Comparing Bayesian and Montecarlo localization for a robot with local vision

María A. Crespo, José M. Cañas, Vicente Matellán

Universidad Rey Juan Carlos,
Móstoles, Spain
{mangeles,jmplaza,vmo}@gsyc.escet.urjc.es

Abstract. Position estimation is one of the classic problems in mobile robotics. The goal of this paper is to compare two probabilistic localization methods based on local vision for a mobile robot. The experimental set up is based on the Aibo league of the RoboCup, where the robotic dogs major sensor is the on-board camera. Two localization algorithms, Bayesian and Montecarlo (MCL), have been implemented and compared, and their behaviour studied in several situations using a simulator.

1 Localization

Robots have to know where in the map they are in order to perform any task involving navigation. Even in highly dynamical environments, like the RoboCup competition, the robot behaviour depends on its position in the playground.

One approach to the localization problem is to rely on explicit position information provided by sensors. Other strategies require environmental engineering such as the placement of active beacons, passive marks, etc... Aproaches such as *Kalman* filtering [Welch02] and *fuzzy logic* [Buschka00], integrate the information obtained from sensors non directly related to position.

Probabilistic algorithms have proved very successful [Thrun00a] in many robotic environments. They calculate the probability of each possible position given some sensor readings and movement data provided by the robot. They cope with uncertainty and sensor errors and can recover from major localization issues. Their main drawback is the computational cost of keeping the history of the probabilities for all the possible locations in the map. Because of this limitation, several sampling techniques have been probed, keeping the power of the Bayes reasoning. In particular, MCL techniques, where a small number of representative samples are randomly selected and continuously updated, have gained popularity recently [Thrun01,Montemerlo02].

2 Environmental set up

We developed a *simulator* which emulates a robot walk through a RoboCup like scenario, as the one showed in figure 1. The environment is provided to the simulator through a map file which describes the white lines of the field and the position of the beacons and their colours. It can be easily changed.

The simulator generates a log file with the record of the images perceived and the encoders data stored while walking. Such information is used by the localization algorithms to estimate the robot position. This way both algorithms use exactly the same data collection. It adds actuation noise to the commands ordered to the motors and sensor errors to the perceived images. For the motors it adds a Gaussian noise both in translation and rotation. For the camera it adds false positive and negative beacons (mutation error), and a displacement in beacon position inside the images (offset error).

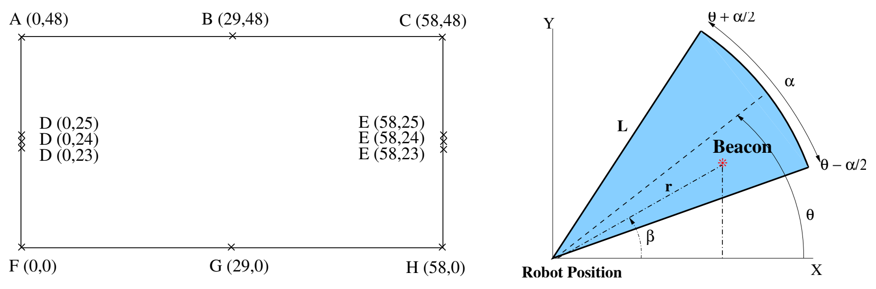

Fig. 1. Sample map provided to the simulator (left) and camera model used (right)

3 Experiments

Bayesian localization was implemented following the formulation developed at [Margaritis98]. Figure 2 represents a typical run using the RoboCup map shown in the left part of figure 1 (real dimension $290cmx240cm$). The map was tessellated in $5cmx5cmx1^\circ$ cells. We held the 58x48x360=1,002,240 possible poses where the robot could be located in what we called *probability cubes*. For each possible 3D cell the algorithm stores and updates its likelihood given the set of observations and motor actions.

Such likelihood is shown on figure 2 in a greyscale, the darker the cell, the lower its probability. The top left slot represents the initial estimation. All the cells have the same likelihood. After the first image is obtained, positions compatible with such observation rise in probability. As can be seen in the top right slot, one cone includes all the compatible cells. A robot forward movement of 10 cells causes the corresponding displacement of the robot evidence, as can be noted in the second row of figure 2. After the 45° turn, shown in the fourth row, the robot sees the C-beacon. This observation lets the robot discriminate the most likely cells once fused with the prior evidence, as displayed in the bottom right figure.

Typically, the Bayesian algorithm locates the robot in 2-4 iterations, which takes 24-48 seconds long in a Pentium-III at 1.1 GHz. Bayesian localization

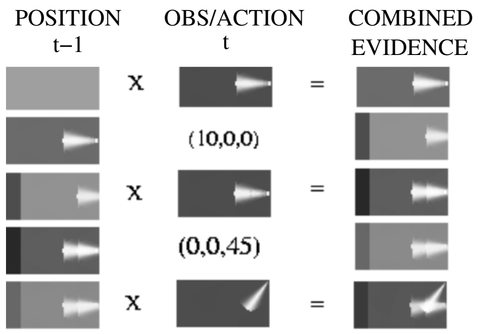

Fig. 2. Probability accumulation using Bayes's rule

has proved to be very robust to actuation and sensor errors. Combining all noises, which resembles the real scenario, it delivers localization errors smaller than 2.5cm in x, y and $5°$ in orientation. In addition, multiple experiments were carried out in many scenarios studying the effect of different parameters in the performance. It has proved robustness in all the tests performed.

Montecarlo localization was also implemented. Instead of computing the probability of all plausible poses, a population of samples evolves in time as new images are collected. The samples are relocated using the information from the last image and they converge to the real location after some iterations as can be seen in the typical run in the left picture on figure 3.

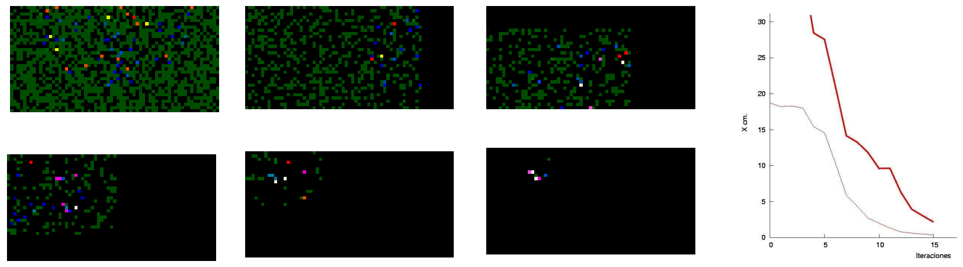

Fig. 3. Samples evolution using Montecarlo localization (left) and typical deviation of samples in x (right)

Typically, 13-16 iterations are required to locate the robot. Nevertheless, those iterations are much faster than the Bayesian ones, as they take only 4 seconds long in the same machine. It provides real location of the robot with $10cm$ maximum deviation, and 0.12 in $\cos\theta$. Regarding noise, Montecarlo localization has proved to be more sensitive to actuation and sensor errors than the Bayesian algorithm. The bigger the noise, the further from real position is the mean value in the final sample set, and fewer samples fall close to it.

To study how well the Montecarlo algorithm scales up, we tested it over a bigger map, a $580cm x 480cm$. Initial deviations are naturally bigger as the samples are spread over a larger area. After the same number of iterations the localization error was bigger than the one obtained in the regular map in absolute terms, but the ratio error/map size kept constant (right picture on Fig. 3).

Fine parameter tuning is a must for MCL. A key parameter is the movement error introduced to shift a little bit the samples after every ideal motor command. The bigger this random movement, the more stable the algorithm is, although worse resolution is achieved. Several tests were carried out to study the effect of other parameters in the performance. The algorithm is very sensitive to the probabilistic sensor model. In contrast, it is very robust to the number of beacons and samples once a minimum of them is provided.

4 Conclusions

The aim of this work was to test the Bayesian and the Montecarlo methods before their implementation on real robots endowed with camera.

The Bayesian algorithm has proved to be very robust to model parameters, motor and sensor errors, which makes it suitable to cope with uncertainty in real sensors and actuators. Despite its good resolution it doesn't scale up to larger environments because it computes the probability of *all* plausible locations. Such processing greediness prohibits its implementation on board a regular robot.

Compared to the Bayesian approach, the Montecarlo algorithm speeds up the localization process, making it easier to implement on board the robot and for bigger environments. In addition, this method doesn't require the tessellation of the space, and so it potentially offers higher resolution than Bayesian localization, when its parameters are tuned properly. Its main drawback is the sensitivity to sensor and motor errors and to its own parameters.

We are working in implementing the MCL algorithm on board an Aibo robot and testing it with other "sensors" like the wireless network card.

References

[Buschka00] Buschka, P., Saffiotti, A., Wasik, Z.:*Fuzzy Landmark-Based Localization for a Legged Robot.* Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) Japan, 2000. pp 1205-1210.

[Margaritis98] Margaritis, D., Thrun, S.:*Learning to Locate an Object in 3D Space from a sequence of images.* Proc. Int. Conf. on Machine Learning (1998) 332–340.

[Montemerlo02] Montemerlo, M., Thrun, S., Whittaker, W.:*Conditional Particle Filters for Simultaneous Mobile Robot Localization and People-Tracking.* IEEE Int. Conf. on Robotics and Automation (ICRA) 2002.

[Thrun00a] Thrun, S.:*Probabilistic Algorithms in Robotics.* AI Magazine, 21(4):93-109, April 2000.

[Thrun01] Thrun, S., Fox, D., Burgard, W., Dellaert, F.: *Robust Montecarlo Localization for Mobile Robots.* Artificial Intelligence Journal, 2001.

[Welch02] Welch, G., Bishop, G.: *An Introduction to the Kalman Filter.* UNC-Chapel Hill, TR 95-041, March 11, 2002.