Bayes Filter, Kalman Filter and Particle Filter

Table of Contents

Motivation

Suppose that a robot moves along a line. Our objective is to estimate its location at each time.

- Suppose we have prior knowledge about the motion of the robot. We can use it to predict the location. However, the real behavior of the robot may deviate from that ideal prediction.

- Suppose we have observation data from sensors. These data may be the locations or the velocities of the robot at each time. We can use them to estimate the location. However, these data may be noisy.

Motivation

Suppose that a robot moves along a line. Our objective is to estimate its location at each time.

- Can we take into account the motion model and observation data at the same time for prediction?
- Is there a good mechanism so that we can strike a good balance between the weight of motion model and observation data under the uncertainty of motion noise and sensor noise?

Motivation

Suppose that a robot moves along a line. Our objective is to estimate its location at each time.

- Can we take into account the motion model and observation data at the same time for prediction? $\rightarrow$ state prediction and update
- Is there a good mechanism so that we can strike a good balance between the weight of motion model and observation data under the uncertainty of motion noise and sensor noise? $\rightarrow$ Kalman gain

Methodology

1. State Prediction : we predict the state value (e.g. the location) at time t, based on its estimated value at time t-1 using a state propagation/extrapolation model.

2. State Update : we compare the prediction from state prediction with the observation data (both for time t) and update/adjust its final estimate for time t based on the difference.

Essentially, we use state prediction and update to combine our knowledge about how the state should mathematically evolve and measurement values. Both steps are crucial for good estimation.

Mathematical Problem Formulation

- Variables
  1. z : sensor observation (e.g. antenna measurements)
  2. u : control data (e.g. external force)
  3. x : state of the system (e.g. location and velocity)

- Question : How likely is the state of the system to be x, given the sensor observation z and the control data u?

- Formal Objective : find $bel(x_t) := p(x_t|z_{1:t}, u_{1:t})$
  $\Rightarrow$ find the belief about the current state at time t, $bel(x_t)$, given the set of all observations, $z_{1:t}$, and the set of all controls, $u_{1:t}$

Derivation

1. Bayes rule

$$
\begin{aligned}
bel(x_t) = p(x_t|z_{1:t}, u_{1:t}) &= p(x_t|z_t, z_{1:t-1}, u_{1:t}) \\
&= \frac{p(x_t, z_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})} \\
&= \eta p(z_t|x_t, z_{1:t-1}, u_{1:t}) p(x_t|z_{1:t-1}, u_{1:t})
\end{aligned}
$$

2. Markov assumption

$$
bel(x_t) = \eta p(z_t|x_t) p(x_t|z_{1:t-1}, u_{1:t})
$$

Since $z_{1:t-1}$ and $u_{1:t}$ are used to produce x_t and we have already known x_t, we do not need $z_{1:t-1}$ and $u_{1:t}$

Derivation

3. The law of total probability

$$bel(x_t) = \eta p(z_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t}) p(x_{t-1}|z_{1:t-1}, u_{1:t}) dx_{t-1}$$

4. Markov assumption

$$bel(x_t) = \eta p(z_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}, u_t) p(x_{t-1}|z_{1:t-1}, u_{1:t-1}) dx_{t-1}$$

- Since $z_{1:t-1}$ and $u_{1:t-1}$ are used to produce x_{t-1} and we have already known x_{t-1}, we do not need $z_{1:t-1}$ and $u_{1:t-1}$
- To produce x_{t-1}, only $z_{1:t-1}$ and $u_{1:t-1}$ are necessary

Bayes Filter

$$bel(x_t) = \eta p(z_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$$

1. State Prediction

$$\overline{bel}(x_t) = \int_{x_{t-1}} p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$$

- We take into account the belief about the previous state of the system and the control data (action to take the system from the previous state to the current one), to get the belief about the current state.
- Motion model : $p(x_t|x_{t-1}, u_t)$

Bayes Filter

$$bel(x_t) = \eta p(z_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$$

2. State Update

$$bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t)$$

- We rely on the predicted belief and the sensor observation to estimate the current updated belief.
- Observation model : $p(z_t|x_t)$

Bayes Filter

- The Bayes filter is a general recursive update framework that allows us to estimate the state of the system, based on the previous state, the control data (current motion), and the current sensor observation, using the motion and observation models.

- In practice, directly computing the integral can be computationally expensive or even infeasible for high-dimensional systems.

Bayes Filter

- The Kalman filter serves as a special case of the Bayes filter for linear dynamic systems with Gaussian noise. It assumes that both the motion and observation models are linear and that the noise follows a Gaussian distribution with known statistics.

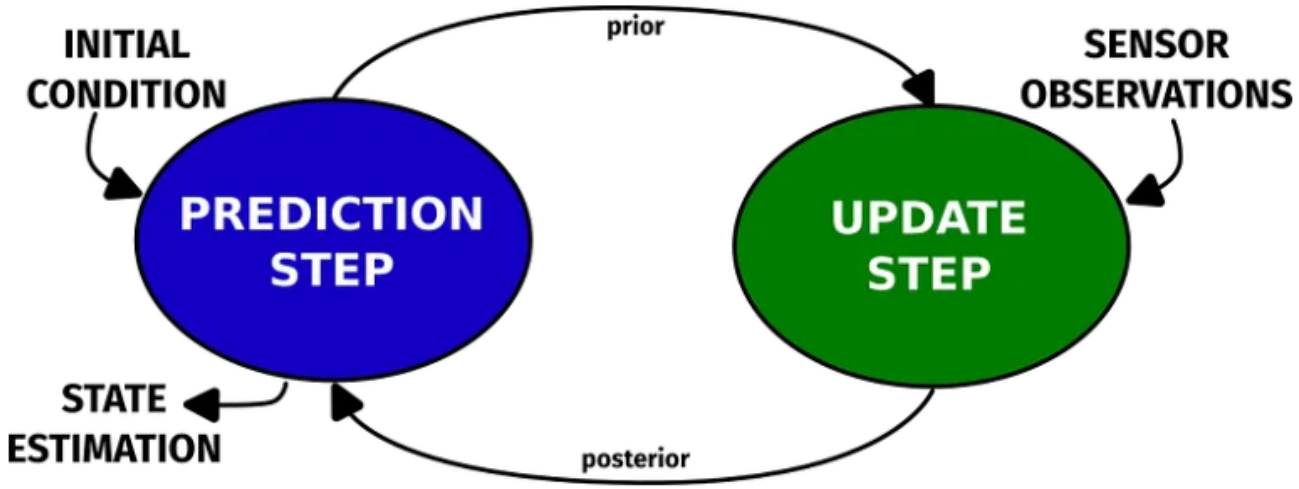
- With these two assumptions, the Kalman filter can estimate the state of the system more efficiently

- There are other variations of the Bayes filter that extend the Kalman filter to handle nonlinear systems or non-Gaussian noise. For example, the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) are commonly used for nonlinear systems.

Table of Contents

Methodology

Kalman Filter = Bayes Filter + Linear model assumption + Gaussian distribution assumption



Linear Model Assumption

1. Motion model : $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$
   - A_t : describes how the system state evolves from t-1 to t without controls or noise. It is a $(n \times n)$ matrix, where n is the dimension of the state vector.
   - B_t : describes how the control u_t changes the state from t-1 to t. It is an $(n \times \ell)$ matrix, where ℓ is the dimension of the control command vector.
   - ϵ_t : a normal random vector with mean 0 and covariance R_t, representing the process (motion) noise.

Linear Model Assumption

2. Observation model : $z_t = C_t x_t + \delta_t$
   - B_t : describes how to map the state x_t to an observation z_t. It is a $(k \times n)$ matrix, where k is the dimension of the observation vector.
     $\Rightarrow$ What should I expect to observe given that the world is in its current state?
   - δ_t : a normal random vector with mean 0 and covariance Q_t, representing the measurement noise. Note that ϵ_t and δ_t are assumed to be independent.

Gaussian Distribution Assumption

1. Motion model : ϵ_t is normally distributed

$$p(x_t|x_{t-1}, u_t) = det(2\pi R_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)\}$$

2. Observation model : δ_t is normally distributed

$$p(z_t|x_t) = det(2\pi Q_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1} (z_t - C_t x_t)\}$$

Kalman Filter

1. Putting all together into the two equations of the Bayes filter.

$$\overline{bel}(x_t) = \int_{x_{t-1}} p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$$
$$bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t)$$

2. Both equations involve Gaussian distributions and the combination of Gaussian distributions. $\Rightarrow$ $\overline{bel}(x_t)$ and $bel(x_t)$ are also Gaussian distributions.
3. Since a Gaussian distribution can be fully characterized by its mean vector and covariance matrix, what we need is the mean vector and covariance matrix of $bel(x_t)$ at each iteration.

Kalman Filter

Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

$$\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$$
$$\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$$

$$K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$$
$$\textit{return}\ \mu_t, \Sigma_t$$

Kalman Gain

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$$

- Case 1 : $Q_t = 0 \Rightarrow$ perfect sensors
  $\Rightarrow K_t = C_t^{-1}$
  $\Rightarrow \mu_t = C_t^{-1} z_t$, $\Sigma_t = 0$
  $\Rightarrow$ completely trust the observation data
- Case 2 : $Q_t \to \infty \Rightarrow$ terrible sensors
  $\Rightarrow K_t = 0$
  $\Rightarrow \mu_t = \overline{\mu}_t$, $\Sigma_t = \overline{\Sigma}_t$
  $\Rightarrow$ completely discard the observation data
- Thus, the Kalman gain can control the weight of the prediction by the motion model and the observation data from the sensors.

Kalman Filter Example

- Objective : estimate the true location of a moving robot
- Variables
  - z_t : the sensor measurement
  - x_t : the state of the system, represented as $x_t = [\mathbf{x}_t, \dot{\mathbf{x}}_t, \mathbf{y}_t, \dot{\mathbf{y}}_t]$ and denoting the location and velocity of the robot in the 2D plane
  - $u_t = 0$

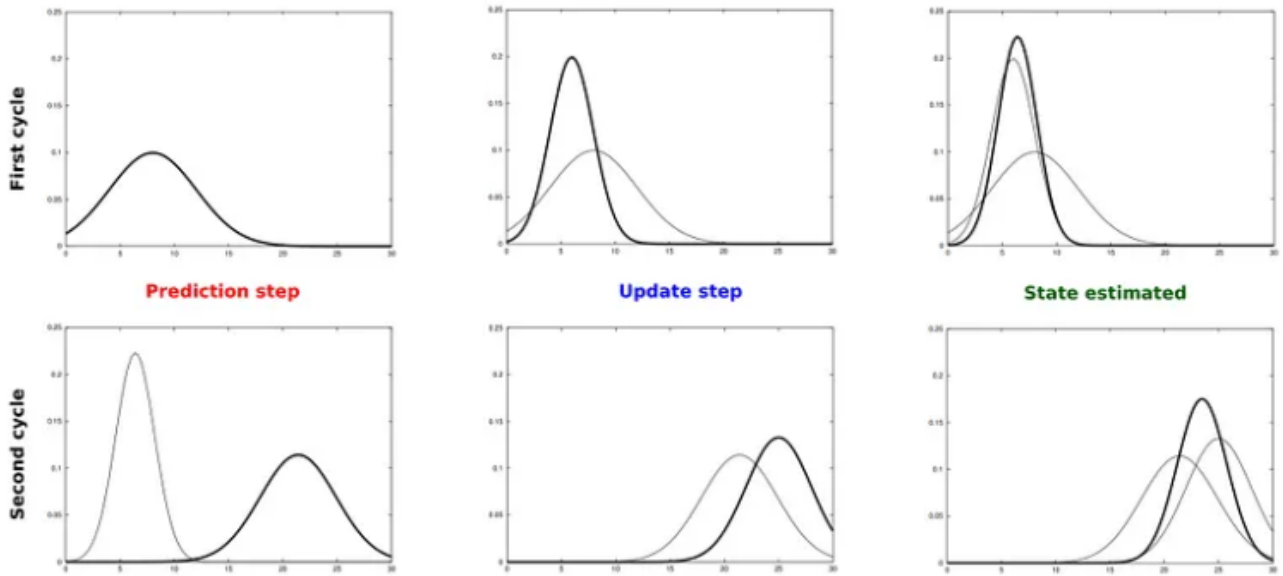
Kalman Filter Example

- Linear model assumption and Gaussian distribution assumption
  - motion model : constant velocity model
    - ★ $\mathbf{x}_t = \mathbf{x}_{t-1} + \dot{\mathbf{x}}_{t-1}\Delta t$, $\dot{\mathbf{x}}_t = \dot{\mathbf{x}}_{t-1}$, $\mathbf{y}_t = \mathbf{y}_{t-1} + \dot{\mathbf{y}}_{t-1}\Delta t$, $\dot{\mathbf{y}}_t = \dot{\mathbf{y}}_{t-1}$

$$\therefore A_t = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

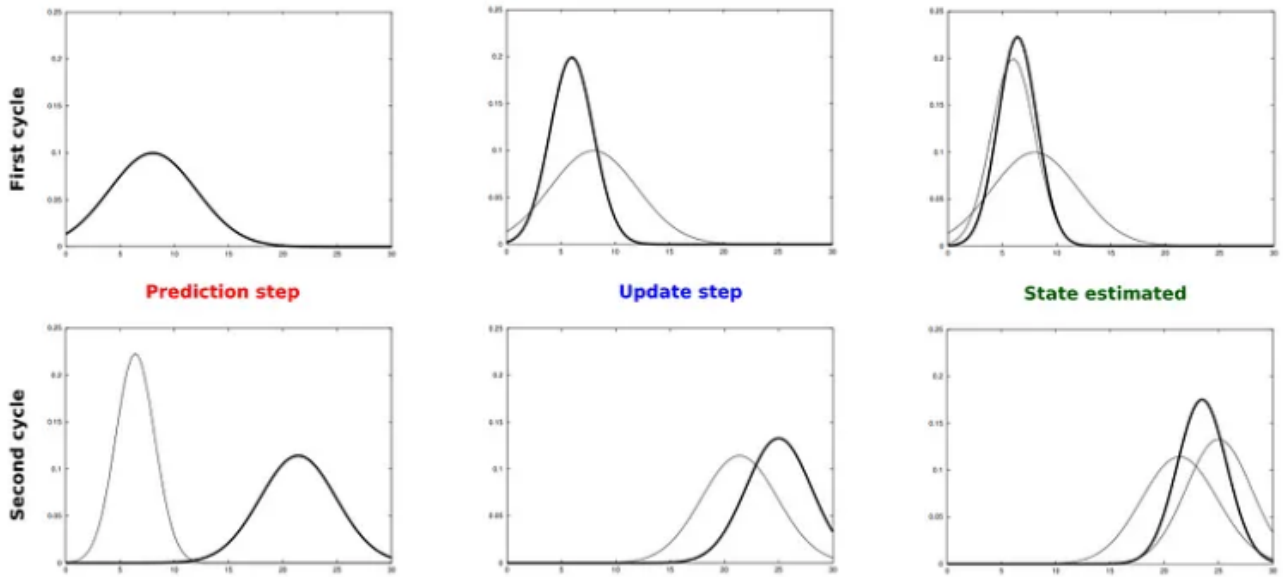
    - ★ B_t is not necessary since $u_t = 0$
    - ★ R_t is a (4×4) covariance matrix, representing the motion noise
  - observation model : the sensors only measure the location of the robot
    - ★ $C_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
    - ★ Q_t is a (2×2) covariance matrix, representing the measurement noise

Kalman Filter Visualization



- The bold curves in the first / second column are the Gaussian distributions produced by the motion / observation model.
- The bold curves in the third column are the Gaussian distributions of updated state.

Kalman Filter Visualization



- When two Gaussian distributions are employed to estimate the state, the resulting Gaussian distribution will have its mean closer to the mean of the more certain one. And the resulting distribution will be more certain than both input distributions, which is the contribution of the Kalman gain.

Summary

- The Bayes filter and the Kalman filter can dynamically balance the information from the prediction and the measurement, iteratively improving the state estimation with each update.

- The Bayes filter and the Kalman filter offer a robust foundation for accurate and dynamic state estimation whenever we want to track the location of an object, estimate the state of a system or navigate through uncertain environments.

Table of Contents

Motivation

- Recall that the Bayes filter is a general powerful framework, which is the following recursive equation

$$bel(x_t) = \eta p(z_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$$

- The Kalman filter incorporates the linear model assumption and the Gaussian distribution assumption so that we can practically solve the equation and come up with a viable implementation.

Motivation

- However, in the real world, these two assumptions may be too naive or ideal. That is, the models could probably be non-linear and the probability distribution could probably be non-Gaussian. Then it could be highly difficult or even impossible to compute the integral. So how can we apply the Bayes filter in this tough scenario?

  $\rightarrow$ Use Mote-Carlo simulation to approximate the integral!

  $\Rightarrow \int f(x_k)p(x_k)dx_k \approx \frac{1}{N}\sum_{i=1}^{N} f(x_k^{(i)})$, where $x_k^{(i)}$ is sampled from the probability distribution $p(x_k)$

Methodology

- With each sample $x_{t-1}^{(i)}$, we generate a new sample $x_t^{(i)}$ from the probability distribution $p(x_t|x_{t-1}^{(i)}, u_t)$. In this way, we can obtain a new sample set

$$\mathcal{X}_t = \{x_t^{(i)}, i = 1, ..., N\}$$

We collect all unique values of $\mathcal{X}_t$ to form the set

$$\mathcal{V}_t = \{x_{t,j}, j = 1, ..., J\}$$

where $x_t^{(i)} \in \mathcal{V}_t$ and each element of $\mathcal{V}_t$ is unique.

Methodology

- $bel(x_{t,j}) = \eta \sum_{i=1}^{N} p(z_t|x_t^{(i)}) bel(x_{t-1}^{(i)}) \delta(x_t^{(i)} = x_{t,j})$, $j = 1, ..., J$

  Since $\sum_{j=1}^{J} bel(x_{t,j})$ should be 1, $\eta = 1/\sum_{i=1}^{N} p(z_t|x_t^{(i)}) bel(x_{t-1}^{(i)})$

- In practice, the particle filter calls each sample $x_t^{(i)}$ a particle and the belief of each particle $bel(x_t^{(i)})$ the importance weight $w_t^{(i)}$.

Sequential Importance Sampling

Input: $\{x_{t-1}^{(i)}, w_{t-1}^{(i)})\}_{i=1}^{N}$, z_t, u_t

$w_{sum} = 0$

for $i = 1, ..., N$ **do**

$$\textit{propagate particle} \ : \ \textit{draw sample } x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_t)$$

$$\textit{update weight} \ : \ w_t^{(i)} = w_{t-1}^{(i)} * p(z_t | x_t^{(i)})$$

$$\textit{cumulate weight} \ : \ w_{sum} = w_{sum} + w_t^{(i)}$$

end

Sequential Importance Sampling

for $i = 1, ..., N$ **do**

$$\textit{normalize weight} \ : \ w_t^{(i)} = w_t^{(i)} / w_{sum}$$

end

Output: $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N}$

Sequential Importance Sampling

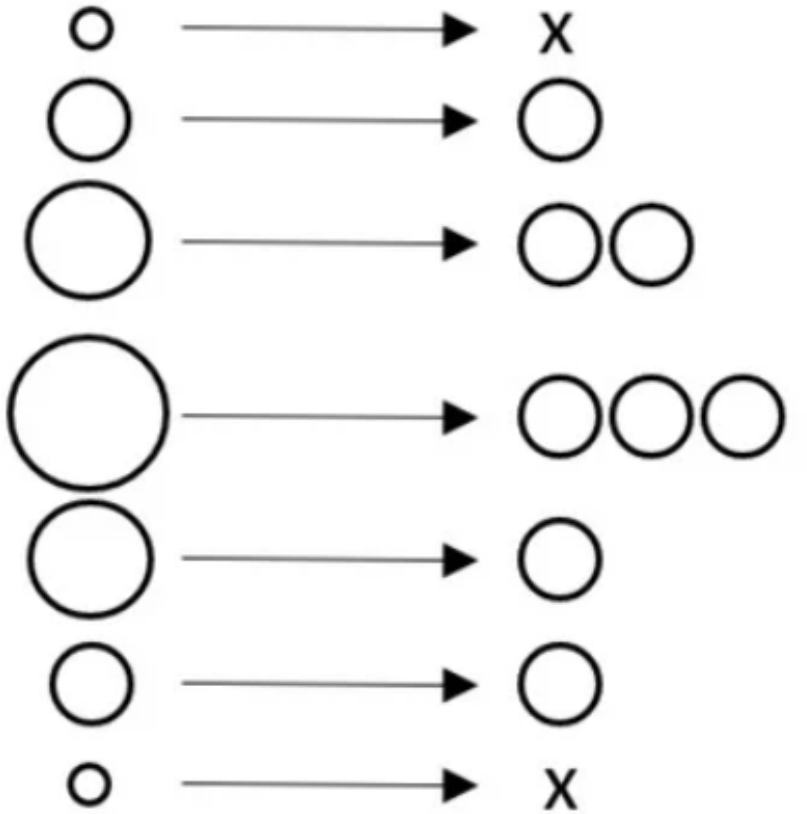
- Empirically, there is a degeneracy problem - after a few iterations the weight of one particle will be very close to one and that of all other particles will be almost zero.
- The consequences of the degeneracy problem are
  - almost all computational effort will be put into computations related to particles that have negligible or no contribution to the overall estimate
  - the number of effective particles is only one.

  In this way, the expressiveness of the filter will be highly limited since a single particle can only represent one point in the state space rather than pdfs of arbitrary shapes. Thus, the filter will fail to approximate the true state and give rise to unacceptably large estimation errors.
- A solution to the degeneracy problem $\rightarrow$ resampling

Resampling

- Resampling is performed directly after the update step. In the resampling step, new particles are randomly selected, with replacement, from the set of weighted particles.
- The probability of selecting a particle is proportional to its weight and the number of particles remains unchanged.
  $\Rightarrow$ Particles with higher weights are likely to be chosen more than once, whereas those with low weights are likely to be ignored.
- After resampling the weights are reset to $1/N$

Resampling



Resampling

- Although resampling effectively avoids the degeneracy problem, it is computationally costly and resampling at every time instant is a conservative way to avoid particle degeneracy. That is, particle degeneracy could still be prevented even we resample less frequently. Furthermore resampling at every time instant usually reduces the diversity in the particle set.
- We can use the following measure to indicate particle degeneracy.

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

If $N_{eff} < N_{thr}$, where N_{thr} is a pre-defined threshold, a relatively small subset of particles gather a relatively large proportion of the total weight. Thus, we perform resampling.

Sequential Importance Resampling / Particle Filter

Input: $\{x_{t-1}^{(i)}, w_{t-1}^{(i)})\}_{i=1}^{N}$, z_t, u_t

$w_{sum} = 0$

for $i = 1, ..., N$ **do**

$$\textit{propagate particle} \ : \ \textit{draw sample } x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_t)$$

$$\textit{update weight} \ : \ w_t^{(i)} = w_{t-1}^{(i)} * p(z_t | x_t^{(i)})$$

$$\textit{cumulate weight} \ : \ w_{sum} = w_{sum} + w_t^{(i)}$$

end

Sequential Importance Resampling / Particle Filter

for $i = 1, ..., N$ **do**

$$normalize\ weight\ :\ w_t^{(i)} = w_t^{(i)} / w_{sum}$$

end

If $N_{eff} < N_{thr}$ **then**

Resample N particles with replacement

Reset weights : $w_t^{(i)} = 1/N$

end

Output: $\{x_t^{(i)}, w_t^{(i)})\}_{i=1}^{N}$

Particle Filter

- Finally, there are two questions remained to be solved.
  1. How do we draw sample $x_t^{(i)}$ from the probability distribution $p(x_t|x_{t-1}^{(i)}, u_t)$?
  2. How do we compute $p(z_t|x_t^{(i)})$?
- For the first question, we resort to the motion model.
- For the second question, we resort to the observation model.

Motion Model

- A motion model describes how the robot's pose (position and orientation) is likely to change from one time instant to the next, given the control inputs (e.g. wheel speeds or motor commands).
- The motion model can be visual odometry (based on data estimated from images or point clouds), velocity model (based on linear and angular velocities), dynamic model (based on wheel slippage, inertia, or external forces), etc.
- In the prediction step, we apply the the robot's motion data to propagate the particles so that they move as the robot has moved.
- Recall that in the Kalman filter, it assumes that this model is a linear model with additive Gaussian noise.

Observation Model

- An observation model describes how the robot's sensors (e.g. cameras, lidar or other perception devices) would observe the environment at a given pose.
- The observation model can measure the distances between the robot and certain landmarks in the environment, the angles between the robot's orientation and known landmarks, etc.
- In the updating step, we update the weight of each particle using the observation model.
- Recall that in the Kalman filter, it assumes that this model is a linear model with additive Gaussian noise.

Additional Notes of The Particle Filter

- Because of the adoption of Monte-Carlo simulation, the particle filter is also known as Sequential Monte Carlo (SMC). Furthermore, since it is widely used in the localization problem of moving robots, it is also known as Monte Carlo Localization (MCL)

- When initializing, we often generate particles uniformly. However, we can generate particles more precisely if we know a-priori that which region is more likely to be the initial states.

Additional Notes of The Particle Filter

- The resampling step plays a crucial role in reducing the number of particle groups. Particles within the "correct" region have higher weights than those in other areas, and the resampling step replaces lightly weighted particles with their more substantial counterparts. Thus, the set of particles tends to converge to a small cloud of particles centered at the most likely state of the system .

- For complex nonlinear systems, implementing the particle filter can be easier than the Kalman filter since deriving linearised models for the Kalman filter may be cumbersome.

Table of Contents

Reference


Kalman Filter Part 1 —Introduction
https://medium.com/@mathiasmantelli/kalman-filter-series-introduction-6d2e2b28d4cf


Kalman Filter Part 2 —Bayes Filter
https://medium.com/@mathiasmantelli/kalman-filter-part-2-bayes-filter-f2fa9c0b5c95


Kalman Filter Part 3 —A Formal Discussion
https://medium.com/@mathiasmantelli/kalman-filter-part-3-a-formal-discussion-e1a61b359fef

Reference


Particle Filter Part 1 —Introduction https://medium.com/@mathiasmantelli/particle-filter-part-1-introduction-fb6954bc12ec


Particle Filter Part 2 —Intuitive example and equations https://medium.com/@mathiasmantelli/particle-filter-part-2-intuitive-example-and-equations-0716223b862b


Particle Filter Part 3 —Motion and Measurement models https://medium.com/@mathiasmantelli/particle-filter-part-3-motion-and-measurement-models-be79857a5490


Elfring, Jos, Elena Torta, and René van de Molengraft. "Particle filters: A hands-on tutorial." Sensors 21.2 (2021): 438.