Theory and Implementation of Particle Filters

ELG 5218 - Uncertainty Evaluation in Engineering Measurements and Machine Learning

Miodrag Bolic

University of Ottawa

February 4, 2026

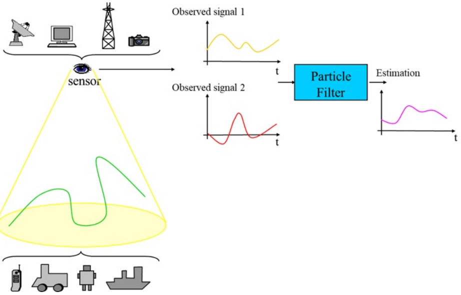
Outline

Notebook: bearings_only_particle_filter.ipynb

Big Picture

Goal: Estimate a latent stochastic process given some noisy observations.

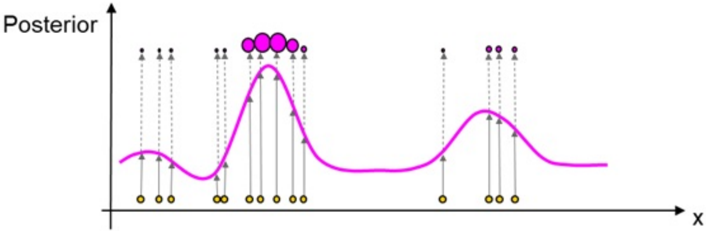
Concepts:
Bayesian filtering
Monte Carlo sampling



Sequential Monte Carlo Techniques

- Particle filter is a technique for implementing recursive Bayesian filter by Monte Carlo sampling
- The idea: represent the posterior density by a set of random particles with associated weights.
- Compute estimates based on these samples and weights

- Bootstrap filtering
- The condensation algorithm
- Particle filtering
- Interacting particle approximations
- Survival of the fittest



Historical Development of Particle Filters (SMC)

Foundations in Monte Carlo (1953): Sampling-based computation becomes practical

- Metropolis et al., "Equation of State Calculations by Fast Computing Machines" (Metropolis MCMC)

Sequential growth idea (1955): Early *sequential* Monte Carlo / "growth principle"

- Rosenbluth & Rosenbluth, "Monte Carlo Calculation of the Average Extension of Molecular Chains"

Modern particle filtering (1993): Recursive Bayesian estimation for nonlinear / non-Gaussian models

- Gordon, Salmond & Smith, "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation" (Bootstrap filter)

Statistical vision adoption (1996–1998): Generalization + practical breakthroughs

- Kitagawa (1996), "Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models"
- Isard & Blake (1998), "CONDENSATION—Conditional Density Propagation for Visual Tracking"
- Liu & Chen (1998), "Sequential Monte Carlo Methods for Dynamic Systems" (SMC framework)

Tutorials

- Arulampalam et al. (2002), "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking"

Applications Across Domains

Applied everywhere where
time series data is needed

Signal Processing & Communications

- Image processing and segmentation
- Model selection
- Tracking and navigation
- Channel estimation
- Blind equalization
- Positioning in wireless networks

Scientific Disciplines

- Biology & Biochemistry
- Chemistry
- Economics & Business
- Geosciences
- Immunology
- Materials Science
- Pharmacology & Toxicology
- Psychiatry/Psychology
- Social Sciences

Example: Bearings-Only Tracking

Problem: Estimate the position and velocity of a target using only bearing measurements.

Variable Definitions:

- $s_{x,k}, s_{y,k}$ — target position in 2D at time k
- $V_{x,k}, V_{y,k}$ — target velocity components
- Δt — sampling interval
- y_k — measured bearing (angle) to the target

Noise Terms:

- $\mathbf{u}_k \sim \mathcal{N}(0, Q)$ — process noise (model uncertainty)
- $v_k \sim \mathcal{N}(0, R)$ — measurement noise (bearing error)

Latent State Vector:
$\mathbf{z}_k = [s_{x,k}, V_{x,k}, s_{y,k}, V_{y,k}]^\top$

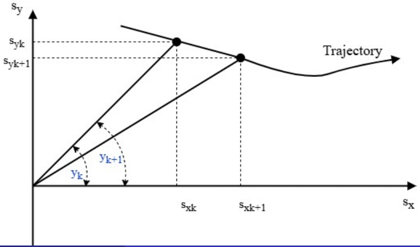
Typically used in passive sonar and EO tracking.

State Equation (Motion Model):

$$\mathbf{z}_{k+1} = \begin{bmatrix} s_{x,k} + V_{x,k}\Delta t \\ V_{x,k} \\ s_{y,k} + V_{y,k}\Delta t \\ V_{y,k} \end{bmatrix} + \mathbf{u}_k$$

Observation Equation (Bearing):

$$y_k = \arctan\left(\frac{s_{y,k}}{s_{x,k}}\right) + v_k$$

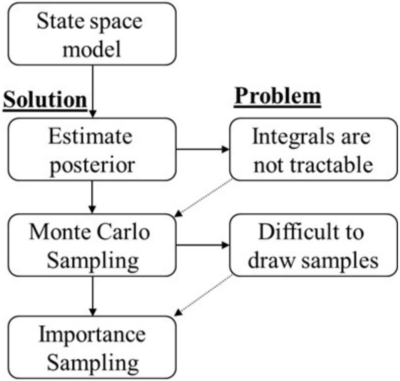


Example: Car Positioning

- Observations: velocity and turn information
- Car equipped with electronic roadmap
- Initial position accuracy: ≈ 1 km
- Particles initially spread evenly on roads
- As car moves, particles concentrate at actual location

Fundamental concepts

- State space representation
- Bayesian filtering
- Monte-Carlo sampling
- Importance sampling



State Space Representation

The latent state sequence $\{\mathbf{z}_k\}$ is a Markov random process.

Latent State Equation:

$$\mathbf{z}_k = f_z(\mathbf{z}_{k-1}, \mathbf{u}_k)$$

- $\mathbf{z}_k$: latent state vector at time k
- f_z: state transition function
- $\mathbf{u}_k$: process noise

Observation Equation:

$$\mathbf{y}_k = f_y(\mathbf{z}_k, \mathbf{v}_k)$$

- $\mathbf{y}_k$: observations
- f_y: observation function
- $\mathbf{v}_k$: observation noise

Density Representation

Dynamic systems can be represented using conditional densities:

Latent State Equation (density form):

$$p(\mathbf{z}_k \mid \mathbf{z}_{k-1})$$

Observation Equation (density form):

$$p(\mathbf{y}_k \mid \mathbf{z}_k)$$

These depend on:

- Functions $f_z(\cdot)$ and $f_y(\cdot)$
- Distributions of $\mathbf{u}_k$ and $\mathbf{v}_k$

Bayesian Filtering Objective

Estimate the unknown latent state $\mathbf{z}_k$ based on observations $\{\mathbf{y}_t\}_{k=1}^{k}$.

Goal: Find the posterior distribution

$$p(\mathbf{z}_{0:k} \mid \mathbf{y}_{1:k})$$

where:

- $\mathbf{z}_{0:k} = (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$
- $\mathbf{y}_{1:k} = (\mathbf{y}_1, \ldots, \mathbf{y}_k)$

Benefit: All kinds of estimates can be computed from the posterior.

$$E(g(\mathbf{z}_{0:k})) = \int g(\mathbf{z}_{0:k})\, p(\mathbf{z}_{0:k} \mid \mathbf{y}_{1:k})\, d\mathbf{z}_{0:k}.$$

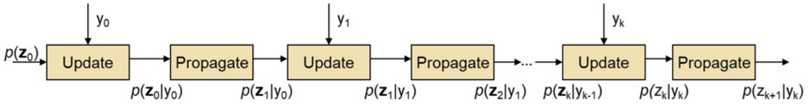
At time $k = 0$:

Update (Bayes' Theorem):

$$p(\mathbf{z}_0 \mid \mathbf{y}_0) = \frac{p(\mathbf{y}_0 \mid \mathbf{z}_0)p(\mathbf{z}_0)}{p(\mathbf{y}_0)}$$

Propagate:

$$p(\mathbf{z}_1 \mid \mathbf{y}_0) = \int p(\mathbf{z}_1 \mid \mathbf{z}_0)p(\mathbf{z}_0 \mid \mathbf{y}_0)\, d\mathbf{z}_0$$



Update and Propagate Steps: General Case

At time $k > 0$:
Filtering Density (Update):

$$p(\mathbf{z}_k \mid \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k \mid \mathbf{z}_k)p(\mathbf{z}_k \mid \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})}$$

Predictive Density (Propagate):

$$p(\mathbf{z}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{z}_k \mid \mathbf{z}_{k-1})p(\mathbf{z}_{k-1} \mid \mathbf{y}_{1:k-1})\, d\mathbf{z}_{k-1}$$

Meaning of the Densities
Posterior: $p(\mathbf{z}_k \mid \mathbf{y}_{1:k})$

- Probability of latent state given all observations

Prior: $p(\mathbf{z}_k \mid \mathbf{z}_{k-1})$

- Transition model: how latent state evolves over time

Likelihood: $p(\mathbf{y}_k \mid \mathbf{z}_k)$

- Probability of observation given latent state

Challenges in Bayesian Filtering

Why is direct solution difficult?

- Required integrals are *not tractable*
- Closed-form solutions exist only in limited cases

Example: Closed-form solution exists for

- Gaussian noise
- Linear state-space model
- $\Rightarrow$ Optimal solution via Kalman filter

Solution Strategy: Use Monte Carlo techniques to approximate integrals.

Classical Monte Carlo Integration

Goal: Estimate expectation

$$\mathbb{E}_Z[f(\mathbf{Z})] = \int f(\mathbf{z})p(\mathbf{z})\,d\mathbf{z}$$

Procedure:

1. Simulate M random samples $\{\mathbf{z}^{(m)}\}_{m=1}^{M}$ from $p(\mathbf{z})$
2. Compute empirical average:

$$\mathbb{E}_Z[f(\mathbf{Z})] \approx \hat{f} = \frac{1}{M}\sum_{m=1}^{M} f(\mathbf{z}^{(m)})$$

Example: Estimate variance of zero-mean Gaussian process.

Importance Sampling

Motivation: Sampling directly from $p(\mathbf{z})$ may be difficult.

Solution: Sample from proposal $q(\mathbf{z})$ and reweight samples.

Key Insight:

$$\mathbb{E}_Z[f(\mathbf{Z})] = \int f(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z})\, d\mathbf{z}$$

Importance Sampling Procedure

Steps:

1. Simulate M samples from proposal: $\{\mathbf{z}^{(m)}\}_{m=1}^{M} \sim q(\mathbf{z})$
2. Compute importance weights:

$$w^{(m)} = \frac{p(\mathbf{z}^{(m)})}{q(\mathbf{z}^{(m)})}$$

3. Compute weighted average:

$$\mathbb{E}_Z[f(\mathbf{Z})] \approx \sum_{m=1}^{M} \tilde{w}^{(m)} f(\mathbf{z}^{(m)})$$

where $\tilde{w}^{(m)} = w^{(m)} / \sum_{j=1}^{M} w^{(j)}$

Design Considerations for Proposal

Choose proposal $q(\mathbf{z})$ such that:

- **Easy to sample from**
- **Resembles original density:** $q(\mathbf{z})$ should be close to $p(\mathbf{z})$
- **Good overlap:** Minimizes variance of weight estimates

Importance sampling is *free* to choose the proposal density!

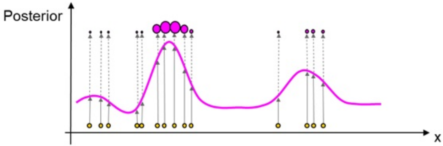
Core Idea:

- Update filtering density recursively using Bayesian filtering
- Compute integrals using importance sampling
- Represent posterior with particles and weights

Representation:

$$p(\mathbf{z}_k \mid \mathbf{y}_{1:k}) \approx \sum_{m=1}^{M} w_k^{(m)} \delta(\mathbf{z}_k - \mathbf{z}_k^{(m)})$$

where $\{\mathbf{z}_k^{(m)}\}$ are particles and $\{w_k^{(m)}\}$ are normalized weights.



Sequential Importance Sampling: Derivation (1/3)

Goal: posterior over the full trajectory

We consider the full posterior of states up to time k: $p(\mathbf{z}_{0:k} \mid \mathbf{y}_{1:k})$.

Bayesian recursion (factorization)

Using the state-space assumptions (Markov + conditional independence), we get:

$$
\begin{aligned}
p(\mathbf{z}_{0:k} \mid \mathbf{y}_{1:k}) &= \frac{p(\mathbf{y}_{1:k} \mid \mathbf{z}_{0:k}) p(\mathbf{z}_{0:k})}{p(\mathbf{y}_{1:k})} \propto p(\mathbf{y}_{1:k} \mid \mathbf{z}_{0:k}) p(\mathbf{z}_{0:k}) \\
&= p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \mathbf{z}_{0:k})\, p(\mathbf{y}_{1:k-1} \mid \mathbf{z}_{0:k})\, p(\mathbf{z}_{0:k}) \\
&\overset{\text{CI}}{=} p(\mathbf{y}_k \mid \mathbf{z}_k)\, p(\mathbf{y}_{1:k-1} \mid \mathbf{z}_{0:k-1})\, p(\mathbf{z}_k \mid \mathbf{z}_{0:k-1})\, p(\mathbf{z}_{0:k-1}) \\
&\overset{\text{Markov}}{=} p(\mathbf{y}_k \mid \mathbf{z}_k)\, p(\mathbf{z}_k \mid \mathbf{z}_{k-1})\, \underbrace{p(\mathbf{y}_{1:k-1} \mid \mathbf{z}_{0:k-1}) p(\mathbf{z}_{0:k-1})}_{\propto p(\mathbf{z}_{0:k-1} \mid \mathbf{y}_{1:k-1})} \\
&\propto p(\mathbf{y}_k \mid \mathbf{z}_k)\, p(\mathbf{z}_k \mid \mathbf{z}_{k-1})\, p(\mathbf{z}_{0:k-1} \mid \mathbf{y}_{1:k-1}).
\end{aligned}
$$

Sequential Importance Sampling: Derivation (2/3)

Start from the (unnormalized) IS weight definition

$$w_k^{(m)} \propto \frac{p(\mathbf{z}_{0:k}^{(m)} \mid \mathbf{y}_{1:k})}{q(\mathbf{z}_{0:k}^{(m)} \mid \mathbf{y}_{1:k})}.$$

Importance sampling idea (trajectory proposal)

Draw particles from a proposal $\mathbf{z}_{0:k}^{(m)} \sim q(\mathbf{z}_{0:k} \mid \mathbf{y}_{1:k}), m = 1, \ldots, M$,

and assign (unnormalized) importance weights proportional to target/proposal:

$$w_k^{(m)} \propto \frac{p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)})\, p(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{k-1}^{(m)})\, p(\mathbf{z}_{0:k-1}^{(m)} \mid \mathbf{y}_{1:k-1})}{q(\mathbf{z}_{0:k}^{(m)} \mid \mathbf{y}_{1:k})}.$$

Recursive (sequential) proposal

Choose the proposal to factor sequentially:

$$q(\mathbf{z}_{0:k} \mid \mathbf{y}_{1:k}) = q(\mathbf{z}_k \mid \mathbf{z}_{0:k-1}, \mathbf{y}_{1:k})\, q(\mathbf{z}_{0:k-1} \mid \mathbf{y}_{1:k-1}).$$

Sequential Importance Sampling: Derivation (3/3)

Derive weight recursion

Using the posterior recursion and the proposal factorization above, terms cancel and we obtain:

$$
\begin{aligned}
w_k^{(m)} &\propto \frac{p(\mathbf{z}_{0:k}^{(m)} \mid \mathbf{y}_{1:k})}{q(\mathbf{z}_{0:k}^{(m)} \mid \mathbf{y}_{1:k})} \\
&\propto \frac{p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)})\, p(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{k-1}^{(m)})\, p(\mathbf{z}_{0:k-1}^{(m)} \mid \mathbf{y}_{1:k-1})}{q(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{0:k-1}^{(m)}, \mathbf{y}_{1:k})\, q(\mathbf{z}_{0:k-1}^{(m)} \mid \mathbf{y}_{1:k-1})} \\
&\propto \underbrace{\frac{p(\mathbf{z}_{0:k-1}^{(m)} \mid \mathbf{y}_{1:k-1})}{q(\mathbf{z}_{0:k-1}^{(m)} \mid \mathbf{y}_{1:k-1})}}_{\propto\, w_{k-1}^{(m)}} \; \frac{p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)})\, p(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{k-1}^{(m)})}{q(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{0:k-1}^{(m)}, \mathbf{y}_{1:k})}.
\end{aligned}
$$

$$
\boxed{w_k^{(m)} \propto w_{k-1}^{(m)} \, \frac{p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)})\; p(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{k-1}^{(m)})}{q(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{0:k-1}^{(m)}, \mathbf{y}_{1:k})}.}
$$

Sequential Importance Sampling (SIS): Algorithm

Initialization

Draw particles from the prior and set uniform weights:

$$\mathbf{z}_0^{(m)} \sim p(\mathbf{z}_0), \qquad w_0^{(m)} = \frac{1}{M}, \qquad m = 1, \ldots, M.$$

For $k = 1, 2, \ldots$ (Prediction + Update)

1. **Prediction / Proposal sampling:** draw new particles

$$\mathbf{z}_k^{(m)} \sim q(\mathbf{z}_k \mid \mathbf{z}_{0:k-1}^{(m)}, \mathbf{y}_{1:k}).$$

2. **Weight update:** update unnormalized weights using the recursion

$$w_k^{(m)} \propto w_{k-1}^{(m)} \frac{p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)})\, p(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{k-1}^{(m)})}{q(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{0:k-1}^{(m)}, \mathbf{y}_{1:k})}.$$

3. **Normalize:** $\tilde{w}_k^{(m)} = \dfrac{w_k^{(m)}}{\sum_{j=1}^{M} w_k^{(j)}}.$

Prior Proposal

Assumption: prior proposal

We choose the proposal equal to the transition model:

$$\boxed{q(\mathbf{z}_k \mid \mathbf{z}_{k-1}^{(m)}, \mathbf{y}_k) = p(\mathbf{z}_k \mid \mathbf{z}_{k-1}^{(m)}).}$$

Why the weight computation simplifies

In the general SIS weight recursion, the incremental factor is

$$\frac{p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)})\, p(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{k-1}^{(m)})}{q(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{k-1}^{(m)}, \mathbf{y}_k)}.$$

If we choose the **prior proposal** $q = p(\mathbf{z}_k \mid \mathbf{z}_{k-1})$, then the transition term *cancels*:

$$\frac{p(\mathbf{z}_k^{(m)} \mid \mathbf{z}_{k-1}^{(m)})}{q(\cdot)} = 1 \quad \Rightarrow \quad w_k^{*(m)} \propto w_{k-1}^{(m)}\, p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)}).$$

Particle Filter (Prior Proposal / Bootstrap PF)

Initialization

Draw particles from the prior and set uniform weights:
$\mathbf{z}_0^{(m)} \sim p(\mathbf{z}_0), \qquad w_0^{(m)} = \frac{1}{M}, \qquad m = 1, \ldots, M.$

For $k = 1, 2, \ldots$ (Prediction + Update)

1. Step 1: Propagate (particle generation) For each particle $m = 1, \ldots, M$:

$$\mathbf{z}_k^{(m)} \sim p(\mathbf{z}_k \mid \mathbf{z}_{k-1}^{(m)}).$$

2. Step 2: Weight update (simplified) + normalization

$$w_k^{*(m)} = w_{k-1}^{(m)}\, p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)}), \qquad w_k^{(m)} = \frac{w_k^{*(m)}}{\sum_{j=1}^{M} w_k^{*(j)}}.$$

3. Step 3: Compute estimate Posterior expectation (for any test function g):

$$\mathbb{E}[g(\mathbf{z}_k) \mid \mathbf{y}_{1:k}] \approx \sum_{m=1}^{M} w_k^{(m)}\, g(\mathbf{z}_k^{(m)}).$$

The Resampling Problem

Weight Degeneracy:

- Over time, most particles have negligible weights
- Computational resources wasted on particles that don't contribute

Solution: Resampling

- Replicate particles in proportion to their weights
- High-weight particles sampled multiple times
- Low-weight particles eliminated

Resampling Step

Standard Resampling:
Sample new particles and reset weights:

$$\{\tilde{\mathbf{z}}_k^{(m)}\}_{m=1}^{M}, \quad \{\tilde{w}_k^{(m)} = 1/M\}_{m=1}^{M}$$

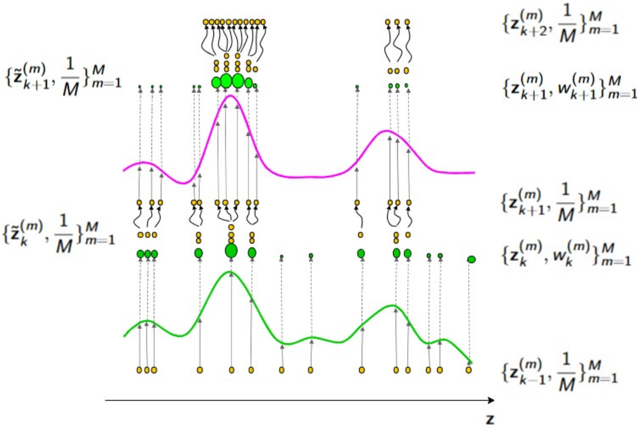
from the empirical distribution:

$$\sum_{j=1}^{M} w_k^{(j)} \delta(\mathbf{z}_k - \mathbf{z}_k^{(j)})$$

Effect:

- Particles with high weights are duplicated
- Particles with low weights disappear
- All weights reset to uniform: $w_k^{(m)} = 1/M$

Visualization of resampling



Complete Particle Filter Algorithm

Initialization

Draw particles from the prior and set uniform weights:
$\mathbf{z}_0^{(m)} \sim p(\mathbf{z}_0), \qquad w_0^{(m)} = \frac{1}{M}, \qquad m = 1, \ldots, M.$

For $k = 1, 2, \ldots$ (Prediction + Update)

1. Step 1: Propagate (particle generation) For each particle $m = 1, \ldots, M$:

$$\mathbf{z}_k^{(m)} \sim p(\mathbf{z}_k \mid \mathbf{z}_{k-1}^{(m)}).$$

2. Step 2: Weight update (simplified) + normalization

$$w_k^{*(m)} = w_{k-1}^{(m)} \, p(\mathbf{y}_k \mid \mathbf{z}_k^{(m)}), \qquad w_k^{(m)} = \frac{w_k^{*(m)}}{\sum_{j=1}^{M} w_k^{*(j)}}.$$

3. Step 3: Resample if needed (weight variance high)
4. Step 4: Compute estimate

$$\hat{\mathbf{z}}_k = \sum_m w_k^{(m)} \mathbf{z}_k^{(m)}$$

Bearings-Only Tracking: Model

Latent State Vector:

$$\mathbf{z}_k = [s_{x,k}, V_{x,k}, s_{y,k}, V_{y,k}]^\top$$

Latent State Equation:

$$\mathbf{z}_k = \mathbf{F}\mathbf{z}_{k-1} + \mathbf{u}_k$$

where $\mathbf{u}_k \sim \mathcal{N}(0, \sigma_u^2)$ and

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

Observation Equation:

$$y_k = \arctan(s_{y,k}/s_{x,k}) + v_k$$

where $v_k \sim \mathcal{N}(0, \sigma_v^2)$

Bearings-Only Tracking: Algorithm

Particle Generation:

$$\mathbf{u}_k^{(m)} \sim \mathcal{N}(0, \sigma_u^2)$$

$$\mathbf{z}_k^{(m)} = \mathbf{F}\mathbf{z}_{k-1}^{(m)} + \mathbf{u}_k^{(m)}$$

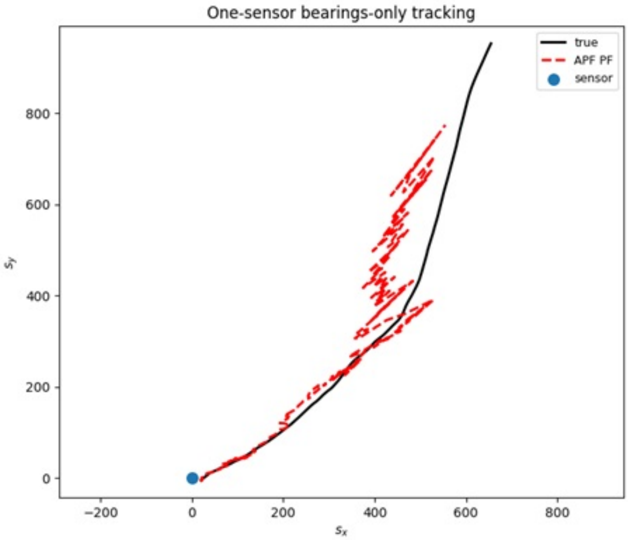
Weight Computation:

$$w_k^{*(m)} = w_{k-1}^{(m)} \mathcal{N}\left(y_k; \arctan(s_{y,k}^{(m)}/s_{x,k}^{(m)}), \sigma_v^2\right)$$

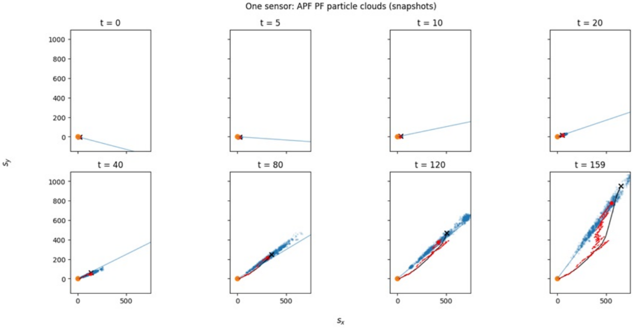
Normalization & Resampling:

$$w_k^{(m)} = \frac{w_k^{*(m)}}{\sum_{j=1}^{M} w_k^{*(j)}}, \quad \text{then resample}$$

Bearings-Only Tracking: Results



Bearings-Only Tracking: Results



Pay attention to the angles!
Measurements vs True

Particle Filter with Observation-Dependent Proposal (1/2)

Motivation: why include the observation in the proposal?

If we propose z_k using only the transition $p(z_k \mid z_{k-1})$, many particles land where $p(y_k \mid z_k)$ is tiny $\Rightarrow$ a few particles get almost all the weight (degeneracy).

General proposal

Choose any proposal that can use the new observation: $q(z_k \mid z_{k-1}, y_k)$.

Correct importance-weight update

For particle m:

$$\boxed{w_k^{(m)} \propto w_{k-1}^{(m)} \frac{p(y_k \mid z_k^{(m)})\, p(z_k^{(m)} \mid z_{k-1}^{(m)})}{q(z_k^{(m)} \mid z_{k-1}^{(m)}, y_k)}}$$

Normalize: $w_k^{(m)} = \dfrac{w_k^{(m)}}{\sum_{j=1}^{M} w_k^{(j)}}$.

Quick mental example

- Suppose y_k strongly suggests $z_k \approx 10$.
- Prior proposal draws z_k near where $p(z_k \mid z_{k-1})$ is large (maybe far from 10).
- Observation-dependent proposal shifts draws toward 10 $\Rightarrow$ fewer wasted samples.

Intuition

A good q puts particles in regions of high likelihood (where the data says the state should be), so weights are less variable.

Particle Filter with Optimal Proposal (2/2)

Optimal proposal (variance-minimizing)

$$q_k^{\mathrm{opt}}(z_k \mid z_{k-1}, y_k) = p(z_k \mid z_{k-1}, y_k)$$

This choice minimizes the conditional variance of the incremental weights.

Key simplification of the weights

Start from the general update:

$$w_k \propto w_{k-1} \, \frac{p(y_k \mid z_k) \, p(z_k \mid z_{k-1})}{q(z_k \mid z_{k-1}, y_k)}.$$

With $q = q^{\mathrm{opt}} = p(z_k \mid z_{k-1}, y_k)$ and Bayes' rule:

$$p(z_k \mid z_{k-1}, y_k) = \frac{p(y_k \mid z_k) p(z_k \mid z_{k-1})}{p(y_k \mid z_{k-1})},$$

so the ratio collapses to the **predictive likelihood**:

$$\boxed{w_k^{(m)} \propto w_{k-1}^{(m)} \, p(y_k \mid z_{k-1}^{(m)})} \qquad p(y_k \mid z_{k-1}) = \int p(y_k \mid z_k) p(z_k \mid z_{k-1}) \, dz_k.$$

Advantages of Particle Filters

- **Arbitrary densities:** Represent multimodal, non-Gaussian distributions
- **Adaptive:** Focus on probable regions of state-space
- **Non-Gaussian noise:** Handle directly without approximation
- **Multiple models:** Framework supports switching models
- **Nonlinear flexibility:** No requirement for linearity

Disadvantages of Particle Filters

- **High complexity:** Computational cost scales with number of particles
- **Particle selection:** Difficult to determine optimal number M a priori
- **Curse of dimensionality:** M must increase with state dimension
- **Weight degeneracy:** Many particles accumulate negligible weights
- **Loss of diversity:** Resampling can cause particle collapse
- **Proposal choice:** Optimal density is problem-dependent

Computational Complexity: Bearings-Only Example

Scenario: $M = 1000$ particles

Per iteration:

- Four random number generations per particle
- Propagation through state equation (matrix multiplication)
- M evaluations of nonlinear functions:
  - Exponential
  - Arctangent
- Weight normalization
- Resampling (if needed)

Overall: $\mathcal{O}(M)$ per time step

Complexity Trade-offs

Component	Cost per Particle	Total
Random number generation	$\mathcal{O}(1)$	$\mathcal{O}(M)$
State propagation	$\mathcal{O}(d^2)$	$\mathcal{O}(Md^2)$
Likelihood evaluation	$\mathcal{O}(1)$	$\mathcal{O}(M)$
Weight normalization	-	$\mathcal{O}(M)$
Resampling	-	$\mathcal{O}(M \log M)$

where d is state dimension.

Summary

Particle Filters: Powerful framework for nonlinear, non-Gaussian estimation.

Key Features:

- Represent arbitrary posterior densities
- Recursive Bayesian filtering via Monte Carlo
- Flexible enough for diverse applications
- Practical algorithm with clear steps

Trade-offs:

- Computational cost vs. accuracy
- Curse of dimensionality
- Algorithm design choices (resampling, proposal)

Acknowledgements/ Document preparation

This document was developed and converted into LaTeX slides and formatted with assistance from ChatGPT (OpenAI) and CoPilot (Microsoft). The instructor modified the source text and verified the final structure and wording.