Assignment 4

Deadline: Friday 16 March 2018, 11:59 PM

In this assignment, we will build a line-based Kalman Filter for robot localization.

> **Note:** Assignment content is credited to ETH Zürich.

---

1. Introduction

As pointed out in the previous exercise, knowledge of the location of a platform is essential for a lot of robotics applications, and we motivated this with an autonomous vehicle hauling goods inside a warehouse from one place to another. Within this scenario, Exercise 3 demonstrated how to acquire a more abstract representation of linear structures perceived with a scanning range finder. This exercise will show that—given a map of the linear features in this representation—the robot can localize itself based on the linear structures it perceives.

This exercise closely follows the example given in [1, p. 331-342]. We **strongly** advice you to read the respective pages prior to attending the exercise session and to turn to this document for advice throughout the exercise. Please note that there are different approaches to implementing Jacobians. While deriving the functions encountered in this exercise is straightfoward, those of you familiar with Matlab's Symbolic Math Toolbox/MuPAD may find that a combination of Matlab's anonymous functions with calls to `sym`, `jacobian` and `matlabFunction` can increase efficiency in solving this problem set. Please also consider turning to [1] for help with the Jacobians.

---

2. Kalman Filter Localization

he Extended Kalman Filter used for localization in this exercise can be structured into a prediction step and an update step. In the following, we will look at the two steps separately.

2.1 State Prediction

Exercise 2 discussed the motion model for a differential drive robot. Given knowledge of the state $\hat{\mathbf{x}}_{t-1} = [x_{t-1}, y_{t-1}, \theta_{t-1}]^T$ at the previous time step and of the wheel displacements $\mathbf{u}_t = [\Delta s_l, \Delta s_r]^T$, the motion model can be employed to obtain an a priori estimate of the current state. Here, we follow the order of u t established on [1, p. 337], which conflicts with the order stated on [1, p. 272]. Please consider any implications of this reversed order on the implementation of the function and its Jacobians.

$$\hat{\mathbf{x}_t} = f(\mathbf{x}_{t-1}, \mathbf{u}_t) = \mathbf{x}_{t-1} + \begin{bmatrix} (\Delta s_l + \Delta s_r)/2 * \cos(\theta_{t-1} + (\Delta s_r - \Delta s_l)/2b) \\ (\Delta s_l + \Delta s_r)/2 * \sin(\theta_{t-1} + (\Delta s_r - \Delta s_l)/2b) \\ (\Delta S_r - \Delta s_l)/b \end{bmatrix}$$

It is reasonable to assume that the motion is subject to noise, which we choose to model as additive Gaussian noise $\nu \sim N(\mathbf{0}, \mathbf{Q})$ applied to the control inputs. As proposed in [1, p. 272], 1the noise on the control inputs can be modelled as independent for each wheel with a covariance proportional to the absolute value of the travelled distance, where a constant factor k is used to account for any non-deterministic effects.

$$\mathbf{Q} = \begin{bmatrix} k \mid \Delta s_l \mid & 0 \\ 0 & k \mid \Delta s_r \mid \end{bmatrix}$$

Hence, an a priori estimate of the covariance of the state can be computed as

$$\hat{\mathbf{P}}_t = \mathbf{F_x} * \mathbf{P}_{t-1} * \mathbf{F_x}^T + \mathbf{F_u} * \mathbf{Q}_t * \mathbf{F_u}^T,$$

where $\mathbf{F_x}$ and $\mathbf{F_u}$ denote the Jacobians of the motion model $f(\mathbf{x}_{t-1}, \mathbf{u}_t)$ with respect to the state estimate and the control inputs respectively. See also [1, p. 270-272, 337].

Task: Write a Matlab/Octave function $[\hat{\mathbf{x}}_t, \hat{\mathbf{F}_\mathbf{x}}, \hat{\mathbf{F}_\mathbf{u}}] = dransitionFunction(\mathbf{x}_{t-1})$ that accepts the previous state $\mathbf{x}_{t-1}$ of a differential drive robot as well as control inputs $\mathbf{u}_t$ and the inter wheel distance b as arguments and computes an estimate of the current state $\hat{\mathbf{x}}_t$ as well as the Jacobians of the state transition function with respect to the state $\hat{\mathbf{F}_\mathbf{x}}$ and the control inputs F̂ u respectively.

Validation: Run the function `validateTransitionFunction()` . This function uses a sequence of control inputs to propagate the state with the supplied motion model. The function reports on the correctness of your implementation. If your implementation is correct, the function plots a ground truth path as well as the forward integration of noisy control inputs using your motion model. You should observe that the ground truth path and your estimate diverge increasingly over the course of the experiment. This

illustrates that for many real-world applications where perturbations occur, relying solely on interoceptive information is insufficient.

2.2 State Update

As illustrated in the previous experiment, perturbations in the control inputs will result in an increasingly inaccurate estimate of the state of the robot. Hence, exteroceptive location cues are commonly employed in robotics application. In this exercise, the robot is capable of sensing linear structures and possesses a map $\mathbf{M}$, which contains all linear structures in its operating environment, expressed in a coordinate frame that will be referred to as *world coordinate frame*.

2.2.1 Measurement Function

As introduced in Exercise 3, lines can be parametrized as $\mathbf{m}^i = \begin{bmatrix}\alpha^i r^i\end{bmatrix}^T$. This parametrization will be applied to both, the output of our perception system z_t as well as the entries of the map $\mathbf{M}$. Note however, that while the parametrization is identical, the coordinate frames in which the measurements and the map are represented differ. While lines in the map are represented in the world coordinate frame, the robot senses lines in its *body coordinate frame* relative to its own, varying pose. Hence, the measurement can be modelled by transforming the lines in the map from the world coordinate frame into the body coordinate frame. A more detailed description of this transformation is given in [1, p. 338-340]. For the remainder of this exercise, a map $\mathbf{M}$ with K entries is represented by a $2 * K$ matrix by horizontally concatenating individual m^i .

Task: Write a Matlab/Octave function $[\hat{\mathbf{z}}_t, \hat{\mathbf{H}}_\mathbf{x}] = measurementFunction(\hat{\mathbf{x}}_t, \mathbf{m}^i)$ that accepts an a priori estimate of the state x̂ t and a map entry m i and that models a measurement $\hat{\mathbf{z}}_t$ as it would be perceived by a robot with state $\hat{\mathbf{x}}_t$, which constitutes the transformation from a line expressed in the world coordinate frame into the body coordinate frame of the robot. Additionally, compute the Jacobian of the measurement model $\hat{\mathbf{H}}_\mathbf{x}$ with respect to the state.

Validation: Run the function `validateMeasurementFunction()` . The function reports on the correctness of your implementation.

2.2. Measurement Association

In order to apply the Kalman filter update correctly, associations between observations and map entries need to be established. To this end we employ the ahalanobis distance

between a predicted measurement ˆz i t and an observation z j. With the innovation vijt as a measure of the difference between a predicted and observed measurement

$$\mathbf{v}_t^{ij} = \mathbf{z}_t^j - \hat{\mathbf{z}}_t^i$$

and the *innovation covariance* $\sum_{1N_t}^{ij}$

$$\sum\nolimits_{1N_t}^{ij} = \hat{\mathbf{H}}_t * \hat{\mathbf{H}}_t$$

the Mahalanobis distance is calculated as
KaTeX parse error: Expected '}', got 'EOF' at end of input: ...f{v}}^{ijT}_{]

---

3. V-Rep Experiment

To be presented in the next practical session Tuesday 13 Match.

---

4. Handing-In

Write your code for the aforementioned task in MATLAB script code provided. Then, compress and upload it to Blackboard.

References

[1] Roland Siegwart, Illah Nourbakhsh, and Davide Scaramuzza. Introduction to Autonomous Mobile Robots. MIT Press, 2nd edition, 2011.
[2] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.