Robot Autonomy Lecture Notes

Huan-Yang Chang (huanyanc@andrew.cmu.edu)
Kai Li (kail2@andrew.cmu.edu)
Zihao Zhang (zihaoz1@andrew.cmu.edu)

March 27th, 2017

Lecture Outline:

1. Failure cases of the Kalman Filter
2. Kidnapped Robot Problem
3. Formal Probabilistic Filter
4. Particle Filter

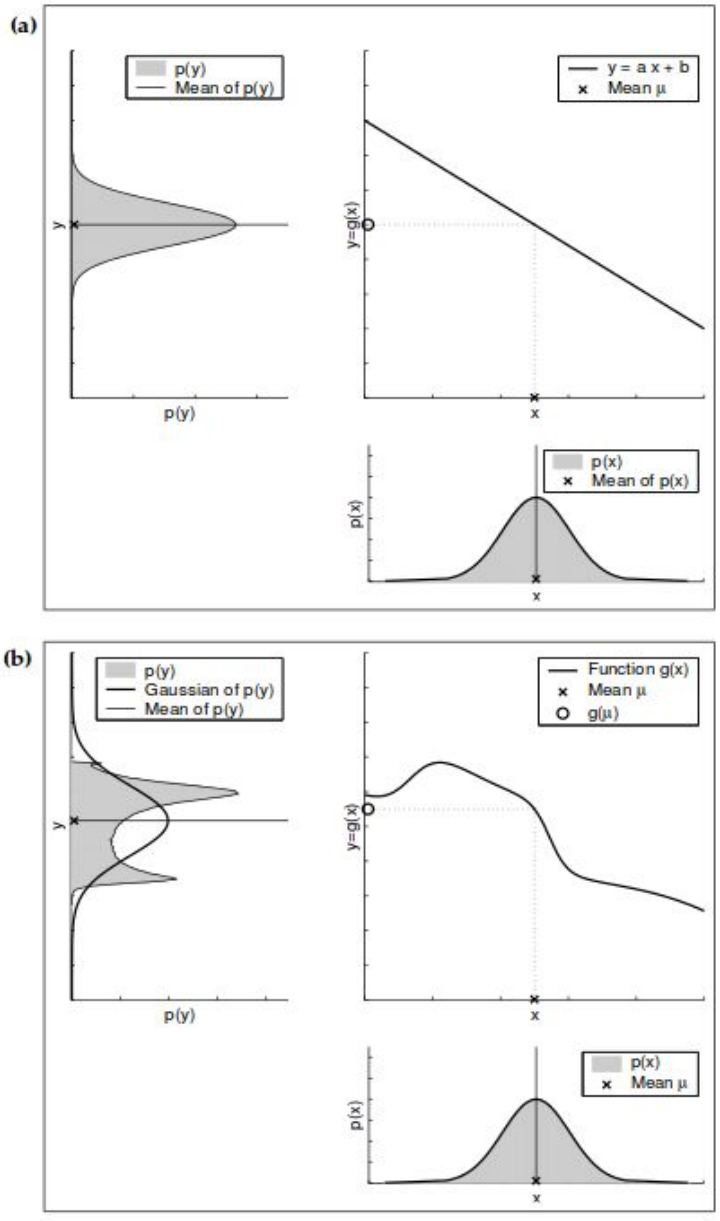
1. Failure cases of Kalman filter

As a quick recap from the last lecture, the Kalman Filter algorithm is mainly composed of two steps, namely the Prediction and the Measurement Update. During the prediction step, the Kalman filter (KF) will transform the initial guess, expressed as a Gaussian distribution, into the prediction (next state's estimation), which will also be expressed as a Gaussian distribution. In the next step, the prediction will be finally transformed into the updated measurements, which will also be in the 8form of a Gaussian distribution. The Kalman Filter can be applied to robot localization problem by running the algorithm iteratively over time.

At the same time, it is very important to keep in mind that successful implementation of the Kalman Filter depends on two basic assumptions:

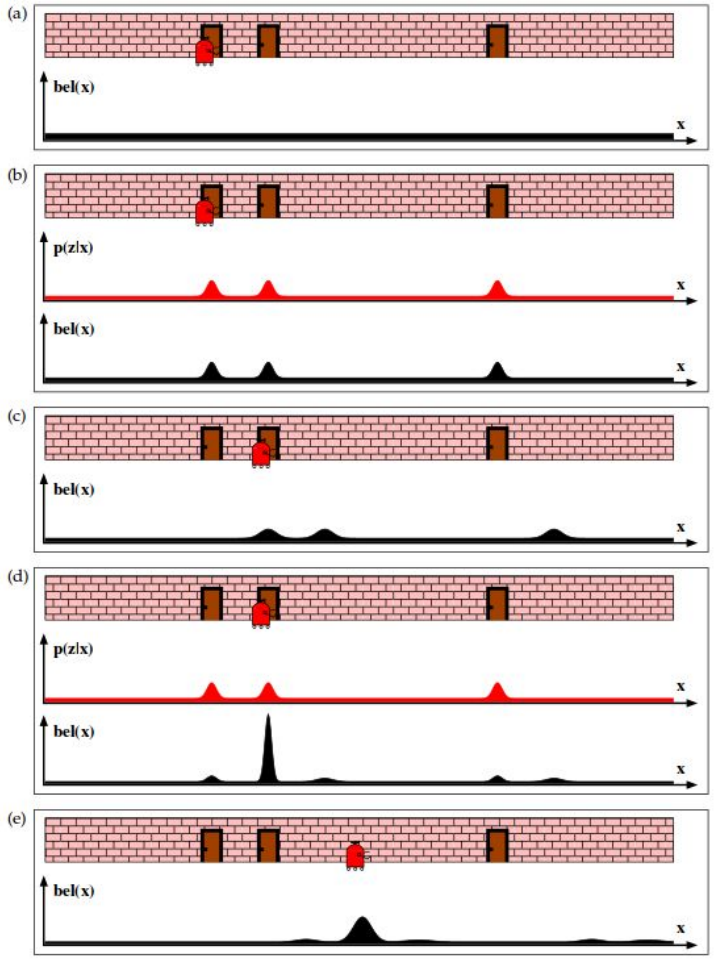
1. *Linear model*: This means that the state transition probability as well as the measurement probability must be linear functions.
2. *Gaussian noise*: This means that all noises involved in the linear system can be expressed using Gaussian Distribution.

Therefore, the transformation from one step another will not result in a Gaussian distribution if the function is nonlinear, and the Kaiman filter will fail in this case. Similarly, if any relevant noise in the system cannot be modeled using Gaussian distribution, the Kalman filter algorithm will fail as well.



2. Kidnapped robot problem

We first studied the robot localization problem by introducing this simple yet classic example as the following: given the map of environment, the sensor model, the data from odometry, we want to know the probability distribution of the robot pose in the environment over time.



As illustrated in the figure above, the robot is initially known to be in front of a door in the environment. Therefore, the probability of the robot being in front of each of the three doors are equally high, while the probability of the robot being at other places are equally low (figure (b)). As the robot moves to the right, the whole distribution moves with the robot solely according to the odometry data (figure (c)). Note that the uncertainty of the robot pose is also increased constantly during this process, as implied by the larger covariance of the Gaussian distributions. When the robot arrives at another door at the next time step, the distribution will be updated

according to the new measurement data received by the robot through observation. By combining the new measurement with prior belief, the update is able to suggest that the robot being in front of the second door has the highest probability (figure (d)).

However, environments in the real world can be rarely as simple as in the example above. In most cases, the robot is required to be able to locate itself in an environment which contains many highly similar scenarios, such as walking through a long, straight hallway with untextured walls on the sides. In such cases, can the robot still locate itself successfully in the environment using the previous algorithm? How can we prevent the undesired situation that the robot is carried to an arbitrary location with the highly similar scenes?

Such kind of problems can be generally referred as the kidnapped robot problem. “It is commonly used to test a robot's ability to recover from catastrophic localization failures”. In the next two section, we will introduce a new filtering technique that is also commonly used for robot localization to better address this problem .

3. Formal Probabilistic Filter

Given the map of the environment, the sensor model, and the odometry data, we can calculate the probability of the robot’s current state based on all the information we have. Mathematically, this can be expressed as:

$$P(x_t \mid u_{1:t}, y_{1:t})$$

This is called a *Posterior* or *Belief*, with the u representing the control input measured by the odometry and the y representing the sensor information.

In Bayes Filters, we can use the prior belief (a.k.a. the estimation on the previous state), the control input, and the sensor information altogether to update the current state. Note that we can actually just use the the last state and the control input to form the prediction on the current state and use the current sensor information for the update, thanks to the nice properties derived from the Markov assumption.

Essentially, the Markov property states that "certain variables are independent of others if one knows the values of a third group of variables, the conditioning variables". This can help us simplify the process model as the following:

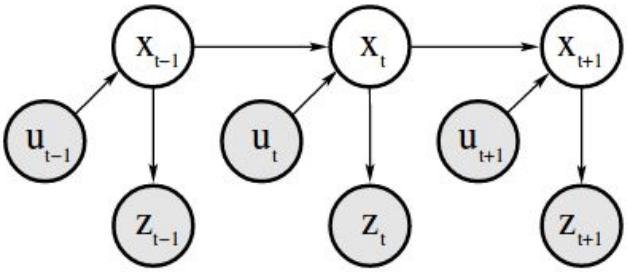
$$P(x_t \mid x_{0:t-1}, y_{1:t-1}, u_{1:t}) = P(x_t \mid x_{t-1}, u_t)$$

In other words, it assumes that "x_{t-1} is a sufficient statistic of all previous controls and measurements up to this point in time. Knowledge of any other variable, such as past measurements, controls, or even past states, is irrelevant if x_t is complete."

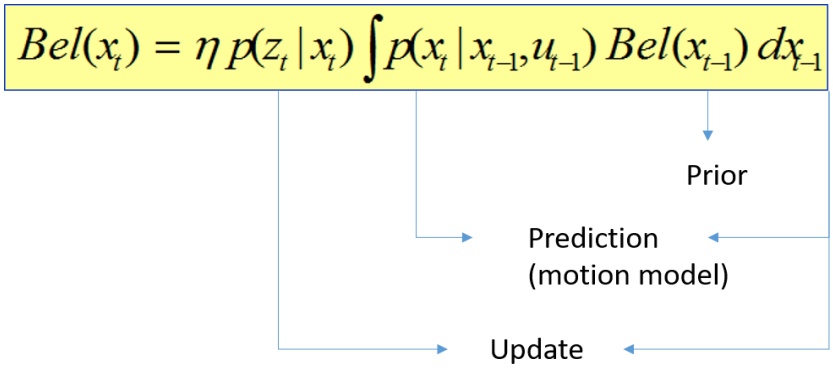
Similarly, this property can be also applied to simplify the measurement model as the following:

$$P(y_t \mid x_{0:t}, y_{1:t-1}, u_{1:t}) = P(y_t \mid x_t)$$

In the figure below, the Bayes network is shown graphically to help better illustrate the idea of the Markov assumption above.



In conclusion, the Bayes Filter algorithm can be summarized as the equation below:



, where *Bel (x)* = $P(x_t \mid u_{1:t}, y_{1:t})$ is the Belief on the state x_t.

4. Particle Filter

"A popular alternative to Gaussian techniques are *nonparametric* filters. Nonparametric filters do not rely on a fixed functional form of the posterior, such as Gaussians. Instead, they approximate posteriors by a finite number of values, each roughly corresponding to a region in state space."

-- Chapter 4, Probabilistic Robotics

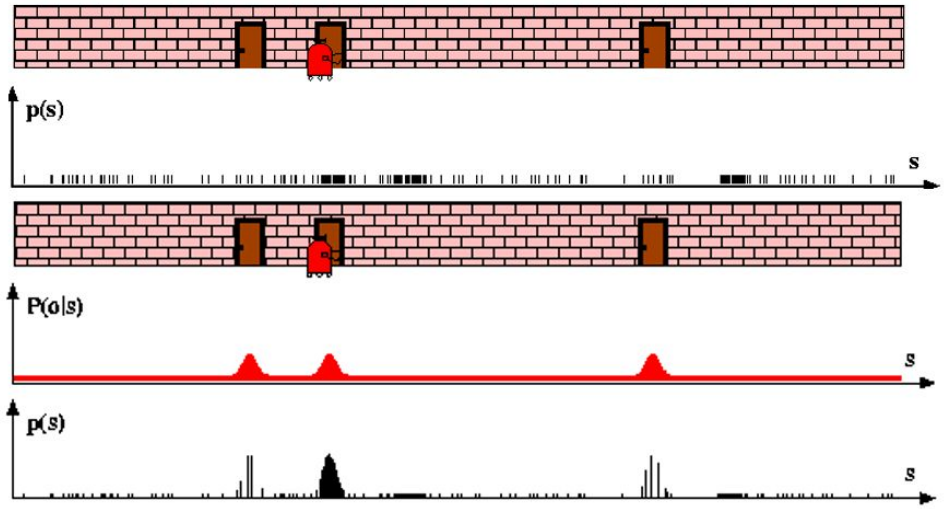
As a nonparametric filter, the Particle Filter (PF) uses multiple samples instead of parametric representation of distributions to deal with any arbitrary distribution. This makes the Particle Filter particularly suitable for estimation of non-Gaussian, nonlinear models.

By taking advantage of the Monte Carlo method, we can takes samples from the prior in any form and create the sample proposal. Theoretically, any distribution can be represented by a set of N samples, where each sample (also called a particle) is in the form of:

$$X = < X^i, w^i >^{i=1:N}$$

The X^i represents the i-th particle and w^i stands for the importance weight associated with this particle. Essentially, each sample/particle corresponds to a hypothesis of the estimated state.

We use the measurement model to count the importance weight and use this importance weight to re-sample the distribution. Intuitively, more particles will be drawn next time around particles that have higher importance weights. With more updates over time, the particle will eventually converge to the most possible location that the robot would be at.



The previous simple example of a robot walking along the wall (shown in the figure above) can be used again here as a great illustration of how the Particle Filter algorithm can be applied to the robot localization problem.

The complete Particle Filter algorithm is summarized as the following:

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \varnothing, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ ***Generate new samples***
4. Sample index *j(i)* from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}
6. $w_t^i = p(z_t \mid x_t^i)$ ***Compute importance weight***
7. $\eta = \eta + w_t^i$ ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert***
9. **For** $i = 1 \ldots n$
10. $w_t^i = w_t^i / \eta$ ***Normalize weights***

Summary:

1. The Kalman Filter algorithm is built upon two basic assumptions, namely the linear model and the Gaussian noise.
2. The successful implementation of the Kalman Filter algorithm also relies on the assumption that a good initial guess is provided.

3. The Bayes Filter recursively forms new prediction based on prior belief and update the prediction to form the current belief.
4. Under the Markov assumption, recursive Bayesian updating can be used to efficiently combine evidence.

5. The Particle Filter is an alternative nonparametric implementation of the Bayes Filter and can be used to track arbitrary distributions.
6. The Particle Filter uses particles (samples) to represent underlying distribution and involves three major steps:

a. Generate sample proposal
b. Assign importance weights
c. Resample particles

Reference:

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Howie M. Choset et al. (2005). *Principles of robot motion: theory, algorithms, and implementation*. The MIT Press

Michael Kaess. (2017). *16-833 Robot Localization and Mapping, Lecture Notes*. The Robotics Institute, Carnegie Mellon University