

# Recursive state estimation: a cookbook

Tim Bretl

Department of Aerospace Engineering  
Beckman Institute for Advanced Science and Technology  
University of Illinois at Urbana-Champaign

AE498MPA  
September 24, 2007

# The discrete Bayes' filter

- In class we derived the following recursive state estimator for finite (discrete) state spaces, from the basic axioms of probability:

$$p(x_t|y_0, \dots, y_t) = \eta p(y_t|x_t) \sum_{x_{t-1}} p(x_t|x_{t-1})p(x_{t-1}|y_0, \dots, y_{t-1})$$

- This filter allows us to keep track of the **posterior**  $p(x_t|y_0, \dots, y_t)$  by processing measurements one by one, rather than all at once. The posterior represents our **belief** about the state  $x_t$ .
- In practice, however, we are often interested in continuous state spaces. Here we will consider three filters that work in continuous state spaces: the **histogram filter**, the **kalman filter**, and the **particle filter**. All three can be derived from the discrete Bayes' filter.
- For an example, see the matlab script "OneDMotion.m" on the wiki.

# The histogram filter (1)

- The state at each time is a continuous random variable  $x_t \in \mathcal{X}_t$ .
- Partition the state space  $\mathcal{X}_t$  into  $m$  bins of equal volume  $V_t$ :

$$\mathcal{X}_t = \mathbf{x}_t^1 \cup \dots \cup \mathbf{x}_t^m.$$

- Define the midpoint of each bin  $\mathbf{x}_t^i$  by  $x_t^i$ .
- Approximate the posterior probability of each bin  $\mathbf{x}_t^i$  by

$$p_t^i = p(\{x_t \in \mathbf{x}_t^i\} | y_1, \dots, y_t) \approx \eta V_t p(x_t^i | y_1, \dots, y_t)$$

where the normalizer is given by

$$\eta = \sum_{i=1}^m V_t p(x_t^i | y_1, \dots, y_t).$$

- By convention we define the prior (i.e., the “initial posterior”) by

$$p_0^i = p(\{x_0 \in \mathbf{x}_0^i\}) \approx \eta V_0 p(x_0^i).$$

## The histogram filter (2)

- Now we can apply the discrete Bayes' filter to estimate the state. For convenience (and clarity), we divide this filter into two parts at each time step:
  1. Time update (prediction):

$$\bar{p}_t^i = \sum_{k=1}^m p(x_t^i | x_{t-1}^k) p_{t-1}^k \quad \text{for all } i = 1, \dots, m$$

2. Measurement update (correction):

$$p_t^i = \eta p(y_t | x_t^i) \bar{p}_t^i \quad \text{for all } i = 1, \dots, m$$

As usual,  $\eta$  is a normalizer in the measurement update.

# The Kalman filter (1)

- We assume that both the measurement model and the state transition model are **linear** and subject to **gaussian** noise:

$$\begin{aligned}x_{t+1} &= Ax_t + Bu_t + v_t \\y_t &= Cx_t + w_t\end{aligned}$$

In this expression,  $u_t$  is a chosen **input** (which may itself be a function of the posterior distribution),  $v_t \sim \mathcal{N}(0, \Sigma_v)$  is **process noise** and  $w_t \sim \mathcal{N}(0, \Sigma_w)$  is **measurement noise**.

- We also assume that  $x_0$  is normally distributed, so  $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ .
- Given these assumptions, the posterior distribution  $p(x_t | y_0, \dots, y_t)$  is always gaussian. (This is quite remarkable.)
- So all the Kalman filter has to do is keep track of the **mean**  $\mu_t$  and the **covariance**  $\Sigma_t$  of the posterior distribution.

# The Kalman filter (2)

- We are given  $\mu_0$  and  $\Sigma_0$ . The first measurement is taken at time  $t = 1$ . At every time step  $t$  thereafter, we do the following:

$$\left. \begin{aligned} \bar{\mu}_t &= A\mu_{t-1} + Bu_t \\ \bar{\Sigma}_t &= A\Sigma_{t-1}A^T + \Sigma_v \end{aligned} \right\} \text{time update (prediction)}$$

$$\left. \begin{aligned} K_t &= \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + \Sigma_w)^{-1} \\ \mu_t &= \bar{\mu}_t + K_t (y_t - C\bar{\mu}_t) \\ \Sigma_t &= (I - K_t C) \bar{\Sigma}_t \end{aligned} \right\} \text{measurement update (correction)}$$

- This involves only matrix multiplication and inversion. It is fast, and in many cases works well. In practice, the values of  $\Sigma_0$ ,  $\Sigma_v$ , and  $\Sigma_w$  should be considered as **tuning parameters** that can be adjusted to improve the performance of the filter.

# The particle filter (1)

- We begin by sampling  $m$  particles

$$x_0^1, \dots, x_0^m \sim p(x_0).$$

- Then, at each time  $t$ , we do the following:
  1. **Simulate:** propagate each sample according to the dynamic model:

$$\bar{x}_t^i \sim p(x_t | x_{t-1}^i) \quad \text{for all } i = 1, \dots, m$$

2. **Weight:** see how likely each particle is, given the measurement:

$$w_t^i = p(y_t | \bar{x}_t^i) \quad \text{for all } i = 1, \dots, m$$

3. **Resample:** sample the  $M$  particles *again*, according to their weights:

for  $i = 1$  to  $m$

sample  $k \in \{1, \dots, m\}$  with probability  $w_t^k$

let  $x_t^i = \bar{x}_t^k$

## The particle filter (2)

- Our belief about the state is encoded by the  $m$  particles.
- For example, the **sample mean** of the posterior is

$$\mu_t = \frac{1}{m} \sum_{i=1}^m x_t^i$$

and the **sample covariance** is

$$\Sigma_t = \frac{1}{m-1} \sum_{i=1}^m (x_t^i - \mu_t)(x_t^i - \mu_t)^T.$$