RMP- Bayesian Filter & Particle Filter

Key words	KF/EKF/UKF VS. Bayes Filter, global localization, non-Gaussian, non-linear, particle filter, resampling, unknown initial position
Status	note complete

KF/EKF/UKF: incremental localization

- robot position: can be modeled by a **single Gaussian distribution**
- **known initial position** (Initialization necessary)

Problem:

- **unknown initial position** —> robot can be anywhere
- distribution of the robot position: **unsmooth, non-Gaussian**

Filtering: **sequentially estimating** the states of a system as **a set of observations available** onine.

- Solution: $p(X_k|Y_k) = p(x_k|Y_k)$ —> no need to keep track of all states.

▼ a general system: **non-linear, non-Gaussian**

- **assumptions:**
    - **states are markovian**: $p(x_k|x_{k-1}, x_{k-2}, \ldots, x_0) = p(x_k|x_{k-1})$
    - observations: $P(y_k|x_k, \ldots, x_0) = P(y_k|x_k)$

Bayesian Inference:

- Goal: construct posterior state x_k given **a sequence observations** Y_k
- Solution: Monte Carlo method —> approximate by **a cloud of weighted discrete particles**

Bayes Filter

- **assumptions:**
    - **states are markovian**: $p(x_k|x_{k-1}, x_{k-2}, \ldots, x_0) = p(x_k|x_{k-1})$
    - observations: $P(y_k|x_k, \ldots, x_0) = P(y_k|x_k)$
    - **arbitrary distribution**: a mixture of multiple Gaussians
- Input:
    - **sequence** of observations: $y_{1:k} = [y_1, y_2, \ldots, y_k]$
    - **sequence** of constrol inputs: $u_{0:k-1} = [u_0, u_1, \ldots, u_{k-1}]$
    - **motion model**: $P(x_k|x_{k-1}, u_{k-1})$ with initial $P(x_1|u_0)$
    - **measurement model** (**map**): $P(y_k|x_k)$
- Prediction:

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

- Correction (measurement update):
    - posterior probability distribution: $P(x_k|y_{1:k}, u_{0:k-1})$
        - **combination of prediction and observation of KF** —> recursive

$$P(x_k|u_{0:k-1}, y_{1:k}) = \eta_k \cdot \underbrace{P(y_k|x_k)}_{\text{observation}} \int_{x_{k-1}} \underbrace{P(x_k|x_{k-1}, u_{k-1})}_{\text{prediction}} \cdot \underbrace{P(x_{k-1}|u_{0:k-2}, y_{1:k-1})}_{\text{recursive instance}}$$

discrete case: Grid-based Localization

- Updating using **motion mdel**: a **discrete convolution** of **prior** by the **driving noise** of planned motion

$$P(x_{k+1}^-) = \Sigma P(x_{k+1}^-|x_k, u_k) \cdot P(x_k)$$

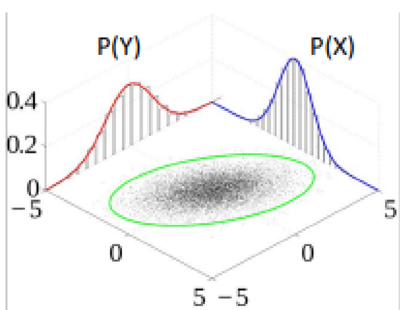
- Updating from **observation**: a **multiplication** of **prior probability** by **likelihood of observation**

  $P(x_{k+1}) = \eta \cdot P(y_{k+1}|x^-_{k+1}) \cdot P(x^-_{k+1})$

  —> a probability distribution —> **maximum likelihood**

Bayes Filter VS. Kalman Filter

- KF:
  - **local navigation:** update the **unique position.** (even with Data association: it will be matched to one position)
  - if multiple solutions: unable to associate observation to prediction. —> **Data association problem —> ambiguities**
- Bayes Filter:
  - **global navigation**
  - combination of prediction and observation in recursion.
  - need to integrate the entire space to calculate posterior.



wikipedia

Particle Filter / Monto Carlo Filter

- **non-Gaussian, nonlinear** processes

- Input:
    - sequence of measurements $y_{1:k}$
    - sequence of control vectors $u_{0:k-1}$
    - **N samples** corresponding to prior belief $P(x)$ — uniform distribution $P(x_0)$ / others from previous call $P(x_{k-1})$
- Output:
- ▼ Process:
    1. Initial state: **uniform distribution of samples**
    2. **draw n samples/particles** from the belief
    3. prediction: **predict state for each particle** with motion model, each sample has **weight = 1**
    4. measurement update: for each mesaurement, **each particle updates measurement prediction** with measurement model
    5. **update weight** with actual measurement.
    6. **resample particles** according to the updated weight
- Resampling:
    - Roulette wheel: roll the wheel n times —> bias towards higher weight, $O(logn)$
    - **Stochastic universal sampling**: generate n pointers evenly on the wheel at 1 time, $O(n)$

Bayes Filter VS. Particle Filter

- Bayes: represent the probability distribution as **n-dimensional function**

  —> space discretization, computational expensive

- Particle: represent **belief** with **samples/particles** —> particle density $\uparrow$, probability $\uparrow$