TEACHING AUTONOMOUS ROBOTICS SYSTEMS: COMPARISON OF FILTERING TECHNIQUES

1
Kalman Filter2 2 Extended Kalman Filter3 3 Particle Filter3 4 Particle Filter5

Email:

Abstract— This is

Keywords— Mobile Robotics, Robotics Development Environment, Baysian Filters

Resumo— Os artigos

Palavras-chave— Robótica móvel, ambiente de desenvolvimento, filtros Bayesianos

1 Introduction

...

This work presents a teaching framework to be applied on robotics autonomous systems (RAS).

2 Position state estimation in robotics

Robots are commonly conceived to perform useful tasks in the real world, interacting physically with its surroundings structures, objects and agents. For achieving a desired task, the robot has to know its relevant intrinsic and extrinsic states - e.g., position, orientation, height, depth, joint velocity, tool temperature.

Sensors provide a way to input real world information into a robotic system. The instrument collected data vector $\mathbf{z} \in \mathbb{R}^N$ can be related to a desired real state vector $\mathbf{x} \in \mathbb{R}^M$ using a *measurement model* $h(.) : \mathbb{R}^M \mapsto \mathbb{R}^N$:

$$\mathbf{z} = h(\mathbf{x}) \tag{1}$$

One would simply need to find the inverse $h^{-1}(.)$ to obtain the value of the desired state $\mathbf{x}$. However, real world problems complexity commonly drops the $h^{-1}(.)$ function accuracy, generates a non-invertible equation for $h(.)$, or even restrains the formulation of a closed form to $h(.)$ at all. One classic approach to overcome these restrictions is by formulating the problem on a probabilistic manner:

$$\mathbf{z} \approx p(\mathbf{z}|\mathbf{x}) \tag{2}$$

In this case, the solution for $\mathbf{x}$ involves the use of statistical estimators (represented here as a generic rule $est(.)$) in function of the distribution $p(\mathbf{z}|\mathbf{x})$:

$$\hat{\mathbf{x}}(\mathbf{z}) \equiv est(p(\mathbf{z}|\mathbf{x})) \tag{3}$$

Using this concept, *filters* basically wrap the estimator technique (Eq. 3) into an algorithm to sequentially find, over a time basis t, the likelihood of a desired state vector $\mathbf{x}_t$ based on, and not restricted only to, sensor(s) reading(s) $\mathbf{z}_t$. The literature is rich in number and quality of filtering methods. Each filter is commonly designed for solving a group of similar problems, having its own characteristics, advantages and drawbacks.

Specifically, the robot pose (position and orientation) estimation still a great challenge as there is no method that solves it for all scenarios. Each implementation tends to be specific, depending on the robot sensory resources, environment complexity and task requirements.

Considering this, we selected three filtering strategies to expose in this work: (i) Kalman Filter; (ii) Extended Kalman Filter; and (iii) Particle filter. They are generic methods simple to understand and compose essentially a great variety of more complex state of the art robot localization methods. The knowledge of these three filters provides a solid base to solve a vast number of common robotics problems, not restricted only to pose estimation.

The sequence of this section is dedicated to expose the basic characteristics and usage of the mentioned filters. As it is not the objective of this work, we do not show any proof of their concepts. An interested reader may though find this information on the literature, as in (Thrun et al., 2005) and (Kovvali et al., 2013).

2.1 Kalman Filter

The Kalman Filter (KF) is a parametric Bayesian optimum estimator designed to sequentially compute the belief of a desired state vector $\mathbf{x}_t$ in a dynamical system. As advantages it holds the computational efficiency, simplicity of implementation, and a smaller variance for the belief in comparison to the motion model and sensor data.

However, it is restricted to linear systems and unimodal Gaussian representations for the belief.

Eq. 4a and 4b represents the state space model for a dynamic linear system. The Markovian assumption allows to write the system on a given time t only in function of its parameters at the previous time step $t-1$.

$$\mathbf{x}_t = A_t\mathbf{x}_{t-1} + B_t\mathbf{u}_t + \epsilon_t \tag{4a}$$

$$\mathbf{z}_t = C_t\mathbf{x}_t + \delta_t \tag{4b}$$

Additionally to the sensor input $\mathbf{z}_t$, it also considers a control signal $\mathbf{u}_t \in \mathbb{R}^P$ for the estimation. The components $A_t \in \mathbb{R}^{M\times M}$ is the state-transition matrix, $B_t \in \mathbb{R}^{M\times P}$ is the control matrix, and $C_t \in \mathbb{R}^{N\times M}$ is the measurement matrix. The arguments $\epsilon_t \in \mathbb{R}^M$ and $\delta_t \in \mathbb{R}^N$ are zero mean noise vectors with covariance matrices R_t and Q_t, respectively. The reader may notice that all parameters depend on the time step t, thus may evolve as a function of time.

Assuming the Gaussian assumption for the states distribution, the posteriors of Eq. 4a and 4b are defined by:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t) \equiv \mathcal{N}(\mathbf{x}_t; A_t\mathbf{x}_{t-1} + B_t\mathbf{u}_t, R_t) \tag{5a}$$

$$p(\mathbf{z}_t|\mathbf{x}_t) \equiv \mathcal{N}(\mathbf{z}_t; C_t\mathbf{x}_t, Q_t) \tag{5b}$$

Regarding this, the Kalman Filter represents the belief of the state vector $\mathbf{x}_t$, at each time step t, by a Gaussian probability density function (*p.d.f.*) with mean μ_t and covariance Σ_t:

$$\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_t; \mu_t, \Sigma_t) \tag{6}$$

Table 1 presents the KF algorithm. As parameters, it receives the previous time step belief $\mathbf{x}_{t-1}$, represented by μ_{t-1} and Σ_{t-1}, and the most recent control signal $\mathbf{u}_t$ and sensor reading $\mathbf{z}_t$. Lines 2 and 3 predicts the belief, for the current time step, considering only the last belief (μ_{t-1} and Σ_{t-1}) and the control signal $\mathbf{u}_t$. Kalman Filter updates the prediction with the sensor reading $\mathbf{z}_t$ in lines 5 and 6. It interestingly regulates the influence of $\mathbf{z}_t$ to the posterior based on a expected reading given the prediction. This task is performed by the *Kalman gain* K_t, which is calculated in line 4. The algorithm finnally returns the posterior belief for $\mathbf{x}_t$, represented by μ_t and Σ_t.

For the first time step, one has to provide an initial belief $\mathbf{x}_0$ as a Gaussian p.d.f.:

$$\mathbf{x}_0\mathcal{N}(\mathbf{x}_t; \mu_0, \Sigma_0) \tag{7}$$

If there is no prior information about the initial condition $\mathbf{x}_0$, a reasonable μ_0 with a large covariance Σ_0 is advisable. Subsequentially, one has to feed the algorithm, at each time step, with the previous belief and the updated control signal and sensor reading to perpetuate the state prediction.

Algorithm 1 Kalman Filter

```
1: procedure KF($\mu_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t$)
2:     $\overline{\mu}_t \leftarrow A_t\mu_{t-1} + B_t\mathbf{u}_t$
3:     $\overline{\Sigma}_t \leftarrow A_t\Sigma_{t-1}A_t^T + R_t$
4:     $K_t \leftarrow \overline{\Sigma}_tC_t^T(C_t\overline{\Sigma}_tC_t^T + Q_t)^{-1}$
5:     $\mu_t \leftarrow \overline{\mu}_t + K_t(\mathbf{z}_t - C_t\overline{\mu}_t)$
6:     $\Sigma_t \leftarrow (I - K_tC_t)\overline{\Sigma}_t$
7:     return $\mu_t, \Sigma_t$
```

2.2 *Extended Kalman Filter*

As its name implies, the Extended Kalman Filter (EKF) is a modified version of the Kalman Filter for estimating non-linear systems states. The KF Gaussian assumption for the belief representation still valid, but there is a relaxation on the linear form of the system model.

Considering a non-linear system, the state transition and measurement models are represented by non-linear functions $g(.)$ and $h(.)$. One may remark that not necessarily both functions have to be non-linear. With this assumption, the system model, represented by Eq. 4a and 4b, becomes:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \epsilon_t \tag{8a}$$

$$\mathbf{z}_t = h(\mathbf{x}_t) + \delta_t \tag{8b}$$

However, regarding the estimation filter process, non-linear functions distort the Gaussian shape for the belief at the propagation step. This breaks the Gaussian assumption of the filter, making impossible to compute accurate estimations for $\mathbf{x}_t$ or $\mathbf{z}_t$.

The EKF overcomes this problem by using the first order Taylor expansion to linearize functions $h(.)$ and $g(.)$. It computes a linear function based on the original function's image and slope at a given point. For the function $g(.)$, the linearization is around the most probable point of $\mathbf{x}_{t-1}$ belief, which is μ_{t-1}. It then becomes:

$$g(\mathbf{x}_{t-1}, \mathbf{u}_t) \approx g(\mu_{t-1}, \mathbf{u}_{t-1}) + g'(\mu_{t-1}, \mathbf{u}_t)(\mathbf{x}_{t-1} - \mu_{t-1}) \tag{9}$$

The term $g'(\mu_{t-1}, \mathbf{u}_t)$ represents the slope of $g(.)$ at μ_{t-1}, given by:

$$g'(\mu_{t-1}, \mathbf{u}_t) = \frac{\partial g(\mathbf{x}_{t-1}, \mathbf{u}_t)}{\partial \mathbf{x}_{t-1}} \tag{10}$$

One can also represent $g'(.)$ by the Jacobian matrix G_t:

$$G_t = \frac{\partial g(.)}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} & \cdots & \frac{\partial g_n}{\partial x_n} \end{bmatrix} \tag{11}$$

What turns Eq. 9 in:

$$g(\mathbf{x}_{t-1}, \mathbf{u}_t) \approx g(\mu_{t-1}, \mathbf{u}_{t-1}) + G_t(\mathbf{x}_{t-1} - \mu_{t-1}) \tag{12}$$

A similar approach is considered for the $h(.)$ function. However, its derivative $h'(.)$ is linearized around the point $\overline{\mu}_t$, of the filter's prediction step, instead. It then becomes:

$$\begin{aligned} h(\mathbf{x}_t) &\approx h(\overline{\mu}_t) + h'(\overline{\mu}_t)(\mathbf{x}_t - \overline{\mu}_t) \\ &\approx h(\overline{\mu}_t) + H_t(\mathbf{x}_t - \overline{\mu}_t) \end{aligned} \tag{13}$$

Table 2 presents the EKF Algorithm, which is essentially the same algorithm as the Kalman Filter (Table 1). The main differences are in line 2, where $g(.)$ replaces matrices A_t and B_t, and in line 5, where $h(.)$ replaces the matrix C_t. The algorithm also uses the Jacobian matrices G_t, described in Eq. 11, and H_t, which is computed by a similar process as G_t. The reader may remark that both G_t and H_t are time dependant, eventually varying for each time step.

Algorithm 2 Extended Kalman Filter

```
1: procedure EKF($\mu_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t$)
2:     $\overline{\mu}_t \leftarrow g(\mu_{t-1}, \mathbf{u}_t)$
3:     $\overline{\Sigma}_t \leftarrow G_t \Sigma_{t-1} G_t^T + R_t$
4:     $K_t \leftarrow \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$
5:     $\mu_t \leftarrow \overline{\mu}_t + K_t(\mathbf{z}_t - h(\overline{\mu}_t))$
6:     $\Sigma_t \leftarrow (I - K_t H_t)\overline{\Sigma}_t$
7:     return $\mu_t, \Sigma_t$
```

Once the described adaptations are implemented, the Extended Kalman Filter operation becomes quite similar to the Kalman Filter. Due to this, it holds the KF computational efficiency, becoming a popular filter for non-linear systems states estimation (Thrun et al., 2005).

As a remark, the Extended Kalman Filter is constrained by two factors. The first refeers to the state belief uncertainty. The bigger is the belief's covariance Σ_t, the less accurate is the imaging process through the linearized function, what drops considerably the EKF accuracy. Another consideration is the local non-linearity of the system model functions. Naturally, the linearization process is more precise for weak non-linear functions.

2.3 Particle Filter

The filter techniques seen so far - the KF and EKF algorithms - share the Gaussian belief representation using μ_t and Σ_t in a normal distribution $\mathcal{N}(;, \mu_t, \Sigma_t)$. Despite the computational simplicity, among other advantages, this representation restrains the state belief to an unimodal probability density function. Thus, it is not possible to represent the state likelihood by more complex density distributions, what is necessary for many robotics applications.

The Particle Filter (PF) is a nonparametric method that overcomes this constrain by representing the $\mathbf{x}_t$ state probability density by a set $\mathcal{X}_t$ of $M \in \mathbb{N}$ finite samples x_t^m, called particles:

$$\mathcal{X}_t \equiv \{x_t^{m=1}, x_t^2, \ldots, x_t^M\}, m \in \mathbb{N} = 1, 2 \ldots M \tag{14}$$

Each particle x_t^m is independent and represents a single possible realization of the system state $\mathbf{x}_t$. A noisy motion model sequentially updates each particle (Eq. 15), normally causing the particles to spread, at a first time, and later concentrating on the most likely regions. Thus, one may consider the set of all particles $\mathcal{X}_t$ as a discrete approximation of the $\mathbf{x}_t$ posterior probability density. The particles independence allows more complex probability density representations of $\mathbf{x}_t$, including multimodal shapes.

$$x_t^{[m]} \equiv p(x_t^{[m]} | x_{t-1}^{[m]}, \mathbf{u}_t) \tag{15}$$

However, if only the procedure of Eq. 15 is performed, several particles would tend to low probability areas as $t \to \infty$. The filter's efficiency drops, as these particles consume computational resources to update and do not provide useful information.

To solve that, the Particle Filter trick is to randomly replace particles with low probability by the more probable ones, while commonly keeping their quantity M. For that, the filter associates a weight $w_t^{[m]}$ to each particle based on the measurement model (Eq. 16). The replacing method uses those weights to perform the particles replacement.

$$w_t^{[m]} \equiv p(\mathbf{z}_t | \mathbf{x}_t^{[m]}) \tag{16}$$

Algorithm 3 Particle Filter

```
1: procedure PF($\mathcal{X}_{t-1}, \mathbf{u}_t$, $\mathrm{z}_t$)
2:     $\overline{\mathcal{X}}_t = \mathcal{X}_t \leftarrow \emptyset$
3:     for $m = 1$ to $M$ do:
4:         sample $x_t^{[m]} \leftarrow p(x_t | x_{t-1}^{[m]}, \mathbf{u}_t)$
5:         $w_t^{[m]} \leftarrow p(\mathbf{z}_t | x_t^{[m]})$
6:         $\overline{\mathcal{X}}_t \leftarrow \overline{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:     endfor
8:     for $m = 1$ to $M$ do
9:         draw $i$ with probability $\propto w_t^{[i]}$
10:        add $x_t^{[i]}$ to $\mathcal{X}_t$
11:    endfor
12:    return $\mathcal{X}_t$
```

Table 4 presents a Particle Filter basic implementation. The algorithm receives as input the last particle set $\mathcal{X}_{t-1}$, and the more recent control signal $\mathbf{u}_t$ and sensor data $\mathbf{z}_t$. Line 2 initializes the new set array $\overline{\mathcal{X}_t}$ and $\mathcal{X}_t$ with no particles. Lines 3 to 7 perform the sample of M particles $x_t^{[m]}$, computes an associated weight $w_t^{[m]}$ for each one, and stores all of them into $\overline{\mathcal{X}_t}$. Ultimately, the re-sampling step (lines 8 to 10) composes the new set $\mathcal{X}_t$ by randomly sorting M particles from $\overline{\mathcal{X}_t}$ based on their weights.

One may need to provide an initial particle set $\mathcal{X}_0$ to run the algorithm. If the system initial state $\mathbf{x}_0$ is known, $\mathcal{X}_0$ should be filled with M identical particles $x_0^{[m]} = \mathbf{x}_0$, or with a distribution of particles with $\mathbf{x}_0$ as its mean, i.e., $\mathcal{N}(x;\mathbf{x}_0, \Sigma_0)$. When there is no prior information about $\mathbf{x}_0$, one may start $\mathcal{X}_0$ as an uniform distribution over all possible states of the system.

The Particle Filter relaxes some assumptions that constrains other classical method, as the Kalman Filter and the Extended Kalman Filter. This flexibility is highly desirable for many applications, not only in robotics, what makes the PF a state of the art method for them (Jouin et al., 2016).

The filter usage requires some practical considerations to avoid particles deprecation. Firstly, there is a trade-off between the filter's efficiency and its computational complexity by regulating the number of particles M. Less particles may not represent all relevant possibilities, leading to an erroneous estimation. In the other hand, more particles requires more computational effort to perform the update step. As a rule of thumb, it is desirable to set a big enough value for M: most applications run well with $M = 1000$ particles (Thrun et al., 2005). Additionally. there are PF techniques that dynamically adjust the total number of particles.

Another sensible point is the particles re-sampling step. As it is a stochastic process, there is a probability of no high probable particles survive, what makes unfeasible the esimation. Therefore, the re-sampling step should not be executed indiscriminately. The literature contains different re-sampling strategies for the Particle Filter algorithm (Li et al., 2015). A common approach is to choose when re-sampling by tracking the robot movement or the particles variance.

3 Implementation

3.1 Kalman

Kalman filter is a Bayesian optimum estimator. The basic Kalman filter applies to linear systems with Gaussian process and measurements noise. The simple Kalman Filter is widely used due to its simplicity and computational efficiency. One great advantage of KF is that its belief has a smaller variance than the motion model and the sensor reading. Modified versions can treat with non-linear systems, and they are reviewed in later sections.

We want to estimate the planar position and velocity of our robot. The system state space is:

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^{\mathbf{T}} \tag{17}$$

Considering a Markov system, we can write the system in a given instant t strictly in function of its state in $t-1$. For our planar case of study, we use the following function $g(.) : \Re^4 \mapsto \Re^4$ to relate these two time instants:

$$\begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} = g(\mathbf{x}_{t-1}) = \begin{bmatrix} x_{t-1} + \dot{x}_{t-1}\Delta_t \\ y_{t-1} + \dot{y}_{t-1}\Delta_t \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \end{bmatrix} \tag{18}$$

For the measurement model, we consider that the sensor provides directly the robot's Cartesian positions an velocities with Gaussian noise. That way, the function $g(.)$ that maps the measurement space to the system state space becomes the identity.

As this is a linear problem, we can write the state space model as:

$$\mathbf{x}_t = A_t \mathbf{x}_{t-1} + \epsilon_t \tag{19a}$$

$$\mathbf{z}_t = C_t \mathbf{x}_t + \delta_t \tag{19b}$$

Where $\epsilon_t, \delta_t \in \Re^4$ are white noise. One may remark that we do not consider a control signal $\mathbf{u}_t$ in Eq. 19a, as this is a tracking problem. We can rewrite Eq. 19a and 19b in matrix form considering Eq. 18 as:

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \end{bmatrix} + \begin{bmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \\ \epsilon_{t,3} \\ \epsilon_{t,4} \end{bmatrix} \tag{20a}$$

$$\mathbf{z}_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} + \begin{bmatrix} \delta_{t,1} \\ \delta_{t,2} \\ \delta_{t,3} \\ \delta_{t,4} \end{bmatrix} \tag{20b}$$

3.2 Extended Kalman Filter

The basic Kalman Filter works only for linear problems. To work with non-linear functions, the Extended Kalman Filter basically linearizes the problem using Jacobian Matrices $\overline{A}$ abd $\overline{C}$ based on the state space model functions. That turns Eq. 19a and 19b in:

$$\mathbf{x}_t = \overline{A}_t \mathbf{x}_{t-1} + \epsilon_t \tag{21a}$$

$$\mathbf{z}_t = \overline{C}_t \mathbf{x}_t + \delta_t \tag{21b}$$

We consider now our sensor as a radar (Fig. 1) located at the origin that gives the distance

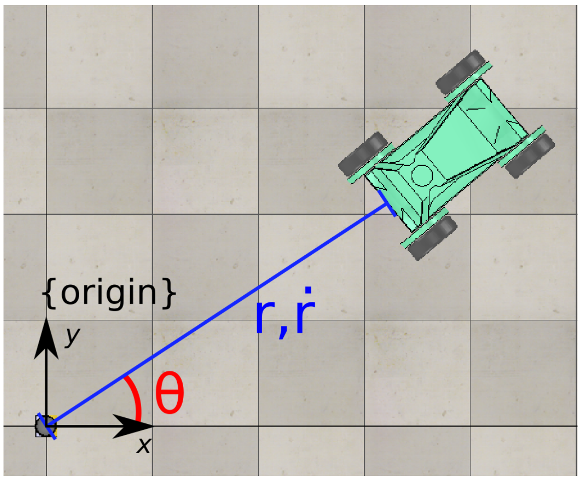

Figure 1: Radar measurement of the robot position.

to the object r, its ratio $\dot{r}$, and the angle of the detected object with respect to the radar frame:

$$\xi = \begin{bmatrix} r & \dot{r} & \theta \end{bmatrix}^{\mathbf{T}} \tag{22}$$

The relation between the radar measurement and the system state space is non-linear. One may use the function $h(.) : \Re^4 \mapsto \Re^3$ to map from the system state space $\mathbf{x}_t$ to the measurement space $\mathbf{z}_t$:

$$\xi = h(\mathbf{x}) = \begin{bmatrix} \sqrt{x^2+y^2} & \frac{x\dot{x}+y\dot{y}}{\sqrt{x^2+y^2}} & atan2(y,x) \end{bmatrix}^{\mathbf{T}} \tag{23}$$

The $atan2(.)$ function is the inverse tangent that considers the legs quadrant for computing the angle. Considering Eq. 18 and 23, we write the state space model for the Extended Kalman Filter as:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}) + \epsilon_t \tag{24a}$$

$$\mathbf{z}_t = h(\mathbf{x}_t) + \delta_t \tag{24b}$$

The system transition function $g(.)$ remains the same as in KF. What changes is the function $h(.)$ that relates the sensor reading to the system state space. For the EKF algorithm, we have so to compute the Jacobian for the measurement model:

$$\overline{C}_t = \frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \frac{\partial h_n}{\partial x_2} & \cdots & \frac{\partial h_n}{\partial x_n} \end{bmatrix} \tag{25}$$

Resulting, in our case in:

$$\overline{C}_t = \begin{bmatrix} \frac{x_n}{\sqrt{x_n^2+y_n^2}} & \frac{y_n}{\sqrt{x_n^2+y_n^2}} & 0 & 0 \\ \frac{y_n(\dot{x}_n y_n - x_n \dot{y}_n)}{\sqrt{(x_n^2+y_n^2)^3}} & \frac{x_n(x_n \dot{y}_n - \dot{x}_n y_n)}{\sqrt{(x_n^2+y_n^2)^3}} & \frac{x_n}{\sqrt{x_n^2+y_n^2}} & \frac{y_n}{\sqrt{x_n^2+y_n^2}} \\ \frac{y}{\sqrt{x^2+y^2}} & \frac{x}{\sqrt{x^2+y^2}} & 0 & 0 \end{bmatrix} \tag{26}$$

The reader may notice that the Jacobian matrix (using Eq. 25) for the system model $h(.)$ equals the matrix A_t from Eq. 20a. This happens because the system propagation model still linear, even if the measurement model is no longer.

3.3 Particle Filter

The Particle Filter (PF) is also a non-parametric Bayesian method to estimate the posterior of a system state. It shares with the Discrete Bayes Filter the property of defining the belief by a finite number of parameters. However, it diverges on how to generate and to populate these parameters over the state space (Thrun et al., 2005).

The PF uses a finite set of samples x_t^k, called *particles*, to estimate the posteriori:

$$\mathcal{X}_t = \left\{ x_t^1, x_t^2 \ldots x_t^N \right\} \tag{27}$$

Each particle is a instantiation of the system state, representing a possible situation of the real state. The key idea is to independently apply the noisy control $\mathbf{u}_t$, and to relate a weight w_t^k to each particle x_t^k. This weight considers the noisy measurement $\mathbf{z}_t$:

$$w_t^k \propto p(\mathbf{z}_t | x_t^k) \tag{28}$$

Finally, the algorithm re-samples the particles. Different variations of the PF exist, but in their majority, they perform a random resampling based on the particles weight w_t^k.

The Alg. 4 shows the basic implementation of a Particle Filter (Thrun et al., 2005).

Algorithm 4 Particle Filter

```
**particleFilter**( $\mathcal{X}_{t-1}, \mathbf{u}_t, \mathbf{z}_t$ )
$\mathcal{X}_t = \bar{\mathcal{X}}_t = \emptyset$
for $m = 1$ to $M$ do
sample $x_t^m \sim p(x_t | \mathbf{u}_t, x_{t-1}^m)$
$w_t^m = p(\mathbf{z}_t | x_t^m)$
$\bar{\mathcal{X}} = \bar{\mathcal{X}} + \langle x_t^m, w_t^m \rangle$
endfor
for $m = 1$ to $M$ do
draw $i$ with probability $\propto w_t^i$
add $x_t^i$ to $\mathcal{X}_t$
endfor
return $\mathcal{X}_t$
```

Each particle is independently stimulated with noisy control signal $\mathbf{u}_t$ and measurement $\mathbf{z}_t$ to compute a posteriori:

$$x_t^n \approx p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{u}_t) \tag{29}$$

In this way, each particle represents a possible state for $\mathbf{x}_t$.

As a great advantage, it does not need to parametrically describe the belief.

We still consider the state and measurement models from the Kalman implementation.

4 V-REP

5 Conclusion

Acknowledgment

The authors would like to thank Instituto Tecnológico Vale - Brasil (ITV) and Universidade Federal de Ouro Preto - Brasil (UFOP) for providing the equipments and facilities. This study was partially financed by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001, and the Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq).

References

Jouin, M., Gouriveau, R., Hissel, D., Péra, M.-C. and Zerhouni, N. (2016). Particle filter-based prognostics: Review, discussion and perspectives, *Mechanical Systems and Signal Processing* **72-73**: 2–31.

Kovvali, N., Banavar, M. and Spanias, A. (2013). *An Introduction to Kalman Filtering with MATLAB Examples*, Morgan & Claypool.

Li, T., Bolic, M. and Djuric, P. M. (2015). Resampling methods for particle filtering: classification, implementation, and strategies, *IEEE Signal Processing Magazine* **32**(3): 70–86.

Thrun, S., Burgard, W. and Fox, D. (2005). *Probabilistic Robotics*, The MIT Press.

Unused \captionsetup[1] on input line