Markov Localization in the CiberRato simulation environment

João Santos, 76912[1 [0000−0003−0393−6456]]

University of Aveiro, Portugal

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: Peception · control · Markov · localization · Bayes'.

1 Introduction

1.1 Environment

1.2 Assumptions

A set of assumptions have been made in order to simplify the implementation. This set is covers not only the specificities of the maps used but also the allowed/expected motions of the robot. They are:

- the maps are 7 rows by 14 columns
- the unit is the diameter of the robot
- cells are squares with side of 2 diameter
- inner walls have a thickness of 0.2 diameter
- walls on the border of tha map are 0 thickness
- execution ends when the robot hits a wall
- motion noise does not exist
- sensor noise is gaussian with mean 0 and variance 0.1
- the robot motion is deterministic
- the robot only moves from left to right
- the Infra Red (IR) sensors detect the closest obstacle measured in the normal direction
- the IR sensors have a symmetric Field of View (FOV) of 60 degrees

1.3 Bayes' Filter

Given that the map is know to the agent and that the state space is finite we can, in a very direct way, implement the Bayes' filter to, iteratively, compute the the agent localization, or more precisely, to compute the probability of the agent being on any given cell.

Algorithm 1 shows the general implementation of the Bayes' filter, as stated on [1]. In it, $bel(x_t)$ represents the belief at time t, u_t and z_t are, respectively, the controls inputted and the measurements taken also at time t and η is a normalization factor.

Algorithm 1: Bayes' filter general algorithm.

Input: $bel(x_{t-1}), u_t, z_t$
Output: $bel(x_t)$
foreach x_t **do**
  $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})bel(x_{t-1})dx$
  $bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$

Note that for updating the belief at any given instant, only the prior estimate is need, which comes from the Markov assumption.

Motion Update On the scope of this assignment, since the agent is on a finite state space (it either moves or not) and can only move right (motion model) we can say that

$$\begin{aligned}\overline{bel}(x_t) &= \int p(x_t|u_t, x_{t-1})bel(x_{t-1})dx \\ &= \sum_x p(x_t|u_1, x)bel(x) \\ &= p(x_t|u_t, x_{t-1})bel(x_{t-1}) + p(x_t|u_1, x_t)bel(x_t)\end{aligned}$$

where $p(x_t|u_t, x_{t-1})$ and $p(x_t|u_1, x_t)$ denote, respectively, the probabilities of the agent has moved right and has stayed stationary, given the control vector u_t.

Carried for this assignment, it simply becomes that if the agent has stayed stationary ($u_t = [0, 0]$) then $p(x_t|u_t, x_{t-1}) = 0$ and $p(x_t|u_1, x_t) = 1$, and vice versa if the agent has moved ($u_t = [1, 0]$).

Measuremnet Integration It is in this step were the data gathered from all sensors is fused. Given that any given measure from each of the sensors follows the $\sim \mathcal{N}(true\_value, 0.1)$ model, we can compute $bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$ from algorithm 1 with algorithm 2, where $gt(x)$ represents the ground truth values (measurements expected to be gathered in each cell) for a given sensor in cell x. This ground truth was computed a priori since the map is known to the agent.

Algorithm 2: Measurement integration algorithm.

Input: $bel(x_{t-1}), z_t, gt$
Output: $bel(x_t)$
foreach x_t **do**
  $product \leftarrow 1$
  $\eta \leftarrow 0$
  foreach *sensor* **do**
    $product = product \times \mathcal{N}(gt(x) - z_t, 0.1)$
  $\overline{bel}(x_t) = product \times bel(x_{t-1})$
  $\eta = \eta + \overline{bel}(x_t)$
foreach x_t **do**
  $bel(x_t) = \eta^{-1}\overline{bel}(x_t)$

2 Implementation

As stated on section 1.2, the robot only moves in a straight line, from left to right. This poses several advantages, the first of which is that the walls are always perpendicular or parallel to the robot's motion direction. To take advantage of this, the author choose to set the robot's four IR distance sensors perpendicular to the cell's border, i. e., to 0, 90, -90 and 180 degrees measured from the robot's travelling direction.

2.1 Debug Tools

3 Results

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics. MIT Press, Cambridge, Mass. (2005)