Dual Monte Carlo Localization algorithm for localization using wireless signal strength maps

Renato Miyagusuku, Atsushi Yamashita, Hajime Asama
Graduate School of Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656 Japan

Gaussian Processes have been previously used to model wireless signals strengths and create location-signal strength mappings. Such mappings can be used for robot localization by computing the posterior probability distributions of robot's positions given signal strength measurements. This work proposes the use of a dual Monte Carlo Localization algorithm that uses such posterior distributions as perceptual likelihood. Our approach is assessed and compared with an argmax and a Monte Carlo Localization algorithm in terms of accuracy and computation time.

1 Introduction

Robot localization or position estimation is the problem of determining a robot's pose relative to a given map of the environment. Most popular localization algorithms that use map information are based on the Bayes filter. In general, the Bayes filter addresses the problem of estimating any state considering the robot state evolution as a partially observable Markov chain (Hidden Markov Model - HMM). In the case of robot localization such state is the location of the robot $\mathbf{l} = (\mathbf{x}_x,\ \mathbf{x}_y,\ \theta)$ with $\mathbf{x}$ being the robot's position in an x-y Cartesian coordinate and θ its heading angle.

Bayes filters estimate this pose using a probability density estimation of the state space $\mathbf{l}_t$ conditioned on the time series data of robot actions $\mathbf{a}_{0:t}$, robot sensory measurement $\mathbf{o}_{0:t}$ and previous states $\mathbf{l}_{0:t-1}$. This posterior is called the *belief* of $\mathbf{l}$ - $Bel(\mathbf{l})$. Using Bayes' rule and the Markov assumption it is obtained that:

$$Bel(\mathbf{l}_t) \propto p(\mathbf{o}_t|\mathbf{l}_t) \int p(\mathbf{l}_t|\mathbf{l}_{t-1}, \mathbf{a}_t) Bel(\mathbf{l}_{t-1}) d\mathbf{l}_{t-1} \quad (1)$$

which is the basic equation for all Bayesian filters (see [4] for full description of the algorithms and proofs).

In order to implement eq. (1), three things are required: (a) a way to represent $Bel(\mathbf{l})$ and a priori distribution for $Bel(\mathbf{l}_0)$, which is usually assumed to be an uniform distribution; (b) the next state transition probability $p(\mathbf{l}_t|\mathbf{l}_{t-1}, \mathbf{a}_t)$; and (c), the perceptual likelihood $p(\mathbf{o}_t|\mathbf{l}_t)$.

In Monte Carlo Localization (MCL) and the dual MCL, the belief $Bel(\mathbf{l})$ is represented by a particle filter. Particle filters represent any distribution by a set of s weighted samples also called *particles*, distributed according to that distribution. The next state transition probability $p(\mathbf{l}_t|\mathbf{l}_{t-1}, \mathbf{a}_t)$ is implemented by a robot motion model - which varies depending on the robot used. A complete description of these models can be found at [4].

Finally, the perceptual likelihood $p(\mathbf{o}_t|\mathbf{l}_t)$ depends on the sensors used for the localization. This probability can be understood as the likelihood of observing a measurement $\mathbf{o}_t$ at location p_t. Works like [3, 1, 2] use Gaussian Processes (GPs) to model location-signal strength mappings and compute the perceptual likelihood by comparing incoming signal strength measurements with the predictions generated by the signal strength maps. Figure 1 shows an example of the mappings used - for the GP formulation it is required to generate a mapping of the mean expected signal strength and one of its expected variance.

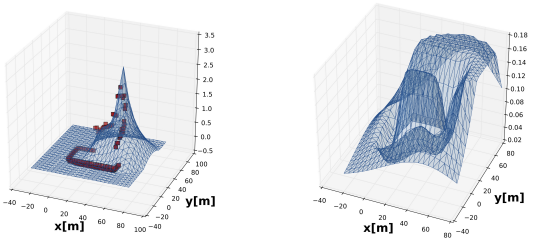

Fig. 1 Signal strength mappings. Figure shows (left) the predicted mean and (right) the predicted variance maps.

2 Dual MCL using signal strength maps

Our proposed dual MCL first samples its perceptual likelihood and projects each sample into a possible predecessor. For each sample it calculates its importance factor as the likelihood of the predecessor to have belong to the previous Bel state. Using the predecessors set and its importance factors, it uses importance sampling to generate a new predecessors sample set. The new predecessors sample is projected back, creating the new sample set. For enhancing the performance of the dual MCL against the *kidnapped robot problem*, some random samples obtained directly from the proposal distribution are finally added to the set, which is used to generate an updated Bel state using kernel density estimation (kde). The pseudo code of the algorithm is shown at alg. 1.

2.1 Implementation details

Perceptual model

The specific implementation of the perceptual model used for testing in this work is the one described at [1]. However, all results can be extrapolated to any method using signal strength mappings.

Motion model

Motion model used was the odometry model, which description can be found in detail at [4]. This model employs odometry information to predict robot's motion, and is widely used as most modern commercial robots make odometry information available in periodic time intervals or by request.

Sampling method

Sampling methods are employed to obtain a set of samples x drawn independently from an arbitrary distribution $f(\cdot)$.

*This work was funded by Tough Robotics Challenge, ImPACT Program of the Council for Science, Technology and Innovation (Cabinet Office, Government of Japan).

Algorithm 1 dual_MCL: Dual MCL algorithm

Input: GP (perceptual model), m (motion model), s (number of samples), $\mathbf{z}$ (stream of new measurements), $\mathbf{a}$ (odometry information)
$Bel := uniform$ ▷ Initialize the Bel state as the uniform distribution
$i := \{1, ..., s\}$ ▷ Initialize number of samples
while True **do**
  $\mathbf{x}_t^{(i)} \sim GP(\mathbf{z})$ ▷ Sample from the perceptual model
  $\mathbf{x}_{t-1}^{(i)} := m(\mathbf{x}_t^{(i)}, -\mathbf{a})$ ▷ Project samples back
  $w^{(i)} = Bel(\mathbf{x}_{t-1}^{(i)})$ ▷ Calculate importance factors
  $\mathbf{x}_{t-1}^{(i)} \sim \{\mathbf{x}_{t-1}^{(i)}, w^{(i)}\}$ ▷ Importance sampling
  $\mathbf{x}_t^{(i)} := m(\mathbf{x}_{t-1}^{(i)}, \mathbf{a})$ ▷ Project samples forward
  $\mathbf{x}2_t^{(i)} \sim RSS\_model(\mathbf{z})$ ▷ Add random samples
  $Bel := kde(\mathbf{x}_t^{(i)}, \mathbf{x}2_t^{(i)})$ ▷ Generate updated Bel state using kde

As it is possible to evaluate the function $f(\cdot)$ (our perceptual likelihood) the accept-reject method is used to sample from this distribution. In a nutshell, the accept-reject method draws a sample x at random from a known *envelope* function $g(\cdot)$, and if $f(x)$ is lower than $Mg(x)$ - with M being a positive constant, the algorithm accepts the sample. This process is repeated until all samples are obtained. M is used so the *envelope* function upper bounds $f(\cdot)$ at all points - i.e., $\forall x \quad Mg(x) \geq f(x)$. The idea behind the algorithm is that if samples are drawn from the envelope function, and all those which probability is higher than that of the target function's are rejected, what is left simulates samples taken from the target function itself. For our implementation an accept-rejection method with a uniform distribution as the envelope function is used.

Kernel density estimation method

Kernel density estimation (kde) is used to estimate the probability density functions of a random variable given a set of samples randomly taken from the function. It would be the logical inverse of sampling methods, in the sense that instead of obtaining random samples from a known distribution, it constructs an approximation of an unknown distribution based on random samples taken from it.

3 Tests and comparisons

Tests are performed using the same dataset described at [2]. The environment is a office-like scenario of 60m × 40m. The results obtained with the dual MCL are compared with an argmax and a MCL algorithm.

argmax

The argmax is the naive approach, it divides the map in a grids and calculates its likelihoods, then it outputs the location of the one with the highest probability as the estimated location. The main parameter that affects accuracy and computation time is the grid spacing used.

MCL

MCL is more commonly used in practice than the dual MCL. MCL starts by randomly generating samples and calculates its importance factor for re-sampling from the perceptual likelihood. As it does not need to sample directly from the perceptual likelihood it is faster than the MCL. However, its main weakness is that it relies heavily on its prior belief, which may yield to estimation errors if its belief places high confidence in a wrong estimation. Same as for the dual MCL the main parameter that affects the accuracy and the required computation time is the number of particles used.

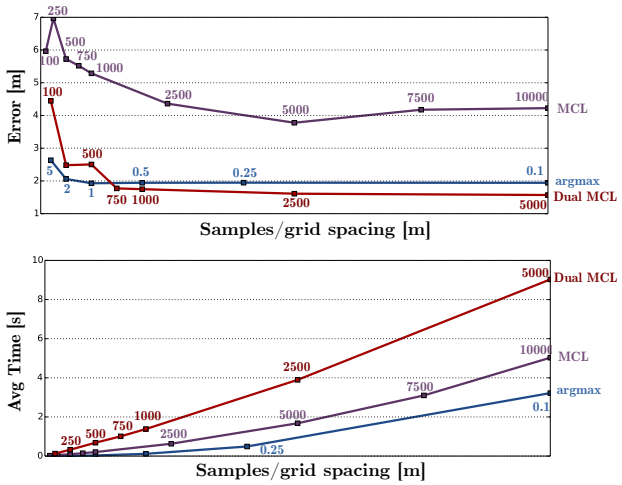

Fig. 2 (upper) Localization accuracy and (lower) required computational time of each algorithm.

4 Conclusions and Future works

From the tests performed, unexpectedly, the usage of MCL for this particular problem yielded the worst accuracy, even compared to the argmax algorithm; while the best accuracy was obtained using a dual MCL with more than 750 particles - the argmax accuracy for a grid spacing of 1m or less has an accuracy of 2m, while the dual MCL with 750 has an accuracy of 1.7m and with 5000 1.5m. However, the dual MCL loses to both other methods in computation time. From further testing we found that the main speed limitation is the accept-reject algorithm. While this method was chosen as it guarantees independence of samples, because $Mg(\cdot)$ is often not close to $f(\cdot)$ for signal strength mappings, many samples are rejected, slowing down the algorithm. It remains as future work to improve this sampling stage. Nonetheless, even with this drawback, the use of dual MCL is recommended.

References

[1] Brian Ferris, Dirk Haehnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *In proc. of robotics science and systems*, 2006.

[2] Renato Miyagusuku, Atsushi Yamashita, and Hajime Asama. Gaussian Processes with input-dependent noise variance for wireless signal strength-based localization. In *Proc. of the 13th IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR2015)*, 2015.

[3] Anton Schwaighofer, Marian Grigoras, Volker Tresp, and Clemens Hoffmann. GPPS: A Gaussian Process Positioning System for Cellular Networks. In *NIPS*, 2003.

[4] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.