COMPARISON OF NONLINEAR STOCHASTIC FILTERS PERFORMANCE FOR DIFFERENTIAL DRIVE ROBOTS LOCALIZATION

Samuel C. Pinto*, Marcos R. O. A. Máximo*, Marcelo G. S. Bruno*

*Instituto Tecnológico de Aeronáutica.
Praça Marechal Eduardo Gomes, 50.
São José dos Campos, São Paulo, Brasil

Emails: sacepi@gmail.com, maximo.marcos@gmail.com, bruno@ita.br

Abstract—This paper presents a performance comparison of nonlinear stochastic filtering techniques, namely EKF, UKF and Particle Filter, used to provide a dynamical estimate of a differential drive mobile robot’s position and heading angle. The performance is evaluated using the processing time and mean square error. These results were obtained through simulation.

Keywords—Mobile Robots, Localization, Stochastic Filtering.

1 Introduction

Differential drive robots are used in many autonomous applications, such as domestic robots, explorer robots and soccer playing robots. An essential task concerning these robots is estimating its position in an environment using its commands and sensor information. This task is denominated localization. Having an accurate estimate of its own position and orientation is fundamental to feedback control systems, trajectory planning and decision making algorithms.

One particularly interesting application of differential drive robots is the IEEE Very Small Size (VSS) competition, (7th Latin American IEEE Student Robotics Competition, 2008). The VSS challenge consists of a soccer based challenge with autonomous robots with maximum dimensions of 7.5 cm x 7.5 cm x 7.5 cm. Each team has three robots on the field and is allowed to have a camera positioned above the field and develop a computational vision system. Most of the processing is centralized in an external computer that establishes a wireless communication link with the robots.

This work is a result of the development of the localization system for ITAndroids’ VSS team. ITAndroids is the Instituto Tecnológico de Aeronáutica’s (ITA’s) robotics competition group, which was founded in 2011 after years of inactivity. This team participates in RoboCup’s Soccer 2D, Soccer 3D and Humanoid Kid-Size leagues and IEEE Very Small Size (VSS) and Standard Educational Kit (SEK) categories, in both national and international competitions. ITAndroids’ VSS team was founded in 2013 and has participated in 2014 Latin America Robotics Competition (LARC) and Brazilian Robotics Competition (CBR).

Probabilistic techniques are the standard approach for implementing mobile robots localization. (Fox et al., 1999) and (Gutmann and Fox, 2002) have proposed and compared the performance of some stochastic filtering techniques for mobile robots. Many RoboCup groups also use these techniques in their robots, as described in (Petrović and Lilienthal, 2009), (Sridharan et al., 2005), (Quinlan and Middleton, 2010). Moreover, some works ((Silva and Bruno, 2006),(Silva and Bruno, 2007)) have dealt with differential drive robots localization and environment parameters estimation.

The goal of this paper is to present some of the most common techniques utilized in mobile robots localization and compare their performances. These techniques will be implemented considering VSS models for the robot dynamics and observations. These filters’ performances will be compared in terms of Root Mean Square Error (RMSE) and processing time. In this paper, Section 1 has provided an overview of this subject. Section 2 provides a brief description of the stochastic filtering techniques used in this paper. Section 3 describes the stochastic model used to represent the differential drive robot. In Section 4, the simulation setup and the results will be presented. Section 5 concludes and shares some ideas for future works.

2 Probabilistic Localization

From a probabilistic point of view, the localization problem consists of determining the probability distribution function (pdf) for a given state $s_k$ at every instant $k$, given the commands $ε_i$ and observations $z_i$, $i = 1, .., k$. The problem is usually modeled as

$$s_0 = p(x_0), \quad (1)$$

$$s_{k+1} = f_k(s_k, ε_k) + u_k, \quad (2)$$

$$z_k = h_k(s_k) + v_k. \quad (3)$$

In Eqs. (1) to (3) $u_k$ and $z_k$ are random noises and $f_k$ and $h_k$ are generic functions. The random
variables \( \{x_0, \{u_k\}_{k>0}, \{v_k\}_{k>0}\} \) are assumed mutually independent and jointly Gaussian. The initial distribution, \( s_0 \), has \( s_{0|0} \), mean and \( P_{0|0} \) covariance matrix. Consider that the noise components have zero mean and that the state estimate \( s_{k-1|k-1} \) has a covariance matrix \( P_{k-1|k-1} \) and noise components \( u_k \) and \( v_k \) have, respectively, covariance matrices \( Q_k \) and \( R_k \). Given these models, some stochastic approaches for estimating the current state pdf are presented in the next subsections.

2.1 Extended Kalman Filter

The Kalman Filter was originally designed considering that the functions \( f_k(s_k, \varepsilon_k) \) and \( h_k(s_k) \) in Equations (2) and (3) were linear and that all the random noises and initial state distribution were Gaussian. This filter is optimal and has a good computational performance. Therefore, an extension for this filter was introduced, in order to also apply those equations to nonlinear models, but still holding the Gaussianity hypothesis. This extension is known as Extended Kalman Filter (EKF) and is described in (Ljung, 1979). It considers a first order (linear) approximation of the nonlinear function. It uses equations similar to the Kalman Filter’s ones, but calculates the matrices \( F_k \) and \( H_k \) as in Eqs. (4) and (5).

\[
F_k = \frac{\partial f_k(s_k, \varepsilon_k)}{\partial s} |_{s=s_{k-1|k-1}} \tag{4}
\]

\[
H_k = \frac{\partial h_k(s_k)}{\partial s} |_{s=s_{k-1|k-1}} \tag{5}
\]

In Eqs. (4) and (5), \( \frac{\partial f_k(s_k, \varepsilon_k)}{\partial s} \) and \( \frac{\partial h_k(s)}{\partial s} \) are the Jacobian matrices of \( f_k \) and \( h_k \). The complete EKF algorithm is presented in Algorithm 1.

---

Algorithm 1: EKF

**Initialization**

\( s_{0|0} \leftarrow s_{0|0,1} \)

\( P_{0|0} \leftarrow P_{0|0,0} \)

for \( k \leftarrow 1 \) to \( N \) do

**Prediction Step**

\( s_{k|k-1} \leftarrow f_k(s_{k-1|k-1}, \varepsilon_k) \)

\( F_k \leftarrow \frac{\partial f_k(s_k, \varepsilon_k)}{\partial s} |_{s=s_{k-1|k-1}} \)

\( H_k \leftarrow \frac{\partial h_k(s_k)}{\partial s} |_{s=s_{k-1|k-1}} \)

**Filtering Step**

\( P_{k|k-1} \leftarrow F_k P_{k-1|k-1} F_k^T + Q_k \)

\( K_k \leftarrow P_{k|k-1} H_k (H_k P_{k-1|k-1} H_k^T + R_k)^{-1} \)

\( s_k|k \leftarrow s_{k|k-1} + K_k (z_k - h(s_k|k-1)) \)

\( P_{k|k} \leftarrow (I - K_k H_k) P_{k|k-1} \)

end

2.2 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) is another extension of the original Kalman Filter to nonlinear problems. However, it does not use a first order Taylor approximation for calculating the moments, as does the EKF. The UKF uses an expansion through deterministic samples, denoted as sigma points, to estimate the distribution’s posterior mean and variance. This method, by carefully choosing the sigma points, leads to better accuracy in higher order moments calculations, (Julier and Uhlmann, 2004) and (Wan and Van Der Merwe, 2000). The result is that UKF often leads to lower mean square errors than the EKF, but has similar processing time. This performance distinction between these two techniques becomes more evident when the model has strong nonlinearities. Another advantage of UKF is that there is no need to calculate Jacobians, which leads to simpler filter design procedure. Algorithm 2 presents the version of UKF known as Scaled UKF.

In Algorithm 2, \( \lambda = \alpha^2(n+\kappa) \) and \( \alpha \) is a constant that determines the spread of sigma points around the mean and is usually a small positive value (e.g., \( 10^{-3} \)). \( \kappa \) is an auxiliary constant that is usually set to 0. \( \beta \) is used to incorporate information from the prior distribution. For Gaussian distributions, the optimal value for \( \beta \) is 2 (Wan and Van Der Merwe, 2000). \( \sqrt{P} \) is the square-root matrix of \( P \), defined as a matrix \( F \) such as \( FF^T = P \). \( \sqrt{P_k} \) denotes the \( k \)-th row of matrix \( \sqrt{P} \). \( n \) and \( m \) are, respectively, the dimension of the state and the observation. The extended state \( s_k^u \) and covariance \( P_k^u \) are defined as

\[
s_k^u = \begin{bmatrix} s_{k-1|k-1} \atop 0_{n \times 1} \atop 0_{m \times 1} \end{bmatrix} \tag{6}
\]

\[
P_k^u = \begin{bmatrix} P_{k-1|k-1} & 0_{n \times n} & 0_{n \times n} \\ 0_{n \times m} & Q_k & 0_{n \times n} \\ 0_{m \times m} & 0_{m \times m} & R_k \end{bmatrix} \tag{7}
\]

2.3 Particle Filter

Particle filters intend to represent the pdf using a set of \( N_p \) samples \( s^{(i)} \). However, since in practical situations the real pdf \( p(s_k|z_{0:k}) \) is not available, it is necessary to sample those particles from an auxiliary pdf known as importance function, denoted \( q^{(i)}(s_k^{(i)}|z_{0:k-1}, z_{0:k}) \). It can be shown, (Liu et al., 2001) and (Bruno, 2013), that if we introduce weight \( w_k^{(i)} \), which can be recursively calculated using Eq. (8), that the relation (10) holds for any measurable function, under the usual Hidden Markov Model assumptions. In (10), "a.s" denotes the almost sure convergence (Bruno, 2013) as the number of samples goes to infinity.
Algorithm 2: UKF

Initialization
\[ s_{0|0} \leftarrow s_{0|i}, \]
\[ P_{0|0} \leftarrow P_{0|i}, \]
for \( k \leftarrow 1 \) to \( N \) do

Draw Sigma Points
\[ \chi^{(k)}_{j} \leftarrow s_{k|k-1} + \sqrt{n + \lambda}\sqrt{P_{k|j}} \]
for \( j \leftarrow 1 \) to \( n \) do
\[ \chi^{(k)}_{j,k} \leftarrow s_{k|k-1}^{a} \]
\[ W^{(k)}_{j,m} \leftarrow \frac{1}{2(n+\lambda)} \]
\[ W^{(k)}_{j,c} \leftarrow \frac{1}{2(n+\lambda)} \]
end for
for \( j \leftarrow n + 1 \) to \( 2n \) do
\[ \chi^{(k)}_{j} \leftarrow s_{k|k-1}^{w} \]
\[ W^{(k)}_{j,m} \leftarrow \frac{1}{2(n+\lambda)} \]
\[ W^{(k)}_{j,c} \leftarrow \frac{1}{2(n+\lambda)} \]
end for

Prediction Step
for \( j \leftarrow 0 \) to \( 2n \) do
\[ \chi^{(k)}_{j,k} \leftarrow f(\chi^{(k)}_{j,x}, \epsilon^{(k)}_{k}) + \chi^{(k)}_{j,u} \]
end for
\[ s_{k|k-1} \leftarrow \sum_{j=0}^{2n} W_{j,m}(\chi^{(k)}_{j,k} - s_{k|k-1}) \]
\[ P_{k|k-1} \leftarrow \sum_{j=0}^{2n} W_{j,c}(\chi^{(k)}_{j,k} - s_{k|k-1})(\chi^{(k)}_{j,k} - s_{k|k-1})^{T} \]

Filtering Step
for \( j \leftarrow 0 \) to \( 2n \) do
\[ \Upsilon_{j,k} \leftarrow h(\chi_{j,k}) + \chi^{(k)}_{j,v} \]
end for
\[ Y_{k} \leftarrow \sum_{j=0}^{2n} W_{j,m}Y_{j,k} \]
\[ P_{yy} \leftarrow \sum_{j=0}^{2n} W_{j,c}(Y_{j,k} - \Upsilon_{k})(Y_{j,k} - \Upsilon_{k})^{T} \]
\[ P_{yk} \leftarrow \sum_{j=0}^{2n} W_{j,c}(\chi_{j,k} - s_{k|k-1})(Y_{j,k} - \Upsilon_{k})^{T} \]
\[ K_{k} \leftarrow \frac{P_{yk}P_{yy}^{-1}s_{k|k-1}}{1 - K_{k}H_{k}P_{k|k-1}} \]
\[ P_{k|k} \leftarrow (I - K_{k}H_{k})P_{k|k-1} \]
end for

\[ w_{k}^{(i)} = B_{k}w_{k-1}^{(i)} \]
\[ p(z_{k}|s_{k}^{(i)})p(s_{k}^{(i)}|s_{k-1}^{(i)}) \]
\[ q(s_{k}^{(i)}|s_{0:k-1}, z_{0:k}) \]
\[ B_{k} = \frac{1}{\sum_{j=1}^{N_{p}} w_{k}^{(j)}} \]
\[ \sum_{j=1}^{N_{p}} w_{k}^{(j)} g(s_{0:k}^{(j)}) \rightarrow_{N_{p} \rightarrow \infty} \int_{D} g(s_{0:k})p(s_{0:k}|z_{0:k})ds_{0:k} \]
end for

One of the simplest implementations of the particle filter, known as the Bootstrap filter assumes that \( q(s_{k}^{(i)}|s_{0:k-1}, z_{0:k}) = p(s^{(i)}_{k}|s^{(i)}_{k-1}) \). This means that \( s_{k}^{(i)} \) is sampled according to \( s_{k}^{(i)} = \sum_{j=1}^{N_{p}} w_{k}^{(j)} g(s_{0:k}^{(j)}) \). With this choice of importance function, Eq. (8) can be simplified to Eq. (11).
\[ w_{k}^{(i)} = B_{k}w_{k-1}p(z_{k}|s_{k}^{(i)}) \]
end for

Algorithm 3: Bootstrap

Initialization
\[ \text{Draw } s_{0}^{(j)} \sim p(s_{0}), j = 1, ..., N \]
\[ w_{0}^{(j)} \leftarrow p(z_{0}|s_{0}^{(j)}) \]
\[ w_{0}^{(j)} \leftarrow \sum_{m=0}^{N} w_{0}^{(m)} \]

Resample
for \( k \leftarrow 1 \) to \( N \) do
\[ \text{Draw } s_{k}^{(j)} \sim p(s_{k}|s_{k-1}^{(j)}), j = 1, ..., N \]
\[ w_{k}^{(j)} \leftarrow p(z_{k}|s_{k}^{(j)}) \]
\[ w_{k}^{(j)} \leftarrow \sum_{m=0}^{N} w_{k}^{(m)} \]
\[ s_{k} \leftarrow \sum_{m=0}^{N} w_{k}^{(m)} s_{k}^{(m)} \]
end for

It is important to note that the resampling step can introduce a large processing time. (Thrun et al., 2005) presents an \( O(n) \) algorithm for resampling and this is used in the simulations presented in this paper.
3 Stochastic Modeling of Differential Drive Robots

Differential drive robots are robots that have only two tensioned wheels. Usually, the robot has more wheels but they are used only to ensure stability. By changing the tensioned wheels’ speed, it is possible to control both the linear and the angular velocities, measured in the center of the robot frame. In this work, the goal is to estimate the linear and angular velocities, measured in the center of the robot frame. The model inputs $e_k$ are the linear and angular velocities, $v$ and $w$. The kinematics of a differential drive robot impose the differential equations shown in Eq. (12).

$$\begin{bmatrix}
\dot{x}(t) \\
\dot{y}(t) \\
\dot{\psi}(t)
\end{bmatrix} =
\begin{bmatrix}
v(t)\cos(\psi(t)) \\
v(t)\sin(\psi(t)) \\
w(t)
\end{bmatrix} (12)$$

We consider that the robot’s dynamics is very fast, so that between two sequential commands the speeds of the wheels are constant. The period between two sequential commands is denoted $T_s$. Integrating Eq. (12) from $(k-1)T_s$ to $kT_s$, a discrete-time model is obtained. Moreover, we model the noise as additive in every dimension of the state and we consider that the noise components are independent. The complete movement model is presented in Eq. (13).

$$\begin{bmatrix}
x_k \\
y_k \\
\psi_k
\end{bmatrix} =
\begin{bmatrix}
x_{k-1} + v_k\cos(\psi_T + \frac{w_k}{2}) \\
y_{k-1} + v_k\sin(\psi_T + \frac{w_k}{2}) \\
\psi_{k-1} + w_kT
\end{bmatrix} +
\begin{bmatrix}
u_k, x \\
u_k, y \\
u_k, \psi
\end{bmatrix} (13)$$

As described in Section 1, the model was developed considering the VSS competition. Therefore, for the observation model, it will be considered that the observations are the result of a vision system with camera positioned above the middle of the field. This vision system is able to observe directly the position and the orientation of the robot. However, this vision system also introduces noise in the measurements. This noise was assumed to be Gaussian. Hence, the observations can be modeled by Eq. (14).

$$z_k =
\begin{bmatrix}
x_k \\
y_k \\
\psi_k
\end{bmatrix} +
\begin{bmatrix}
r_{k, x} \\
r_{k, y} \\
r_{k, \psi}
\end{bmatrix} (14)$$

4 Simulation and Results

4.1 Simulation Setup

In all simulations described in this paper, the positions were measured in cm and the angles in rad. The parameter $T_s$ was set 1/30 s, the velocities inputs were constant and equal to $v = 1$ cm/s and $w = 0.05$ rad/s. The initial state was generated from a Gaussian distribution with mean $s_0 = [0 \ 0 \ 0]^T$ and covariance matrix $P_0 = diag(1, 1, 0.01)$. The noise $u_k$ had zero mean and constant covariance matrix $Q_k = (T_s)^2 diag(0.05, 0.05, 0.0001)$. The noise $v_k$ also had zero mean and constant covariance matrix $R_k = diag(1, 1, 0.01)$. All the simulations have total time of 100 s and in each RMSE the calculations were made considering 1000 realizations.

The results will present the RMSE both in position and orientation, calculated using Eqs. (15) and (16), over all time instants, $k$, and the $N_r$ realizations.

$$RMSE_k^p = \frac{1}{N_r} \sum_{m=0}^{N_r} (\hat{x}_k^{(m)} - x_k^{(m)})^2 + (\hat{y}_k^{(m)} - y_k^{(m)})^2$$

$$RMSE_k^o = \frac{1}{N_r} \sum_{m=0}^{N_r} (\hat{\psi}_k^{(m)} - \psi_k^{(m)})^2$$ (16)

4.2 Results

All the filters described in Sec. 2 were implemented. Fig. 1 graphically represents one realization of the simulation. It compares the estimated position using one of those techniques and the simulated position generated using the setup described in Subsection 4.1.

Initially, the Bootstrap performance is simulated for four different values of number of particles (Np). The RMSE is presented in Figures 2 and 3.

In order to compare the Bootstrap with other filters, Figures 4 and 5 show only the implementation with 300 particles. These Figures also show the RMSE for the other filters implemented in this paper.
Analyzing Figures 2 to 5, it is possible to conclude that, for this model, EKF and UKF have very similar performances in terms of RMSE. The Bootstrap filter has also presented similar performance for number of particles greater than 300. However, the processing time of Bootstrap filters is significantly higher than the measured using EKF and UKF. Therefore, for the model described in this paper, Bootstrap would not be a good choice in terms of performance.

Comparing EKF and UKF, it can be noted that EKF is less computationally expensive, in terms of processing time. Hence, for this model, EKF would be the best filtering technique between the ones presented in this paper.

5 Conclusions and Future Works

In this problem, the models do not present strong nonlinearities. The main contribution of this work is showing that the nonlinear extensions of the Kalman Filter outperform Particle Filters for the model described in Sec. 3, which has only weak nonlinearities. This result contrasts with the ones presented in (Silva and Bruno, 2007) and (Gutmann and Fox, 2002), which have concluded
that, for their models, Particle Filters presents a better performance in terms of RMSE. If the model used in this paper had strong nonlinearities, particle filters would present a better performance.

Despite having a larger processing time, UKF has some advantages over EKF. For example, if it is desired to extend the number of states, it would take almost no effort to redesign and implement UKF. To redesign EKF, it would be necessary to recalculate the derivatives of both the prediction and observation models, which can be difficult, depending on the model. Since UKF has an affordable processing time for real time applications, it should also be considered while developing the localization system.

In future works, we intend to experimentally obtain the model parameters described in Sec. 3 and implement those algorithms in ITAndoids’ VSS team. Then, it would be possible to also evaluate the performance of these techniques on real robots.

6 Acknowledgements

The authors would like to thank the CNPq and ITAndoids’ sponsors (CESAR, Poliedro and Mecatron Odebrecht) for their financial support. We also acknowledge all the help provided by ITAndroids.

References


