PARTICLE FILTERS

System Identification, Prof. Battilotti
Ahmed Elhelow and Michela Ricciardi Celsi

1.INTRODUCTION

An alternative to Gaussian techniques are nonparametric filters. They do not rely on a fixed functional form of the posterios, such as Gaussians, but they approximate posteriors by a finite number of values, each corresponding to a region in state space. Some nonparametric Bayes filters rely on a decomposition of the state space, where each value corresponds to the probability of the posterior density in a compact subregion of the state space. Others approximate the state space by random samples drawn from the posterior distribution. In any case, the quality of the approximation depends on the number of parameters used to represent posterior. In fact, as the number of parameters goes to infinity, nonparametric techniques tend to converge uniformly to the correct posterior.
The most common nonparametric filters are: *Histogram filters*, that decompose the state space into finitely many regions and represent the posterior by histogram, and *Particle filters*, that represent posteriors by finitely many samples. They both do not make strong parametric assumptions on the posterior density. They are chosen usually when a robot has to cope with phases of global uncertainty and when it faces hard data association problems that yield distinct hypotheses.
Nonparametric techniques that we studied make possible to adapt the number of parameters to the complexity of the posterior. For example, when the posterior is of low complexity, they use only small numbers of parameters. On the other hand, for complex posteriors, the number of parameters grows lagers. These types of techniques are called *adaptive* and they enable robots to make decisions in real time.

2.PROBLEM DEFINITION

The key idea of the particle filter is to represent the posterior Bel(X_t) by a set of random state samples drawn from this posterior.
Considering u_t as the control information measuring the dynamics of the system and z_t as the sensor measurement, the problem to be solved is:
given a sample-based representation of $\text{Bel}(X_t) = P(X_t|Z_1, \dots., Z_t, u_1, \dots., u_t)$, find a sample-based representation $\text{Bel}(X_{t+1}) = P(X_{t+1}|Z_1, \dots., Z_t, z_{t+1}, u_1, \dots., u_{t+1})$.
The samples of a posterior distribution are called particles and are denoted as $S_t = \{x_t^1, x_t^2, \dots., x_t^N\}$.
Each particle x_t^n, with $1 \leq n \leq N$, is a concrete instantiation of the state at time t, that is a hypothesis as to what the true world state may be at time t. N denotes the number of particles in the particle set S_t and it is often a large number, e.g., N = 1000. In some implementations N is a function of t or of other quantities related to the belief bel(x_t).
The intuition behind particle filters is to approximate the belief bel(x_t) by the set of particles. Ideally, the likelihood for a state hypothesis x_t to be included in the particle set shall be proportional to its Bayes filter posterior bel(x_t):

$$x_t^n \sim p(x_t|z_{1:t}, u_{1:t})$$

This property holds only asymptotically for $N \to \infty$. For finite N, particles are drawn from a slightly different distribution. This difference is negligible as long as the number of particles is not too small, like $N \geq 100$. As a consequence, the denser a subregion of the state space is populated by samples, the more likely it is that the true state falls into this region. b
We then follow two different phases, the first one is the dynamics update, that consists of:

given a sample-based representation of $\text{Bel}(X_t) = P(X_t|Z_1, \ldots, Z_t, u_1, \ldots, u_t)$, find a sample-based representation of $P(X_{t+1}|Z_1, \ldots, Z_t, z_{t+1}, u_1, \ldots, u_{t+1})$.
The solution is given by:
For $i = 1, \ldots, N$, sample x_t^i from $P(X_{t+1}|X_t = x_t^i,\ u_t)$.

The second one is the observation update. During this phase, we take the sensor measurements and assign each particle a weight that is equal to the probability of observing the sensor measurements from that particle's state. In particular, it consists of:
given a sample-based representation of $P(X_t|Z_1, \ldots, Z_t)$, find a sample-based representation of $P(X_{t+1}|Z_1, \ldots, Z_t, z_{t+1}) = C * P(X_{t+1}|Z_1, \ldots, Z_t, z_{t+1}) * P(Z_{t+1}|X_{t+1})$.
The solution is given by:
For $i = 1, \ldots, N$, sample x_t^i from $w_{t+1}^i = w_t^i * P(Z_{t+1}|X_{t+1} = x_{t+1}^i)$. Hence, the distribution is represented by the weighted set of samples $\{< x_{t+1}^1,\ w_{t+1}^1 >, < x_{t+1}^2,\ w_{t+1}^2 >, \ldots, < x_{t+1}^N,\ w_{t+1}^N >\}$. w_t^i are non-negative numerical factors called importance weights, which sum up to 1.

3.IMPORTANCE SAMPLING (IS)

In practice, we rarely can directly draw samples from the distribution $p(x)$, that is the reason why we apply the trick known as the importance sampling (IS). It allows to draw samples from the known distribution called *proposal density* $\pi(.)$, having $x^i \sim \pi$. The trick consists of sampling from the available distribution and reweight samples to fix it.

For any function f, manipulating the expectation over the distribution yields the following approximation:

$$E_{X\sim p}[f(X)] = \int_x f(x)p(x)dx = \int_x f(x)p(x)\frac{\pi(x)}{\pi(x)}dx$$

$$= \int_x f(x)\frac{p(x)}{\pi(x)}\pi(x)dx = E_{X\sim\pi}\left[\frac{p(X)}{\pi(X)}f(X)\right] \approx \frac{1}{n}\sum_{i=1}^{N}\frac{p(x^i)}{\pi(x^i)}f(x^i)$$

Monte Carlo approximation

f could be whether a grid cell is occupied or not or whether the position of a robot is within 5 cm of some (x, y). Therefore, $w^i = \frac{1}{N}\frac{p(X)}{\pi(X)}$ are the importance weights. It is straightforward to notice that if $\pi(x) = 0$, then $p(x) = 0$.

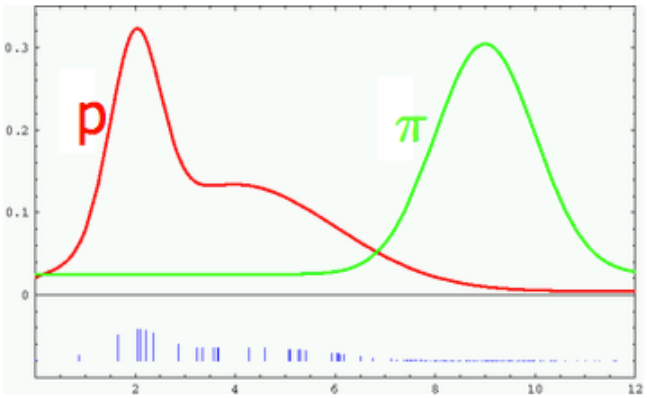


Figure 1 Comparison of distribution we wish to sample from (p) and distribution we can sample from (π).

Figure 1 shows that there are regions to sample that we are not sampling at all. In fact, the efficiency depends on the choice of the *importance distribution* (also knowns as *proposal density*). The revisited algorithm of particle filters is shown below:

1. Algorithm particle_filter(S_{t-1}, u_t , z_t):
2. $S_t = \varnothing, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ — ***Generate new samples***
4. Sample index $j(i)$ from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $\pi(x_t \mid x_{t-1}^{j(i)}, u_t, z_t)$
6. $w_t^i = \frac{p(z_t \mid x_t^i)\, p(x_t^i \mid x_{t-1}^i, u_t)}{\pi(x_t^i \mid x_{t-1}^i, u_t, z_t)}$ — ***Compute importance weight***
7. $\eta = \eta + w_t^i$ — ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ — ***Insert***
9. **For** $i = 1 \ldots n$
10. $w_t^i = w_t^i / \eta$ — ***Normalize weights***
11. ***Return*** S_t

Figure 2 Particle Filter using IS

After computing the *importance weights* as described above, the weights must be normalized as :

$$w_{norm}^i = \frac{w_t^i}{\sum_{j=1}^N w_t^j}$$

We must pay attention on choosing an optimal *proposal density*, in order to have a uniform distribution that could cover all the regions that must be sampled. In particular, the proposal should cover all the relevant portion of the target otherwise some feasible samples might not be generated:

$$p(x) > 0 \Rightarrow \pi(x) > 0$$

To reach this purpose, we search for an *optimal proposal* π.

If we choose $\pi(x_t|x_{t-1}^i, u_t, z_t) = p(x_t|x_{t-1}^i, u_t, z_t)$, the weights are compted as:

$$w_t^i = \frac{p(z_t|x_t^i)p(x_t^i|x_{t-1}^i, u_t)}{p(x_t^i|x_{t-1}^i, u_t, z_t)}.$$

By applying the Bayes rule we obtain: $p(x_t^i|x_{t-1}^i, u_t, z_t) = \frac{p(z_t|x_t^i, u_t, x_{t-1}^i)p(x_t^i|x_{t-1}^i, u_t)}{p(z_t|x_{t-1}^i, u_t)}$.

By substitution and semplification, the weights are:

$$w_t^i = p(z_t|x_{t-1}, u_t) = \int p(z_t|x_t)\ p(x_t|x_{t-1}^i, u_t)dx_t$$

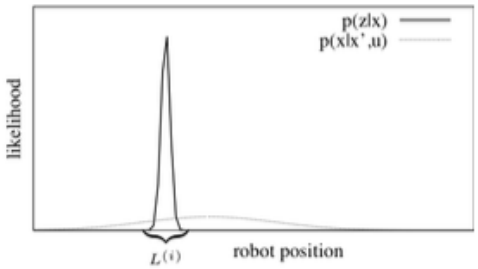
However, it is difficult to sample $p(x_t|x_{t-1}^i, u_t, z_t)$ and the *importance weights* are computed with the integral, that is a very expensive procedure. In the ideal case of sampling from the target distribution, the weights would be uniform.

Example 1- Nonlinear Gaussian State Space Model:

The model is represented by: $\begin{cases} x_t = f(x_{t-1}, u_t) + v_t \\ z_t = Cx_t + w_t \end{cases}$ where $v_t \sim \mathcal{N}(0, \Sigma_v)$ and $w_t \sim \mathcal{N}(0, \Sigma_w)$

Then, $p(x_t|x_{t-1}^i, u_t, z_t) = \mathcal{N}(m_t, \Sigma)$.
Taking into account that $\Sigma = (\Sigma_v^{-1} + C^T\Sigma_v^{-1}C)^{-1}$ and $m_t = \Sigma(\Sigma_v^{-1} f(x_{t-1}, u_t) + C^T\Sigma_w^{-1}z_t)$ and $p(z_t|x_{t-1}, u_t) \propto \exp\left(-\frac{1}{2}(z_t - Cf(x_{t-1}, u_t))^T(\Sigma_v + C\Sigma_w C^T)^{-1}(z_t - Cf(x_{t-1}, u_t))\right)$.

Example 2 – Approximating optimal proposal for localization:



[Grisetti, Stachniss, Burgard, T-RO2006]

Particles are propagated according to the motion model. Each particle is a pose hypothesis and the proposal is the motion model. A solution could be to use smoothed likelihood such that more particles retain a meaninful weight, but in this case information is lost. A better approach consists of generating a gaussian approximation to optimal sequence proposal, as it is done in the following algorithm:

1. Initial guess $x_t'^i = f(x_{t-1}^i, u_t)$
2. Execute scan matching starting from the initial guess $\hat{x}_t^i$, resulting in pose estimate $x_t'^i$.
3. Sample K points $\{x_1, \ldots, x_K\}$ region around $\hat{x}_t^i$.
4. Proposal distribution is Gaussian with mean and covariance:

$$\mu_t^i = \frac{1}{\eta^i}\sum_{j=1}^{K} x_j p(z_t|x_j, m)p(x_j|x_{t-1}^i, u_t)$$

$$\Sigma_t^i = \frac{1}{\eta^i}\sum_{j=1}^{K} p(z_t|x_j, m)p(x_j|x_{t-1}^i, u_t)(x_j - \mu_t^i)(x_j - \mu_t^i)^\top$$

$$\eta^i = \sum_{j=1}^{K} p(z_t|x_j, m)p(x_j|x_{t-1}^i, u_t)$$

4.RESAMPLING

The resample step consists of choosing a new set of particles so that each particle survives in proportion to its weight. It means that part of particles will be duplicated and some other will be rejected: the higher is the weight, the greater is the chance that the particle will be drawn several times.
Consider running a particle filter for a system with deterministic dynamics and no sensors, due to resampling some particles will disappear and after running sufficiently long with very high probability all particles will be identical. Moreover, on the surface it could seem that the particle filter has uniquely determined the state.
Resampling induces loss of density. The variance of the particles decreases, while the variance of the particle set as an estimator of the true belief increases. The possible sultions using resampling could be:

- Adaptive resampling: compute the effective sample size $\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{N}(w_k^{(i)})^2}$ and resample only if the effective sample size is low.
- Low variance sampling: we choose a random number r and then select those particles that correspond to $u = r + (n-1)\cdot N^{-1}$ with $n = 1, \dots, N$ and $r \in [0, 1/N]$. It is very efficient because if all samples have same importance weight, then no samples are lost and it also requires lower computational complexity.
- Since loss diversity is caused by resampling from a discrete distribution, a possible solution could be regularization. It consists of considering the particles and then sample from the continuos density. For example, having the particles $\{x^{(1)}, \dots., x^{(K)}\}$, sample from the density $p(x) = \sum_{k=1}^{K} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-x^{(k)})^2}{\sigma^2}}$.

5.SEQUENTIAL IMPORTANCE SAMPLING (SIS) AND SEQUENTIAL IMPORTANCE RESAMPLING (SIR)

The SIS (Sequential Importance Sampling) is shown in the following algorithm:

- Sample $x^1_1, x^2_1, \dots, x^N_1$ from $P(X_1)$
- Set $w^i_1 = 1$ for all i=1,…,N
- For t=1, 2, …
  - Dynamics update:
    - For i=1, 2, …, N
      - Sample x^i_{t+1} from $P(X_{t+1} \mid X_t = x^i_t, u_{t+1})$
  - Observation update:
    - For i=1, 2, …, N
      - $w^i_{t+1} = w^i_t * P(z_{t+1} \mid X_{t+1} = x^i_{t+1})$
- At any time t, the distribution is represented by the weighted set of samples

  $\{ \langle x^i_t, w^i_t \rangle ; i=1,\dots,N\}$

It uses importance sampling sequentially. The problem is that the algorithm is degenerate. It can be shown that the variance of the weights increases at every step. Thus, it means that we will always converge to single non-zero weight $w^{(i)} = 1$ and the rest being zero. A solution to this problem consists of using resampling, that is represented by the following algorithm:

1. Algorithm **particle_filter**(S_{t-1}, u_t , z_t):
2. $S_t = \varnothing, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ — ***Generate new samples***
4. Sample index *j(i)* from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}^{j(i)}, u_t)$
6. $w_t^i = p(z_t \mid x_t^i)$ — ***Compute importance weight***
7. $\eta = \eta + w_t^i$ — ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ — ***Insert***
9. **For** $i = 1 \ldots n$
10. $w_t^i = w_t^i / \eta$ — ***Normalize weights***
11. ***Return*** S_t

Figure 3 SIR: Sequential Importance Resampling

6.PROPERTIES OF PRACTICAL FILTER

- Even with a large number of particles, it may happen that there are no particles in the vicinity of the correct state. This drawback is called *particle deprivation*.
  To avoid this situation it could be useful to add a small number of randomly generated particles when resampling. In this way, the particle deprivation reduces and adding random samples will cut out particles that were not very consistent with past evidence, but it may happen that an incorrect posterior estimate is done. The number of particles that must be added can be chosen by fixing a number, by monitoring the probability of sensor measurements or by comparing the values of average estimates when having reasonable state estimates and if the average estimate is low, inject the random particles.
- Consider the case a measurement is obtained with a noise-free sensor, then all particles would end up with weight zero, since it is very unlikely to have had a particle matching the measurement exactly. A possible solution to this problem is to artificially inflate the amount of noise in the sensors or to choose a better proposal density.

7.ADAPTING NUMBER OF PARTICLES: KLD-SAMPLING

Estimating the state of dynamic system based on noisy sensor measurements is extremely important in area as different as speech recognition, target tracking, mobile robot navigation and computer vision. Particle filters have been applied with great success to a variety of state estimation problems. The posterior probability density is estimated over the state space of a dynamic system. The key idea of this technique is to represent probability densities by sets of samples, as we described before. It is due to this representation, that particle filters combine efficiency with the ability to represent a

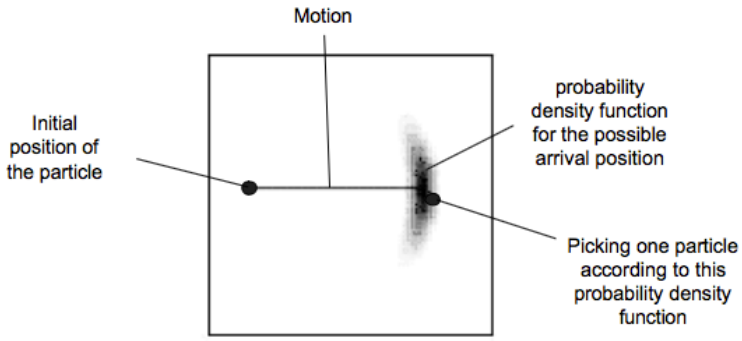
wide range of probability densities. By sampling proportion to likelihood, particle filters focus the computational resources on regions with high likelihood. An important way to increase efficiency of particle filters consists of adapting the number of samples over time. In fact, most existing approaches to particle filters use a fixed number of samples during the whole state estimation process. This can be inefficient since the probability densities can vary drastically over time. Referring to a robot localization problem, typically more particles are needed at the beginning of the localization run. The idea of KLD-Sampling is to partition the state-space. When sampling, keep track of number of bins occupied. Then, stop sampling when a threshold that depends on the number of occupied bins is reached. The bins can be implemented either as a fixed, multi-dimensional grid or more efficiently as tree structures. The objective is to bound the error introduced by the sample-based representation of the particle filter. To derive this bound, we assume that the true posterior is given by a discrete, piecewise constant distribution such as a discrete density tree or a multi- dimensional histogram. For such a representation we can determine the number of samples so that the distance between the maximum likelihood estimate (MLE) based on the samples and the true posterior does not exceed a pre-specified threshold . This approach is called KLD-sampling algorithm since the distance between the MLE and the true distribution is measured by the Kullback-Leibler distance, that is:

$K(p,q) = \sum_x p(x) log \frac{p(x)}{q(x)}$, where p and q are two probability distributions.

8.ROBOT LOCALIZATION WITH PARTICLE FILTERS

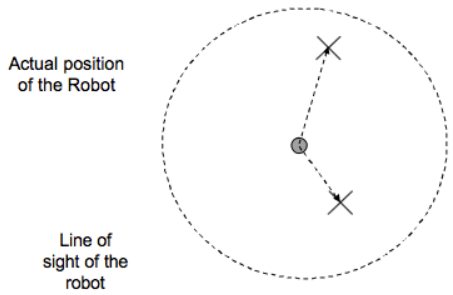
In robot localization, the particle filters are applied doing the following steps:

- Propagate each particle by applying dynamic model to the robot's position:



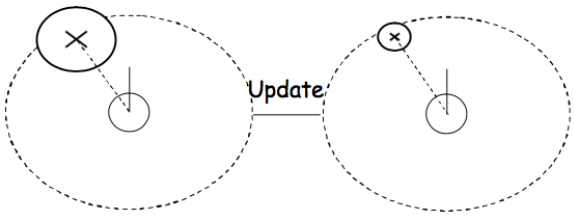
When the robot moves from the motion model, we know the posterior probability density function. We pick one particle to replace the former one.

- Observe landmarks:

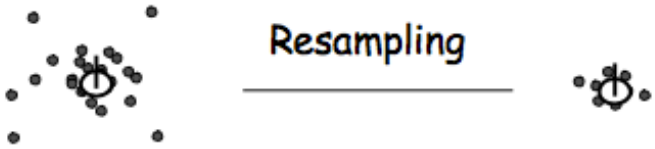


- Generate data association hypothesis according to their likelihood: each hypothesis is given a weight according to its likelihood and each particle's hypothesis is taken according to this weight.

- Update landmarks' estimates for each particle: after an observation, the robot updates its estimate concerning this landmark, and the uncertainty of this estimate.

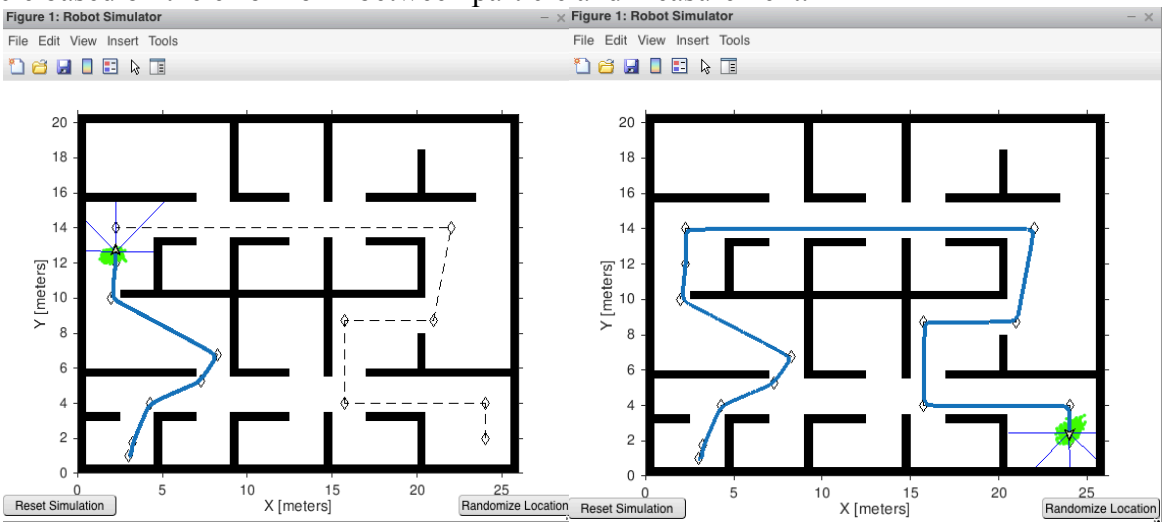


- Resample particles according to their likelihood:



9.SIMULATION

In the context of mobile robotics, particle filters are often used to track the position of the robot.
The so-called motion model $p(x_t|x_{t-1}, u_{t-1})$ is used to draw the next generation of particles. The importance weight $w_t^{(i)}$ of the i-th sample has to be computed based on the observation likelihood $p(z_t|m, x_t^{(i)})$ of the most recent sensor observation z_t given a map m of the environment and the corresponding pose of the particle.
We did a simulation considering a point robot. We defined a set of waypoints for the desired path of the robot: path = [3.00 1.00; 3.25 1.75; 4.25 4.00; 7.25 5.25; 8.25 6.75; 2.00 10.00; 2.25 12.00; 2.25 14.00; 22.00 14.00; 21.00 8.75; 15.75 8.75; 15.75 4.00; 24.00 4.00; 24.00 2.00].
Then, we built a map from range sensor readings and known pose of the robot and to do this it is essential that robot pose and laser sensor reading correspond to the same time.
Moreover, we configured the particle filter using 2000 particles. Initially all particles are randomly picked from a normal distribution with mean at initial state and unit covariance.
Each particle contains 3 variables: $(x \quad y \quad \vartheta)$, that represent the robot position and orientation.
It is also important to specify two functions: the *state transition* function, that evolves the particles based on a prescribed motion model so that the particles will form a representation of the proposal distribution, and the *measurement likelihood* function, that computes the likelihood for each predicted particle based on the error norm between particle and measurement.



10.BIBLIOGRAPHY

- Particle Filters. Pieter Abbeel, UC Berkley EECS.
- Probabilistic Robotics. Thrun S., Burgard W., Fox D. 1999-2000.
- Short Introduction to Particle Filters and Monte Carlo Localization. Cryill Stachniss.
- Real-time Particle Filters. Kwok C., Fox D., Meila M. University of Washington. 2004.
- Importance Sampling and Particle Filters. Bagnell D., Bartels J.
- Bayesian Approached to Localization, Mapping and SLAM. Choset H.
- Particle Filtering. Sarkka S. 2012.
- Particle Filters and Their Applications. Hsiao K., De Plinval-Salgues H., Miller J. 2005.
- KLD-Sampling: Adaptive Particle Filters. Fox D. University of Washington.2002
- Adapting the Sample Size in Particle Filters Through KLD-Sampling. Fox D. University of Washington. 2003