Principles of Robot Autonomy I

Particle Filter, Importance Weighting, Monte Carlo Localization

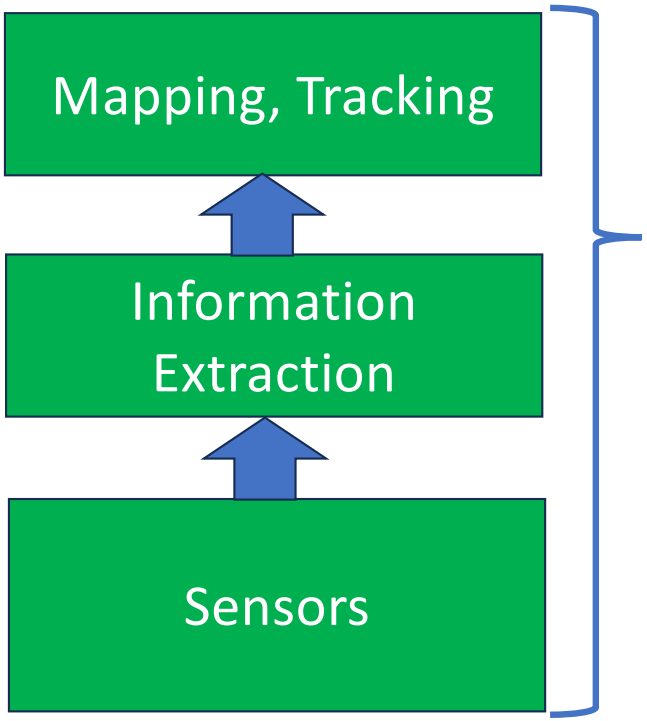
Target Tracking EKF and PF

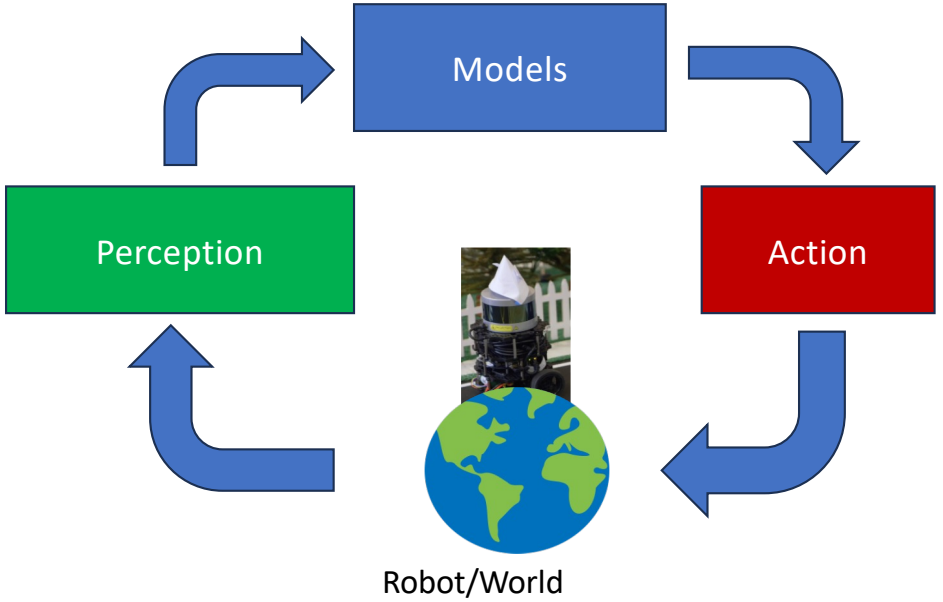
Logistics

- Homework 5: Due 12/4 (extended deadline, last one!)
- Final exam, window Sat 12/6, 6:30pm – Tues 12/9, 6:30pm:
  - Take home, 72 hour window
  - Check out exam on gradescope
  - You will have personal 5 hour time slot
  - Open notes, book, HW solutions
  - No internet, no GenAI, no working with others
- Lecture 17:
  - Particle Filter
  - Importance Sampling
  - Monte Carlo Localization
  - Target tracking EKF and PF

Robot Perception

Perception Stack





Perception Stack: Particle Filtering

Use sensor data to update the models

Localization, Mapping, Tracking:

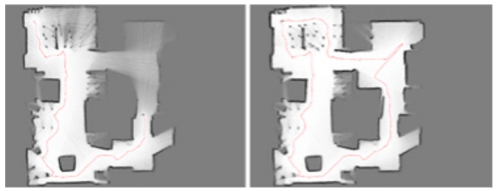
- EKF/Monte Carlo localization
- Occupancy grid mapping
- Factor graphs/SLAM
- Tracking (EKF and Particle Filter)
- **AA273: Filtering (Schwager)**
- **AA275: Navigation (Gao)**

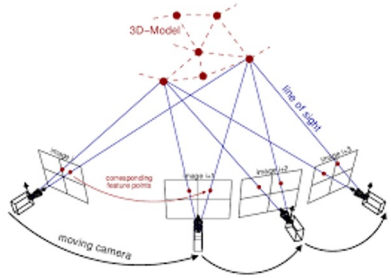
Information extraction

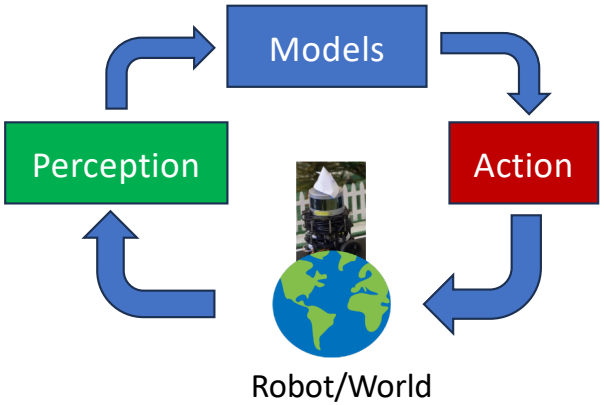
- Computer vision: features, correspondences, Structure from Motion (SfM), depth
- Lidar scan matching, ICP
- **CS231A: Comp Vision**

Sensors:

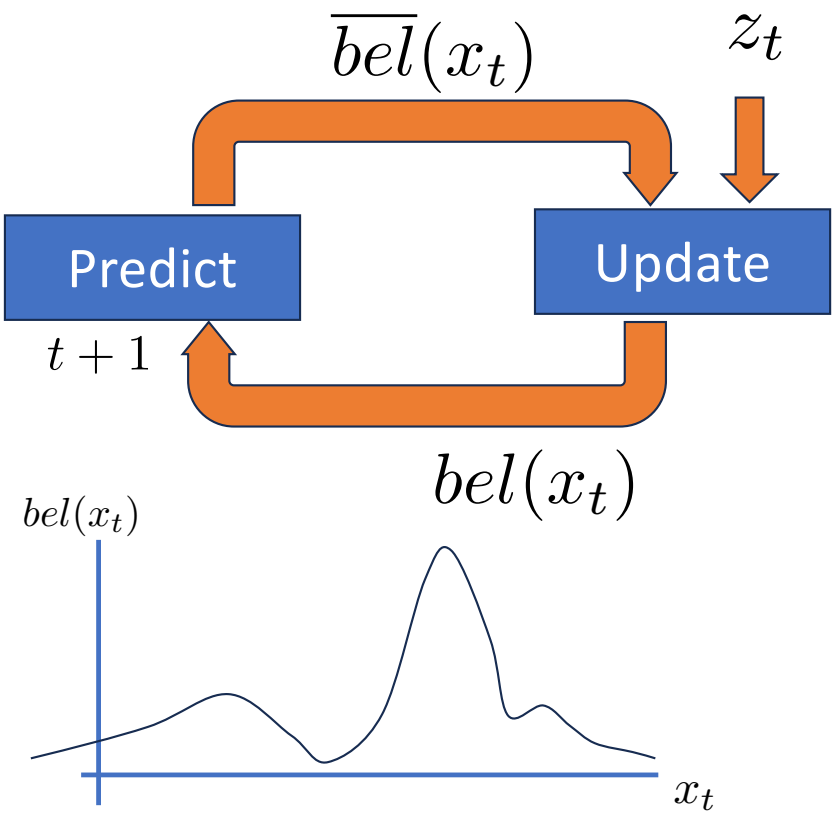
- RGB Cameras, RGB-D/stereo cameras, Lidar
- IMU, GPS, wheel encoders







Recap: Bayes filter algorithm

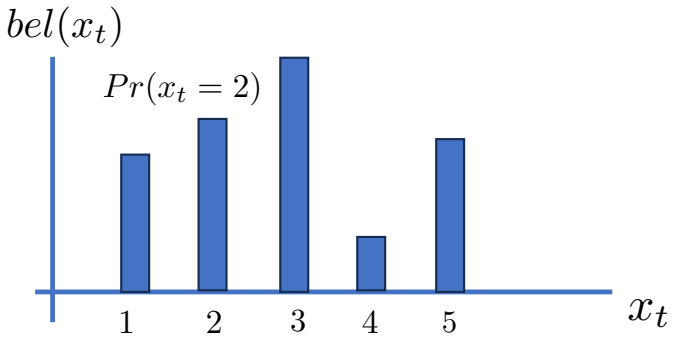


- Algorithm initialized with $bel(x_0)$ (e.g., uniform or points mass)
- Usually intractable, mostly a theoretical tool

Data: $bel(x_{t-1}), u_t, z_t$
Result: $bel(x_t)$
foreach x_t **do**
  $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$;
  $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$;
end
Return $bel(x_t)$

Recap: Discrete Bayes' filter

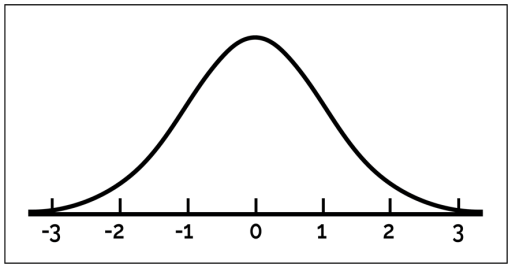
- Applies to problems with discrete finite state spaces (e.g. Occ grid mapping)
- Belief $bel(x_t)$ represented as pmf $\{p_{k,t}\}$

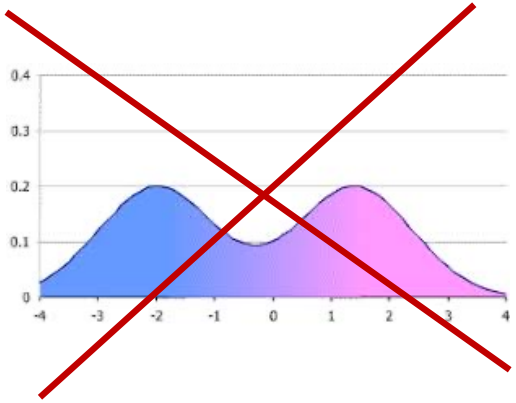


Data: $\{p_{k,t-1}\}, u_t, z_t$
Result: $\{p_{k,t}\}$
foreach k **do**
  $\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\, p_{i,t-1}$;
  $p_{k,t} = \eta\, p(z_t \mid X_t = x_k)\, \bar{p}_{k,t}$;
end
Return $\{p_{k,t}\}$

Recap: Kalman filter/Extended Kalman Filter

- Applies to state space systems with continuous state spaces
- Can only express Gaussian distributions



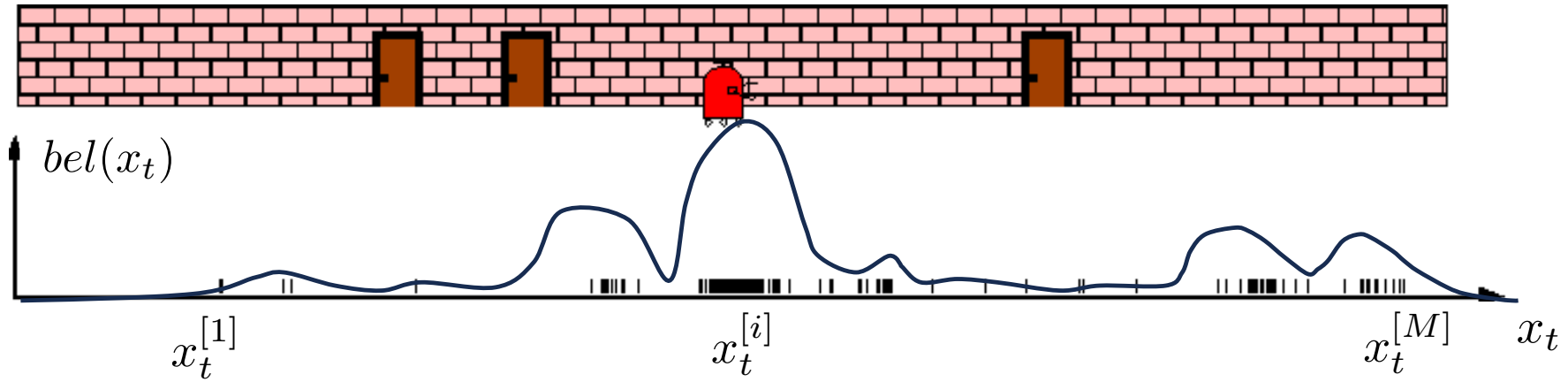


Extended Kalman Filter:

$$\textbf{Data: } (\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$$
$$\textbf{Result: } (\mu_t, \Sigma_t)$$
$$\overline{\mu}_t = g(u_t, \mu_{t-1}) \ ;$$
$$\overline{\Sigma}_t = G_t \Sigma_{t-1} {G_t}^T + R_t;$$
$$K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1};$$
$$\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t));$$
$$\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t;$$
$$\text{Return } (\mu_t, \Sigma_t)$$

Particle filter

- Key idea: represent posterior $bel(x_t)$ by a set of random samples
- Can represent arbitrary distributions
- Requires much more computation than KF/EKF



- Allows one to represent non-Gaussian distributions and handle nonlinear transformations in a direct way

Particle filter

- Samples of posterior distribution are called *particles*, denoted as

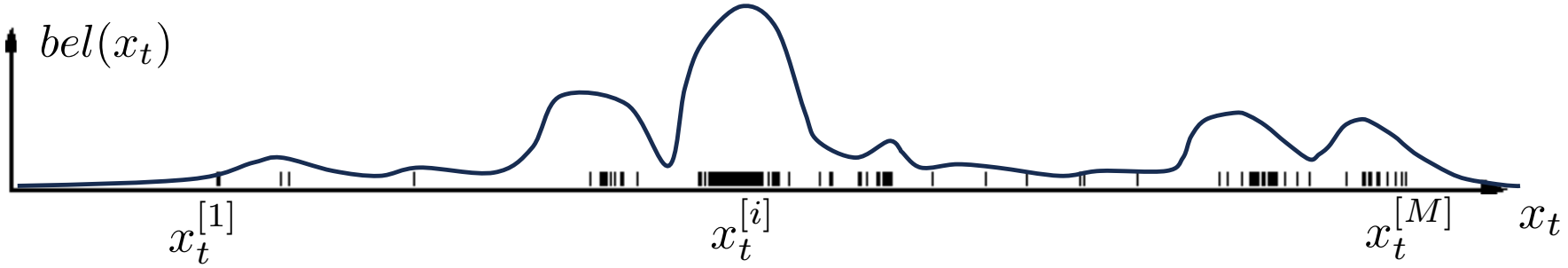
$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$$

- A particle represents a hypothesis about what the true world state might be at time *t*
- Ideally, particles should be distributed according to

$$x_t^{[m]} \sim p(x_t \mid z_{1:t},\ u_{1:t}) = bel(x_t)$$

- Matching exact only as $M \to \infty$, but $M \approx 10^n - 100^n$ usually good enough for n dimensional state space
- A particle filter constructs the particle set $\mathcal{X}_t$ from the particle set $\mathcal{X}_{t-1}$ recursively, with the goal of matching the distribution $bel(x_t)$

Representing distributions as particle sets



The particles let us **approximate** any desired probability or moment:

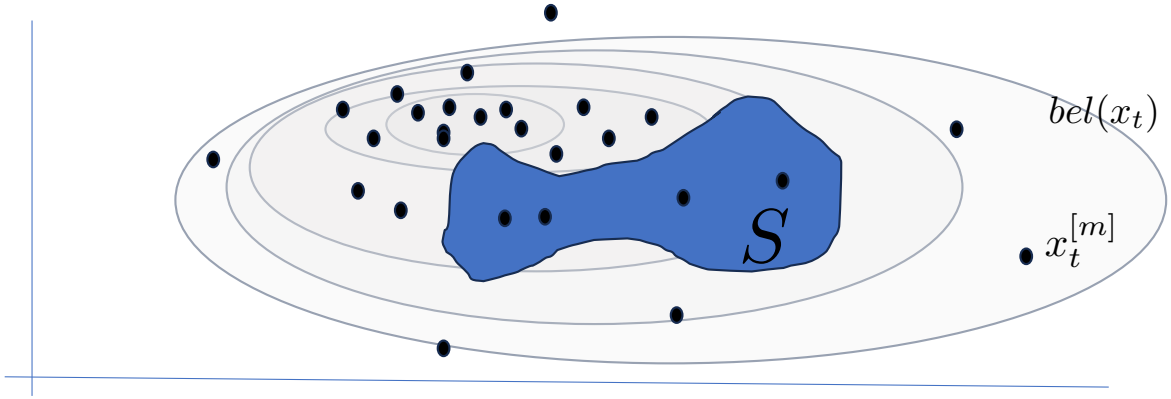
Mean: $\mu_t \approx \frac{1}{M} \sum_{m=1}^{M} x_t^{[m]} = \tilde{\mu}_t$

Probability:

$$Pr(x_t \in S) \approx \frac{\#\text{ particles in S}}{M} = \frac{|\{m \mid x_t^{[m]} \in S\}|}{M}$$

Covariance: $\Sigma_t \approx \frac{1}{M} \sum_{m=1}^{M} (x_t^{[m]} - \tilde{\mu}_t)(x_t^{[m]} - \tilde{\mu}_t)^T = \tilde{\Sigma}_t$

Representing distributions as particle sets



The particles let us **approximate** any desired probability or moment:

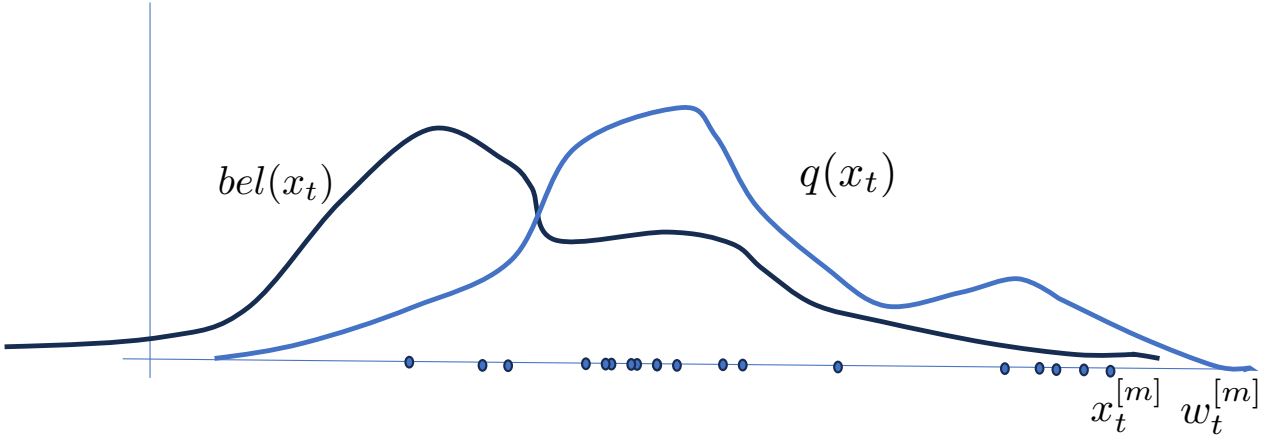
Mean: $\mu_t \approx \frac{1}{M} \sum_{m=1}^{M} x_t^{[m]} = \tilde{\mu}_t$

Covariance: $\Sigma_t \approx \frac{1}{M} \sum_{m=1}^{M} (x_t^{[m]} - \tilde{\mu}_t)(x_t^{[m]} - \tilde{\mu}_t)^T = \tilde{\Sigma}_t$

Probability:

$$Pr(x_t \in S) = \frac{\#\text{ particles in S}}{M} = \frac{|\{m \mid x_t^{[m]} \in S\}|}{M}$$

Representing distributions **weighted** as particle sets



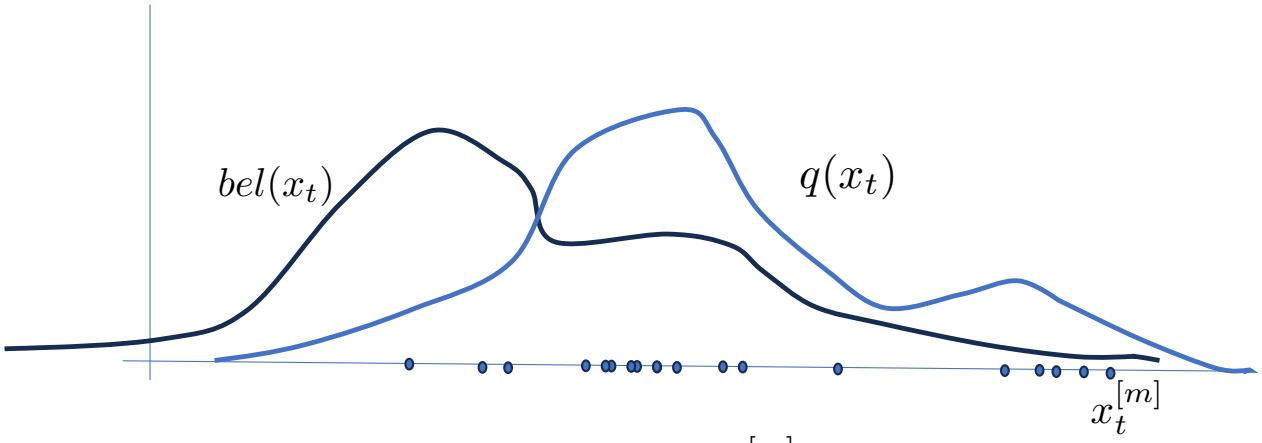
The particles let us **approximate** any desired probability or moment:

Mean: $$\mu_t \approx \sum_{m=1}^{M} w_t^{[m]} x_t^{[m]} = \tilde{\mu}_t$$

Probability: $$Pr(x_t \in S) \approx \sum_{\{m | x_t^{[m]} \in S\}} w_t^{[m]}$$

Covariance: $$\Sigma_t \approx \sum_{m=1}^{M} w_t^{[m]} (x_t^{[m]} - \tilde{\mu}_t)(x_t^{[m]} - \tilde{\mu}_t)^T = \tilde{\Sigma}_t$$

Importance Weighting



We want to represent $bel(x_t)$ with particles $x_t^{[m]} \sim q(x_t)$ from a different distribution

Weight the particles!

$$E_{bel}[f(x_t)] = \int_{x_t} f(x_t) bel(x_t)\, dx_t = \int_{x_t} f(x_t) \frac{bel(x_t)}{q(x_t)} q(x_t)\, dx_t$$

We have:

$$w(x) = \frac{bel(x)}{q(x)}$$

$$E_{bel}[f(x_t)] = \int_{x_t} f(x_t) w(x_t) q(x_t)\, dx_t \approx \sum_{x_t^{[m]}} f(x_t^{[m]}) w_t^{[m]}$$

Importance weights:

$$w_t^{[m]} = \frac{bel(x_t^{[m]})}{M q(x_t^{[m]})}$$

Stochastic System Model

Measurement distribution:

$$p(z_t \mid x_t)$$

We **evaluate probabilities** from this model to get weights.

$$w_t^{[m]} = p(z_t \mid x_t^{[m]})$$

Nonlinear state space form:

$$z_t = h(x_t) + \delta_t$$

$$\delta_t \sim \mathcal{D}(\delta; \theta)$$ Non-Gaussian or Gaussian measurement noise

Plug $z_t, x_t^{[m]}$ into formula for $\mathcal{D}(z_t - h(x_t^{[m]}); \theta)$ to obtain $w_t^{[m]}$

Stochastic System Model

E.g., Gaussian measurement noise

$$\delta_t \sim \mathcal{N}(\delta; 0, Q_t) = \frac{1}{\sqrt{(2\pi)^n |Q_t|}} e^{-\frac{1}{2}\delta^T Q_t^{-1} \delta}$$

Plug $z_t, x_t^{[m]}$ into

$$w_t^{[m]} = \mathcal{N}(z_t - h(x_t^{[m]}); 0, Q_t) = \frac{1}{\sqrt{(2\pi)^n |Q_t|}} e^{-\frac{1}{2}(z_t - h(x_t^{[m]}))^T Q_t^{-1} (z_t - h(x_t^{[m]}))}$$

To get $w_t^{[m]}$

Stochastic System Model

State transition distribution:

$$p(x_t \mid x_{t-1}, u_t)$$

We **draw samples** from this model to get new particles at time *t*.

$$x_t^{[m]} \sim p(x_t \mid x_{t-1}^{[m]}, u_t)$$

Nonlinear state space model:

$$x_t = g(x_{t-1}, u_t) + \epsilon_t$$

$$\epsilon_t \sim \mathcal{D}(\epsilon; \theta)$$ Non-Gaussian or Gaussian process noise

(1) Draw noise sample $\epsilon_t^{[m]}$

(2) Plug into nonlinear state space model to get $x_t^{[m]} = g(x_{t-1}^{[m]}, u_t) + \epsilon_t^{[m]}$

This draws $x_t^{[m]}$ from the state transition distribution $x_t^{[m]} \sim \mathcal{D}(x_t - g(x_{t-1}^{[m]}, u_t); \theta)$

Stochastic System Model

E.g., Gaussian process noise

(1) Draw: $\epsilon_t^{[m]} \sim \mathcal{N}(0, R_t)$

(2) Evaluate: $x_t = g(x_{t-1}, u_t) + \epsilon_t$

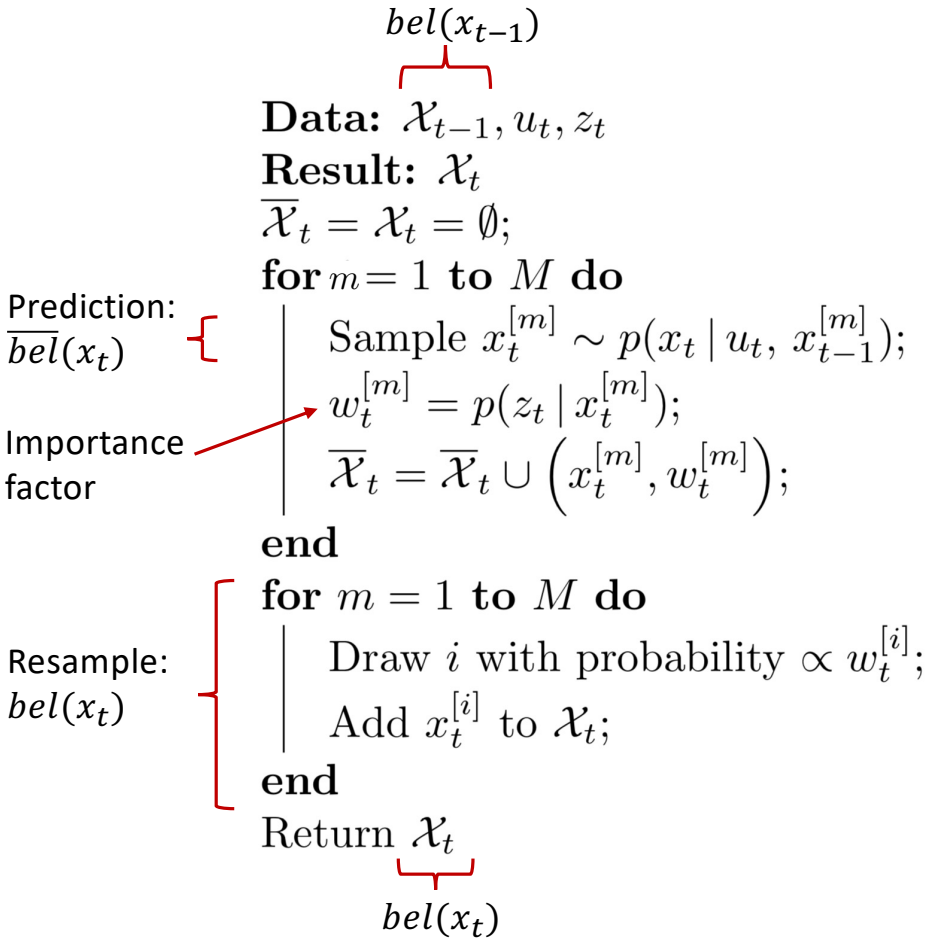
Equivalent to drawing from non-gaussian state transition distribution:

$$x_t^{[m]} \sim \mathcal{N}(x_t - g(x_{t-1}^{[m]}, u_t); 0, R_t)$$

Particle filter: algorithm

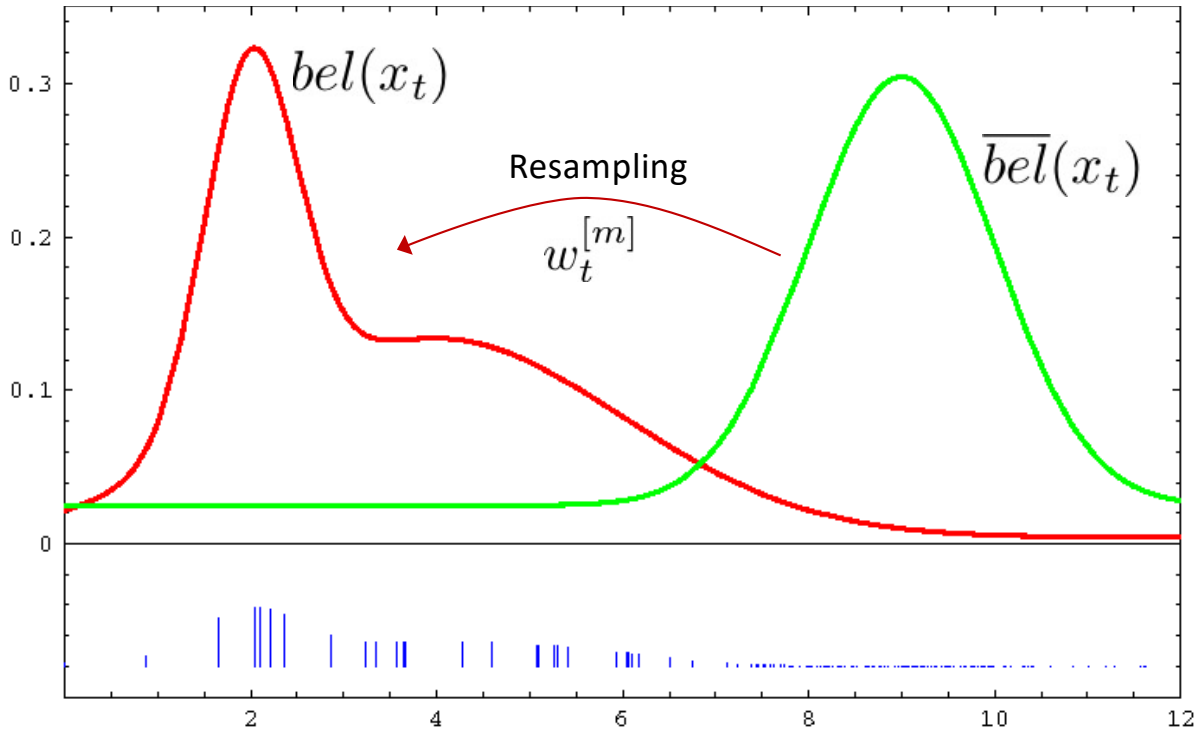
- The temporary particle set $\overline{\mathcal{X}}_t$ represents the belief $\overline{bel}(x_t)$
- The particle set $\mathcal{X}_t$ represents the belief $bel(x_t)$
- Importance factors are used to incorporate measurement z_t in the particle set
- After resampling, particles are (as $M \to \infty$) distributed as

$$bel(x_t) = \eta\, p(z_t \mid x_t^{[m]})\, \overline{bel}(x_t)$$

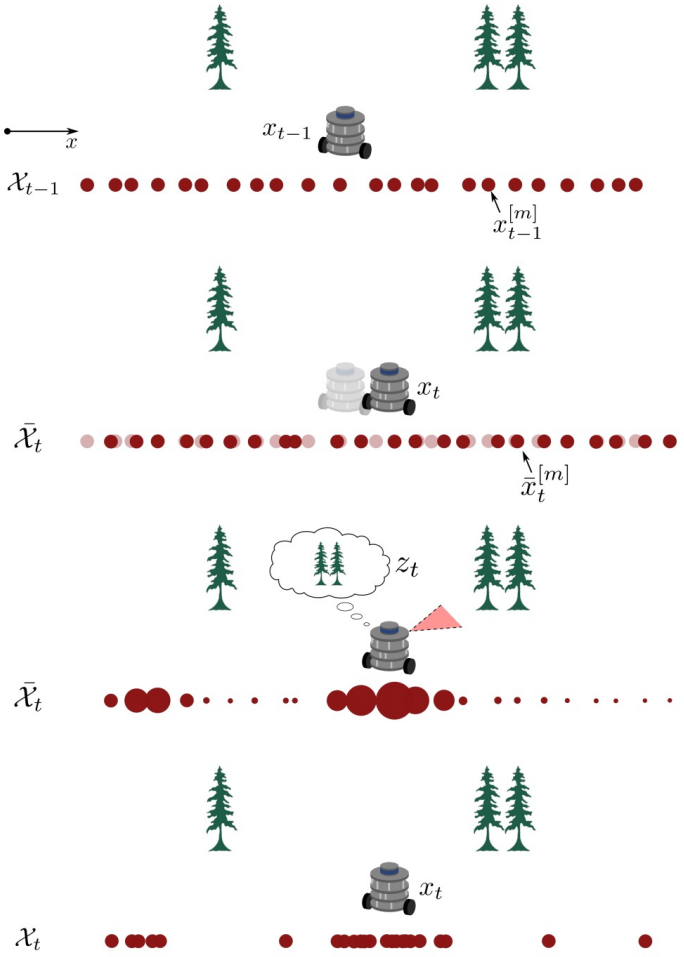


Resampling

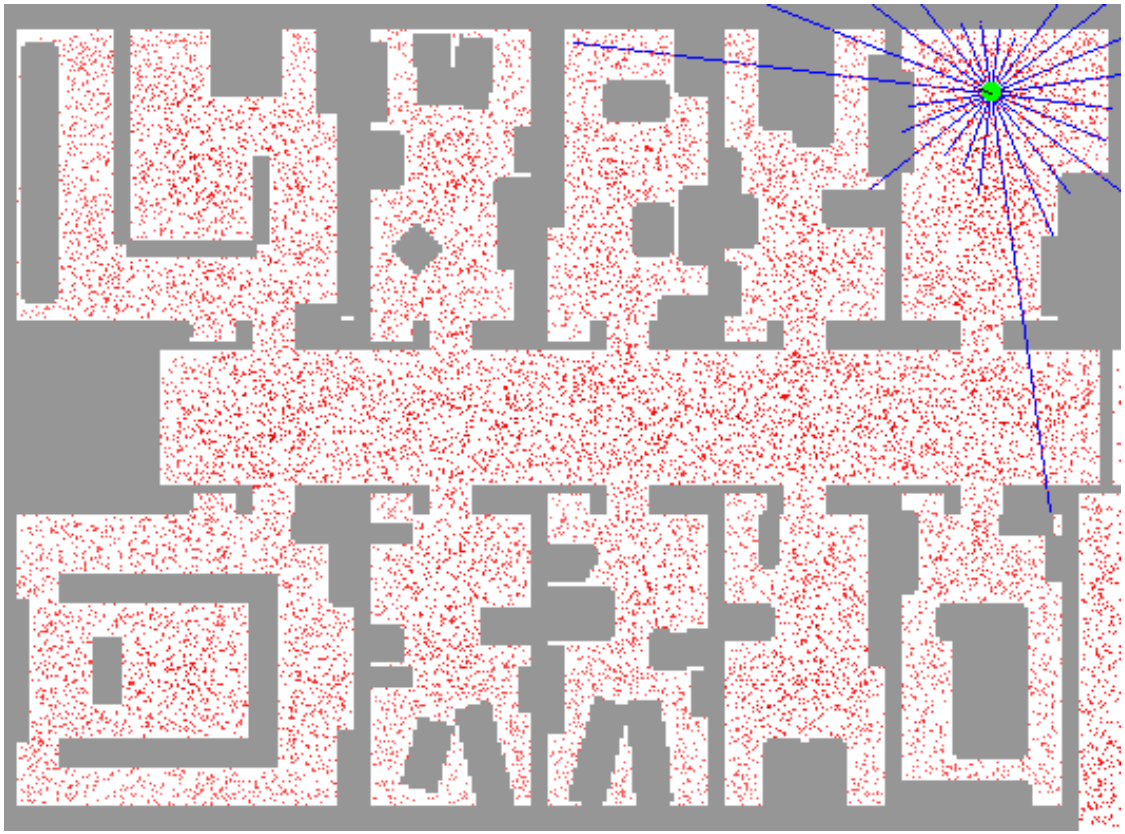
- Variants exist to sample less frequently (not every time step) to avoid particle diversity problems.



Particle filter: example

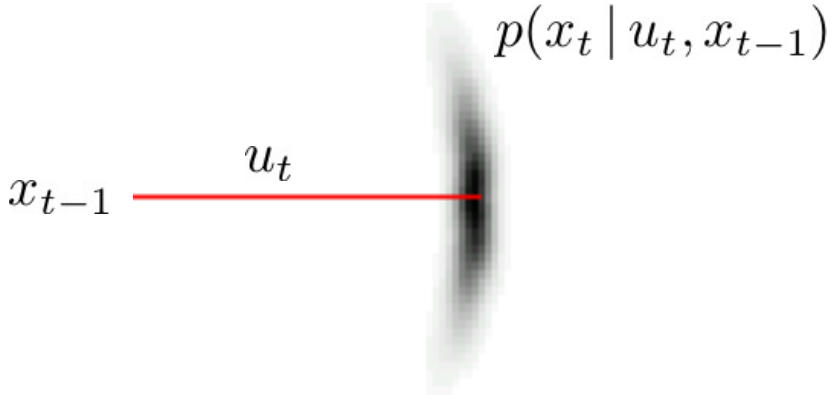


MCL: example



E.g.: Monte Carlo localization

- Motion model is probabilistic



PF requires us to sample:

$$x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$$

Example for dynamics with additive noise:

$$x_t^{[m]} = f(x_{t-1}^{[m]}, u_t) + \epsilon_{t-1}^{[m]}, \quad \epsilon_{t-1}^{[m]} \sim \mathcal{N}(0, R_{t-1})$$

Also requires to evaluate measurement likelihood:

$$w_t^{[m]} = p(z_t \mid x_t^{[m]})$$

E.g., for Gaussian:

$$w_t^{[m]} = \mathcal{N}(C_t x_t^{[m]}, Q_t)$$

Monte Carlo localization

- Key idea: represent belief $bel(x_t)$ by a set of *M* particles

$$\mathcal{X}_t = \{x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}\}$$

- Requires a map *m* as input
- Can handle dynamic environments via outlier rejection

Compute for occupancy map
Or feature based map

Data: $\mathcal{X}_{t-1}, u_t, z_t, m$
Result: $\mathcal{X}_t$
$\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$;
for $i = 1$ **to** M **do**
  Sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]}, m)$;
  $w_t^{[m]} = p(z_t \mid x_t^{[m]}, m)$;
  $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t \cup \left(x_t^{[m]}, w_t^{[m]}\right)$;
end
for $i = 1$ **to** M **do**
  Draw i with probability $\propto w_t^{[i]}$;
  Add $x_t^{[i]}$ to $\mathcal{X}_t$;
end
Return $\mathcal{X}_t$

Sample reject collisions

Target Tracking with EKF and PF

Recap: Kalman filter/Extended Kalman Filter

Kalman Filter:

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
Result: (μ_t, Σ_t)

$$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \ ;$$
$$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t;$$
$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1};$$
$$\mu_t = \overline{\mu}_t + K_t (z_t - C_t \overline{\mu}_t);$$
$$\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t;$$

Return (μ_t, Σ_t)

Extended Kalman Filter:

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
Result: (μ_t, Σ_t)

$$\overline{\mu}_t = g(u_t, \mu_{t-1}) \ ;$$
$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t;$$
$$K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1};$$
$$\mu_t = \overline{\mu}_t + K_t (z_t - h(\overline{\mu}_t));$$
$$\Sigma_t = (I - K_t H_t) \overline{\Sigma}_t;$$

Return (μ_t, Σ_t)

Nonlinear dynamics

dynamics Jacobian

Nonlinear measurement equation

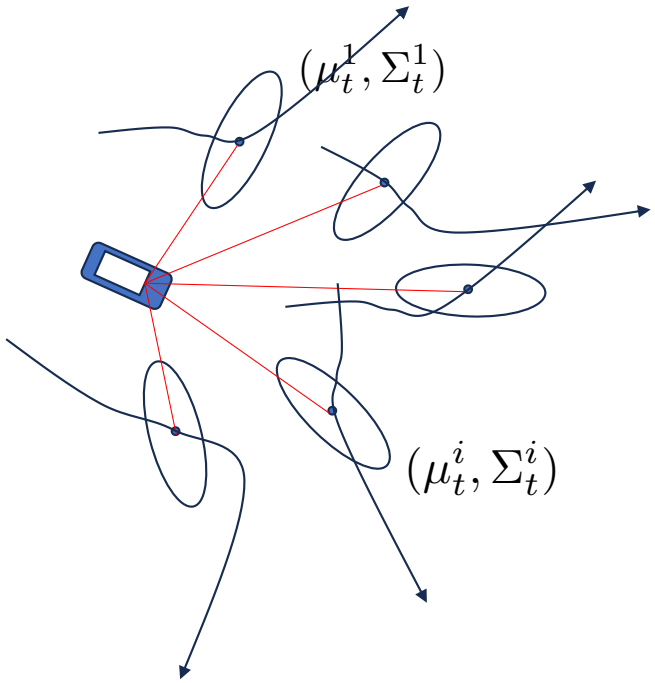
Measurement Jacobian

EKF (Multi-)Target Tracking

- Targets represented by distribution over their unknown poses

$$p(m_i \mid z^i_{1:t}) \sim \mathcal{N}(\mu^i_t, \Sigma^i_t)$$

- Assume targets move with a known dynamics model



EKF (Multi-) Target Tracking

Known robot pose: $q_t = [x_t \;\; y_t \;\; \theta_t]^T \quad p_t = [x_t \;\; y_t]^T$

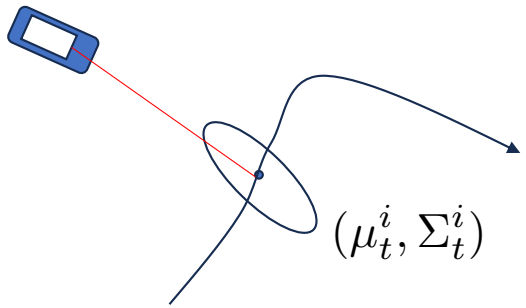
Feature location prior: $m_0^i \sim \mathcal{N}(\mu_0^i, \Sigma_0^i)$

Common sensor models ($z_t = h(x_t, u_t) + \delta_t$):

Linear (use KF): $z_t^i = R(\theta_t)(m_t^i - p_t) + \delta_t^i, \quad \delta_t^i \sim \mathcal{N}(0, Q_t^i)$

Range (EKF): $z_t^i = \|m_t^i - p_t^i\| + \delta_t^i$

Bearing (EKF): $z_t^i = \text{atan}\left(\frac{m_t^i(2) - y_t}{m_t^i(1) - x_y}\right) - \theta_t + \delta_t^i$



Target dynamics model:

$$m_t^i = g_i(m_{t-1}^i, u_t^i) + \epsilon_t^i \qquad \epsilon_t^i \sim \mathcal{N}(0, R_t^i)$$

Unknown control input!

Next time

- Last lecture!
- Emerging topics: NeRFs/Gaussian Splatting in robotics, end-to-end policies with RL/IL