Development and Implementation of SLAM Algorithms

Kasra Khosoussi

Supervised by: Dr. Hamid D. Taghirad

Advanced Robotics and Automated Systems (ARAS)

Industrial Control Laboratory

K.N. Toosi University of Technology

July 13, 2011

Outline

1. Introduction
   - SLAM Problem
   - Bayesian Filtering
   - Particle Filter
2. RBPF-SLAM
3. Monte Carlo Approximation of the Optimal Proposal Distribution
   - Introduction
   - LRS
   - LIS-1
   - LIS-2
4. Results
   - Simulation Results
   - Experiments On Real Data
5. Conclusion
6. Future Work

Table of Contents

Autonomous Mobile Robots

- The ultimate goal of mobile robotics is to design autonomous mobile robots.

- "The ability to simultaneously localize a robot and accurately map its environment is a key prerequisite of truly autonomous robots."
- SLAM stands for Simultaneous Localization and Mapping.

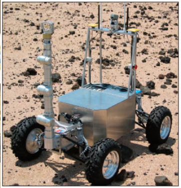
Autonomous Mobile Robots

- The ultimate goal of mobile robotics is to design autonomous mobile robots.

- “The ability to simultaneously localize a robot and accurately map its environment is a key prerequisite of truly autonomous robots.”
- SLAM stands for Simultaneous Localization and Mapping.

Autonomous Mobile Robots

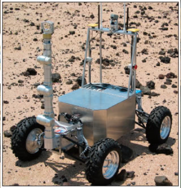
- The ultimate goal of mobile robotics is to design autonomous mobile robots.



- "The ability to simultaneously localize a robot and accurately map its environment is a key prerequisite of truly autonomous robots."
- SLAM stands for Simultaneous Localization and Mapping.

Autonomous Mobile Robots

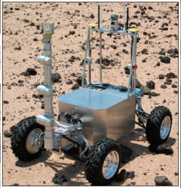
- The ultimate goal of mobile robotics is to design autonomous mobile robots.




- "The ability to simultaneously localize a robot and accurately map its environment is a key prerequisite of truly autonomous robots."
- SLAM stands for Simultaneous Localization and Mapping.

Autonomous Mobile Robots

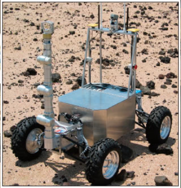
- The ultimate goal of mobile robotics is to design autonomous mobile robots.



- "The ability to simultaneously localize a robot and accurately map its environment is a key prerequisite of truly autonomous robots."
- SLAM stands for Simultaneous Localization and Mapping.

Autonomous Mobile Robots

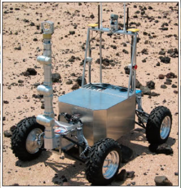
- The ultimate goal of mobile robotics is to design autonomous mobile robots.



- "The ability to simultaneously localize a robot and accurately map its environment is a key prerequisite of truly autonomous robots."
- SLAM stands for Simultaneous Localization and Mapping.

Autonomous Mobile Robots

- The ultimate goal of mobile robotics is to design autonomous mobile robots.



- "The ability to simultaneously localize a robot and accurately map its environment is a key prerequisite of truly autonomous robots."
- SLAM stands for Simultaneous Localization and Mapping.

The SLAM Problem

Assumptions

- No *a priori* knowledge about the environment (i.e. map)
- No independent position information (i.e. GPS)
- Static environment

Given

- Observations of the environment
- Control signals

Goal

- Estimate the map of the environment (e.g. locations of the features)
- Estimate the position and orientation (pose) of the robot

The SLAM Problem

Assumptions

- No *a priori* knowledge about the environment (i.e. map)
- No independent position information (i.e. GPS)
- Static environment

Given

- Observations of the environment
- Control signals

Goal

- Estimate the map of the environment (e.g. locations of the features)
- Estimate the position and orientation (pose) of the robot

The SLAM Problem

Assumptions

- No *a priori* knowledge about the environment (i.e. map)
- No independent position information (i.e. GPS)
- Static environment

Given

- Observations of the environment
- Control signals

Goal

- Estimate the map of the environment (e.g. locations of the features)
- Estimate the position and orientation (pose) of the robot

Map Representation

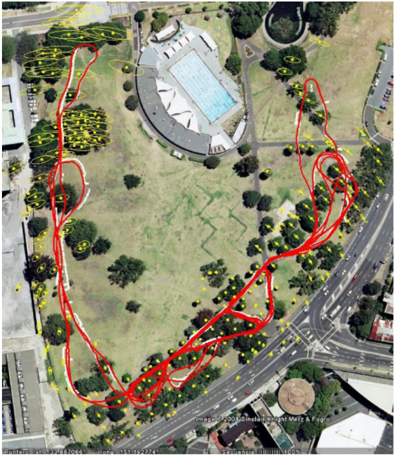
- Topological maps
- Grid maps
- Feature-based maps

Map Representation

- Topological maps
- Grid maps
- Feature-based maps

Map Representation

- Topological maps
- Grid maps
- Feature-based maps



Probabilistic Methods

Probabilistic methods outperform deterministic algorithms

- Describe uncertainty in data, models and estimates
- Bayesian estimation
  - Robot pose $\mathbf{s}_t$
  - Robot pose is assumed to be a Markov process with initial distribution $p(\mathbf{s}_0)$
  - Feature's location θ_i
  - Map $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$
  - Observation $\mathbf{z}_t$, and control input $\mathbf{u}_t$
  - $\mathbf{x}_t = [\mathbf{s}_t \quad \boldsymbol{\theta}]^T$
  - $\mathbf{x}_{1:t} \triangleq \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$
- Filtering distribution: $p(\mathbf{s_t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Smoothing distribution: $p(\mathbf{s_{0:t}}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- MMSE estimate:
  $\hat{\mathbf{x}}_t = \mathbb{E}[\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$
  $\hat{\mathbf{x}}_{0:t} = \mathbb{E}[\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$

Probabilistic Methods

Probabilistic methods outperform deterministic algorithms

- Describe uncertainty in data, models and estimates
- Bayesian estimation
  - Robot pose $\mathbf{s}_t$
  - Robot pose is assumed to be a Markov process with initial distribution $p(\mathbf{s}_0)$
  - Feature's location θ_i
  - Map $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$
  - Observation $\mathbf{z}_t$, and control input $\mathbf{u}_t$
  - $\mathbf{x}_t = [\mathbf{s}_t \quad \boldsymbol{\theta}]^T$
  - $\mathbf{x}_{1:t} \triangleq \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$
- Filtering distribution: $p(\mathbf{s_t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Smoothing distribution: $p(\mathbf{s_{0:t}}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- MMSE estimate: $\hat{\mathbf{x}}_t = \mathbb{E}[\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$ $\hat{\mathbf{x}}_{0:t} = \mathbb{E}[\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$

Probabilistic Methods

Probabilistic methods outperform deterministic algorithms

- Describe uncertainty in data, models and estimates
- Bayesian estimation
  - Robot pose $\mathbf{s}_t$
  - Robot pose is assumed to be a Markov process with initial distribution $p(\mathbf{s}_0)$
  - Feature's location θ_i
  - Map $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$
  - Observation $\mathbf{z}_t$, and control input $\mathbf{u}_t$
  - $\mathbf{x}_t = [\mathbf{s}_t \quad \boldsymbol{\theta}]^T$
  - $\mathbf{x}_{1:t} \triangleq \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$

- Filtering distribution: $p(\mathbf{s_t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Smoothing distribution: $p(\mathbf{s_{0:t}}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- MMSE estimate:
  $\hat{\mathbf{x}}_t = \mathbb{E}[\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$
  $\hat{\mathbf{x}}_{0:t} = \mathbb{E}[\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$

Probabilistic Methods

Probabilistic methods outperform deterministic algorithms

- Describe uncertainty in data, models and estimates
- Bayesian estimation
- Robot pose $\mathbf{s}_t$
- Robot pose is assumed to be a Markov process with initial distribution $p(\mathbf{s}_0)$
- Feature's location θ_i
- Map $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$
- Observation $\mathbf{z}_t$, and control input $\mathbf{u}_t$
- $\mathbf{x}_t = [\mathbf{s}_t \quad \boldsymbol{\theta}]^T$
- $\mathbf{x}_{1:t} \triangleq \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$

- Filtering distribution: $p(\mathbf{s_t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Smoothing distribution: $p(\mathbf{s_{0:t}}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- MMSE estimate: $\hat{\mathbf{x}}_t = \mathbb{E}[\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$ $\hat{\mathbf{x}}_{0:t} = \mathbb{E}[\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$

Probabilistic Methods

Probabilistic methods outperform deterministic algorithms

- Describe uncertainty in data, models and estimates
- Bayesian estimation
  - Robot pose $\mathbf{s}_t$
  - Robot pose is assumed to be a Markov process with initial distribution $p(\mathbf{s}_0)$
  - Feature's location θ_i
  - Map $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$
  - Observation $\mathbf{z}_t$, and control input $\mathbf{u}_t$
  - $\mathbf{x}_t = [\mathbf{s}_t \quad \boldsymbol{\theta}]^T$
  - $\mathbf{x}_{1:t} \triangleq \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$

- Filtering distribution: $p(\mathbf{s_t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Smoothing distribution: $p(\mathbf{s_{0:t}}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- MMSE estimate:
  $\hat{\mathbf{x}}_t = \mathbb{E}[\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$
  $\hat{\mathbf{x}}_{0:t} = \mathbb{E}[\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$

Probabilistic Methods

Probabilistic methods outperform deterministic algorithms

- Describe uncertainty in data, models and estimates
- Bayesian estimation
  - Robot pose $\mathbf{s}_t$
  - Robot pose is assumed to be a Markov process with initial distribution $p(\mathbf{s}_0)$
  - Feature's location θ_i
  - Map $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$
  - Observation $\mathbf{z}_t$, and control input $\mathbf{u}_t$
  - $\mathbf{x}_t = [\mathbf{s}_t \quad \boldsymbol{\theta}]^T$
  - $\mathbf{x}_{1:t} \triangleq \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$

- Filtering distribution: $p(\mathbf{s_t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Smoothing distribution: $p(\mathbf{s_{0:t}}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- MMSE estimate:
  $\hat{\mathbf{x}}_t = \mathbb{E}[\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$
  $\hat{\mathbf{x}}_{0:t} = \mathbb{E}[\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$

Probabilistic Methods

Probabilistic methods outperform deterministic algorithms

- Describe uncertainty in data, models and estimates
- Bayesian estimation
- Robot pose $\mathbf{s}_t$
- Robot pose is assumed to be a Markov process with initial distribution $p(\mathbf{s}_0)$
- Feature's location θ_i
- Map $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$
- Observation $\mathbf{z}_t$, and control input $\mathbf{u}_t$
- $\mathbf{x}_t = [\mathbf{s}_t \quad \boldsymbol{\theta}]^T$
- $\mathbf{x}_{1:t} \triangleq \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$

- Filtering distribution: $p(\mathbf{s_t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Smoothing distribution: $p(\mathbf{s_{0:t}}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- MMSE estimate:
  $\hat{\mathbf{x}}_t = \mathbb{E}[\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$
  $\hat{\mathbf{x}}_{0:t} = \mathbb{E}[\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}]$

State-Space Equations

- Robot motion equation:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{u}_t, \mathbf{v}_t)$$

- Observation equation:

$$\mathbf{z}_t = g(\mathbf{s}_t, \theta_{n_t}, \mathbf{w}_t)$$

State-Space Equations

- Robot motion equation:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{u}_t, \mathbf{v}_t)$$

- Observation equation:

$$\mathbf{z}_t = g(\mathbf{s}_t, \theta_{n_t}, \mathbf{w}_t)$$

- $f(\cdot, \cdot, \cdot)$ and $g(\cdot, \cdot, \cdot)$ are non-linear functions

State-Space Equations

- Robot motion equation:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{u}_t, \mathbf{v}_t)$$

- Observation equation:

$$\mathbf{z}_t = g(\mathbf{s}_t, \theta_{n_t}, \mathbf{w}_t)$$

- $f(\cdot,\cdot,\cdot)$ and $g(\cdot,\cdot,\cdot)$ are non-linear functions
- $\mathbf{v}_t$ and $\mathbf{w}_t$ are zero-mean white Gaussian noises with covariances matrices $\mathbf{Q}_t$ and $\mathbf{R}_t$

Bayes Filter

How to estimate the posterior distribution recursively in time? Bayes filter!

1. Prediction
2. Update

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)\, p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t-1})\, d\mathbf{x}_{t-1} \quad (1)$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t},\mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t})}{\int p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}) d\mathbf{x}_t} \quad (2)$$

Motion model

Observation model

There is a similar recursive formula for the smoothing density

Bayes Filter

How to estimate the posterior distribution recursively in time? Bayes filter!

1. Prediction
2. Update

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)\, p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})\, d\mathbf{x}_{t-1} \quad (1)$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{\int p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_t} \quad (2)$$

Motion model

Observation model

There is a similar recursive formula for the smoothing density

Bayes Filter

How to estimate the posterior distribution recursively in time? Bayes filter!

1. Prediction
2. Update

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)\, p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t-1})\, d\mathbf{x}_{t-1} \tag{1}$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t},\mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t})}{\int p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}) d\mathbf{x}_t} \tag{2}$$

Motion model

Observation model

There is a similar recursive formula for the smoothing density

Bayes Filter

How to estimate the posterior distribution recursively in time? Bayes filter!

1. Prediction
2. Update

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)\; p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t-1})\, d\mathbf{x}_{t-1} \tag{1}$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t},\mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)\; p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t})}{\int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t})d\mathbf{x}_t} \tag{2}$$

Motion model

Observation model

There is a similar recursive formula for the smoothing density

Bayes Filter

How to estimate the posterior distribution recursively in time? Bayes filter!

1. Prediction
2. Update

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)\ p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})\ d\mathbf{x}_{t-1} \quad (1)$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)\ p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{\int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})d\mathbf{x}_t} \quad (2)$$

Motion model

Observation model

There is a similar recursive formula for the smoothing density

Bayes Filter

How to estimate the posterior distribution recursively in time? Bayes filter!

1. Prediction
2. Update

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \; p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \, d\mathbf{x}_{t-1} \quad (1)$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t) \; p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{\int p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_t} \quad (2)$$

Motion model

Observation model

There is a similar recursive formula for the smoothing density

Bayes Filter

How to estimate the posterior distribution recursively in time? Bayes filter!

1. Prediction
2. Update

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)\; p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t-1})\, d\mathbf{x}_{t-1} \quad (1)$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t},\mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)\; p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t})}{\int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t})d\mathbf{x}_t} \quad (2)$$

Motion model

Observation model

There is a similar recursive formula for the smoothing density

Bayes Filter Cont'd.

Example

For the case of linear-Gaussian models, Bayes filter equations would be simplified into the Kalman filter equations.

Bayes Filter Cont'd.

Example

For the case of linear-Gaussian models, Bayes filter equations would be simplified into the Kalman filter equations.

But . . .

In general, it is impossible to implement the exact Bayes filter because it requires the ability to evaluate complex high-dimensional integrals.

Bayes Filter Cont'd.

Example

For the case of linear-Gaussian models, Bayes filter equations would be simplified into the Kalman filter equations.

But ...

In general, it is impossible to implement the exact Bayes filter because it requires the ability to evaluate complex high-dimensional integrals.

So we have to use approximation ...

- Extended Kalman Filter (EKF)
- Unscented Kalman Filter (UKF)
- Gaussian-Sum Filter
- Extended Information Filter (EIF)
- Particle Filter (A.K.A. Sequential Monte Carlo Methods)

Bayes Filter Cont'd.

Example

For the case of linear-Gaussian models, Bayes filter equations would be simplified into the Kalman filter equations.

But . . .

In general, it is impossible to implement the exact Bayes filter because it requires the ability to evaluate complex high-dimensional integrals.

So we have to use approximation . . .

- Extended Kalman Filter (EKF)
- Unscented Kalman Filter (UKF)
- Gaussian-Sum Filter
- Extended Information Filter (EIF)
- Particle Filter (A.K.A. Sequential Monte Carlo Methods)

Perfect Monte Carlo Sampling

Q. How to compute expected values such as $\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x}$ for any integrable function $h(\cdot)$?

Perfect Monte Carlo Sampling

Q. How to compute expected values such as $\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x}$ for any integrable function $h(\cdot)$?

Perfect Monte Carlo (A.K.A. Monte Carlo Integration)

1. Generate N i.i.d. samples $\{\mathbf{x}^{[i]}\}_{i=1}^{N}$ according to $p(\mathbf{x})$
2. Estimate the PDF as $P_N(\mathbf{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}^{[i]})$
3. Estimate $\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] \approx \int h(\mathbf{x})P_N(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} h(\mathbf{x}^{[i]})$

Perfect Monte Carlo Sampling

Q. How to compute expected values such as $\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x}$ for any integrable function $h(\cdot)$?

Perfect Monte Carlo (A.K.A. Monte Carlo Integration)

1. Generate N i.i.d. samples $\{\mathbf{x}^{[i]}\}_{i=1}^{N}$ according to $p(\mathbf{x})$
2. Estimate the PDF as $P_N(\mathbf{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}-\mathbf{x}^{[i]})$
3. Estimate $\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] \approx \int h(\mathbf{x})P_N(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} h(\mathbf{x}^{[i]})$

- Convergence theorems for $N \to \infty$ using central limit theorem and strong law of large numbers
- Error decreases with $O(N^{-1/2})$ regardless of the dimension of $\mathbf{x}$

Perfect Monte Carlo Sampling

Q. How to compute expected values such as $\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x}$ for any integrable function $h(\cdot)$?

Perfect Monte Carlo (A.K.A. Monte Carlo Integration)

1. Generate N i.i.d. samples $\{\mathbf{x}^{[i]}\}_{i=1}^{N}$ according to $p(\mathbf{x})$
2. Estimate the PDF as $P_N(\mathbf{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}-\mathbf{x}^{[i]})$
3. Estimate $\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] \approx \int h(\mathbf{x})P_N(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} h(\mathbf{x}^{[i]})$

- Convergence theorems for $N \to \infty$ using central limit theorem and strong law of large numbers
- Error decreases with $O(N^{-1/2})$ regardless of the dimension of $\mathbf{x}$

But ...

It is usually impossible to sample directly from the filtering or smoothing distribution (high-dimensional, non-standard, only known up to a constant)

Importance Sampling (IS)

Idea

Generate samples from another distribution called the importance function like $\pi(\mathbf{x})$, and weight these samples according to $w^*(\mathbf{x}^{[i]}) = \frac{p(\mathbf{x}^{[i]})}{\pi(\mathbf{x}^{[i]})}$:

Importance Sampling (IS)

Idea

Generate samples from another distribution called the importance function like $\pi(\mathbf{x})$, and weight these samples according to $w^*(\mathbf{x}^{[i]}) = \frac{p(\mathbf{x}^{[i]})}{\pi(\mathbf{x}^{[i]})}$:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\pi(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} w^*(\mathbf{x}^{[i]})h(\mathbf{x}^{[i]})$$

Importance Sampling (IS)

Idea

Generate samples from another distribution called the importance function like $\pi(\mathbf{x})$, and weight these samples according to $w^*(\mathbf{x}^{[i]}) = \frac{p(\mathbf{x}^{[i]})}{\pi(\mathbf{x}^{[i]})}$:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\pi(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} w^*(\mathbf{x}^{[i]})h(\mathbf{x}^{[i]})$$

But . . .

How to do this recursively in time?

Sequential Importance Sampling (SIS)

Sampling from scratch from the importance function $\pi(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ implies growing computational complexity for each step over time

Sequential Importance Sampling (SIS)

Sampling from scratch from the importance function $\pi(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ implies growing computational complexity for each step over time

Q. How to estimate $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using importance sampling recursively?

Sequential Importance Sampling (SIS)

Sampling from scratch from the importance function $\pi(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ implies growing computational complexity for each step over time

Q. How to estimate $p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using importance sampling recursively?

Sequential Importance Sampling

At time t, generate $\mathbf{x}_t^{[i]}$ according to $\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{[i]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ (proposal distribution), and merge it with the previous samples $\mathbf{x}_{0:t-1}^{[i]}$ drawn from $\pi(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})$:

$$\mathbf{x}_{0:t}^{[i]} = \{\mathbf{x}_{0:t-1}^{[i]}, \mathbf{x}_t^{[i]}\} \sim \pi(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

$$w(\mathbf{x}_{0:t}^{[i]}) = w(\mathbf{x}_{0:t-1}^{[i]}) \frac{p(\mathbf{z}_t|\mathbf{x}_t^{[i]})p(\mathbf{x}_t^{[i]}|\mathbf{x}_{t-1}^{[i]}, \mathbf{u}_t)}{\pi(\mathbf{x}_t^{[i]}|\mathbf{x}_{0:t-1}^{[i]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}$$

Degeneracy and Resampling

Degeneracy Problem

After a few steps, all but one of the particles (samples) would have very insignificant normalized weight

Degeneracy and Resampling

Degeneracy Problem

After a few steps, all but one of the particles (samples) would have very insignificant normalized weight

Resampling

Eliminate particles with low normalized weights and multiply those with high normalized weights in a probabilistic manner

Degeneracy and Resampling

Degeneracy Problem

After a few steps, all but one of the particles (samples) would have very insignificant normalized weight

Resampling

Eliminate particles with low normalized weights and multiply those with high normalized weights in a probabilistic manner

- Resampling will cause sample impoverishment

Degeneracy and Resampling

Degeneracy Problem

After a few steps, all but one of the particles (samples) would have very insignificant normalized weight

Resampling

Eliminate particles with low normalized weights and multiply those with high normalized weights in a probabilistic manner

- Resampling will cause sample impoverishment
- Effective sample size (ESS) is a measure of the degeneracy of SIS that can be used in order to avoid unnecessary resampling steps

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \tilde{w}(\mathbf{x}_{0:t}^{[i]})^2}$$

Perform resampling only if N_{eff} is lower than a fixed threshold N_{T}

Proposal Distribution

- Selecting an appropriate proposal distribution $\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ plays an important role in the success of particle filter
- The simplest and most common choice is the motion model (transition density) $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$
- $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \boxed{\mathbf{z}_t}, \mathbf{u}_t)$ is known as the **optimal proposal distribution** and limits the degeneracy of the particle filter by minimizing the conditional variance of unnormalized weights
- Importance weights for the optimal proposal distribution can be obtained as:

$$w(\mathbf{x}_{0:t}^{[i]}) = w(\mathbf{x}_{0:t-1}^{[i]})p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$$

But . . .

In SLAM, neither $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{u}_t)$ nor $p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$ can be computed in closed form and we have to use approximation

Proposal Distribution

- Selecting an appropriate proposal distribution $\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ plays an important role in the success of particle filter
- The simplest and most common choice is the motion model (transition density) $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$
- $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{u}_t)$ is known as the **optimal proposal distribution** and limits the degeneracy of the particle filter by minimizing the conditional variance of unnormalized weights
- Importance weights for the optimal proposal distribution can be obtained as:

$$w(\mathbf{x}_{0:t}^{[i]}) = w(\mathbf{x}_{0:t-1}^{[i]})p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$$

But . . .

In SLAM, neither $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{u}_t)$ nor $p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$ can be computed in closed form and we have to use approximation

Proposal Distribution

- Selecting an appropriate proposal distribution $\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ plays an important role in the success of particle filter
- The simplest and most common choice is the motion model (transition density) $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$
- $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{u}_t)$ is known as the **optimal proposal distribution** and limits the degeneracy of the particle filter by minimizing the conditional variance of unnormalized weights
- Importance weights for the optimal proposal distribution can be obtained as:

$$w(\mathbf{x}_{0:t}^{[i]}) = w(\mathbf{x}_{0:t-1}^{[i]})p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$$

But ...

In SLAM, neither $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{u}_t)$ nor $p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$ can be computed in closed form and we have to use approximation

Proposal Distribution

- Selecting an appropriate proposal distribution $\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ plays an important role in the success of particle filter
- The simplest and most common choice is the motion model (transition density) $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$
- $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \boxed{\mathbf{z}_t}, \mathbf{u}_t)$ is known as the **optimal proposal distribution** and limits the degeneracy of the particle filter by minimizing the conditional variance of unnormalized weights
- Importance weights for the optimal proposal distribution can be obtained as:

$$w(\mathbf{x}_{0:t}^{[i]}) = w(\mathbf{x}_{0:t-1}^{[i]})p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$$

But ...

In SLAM, neither $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{u}_t)$ nor $p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$ can be computed in closed form and we have to use approximation

Proposal Distribution

- Selecting an appropriate proposal distribution $\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{z}_{1:t},\mathbf{u}_{1:t})$ plays an important role in the success of particle filter
- The simplest and most common choice is the motion model (transition density) $p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)$
- $p(\mathbf{x}_t|\mathbf{x}_{t-1},\ \mathbf{z}_t\ ,\mathbf{u}_t)$ is known as the **optimal proposal distribution** and limits the degeneracy of the particle filter by minimizing the conditional variance of unnormalized weights
- Importance weights for the optimal proposal distribution can be obtained as:

$$w(\mathbf{x}_{0:t}^{[i]}) = w(\mathbf{x}_{0:t-1}^{[i]})p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$$

But . . .

In SLAM, neither $p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{z}_t,\mathbf{u}_t)$ nor $p(\mathbf{z}_t|\mathbf{x}_{t-1}^{[i]})$ can be computed in closed form and we have to use approximation

Table of Contents

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k | \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- Motion model is used as the proposal distribution in FastSLAM 1.0
- FastSLAM 2.0 linearizes the observation equation and approximates the optimal proposal distribution with a Gaussian distribution
- FastSLAM 2.0 outperforms FastSLAM 1.0

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k|\mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- Motion model is used as the proposal distribution in FastSLAM 1.0
- FastSLAM 2.0 linearizes the observation equation and approximates the optimal proposal distribution with a Gaussian distribution
- FastSLAM 2.0 outperforms FastSLAM 1.0

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k| \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- Motion model is used as the proposal distribution in FastSLAM 1.0
- FastSLAM 2.0 linearizes the observation equation and approximates the optimal proposal distribution with a Gaussian distribution
- FastSLAM 2.0 outperforms FastSLAM 1.0

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k| \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- Motion model is used as the proposal distribution in FastSLAM 1.0
- FastSLAM 2.0 linearizes the observation equation and approximates the optimal proposal distribution with a Gaussian distribution
- FastSLAM 2.0 outperforms FastSLAM 1.0

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k | \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- Motion model is used as the proposal distribution in FastSLAM 1.0
- FastSLAM 2.0 linearizes the observation equation and approximates the optimal proposal distribution with a Gaussian distribution
- FastSLAM 2.0 outperforms FastSLAM 1.0

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k | \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- Motion model is used as the proposal distribution in FastSLAM 1.0
- FastSLAM 2.0 linearizes the observation equation and approximates the optimal proposal distribution with a Gaussian distribution
- FastSLAM 2.0 outperforms FastSLAM 1.0

FastSLAM 2.0

- Linearization error

FastSLAM 2.0

- Linearization error
- Gaussian approximation

FastSLAM 2.0

- Linearization error
- Gaussian approximation
- Linear motion models with respect to the noise variable $\mathbf{v}_t$

Table of Contents

FastSLAM 2.0 v.s. The Proposed Algorithms

FastSLAM 2.0

1. Approximate the optimal proposal distribution with a Gaussian using the linearized observation equation. Sample from this Gaussian

FastSLAM 2.0 v.s. The Proposed Algorithms

FastSLAM 2.0

1. Approximate the optimal proposal distribution with a Gaussian using the linearized observation equation. Sample from this Gaussian
2. Update the landmarks EKFs for the observed features

FastSLAM 2.0 v.s. The Proposed Algorithms

FastSLAM 2.0

1. Approximate the optimal proposal distribution with a Gaussian using the linearized observation equation. Sample from this Gaussian
2. Update the landmarks EKFs for the observed features
3. Compute the importance weights using the linearized observation equation

FastSLAM 2.0 v.s. The Proposed Algorithms

FastSLAM 2.0

1. Approximate the optimal proposal distribution with a Gaussian using the linearized observation equation. Sample from this Gaussian
2. Update the landmarks EKFs for the observed features
3. Compute the importance weights using the linearized observation equation
4. Perform resampling

FastSLAM 2.0 v.s. The Proposed Algorithms

FastSLAM 2.0

1. Approximate the optimal proposal distribution with a Gaussian using the linearized observation equation. Sample from this Gaussian
2. Update the landmarks EKFs for the observed features
3. Compute the importance weights using the linearized observation equation
4. Perform resampling

Proposed Algorithms

1. Sample from the optimal proposal distribution using Monte Carlo sampling methods

FastSLAM 2.0 v.s. The Proposed Algorithms

FastSLAM 2.0

1. Approximate the optimal proposal distribution with a Gaussian using the linearized observation equation. Sample from this Gaussian
2. Update the landmarks EKFs for the observed features
3. Compute the importance weights using the linearized observation equation
4. Perform resampling

Proposed Algorithms

1. Sample from the optimal proposal distribution using Monte Carlo sampling methods
2. Update the landmarks EKFs for the observed features

FastSLAM 2.0 v.s. The Proposed Algorithms

FastSLAM 2.0

1. Approximate the optimal proposal distribution with a Gaussian using the linearized observation equation. Sample from this Gaussian
2. Update the landmarks EKFs for the observed features
3. Compute the importance weights using the linearized observation equation
4. Perform resampling

Proposed Algorithms

1. Sample from the optimal proposal distribution using Monte Carlo sampling methods
2. Update the landmarks EKFs for the observed features
3. Compute the importance weights using Monte Carlo integration

FastSLAM 2.0 v.s. The Proposed Algorithms

FastSLAM 2.0

1. Approximate the optimal proposal distribution with a Gaussian using the linearized observation equation. Sample from this Gaussian
2. Update the landmarks EKFs for the observed features
3. Compute the importance weights using the linearized observation equation
4. Perform resampling

Proposed Algorithms

1. Sample from the optimal proposal distribution using Monte Carlo sampling methods
2. Update the landmarks EKFs for the observed features
3. Compute the importance weights using Monte Carlo integration
4. Perform resampling only if it is necessary according to ESS

MC Approximation of The Optimal Proposal Dist.

- MC sampling methods such as importance sampling (IS) and rejection sampling (RS) can be used in order to sample from the optimal proposal distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$

MC Approximation of The Optimal Proposal Dist.

- MC sampling methods such as importance sampling (IS) and rejection sampling (RS) can be used in order to sample from the optimal proposal distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$

Instead of sampling directly from the optimal proposal distribution . . .

- We can generate M samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to another distribution like $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$ and then weight those samples proportional to $\frac{p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}{q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}$

MC Approximation of The Optimal Proposal Dist.

- MC sampling methods such as importance sampling (IS) and rejection sampling (RS) can be used in order to sample from the optimal proposal distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$

Instead of sampling directly from the optimal proposal distribution ...

- We can generate M samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to another distribution like $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$ and then weight those samples proportional to $\frac{p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}{q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}$ Local Importance Sampling (LIS)

MC Approximation of The Optimal Proposal Dist.

- MC sampling methods such as importance sampling (IS) and rejection sampling (RS) can be used in order to sample from the optimal proposal distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$

Instead of sampling directly from the optimal proposal distribution . . .

- We can generate M samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to another distribution like $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$ and then weight those samples proportional to $\frac{p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}{q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}$ Local Importance Sampling (LIS)
- We can generate M samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to another distribution like $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$ and accept some of them as samples of the optimal proposal distribution according to rejection sampling criteria

MC Approximation of The Optimal Proposal Dist.

- MC sampling methods such as importance sampling (IS) and rejection sampling (RS) can be used in order to sample from the optimal proposal distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$

Instead of sampling directly from the optimal proposal distribution ...

- We can generate M samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to another distribution like $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$ and then weight those samples proportional to $\frac{p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)}{q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)}$ Local Importance Sampling (LIS)
- We can generate M samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to another distribution like $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$ and accept some of them as samples of the optimal proposal distribution according to rejection sampling criteria Local Rejection Sampling (LRS)

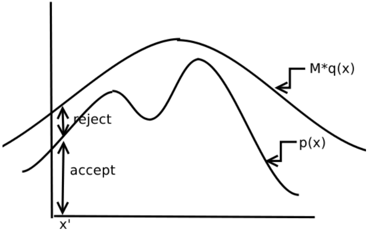
MC Approximation of The Optimal Proposal Dist.

- MC sampling methods such as importance sampling (IS) and rejection sampling (RS) can be used in order to sample from the optimal proposal distribution $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$

Instead of sampling directly from the optimal proposal distribution ...

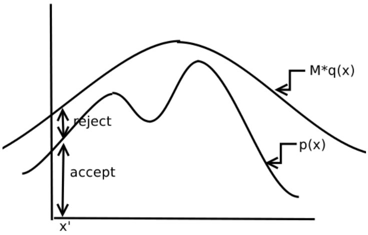
- We can generate M samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to another distribution like $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$ and then weight those samples proportional to $\frac{p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)}{q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)}$ Local Importance Sampling (LIS)
- We can generate M samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to another distribution like $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)$ and accept some of them as samples of the optimal proposal distribution according to rejection sampling criteria Local Rejection Sampling (LRS)
- $q(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t, \mathbf{z}_t) = p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$

Local Rejection Sampling (LRS)



graphics by M. Jordan

Local Rejection Sampling (LRS)

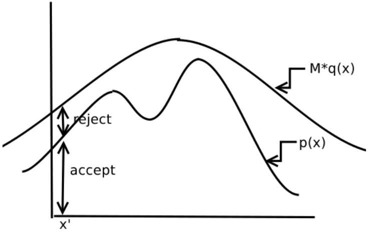


graphics by M. Jordan

Rejection Sampling

1. Generate $u^{[i]} \sim \mathcal{U}[0,1]$ and $\mathbf{s}_t^{[i,j]} \sim p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$

Local Rejection Sampling (LRS)

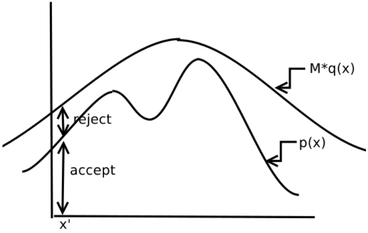


graphics by M. Jordan

Rejection Sampling

1. Generate $u^{[i]} \sim \mathcal{U}[0,1]$ and $\mathbf{s}_t^{[i,j]} \sim p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
2. Accept $\mathbf{s}_t^{[i,j]}$ if $u^{[i]} \leq \frac{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}{C \cdot p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{u}_t)}$

Local Rejection Sampling (LRS)



graphics by M. Jordan

Rejection Sampling

1. Generate $u^{[i]} \sim \mathcal{U}[0,1]$ and $\mathbf{s}_t^{[i,j]} \sim p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
2. Accept $\mathbf{s}_t^{[i,j]}$ if $u^{[i]} \leq \frac{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)}{C \cdot p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)} = \frac{p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})}{\max_j p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})}$

LRS Cont'd.

Now we have to compute the importance weights for the set of accepted samples

LRS Cont'd.

Now we have to compute the importance weights for the set of accepted samples

- We should compute $p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]})$

LRS Cont'd.

Now we have to compute the importance weights for the set of accepted samples

- We should compute $p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]})$

$$p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]}) = \int p(\mathbf{z}_t|\mathbf{s}_t)p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)d\mathbf{s}_t \underset{\text{MC Intergration}}{\approx} \frac{1}{M}\sum_{j=1}^{M} p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$$

LRS Cont'd.

Now we have to compute the importance weights for the set of accepted samples

- We should compute $p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]})$

$$p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]}) = \int p(\mathbf{z}_t|\mathbf{s}_t)p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)d\mathbf{s}_t \underset{\text{MC Intergration}}{\approx} \frac{1}{M}\sum_{j=1}^{M} p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$$

- $p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$ can be approximated by a Gaussian

LRS Cont'd.

Now we have to compute the importance weights for the set of accepted samples

- We should compute $p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]})$

$$p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]}) = \int p(\mathbf{z}_t|\mathbf{s}_t)p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)d\mathbf{s}_t \underset{\text{MC Intergration}}{\approx} \frac{1}{M}\sum_{j=1}^{M} p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$$

- $p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$ can be approximated by a Gaussian

MC Integration

LRS Cont'd.

Now we have to compute the importance weights for the set of accepted samples

- We should compute $p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]})$

$$p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]}) = \int p(\mathbf{z}_t|\mathbf{s}_t)p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)d\mathbf{s}_t \underset{\text{MC Intergration}}{\approx} \frac{1}{M}\sum_{j=1}^{M} p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$$

- $p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$ can be approximated by a Gaussian

MC Integration $\Rightarrow$ Large number of local particles M

Local Importance Sampling (LIS)

LIS-1

LIS-1

1. Similar to LRS, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$

Local Importance Sampling (LIS)

LIS-1

LIS-1

1. Similar to LRS, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
2. Local IS weights: $w^{(\mathsf{LIS})}(\mathbf{s}_t^{[i,j]}) = p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]}) \propto \frac{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{u}_t)}$

Local Importance Sampling (LIS)

LIS-1

LIS-1

1. Similar to LRS, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
2. Local IS weights: $w^{(\mathsf{LIS})}(\mathbf{s}_t^{[i,j]}) = p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]}) \propto \frac{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{u}_t)}$
3. Local resampling among $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ using $w^{(\mathsf{LIS})}(\mathbf{s}_t^{[i,j]})$

Local Importance Sampling (LIS)

LIS-1

LIS-1

1. Similar to LRS, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
2. Local IS weights: $w^{(\text{LIS})}(\mathbf{s}_t^{[i,j]}) = p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]}) \propto \frac{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]}, \mathbf{z}_t, \mathbf{u}_t)}{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)}$
3. Local resampling among $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ using $w^{(\text{LIS})}(\mathbf{s}_t^{[i,j]})$
4. Main weights: $w(\mathbf{s}_t^{*[i,j]}) = w(\mathbf{s}_{t-1}^{[i]})p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]})$

Local Importance Sampling (LIS)

LIS-1

LIS-1

1. Similar to LRS, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^M$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]},\mathbf{u}_t)$
2. Local IS weights: $w^{(\text{LIS})}(\mathbf{s}_t^{[i,j]}) = p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]}) \propto \frac{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{u}_t)}$
3. Local resampling among $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^M$ using $w^{(\text{LIS})}(\mathbf{s}_t^{[i,j]})$
4. Main weights: $w(\mathbf{s}_t^{*[i,j]}) = w(\mathbf{s}_{t-1}^{[i]})p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]})$
5. $p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]}) \underset{\text{MC Integration}}{\approx} \frac{1}{M}\sum_{j=1}^M p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$

Local Importance Sampling (LIS)

LIS-1

LIS-1

1. Similar to LRS, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
2. Local IS weights: $w^{(\mathsf{LIS})}(\mathbf{s}_t^{[i,j]}) = p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]}) \propto \frac{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{z}_t,\mathbf{u}_t)}{p(\mathbf{s}_t^{[i,j]}|\mathbf{s}_{t-1}^{[i]},\mathbf{u}_t)}$
3. Local resampling among $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ using $w^{(\mathsf{LIS})}(\mathbf{s}_t^{[i,j]})$
4. Main weights: $w(\mathbf{s}_t^{*[i,j]}) = w(\mathbf{s}_{t-1}^{[i]})p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]})$
5. $p(\mathbf{z}_t|\mathbf{s}_{t-1}^{[i]}) \underset{\text{MC Integration}}{\approx} \frac{1}{M}\sum_{j=1}^{M} p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$

Local Importance Sampling (LIS)

LIS-2

LIS-2

- Similar to LRS and LIS-1, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$

Local Importance Sampling (LIS)

LIS-2

LIS-2

- Similar to LRS and LIS-1, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
- Total weights of generated particles $\{\mathbf{s}_t^{[i,j]}\}_{(i,j)=(1,1)}^{N,M}$ = local weights $\times$ SIS weights

Local Importance Sampling (LIS)

LIS-2

LIS-2

- Similar to LRS and LIS-1, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
- Total weights of generated particles $\{\mathbf{s}_t^{[i,j]}\}_{(i,j)=(1,1)}^{N,M}$ = local weights $\times$ SIS weights
- Instead of eliminating local weights through local resamplings (LIS-1), total weights are computed in LIS-2

Local Importance Sampling (LIS)

LIS-2

LIS-2

- Similar to LRS and LIS-1, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
- Total weights of generated particles $\{\mathbf{s}_t^{[i,j]}\}_{(i,j)=(1,1)}^{N,M}$ = local weights $\times$ SIS weights
- Instead of eliminating local weights through local resamplings (LIS-1), total weights are computed in LIS-2
- It is proved in the thesis that total weights would be equal to

$$w(\mathbf{s}_{0:t}^{[i,j]}) = w(\mathbf{s}_{0:t-1}^{[i]})p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$$

Local Importance Sampling (LIS)
LIS-2

LIS-2

- Similar to LRS and LIS-1, we have to generate M random samples $\{\mathbf{s}_t^{[i,j]}\}_{j=1}^{M}$ according to $p(\mathbf{s}_t|\mathbf{s}_{t-1}^{[i]}, \mathbf{u}_t)$
- Total weights of generated particles $\{\mathbf{s}_t^{[i,j]}\}_{(i,j)=(1,1)}^{N,M}$ = local weights $\times$ SIS weights
- Instead of eliminating local weights through local resamplings (LIS-1), total weights are computed in LIS-2
- It is proved in the thesis that total weights would be equal to

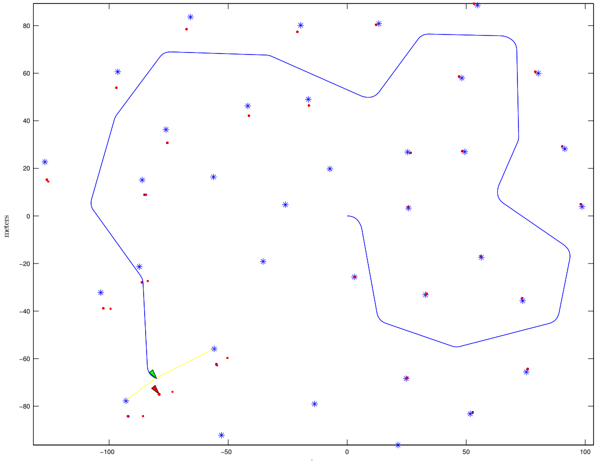
$$w(\mathbf{s}_{0:t}^{[i,j]}) = w(\mathbf{s}_{0:t-1}^{[i]})p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$$

Local weights are used in LIS-2 to filter those samples with low $p(\mathbf{z}_t|\mathbf{s}_t^{[i,j]})$

Table of Contents

Simulation

- 50 Monte Carlo runs with different seeds
- 200m $\times$ 200m simulated environment



Number of Resamplings

Table: Average number of resampling steps over 50 MC simulations

N	FastSLAM 2.0	LRS ($M = 50$)	LIS-2 ($M = 3$)
20	221.01	180.84	188.85
30	229.79	186	193.13
40	235.71	186.55	195.33
50	237.80	188.96	196.32
60	239.55	189.62	195.99
70	240.10	189.40	197.85
80	243.61	190.39	197.96
90	244.16	190.94	198.73
100	245.70	192.69	199.50

Runtime

Table: Average Runtime over 50 MC simulations

N	FastSLAM 2.0 (sec)	LRS $(M = 50)$ (sec)	LIS-2 $(M = 3)$ (sec)
20	36	318	38
30	52	480	56
40	68	635	75
50	84	794	93
60	103	953	112
70	117	1106	130
80	134	1265	148
90	149	1413	165
100	167	1588	183

Mean Square Error

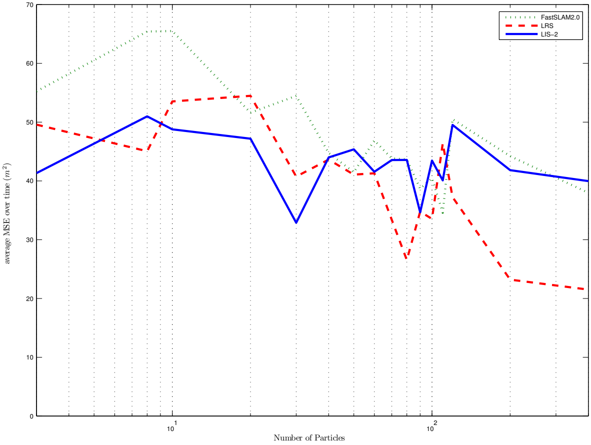


Figure: Average MSE of estimated robot pose over time. The parameter M is set to 50 for LRS and 3 for LIS-2.

Mean Square Error Cont'd.

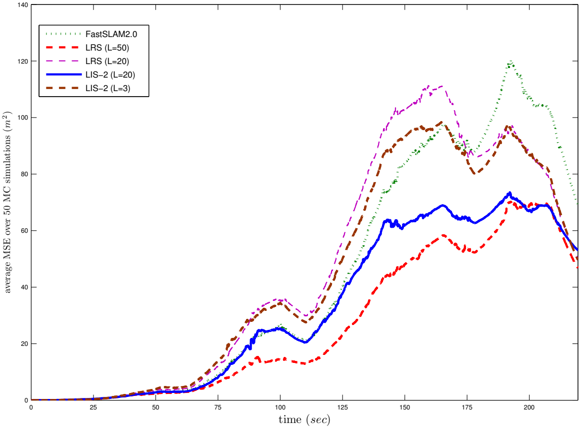


Figure: Average MSE of estimated robot position over 50 Monte Carlo simulations.

Victoria Park Dataset

- Victoria Park dataset by E. Nebot and J. Guivant

Victoria Park Dataset

- Victoria Park dataset by E. Nebot and J. Guivant
- Large environment 200m $\times$ 300m

Victoria Park Dataset

- Victoria Park dataset by E. Nebot and J. Guivant
- Large environment 200m $\times$ 300m
- More than 108,000 controls and observations

Victoria Park Dataset

- Victoria Park dataset by E. Nebot and J. Guivant
- Large environment 200m $\times$ 300m
- More than 108,000 controls and observations

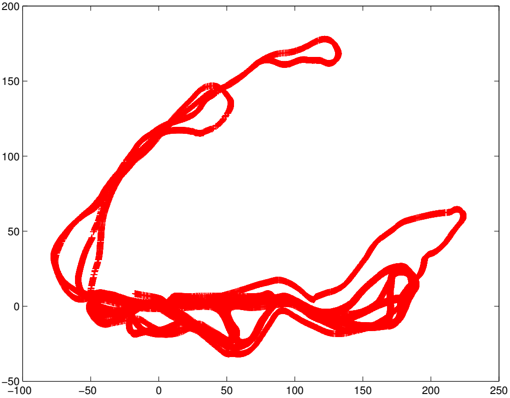


Figure: Estimated Robot Path

Table of Contents

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Conclusion

- LRS and LIS can describe non-Gaussian (e.g. multi-modal) proposal distributions
- Nonlinear motion models
- Linearization error
- LRS and LIS have control over the accuracy of the approximation through M
- Much lower number of resampling steps in LRS and LIS-2 than in FastSLAM 2.0 $\Rightarrow$ Slower rate of degeneracy $\Rightarrow$ Better approximation of the optimal proposal distribution $\Rightarrow$ Sample impoverishment problem
- Monte Carlo Integration in LRS and LIS-1 $\Rightarrow$ Large M $\Rightarrow$ High computational cost
- Accurate results of LRS come at the cost of large runtime
- LIS-2 (M=3) outperform FastSLAM 2.0 and LRS (M=50) for moderate number of particles (N)

Table of Contents

Future Work

- How to guess an "appropriate" value for M?
- More Monte Carlo runs
- Hybrid algorithm (linearization + Monte Carlo sampling)
- Repeat the simulations for the case of unknown data association

Future Work

- How to guess an "appropriate" value for M?
- More Monte Carlo runs
- Hybrid algorithm (linearization + Monte Carlo sampling)
- Repeat the simulations for the case of unknown data association

Future Work

- How to guess an "appropriate" value for M?
- More Monte Carlo runs
- Hybrid algorithm (linearization + Monte Carlo sampling)
- Repeat the simulations for the case of unknown data association

Future Work

- How to guess an "appropriate" value for M?
- More Monte Carlo runs
- Hybrid algorithm (linearization + Monte Carlo sampling)
- Repeat the simulations for the case of unknown data association

Papers

- IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2011: Monte-Carlo Approximation of The Optimal Proposal Distribution in RBPF-SLAM (submitted)
- . . .

Thank You . . .

Figure: **Melon:** The Semi-autonomous Mobile Robot of K.N. Toosi Univ. of Tech.

Thank You . . .

Figure: **Melon:** The Semi-autonomous Mobile Robot of K.N. Toosi Univ. of Tech.

Thank you . . .

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k|\mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- We can estimate $p(\mathbf{s}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter
- $p(\theta_k|\mathbf{s}_{0:t}^{[i]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ can be estimated using an EKF for each robot path particle $\mathbf{s}_{0:t}^{[i]}$
- A map (estimated locations of the features) is attached to each robot path particle

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k | \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- We can estimate $p(\mathbf{s}_{0:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter
- $p(\theta_k|\mathbf{s}_{0:t}^{[i]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ can be estimated using an EKF for each robot path particle $\mathbf{s}_{0:t}^{[i]}$
- A map (estimated locations of the features) is attached to each robot path particle

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k | \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- We can estimate $p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter
- $p(\theta_k | \mathbf{s}_{0:t}^{[i]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ can be estimated using an EKF for each robot path particle $\mathbf{s}_{0:t}^{[i]}$
- A map (estimated locations of the features) is attached to each robot path particle

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k | \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- We can estimate $p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter
- $p(\theta_k | \mathbf{s}_{0:t}^{[i]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ can be estimated using an EKF for each robot path particle $\mathbf{s}_{0:t}^{[i]}$
- A map (estimated locations of the features) is attached to each robot path particle

Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k | \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- We can estimate $p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter
- $p(\theta_k | \mathbf{s}_{0:t}^{[i]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ can be estimated using an EKF for each robot path particle $\mathbf{s}_{0:t}^{[i]}$
- A map (estimated locations of the features) is attached to each robot path particle

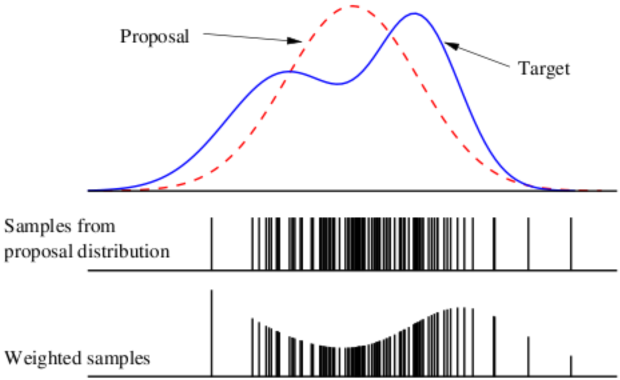
Rao-Blackwellized Particle Filter in SLAM

- SLAM is a very high-dimensional problem
- Estimating the $p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter can be very inefficient
- We can factor the smoothing distribution into two parts as

$$p(\mathbf{s}_{0:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_{k=1}^{M} p(\theta_k | \mathbf{s}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

- We can estimate $p(\mathbf{s}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ using a particle filter
- $p(\theta_k | \mathbf{s}_{0:t}^{[i]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ can be estimated using an EKF for each robot path particle $\mathbf{s}_{0:t}^{[i]}$
- A map (estimated locations of the features) is attached to each robot path particle

Importance Sampling



Importance Sampling (IS)

Idea

Generate samples from another distribution called the importance function like $\pi(\mathbf{x})$, and weight these samples according to $w^*(\mathbf{x}^{[i]}) = \frac{p(\mathbf{x}^{[i]})}{\pi(\mathbf{x}^{[i]})}$:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\pi(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} w^*(\mathbf{x}^{[i]})h(\mathbf{x}^{[i]})$$

In practice we compute importance weights $w(\mathbf{x})$ proportional to $\frac{p(\mathbf{x})}{\pi(\mathbf{x})}$ and normalize them to estimate the expected value as:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] \approx \sum_{i=1}^{N} \frac{w(\mathbf{x}^{[i]})}{\sum_{j=1}^{N} w(\mathbf{x}^{[j]})} h(\mathbf{x}^{[i]})$$

But ...

How to do this recursively in time?

Importance Sampling (IS)

Idea

Generate samples from another distribution called the importance function like $\pi(\mathbf{x})$, and weight these samples according to $w^*(\mathbf{x}^{[i]}) = \frac{p(\mathbf{x}^{[i]})}{\pi(\mathbf{x}^{[i]})}$:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\pi(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} w^*(\mathbf{x}^{[i]})h(\mathbf{x}^{[i]})$$

In practice we compute importance weights $w(\mathbf{x})$ proportional to $\frac{p(\mathbf{x})}{\pi(\mathbf{x})}$ and normalize them to estimate the expected value as:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] \approx \sum_{i=1}^{N} \frac{w(\mathbf{x}^{[i]})}{\sum_{j=1}^{N} w(\mathbf{x}^{[j]})} h(\mathbf{x}^{[i]})$$

But ...

How to do this recursively in time?

Importance Sampling (IS)

Idea

Generate samples from another distribution called the importance function like $\pi(\mathbf{x})$, and weight these samples according to $w^*(\mathbf{x}^{[i]}) = \frac{p(\mathbf{x}^{[i]})}{\pi(\mathbf{x}^{[i]})}$:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\pi(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} w^*(\mathbf{x}^{[i]})h(\mathbf{x}^{[i]})$$

In practice we compute importance weights $w(\mathbf{x})$ proportional to $\frac{p(\mathbf{x})}{\pi(\mathbf{x})}$ and normalize them to estimate the expected value as:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] \approx \sum_{i=1}^{N} \frac{w(\mathbf{x}^{[i]})}{\sum_{j=1}^{N} w(\mathbf{x}^{[j]})} h(\mathbf{x}^{[i]})$$

But ...

How to do this recursively in time?

Importance Sampling (IS)

Idea

Generate samples from another distribution called the importance function like $\pi(\mathbf{x})$, and weight these samples according to $w^*(\mathbf{x}^{[i]}) = \frac{p(\mathbf{x}^{[i]})}{\pi(\mathbf{x}^{[i]})}$:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] = \int h(\mathbf{x})\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\pi(\mathbf{x})d\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N} w^*(\mathbf{x}^{[i]})h(\mathbf{x}^{[i]})$$

In practice we compute importance weights $w(\mathbf{x})$ proportional to $\frac{p(\mathbf{x})}{\pi(\mathbf{x})}$ and normalize them to estimate the expected value as:

$$\mathbb{E}_{p(\mathbf{x})}[h(\mathbf{x})] \approx \sum_{i=1}^{N} \frac{w(\mathbf{x}^{[i]})}{\sum_{j=1}^{N} w(\mathbf{x}^{[j]})} h(\mathbf{x}^{[i]})$$

But ...

How to do this recursively in time?

SIR