Localization with a low-cost robot*

Stanislav Slušný, Roman Neruda, and Petra Vidnerová

Institute of Computer Science, Academy of Sciences, Czech Republic
slusny@cs.cas.cz

Abstract. *The robot localization problem is a fundamental and well studied problem in robotics research. Algorithms used to estimate pose on the map are usually based on Kalman or particle filters. These algorithms are able to cope with errors, that arise due to inaccuracy of robot sensors and effectors. The performance of the localization algorithm depends heavily on their quality.*
This work shows performance of localization algorithm based on particle filter with small miniature low-cost E-puck robot. Information from VGA camera and eight infrared sensors are used to correct estimation of the robot's pose.

1 Introduction

The robot localization problem is a fundamental and well studied problem in robotics research. Several algorithms are used to estimate pose on the known map and cope with errors, that arise due to inaccuracy of robot sensors and effectors. Their performance depends heavily on quality of robot's equipment: the more precise (and usually more expensive) sensors, the better results of localization procedure.

This work deals with localization algorithm based on particle filter with small miniature low-cost E-puck robot. Information from cheap VGA camera and eight infrared sensors are used to correct estimation of the robot's pose. To achieve better results, several landmarks are put into the environment. We assume, that robot knows the map of the environment in advance (distribution of obstacles and walls in the environment and position of the landmarks). We do not consider the more difficult simultaneous localization and mapping (SLAM) problem in this work (the case, when robot does not know it's own position in advance and does not have the map of the environment available).

E-puck is a widely used robot for scientific and educational purposes - it is open-source and low-cost. Despite its cheapness and limited sensor system, localization can be successfully implemented, as will be shown in this article. We are not aware of any published results of localization algorithms with E-puck robot. The survey of localization methods can be found in [1].

* This work was supported by GA ČR grant 201/08/1744, Institutional Research Plan AV0Z10300504 and by the Ministry of Education of Czech Republic under the project Center of Applied Cybernetics No. 1M684077004 (1M0567).

Fig. 1. Miniature e-puck robot has eight infrared sensors and two motors.

2 Introducing E-puck robot

E-puck ([2, 3], Figure 1) is a mobile robot with a diameter of 70 mm and a weight of 50 g. The robot is supported by two lateral wheels that can rotate in both directions and two rigid pivots in the front and in the back. The sensory system employs eight "active infrared light" sensors distributed around the body, six on one side and two on other side. In "passive mode", they measure the amount of infrared light in the environment, which is roughly proportional to the amount of visible light. In "active mode" these sensors emit a ray of infrared light and measure the amount of reflected light. The closer they are to a surface (the e-puck sensors can detect a white paper at a maximum distance of approximately 8 cm), the higher is the amount of infrared light measured. Unfortunately, because of their imprecision and characteristics (see Figure 2), they can be used as bumpers only. As can be seen, they provide high resolution only within few millimeters. They are very sensitive to the obstacle surface, as well. Besides infrared sensors, robot is equipped with low-cost VGA camera. The camera and image processing will be described in the following section.

Two stepper motors support the movement of the robot. A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. It can divide a full rotation into a 1000 steps, the maximum speed corresponds to about a rotation every second.

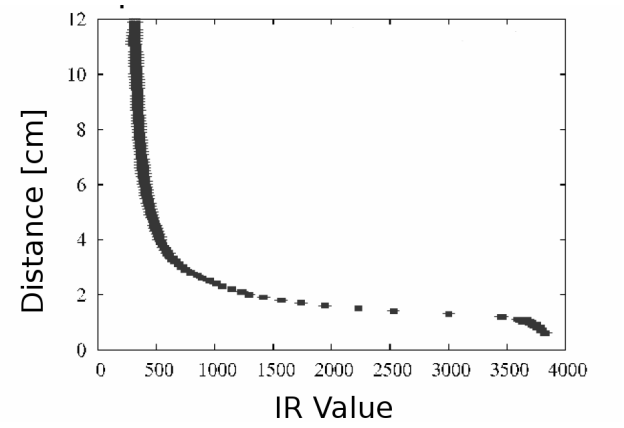

Fig. 2. Multiple measurements of front sensor. E-puck was placed in front of the wall at a given distance and average IR sensor value from 10 measurements was drown into the graph.

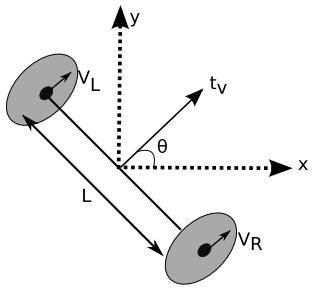

Fig. 3. Differential drive robot schema.

3 Dead reckoning

Dead reckoning ([4], derived originally from deduced reckoning) is the process of estimating robot's current position based upon a previously determined position. For shorter trajectories, position can be estimated using shaft encoders and precise stepper motors.

E-puck is equipped with a differential drive (Figure 3) - a simplest method to control robot. For a differential drive robot the position of the robot can be estimated by looking at the difference in the encoder values Δs_R and Δs_L. By estimating the position of the robot, we mean the computation of tuple x, y, Θ as a function of previous position $(x_{OLD}, y_{OLD}, \Theta_{OLD})$ and encoder values (Δs_R and Δs_L).

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} x_{OLD} \\ y_{OLD} \\ \theta_{OLD} \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{pmatrix} \quad (1)$$

$$\Delta\theta = \frac{\Delta s_R - \Delta s_L}{L} \quad (2)$$

$$\Delta s = \frac{\Delta s_R + \Delta s_L}{2} \quad (3)$$

Parameters	Value
Maximum translational velocity	12.8 cm / sec
Maximum rotational velocity	4.86 rad / sec
Stepper motor maximum speed	+- 1000 steps / sec
Distance between tires	5.3 cm

Table 1. Velocity parameters of E-puck mobile robot.

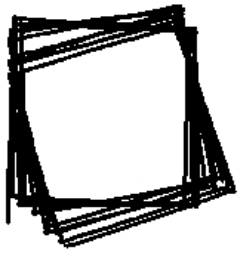

Fig. 4. Illustration of error accumulation. Robot was ordered to make 10 squares of size 30 cm. Odometry errors are caused mostly by rotation movement.

$$\Delta x = \Delta s.cos(\theta + \frac{\Delta\theta}{2}) \quad (4)$$

$$\Delta y = \Delta s.sin(\theta + \frac{\Delta\theta}{2}) \quad (5)$$

The major drawback of this procedure is error accumulation. At each step (each time you take an encoder measurement), the position update will involve some error. This error accumulates over time and therefore renders accurate tracking over large distances impossible (see Figure 4). Tiny differences in wheel diameter will result in important errors after a few meters, if they are not properly taken into account.

4 Image processing

The robot has a low-cost VGA camera with resolution of 480x640 pixels. Unfortunately, the Bluetooth connection supports only a transmission of 2028 colored pixel. For this reason a resolution of 52x39 pixels maximizes the Bluetooth connection and keeps a 4:3 ratio. This is the resolution we have used in our experiments (see Figure 4). Another drawback of the camera is that it is very sensitive to the light conditions.

Despite these limitations, camera can be used to detect objects or landmarks. However, the information about distance to the landmark extracted from the

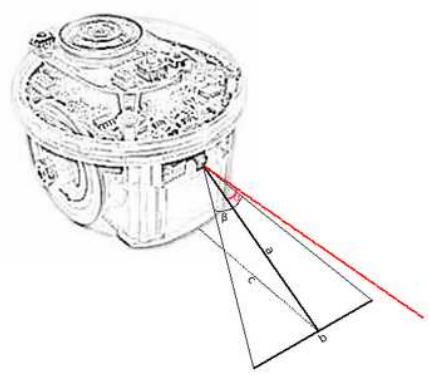

Fig. 5. The physical parameters of the real camera (picture taken from [2]). Camera settings used in experiments corresponds to parameters $a = 6$ cm, $b = 4.5$ cm, $c = 5.5$ cm, $\alpha = 0.47$ rad, $\beta = 0.7$ rad.

camera is not reliable (due to the noise), and we do not use it in following section.

Landmarks are objects of rectangular shape of size 5x5 cm and three different colors - red, green and blue. We implemented image processing subsystem, that detects relative position of the landmark from the robot. Following steps are included:

- Gaussian filter is used to reduce camera noise ([5])
- Color segmentation into the red, blue and green color. ([6])
- Blob detection is used to detect position and size of the objects on the image. ([7])
- Object detection is used to remove objects from image, that have non-rectangular shape.

Output from the image processing is the relative position and color of the detected landmarks (for example - I see red landmark by angle 15 degrees).

5 Particle filter localization

As shown previously (Figure 4), pose estimation based on dead reckoning is possible for short distances only. For longer trajectories, more clever methods are needed. These methods are based either on Kalman filter [8] (or some of its variants) or particle filter (PF) [9].

The PF possesses three basic steps - state prediction, observation integration and resampling. It works with quantity $p(x_t)$ - the probability, that robots is located at the position x_t in time t. In the case of PF, the probability distribution is represented by the set of particles. Such a representation is approximate, but can represent much broader space of distributions that, for example, Gaussians, as it is nonparametric.

Each particle $x_t^{[m]}$ is a hypothesis, where the robot can be at time t. We have used M particles in our

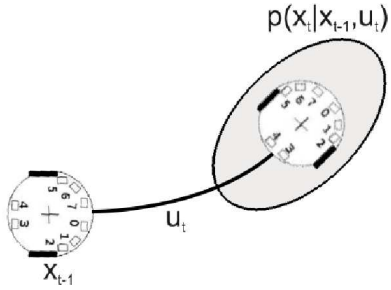

Fig. 6. First step in PF algorithm - to each position hypothesis x_{t-1} is applied odometry model based on movement u_{t-1} and new hypothesis x_t is sampled from distribution $p(x_t|x_{t-1}, u_{t-1})$.

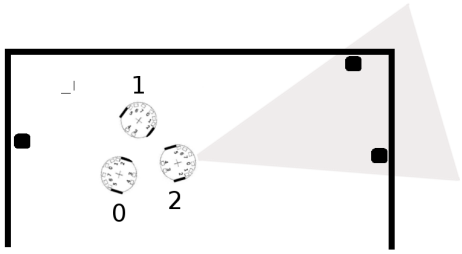

Fig. 7. Second step in PF algorithm - each particle is assigned a importance factor, corresponding to the probability of observation z_t. If image processing detects two landmarks on the actual camera image, particles 0 and 1 will be assigned small weight.

experiment. The input of the algorithm is the set of particles X_t, most recent control command u_t and the most recent sensor measurements z_t.

1. State prediction based on odometry.
   The first step is the computation of temporary particle set X from X_t. It is created by applying odometry model $p(x_t|u_t, x_{t-1})$ to each particle $x_t^{[m]}$ from X_t.
2. Correction step - Observation integration
   The next step is the computation of *importance factor* $w_t^{[m]}$. It is the probability of the measurement z_t under particle $x_t^{[m]}$, given by $w_t^{[m]} = p(z_t|x_t^{[m]})$.
   Two types of measurements were considered:
   - Measurement coming from distance sensors
     Distance sensor (one averaged value for front, left, right and back direction) were used as bumpers only. In case of any contradiction between real state and hypothesis, importance factor was decreased correspondingly.
   - Measurement obtained from image processing

Fig. 8. Placement of red, green and blue landmarks in rectangular arena of size 1x0.75m.

Output from image processing was compared with expected position of the landmarks. In case of any contradiction (colors and relative angle of landmarks were checked), importance factor was decreased. The bigger mismatch, the smaller importance factor was assigned to the hypothesis.

3. Re-sampling
   The last step incorporates so-called *importance sampling*. The algorithm draws with replacement M particles from temporary set X and creates new particle set X_{t+1}. The probability of drawing each particles is given by its importance weight. This principle is called *survival of the fittest* in AI ([10]).

6 Experiments

Experiments were carried out in an arena of size 1x0.75 meters. Three landmarks (red, blue and green, one of each color) were placed into the arena, as shown on the figure 8. Robot was controlled by commands sent from computer, values from sensors were sent back to computer by using Bluetooth. Execution of each command took 64 milliseconds.

The experiment started by putting robot into the arena and randomly distributing 2000 particles. After several steps, the PF algorithm relocated the particles into real location of the robot. The robot was able to localize itself. The convergence of the algorithm depends on the fact, if robot is moving near the wall or in the middle of the arena. The impact of infrared sensors was obvious. Algorithms were verified in the simulator([11]) and in reality, as well. The video demonstration can be found at ([12]).

The localization algorithm was able to cope with even bigger areas, up to the size of three meters. However, we had to add more landmarks to simplify the localization process. Localization algorithm showed satisfiable performance, relocating hypothesis near real robot pose.

7 Conclusions

Localization and pose estimation is an opening gate towards more sophisticated robotics experiments. As we have shown, the localization process can be carried out even with low-cost robot. Experiments were executed both in simulation and real environment.

A lot of work remains to be done. The experiments in this work considered static environment only. Addition of another robot will make the problem much more difficult.

As we have mentioned already, there are certain areas in the environment, where convergence of the localization algorithm is very fast - in corners or near walls. *Sensor fusion* is the process of combining sensory data from disparate sources such that the resulting information is in some sense better than would be possible when these sources were used individually. We are dealing with sensor fusion of infrared sensors and input from camera.

As a future work, we would like to implement path planning, that takes into account performance of the localization algorithm. Suggested path (generated by path planning algorithm) should be safe (the chance to get lost should be small) and short. Multi-criterial path planning will be based on dynamic programming ([13]). The idea is to learn areas with high loss probability from experience.

References

1. S. Thrun, W. Burgard, and D. Fox: *Probabilistic Robotics.* Cambridge, MA: MIT Press, 2005.
2. `http://en.wikibooks.org/wiki/Cyberbotics_Robot_Curriculum/`
3. E-puck, online documentation. `http://www.e-puck.org`
4. R.C. Arking: *Behavior-Based Robotics.* The MIT Press, 1998.
5. L.G. Shapiro, G.C. Stockman: *Computer Vision.* Prentence Hall, 150, 2001, p. 137.
6. J. Bruce, T. Balch and M. Veloso: *Fast and Inexpensive Color Image Segmentation for Interactive Robots.* In Proceedings of IROS-2000, 2000, 2061–2066.
7. `http://www.v3ga.net/processing/BlobDetection/index-page-home.html`
8. R.E. Kalman: *A new approach to linear filtering and prediction problems.* Trans. ASME, Journal of Basic Engineering 82, 35–45.
9. R.Y. Rubinstein: *Simulation and the Monte Carlo Method..* John Wiley and Sons, Inc.
10. K. Kanazawa, D. Koller, and S.J. Russel: *Stochastic simulation algorithms for dynamic probabilistic networks.* In Proceedings of the 11th Annual Conference on Uncertainty in AI, Montreal, Canada.
11. Webots simulator. `http://www.cyberbotics.com`.
12. Video demonstration. `http://www.cs.cas.cz/slusny`.
13. R.S. Sutto, and A. Barto: *Reinforcement Learning: An Introduction.* The MIT Press, 1998.