Particle Filters

16-833 Robot Localization and Mapping

Fall 2024

Montiel Abello

Slides adapted from Eric Westman

State Estimation

Parametric Methods

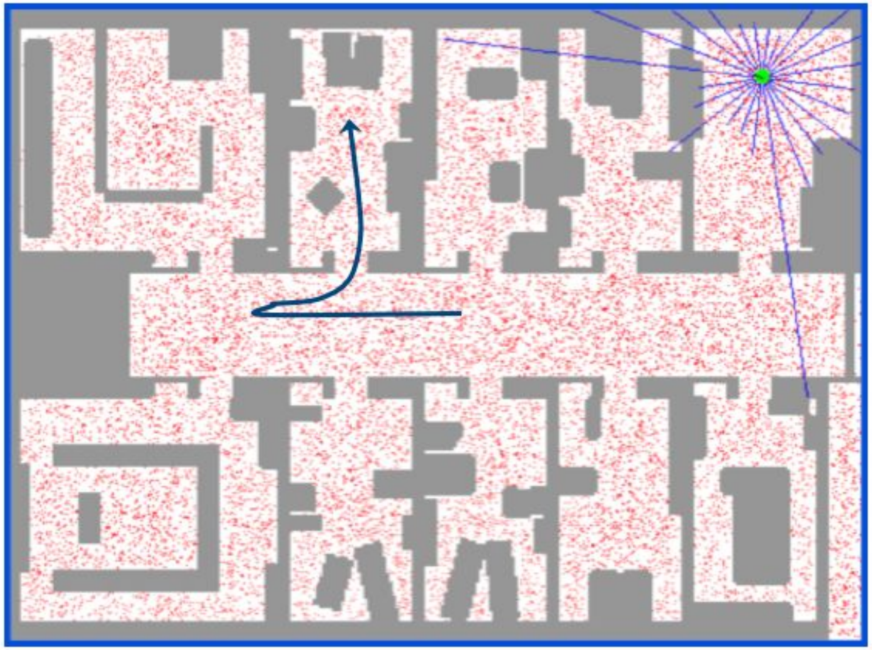
- Model state using a parametric distribution (e.g. normal distribution)
- Can give optimal estimate if assumptions hold
- Examples: Kalman filter, EKF, factor graph optimization

Non-parametric Methods

- Do not assume a particular model
- Allows tracking arbitrary distributions
- Often use particles or kernels to represent underlying distribution

Particle Filter Applications

- Localization (focus in this class)
- SLAM (e.g. FASTSLAM algorithm)
- State estimation, generally

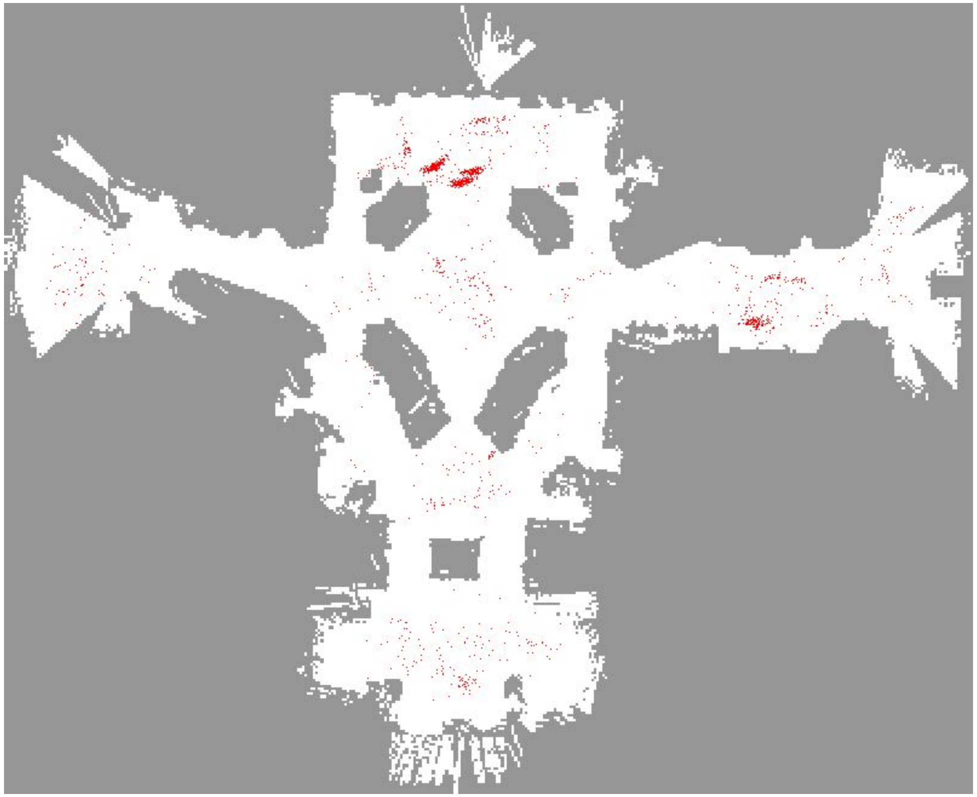


The Big Idea

Use particles as **samples** of the distribution that represents our **belief** of the state

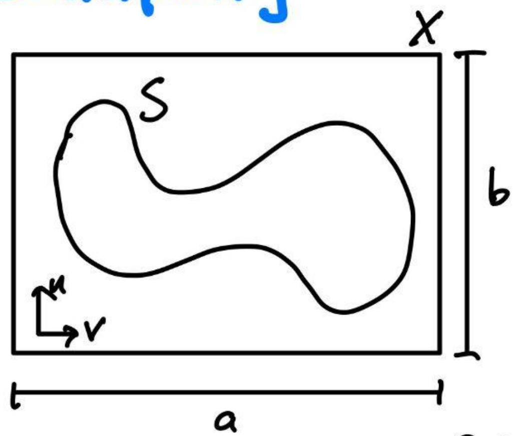
The Big Idea

Use particles as **samples** of the distribution that represents our **belief** of the state



Background...

Sampling



$$I(x) = I\left(\begin{bmatrix} u \\ v \end{bmatrix}\right) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}$$

Approximate Area:

- sample N $x_i \in X$
- Area $\approx \frac{1}{N} \sum_{i=1}^{N} I(x_i) a \cdot b$

Assumptions:

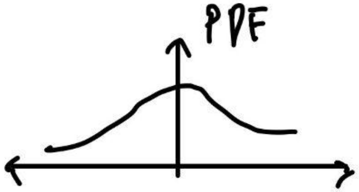
- have indicator function
- have method of drawing <u>uniformly distributed</u> random numbers

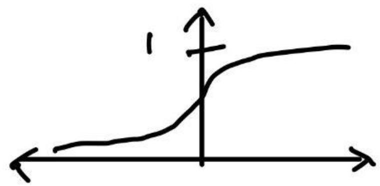
Why is sampling nontrivial?

- Bernoulli distribution: $x \in \{0, 1\}$
  $P(x = 1) = p$
- Uniform distribution: $x \sim Uni(0, 1)$
  - hardware, pseudorandom number generators
  - allow us to sample more complex distributions

Inverse Transform Sampling Method

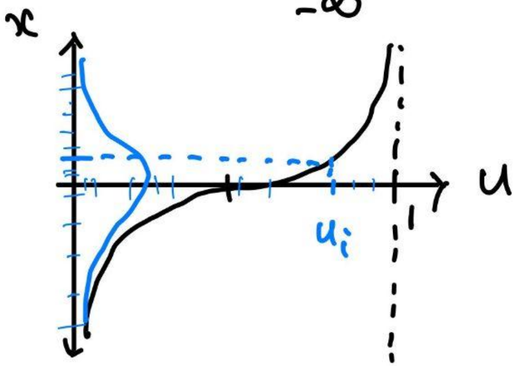
- e.g. sample 0-mean Gaussian





① find or approx CDF

$$CDF(x) = \int_{-\infty}^{x} P(t)\,dt$$



② generate uniformly distributed random no. $u_i \sim Uni(0,1)$

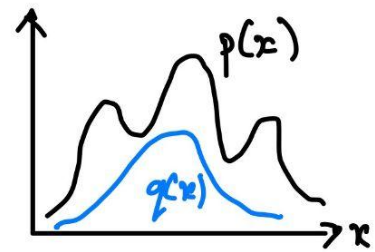
$$x_i = CDF^{-1}(u_i)$$

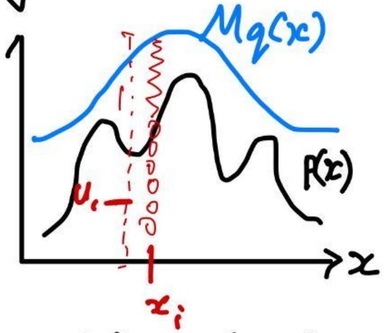
Problem: can't do this for multivariate distributions

Rejection Sampling

- Goal: sample target distribution $p(x)$ (can evaluate, can't sample)
- Known: proposal distribution $q(x)$ (can sample, can evaluate)

  e.g. uniform, gaussian





- Set up: Choose M s.t. $p(x) < Mq(x)$ over entire support of $p(x)$

① sample $x_i \sim q(x)$ and $u_i \sim Uni(0,1)$

② if $u_i < \dfrac{P(x_i)}{Mq(x_i)}$, accept x_i

else reject x_i, go to ①

$\Rightarrow$ densely sample where $p(x)$ close to $Mq(x)$ i.e. where probability density higher

Problem: in high dim, need $M \uparrow$
hard to get good fit

Importance Sampling

- Notation: $x \sim p(x)$

$$E[x] = \int_{-\infty}^{\infty} x\, p(x)\, dx \triangleq E_{p(x)}[x]$$

$$E_{p(x)}[f(x)] = \int_{-\infty}^{\infty} f(x)\, p(x)\, dx$$

- target $p(x)$ - can't sample
- proposal $q(x)$ - can sample

$$E_{p(x)}[f(x)] = \int f(x)\, p(x)\, dx$$
$$= \int f(x)\, p(x) \frac{q(x)}{q(x)}\, dx$$
$$= \int \frac{p(x)}{q(x)} f(x)\, q(x)\, dx$$
$$= E_{q(x)}\left[\frac{p(x)}{q(x)} f(x)\right]$$

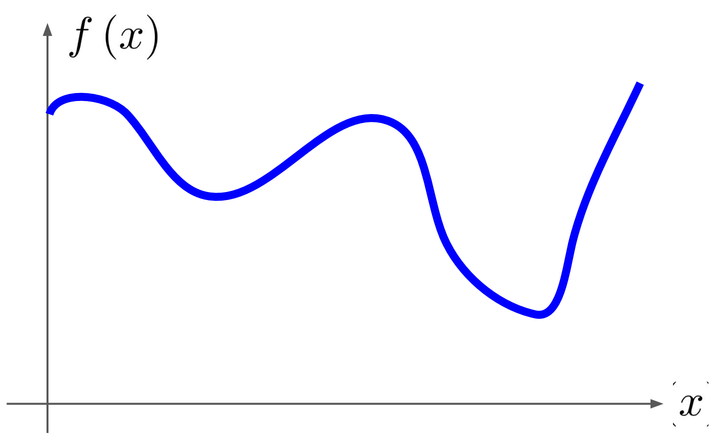
- now generate samples $x_i \sim q(x)$

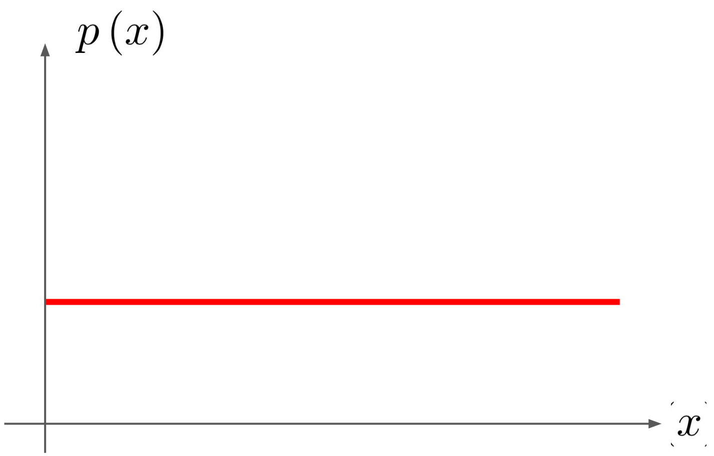
$$E_{p(x)}[f(x)] \simeq \frac{1}{N} \sum_{i=1}^{N} \frac{p(x_i)}{q(x_i)} f(x_i)$$

- need to know ratio $\frac{p(x)}{q(x)}$

Importance Sampling

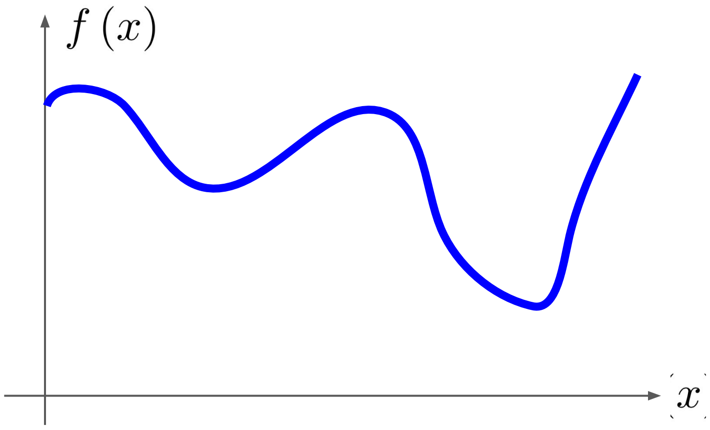
$$\begin{aligned}
\mathrm{E}_{p(x)}\left[f\left(x\right)\right] &= \int_{-\infty}^{\infty} f\left(x\right) p\left(x\right) dx \\
&= \int_{-\infty}^{\infty} f\left(x\right) p\left(x\right) \frac{q\left(x\right)}{q\left(x\right)} dx \\
&= \int_{-\infty}^{\infty} \frac{p\left(x\right)}{q\left(x\right)} f\left(x\right) q\left(x\right) dx \\
&= \mathrm{E}_{q(x)}\left[\frac{p\left(x\right)}{q\left(x\right)} f\left(x\right)\right]
\end{aligned}$$

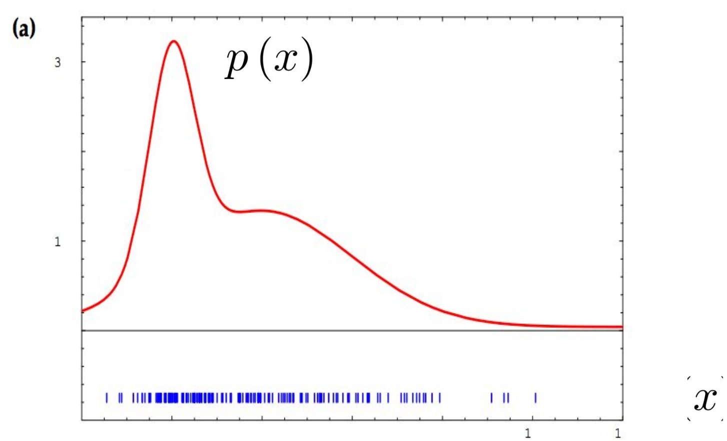




Importance Sampling

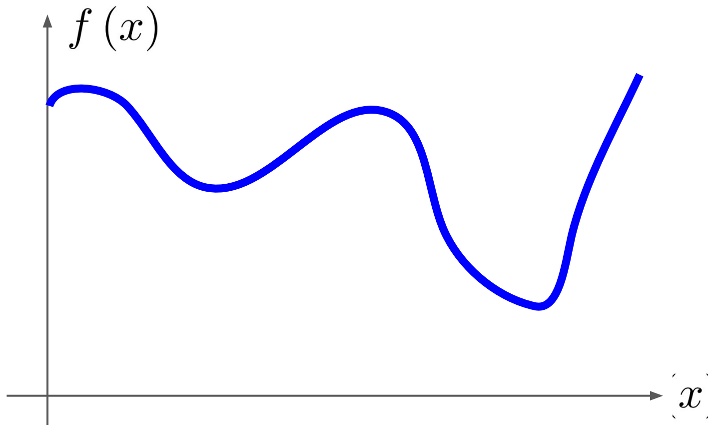
$$
\begin{aligned}
\mathrm{E}_{p(x)}\left[f(x)\right] &= \int_{-\infty}^{\infty} f(x)\,p(x)\,dx \\
&= \int_{-\infty}^{\infty} f(x)\,p(x)\,\frac{q(x)}{q(x)}dx \\
&= \int_{-\infty}^{\infty} \frac{p(x)}{q(x)}f(x)\,q(x)\,dx \\
&= \mathrm{E}_{q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]
\end{aligned}
$$

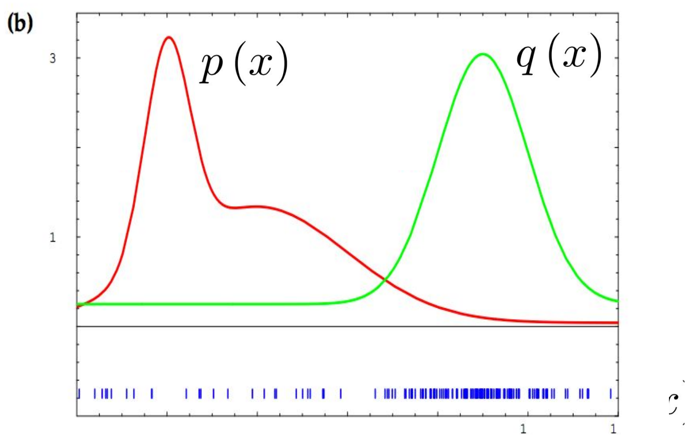




Importance Sampling

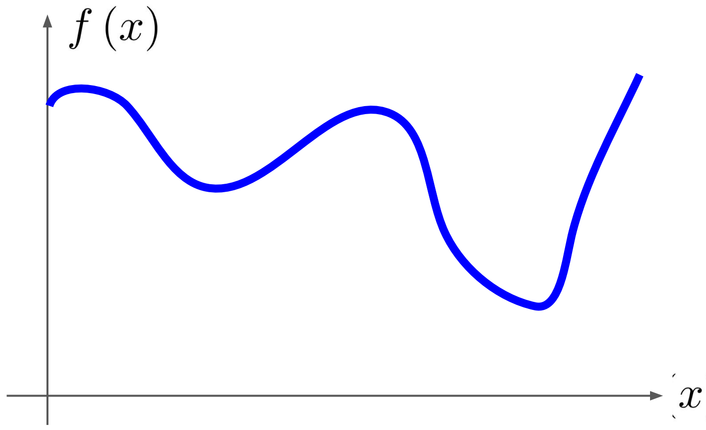
$$
\begin{aligned}
\mathrm{E}_{p(x)}\left[f(x)\right] &= \int_{-\infty}^{\infty} f(x)\,p(x)\,dx \\
&= \int_{-\infty}^{\infty} f(x)\,p(x)\,\frac{q(x)}{q(x)}dx \\
&= \int_{-\infty}^{\infty} \frac{p(x)}{q(x)} f(x)\,q(x)\,dx \\
&= \mathrm{E}_{q(x)}\left[\frac{p(x)}{q(x)} f(x)\right]
\end{aligned}
$$

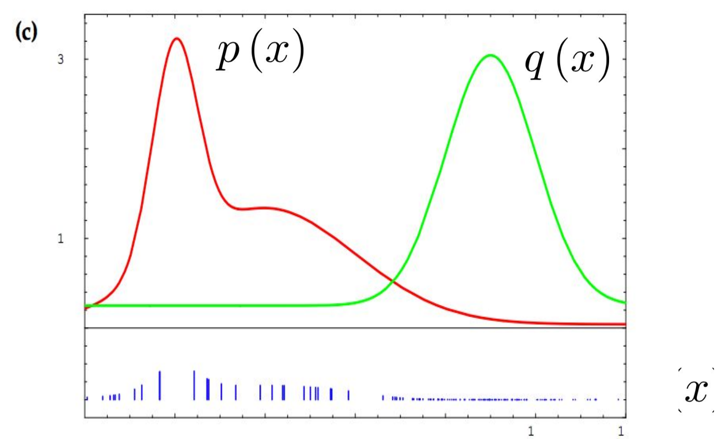




Importance Sampling

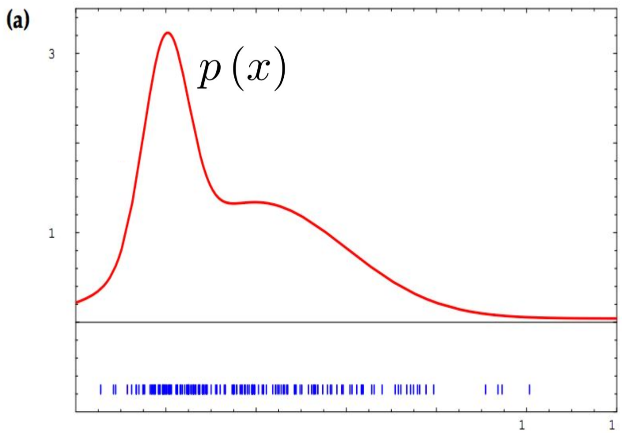
$$\begin{aligned}
\mathrm{E}_{p(x)}\left[f(x)\right] &= \int_{-\infty}^{\infty} f(x)\, p(x)\, dx \\
&= \int_{-\infty}^{\infty} f(x)\, p(x)\, \frac{q(x)}{q(x)} dx \\
&= \int_{-\infty}^{\infty} \frac{p(x)}{q(x)} f(x)\, q(x)\, dx \\
&= \mathrm{E}_{q(x)}\left[\frac{p(x)}{q(x)} f(x)\right]
\end{aligned}$$

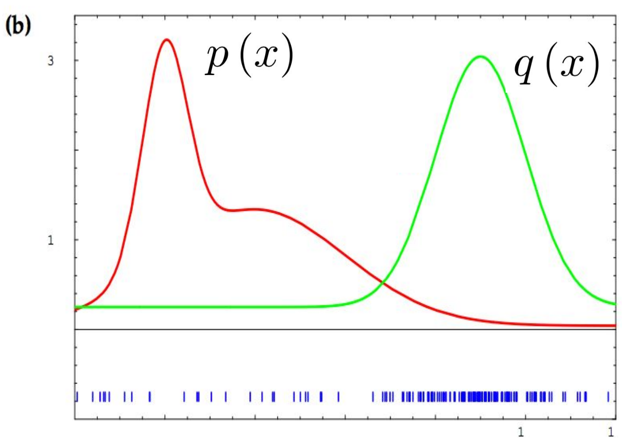


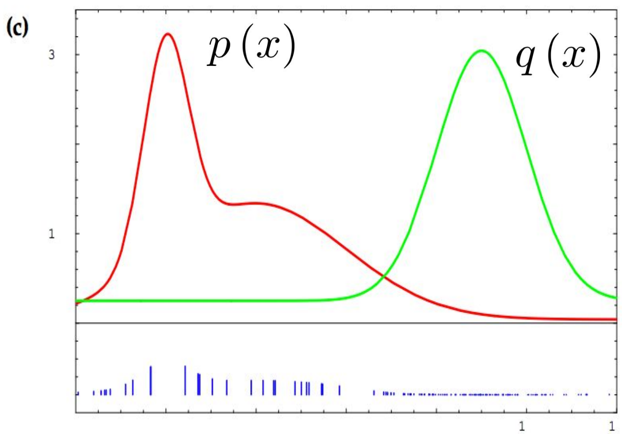


Importance Sampling

$$
\begin{aligned}
\mathrm{E}_{p(x)}\left[f\left(x\right)\right] &= \int_{-\infty}^{\infty} f\left(x\right)p\left(x\right)dx \\
&= \int_{-\infty}^{\infty} f\left(x\right)p\left(x\right)\frac{q\left(x\right)}{q\left(x\right)}dx \\
&= \int_{-\infty}^{\infty} \frac{p\left(x\right)}{q\left(x\right)}f\left(x\right)q\left(x\right)dx \\
&= \mathrm{E}_{q(x)}\left[\frac{p\left(x\right)}{q\left(x\right)}f\left(x\right)\right]
\end{aligned}
$$



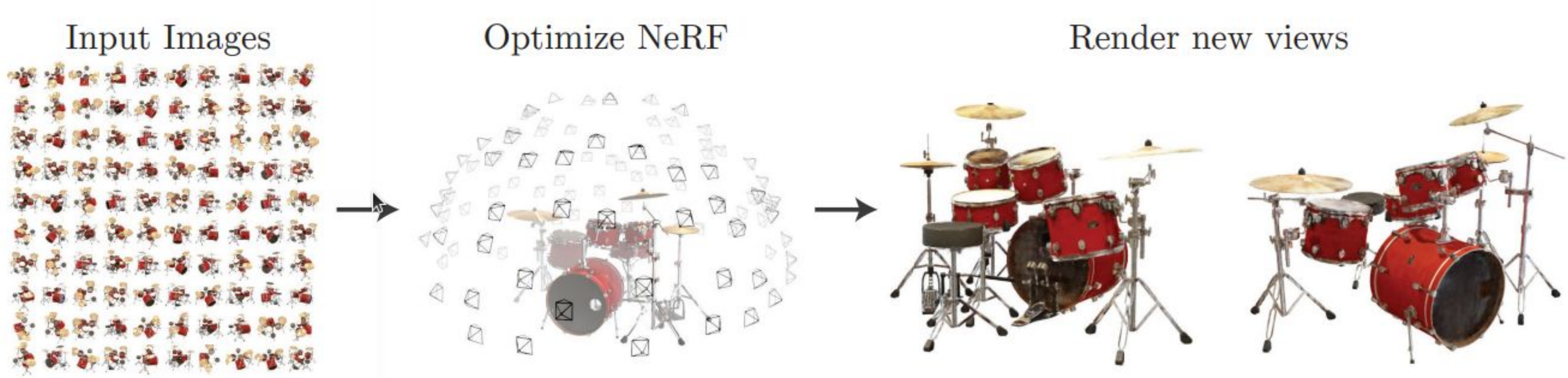




Sampling Application: Neural Rendering

Inverse transform sampling:

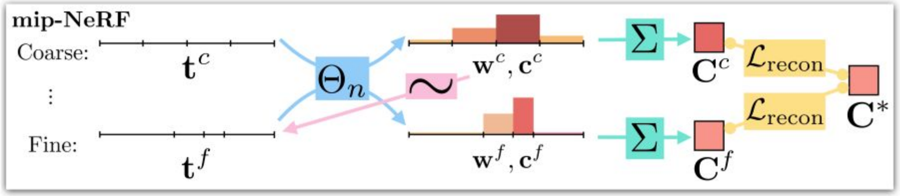
- NeRF: Representing scenes as neural radiance fields for view synthesis [Mildenhall et al., 2021]
- Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields [Barron et al., 2021]

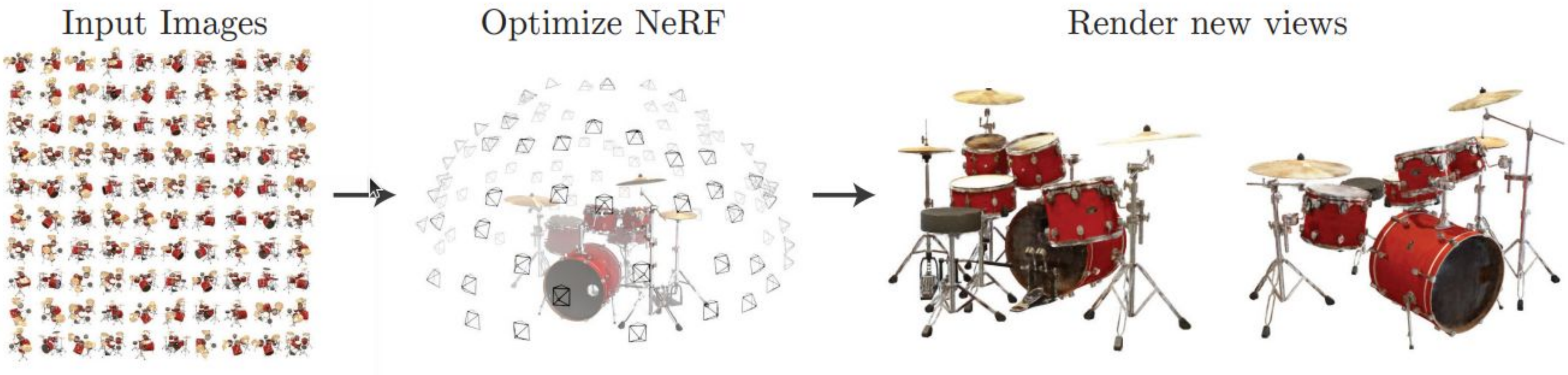


Sampling Application: Neural Rendering

Inverse transform sampling:

- NeRF: Representing scenes as neural radiance fields for view synthesis [Mildenhall et al., 2021]
- Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields [Barron et al., 2021]

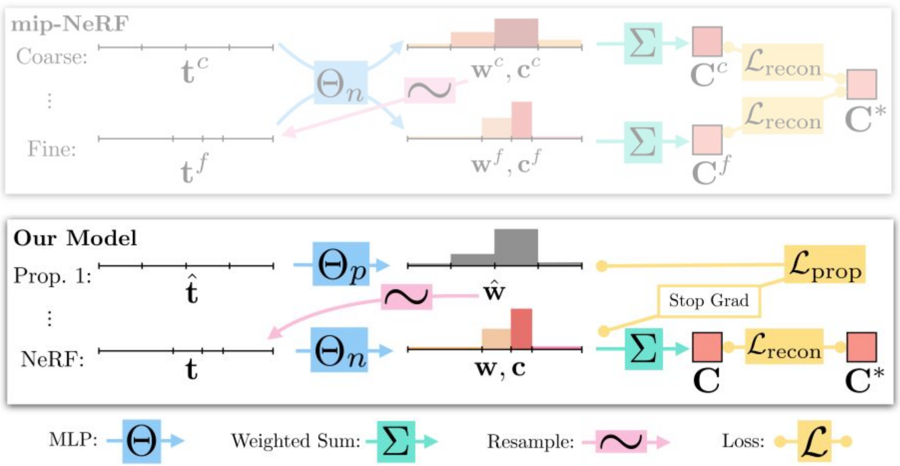




Sampling Application: Neural Rendering

Inverse transform sampling:

- NeRF: Representing scenes as neural radiance fields for view synthesis [Mildenhall et al., 2021]
- Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields [Barron et al., 2021]



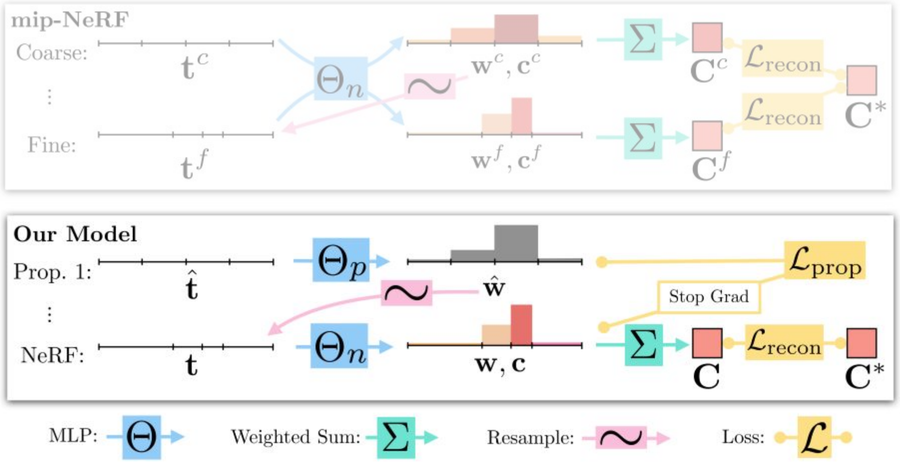
Proposal Network:

- Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields [Barron et al., 2022]
- Zip-NeRF: Anti-aliased grid-based neural radiance fields [Barron et al., 2022]

Sampling Application: Neural Rendering

Inverse transform sampling:

- NeRF: Representing scenes as neural radiance fields for view synthesis [Mildenhall et al., 2021]
- Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields [Barron et al., 2021]



Proposal Network:

- Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields [Barron et al., 2022]
- Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields [Barron et al., 2022]

Particle Filter Derivation

Bayes' Rule

$$p(x|y,e) = \frac{p(y|x,e)p(x|e)}{p(y|e)}$$

$$bel(x_{0:t}) \quad = \quad p(x_{0:t} \mid u_{1:t}, z_{1:t})$$

$$\begin{aligned}
& p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\
& \quad \overset{\text{Bayes}}{=} \quad \eta\, p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& \quad \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& \quad = \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t}) \\
& \quad \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})
\end{aligned}$$

Particle Filter Derivation

Bayes' Rule

$$p(x|y,e) = \frac{p(y|x,e)p(x|e)}{p(y|e)}$$

$$bel(x_{0:t}) = p(x_{0:t} \mid u_{1:t}, z_{1:t})$$

$$\begin{aligned}
&p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\
&\overset{\text{Bayes}}{=} \eta\, p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
&\overset{\text{Markov}}{=} \eta\, p(z_t \mid x_t)\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
&= \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t}) \\
&\overset{\text{Markov}}{=} \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})
\end{aligned}$$

Particle Filter Derivation

Bayes' Rule

$$p(x|y,e) = \frac{p(y|x,e)p(x|e)}{p(y|e)}$$

$$bel(x_{0:t}) \quad = \quad p(x_{0:t} \mid u_{1:t}, z_{1:t})$$

$$\begin{aligned}
& p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\
& \overset{\text{Bayes}}{=} \quad \eta\, p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& = \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t}) \\
& \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})
\end{aligned}$$

Particle Filter Derivation

Bayes' Rule

$$p(x|y,e) = \frac{p(y|x,e)p(x|e)}{p(y|e)}$$

$$bel(x_{0:t}) = p(x_{0:t} \mid u_{1:t}, z_{1:t})$$

$$\begin{aligned}
& p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\
& \stackrel{\text{Bayes}}{=} \eta\, p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& \stackrel{\text{Markov}}{=} \eta\, p(z_t \mid x_t)\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& = \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t}) \\
& \stackrel{\text{Markov}}{=} \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})
\end{aligned}$$

$$P(A,B) = P(A \mid B)\, P(B)$$

$$P(A,B \mid C) = P(A \mid B,C)\, P(B \mid C)$$

Particle Filter Derivation

Bayes' Rule

$$p(x|y,e) = \frac{p(y|x,e)p(x|e)}{p(y|e)}$$

$$bel(x_{0:t}) \quad = \quad p(x_{0:t} \mid u_{1:t}, z_{1:t})$$

$$\begin{aligned}
&p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\
&\quad \overset{\text{Bayes}}{=} \quad \eta\, p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
&\quad \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
&\quad = \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t}) \\
&\quad \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})
\end{aligned}$$

Particle Filter Derivation

Bayes' Rule

$$p(x|y,e) = \frac{p(y|x,e)p(x|e)}{p(y|e)}$$

$$bel(x_{0:t}) \quad = \quad p(x_{0:t} \mid u_{1:t}, z_{1:t})$$

$$\begin{aligned}
& p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\
& \overset{\text{Bayes}}{=} \quad \eta\, p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
& = \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t}) \\
& \overset{\text{Markov}}{=} \quad \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})
\end{aligned}$$

$bel(x_{0:t-1})$ Recursion!

Particle Filter Derivation

$$\boxed{p(x_{0:t} \mid z_{1:t}, u_{1:t}) \;=\; \eta \, p(z_t \mid x_t) \, p(x_t \mid x_{t-1}, u_t) \, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}$$

- Recall importance weighting

$$w_t^{[m]} \;=\; \frac{\text{target distribution}}{\text{proposal distribution}}$$

Particle Filter Derivation

$$\boxed{p(x_{0:t} \mid z_{1:t}, u_{1:t}) \;=\; \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}$$

- Recall importance weighting

$$w_t^{[m]} \;=\; \frac{\text{target distribution}}{\text{proposal distribution}} \qquad \longleftarrow \quad bel(x_{0:t})$$

Particle Filter Derivation

$$\boxed{p(x_{0:t} \mid z_{1:t}, u_{1:t}) \;=\; \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}$$

- Recall importance weighting

$$w_t^{[m]} \;=\; \frac{\text{target distribution}}{\text{proposal distribution}} \qquad \begin{array}{l} \longleftarrow \; bel(x_{0:t}) \\ \longleftarrow \; p(x_t \mid x_{t-1}, u_t)\; bel(x_{0:t-1}) \end{array}$$

Particle Filter Derivation

$$\boxed{p(x_{0:t} \mid z_{1:t}, u_{1:t}) \;=\; \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}$$

- Recall importance weighting

We don't know this (don't have samples… yet)

$$w_t^{[m]} = \frac{\text{target distribution}}{\text{proposal distribution}} \qquad \begin{array}{l} \leftarrow bel(x_{0:t}) \\ \leftarrow p(x_t \mid x_{t-1}, u_t)\; bel(x_{0:t-1}) \end{array}$$

We do know this (can generate samples)

Particle Filter Derivation

$$\boxed{p(x_{0:t} \mid z_{1:t}, u_{1:t}) \;=\; \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}$$

- Recall importance weighting

$$\begin{aligned} w_t^{[m]} &= \frac{\text{target distribution}}{\text{proposal distribution}} \quad \begin{array}{l} \longleftarrow \; bel(x_{0:t}) \\ \longleftarrow \; p(x_t \mid x_{t-1}, u_t)\; bel(x_{0:t-1}) \end{array} \\ &= \frac{\eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}{p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{0:t-1}, u_{0:t-1})} \\ &= \eta\, p(z_t \mid x_t) \end{aligned}$$

But we know the ratio!

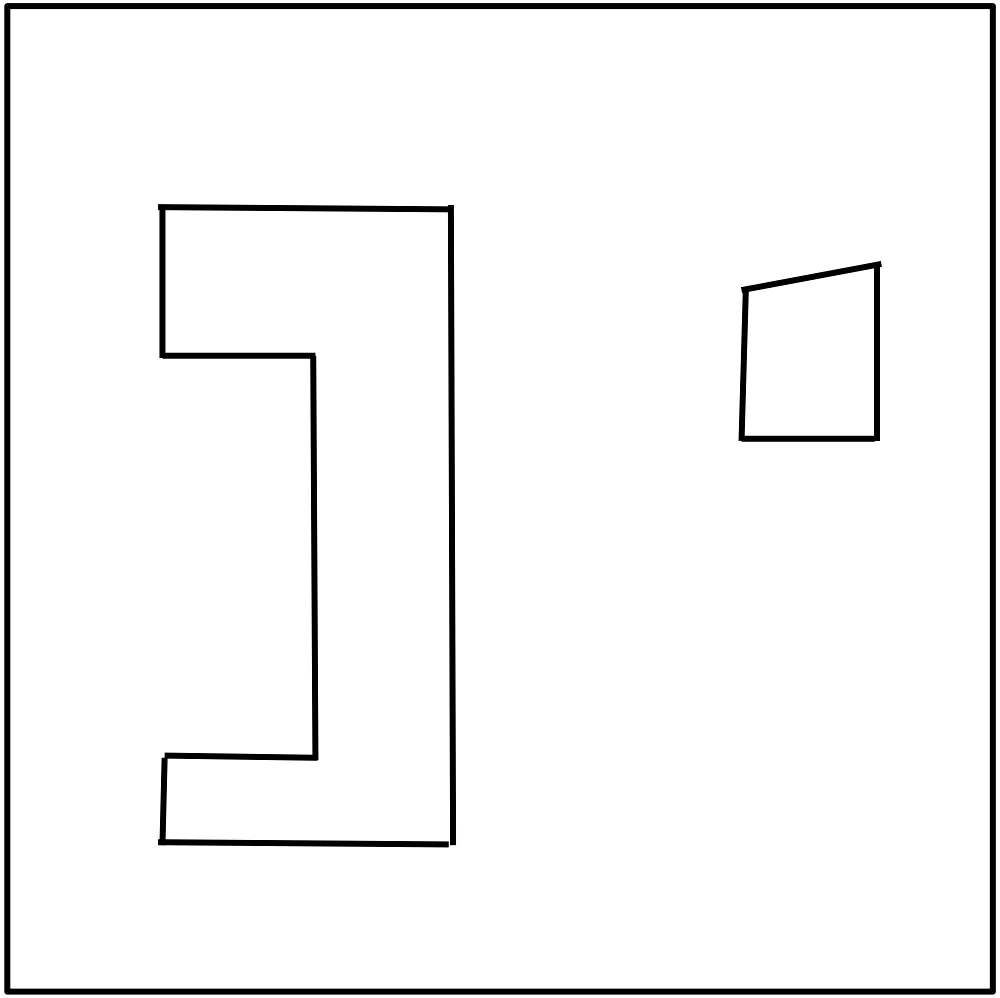
Particle Filter Algorithm

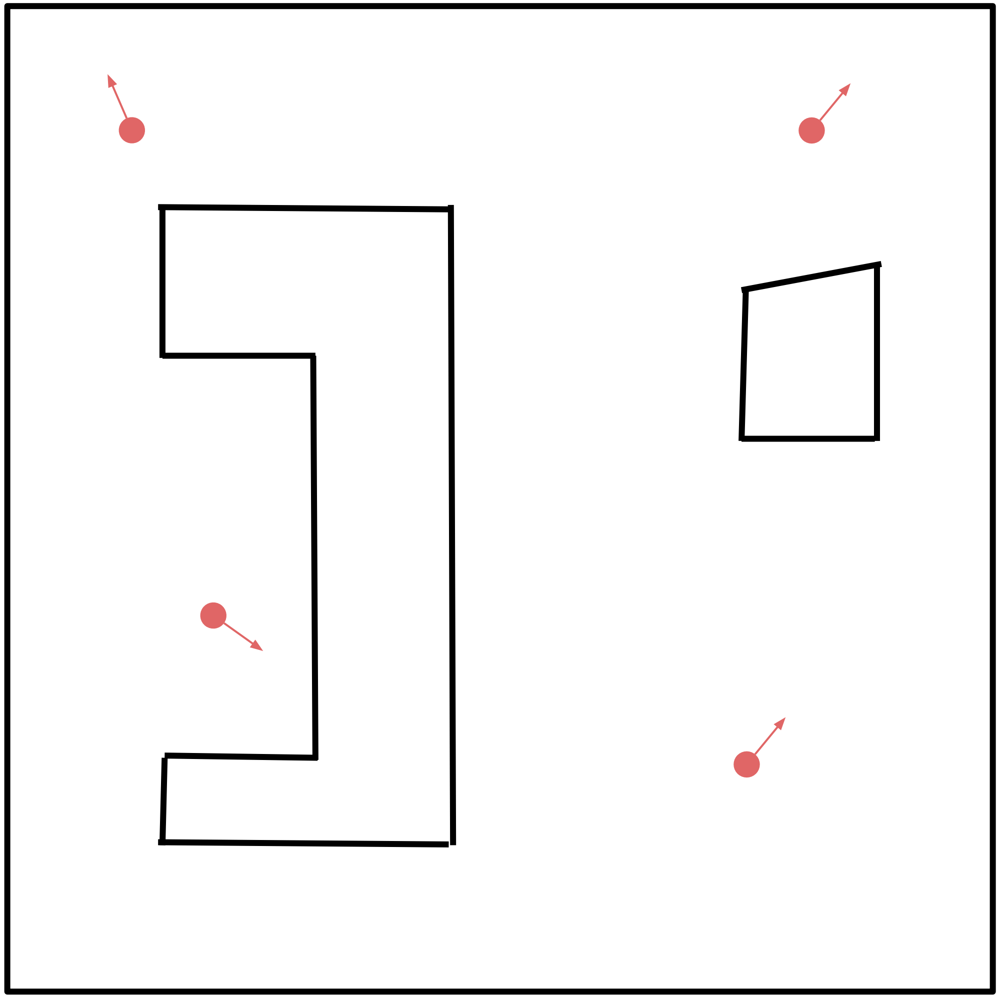
$$p(x_{0:t} \mid z_{1:t}, u_{1:t}) \;=\; \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})$$

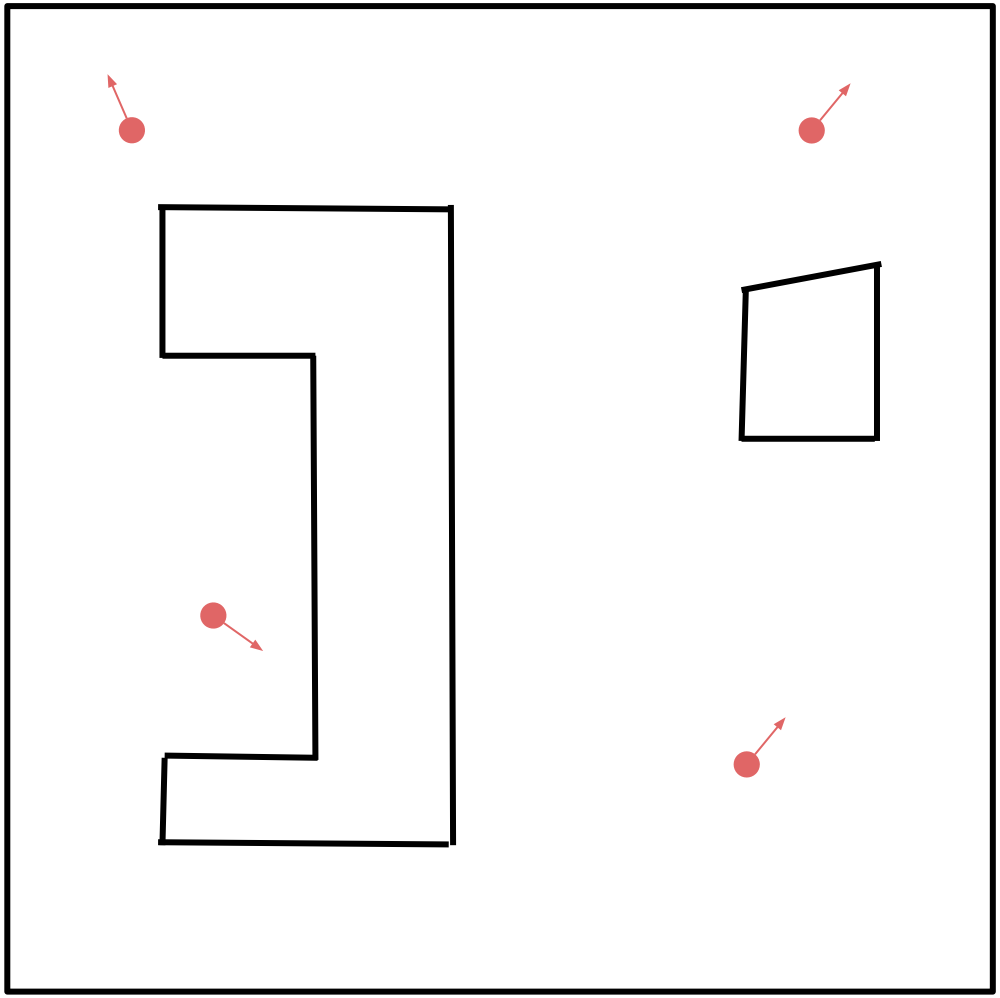
Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):
- $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
- for $m = 1$ to M do
  - sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$ — motion model
  - $w_t^{[m]} = p(z_t \mid x_t^{[m]})$ — sensor model
  - $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
- endfor
- for $m = 1$ to M do
  - draw i with probability $\propto w_t^{[i]}$ — importance sampling
  - add $x_t^{[i]}$ to $\mathcal{X}_t$
- endfor
- return $\mathcal{X}_t$

$\mathcal{X}_{t-1}$ - previous particle set

$\mathcal{X}_t$ - output particle set

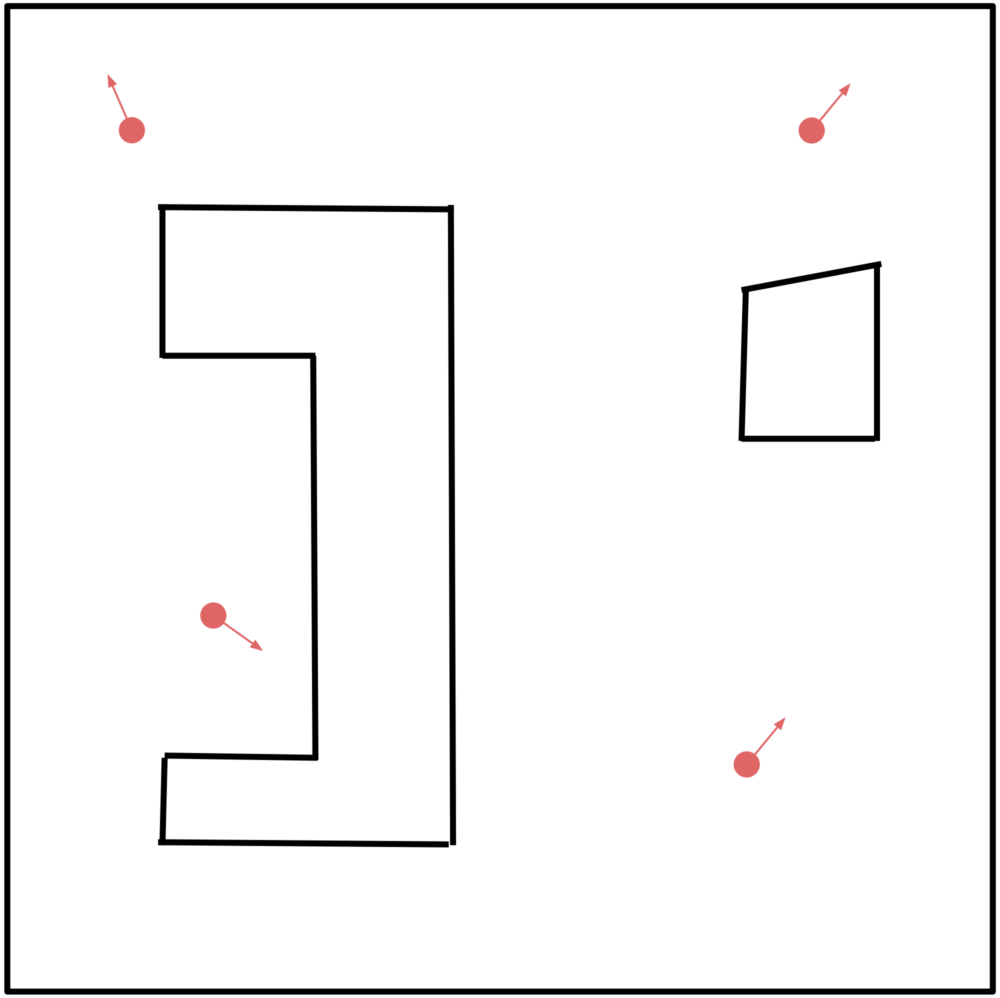






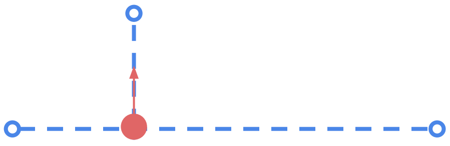
Measurement z

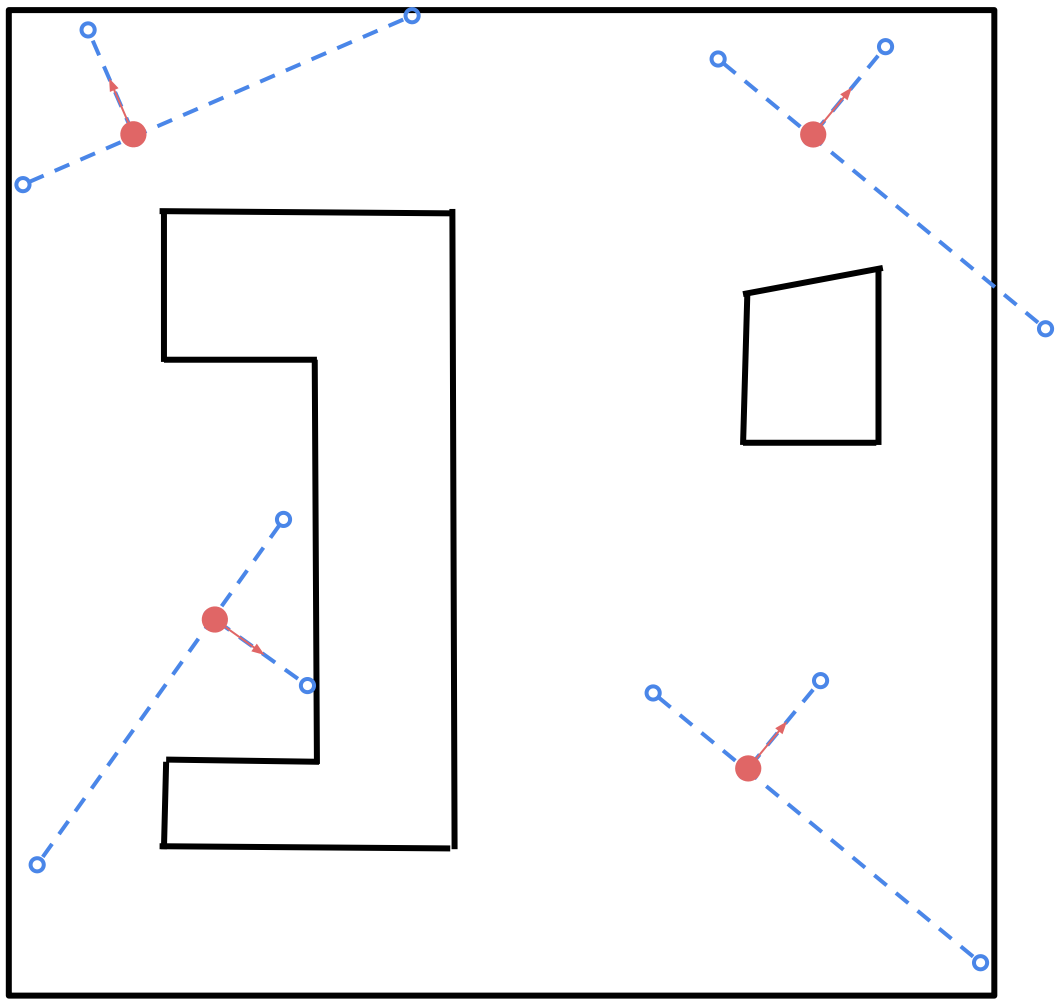
$p(z_t \mid x_t)$



Measurement z

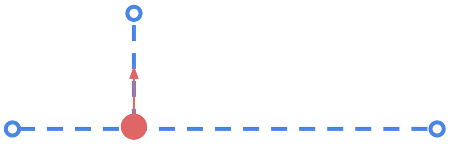
$$p(z_t \mid x_t)$$

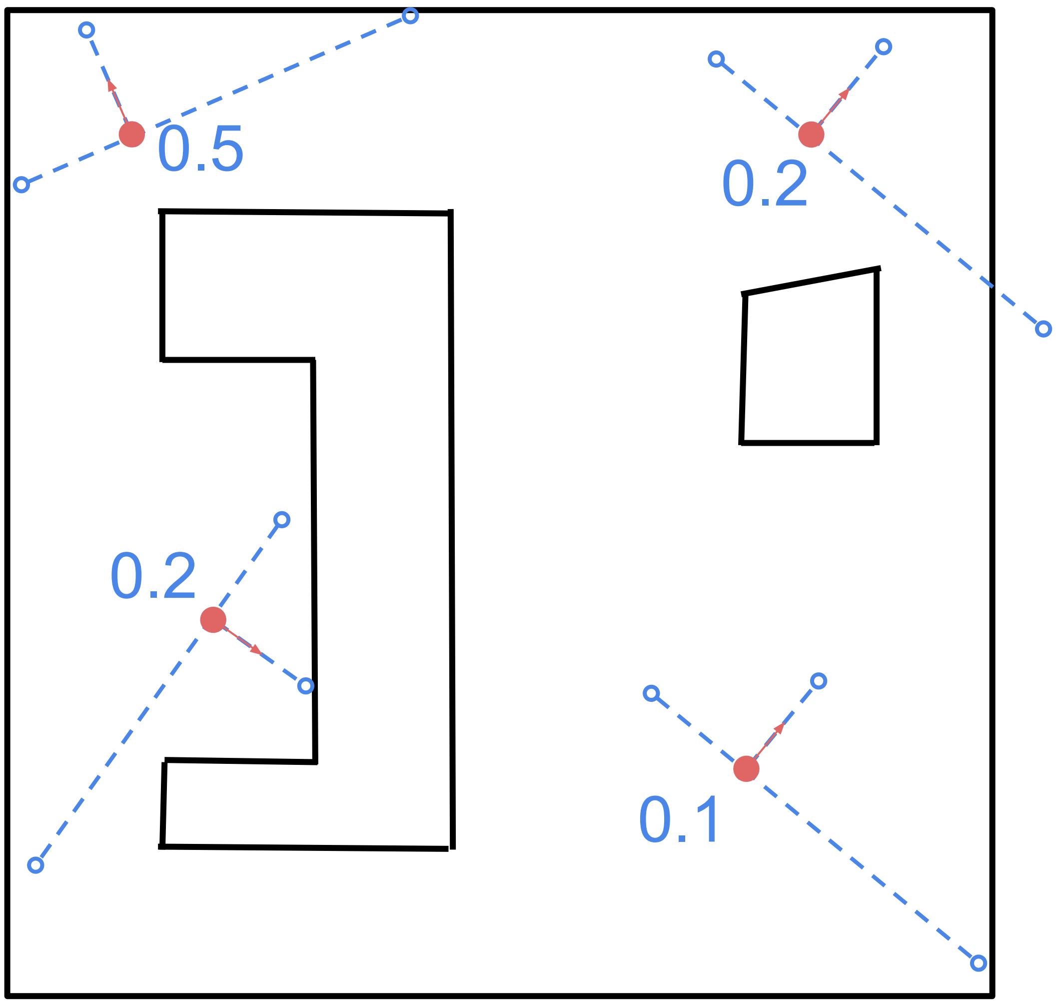




Measurement z

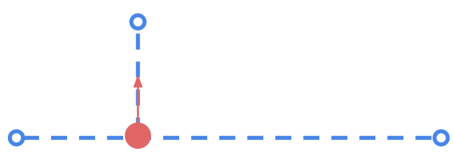
$$p(z_t \mid x_t)$$

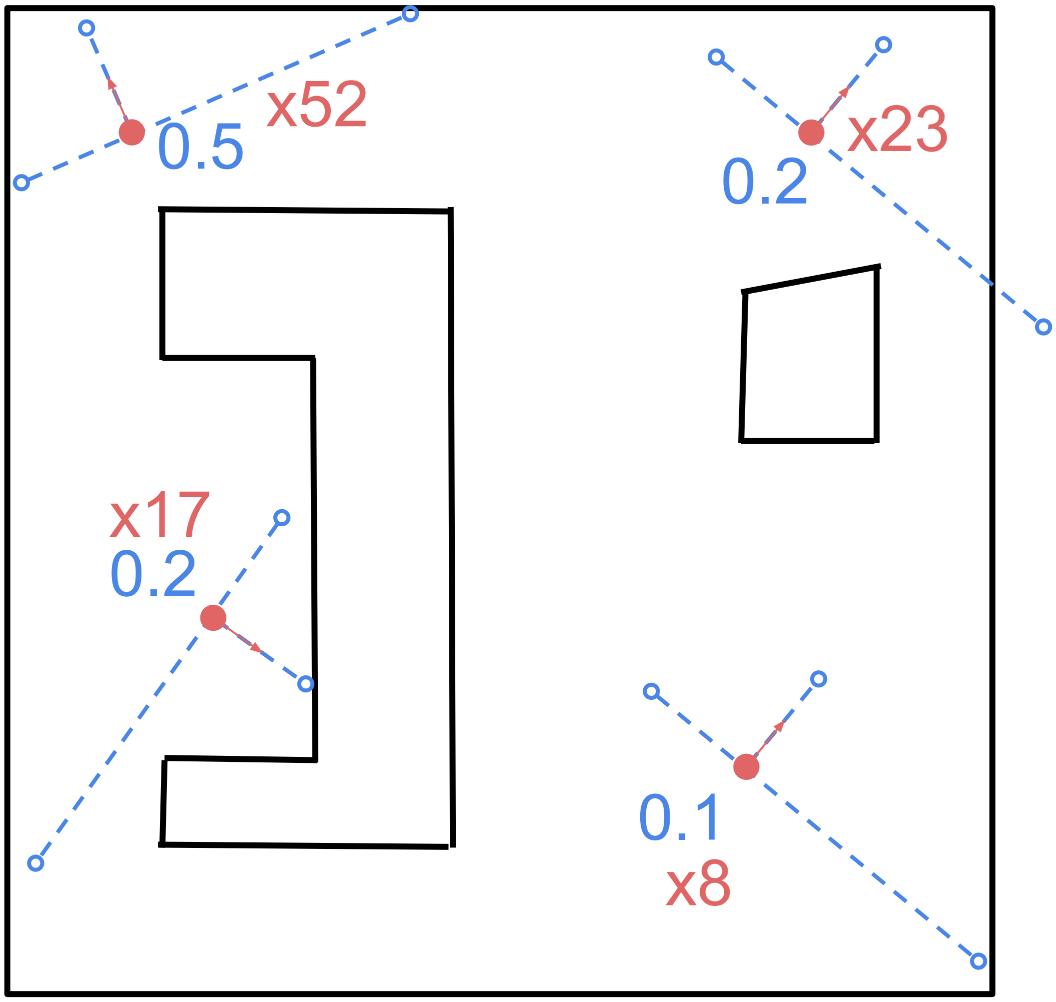




Measurement z

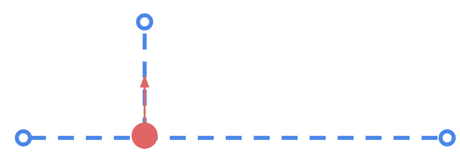
$p(z_t \mid x_t)$

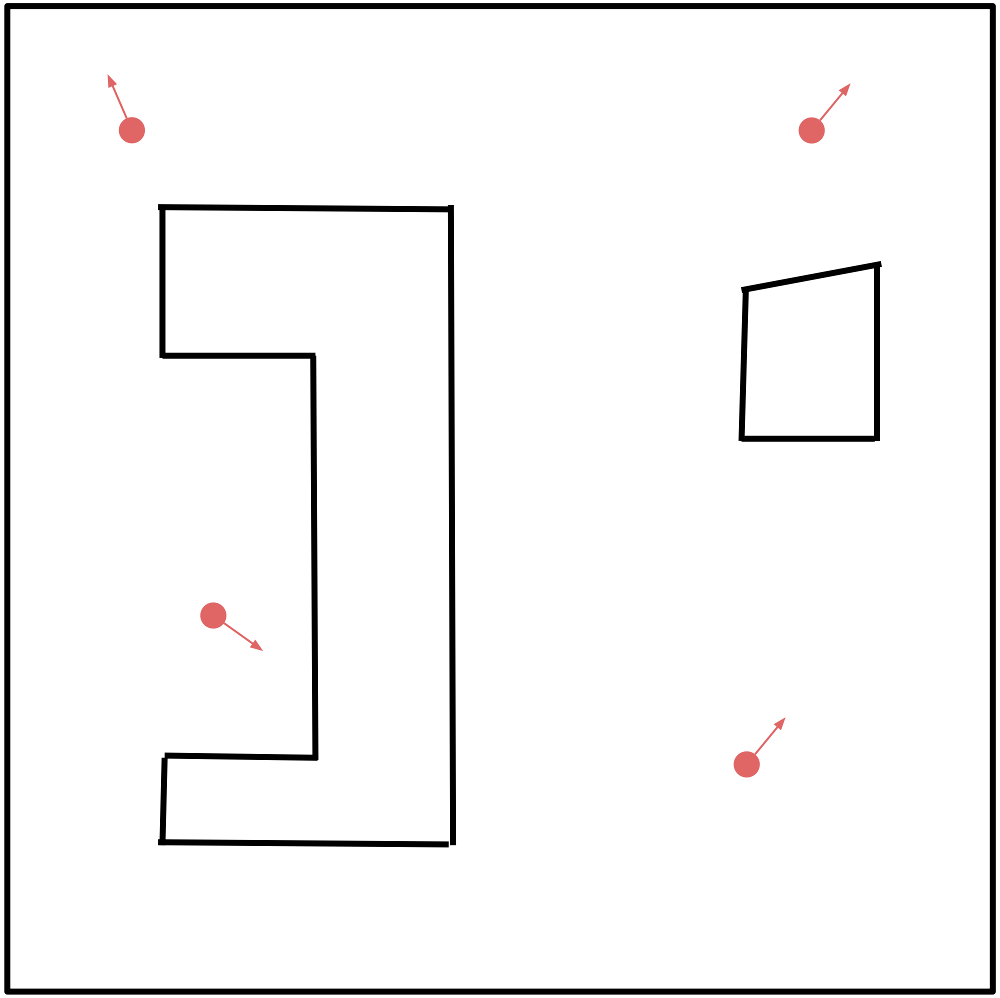




Measurement z

$$p(z_t \mid x_t)$$



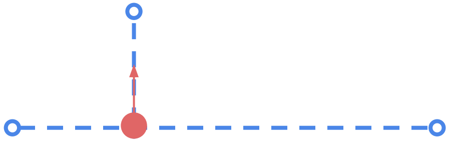


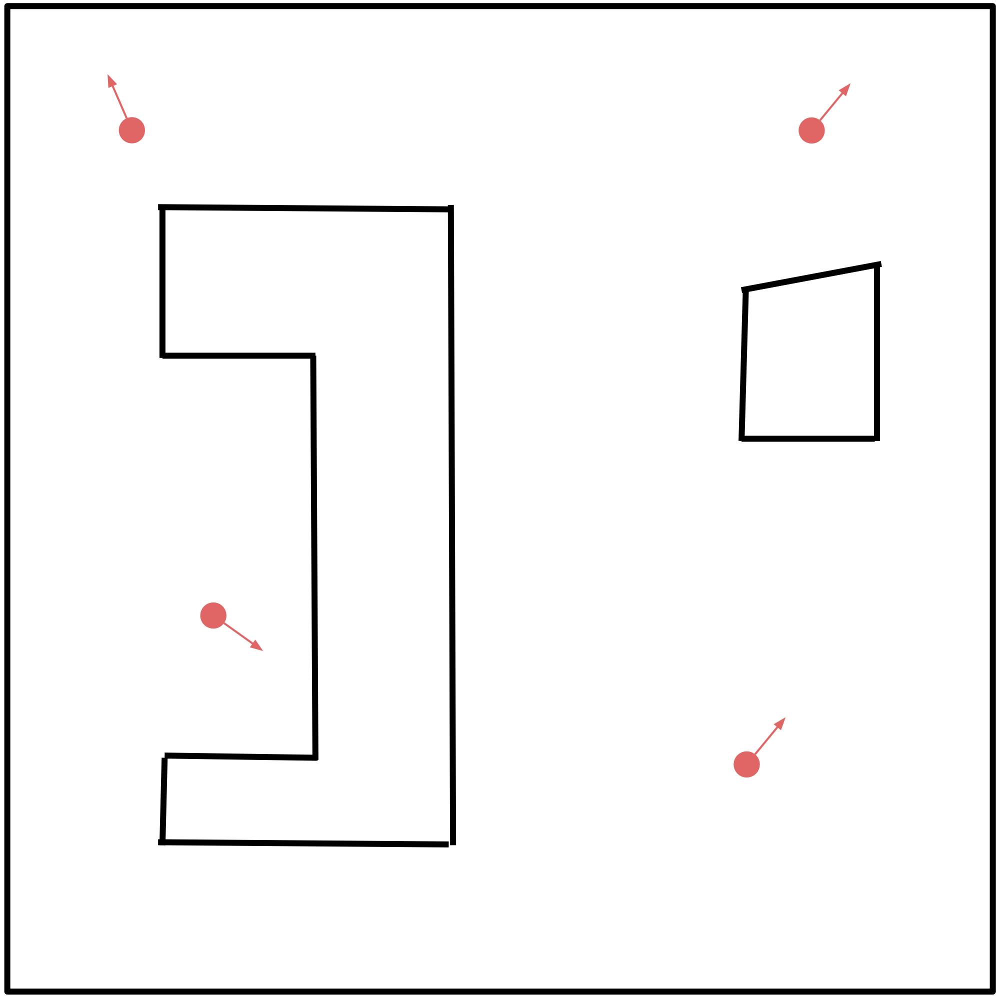
Motion u

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

Measurement z

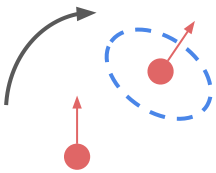
$$p(z_t \mid x_t)$$





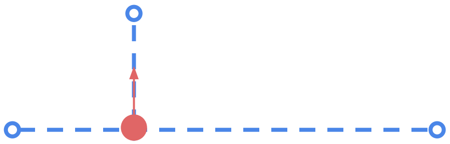
Motion u

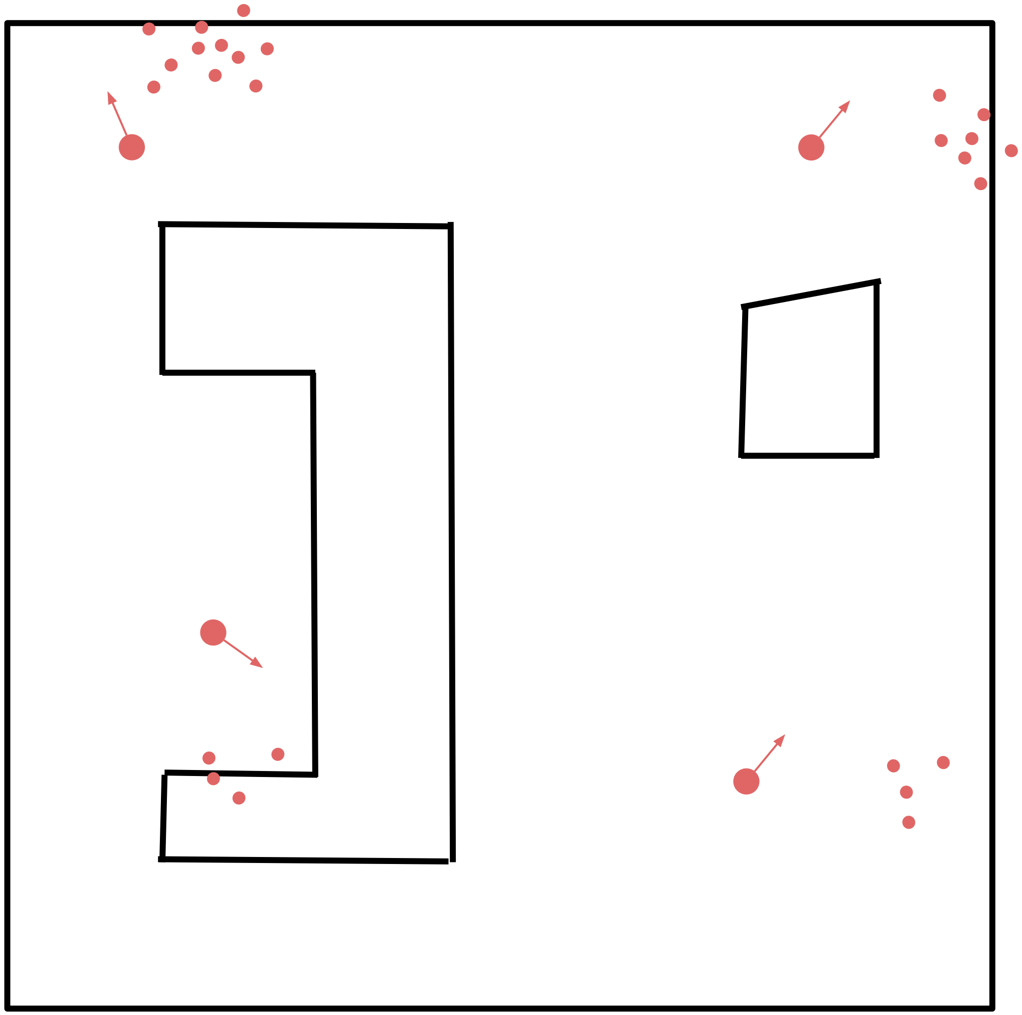
$$p(x_t \mid x_{t-1}, u_t)$$



Measurement z

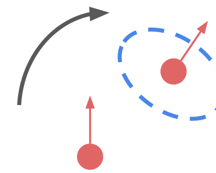
$$p(z_t \mid x_t)$$





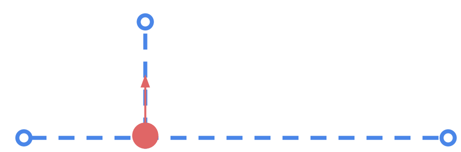
Motion u

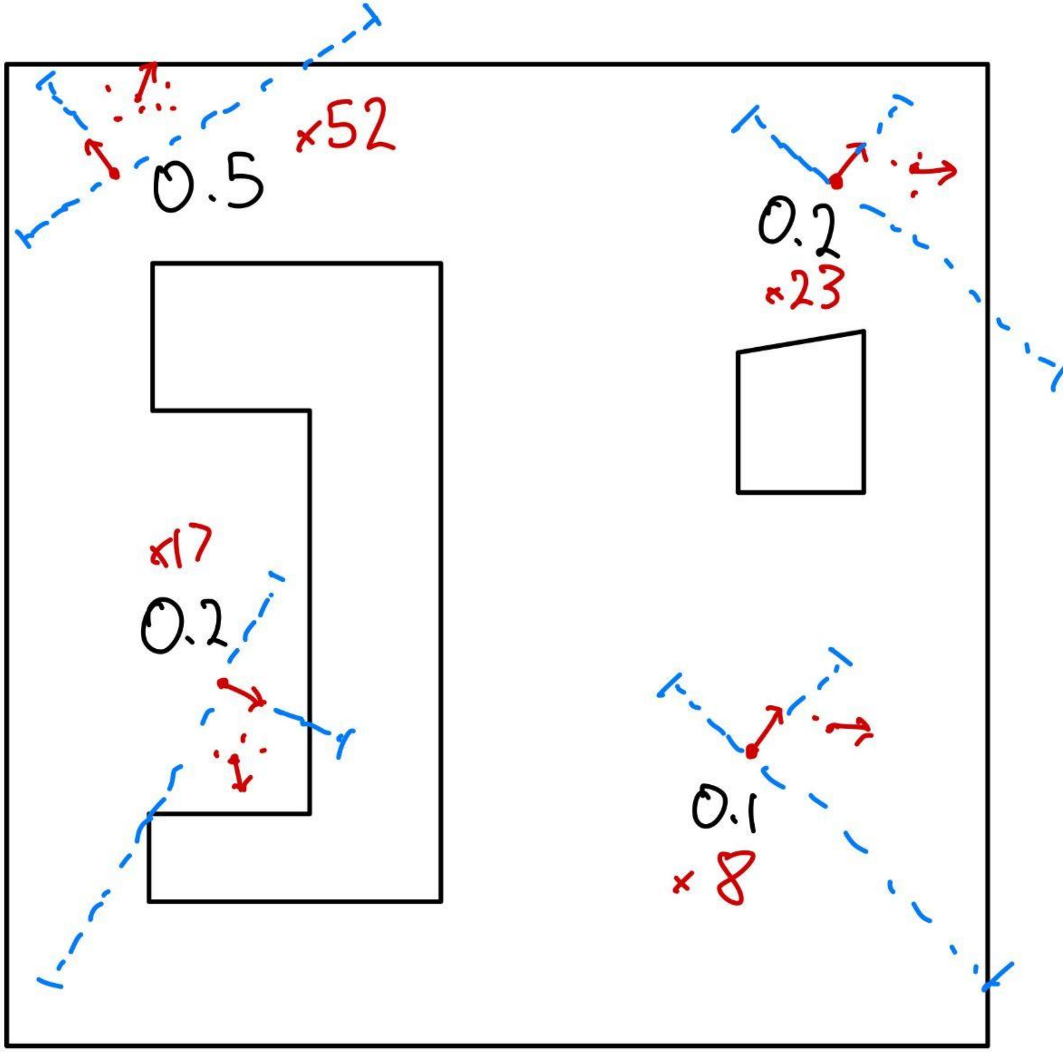
$$p(x_t \mid x_{t-1}, u_t)$$



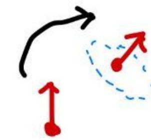
Measurement z

$$p(z_t \mid x_t)$$



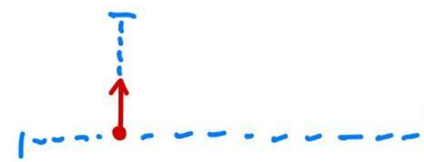


Motion u



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

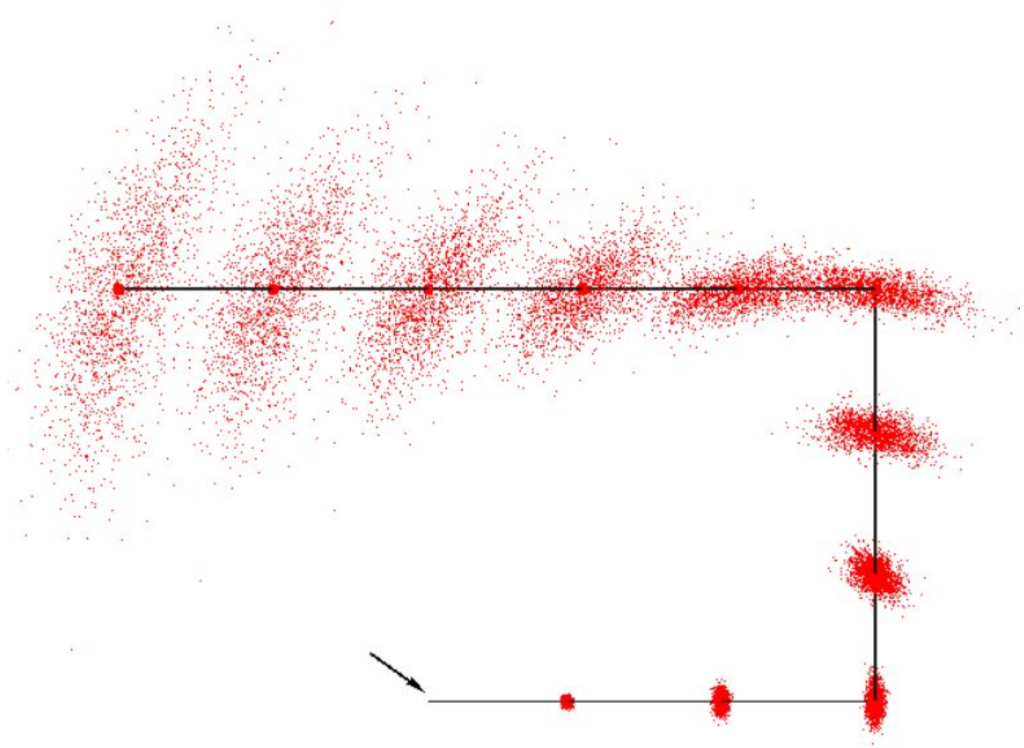
Measurement z



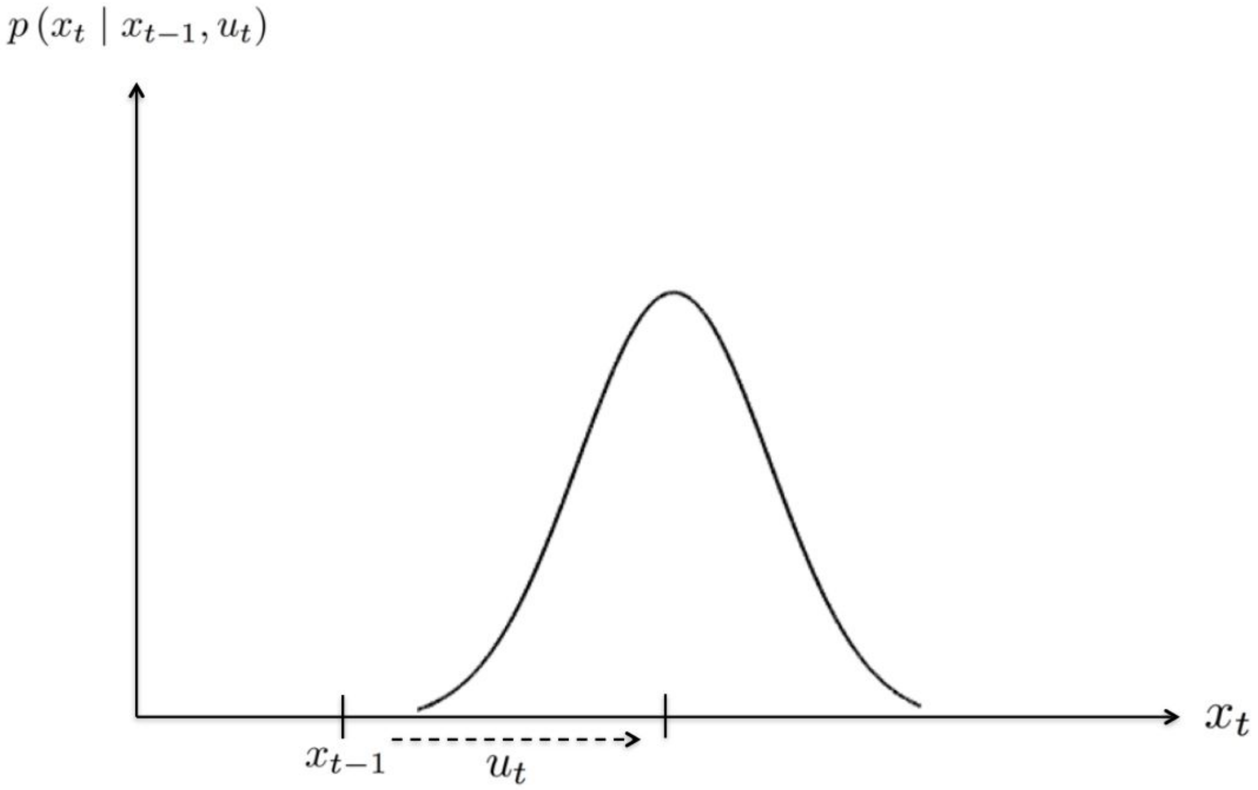
$p(z_t \mid x_t)$

What should the motion model look like?

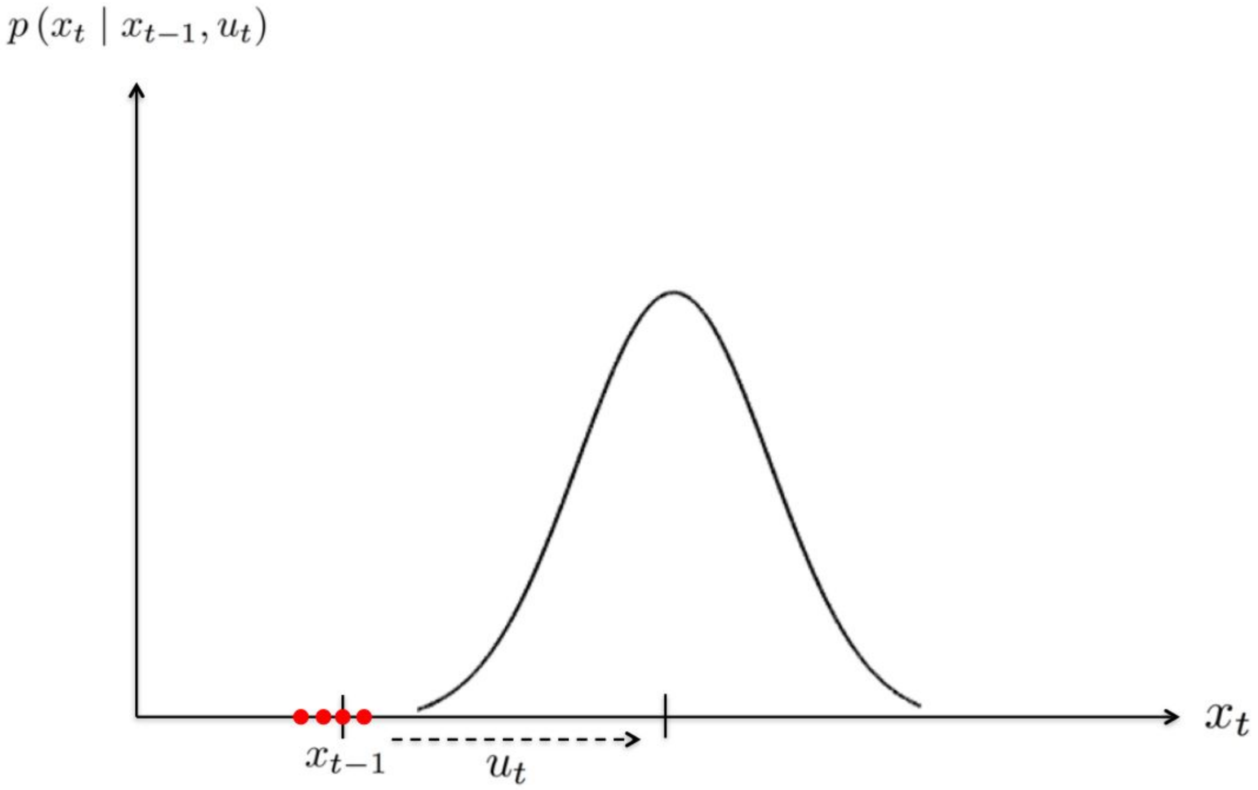
- Needs to be a distribution from which we can draw samples
- Usually use a normal distribution with odometry measurement as mean



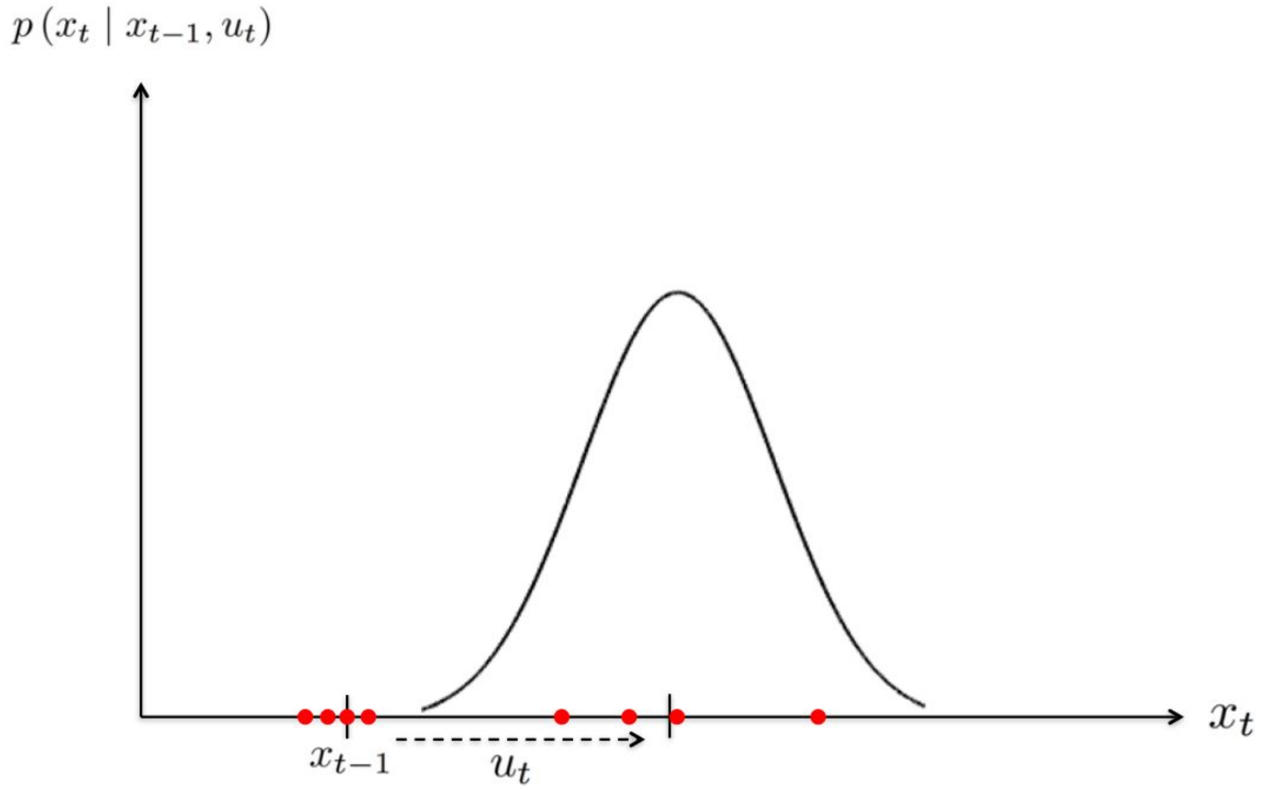
Sampling from the motion model



Sampling from the motion model

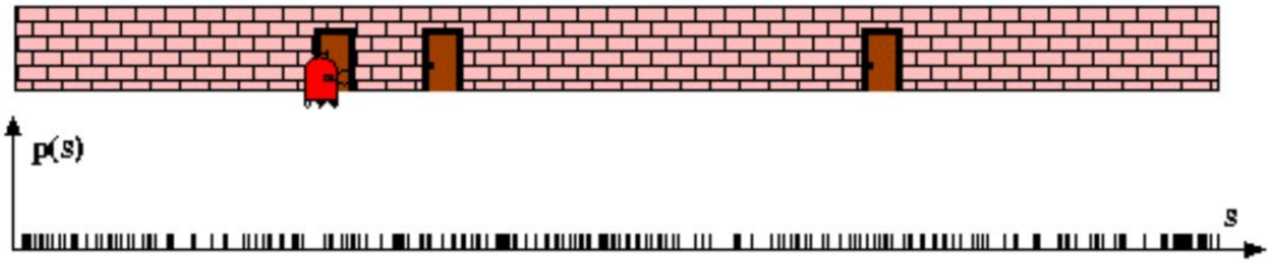


Sampling from the motion model



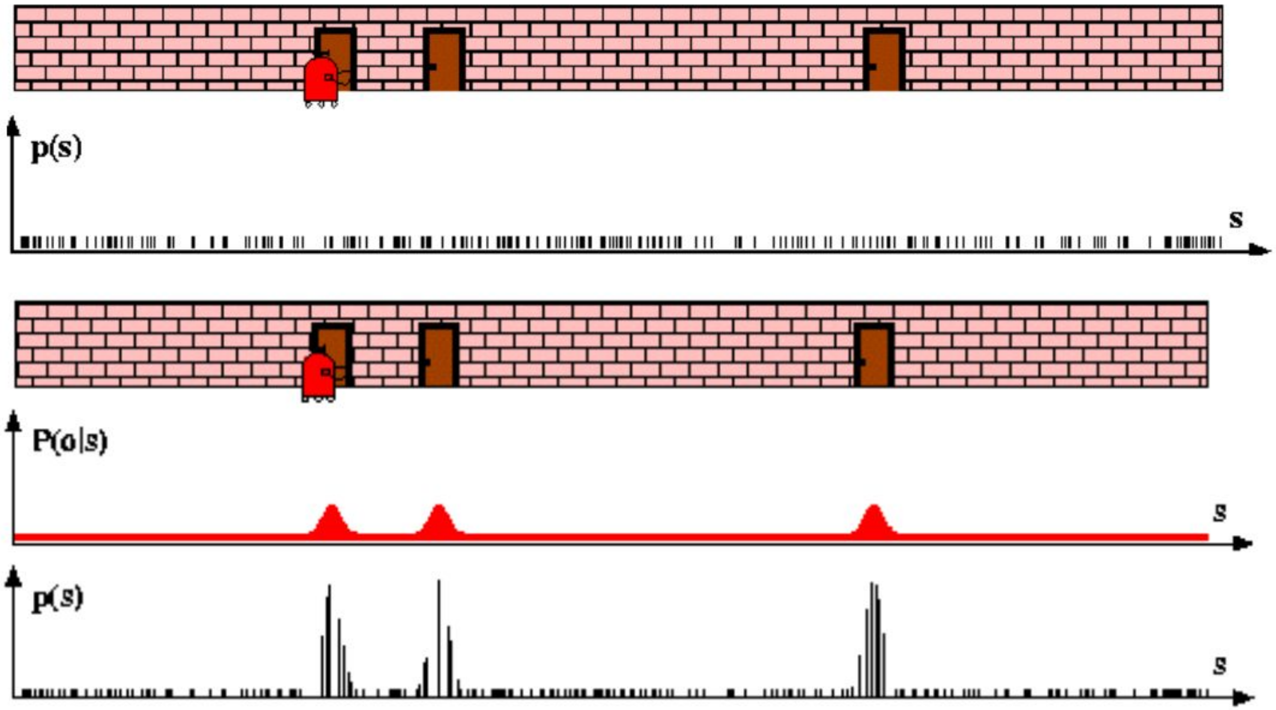
1D Example

- State uniformly distributed initially



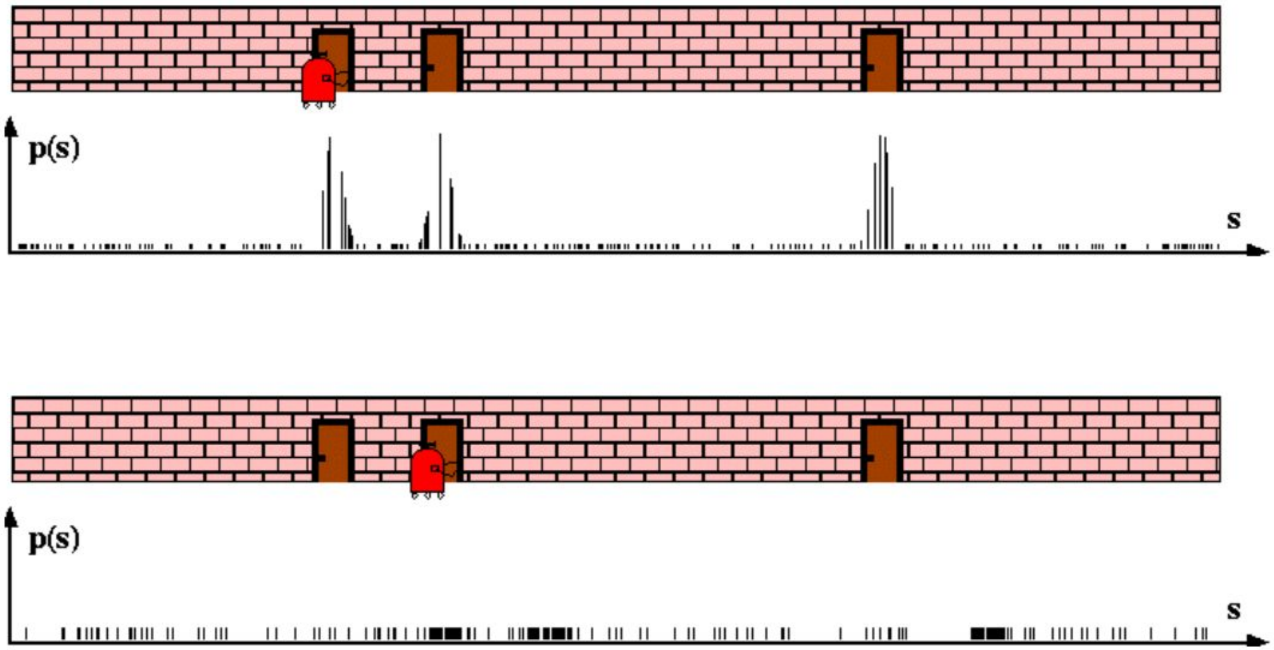
1D Example

- Sensor model, update weights



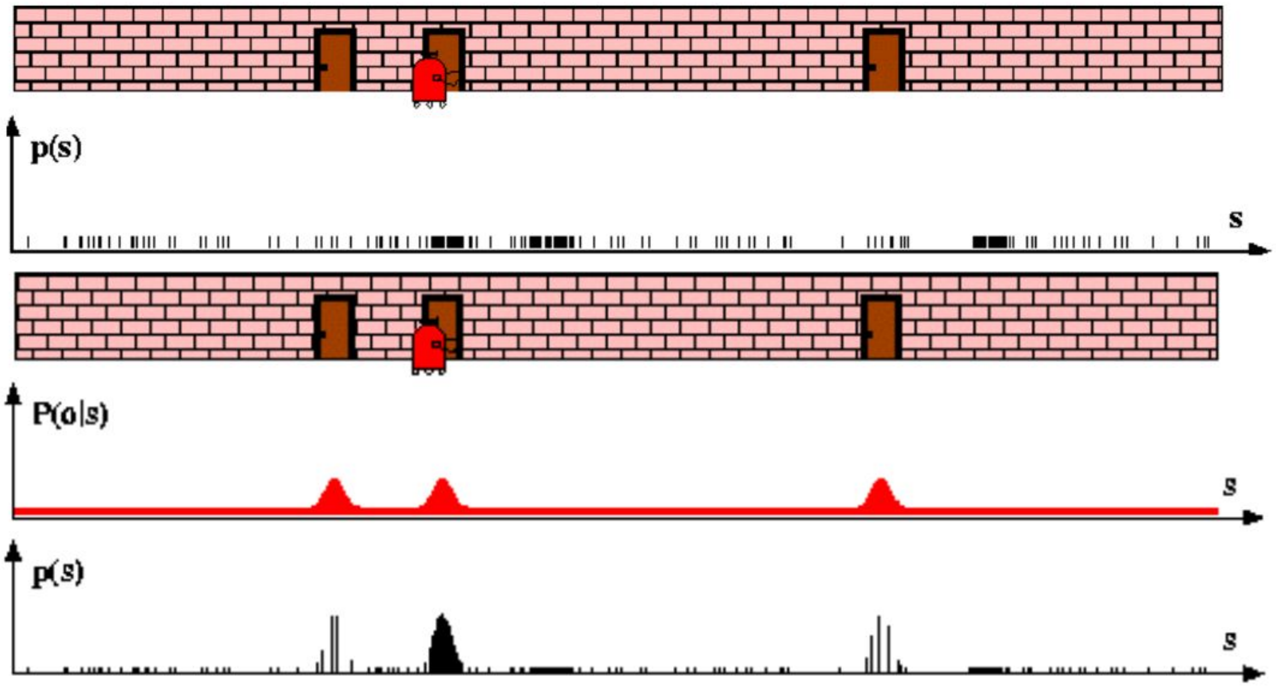
1D Example

- Motion model, resample particles



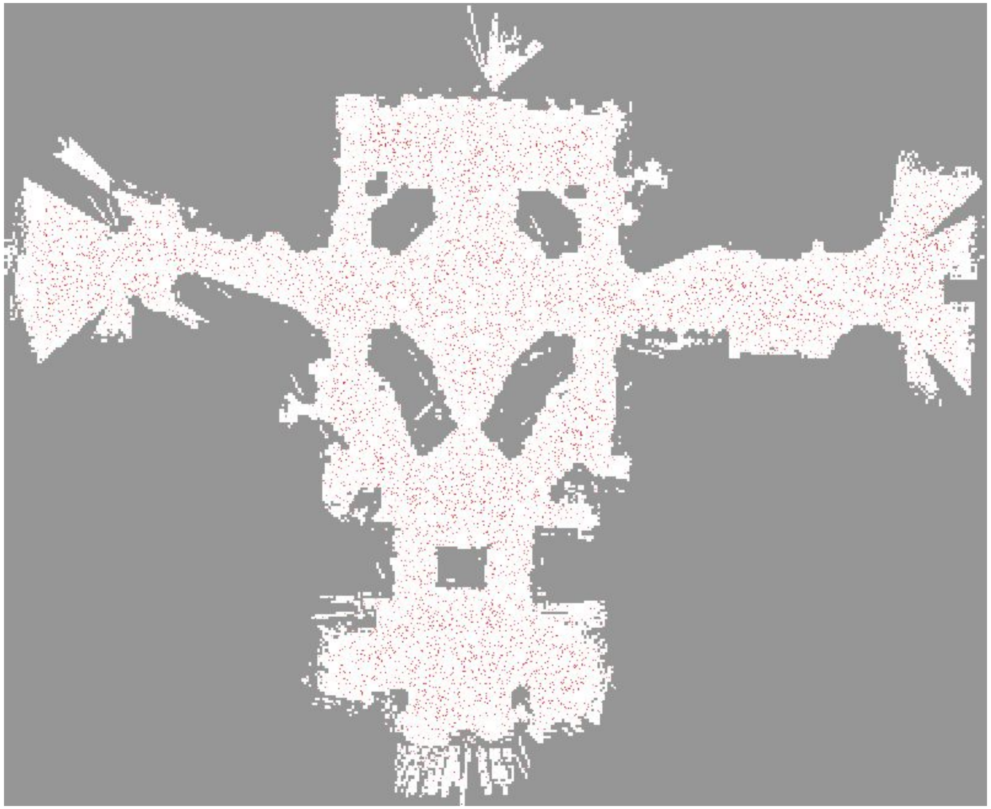
1D Example

- Sensor model, update weights



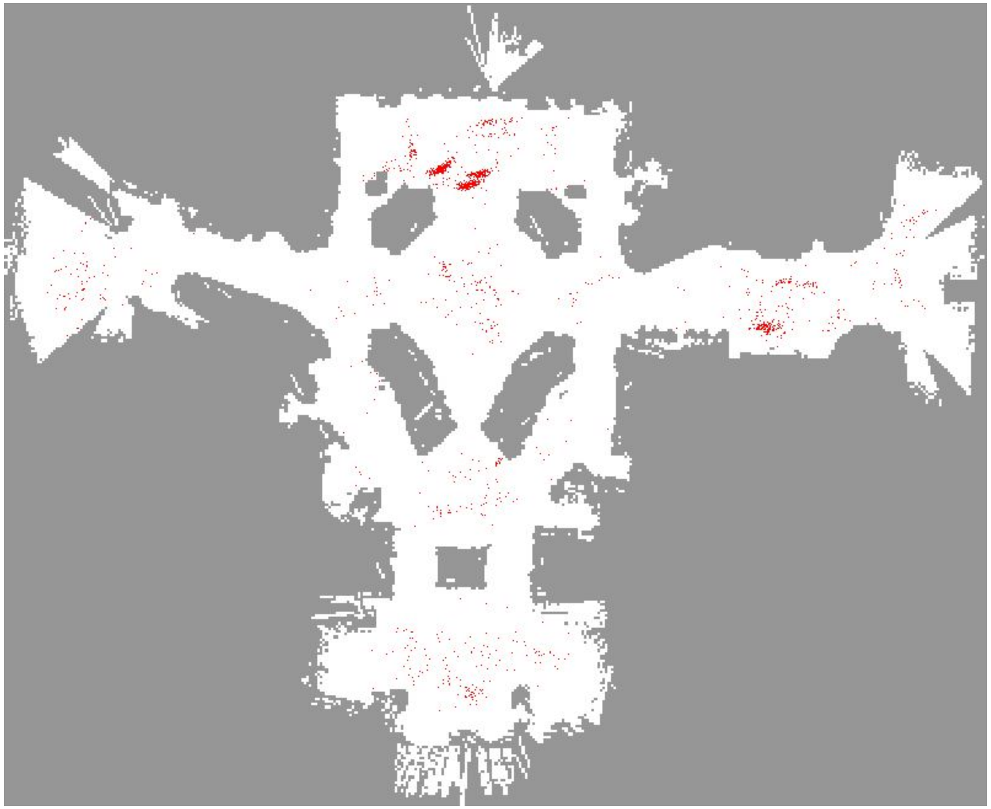
2D Example

- Initially particles uniformly distributed



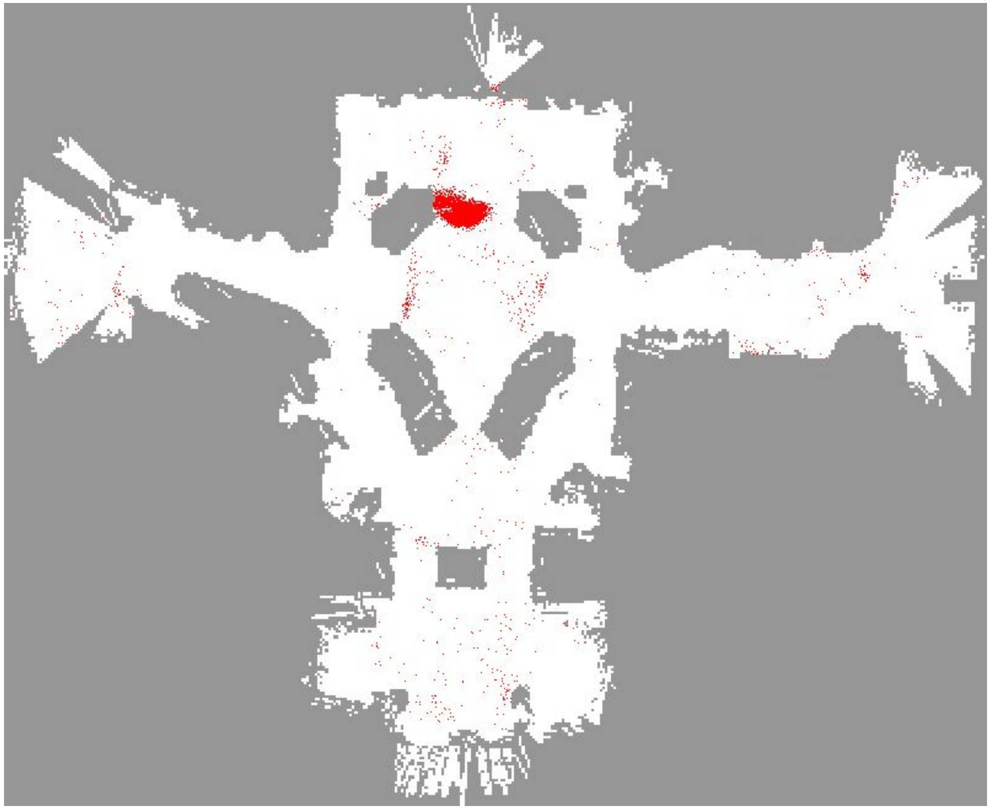
2D Example

- Sensor model, update weights, resample



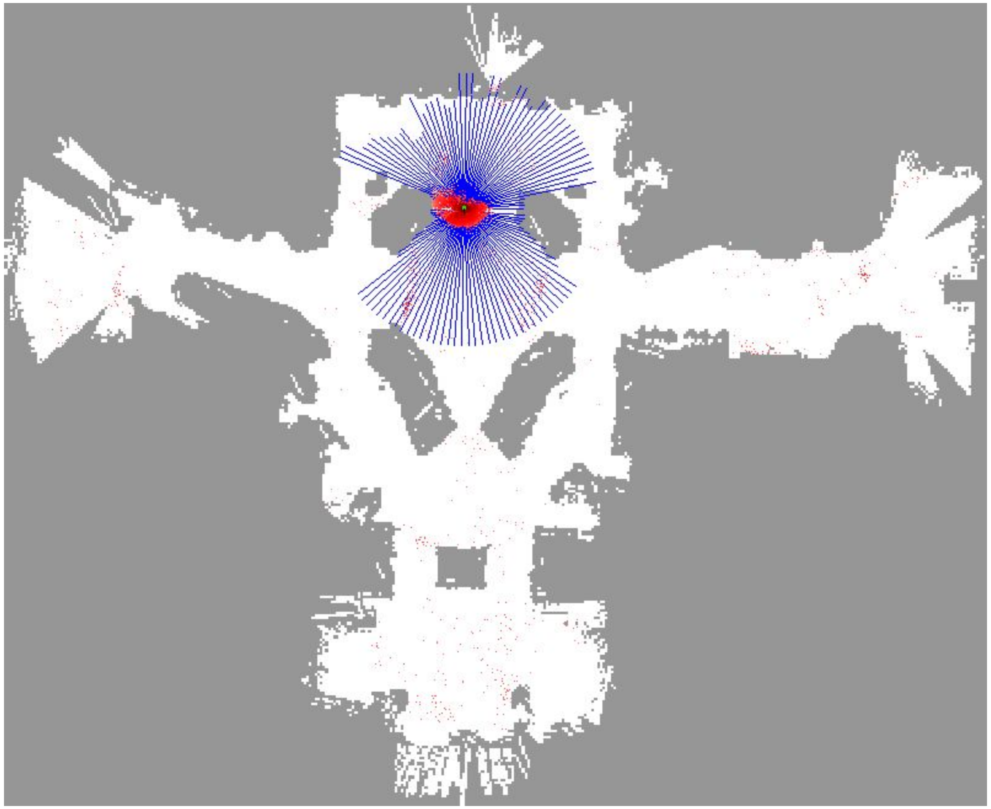
2D Example

- Propagate with motion model



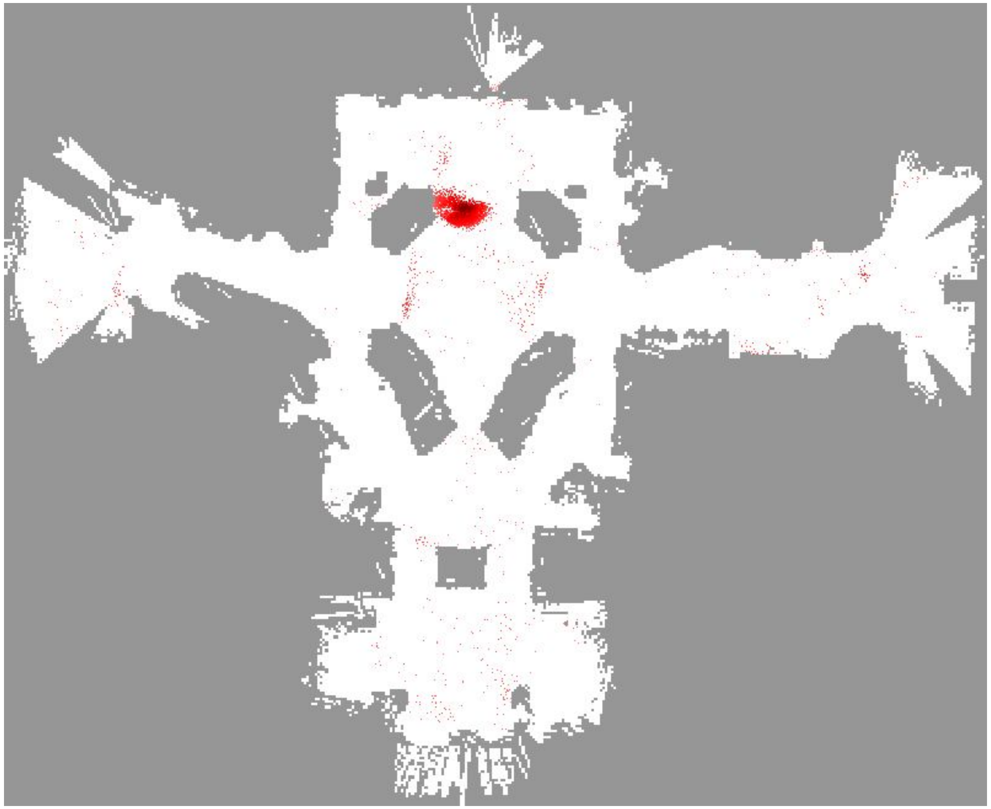
2D Example

- Get measurements



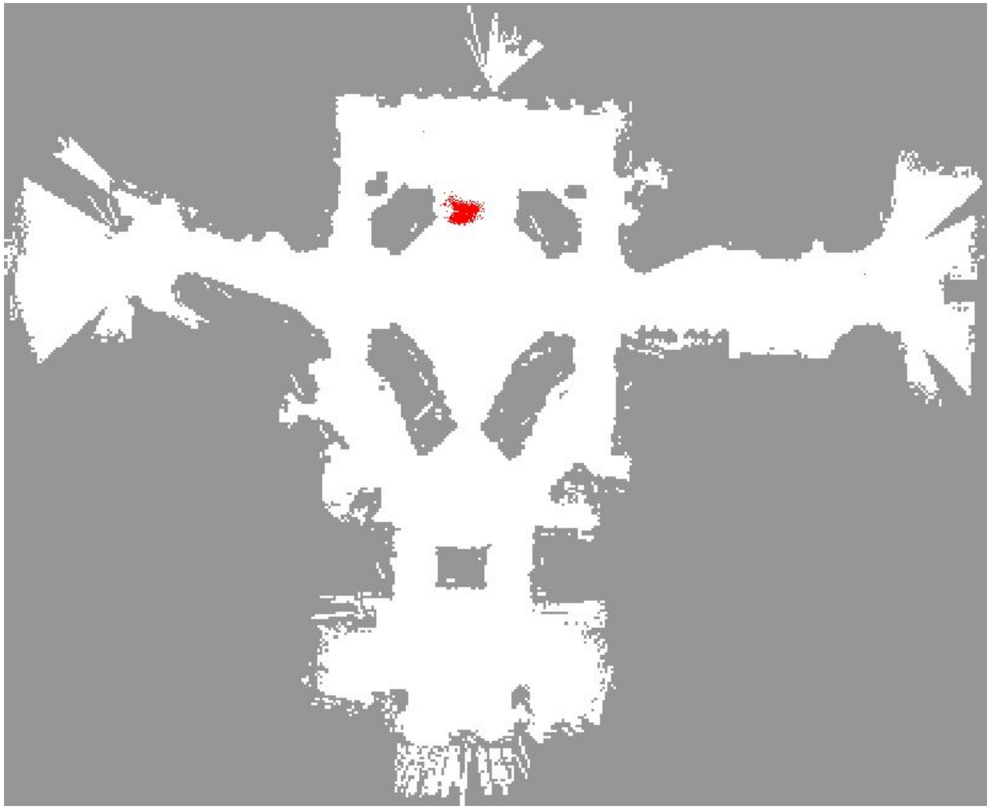
2D Example

- Get measurements



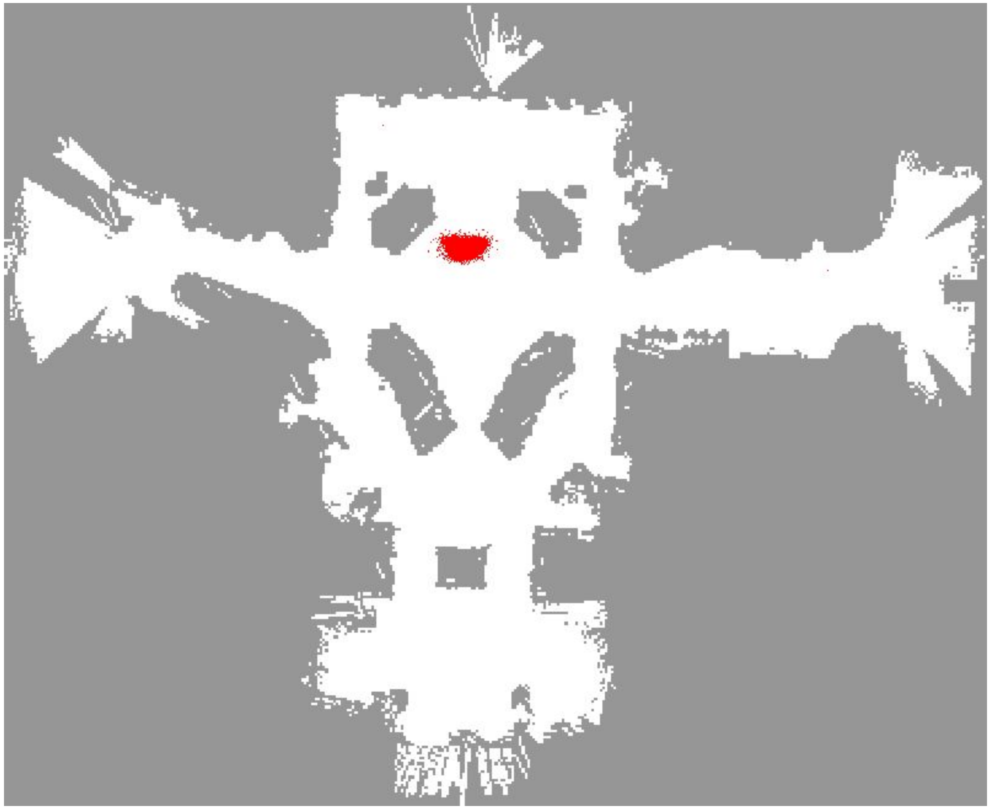
2D Example

- Resample



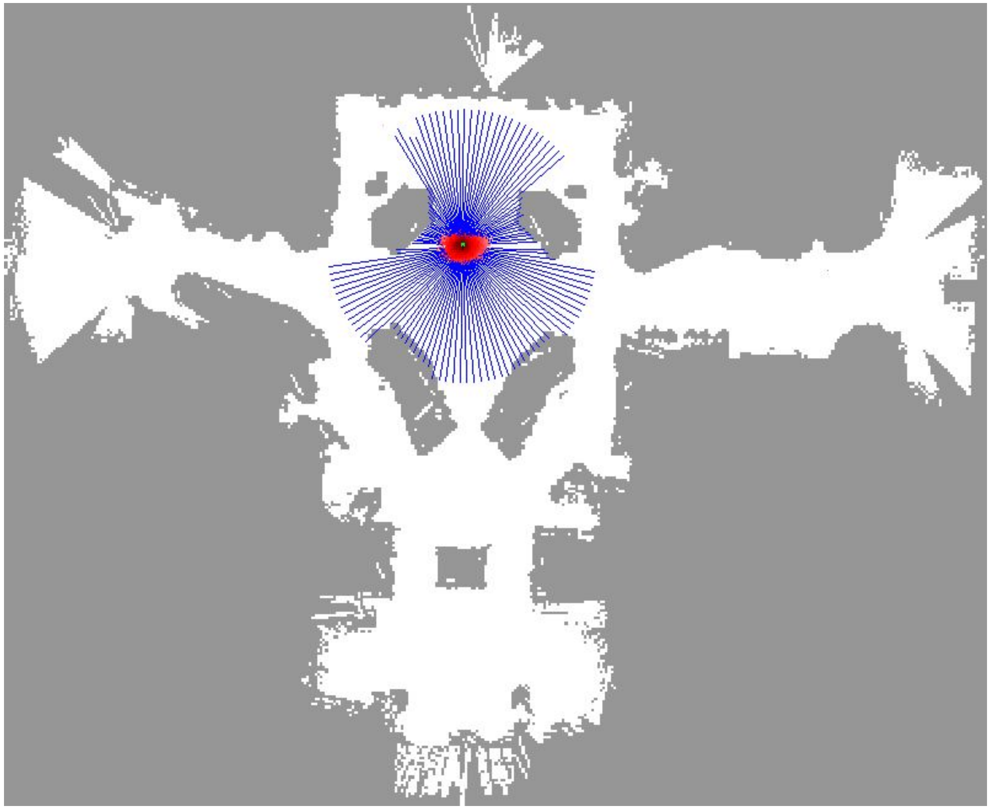
2D Example

- Propagate with motion model



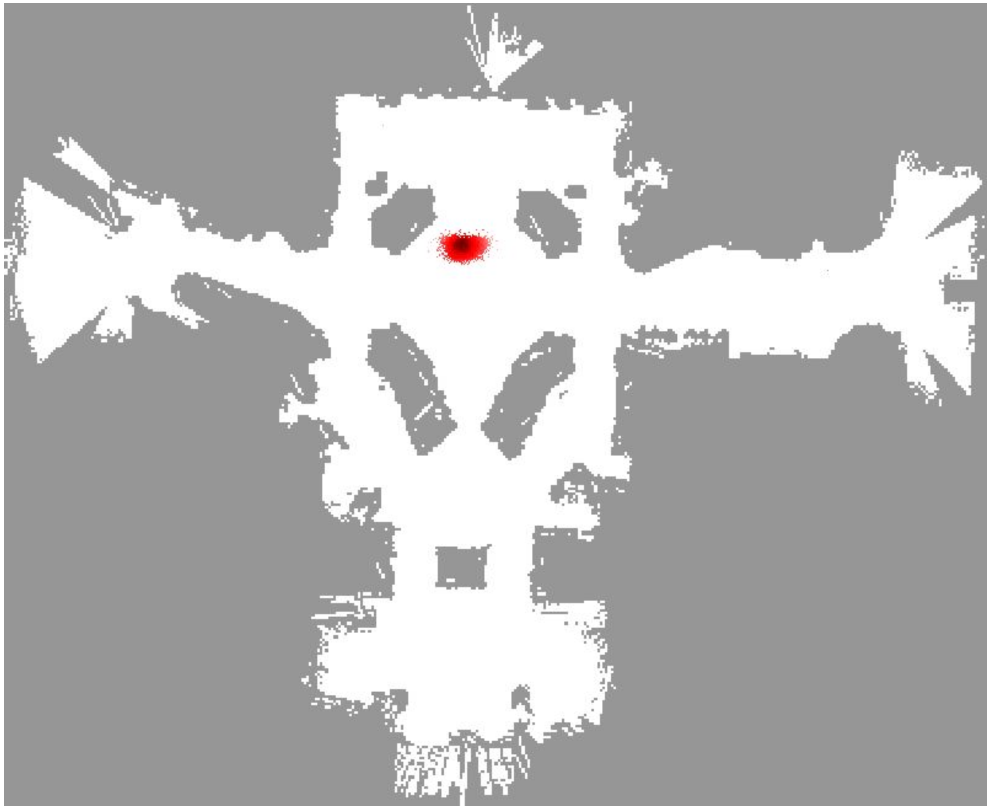
2D Example

- Get measurements



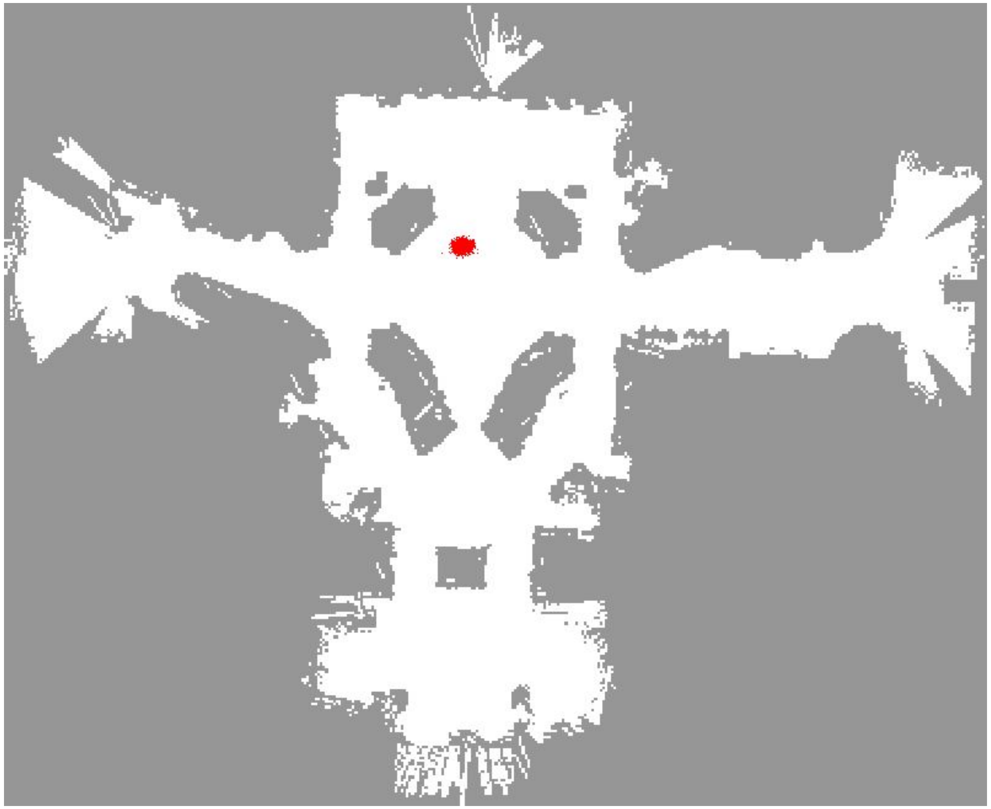
2D Example

- Get measurements



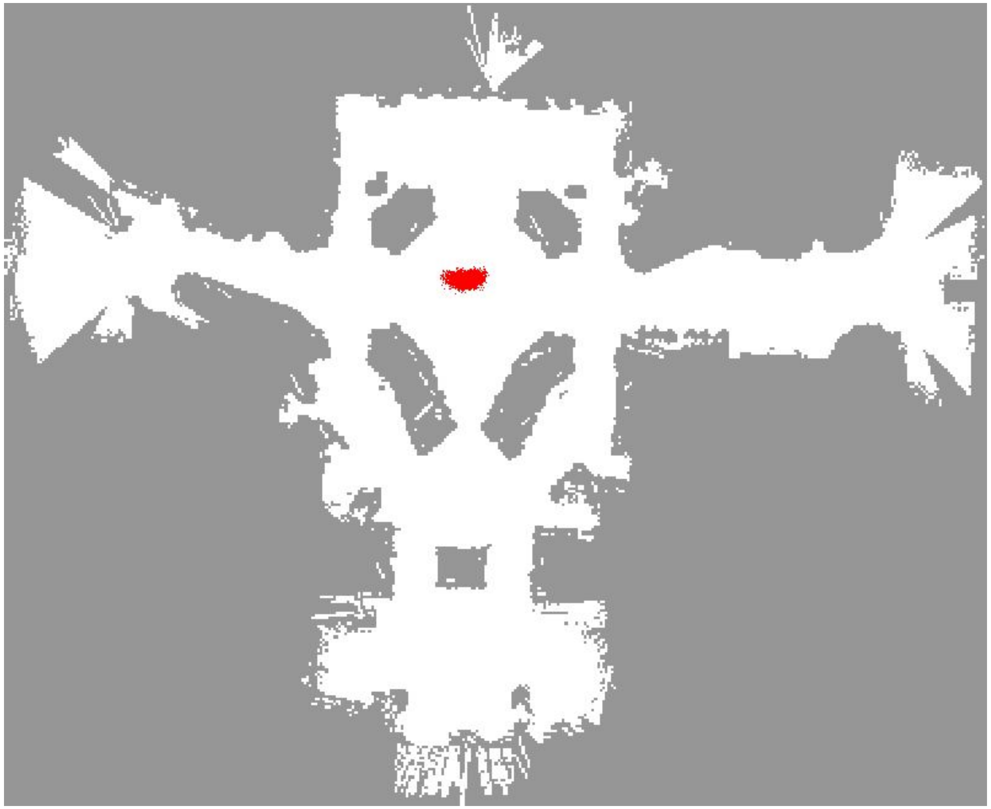
Resample

- Resample



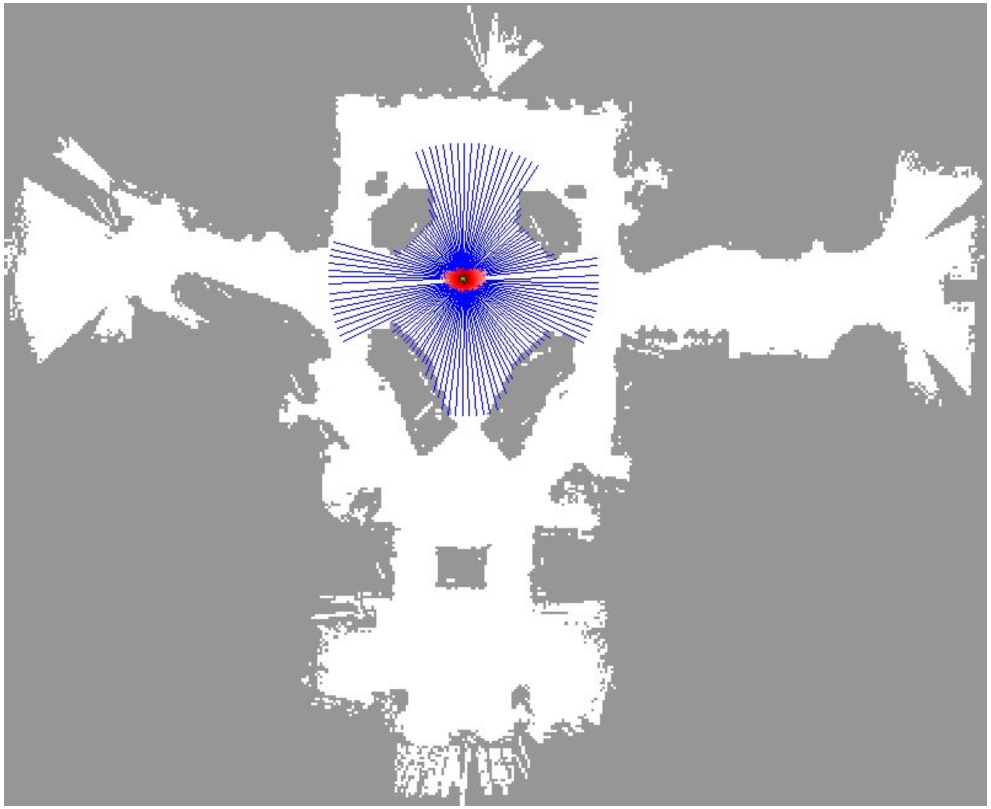
2D example

- Propagate with motion model



2D example

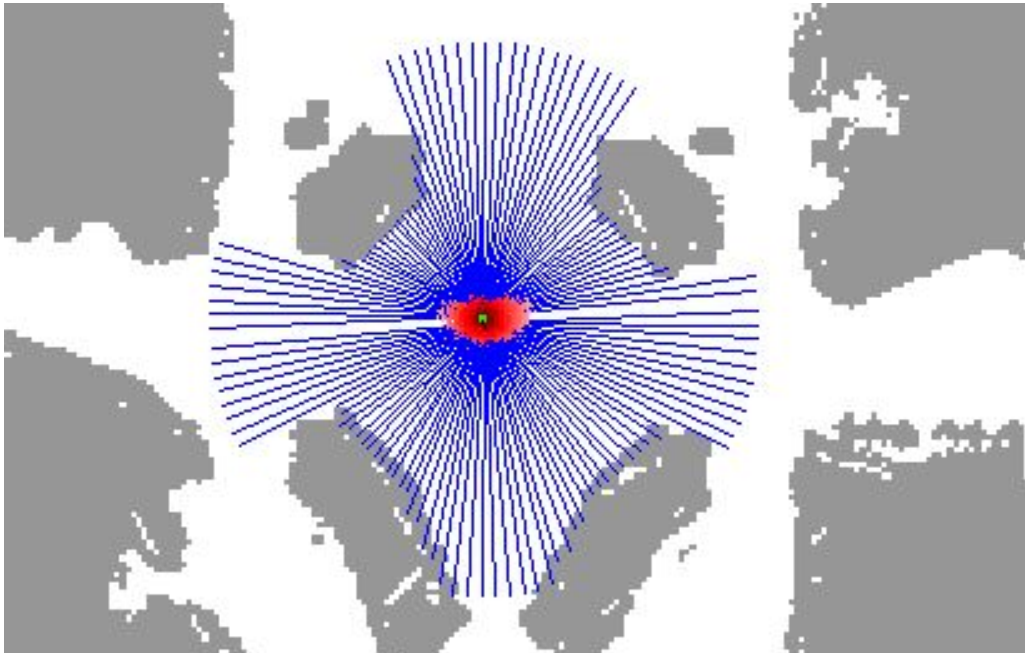
- Get measurements



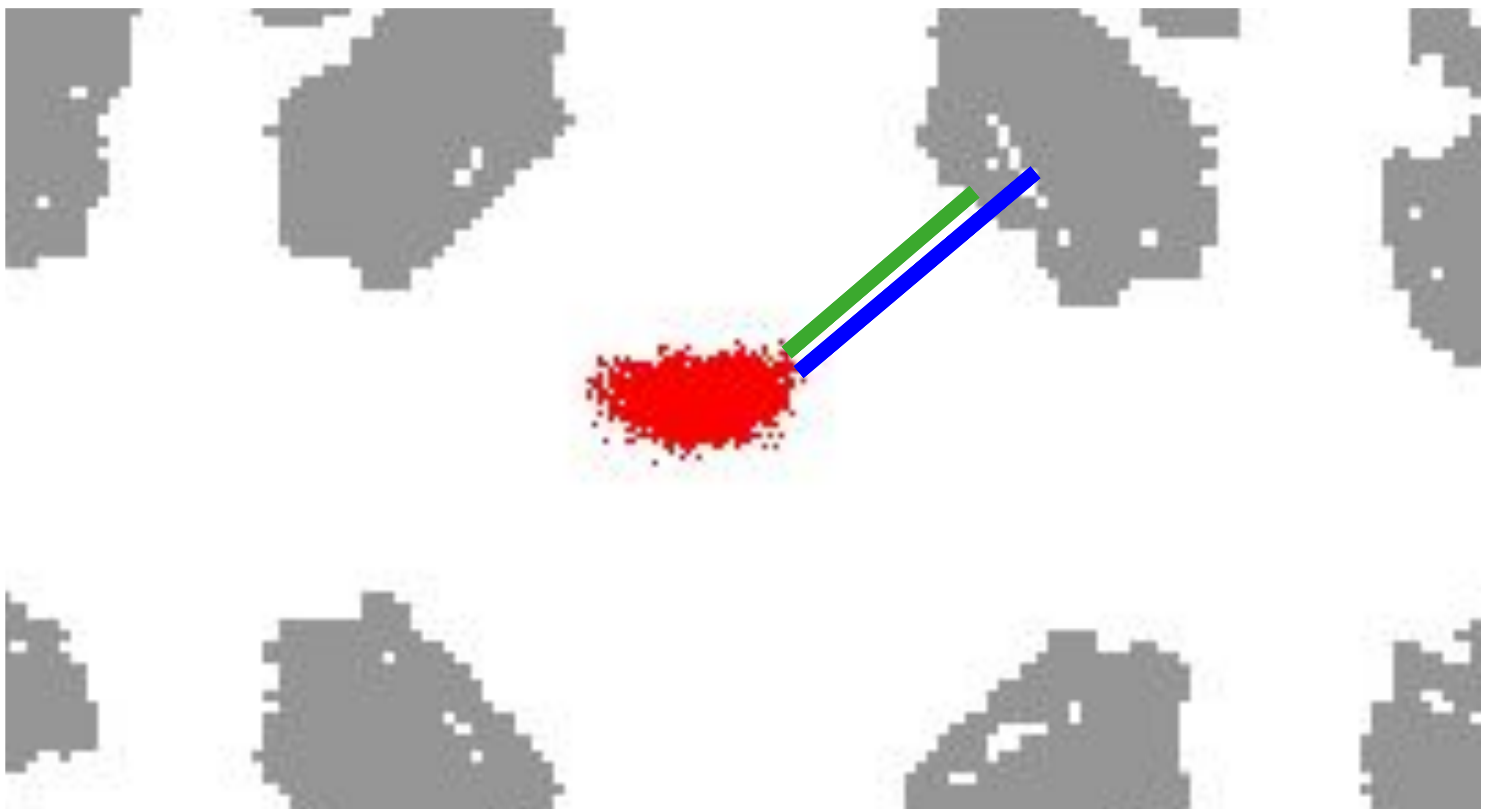
and so forth...

What should the sensor model look like?

- Depends on the sensor
- Previous example uses a laser range finder (LRF), which sends out lasers at known angles and measures the range
- Therefore we have a 1-dimensional measurement which is conditioned on the estimated state and the map



Range Sensor Model



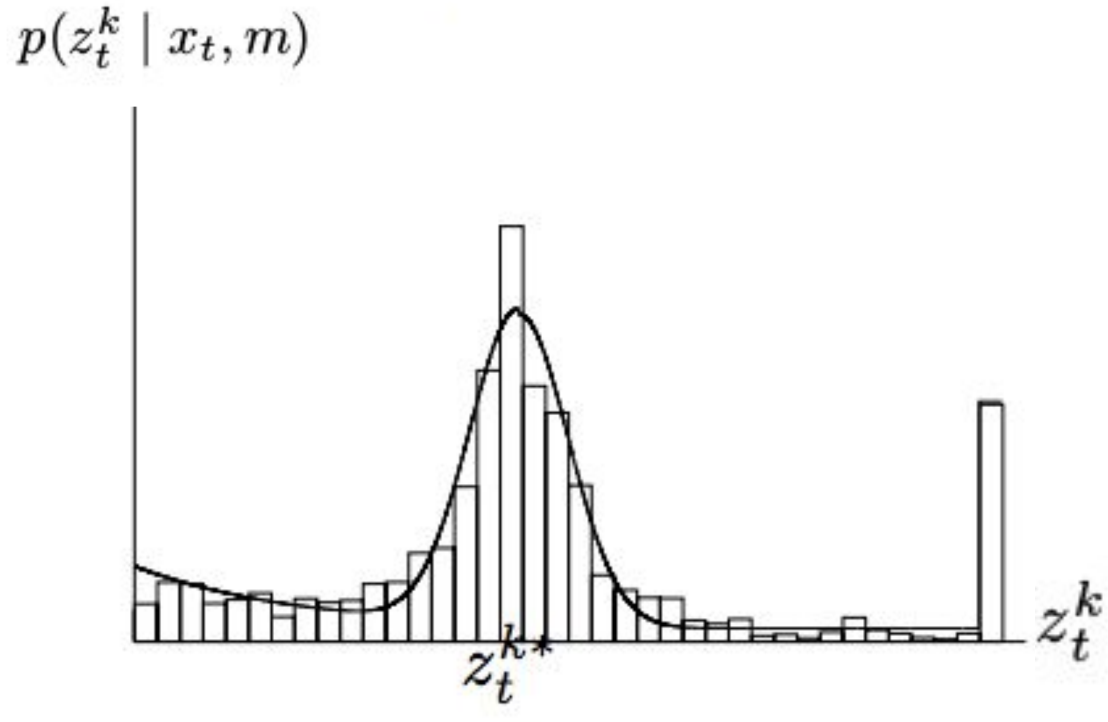
z_t^*

z_t

$p(z_t^k \mid x_t, m)$??

Range Sensor Model

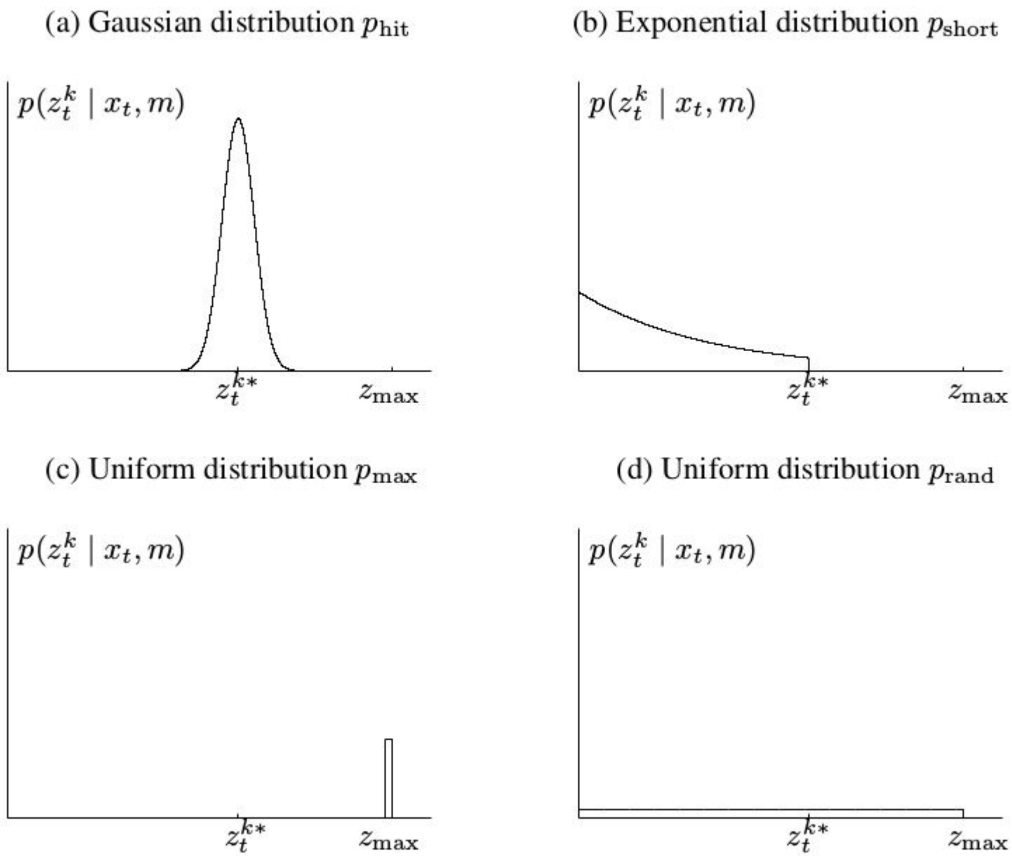
- Gives likelihood of seeing measurement z_t, given:
  - State x_t
  - Map m



Range Sensor Model

Superposition of:

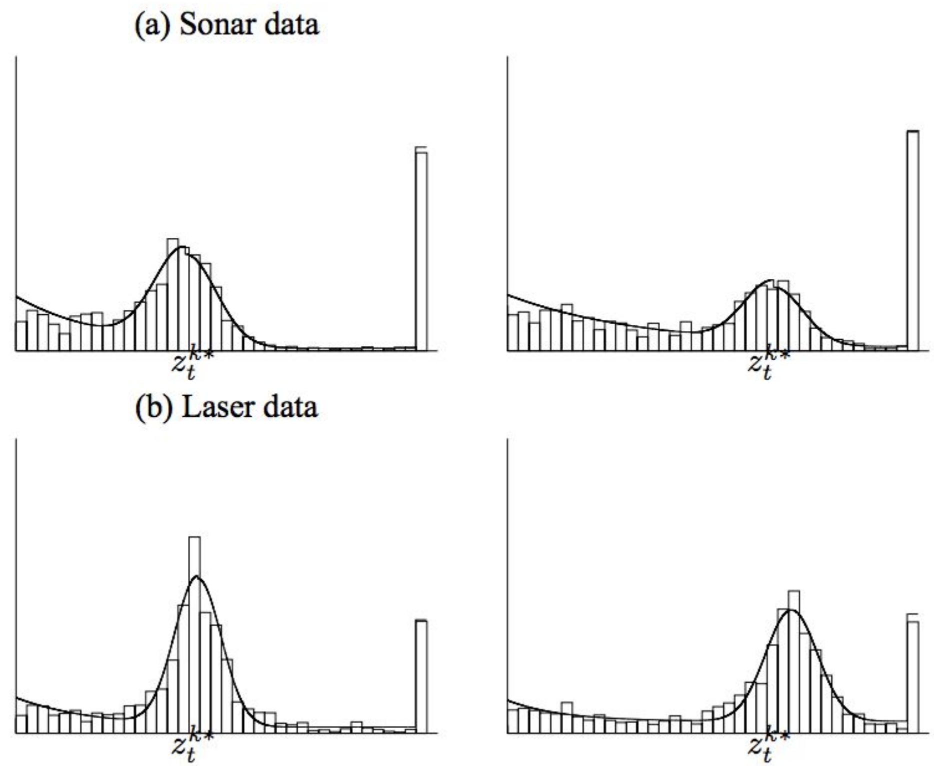
- Gaussian centered at predicted range according to map + particle
- Uniform at end-of-range
- Exponential decay
- Maybe another uniform over entire range
- ...



Range Sensor Model

Superposition of:

- Gaussian centered at predicted range according to map + particle
- Uniform at end-of-range
- Exponential decay
- Maybe another uniform over entire range
- ...



Particle Filter Algorithm

$$\boxed{p(x_{0:t} \mid z_{1:t}, u_{1:t}) \;=\; \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}$$

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$)**:**
  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
  for $m = 1$ to M do
    sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$ ← motion model
    $w_t^{[m]} = p(z_t \mid x_t^{[m]})$ ← sensor model
    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  endfor
  for $m = 1$ to M do
    draw i with probability $\propto w_t^{[i]}$ ← importance sampling
    add $x_t^{[i]}$ to $\mathcal{X}_t$
  endfor
  return $\mathcal{X}_t$

$\mathcal{X}_{t-1}$ - previous particle set

$\mathcal{X}_t$ - output particle set

Implementation problems

- Stationary robot
  - No motion or sensor data
  - Converges to one particle (likely incorrect)
- Particle deprivation
  - Too few particles
  - “Correct” particles die out
- Very precise sensor models can result in poor performance
  - Accurate odometry, poor range sensor?
  - Poor odometry, accurate range sensor?

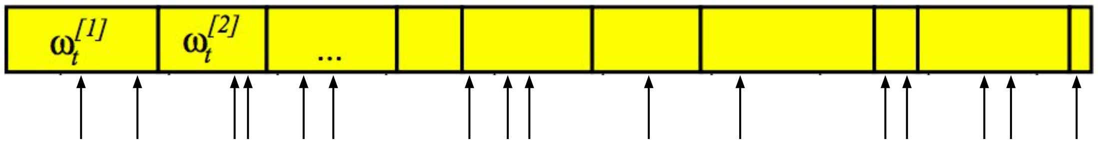
Tips & Tricks

- Low variance resampling

- Store logs of weights
- Experiment with different sensor models

Tips & Tricks

- Low variance resampling



- Store logs of weights
- Experiment with different sensor models

Tips & Tricks

- Low variance resampling

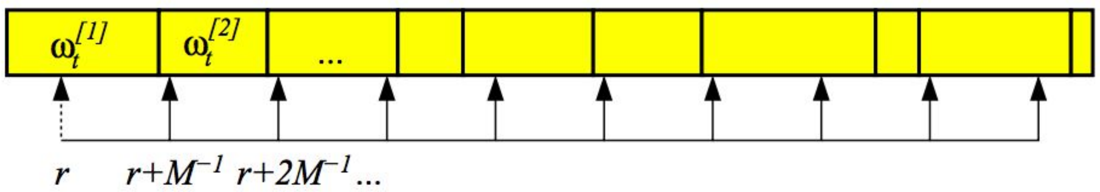
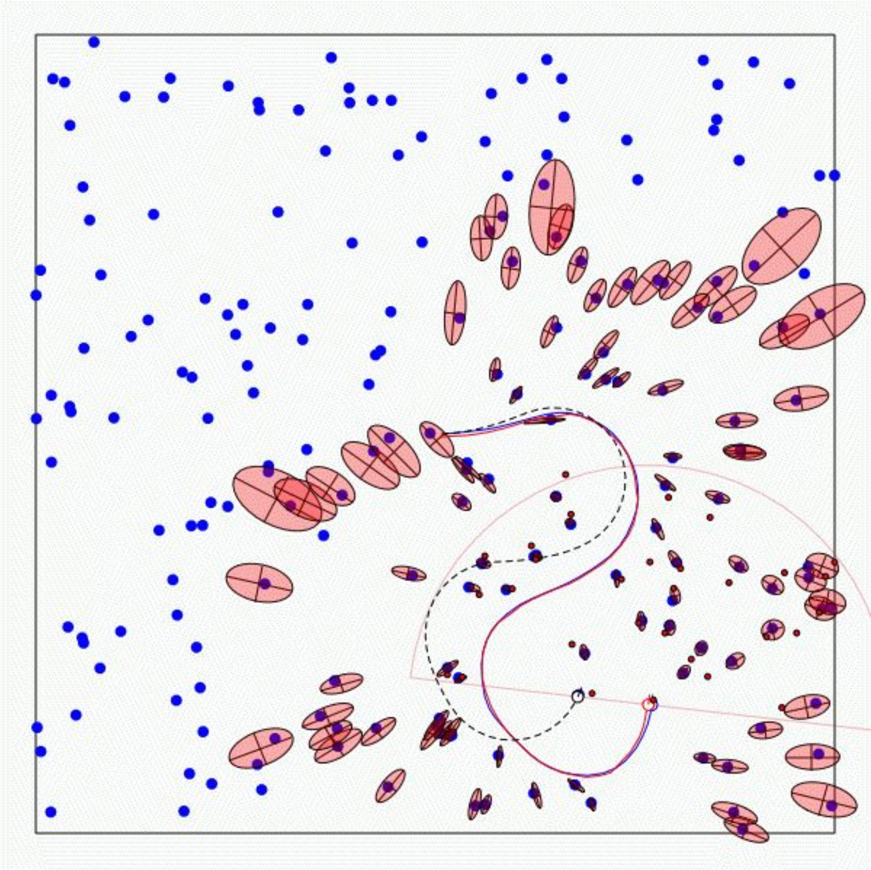


Figure 4.3 Principle of the low variance resampling procedure. We choose a random number r and then select those particles that correspond to $u = r + (m-1) \cdot M^{-1}$ where $m = 1, \ldots, M$.

- Store logs of weights
- Experiment with different sensor models

FastSLAM

- FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem [Montemerlo et al., 2002]



Localization vs. SLAM

- A particle filter can be used to solve both problems
- Localization: state space < x, y, θ>
- SLAM: state space < x, y, θ, map>
  - for landmark maps = $< l_1 , l_2 , ..., l_m >$
  - for grid maps = $< c_{11} , c_{12} , ..., c_{1n} , c_{21} , ..., c_{nm} >$
- Problem: The number of particles needed to represent a posterior grows exponentially with the dimension of the state space!

Dependencies

- Localization:

$$p(x_{0:t} \mid z_{1:t}, u_{1:t}) \;=\; \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})$$

- Is there a dependency between the dimensions of the state space?
- If so, can we use the dependency to solve the problem more efficiently?
- In the SLAM context
  - The map depends on the poses of the robot.
  - We know how to build a map given the position of the sensor is known.

Factored Posterior (Landmarks)

poses map observations & control inputs

$$p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) =$$

$$p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} \mid x_{1:t}, z_{1:t})$$

SLAM posterior

Robot path posterior landmark positions

Does this help to solve the problem?

Factorization first introduced by Murphy in 1999

Factored Posterior (Landmarks)

poses map observations & control inputs

$$p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) =$$

$$p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} \mid x_{1:t}, z_{1:t})$$

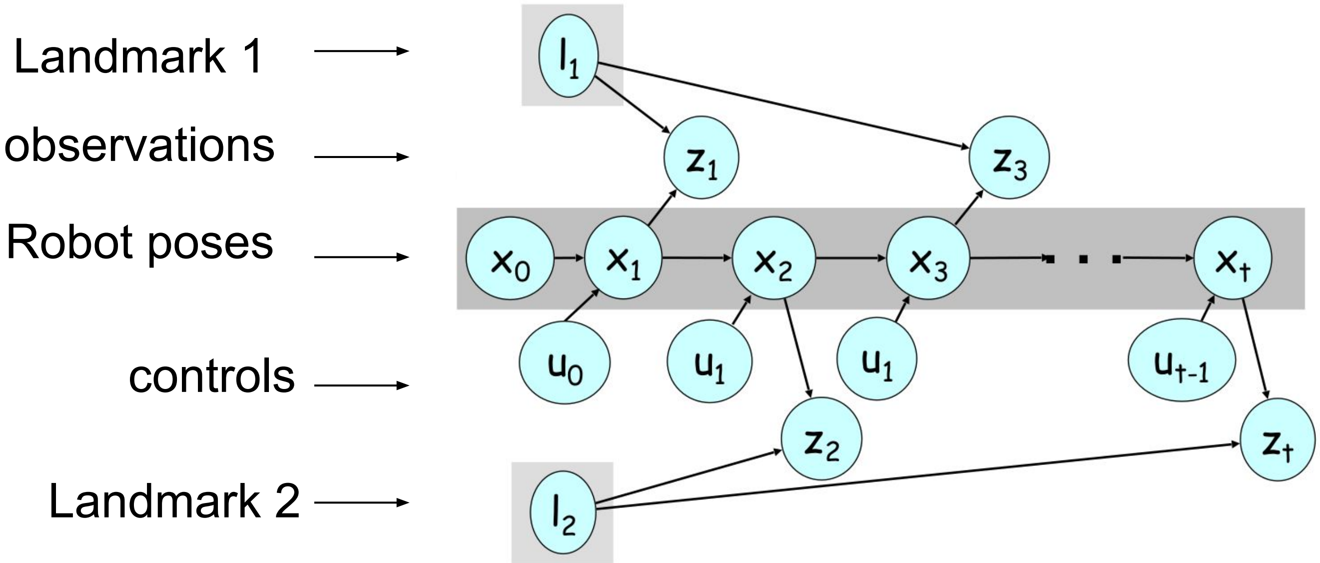
SLAM posterior

Robot path posterior landmark positions

Solve localization and mapping separately!

Factorization first introduced by Murphy in 1999

Mapping using Landmarks



Knowledge of the robot’s true path renders landmark positions conditionally independent

Factored Posterior

$$
\begin{aligned}
& p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) \\
&= p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} \mid x_{1:t}, z_{1:t}) \\
&= p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^{M} p(l_i \mid x_{1:t}, z_{1:t})
\end{aligned}
$$

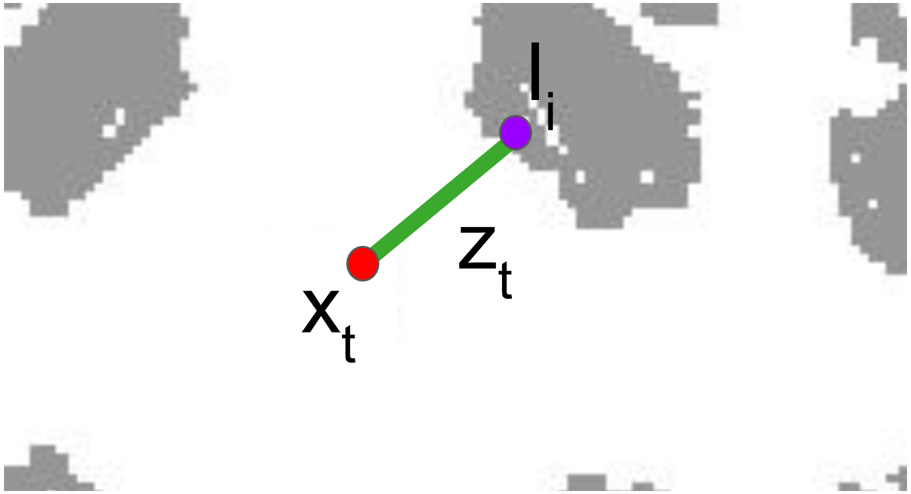
Robot path posterior
(localization problem)

Conditionally independent
landmark positions

Rao-Blackwellization

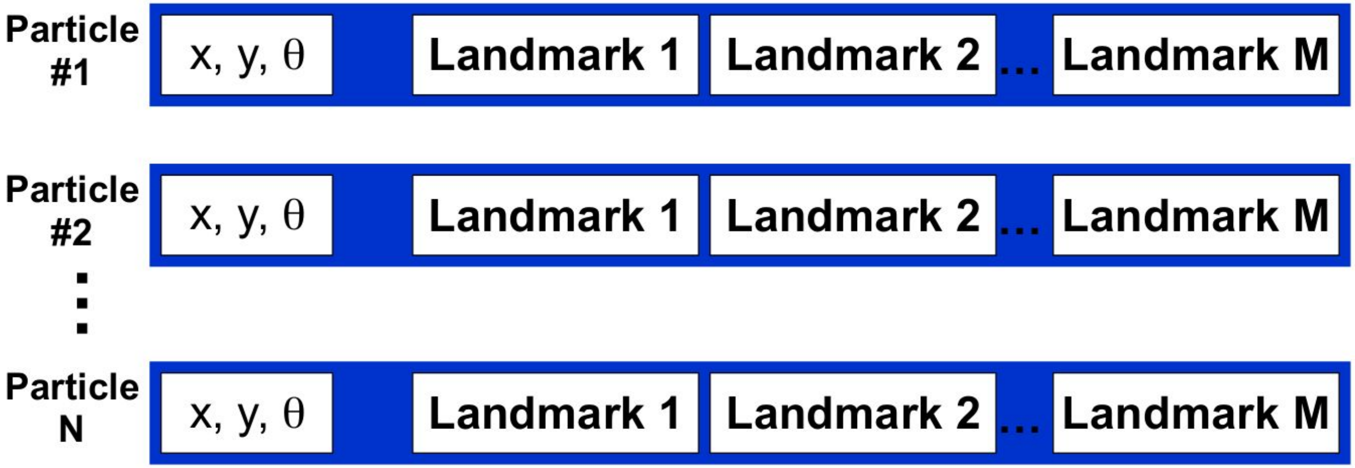
$$p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) =$$
$$p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^{M} p(l_i \mid x_{1:t}, z_{1:t})$$

- This factorization is also called Rao-Blackwellization
- Given that the second term can be computed efficiently, particle filtering becomes possible!

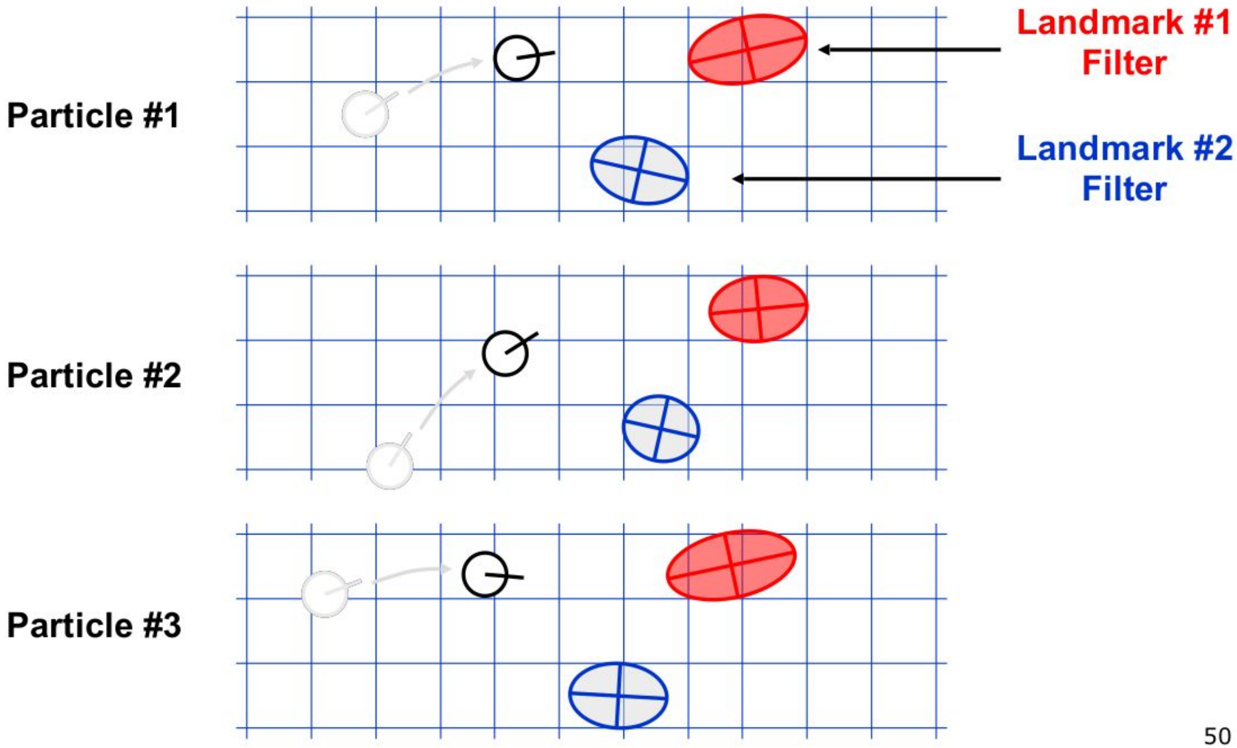


FastSLAM

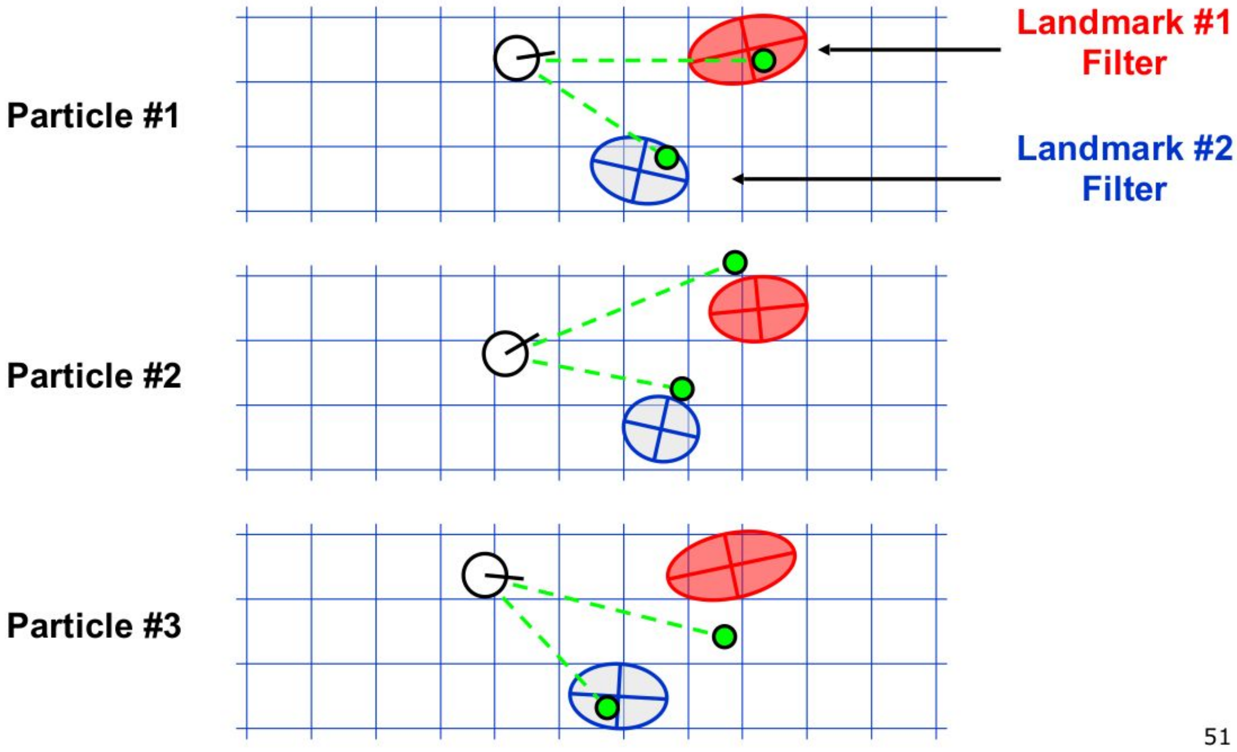
- Rao-Blackwellized particle filtering based on landmarks [Montemerlo et al., 2002]
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain M EKFs



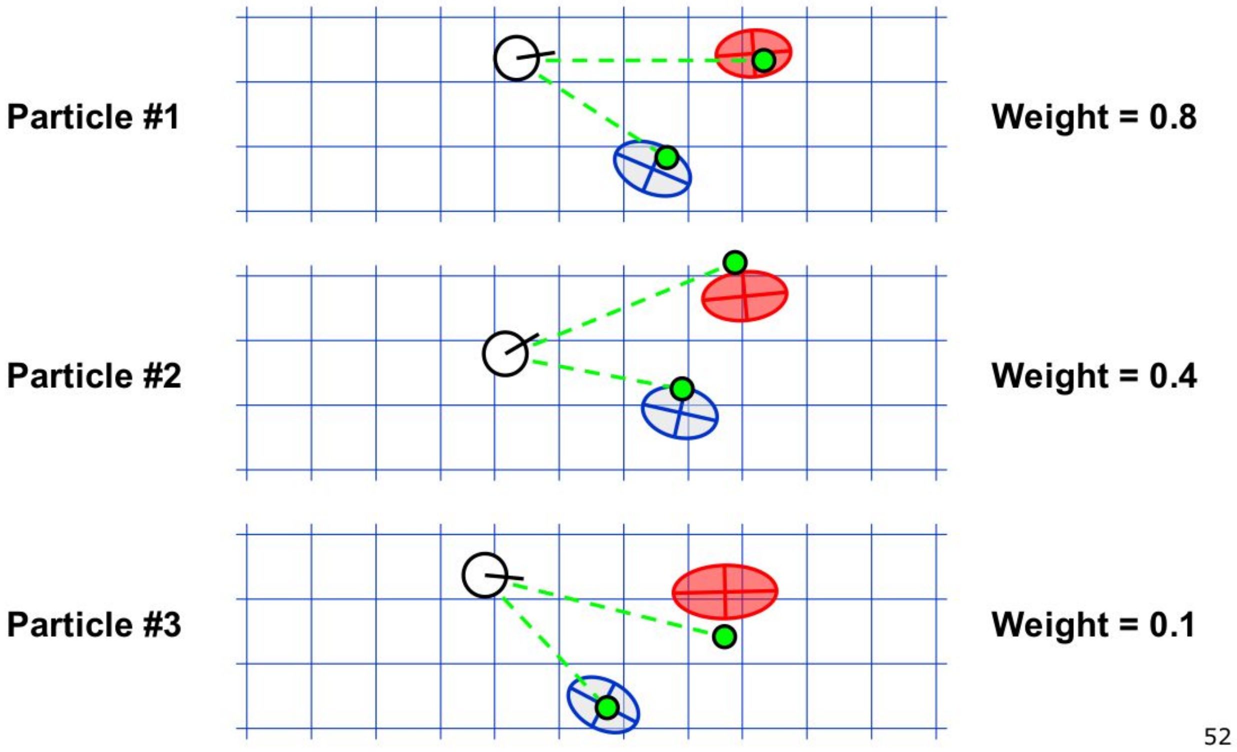
FastSLAM - Action Update



FastSLAM - Sensor Update



FastSLAM - Sensor Update



FastSLAM Complexity

- Update robot particles based on control u_{t-1}

O(N)
Constant time per particle

- Incorporate observation z_t into Kalman filters

O(N•log(M))
log time per particle

- Resample particle set

O(N•log(M))
log time per particle

N = Number of particles
M = Number of map features

O(N•log(M))
log time per particle

FastSLAM Complexity

- Update robot particles based on control u_{t-1}

  O(N)

  Constant time per particle

- Incorporate observation z_t into Kalman filters

  O(N•log(M))

  log time per particle

- Resample particle set

  O(N•log(M))

  log time per particle

N = Number of particles
M = Number of map features

O(N•log(M))

log time per particle

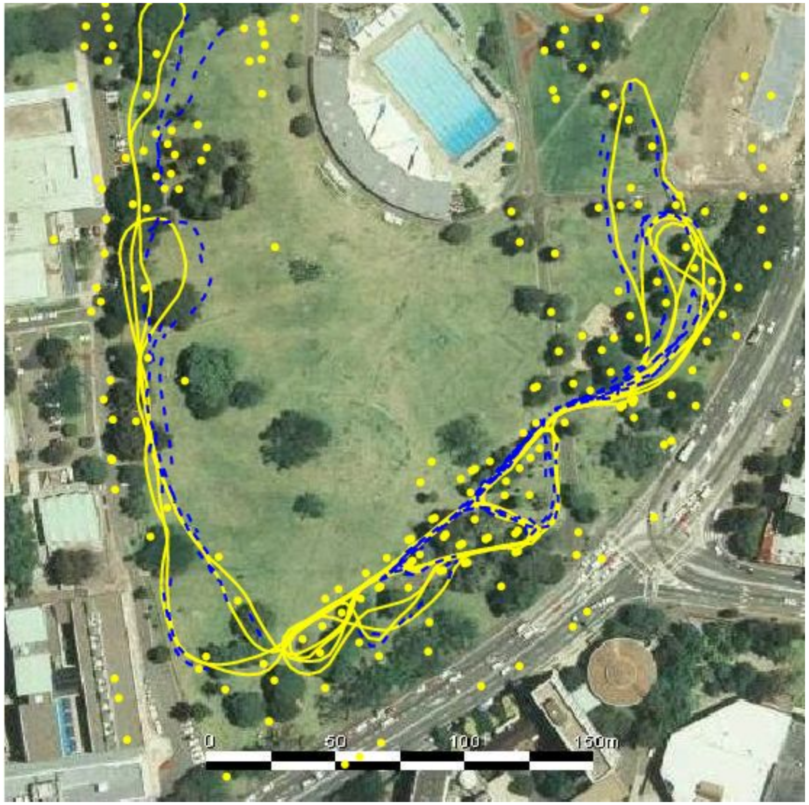
EKF SLAM: O(K^2.8 + N^2)

K = Measurement vector dimension
N = State vector dimension

Results - Victoria Park

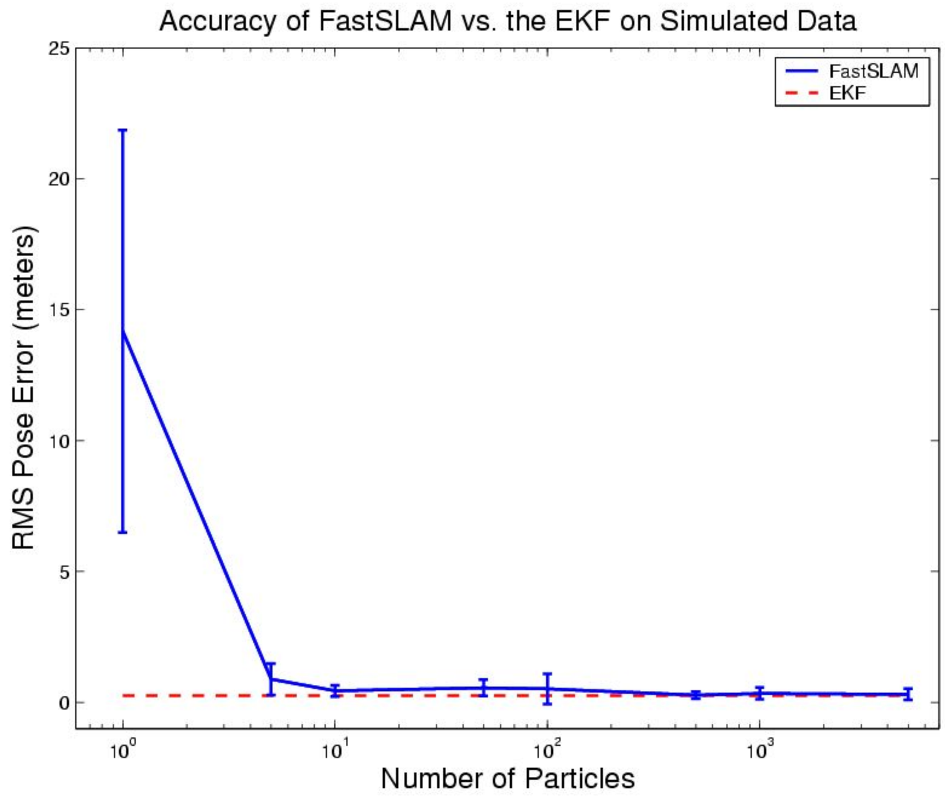
- 4 km traverse
- < 5 m RMS position error
- 100 particles

Blue = GPS
Yellow = FastSLAM

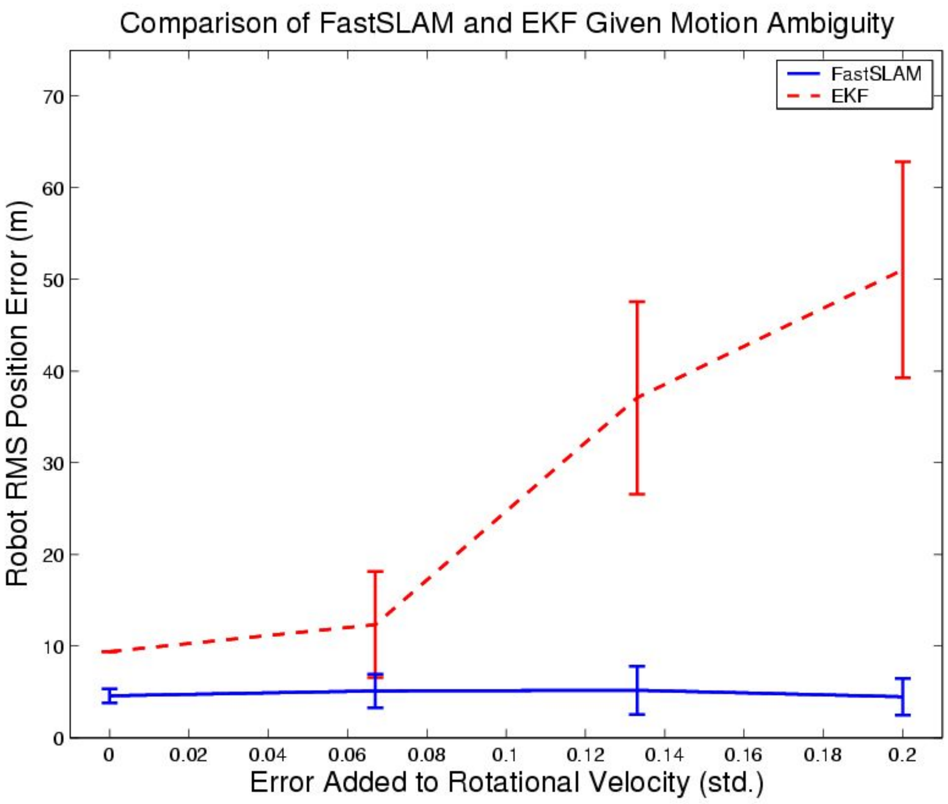


Dataset courtesy of University of Sydney

Results - Number of Particles



Results - Motion Uncertainty



FastSLAM slides courtesy of

Sebastian Thrun, Wolfram Burgard, Dieter Fox

Publicly available at www.probabilistic-robotics.org