Autonomous Robotic Systems
Bayes Filter

Christoph Emunds (i6146758)
Dominik Nerger (i6146759)

March 19, 2017

Contents

1 Introduction

The Bayes filter, in the context of mobile robot localization also called *Recursive Bayesian Estimation*, uses sensor measurements and a mathematical process model to estimate an unknown probability density function recursively over time. This document describes the implementation of a Bayes filter in the Robot Operating System (ROS) framework and reflects on the filter's performance with regard to its localization capabilities in different situations. The world in which the robot is situated consists of a corridor with several doors.

2 The world model

The robot can move between 10 grid cells in the world model. Since the robot is able to traverse through them in both directions, 20 states are available.

Therefore, the world model needs to store information about 20 states. It is necessary that through the sensing measurements it can be deduced which states match the measured state, because this needs to be used in the calculations for the sensing update.

The possible values for each state are represented through an enumeration. This enumeration contains the four values *L*,*LR*,*LRF* and *R*. The characters express whether a wall has been measured on the left *(L)*, right *(R)* or front *(F)* side, with a combination indicating that multiple walls have been found.

Therefore, the world model is represented through a vector with size 20 and information about which measurements should be true in this state contained in the respective enumeration value.

3 The Bayes filter

The Bayes Filter is initialized with a uniformly distributed belief state. This results in an initial probability of $p(x_i) = \frac{1}{20} = 0.05$ for each state, since there are 20 states.

3.1 Control update

The update step is executed whenever the robot moves forward or makes a 180 degree turn. To calculate the probability for being in a state x after a motion command, the probabilities of all possible starting states is weighted according to the given transition probabilities and summed up. For the *move forward* command, this equates to:

$$p(x_i) = 0.8 \times p(x_{i-1}) + 0.1 \times p(x_{i-2}) + 0.1 \times p(x_i)$$

The world model is not cyclic, therefore two additional calculations are necessary for certain states. States for which $i\%10 == 1$ holds true, the equation is:

$$p(x_i) = 0.8 \times p(x_{i-1}) + 0.1 \times p(x_i)$$

Furthermore, the equation for states for which $i\%10 == 0$ holds true is:

$$p(x_i) = 0.1 \times p(x_i)$$

This is necessary because, after a movement update, the states *0* and *10* can only be reached by being in the respective state already and the move failing. The same holds true for states *1* and *11*, which can be reached by correctly moving from states *0* or *10* or failing to move when they are in their state already.

For the *turn* command, the equation results in:

$$p(x_i) = 0.9 \times p(x_{N-1-i}) + 0.1 \times p(x_i)$$

3.2 Measurement update

The prior belief $bel(\bar{x}_t)$ before incorporating the measurement update is now multiplied by the probability that the measurement z_t has been taken. This is done for each possible posterior state. The resulting product is not a probability anymore and needs to be normalized again, such that the probabilities of the states sum to 1.

A measurement is taken, i.e. either there is a wall or there is none. Then, for each possible state it is checked whether the measurement agrees with the layout of the state, that is, if there is a wall when a wall has been detected by the sensors or not. Then, the probability for being in a state that is conform with the measurement is multiplied by a factor of 0.6, the probability for every other state is multiplied by 0.2.

During this first loop, the resulting products are also summed up. In a second loop, the calculated values are then divided by the sum, such that they represent proper probabilities again.

4 Results

There is practically no information that needs to be remembered to calculate the next belief state, so we can assume that the state is complete and that the Markov assumption holds.

Most states in the corridor have a wall on both sides, which makes them indistinguishable. When issuing the *measurement* command in a state that has a door, the probabilities of the three states containing a door raises. Multiple continuous measurements cause the robot's confidence for being in one of these three states to grow.

5 Conclusion