Budapest University of Technology and Economics
Dep. of Control for Transportation and Vehicle Systems

Domonkos Csuzdi

Students' Scientific Conference Report

Daum–Huang Filter for LiDAR-based Mobile Robot Localization

Consultant:

Olivér Törő

PhD candidate

Budapest, 2021

Acknowledgement

This work has been supported by the Hungarian Government and co-financed by the European Social Fund. (Contract Identifier: EFOP-3.6.3-VEKOP-16-2017-00001. Talent management in autonomous vehicle control technologies.) Their support is gratefully acknowledged.

1 Mobile Robot Localization

1.1 The Localization Task

During localization the main goal is to determine the coordinate transform between the local coordinate system of the robot, and a given global frame. For most of the problems the Global Navigation Satellite System (GNSS), like GPS provides this information. In an ideal world the GPS data is precise and reliant, making localization algorithms unnecessary. However, it is well known that global positioning by satellites cannot be performed in shielded environments (for example indoors), and even outdoors, the provided precision is often not sufficient.

To overcome these deficiencies, different state estimation algorithms are used to obtain the pose of the robot in the global frame. Localization and mapping often goes hand in hand: localization without a map (or some kind of a representation of the environment on which the global frame is defined), and map creation without the information about the pose is hardly possible. If one of them is assumed to be known, the task is much easier: localization, or mapping. If both are sought after, the Simultaneous Localization and Mapping (SLAM) problem arises, which is significantly harder the any of them separately. In the scope of this report, only the localization task is addressed on a given (ground truth) map.

One possible way to achieve localization is the introduction of pose hypotheses. By this the robot's belief of its pose is a probability distribution, instead of a crisp value [1]. Knowing the pose exactly is not feasible in a noisy real world environment. For a localization task, two main hardware components are used: an effector which is responsible for moving the robot, and an exteroceptive sensor, which is responsible to obtain information about the surrounding environment (like a LiDAR, a sonar, or often a camera). Both introduce errors and noise to the system which could be dealt with by the application of the probabilistic approach. In the following, this modelling method is detailed.

Almost every state estimation (e.g. localization) algorithm is based on Bayesian filtering. It servers as a foundation for these methods, and could not be implemented on its own. A Bayes filter conducts of two main parts which are iterated over time: prediction and update (see more in [1]). These have the following forms (respectively):

$$\overline{bel}(\mathbf{x}_t) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) bel(\mathbf{x}_{t-1}) \mathrm{d}\mathbf{x}_{t-1}, \tag{1.1}$$

$$bel(\mathbf{x}_t) = \eta p(\mathbf{z}_t|\mathbf{x}_t) \overline{bel}(\mathbf{x}_t), \tag{1.2}$$

where

$$bel(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}), \tag{1.3}$$

$$\overline{bel}(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}). \tag{1.4}$$

The notations are the following: $\mathbf{x}_t$ is the pose (position in 2D, and heading direction) at time t, $\mathbf{u}_t$ is the input vector at t, η is a normalization constant from Bayes' theorem,

$\mathbf{z}_t$ is the measurement at t, $(.)_{1:t}$ denotes values from time $t = 1$ to t, $bel(\mathbf{x}_t)$ is the belief (also called as posterior), and $\overline{bel}(\mathbf{x}_t)$ is the predicted belief.

First, $\overline{bel}(\mathbf{x}_t)$ is calculated from the prior $bel(\mathbf{x}_{t-1})$, using the motion model (first part in the integral). This is a prediction, because only the kinematics are incorporated, not the measurements. Then in the update part the measurement model is considered. This corrigates (updates) the prediction by incorporating the observations.

However, these probability distributions and integrals cannot be calculated on their own. Each distinct filter realization addresses the solution of the Bayesian recursion differently: the Kalman Filters use Gaussian distributions and their parametric description to estimate the pose hypotheses, while particle filters produce a more general numerical solution by describing an arbitrary distribution via particles. The Daum–Huang filters also use particles, but describes their movement with the help of the Fokker–Planck equation. These realizations are explained in greater detail in Section ****.

1.2 The Mobile Robot

An important preliminary of localization is the introduction of the utilized hardware, the operating environment, and their models. In this subsection, the mobile robot itself is discussed, along with its relevant sensors, followed by the environment representation where the robot has to be localized.

As an agent, the simulated version of ROBOTIS' TurtleBot3 is used via Robot Operating System (ROS) and Gazebo. This two-wheeled platform is widely used for educational and prototyping purposes due to its easy handling and well developed simulational counterpart. Although it has many useful components, here only the LiDAR and the differential drive are discussed due to their relevancy in the localization task. Those and the robot itself can be seen in Figure 1. The mounted 2D LiDAR provides range and angle meas-

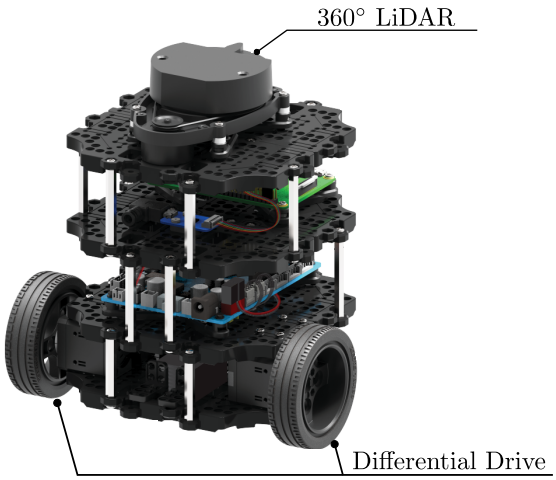

Fig. 1: ROBOTIS' TurtleBot3 Burger platform, with a mounted 360° LiDAR on top, and a differential drive (image source: www.robosklep.com).

urements from the environment with 360° field of view. This particular model (LDS-01) has an angular resolution of 1°, detection range of $0.12 - 3.5$ m-s, and accuracy (3σ) of ± 15 mm-s (actually the precision is distance dependent, but this effect is not considered). Invalid readings indicate out of range measurements. One full measurement is shown in Figure 2.

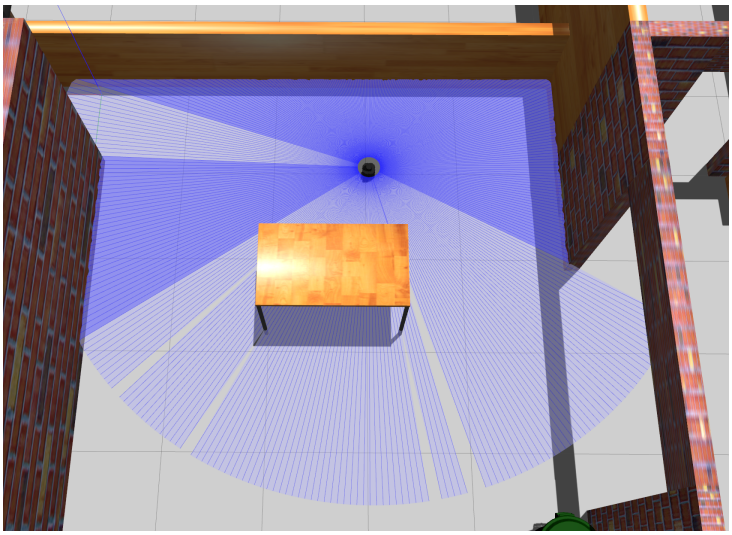

Fig. 2: Visualized 2D LiDAR measurement of the TurtleBot3 in a house environment, using the Gazebo simulator.

The platform has 2 independently-driven wheels, and one free turning wheel, which makes it eligible to be modelled by differential drive kinematics. The odometry is conducted by (simulated) rotary encoder readings from the two wheels separately. Originally the out-of-the-box ROS-Gazebo model by ROBOTIS did not considered odometry noise, which had to be manually added in order to efficiently model real world conditions. Precisely modelling odometry error is a difficult task, and even nowadays is an actively researched topic [2]. However, this degree of precision is not required here. Instead, both encoder readings are simply corrupted by a random variable each (ξ_L for the left wheel, ξ_R for the right), obtained as

$$\xi_L \sim \mathcal{N}(0, \alpha v_L^2), \tag{1.5}$$

$$\xi_R \sim \mathcal{N}(0, \alpha v_R^2), \tag{1.6}$$

where $\mathcal{N}(\mu, \sigma^2)$ stands for a normal distribution with mean μ and variance σ^2, α is a scaling parameter, and v_L, v_R are the corresponding velocities. Applying this small modification, ROS provides noisy odometry data at each tilmestep, which then can be used to establish the motion model of the robot (see Subsection 1.4).

1.3 The Map

The environment of the agent is represented by a 2D Occupancy Grid Map (OGM) [3]. This model describes the environment by dividing it to finitely many grid cells, where each

grid cell is a random binary variable, representing that whether it's occupied, or not. Upon creating the OGM, the occupancy value of each cell (the probability, that its occupied) is iteratively updated. To obtain a final map, these values are thresholded, producing values of 1 if the cell is *mostly* occupied, or 0 is its *mostly not*. The top-down view of the TurtleBot3 House by ROBOTIS and the corresponding OGM can be seen in Figure 3, which was obtained by the SLAM GMapping algorithm [4]. For a pure localization task, the map is considered as ground truth, therefore the previously introduced odometry noise was omitted during the mapping process.

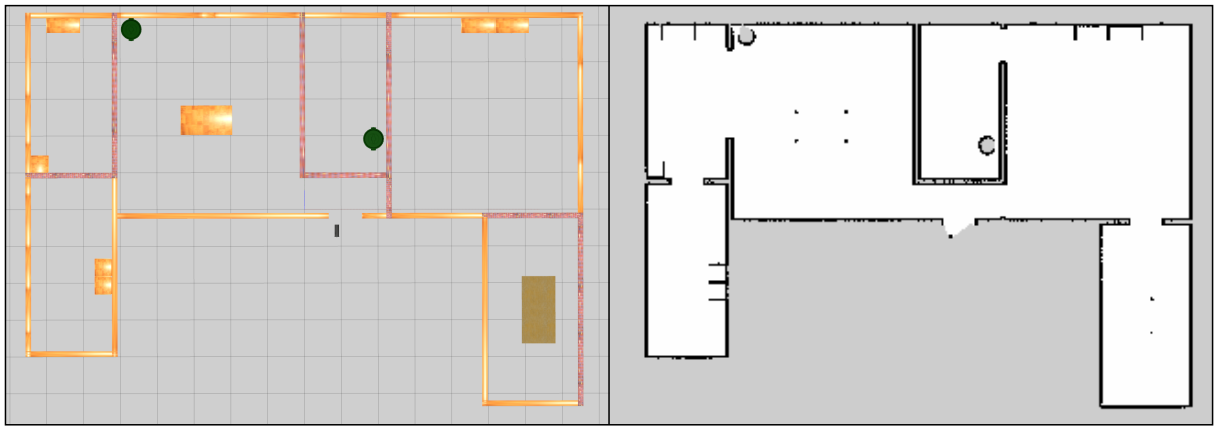

Fig. 3: The top-down view of the TurtleBot3 House in Gazebo, and its OGM representation (mind the open doors in the building). Grey pixels represent unknown area, black pixels are occupied, white pixels are free cells. 1 pixel (cell) in the OGM has a size of 0.05×0.05 m.

1.4 The Motion Model

Now that the hardware and the underlying localization task is introduced, the two main parts of the Bayesian recursion is going to be described in the following subsections: the motion model, and the measurement model.

The motion model is used to describe the probability distribution of $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$. Without any external information, the pose of the robot at time t can be estimated based on the previous pose at time $t-1$, and the control input at time t. Naturally, this estimation will be disrupted by tire slippage and drift. Due to the incremental nature, these errors are integrated over time, thus making the estimation more and more uncertain.

The motion model describes this transition using the kinematic model of the agent. Based on the inputs, two distinct probabilistic models can be established, introduced in [1]: the velocity motion model, and the odometry motion model. Here, only the latter is detailed.

If the odometry of the robot is available (i.e. by integrating wheel encoder measurements), they can be treated as a control input:

$$\mathbf{u}_t = (\overline{\mathbf{x}}_t \;\; \overline{\mathbf{x}}_{t-1})^\top = \left(\overline{x}_t \;\; \overline{y}_t \;\; \overline{\theta}_t \;\; \overline{x}_{t-1} \;\; \overline{y}_{t-1} \;\; \overline{\theta}_{t-1}\right)^\top. \tag{1.7}$$

The key is the fact that the relative difference between two consecutive odometry data is a good estimation of the relative difference between the two consecutive true poses, if the timestep is sufficiently small.

The transition between the state $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ is simplified as a sequence of a rotation, a translation, and another rotation. These are indicated in Figure 4. with δ_{rot1}, δ_{trans}, and δ_{rot2} respectively. It's important to mention that this separation to rotational and translational components is arbitrary (introduced by Thrun et al. in [1]); for another approach, see [5].

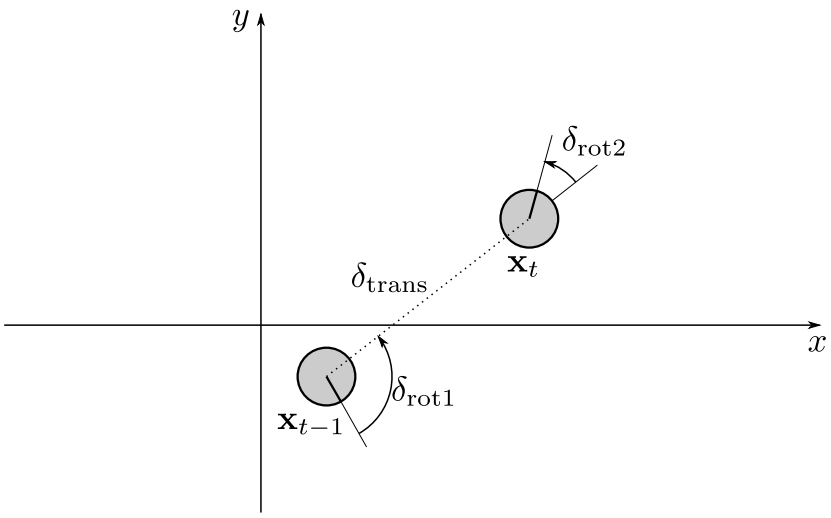

Fig. 4: The rotation-translation-rotation transition sequence from state $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$.

The control input $\mathbf{u}_t$ then transformed to the three transition components as:

$$\delta_{\text{rot1}} = \arctan2\left(\overline{y}_t - \overline{y}_{t-1}, \overline{x}_t - \overline{x}_{t-1}\right) - \overline{\theta}_{t-1}, \tag{1.8}$$

$$\delta_{\text{trans}} = \sqrt{\left(\overline{x}_{t-1} - \overline{x}_t\right)^2 + \left(\overline{y}_{t-1} - \overline{y}_t\right)^2}, \tag{1.9}$$

$$\delta_{\text{rot2}} = \overline{\theta}_t - \overline{\theta}_{t-1} - \delta_{\text{rot1}}. \tag{1.10}$$

To model odometry noise, the inputs are treated as random variables, formulated by

$$\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} + \xi_{\text{rot1}}, \qquad \xi_{\text{rot1}} \sim \mathcal{N}(0, \alpha_1\delta_{\text{rot1}}^2 + \alpha_2\delta_{\text{trans}}^2), \tag{1.11}$$

$$\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} + \xi_{\text{trans}}, \qquad \xi_{\text{trans}} \sim \mathcal{N}(0, \alpha_3\delta_{\text{trans}}^2 + \alpha_4(\delta_{\text{rot1}}^2 + \delta_{\text{rot2}}^2)), \tag{1.12}$$

$$\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} + \xi_{\text{rot2}}, \qquad \xi_{\text{rot2}} \sim \mathcal{N}(0, \alpha_1\delta_{\text{rot2}}^2 + \alpha_2\delta_{\text{trans}}^2), \tag{1.13}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are error parameters.

Then, using the control inputs and the previous state, samples from $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$ are obtained by,

$$x_t = x_{t-1} + \hat{\delta}_{\text{trans}}\cos(\theta + \hat{\delta}_{\text{rot1}}), \tag{1.14}$$

$$y_t = y_{t-1} + \hat{\delta}_{\text{trans}}\sin(\theta + \hat{\delta}_{\text{rot1}}), \tag{1.15}$$

$$\theta_t = \theta_{t-1} + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}, \tag{1.16}$$

$$\mathbf{x}_t = \begin{pmatrix} x_t & y_t & \theta_t \end{pmatrix}^\top. \tag{1.17}$$

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. [Online]. Available: http://www.amazon.de/gp/product/0262201623/102-8479661-9831324?v=glance&n=283155&n=507846&s=books&v=glance

[2] M. Fazekas, P. Gáspár, and B. Németh, "Calibration and improvement of an odometry model with dynamic wheel and lateral dynamics integration," *Sensors (Switzerland)*, vol. 21, pp. 1–29, 1 2021.

[3] H. P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 116–121, 1985.

[4] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2 2007.

[5] A. I. Eliazar and R. Parr, "Learning probabilistic motion models for mobile robots," *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, pp. 249–256, 2004.