic in the number of states
- non-parametric algorithm, so does not make any assumption about distribution being estimated
- typical exponential dependency between the dimensions of the state space and the number of states
  - can be managed for a mobile robot (three dimensional state space), but becomes quickly problematic (*curse of dimensionality*)
- computes estimates everywhere, even in regions with low probability
- typically uses uniform resolution even when inappropriate
  - adaptive resolution possible, but complicates the implementation

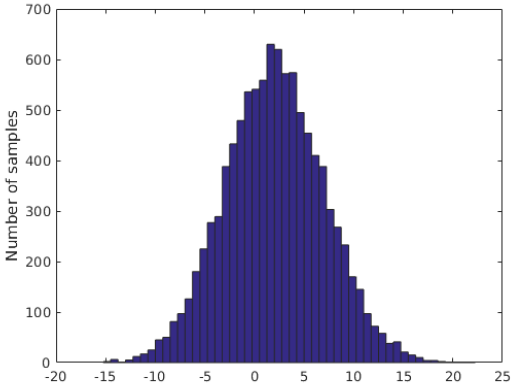
Particle Filters

- ideal for non-parametric (e.g., non Gaussian), non linear estimation problems
- idea: represent the posterior through a set of samples (called particles) drawn from it
  - more samples where the PDF is high, less samples where the PDF is small
- figuratively, one can think at every particle as a possible hypothesis for the solution to the estimation problem

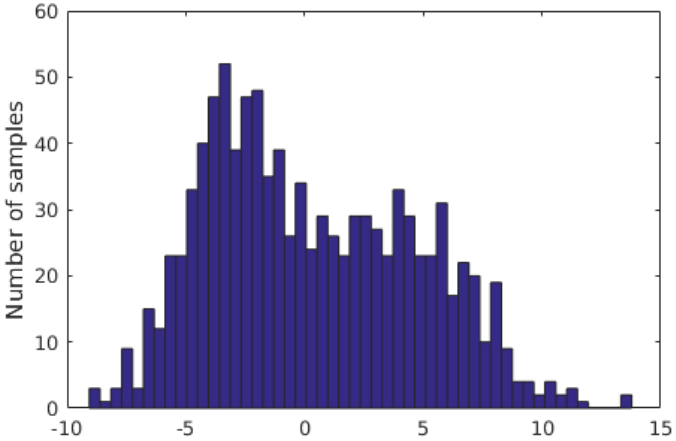
Sampling from posteriors



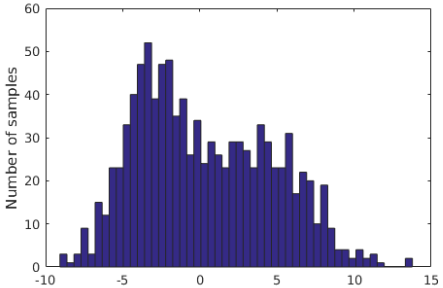
Sampling from posteriors

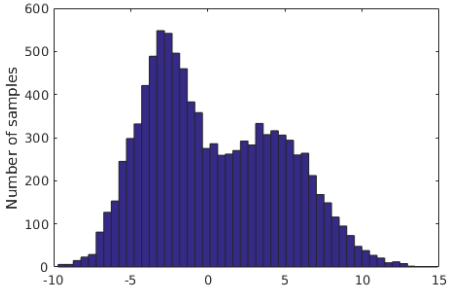


Sampling from posteriors



Sampling from posteriors





Challenges

- we can not sample from the posterior PDF
  - the very reason why we are solving an estimation problem is because we want to determine such PDF
- we can sample from other distributions we know
  - once again we need the *motion model* and *sensor model* – this time represented as PDFs

Particle filters

- posterior at time $t-1$ is represented by particles set $\mathcal{P}_{t-1}$
- recursive estimation: given $\mathcal{P}_{t-1}$, z_t, and u_t, determine new particle set $\mathcal{P}_t$
- probabilistic motion model PDF: $f(x'|x,u)$
- probabilistic sensor model PDF: $g(h|x)$
- these PDFs are easy to determine and ready available
- each particle is assigned a weight, i.e., *how well it explains the data*.
- build next set of particles retaining particles with high weight
  - can be though as *survival of the fittest*, but with much caution
- the probability of retaining a particle is proportional to its weight

Particle filters

1. Create new (intermediate) particles x_t^i, from old ones $\hat{x}_{t-1}$ (one each), sampling from the probabilistic motion model $f(x'|x, u)$
2. Assign each intermediate particle a weight based on the probabilistic sensor model $g(h|x)$
3. Sample new particles $\hat{x}_t$ from intermediate particles x_t^i

Particle Filter Algorithm

Data: $\mathcal{P}_{t-1}$, z_t, u_t
Result: New set of particles $\mathcal{P}_t$
for $i \leftarrow 1$ **to** N **do**
  $x_t^i \sim f(\hat{x}_{t-1}^i, u_t)$;
  $w_i \leftarrow g(z_t | x_t^i)$;
for $i \leftarrow 1$ **to** N **do**
  $\hat{x}_t^i \sim$ sample from $\mathcal{P}_t'$ with probability proportional to w_i;

Sample Generation

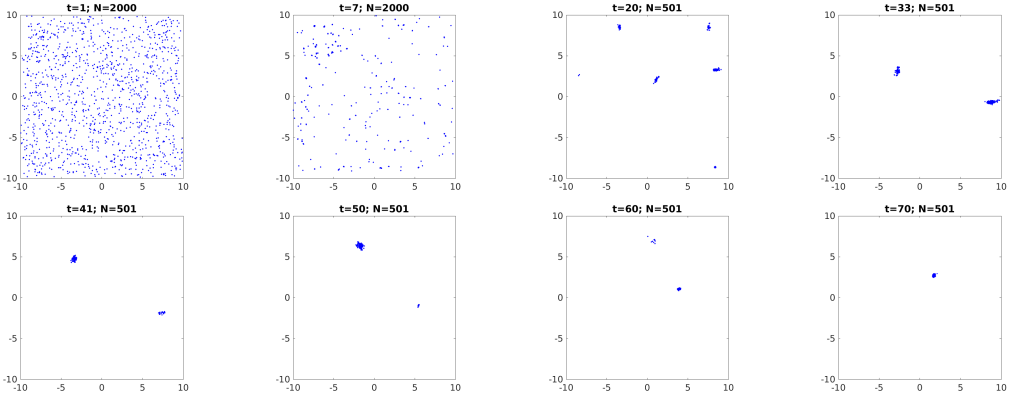
Data: $x_t = (x, y, \vartheta)$, $u_t = (v_t, v_r)$
Result: $x_{t+1} \sim f(x_t, u_t)$
1 $v_t' \leftarrow v_t + s(\alpha_1 v_t^2 + \alpha_2 v_r^2)$;
2 $v_r' \leftarrow v_r + s(\alpha_3 v_t^2 + \alpha_4 v_r^2)$;
3 $\delta\vartheta \leftarrow s(\alpha_5 v_t^2 + \alpha_6 v_r^2)$;
4 $(x', y', \vartheta') \leftarrow \text{ForwardModel}(x, y, \vartheta, v_t', v_r')$;
5 $\vartheta' \leftarrow \vartheta' + \delta\vartheta \Delta t$;
6 **return** (x', t', ϑ');

Algorithm 3: Sample generation

Implementation details

- how many particles?
  - not an easy answer. Answer is problem-dependent and requires experience
  - the number of particles is not necessarily constant but may vary
  - good reasons to have many particles and few particles
- how do we initialize particles set?
  - draw initial set from prior (e.g., Gaussian)
  - if no prior given, assume prior is a uniform (represents lack of knowledge)

Example


t=1; N=2000
t=7; N=2000
t=20; N=501
t=33; N=501
t=41; N=501
t=50; N=501
t=60; N=501
t=70; N=501

Kalman Filter

- parametric estimation technique for the case where posterior is represented by a multivariate Gaussian
  - algorithm determines mean vector and covariance matrix
- recursive estimation technique
- optimal under Gaussian and linear hypotheses
- linear refers to the state transition equation and the observation equations
- can be extended to the nonlinear case (but then no-longer optimal)

Relevant facts

- Let $\mathbf{x} \sim \mathbf{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$. Then

$$\mathbf{y} = \mathbf{A}\mathbf{x}.$$

  is also a Gaussian random vector $\boldsymbol{\mu}_y \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_x, \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^T)$
- in general $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ means $\boldsymbol{\mu}_y \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_x + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^T)$
- $\mathbf{x}$ is the sum of two independent Gaussian vectors $\mathbf{y}$ and $\mathbf{z}$

$$\mathbf{x} = \mathbf{y} + \mathbf{z}$$

  then $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_y + \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_y + \boldsymbol{\Sigma}_z)$

Relevant facts

- if $\mathbf{x}$ is a random vector (not necessarily Gaussian) and $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$
- then $\boldsymbol{\mu}_y = \mathbf{A}\mu_x + \mathbf{b}$ and $\boldsymbol{\Sigma}_y = \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^T$
  - easy to show from definitions of mean and variance
- however mean and variance in this case do not necessarily fully describe $\mathbf{y}$'s PDF

Relevant case

- Let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ and $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$ be two independent random vectors.
- Let $\mathbf{b}$ be a constant vector.
- Let $\mathbf{z} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} + \mathbf{b}$.
- then $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$
- with $\boldsymbol{\mu}_z = \mathbf{A}\boldsymbol{\mu}_x + \mathbf{B}\boldsymbol{\mu}_y + \mathbf{b}$ and $\boldsymbol{\Sigma}_z = \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^T + \mathbf{B}\boldsymbol{\Sigma}_y\mathbf{B}^T$.

Linear systems

- State transition equation:

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{v}_t$$

  where $\mathbf{v}_t$ is a Gaussian vector with 0 mean and covariance matrix $\mathbf{R}_t$
- State observation equation

$$\mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{w}_t$$

  where $\mathbf{w}_t$ is a Gaussian vector with 0 mean and covariance matrix $\mathbf{Q}_t$.
- additional independence assumptions
  - $\mathbf{v}_i$ is independent from $\mathbf{v}_j$ for each i, j
  - $\mathbf{w}_i$ is independent from $\mathbf{w}_j$ for each i, j
- $\mathbf{x}$ will be a random vector (even when $\mathbf{x}_0$ is deterministically known)

Relevant quantities for the Kalman Filter (KF)

- μ_{t-1}: mean of the estimate for $\mathbf{x}$ at time $t-1$;
- Σ_{t-1}: covariance of the estimate for $\mathbf{x}$ at time $t-1$;
- $\mathbf{u}_t$: input at time t;
- $\mathbf{z}_t$: observation at time t;
- $\mathbf{R}_t$: covariance matrix for the system evolution noise at time t;
- $\mathbf{Q}_t$: covariance matrix for the observation noise at time t.

Recall that the objective is to estimate mean and covariance of $\mathbf{x}$ at time t

KF algorithm

Prediction:

1. $\bar{\mu}_t = \mathbf{A}_t \mu_{t-1} + \mathbf{B}_t \mathbf{u}_t$
2. $\bar{\Sigma}_t = \mathbf{A}_t \Sigma_{t-1} \mathbf{A}_t^T + \mathbf{R}_t$

Correction:

1. $\mathbf{K}_t = \bar{\Sigma}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\Sigma}_t \mathbf{H}_t^T + \mathbf{Q}_t)^{-1}$
2. $\mu_t = \bar{\mu}_t + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H}_t \bar{\mu}_t)$
3. $\Sigma_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\Sigma}_t$

Note that in order to *bootstrap* the method, we need an initial estimate for the mean and covariance, i.e., we need μ_0 and Σ_0.

KF Algorithm

- KF method provides an *optimal* estimate. This optimality is measured with respect to

$$\mathbb{E}_{\mathbf{x}_t,\hat{\mathbf{x}}_t}[||\mathbf{e}_t||^2] = \mathbb{E}_{\mathbf{x}_t,\hat{\mathbf{x}}_t}[||\hat{\mathbf{x}}_t - \mathbf{x}_t||^2]$$

- if the vector $\mathbf{u}_t$ is also a Gaussian vector with mean μ_{ut} and covariance matrix Σ_{ut}, then the first equation in the prediction step becomes $\bar{\mu}_t = \mathbf{A}_t\mu_{t-1} + \mathbf{B}_t\mu_{ut}$ and the second equation becomes $\bar{\Sigma}_t = \mathbf{A}_t\Sigma_{t-1}\mathbf{A}_t^T + \mathbf{B}_t\Sigma_{ut}\mathbf{B}_t^T + \mathbf{R}_t$. The correction step instead does not change.
- in a practical scenario it could be that inputs and observations do not alternate one to one, e.g., the system could receive an observation every three inputs or multiple observations in a row. In such case the step corresponding to the missing data is simply skipped, i.e., one would perform multiple predictions in sequence or multiple corrections in sequence.
- the term $(\mathbf{z}_t - \mathbf{H}_t\bar{\mu}_t)$ is called *innovation*.

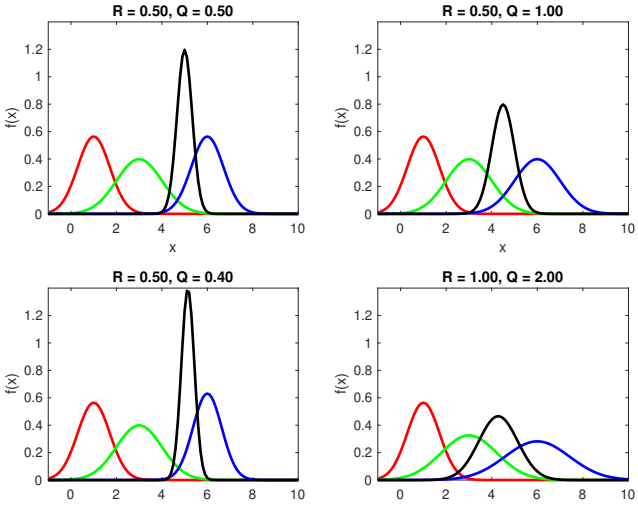
Unidimensional example

Let us consider a unidimensional case governed by the following simple equations:

$$x_t = x_{t-1} + u_t$$
$$z_t = x_t$$

Let v_t and w_t have constant covariance, as specified in the following. The first input u_1 is 2 and the first sensor reading z_1 is 6. Finally, let $x_0 \sim \mathcal{N}(1, 0.5)$.

Unidimensional example



A robot with two sensors

We now consider a small variant of the previous example, where the robot is now equipped with two sensors returning the value of the unidimensional state. This is modeled as follows:

$$x_t = x_{t-1} + u_t$$
$$z_t = \mathbf{H}x_t$$

$$\mathbf{H} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.5 \end{bmatrix}$$

Let $x_0 \sim \mathcal{N}(1, 0.5)$, let $u_1 = 1$ and $Q = 1$. Moreover, let the first sensor reading be $\mathbf{z}_1 = [3 \quad 3]'$.

A robot with two sensors

Prediction: $\bar{\mu}_1 = 2$ and $\bar{\sigma}_1^2 = 1$
Correction:

$$\mathbf{K}_1 = \bar{\sigma}_1^2 \mathbf{H}^T (\mathbf{H} \bar{\sigma}_1^2 \mathbf{H}^T + \mathbf{Q})^{-1} = [0.5263 \quad 0.2105]$$

$$\mu_1 = \bar{\mu}_1 + \mathbf{K}_1 (\mathbf{z}_1 - \mathbf{H} \bar{\mu}_1) = 2.3158$$

$$\sigma_1 = (1 - \mathbf{K}\mathbf{H}) \bar{\sigma}_1^2 = 0.0526$$

Note that the Kalman gain gives more weight to the first sensor.

Extended Kalman Filter

In most cases, state transition equations and sensor functions are non-linear:

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{v}_t$$

$$\mathbf{z}_t = h_t(\mathbf{x}_t) + \mathbf{w}_t$$

In this case we can linearize those relationship and evaluated then at the current estimate:

$$\mathbf{A}_t = \frac{\partial f_t(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \quad \mathbf{B}_t = \frac{\partial f_t(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \quad \mathbf{H}_t = \frac{\partial h_t(\mathbf{x})}{\partial \mathbf{x}}.$$

Extended Kalman Filter

Data: $\mu_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t$
Result: μ_t, Σ_t
/* Prediction */
1 $\bar{\mu}_t = f_t(\mu_{t-1}, \mathbf{u}_t)$;
2 $\bar{\Sigma}_t = \mathbf{A}_t \Sigma_{t-1} \mathbf{A}_t^T + \mathbf{R}_t$;
/* Update */
3 $\mathbf{K}_t = \bar{\Sigma}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\Sigma}_t \mathbf{H}_t^T + \mathbf{Q}_t)^{-1}$;
4 $\mu_t = \bar{\mu}_t + \mathbf{K}_t(\mathbf{z}_t - h_t(\bar{\mu}_t))$;
5 $\Sigma_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\bar{\Sigma}_t$;
6 **return** μ_t, Σ_t;

Algorithm 4: Extended Kalman Filter

EKF for robot localization

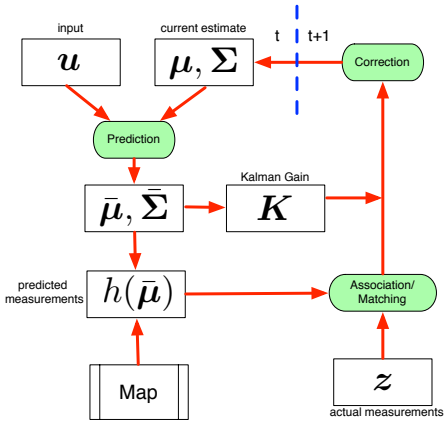


Figure: EKF localization cycle

Pose Tracking with EKF

Motion Model:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{v}_t = \begin{bmatrix} x_{t-1} + l_t \cos(\theta_{t-1}) \\ y_{t-1} + l_t \sin(\theta_{t-1}) \\ \theta_{t-1} + r_t \end{bmatrix} + \mathbf{v}_t.$$

$\mathbf{v}_t$ is a zero mean Gaussian vector with covariance matrix $\mathbf{R}$

$$\mathbf{R} = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$$

The initial state is Gaussian distributed with mean $\boldsymbol{\mu}_0 = [0\ 0\ 0]^T$ and covariance

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$$

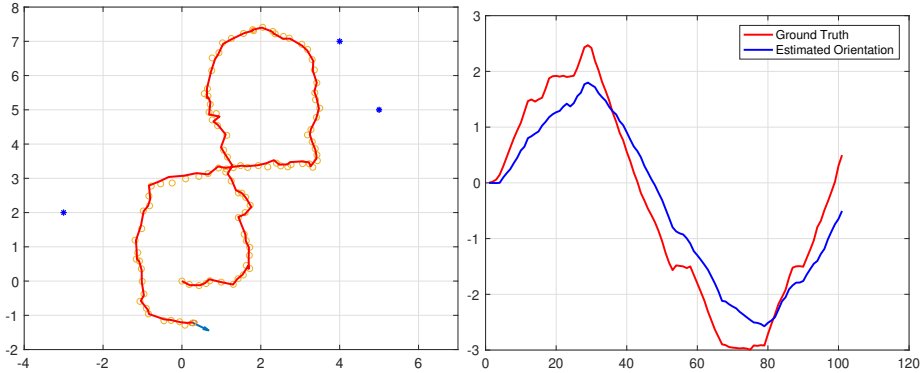
Pose Tracking with EKF

Sensor Model (distance from three distinguishable landmarks)

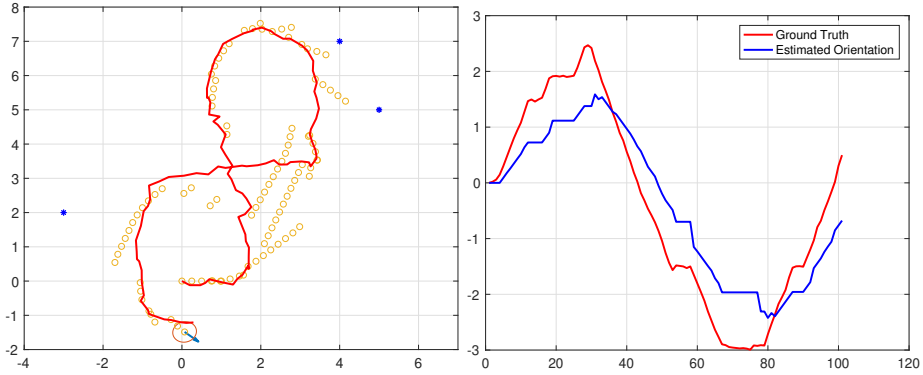
$$\mathbf{z}_t = h_t(\mathbf{x}_t) + \mathbf{w}_t = \begin{bmatrix} \sqrt{(x_L^1 - x_t)^2 + (y_L^1 - y_t)^2} \\ \sqrt{(x_L^2 - x_t)^2 + (y_L^2 - y_t)^2} \\ \sqrt{(x_L^3 - x_t)^2 + (y_L^3 - y_t)^2} \end{bmatrix} + \mathbf{w}_t$$

$$\mathbf{Q} = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.001 \end{bmatrix}.$$

Estimation Error – Continuous Measurement

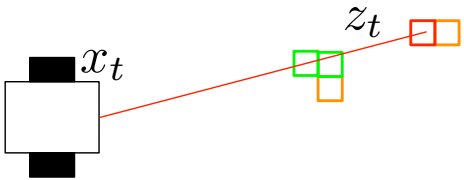


Estimation Error – Sporadic Measurement



Mapping with known poses

Mapping is another estimation problem: assign an occupancy probability to each grid cell.



Assumptions: available sensors model; all cells independent; known robot pose.

Mapping with known poses – derivation

$$p(m_i = 1 | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_n)$$

$$\begin{aligned}
p(m_i | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_n) = \\
&= \frac{p(z_n | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1}, m_i) p(m_i | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1})}{p(z_n | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1})} \\
&= \frac{p(z_n | x_n, m_i) p(m_i | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1})}{p(z_n | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1})} \\
&= \frac{p(m_i | x_n, z_n) p(z_n | x_n) p(m_i | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1})}{p(m_i) p(z_n | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1})} \\
&= \frac{p(m_i | x_n, z_n) p(z_n | x_n) p(m_i | x_1, x_2, \ldots, x_{n-1}, z_1, z_2, \ldots, z_{n-1})}{p(m_i) p(z_n | x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1})}
\end{aligned}$$

Mapping with known poses – derivation

Next compute the probability of the cell being free:

$$p(\neg m_i|x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_n) = \\ = \frac{p(\neg m_i|x_n, z_n)p(z_n|x_n)p(\neg m_i|x_1, x_2, \ldots, x_{n-1}, z_1, z_2, \ldots, z_{n-1})}{p(\neg m_i)p(z_n|x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{n-1})}$$

Take the ratio between the two estimates and cancel out terms,

$$\frac{p(m_i|x_1, \ldots, x_n, z_1, \ldots, z_n)}{p(\neg m_i|x_1, \ldots, x_n, z_1, \ldots, z_n)} = \frac{p(m_i|x_n, z_n)p(\neg m_i)p(m_i|x_1, \ldots, x_{n-1}, z_1, \ldots, z_{n-1})}{p(m_i)p(\neg m_i|x_n, z_n)p(\neg m_i|x_1, \ldots, x_{n-1}, z_1, \ldots, z_{n-1})}$$

Mapping with known poses – derivation

$p(m_i) = 1 - p(\neg m_i)$ Introduce odds of a binary random variable:

$$Odds(x) = \frac{p(x)}{1 - p(x)}.$$

Note it is invertible if $p(x) \neq 1$. Previous ratio becomes:

$$Odds(p(m_i|x_1, \ldots, x_n, z_1, \ldots, z_n)) = \frac{Odds(p(m_i|x_n, z_n)Odds(p(m_i|x_1, \ldots, x_{n-1}, z_1, \ldots, z_{n-1}))}{Odds(m_i)}$$

Mapping with known poses – derivation

Introduce log for numerical stability:

$$\begin{aligned}
&\log Odds(p(m_i|x_1,\ldots,x_n,z_1,\ldots,z_n)) = \\
&\quad = \log \frac{Odds(p(m_i|x_n,z_n)Odds(p(m_i|x_1,\ldots,x_{n-1},z_1,\ldots,z_{n-1}))}{Odds(m_i)} \\
&\quad = \log Odds(p(m_i|x_n,z_n)) + \log Odds(p(m_i|x_1,\ldots,x_{n-1},z_1,\ldots,z_{n-1})) - \log Odds(m_i)
\end{aligned}$$

Mapping with known poses

Data: $x_n = (x, y, \vartheta)$, z_n, l_{n-1}, m_i
Result: l_n
if *m_i in perceptual range of x_n, z_n* **then**
    $l_n^i \leftarrow \log Odds(m_i|x_n, z_n) + l_{n-1}^i - \log Odds(m_i)$;
else
    $l_n^i \leftarrow l_{n-1}^i$;
return l_n;

Algorithm 5: Mapping with known poses