Robotica (2008) volume 26, pp. 571–585. © 2008 Cambridge University Press
doi:10.1017/S0263574708004153 Printed in the United Kingdom

A comparison of Bayesian prediction techniques for mobile robot trajectory tracking[1]

J. L. Peralta-Cabezas*, M. Torres-Torriti* and M. Guarini-Hermann

Department of Electrical Engineering, Pontificia Universidad Católica de Chile, Vicuña Mackenna 4860, Casilla 306-22, Santiago, Chile.

(Received in Final Form: December 28, 2007. First published online: March 4, 2008)

SUMMARY

This paper presents a performance comparison of different estimation and prediction techniques applied to the problem of tracking multiple robots. The main performance criteria are the magnitude of the estimation or prediction error, the computational effort and the robustness of each method to non-Gaussian noise. Among the different techniques compared are the well-known Kalman filters and their different variants (e.g. extended and unscented), and the more recent techniques relying on Sequential Monte Carlo Sampling methods, such as particle filters and Gaussian Mixture Sigma Point Particle Filter.

KEYWORDS: Mobile robot tracking; Latency compensation; Bayesian prediction; Kalman filtering; Monte Carlo methods; Sigma-Point filters.

1. Introduction

Filtering and prediction methods to accurately track multiple moving objects under noisy measurements, disturbances and model uncertainty have received much attention during the last decades in a wide variety of fields, such as aeronautics, defence, computer vision, robotics and economics, to name a few.[1–8] It is a well-known fact that the ability of any control or decision system to meet its targets directly depends on accuracy of the measurements and the state estimation.[9] On the other hand, when the measurements of the controlled variables are available with some delay, the ability to make accurate predictions of the measurements becomes particularly important in order to compensate the measurements' latency.[10–12]

In view of the large number of existing approaches (see references in Table I) and the key role of filtering and estimation techniques for accurate tracking, it is important to identify those techniques that have the best performance, not only in terms of accuracy, but also in terms of their ability to handle non-Gaussian noise and their computational cost. The aim of the present work is thus to evaluate the performance of classic filtering techniques for trajectory prediction and perception-latency compensation, such as the Kalman Filter (KF)[15] and the Extended Kalman Filter (EKF),[16] as well as more recent ones, such as the Unscented Kalman Filter (UKF),[17] Particle Filtering (PF)[2] and other variants in the class of the so-called Sigma-Point Kalman Filters (SPKF).[10,18,19] The prediction strategies compared in this paper, which are summarized in Table I, can be grouped into two main families: (i) the Kalman Filter based approaches[10,15–20] and (ii) the sequential sampling or particle filter based approaches.[2,7,21,22]

Although there exist many studies and comparisons in the literature,[3,10,18,24–26] most of these compare just a few approaches and in general do not consider directly issues related to the robustness of the methods to non-Gaussian noise. The Gaussianity of the disturbances is a common assumption that often simplifies analytic developments, but is far from the actual characteristis of many real systems. In fact, some of the existing algorithms yield poor estimates or simply diverge when the Gaussianity assumption is not satisfied, while others require a significantly large computation time that may render them unpractical for real-time control and decision applications.[10] The KF and EKF are unarguably the most popular prediction approaches.[2] However, both rely on a first order Taylor series approximation of the nonlinear state space model of the system. Furthermore, both assume that process and measurement noises are identically and independently Gaussian distributed. Because of these assumptions, the KF or EKF perform poorly when the system dynamics is highly nonlinear and the system noises are non-Gaussian. These limitations have motivated the development of the Sigma-Point Kalman Filters (SPKF)[10] and the rebirth of the computationally intensive Sequential Monte Carlo Sampling methods[7] aided by the availability of more powerful computers.

The main contribution of this paper is the evaluation and comparison of the performance of the 10 different filtering strategies listed in Table I as applied to a nonlinear system with measurements subject to non-Gaussian noise. The performance of each filter is measured in terms of the root mean squared-error of the filter's prediction error, the filter's effectiveness to handle non-Gaussian noise and its computational cost. Concisely, but in a precise manner, the paper also presents the general algorithm of the KF-based and the PF-based filtering approaches. The system employed as test bench is an ensemble of fast moving RoboCup F-180 robots.[11] The performance of the filters is first evaluated through numerical simulations

* Corresponding authors. E-mail: jlperalt@puc.cl; mtorrest@ing.puc.cl

[1] This work was supported by Conycit of Chile under Fondecyt Grant 11060251.

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

Table I. Summary of Bayesian prediction techniques compared in the present work.

Filter	Abrev.	Noise assumption	Linearization technique	Reference
Extended Kalman Filter	EKF	Gaussian	1st Order Taylor	15, 16
Unscented Kalman Filter	UKF	Gaussian	None	17
Central Difference Kalman Filter	CDKF	Gaussian	2nd Order Stirling	18
Divided Difference 1st Order Kalman Filter	DD1	Gaussian	1st Order Stirling	19
Divided Difference 2nd Order Kalman Filter	DD2	Gaussian	2nd Order Stirling	19
Square Root Unscented Kalman Filter	SRUKF	Gaussian	None	10, 20
Square Root Central Difference Kalman Filter	SRCDKF	Gaussian	2nd Order Stirling	10, 20
Generic Particle Filter	PF	none	None	2, 7
Sigma-Point Particle Filter	SPPF	Gaussian	None (w/SRUKF), 2nd Order Stirling (w/SRCDKF)	21
Gaussian Mixture Sigma Point Particle Filter	GMSPPF	Gaussian	None (w/SRUKF), 2nd Order Stirling (w/SRCDKF)	22, 23

of the robots' trajectory prediction. The best performing approach is then tested with the real robots. In this context, the performance is discussed in terms of the ability of the estimation strategies to filter perception errors and compensate perception delays in real time. The results are valuable not only to applications related to robotic soccer competitions, but to a wide range of fields characterized by nonlinear dynamics/non-Gaussian disturbances that require accurate state trajectory predictions and the compensation of perception and computation delays. Examples of these fields include the process control industry,[9] the visual tracking applications[27,28] and the autonomous localization and navigation.[10,12,13]

The paper is organized as follows. Section 2 presents the mathematical background on which the KF-based and PF-based filtering approaches rely. The derivation of the kinematic and dynamic equations of a three-wheel omni-directional robot employed in the filters implementation is presented in Section 3. The testing methodology and results are presented in Sections 4 and 5, respectively. The conclusions of this work and a discussion of some aspects concerning ongoing research are presented in Section 6.

2. Bayesian Estimation

In order to define the problem of Bayesian estimation, let $x_k \in \mathbb{R}^n$ denote the n-dimensional vector representing the state of some system at time $t_k = kT$, $k \in \mathbb{N}$, where $T > 0$ is the sampling period (assumed constant for simplicity of exposition), and consider the evolution of the state according to a discrete-time stochastic *model equation*:

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, v_{k-1}) \tag{1}$$

where f_{k-1} is a known function of the state x_{k-1}, an m-dimensional input vector $u_{k-1} \in \mathbb{R}^m$, whose purpose is to drive the state of the system, and an n-dimensional process noise $v_{k-1} \in \mathbb{R}^n$ that accounts for external disturbances and mismodelling effects. The function f_{k-1} is often nonlinear and also may be time-dependent.

The optimal estimation problem consists in finding the best estimate of the state x_k of the stochastic process (1) whose actual state cannot be directly observed, but can be inferred from the measurements of the system outputs $y_k \in \mathbb{R}^p$. The later are related to the system's state by the so-called *measurement equation*, which is also known as *sensor or observation model*:

$$y_k = h_k(x_{k-l}, w_{k-l}) \tag{2}$$

where h_k is a known, possibly nonlinear and time-dependent, function of the state x_{k-l}, at some time $t_{k-l} = (k-l)T$, and a p-dimensional measurement noise $w_{k-l} \in \mathbb{R}^n$ vector that accounts for sensor noise and measurement uncertainties. Here $l \in \mathbb{N}$ represents a possibly non-zero delay introduced by limitations in the speed at which measurements can be acquired.

In the Bayesian formulation, the optimal state estimate at time t_k, often denoted by $\hat{x}_k$, is the one that maximizes the *a posteriori* conditional probability of the state having certain value given the system output measurements. More rigorously, if $Y_k \stackrel{def}{=} \{y_i, i = 1, 2, \ldots, k\}$ denotes the set of all sensor measurements up to time t_k, then the optimal state estimate in the Bayesian framework is the one that maximizes $p(x_k|Y_k)$, or equivalently, that which minimizes the expected error between the actual state x_k and the estimate $\hat{x}_k$. Such optimal estimate is uniquely specified by the *conditional mean* or *minimum variance* estimate (see Theorem 3.1, in Ref. 14) given by

$$\hat{x}_k \stackrel{def}{=} E[x_k \mid Y_k] = \int x_k p(x_k \mid Y_k)\, dx_k. \tag{3}$$

Central to the calculation of the optimal estimate is the calculation of the posterior probability density $p(x_k \mid Y_k)$, which by Bayes' rule can be computed recursively as new measurements arrive according to

$$p(x_k \mid Y_k) = C \cdot p(y_k \mid x_k)\, p(x_k \mid Y_{k-1}), \tag{4}$$

where

$$p(x_k \mid Y_{k-1}) = \int p(x_k \mid x_{k-1})\, p(x_{k-1} \mid Y_{k-1})\, dx_{k-1}$$

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

and

$$C = \left(\int p\left(y_k \mid x_k\right) p\left(x_k \mid Y_{k-1}\right) dx_k \right)^{-1}.$$

If the process is governed by a linear stochastic difference (or differential) equation with Gaussian noises, then the optimal estimate (3) can be computed recursively using the well known Kalman Filter.[15,14] However, most real world systems are not linear and are subject to non-Gaussian noises. Thus analytical closed form expressions for the estimate (3) usually do not exist, not even in recursive form, and only suboptimal solutions may be computed to approximate in some sense the optimal solution. Among these approximated solutions are the Extended Kalman Filter (EKF),[2,16] Gaussian Sum Filters (GSF),[22,23] Sigma-Point Kalman Filters (SPKF)[10] and Sequential Monte Carlo (SMC) methods that give origin to the Particle Filter (PF) and its variants.[2,21–23]

2.1. Kalman filter approaches

The Kalman Filter (KF) and its extended version (EKF) are perhaps the most popular prediction approaches employed in a variety of tracking applications.[2,14–16] Both, the KF and the EKF, rely on a first order Taylor series approximation of the nonlinear state space model of the system, and assume that process and measurement noises are identically and independently Gaussian distributed. These assumptions allow the filters to be implemented in a relatively simple recursive way in terms of a *state prediction step* and an *optimal estimate update* or *correction step* as shown in Algorithm 1. However, the main disadvantage of such filtering approaches is their inability to handle systems with non-Gaussian noises and highly nonlinear dynamics, often yielding diverging predictions of the system's state. This has lead to the recent development of the so-called Sigma-Point Kalman Filters (SPKF),[10] which include the UKF,[17] CDKF,[18] DD1,[19] DD2,[19] SRCDKF[10] and SRUKF[20] approaches. Some examples of their applications can be found in refs. [10] and [23].

The SPKF filters do not require the calculation of the system's Jacobians F_x, F_v (see Algorithm 1), instead samples (sigma points) are generated in a deterministic fashion so that some properties (such as first and second statistical moments) match those of the prior distribution. Each point undergoes a transformation determined by the system's model equations or some approximation of the model. The properties of the posterior distribution are obtained from the properties of the transformed sigma points. The transformation in Julier's UKF is given directly by the model equations, while Ito's CDKF and Nørgaard's DD2 employ a second order Stirling polynomial interpolation formula to approximate the model equations, thereby better retaining second order properties of the samples than the standard EKF and with a potentially smaller computational cost than that of the UKF. The difference between the DD1 and DD2 filters is that the former simplifies the model approximation by using a first order Stirling polynomial interpolation formula instead of a second order one. This simplification allows for some computational time reductions at the expense of accuracy. Ito's CDKF approach and Nørgaard's DD1, DD2 approaches were developed independently, but are conceptually the same with minor implementation differences. Later Van der Merwe developed the SRCDKF and SRUKF approaches, suggesting the use of the QR and Cholesky decomposition to compute the square root of the state covariance matrix P_k. The square root of P_k is required to generate the sigma points in the CDKF, DD1, DD2 or UKF approaches and is one of the most computationally expensive steps of the SPKF filters. Hence, SRCDKF and SRUKF aim at improving the speed and numerical stability of the CDKF and UKF by placing special attention to the efficient calculation of the square root of the state covariance matrix.

It must be stressed that no random sampling is used by the SPKF approaches, despite their resemblance to a Monte Carlo method. Thus only a small number of samples are required to propagate means and covariances. Because of this, SPKF filters have much smaller computational costs than filters based on Monte Carlo methods.[10] On the other hand, SPKF filters should yield smaller estimation errors than the EKF because of their better approximation of the system's model equations and second order statistical moments. Much like the KF or EKF, the SPKF approaches cannot handle non-Gaussian noises effectively. The Particle Filters discussed next may overcome this disadvantage.

Algorithm 1: Extended Kalman Filter (EKF)

Initial asumptions/conditions:

Initial state estimate: $\hat{x}_0 = E[x_0]$

Initial state covariance: $P_0 = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$

Process noise covariance: $Q_0 = E[(v_0 - \bar{v}_0)(v_0 - \bar{v}_0)^T]$

Measurement noise covariace: $R_0 = E[(w_0 - \bar{w}_0)(w_0 - \hat{w}_0)^T]$

Iteration k: For $k = 1, 2, \ldots, \infty$

1. Prediction step:

- Compute the process model Jacobians
  $F_{xk} = \nabla_x f|_{(\hat{x}_{k-1}, u_{k-1}, \bar{v}_{k-1})}$, $F_{uk} = \nabla_u f|_{(\hat{x}_{k-1}, u_{k-1}, \bar{v}_{k-1})}$, $F_{vk} = \nabla_v f|_{(\hat{x}_{k-1}, u_{k-1}, \bar{v}_{k-1})}$.
- Compute the predicted state mean and covariance
  State prediction: $\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, \bar{v}_{k-1})$
  State covariance prediction: $P_k^- = F_{xk} P_{k-1} F_{xk}^T + F_{vk} Q_k F_{vk}^T$.

2. Correction step:

- Compute the observation model Jacobians
  $H_{xk} = \nabla_x h|_{(\hat{x}_k^-, \bar{w}_k)}$, $H_{wk} = \nabla_w h|_{(\hat{x}_k^-, \bar{w}_k)}$.
- Update the state estimate with the lastest observation (measurement update)
  Kalman gain: $K_k = P_k^- H_{xk}^T \left(H_{xk} P_k^- H_{xk}^T + H_{wk} R_k H_{wk}^T \right)^{-1}$
  State estimate: $\hat{x}_k = \hat{x}_k^- + K_k[y_k - h(\hat{x}_k^-, \bar{w}_k)]$
  State covariance: $P_k = (I - K_k H_{xk}) P_k^-$.

2.2. Particle filter approaches

Particle Filters (PF) belong to the class of Sequential Monte Carlo (SMC) methods, which rely on the idea of recursively computing the posterior probability distributions using concepts of importance sampling and the approximation of distributions by discrete random measures (particles) from the space of unknown states.[2,7,21,22] The application of SMC

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

methods dates back to the 1950s,[29] even before the development of the KF; however, it has not been until the last decade that PF have received significant attention,[2,7,30] mainly because the computational complexity involved demands adequate computing resources that did not exist before.

The main steps of the PF computation are shown in Algorithm 2. Whereas KF/EKF and SPKF methods assume that noises are Gaussian and independently identically distributed (i.i.d.) in order to simplify the formulation of the optimal recursive Bayesian estimator, PF methods make no assumptions on the form of the probability densities. Moreover, the particles (samples) interact and thus PF methods may take into account the statistical dependence of the samples. Therefore, PF methods present a greater potential to successfully handle strongly nonlinear, non-Gaussian processes. The shortcoming of SMC approaches is the high computational effort that they require due to the large number of samples needed to obtain accurate state estimates. The number of samples may grow exponentially with respect to the dimension of the state space, especially for the standard PF.[2] To overcome this disadvantage, several 'hybrid' methods have been proposed to generate the particles and weights based on KF techniques and its extensions. These hybrid methods allow to obtain accurate state estimates without having to generate a large number of samples, but yet again do not perform well if noises do not conform to the Gaussian assumption.

A method to deal with non-Gaussian noises by discarding only those measurements that are deemed unlikely to occur under the Gaussian noise assumption has been proposed by Browning *et al*.[33] The method is referred to as Improbability Filter (IF), and can be employed in combination with the KF or the PF methods that cannot handle non-Gaussian noises properly. The next section briefly explains the IF because of its usefulness towards improving the estimates' accuracy.

Algorithm 2: Particle Filter (PF)

Initialization:

For $i = 1, 2, \ldots, N$ draw samples (particles) $x_0^{(i)}$ from the prior $p(x_0)$.

Iteration k: For $k = 1, 2, \ldots, \infty$

1. Importance Sampling Step:

For $i = 1, 2, \ldots, N$:

- Draw samples $x_k^{(i)} \sim p\left(x_k \mid x_{k-1}^{(i)}\right) \leftarrow f\left(x_{k-1}^{(i)}, u_{k-1}, v_{k-1}^{(i)}\right)$.
- Evaluate the importance weights up to a normalizing constant: $\tilde{w}_k^{(i)} = w_{k-1}^{(i)} p\left(y_k \mid x_k^{(i)}\right) = w_{k-1}^{(i)} p_w\left(y_k - h\left(x_k^{(i)}\right)\right)$, asuming $y_k = h(x_k) + w_k$, $w_k \sim p_w$.
- Normalize the importance weights: $w_k^{(i)} = \dfrac{\tilde{w}_k^{(i)}}{\sum_{j=1}^{N} \tilde{w}_k^{(i)}}$.

2. Selection Step (Resampling):

- Multiply/suppress samples $x_k^{(i)}$ with high/low importance weights $w_k^{(i)}$ to obtain N random samples approximately distributed according to $p(x_k \mid Y_k)$.
- For $i = 1, 2, \ldots, N$ set $w_k^{(i)} = N^{-1}$.

3. Output:

- Using the samples, the state estimate is calculated as the samples expected value $\hat{x}_k = E\left(x_k \mid Y_k\right) \approx \frac{1}{N} \sum_{i=1}^{N} x_k^{(i)}$.

2.3. Improbability filter

Browning's Improbability Filter (IF) was proposed as a simple technique to prevent abnormal observations from degrading the state estimates while preserving a good filter response. The implicit assumption of this filter is that the measurements are locally Gaussian with mean given by the predicted measurement, $\hat{y}_k = h_k(\hat{x}_k, \bar{w}_k)$, and measurement covariance matrix $S_k = H_{xk} P_k H_{xk}^T + H_{wk} R_k H_{wk}^T$. With this assumption, a criterion to discard *unlikely* measurements is to reject those observations y_k whose conditional probability of occurrence given $\hat{y}_k$ and S_k

$$P\left(y_k \mid \hat{y}_k,\ S_k\right) = \frac{1}{(2\pi\ |S_k|)^{\frac{n}{2}}} \cdot e^{-\frac{1}{2}(y_k - \hat{y}_k)^T S_k^{-1} (y_k - \hat{y}_k)}, \quad (5)$$

falls below certain threshold. An equivalent criterion is to use the Mahalanobis distance

$$d_M(y_k; \hat{y}_k, S_k) = (y_k - \hat{y}_k)^T\ S_k^{-1}\ (y_k - \hat{y}_k)\,, \quad (6)$$

in order to discard those observations y_k whose distance with respect to the predicted measurement $\hat{y}_k$ exceeds a given threshold. If an observation is discarded, the Improbability Filter assumes that the predicted measurement is the correct measurement and no state update is performed. However, state covariances are updated through the dynamic update equation. If measurements are repeatedly rejected, the state covariances P_k will grow until observations become likely and are accepted. Furthermore, the large covariances will make the state estimate to converge immediately according to the new observations.

2.4. Differences between KF, PF and IF

Concisely, the difference between KF and PF approaches lies in the fact that KFs assume the noises have a Gaussian distribution and that the model can be reasonably approximated by a linear Taylor series approximation, while PF make no specific assumption about the distribution of the process or measurement noises, nor do they necessarily make any approximation of the model equations. This difference is reflected by the fact that the EKF uses the Jacobians of the system Eqs. (1) and (2), (denoted by ∇ in Algorithm 1), to propagate the covariances that define the probability of the optimal state estimate $\hat{x}_k$, while the PF approaches propagate samples $x_k^{(i)}$ with transition probability $p(x_k \mid x_{k-1}^{(i)})$ defined by the model Eq. (1) and the known statistics of the process noise v_{k-1}. In simple terms, the PF requires generating a number N of possible states, which after undergoing a transformation defined by the model Eq. (1), allow to reconstruct the posterior probability (4) using importance weights that are calculated in terms of the measurements likelihood $p(y_k \mid x_k^{(i)})$; (see

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

Algorithm 2). The KF filters propagate only one state, whose probability of being the actual state is given by a Gaussian distribution with mean $\hat{x}_k$ and covariance P_k resulting from the update equations; (see Algorithm 1). It is to be noted that the UKF algorithm involves analogous steps to those of Algorithm 1, but does not require the computation of the Jacobians because the model Eq. (1) is employed to deterministically generate some samples that preserve first and second order statistics. Other SPKF algorithms, such as CDKF, DD1, DD2, SRCDKF and SRUKF are similar to UKF in the sense that the statistics information is also carried by a number of particles that are deterministically chosen, but also rely on first or second order model approximations.

From an implementation perspective, the computational complexity of the KF approaches per recursion is of order $\mathcal{O}(n^3 + m^3 + 2n^2m + 2nm^2 + c_0)$, where n and m are the dimensions of the state space model and the measurement model, respectively[31], and c_0 is the cost of computing the gradients or model equations. On the other hand, the PF methods require a computational complexity which is roughly of order $\mathcal{O}(\frac{4}{3}n^3 + 2n^2) + N \cdot \mathcal{O}(4n^2 + n + c_1 + c_2 + nc_3)$, where n is the dimension of the state space, N is the number of particles, c_1 is the cost of the calculation of the likelihood function, c_2 is the cost of the resampling step and c_3 is the complexity of the random number generator.[32] Assuming $n = m$, for large n the complexity of KFs is roughly $\mathcal{O}(6n^3)$, while that of PFs is $\mathcal{O}(4Nn^2) + \mathcal{O}(\frac{4}{3}n^3) \gg \mathcal{O}(6n^3)$ since $N \gg n$. It is hard to make an exact comparison because the complexity may vary depending on the level of nonlinearity exhibited by the system equations and whether first or second order approximations are employed. However, it is to be noted that the main costs of the KF arise from the computation of the Riccati equation for matrix P_k^- due to the matrix multiplication $F_{x_k} P_{k-1}$, which is $\mathcal{O}(2n^3)$. The general PF does not involve the matrix times matrix multiplication, instead only $\mathcal{O}(n^2)$ matrix times vector multiplications $F_{x_k} x_k$ are required to generate the samples N times, with N several times larger than n. It is also worth pointing that even if the relationship between the theoretical complexity (expressed in terms of the number of floating-point operations) and the actual computation time is not straightforward because of the influence of memory locality and cache boundaries, the results nonetheless confirm that PF-based approaches are computationally more time consuming than KF-based methods.

Finally, it must be noticed that the IF is not a recursive Bayesian estimator, but it exploits the assumption that the measurement noises are Gaussian to discard those measurements which deviate from the expected measurement and whose probability in the Gaussian sense is low given the knowledge of the updated measurement covariance. Using the IF in conjunction with KF, EKF or other SPKF approaches makes sense because the IF should be able to reject measurements that are not Gaussian and that would otherwise deteriorate the prediction and update of the covariances. Since the PF methods do not require to assume Gaussian noises norlinear models, using the IF with the PF approaches should not offer any benefit and just increase the computational cost.

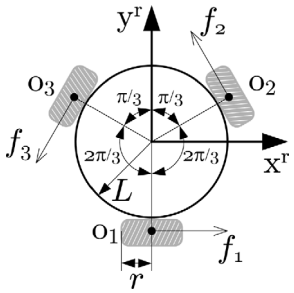


Fig. 1. Robot coordinate frame $\{\mathbf{x}^\mathbf{r}, \mathbf{x}^\mathbf{r}\}$ fixed to the robot's body (local coordiantes) and the traction forces f_1, f_2 and f_3 of the wheels.

3. Omni-Directional Robot Model

The performance of the filtering techniques mentioned in the previous section, as applied to the position tracking and prediction problem, is evaluated using a three-wheel omni-directional robot as test platform. It is worth pointing out that the particular choice of model does not limit the validity of the performance tests. If a robot with a different configuration is employed, then the model equations should be updated accordingly. The choice of the three-wheel configuration is based on the fact that: (i) it possesses a nonlinear dynamics which offers an adequate challenge to the filtering techniques, (ii) it is simple to control as there are less actuators and (iii) it is the type of robot that is currently being employed by the PUC's F-180 RoboCup team.

The kinematic and dynamic model is presented next following the work by Liu *et al.*;[34] for a detailed and rigorous derivation the reader is referred to the book by Angeles.[35] For clarity of exposition, the notation employed in the formulation of the robot's model is summarized in Table II. The model equations are derived by expressing velocities and forces acting on the robot in terms of their components along the local coordinate system, $\{\mathbf{x}^\mathbf{r}, \mathbf{y}^\mathbf{r}\}$, which is fixed to the robot's body as illustrated in Fig. 1. Simple vector rotation operations allow to rewrite the robot's kinematic and dynamic equations in terms of the global coordinate system, $\{\mathbf{x}^\mathbf{w}, \mathbf{y}^\mathbf{w}\}$, which is fixed to the court as shown in Fig. 2.

The following simplifying assumptions regarding the system have been made:

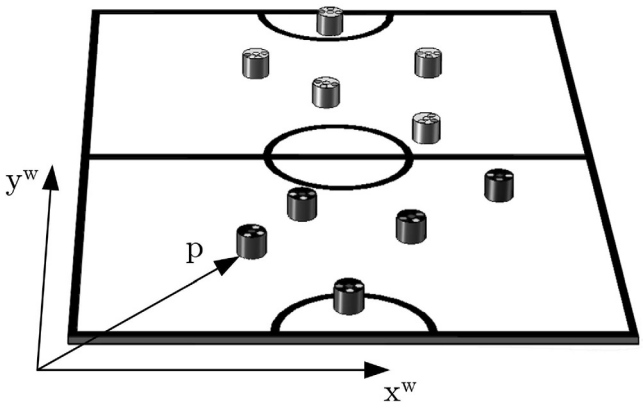


Fig. 2. World coordinate frame $\{\mathbf{x}^\mathbf{w}, \mathbf{y}^\mathbf{w}\}$ fixed to the court (global coordinates) and two teams of RoboCup F-180 robots.

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

Table II. Notation for the model equations of the three-wheel robot.

Symbol	Description
Robot parameters	
$\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3$	Rear, front-right and front-left wheels
m	Robot's mass
I_0	Robot's inertia moment
r	Wheel radius
L	Body radius
N	Gear ratio
J_0	Motor armature and wheel inertia moment
K_T, K_G, R_a, L_a	Motor constants (torque, EMF, resistance and inductance)
Robot local variables (body frame)	
$\mathbf{x}^\mathbf{r}, \mathbf{y}^\mathbf{r}$	Axes of the local coordinate frame fixed to the robot's body
$F_{\mathbf{x}^\mathbf{r}}, F_{\mathbf{y}^\mathbf{r}}$	Traction forces acting on the robot along $\mathbf{x}^\mathbf{r}$ and $\mathbf{y}^\mathbf{r}$
$\dot{x}^r, \dot{y}^r$	Robot velocities along $\mathbf{x}^\mathbf{r}$ and $\mathbf{y}^\mathbf{r}$
τ_θ	Body rotation torque
$\dot{\theta}$	Robot's heading rate of change
f_1, f_2, f_3	Traction forces of the wheels
$\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3$	Angular velocities of the motors
u_1, u_2, u_3	Input voltages of the motors
Robot world variables (court frame)	
$\mathbf{x}^\mathbf{w}, \mathbf{y}^\mathbf{w}$	Axes of the global coordinate frame fixed to the court
$\mathbf{p} = (x^w, y^w)$	Robot location
θ	Robot heading (defined as the angle between the robot's $\mathbf{y}^\mathbf{r}$ axis and the global $\mathbf{x}^\mathbf{w}$ axis)

1. The robot body only undergoes planar motion and there is no wheel slippage nor skidding.
2. Dynamic friction on the robot is negligible and only viscous friction in the motors and bearings is considered.
3. The mass and moment of inertia of the wheels about the $\mathbf{z}^\mathbf{r}$ axis perpendicular to $\mathbf{x}^\mathbf{r}$ and $\mathbf{y}^\mathbf{r}$ have been lumped into a mass term m and a general inertia I_0.

By considering these standard assumptions, additional uncertainty about the true model parameters is introduced. However, this uncertainty is overcome by the estimation strategy. The increased uncertainty poses an additional *desirable challenge* to the algorithms that are to be evaluated in the following section.

3.1. Robot kinematics

Defining the matrices

$$P_0 \stackrel{def}{=} \begin{bmatrix} \sin(\theta) & \cos(\theta) & 0 \\ -\cos(\theta) & \sin(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$P_1 \stackrel{def}{=} \begin{bmatrix} 1 & 0 & L \\ -\sin(\psi) & \cos(\psi) & L \\ -\sin(\psi) & -\cos(\psi) & L \end{bmatrix}, \text{ with } \psi = \frac{\pi}{6}.$$

it is easy to verify from the geometry of the robot shown in Fig. 1 and Fig. 3, that the robot's velocity components $[\dot{x}^r, \dot{y}^r, \dot{\theta}]$ in the local frame $\{\mathbf{x}^\mathbf{r}, \mathbf{y}^\mathbf{r}\}$ are related to the wheel velocities according to:

$$P_1 \mathbf{v}^\mathbf{r} = \frac{r}{N}\dot{\boldsymbol{\phi}}. \tag{7}$$

where $\mathbf{v}^\mathbf{r} = [\dot{x}^r, \dot{y}^r, \dot{\theta}]^T$, $\dot{\boldsymbol{\phi}} = [\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3]^T$. The kinematics equation defining robot's velocity components vector in the global coordinates, $\mathbf{v}^\mathbf{w}$, as a function of the wheels' angular velocities is then given by

$$\mathbf{v}^\mathbf{w} = P_0 \mathbf{v}^\mathbf{r} = \frac{r}{N} P_0 P_1^{-1} \dot{\boldsymbol{\phi}}. \tag{8}$$

3.2. Robot dynamics

Defining the robot's generalized inertia matrix as:

$$I \stackrel{def}{=} \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_0 \end{bmatrix},$$

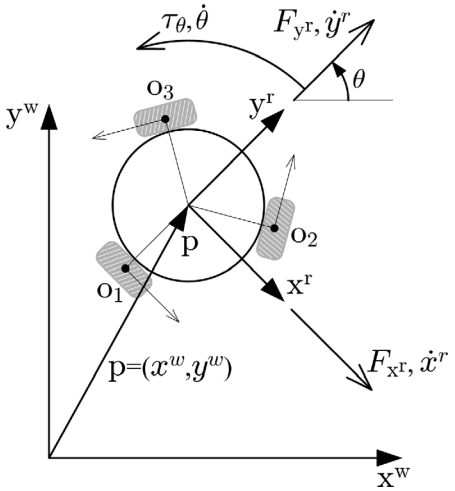


Fig. 3. Forces acting on the robot and velocity components (see Table II for notation details).

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

and applying Newton's Law in the body frame

$$I\dot{\mathbf{v}}^{\mathbf{r}} = I\dot{\theta}\mathbf{z}^{\mathbf{r}} \times \mathbf{v}_{\mathbf{p}}^{\mathbf{r}} + P_1^T\mathbf{f}, \tag{9}$$

where $\mathbf{v}_{\mathbf{p}}^{\mathbf{r}} = \dot{x}^r\mathbf{x}^{\mathbf{r}} + \dot{y}^r\mathbf{y}^{\mathbf{r}}$ is the velocity of the robot center of mass $\mathbf{p}$ in the $\mathbf{x}^{\mathbf{r}}$ and $\mathbf{y}^{\mathbf{r}}$ directions, and $\mathbf{f} = [f_1, f_2, f_3]^T$ is the vector of net traction forces of the wheels.

The dynamics of the DC motors is described by the equations:

$$J_0\ddot{\boldsymbol{\phi}} = K_T\mathbf{i}_a - b_0\dot{\boldsymbol{\phi}} - \frac{r}{N}\mathbf{f}, \tag{10}$$

$$\mathbf{u} = R_a\mathbf{i}_a + L_a\frac{d\mathbf{i}_a}{dt} + K_G\dot{\boldsymbol{\phi}}, \tag{11}$$

where $\mathbf{i}_a = [i_{a1}, i_{a2}, i_{a3}]$ is the vector of armature currents, $\mathbf{u} = [u_1, u_2, u_3]$ is the vector of armature voltages. Considering that the time constant of the electrical equation is very small compared to that of the mechanical equation, it is possible to neglect the dynamics of the motors' electrical circuits, i.e. $d\mathbf{i}_a/dt = 0$, $\mathbf{i}_a = (\mathbf{u} - K_G\dot{\boldsymbol{\phi}})/R_a$, and therefore, it is possible to rewrite the motors' mechanical Eq. (10) as:

$$J_0\ddot{\boldsymbol{\phi}} = K_T\frac{\mathbf{u} - K_G\dot{\boldsymbol{\phi}}}{R_a} - b_0\dot{\boldsymbol{\phi}} - \frac{r}{N}\mathbf{f}. \tag{12}$$

From (7), $\dot{\boldsymbol{\phi}} = \frac{N}{r}P_1\mathbf{v}^{\mathbf{r}}$ and $\ddot{\boldsymbol{\phi}} = \frac{N}{r}P_1\dot{\mathbf{v}}^{\mathbf{r}}$, which replaced in (12) yield:

$$\mathbf{f} = -a_1P_1\dot{\mathbf{v}}^{\mathbf{r}} + a_2P_1\mathbf{v}^{\mathbf{r}} + a_3\mathbf{u}, \tag{13}$$

where

$$a_1 = J_0\frac{N^2}{r^2},\ a_2 = -\frac{N^2}{r^2}\left(\frac{K_TK_G}{R_a} + b_0\right),\ a_3 = \frac{K_TN}{R_ar}.$$

Finally, replacing (13) in (9) results in the following equation for the robot's dynamics:

$$H\dot{\mathbf{v}}^{\mathbf{r}} = \dot{\theta}\mathbf{z}^{\mathbf{r}} \times \mathbf{v}_{\mathbf{p}}^{\mathbf{r}} + a_2I^{-1}P_1^TP_1\mathbf{v}^{\mathbf{r}} + a_3I^{-1}P_1^T\mathbf{u} \tag{14}$$

with $H \overset{def}{=} (\mathbb{I}_3 + a_1I^{-1}P_1^TP_1)$. Here, $\mathbb{I}_{3\times3}$ denotes the 3×3 identity matrix. In vector form, Eq. (14) can be written as

$$H\begin{bmatrix}\ddot{x}^r\\ \ddot{y}^r\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}\dot{\theta}\dot{y}^r\\ -\dot{\theta}\dot{x}^r\\ 0\end{bmatrix} + a_2I^{-1}P_1^TP_1\begin{bmatrix}\dot{x}^r\\ \dot{y}^r\\ \dot{\theta}\end{bmatrix} + a_3I^{-1}P_1^T\begin{bmatrix}u_1\\ u_2\\ u_3\end{bmatrix}. \tag{15}$$

3.3. Linear discrete-time state space model

In order to formulate a discrete-time model for predicting the robot's motion, let $x_k \overset{def}{=} [x_k^w, y_k^w, \theta_k, \dot{x}_k^r, \dot{y}_k^r, \dot{\theta}_k]^T$ be the state vector that describes the robot's motion at discrete time instants $t_k = kT$, $k = 0, 1, 2, \ldots$, for some sampling period $T > 0$. It is assumed that the state of the robot is affected by an i.i.d. zero-mean disturbance v_k with (constant) covariance $Q_k = E(v_kv_k^T)$. Linearization of the kinematic and dynamic Eqs. (8) and (15) about a nominal state $x_k^e \overset{def}{=} [x_k^w, y_k^w, \theta_k, 0, 0, 0]^T$, followed by discretization with sampling period T allows to obtain the following difference equation defining the evolution of the robot's state

$$x_k = A_{k-1}x_{k-1} + Bu_{k-1} + v_{k-1}, \tag{16}$$

with

$$A_j = \begin{bmatrix}\mathbb{I}_{3\times3} & P_0(\theta_j)T\\ \mathbf{0}_{3\times3} & \mathbb{I}_{3\times3} + A_{22}T\end{bmatrix},$$

$$A_{22} = \begin{bmatrix}\frac{3a_2}{2m+3a_1} & 0 & 0\\ 0 & \frac{3a_2}{2m+3a_1} & 0\\ 0 & 0 & \frac{3a_2}{I_0/L^2+3a_1}\end{bmatrix}, \tag{17}$$

and

$$B = \begin{bmatrix}\mathbf{0}_{3\times3}\\ B_2T\end{bmatrix},\ B_2 = \frac{a_3}{2m}\begin{bmatrix}2 & -1 & -1\\ 0 & -\sqrt{3} & -\sqrt{3}\\ \frac{2mL}{I_0} & \frac{2mL}{I_0} & \frac{2mL}{I_0}\end{bmatrix}. \tag{18}$$

Here $\mathbf{0}_{3\times3}$ represents the 3×3 matrix of zeroes, while $P_0(\theta_j)$ is employed to emphasize the dependence of the rotation matrix P_0 on the value of the heading angle θ_j. It is also important to point out that the inputs vector is actually defined as the vector of delayed armature voltages as $u_k = [V_{1_{k-l}}, V_{2_{k-l}}, V_{3_{k-l}}]^T$, where $V_{i_{k-l}}$, $i = 1, 2, 3$, is the armature voltage of the i-th motor delayed l sampling periods. The delay l accounts for the system's latency, mostly due to the delay arising in the image acquisition and segmentation process that is employed to obtain each measurement of the robot's position and heading.

The measurement model considers an observation vector given by

$$y_k = h(x_k, w_k) = \left[x_k^w + w_{1_k}, y_k^w + w_{2_k}, \theta_k + w_{3_k}\right]^T, \tag{19}$$

where $w_{i_k}, i = 1, 2, 3$, are the measurement noises, which are assumed to be the mixture of a Gaussian and a non-Gaussian random process, i.e. $w_{i_k} = w_{i_k}^g + w_{i_k}^n$, $i = 1, 2, 3$, where $w_{i_k}^g$ and $w_{i_k}^n$ are the Gaussian and non-Gaussian components, respectively. The Gaussian process $w_k^g = [w_{1_k}^g, w_{2_k}^g, w_{3_k}^g]^T$ is assumed to be i.i.d. for all $k \in \mathbb{N}$, with zero-mean and (constant) covariance matrix $R_k = E(w_k^gw_k^{gT})$. The non-Gaussian process $w_k^n = [w_{1_k}^n, w_{2_k}^n, w_{3_k}^n]^T$ takes values from a uniform distribution with a probability of occurrence at each instant k defined by a Bernoulli distribution.

The discrete-time state-space model defined by Eqs. (16) and (19) allows to generate the ideal and noisy trajectories for typical input commands applied to the robots during their operation. These trajectories will serve in the evaluation of the different filtering and prediction approaches as explained in the next sections.

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

4. Testing Platform and Methodology

This section briefly describes the general architecture of the RoboCup F-180 team of robots employed as testbed, the filter implementation and the testing methodology.

4.1. RoboCup F-180 system architecture

The test bench system employed to assess the prediction capability of the filters is based on the simulation of motion tracking under noisy conditions of the three-wheel omni-directional robot presented in the previous section. The best performing method is verified experimentally using real robots of the PUC RoboCup team. The team of five robots is controlled by a system whose architecture is illustrated in Fig. 4. As shown in Fig. 4, the vision system implemented in a standard PC receives images from a Samsung SCD107 digital video camera with a Sony VCL-0625 wide angle lens via an IEEE 1394 Firewire connection. The vision system extracts the position (x^w, y^w) and heading θ of the robots using a standard colour segmentation process. These measurements are then passed to the planning and control PC, which is responsible for filtering the measurements, predicting future states and generating motion commands aimed at driving certain robots to the ball and scoring. The filter is required to reduce robot misdetection and false positives that may occur due to fast motion and colour confusion. The prediction of states is fundamental due to the unavoidable delays arising mainly from actuator response times, the computation time required by the vision and control system, as well as the communication bandwidth that limits data throughput. All these delays can add up to 160 ms, which are equivalent to missing 5 video frames for a standard 30 fps video camera. The total delay is non-negligible, especially when a robot reaches its maximum velocity of 1.25 m/s, as this implies that the robot's displacement between one frame and the next processed one, 160 ms later, amounts to approximately 20 cm. This position difference is quite significant considering that the robot maximum diameter is 18 cm. For this reason, the planning and control system must receive an estimate of the future position of the robot corresponding to four frames time ahead, so that the planning and control system can issue adequate command decisions.

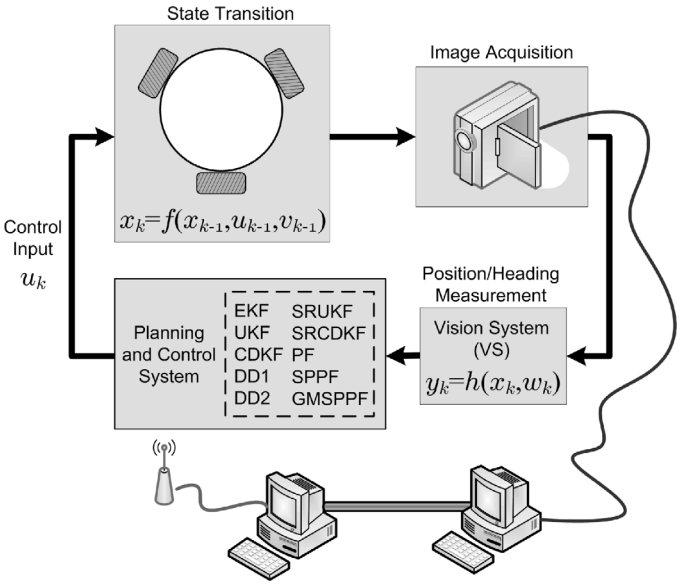


Fig. 4. PUC's RoboCup F-180 system architecture.

4.2. Filter implementation

The filters were implemented using Matlab/Simulink® and some modified functions of the ReBEL[36] and Kalmtool[37] Matlab® toolkits designed to facilitate recursive Bayesian estimation and prediction calculations.

The process noise covariance matrix was defined as $Q_k \stackrel{def}{=} \text{diag}(\sigma^2_{\dot{x}^w}, \sigma^2_{\dot{y}^w}, \sigma^2_{\dot{\theta}}, \sigma^2_{\ddot{x}^r}, \sigma^2_{\ddot{y}^r}, \sigma^2_{\ddot{\theta}})$, where diag(·) stands for the diagonal matrix of its arguments. The standard deviations for the accelerations $\ddot{x}^r$, $\ddot{y}^r$, $\ddot{\theta}$ were chosen so that their values reflect the acceleration limits imposed by the motors and wheel slippage.[33] In a similar way, the values of the standard deviations for the velocities $\dot{x}^w$, $\dot{y}^w$, $\dot{\theta}$ were defined as the largest change in velocity achievable over a sampling period T. It was verified experimentally, that the maximum robot linear acceleration was 1.5 m/s^2; therefore, $\sigma_{\ddot{x}^r} = \sigma_{\ddot{y}^r} = 2\,m/s^2$ is a reasonable choice to characterize the largest possible acceleration disturbance. The angular acceleration value is also chosen slightly above the maximum angular acceleration measured, setting $\sigma_{\ddot{\theta}} = 2\,rad/s$. With a sampling period $T = 0.0333\ s$, experimental verification lead to the following values for the velocities standard deviations: $\sigma_{\dot{x}^w} = \sigma_{\dot{y}^w} = 0.067\,m/s$, $\sigma_{\dot{\theta}} = 0.2\,rad/s$.

Similarly, the Gaussian measurement noise covariance matrix was defined as a diagonal matrix $R_k \stackrel{def}{=} \text{diag}(\sigma^2_{x^w}, \sigma^2_{y^w}, \sigma^2_{\theta})$ with $\sigma_{x^w} = \sigma_{y^w} = 0.005\,m$ and $\sigma_{\theta} = 0.020\,rad$ experimentally determined. The position and heading measurements also affected by non-Gaussian noise generated from a uniform distribution that may produce an observation corresponding to an apparent *instantaneous relocation* of the robot to any position within the court and with a possible sudden change in heading.

The initial value of the state vector is taken to be $x_0 \stackrel{def}{=} [x^w_0, y^w_0, \theta_0, \dot{x}^r_0, \dot{y}^r_0, \dot{\theta}_0]^T = [\bar{x}^w_0, \bar{y}^w_0, \bar{\theta}_0, 0, 0, 0]^T$, where $\bar{x}^w_0$, $\bar{y}^w_0$, $\bar{\theta}_0$ denote the measured initial position and heading. The state covariance matrix is initialized with the value $P_0 \stackrel{def}{=} Q_k \cdot T^2$, which corresponds to the largest expected initial position and velocities deviation after one sampling period $T = 0.0333\ s$.

4.3. Testing methodology

The ability of the different filters to accurately predict the robot's state at different time horizons is measured in terms of the error between the ideal (noise-free) trajectory and the estimated one. To this end, simulations were performed using six different reference trajectories, which are shown in Fig. 5 and that were generated using the noise-free version of the state space model of Section 3.3. The trajectories comprehend different manoeuvres that are common in RoboCup F-180 competitions, including rectilinear and circular paths with different motion regimes, such as constant acceleration, constant velocity, constant rate of turning or their combination. The test trajectories are purposely constructed to provide challenging situations in which it is more difficult to track turns or sudden changes in direction. The trajectories employed here are not intended to be optimal

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

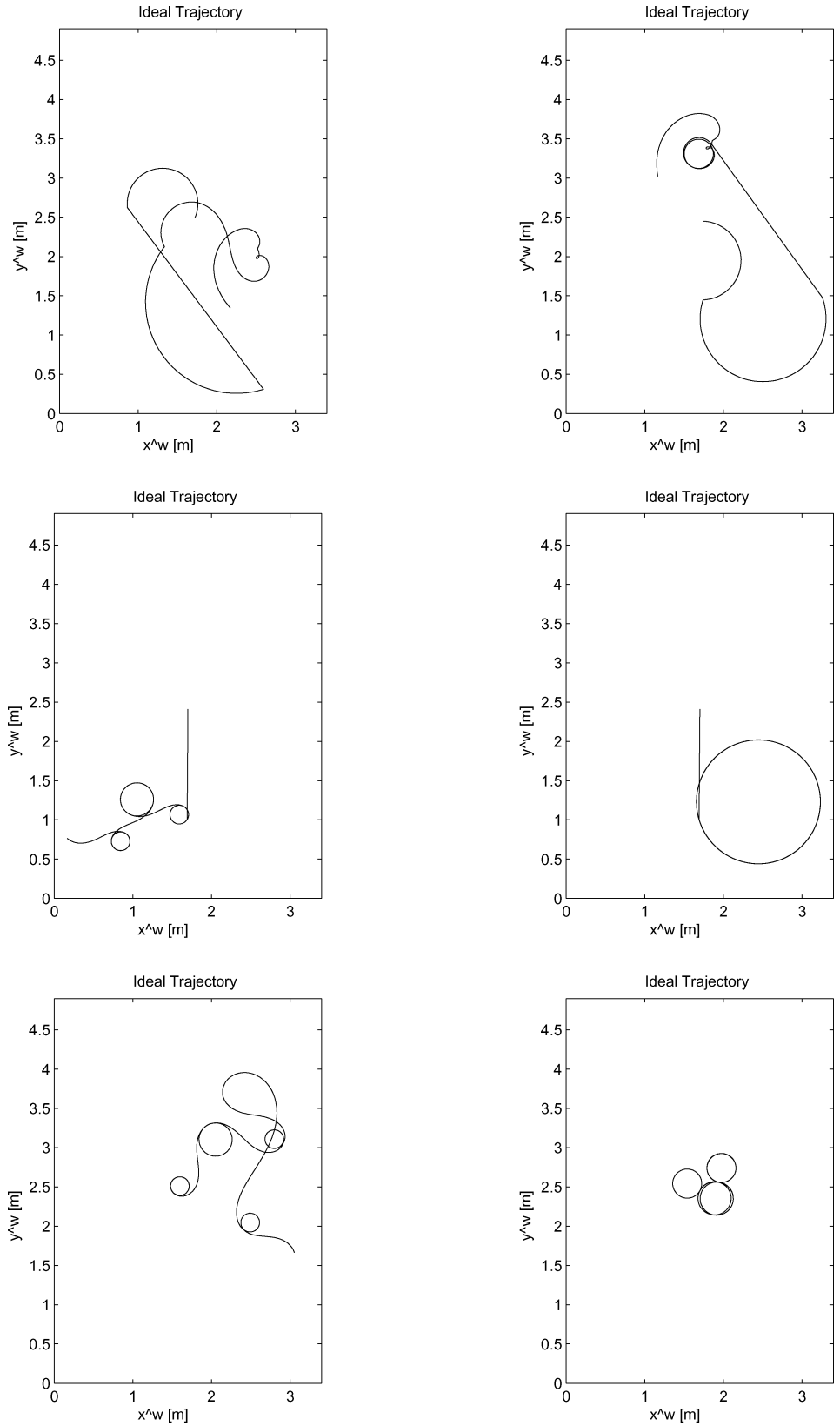


Fig. 5. Six different reference trajectories employed in the simulations.

trajectories, although our trajectory planner does solve an optimization problem in real-time to dynamically find the best path that avoids the robots of the other team.

Twenty simulation runs were repeated for each of the six reference trajectories using a new random number generator seed each time. Figure 6 shows 4 of the 20 simulation runs corresponding to the reference trajectory in the first row, second column of Fig. 5. In the noisy trajectories of Fig. 6, it is possible to appreciate the non-Gaussian noise in the observations due to erroneous detections, which appear as

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

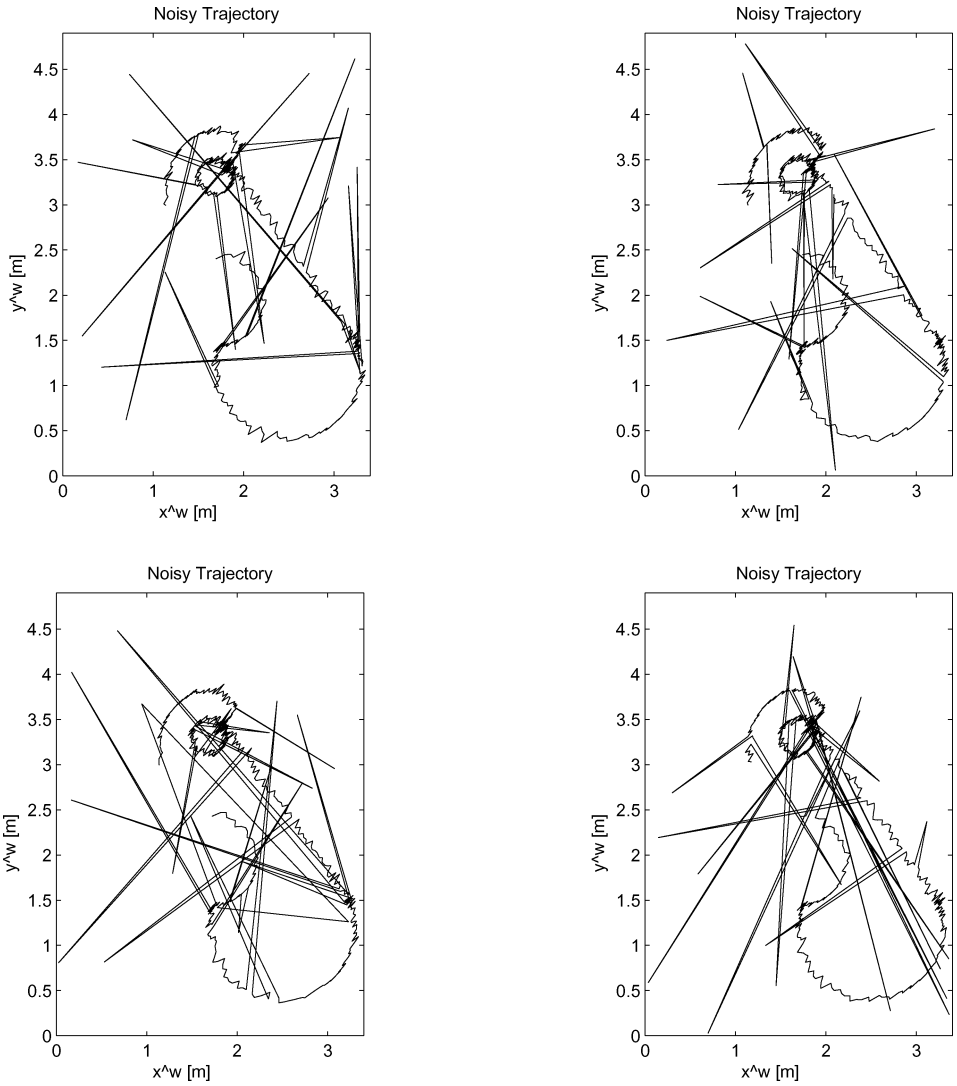


Fig. 6. Examples of the measured noisy trajectory corresponding to trajectory in the first row, second column of Fig. 5 for different noise realizations.

sudden jumps in the *measured* location of the robot. Each trajectory simulation represents 20 seconds of motion with measurement sampling frequency of 30 Hz (corresponding to the frame-grabber 30 fps acquisition rate), and thus each simulation results in 601 trajectory points.

The Root Mean Squared Error (RMSE) of the position $e_p = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(\check{x}_k^w - \hat{x}_k^w)^2 + (\check{y}_k^w - \hat{y}_k^w)^2}$ (i.e., Euclidean distance between a reference position $(\check{x}^w, \check{y}^w)$ and the estimated position $(\hat{x}^w, \hat{y}^w)$) and the RMSE of the heading angle $e_\theta = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(\check{\theta}_k - \hat{\theta}_k)^2}$ (where $\check{\theta}_k$ is the reference heading) are calculated for each of the simulation runs, and then averaged to assess the quality of the estimation resulting from the different filters. With the exception of the Particle Filter, which makes no assumption about Gaussianity of the measurements, all filters were evaluated with and without a cascaded Improbability Filter in order to verify possible improvements achievable with this scheme for rejecting non-Gaussian disturbances. The performance of each filter is also measured in terms of its computation time, which is expressed in relative terms with respect to the fastest filter. Expressing computation time in relative terms is more useful than presenting absolute time amounts because absolute computation times may further decrease in the future with the availability of new computing hardware. However, the relative computational complexity and hence, the relative computation time, should be independent of the hardware's computational power at equal levels of parallelization.

Testing of the best performing algorithm identified through the simulations was carried out using our F-180 robots as final validation step. The best filter was implemented using Visual C++ and the Bayesian Filtering Library.[38] Different predefined elliptical and rectilinear trajectories were generated by the planning and control system for the practical experiment. Non-Gaussian noise was deliberately generated by placing in the court a decoy robot with similar colour patches moving along a trajectory that intersects the path of robot being tracked. Even if the likelihood of an erroneous identification is small provided an adequate illumination and image acquisition rate, in actual competitions erroneous identification does occur in part due to particular positions that the fastly moving robots may assume with respect to the field lines, the ball or other robots.

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

Table III. Results for the 1-frame ahead prediction.

	RMSE w/IF		RMSE wo/IF		$T_c/T_{c_{min}}$	$T_c/T_{c_{min}}$
Filter	Position m	Heading [rad]	Position [m]	Heading [rad]	w/IF	wo/IF
EKF	0.0048[b]	0.2326	0.0229[b]	0.2674	1.000	1.000
DD1	0.0434	0.1386	0.0491[c]	0.1951	3.566	3.729
DD2	0.0342	0.1363	0.0491	0.1947	3.566	3.729
UKF	0.0379	0.0398[c]	0.1101	0.1399[c]	3.758	3.971
CDKF	0.0105[c]	0.0518	0.1100	0.1398[b]	3.727	3.960
SR-UKF	0.0397	0.0364[a]	0.1099	0.1399[c]	3.872	3.956
SR-CDKF	0.1656	0.0941	0.1099	0.1402	3.734	3.956
PF	—	—	0.0014[a]	0.0327[a]	—	4.901
GMSPPF	0.0034[a]	0.0392[b]	0.0637	1.0703	31.973	33.855
SPPF	0.0690	0.1042	0.1103	0.1404	42.687	45.996

a = best, b = second best, c = third best result.

Table IV. Results for the 4-frame ahead prediction.

	RMSE w/IF		RMSE wo/IF		$T_c/T_{c_{min}}$	$T_c/T_{c_{min}}$
Filter	Position [m]	Heading [rad]	Position [m]	Heading [rad]	w/IF	wo/IF
EKF	0.0234[b]	0.2078	0.0478[b]	0.2510	1.000	1.000
DD1	0.0548	0.1504	0.1232	0.1351	2.499	2.566
DD2	0.0658	0.1535	0.1232	0.1360[b]	1.749	2.751
UKF	0.1187	0.0541	0.0617[c]	0.2108	3.790	3.965
CDKF	0.0561	0.0325[a]	0.0617	0.2115	3.594	3.756
SR-UKF	0.0515[c]	0.0363[c]	0.1232	0.1356	3.582	3.736
SR-CDKF	0.0618	0.0333[b]	0.1231	0.1361[c]	3.582	3.736
PF	—	—	0.0191[a]	0.1189[a]	—	20.424
GMSPPF	0.0927	0.1353	0.1157	0.2110	112.630	116.457
SPPF	0.0192[a]	0.1483	0.1160	0.2137	145.480	157.970

a = best, b = second best, c = third best result.

Therefore, in order to test the filter under an extreme situation, it is desirable to force the occurrence of non-Gaussian noise using a decoy robot to *distract* the tracker. The next section presents and discusses the simulation and practical results obtained.

5. Experimental Results

The average position and heading RMSE obtained with the different filters for 1, 4 and 8 frame prediction horizons are presented in Tables III, IV and V, respectively. The 1, 4 and 8 frame periods correspond to 0.033, 0.133, 0.267 seconds, respectively, since the sampling period $T = 1/30$ seconds is defined by the standard frame rate of the video camera. Tables III, IV and V include RMSE values for the different filters tested with and without the Improbability Filter (IF). As earlier mentioned, the IF is not employed with the Particle Filter (PF), because the PF does not require the Gaussian noise assumption and hence it should be able to reject non-Gaussian disturbances without necessitating the pre-filtering

Table V. Results for the 8-frame ahead prediction.

	RMSE w/IF		RMSE wo/IF		$T_c/T_{c_{min}}$	$T_c/T_{c_{min}}$
Filter	Position [m]	Heading [rad]	Position [m]	Heading [rad]	w/IF	wo/IF
EKF	0.0611[c]	0.2138	0.0828[b]	0.2577	1.000	1.000
DD1	0.0884	0.1541	0.0928	0.2145	2.220	2.261
DD2	0.0582[b]	0.1373	0.0927[c]	0.2143	2.210	2.261
UKF	0.0812	0.0784	0.1549	0.1402[c]	3.874	4.024
CDKF	0.1208	0.0400[a]	0.1549	0.1395[b]	3.561	3.696
SR-UKF	0.1151	0.0438[b]	0.1548	0.1389[a]	3.569	3.670
SR-CDKF	0.1468	0.0704[c]	0.1549	0.1404	3.569	3.670
PF	—	—	0.0416[a]	0.3107	—	28.757
GMSPPF	0.2269	0.4195	0.1316	0.4340	159.053	162.990
SPPF	0.0308[a]	0.3835	0.1318	0.4364	204.304	221.425

a = best, b = second best, c = third best result.

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

stage in the form of the IF. The PF was implemented with 600 particles.

The relative computation time T_c/T_{cmin} in Tables III, IV and V is expressed relative to the fastest method, i.e. the computation time T_c required by each method is divided by the computation time of the fastest method T_{cmin}. The latter is a reasonable quantification of the relative computational complexity of the different filters implemented. For example, Table IV shows that the 4-frame ahead trajectory predictions using the PF took on average approximately 20.4 times longer than the same 4-frame ahead predictions using the EKF with the IF. As reference, the EKF 4-frame ahead prediction of the total 601 points of the robot's trajectory took on average 789.84 ms running in Matlab 7.0 under Windows XP on a computer with an AMD Turion 64 bits processor and 1 GB RAM. This means that each 4-frames ahead prediction using the EKF takes approximately 1.31 ms ($= 789.84/601\,[ms/cycle]$). Since this value is smaller than the 33.33 ms acquisition period, the EKF is a suitable candidate for real-time implementation.

The numerical simulations also revealed that there is an extra 8.9% computational cost in the 1-frame ahead prediction when the EKF is used together with the IF. This extra computational cost decreases to 6.3% and 5.3% for the 4 and 8-frame prediction horizons, respectively. For the most computationally demanding filter, the SPPF, the additional computational cost introduced by the IF is negligible and around 2.7% for the three prediction horizons. This is expectable because in the case of the SPPF its complexity makes its cost quite considerable with respect to the small cost of the simple IF.

Tables III and IV show that the smallest position RMSE for the 1 and 4-frame prediction is obtained with the PF. For the 8-frame ahead prediction, the results in Table V indicate that the SPPF with IF yields slightly better results than the PF; however, the SPPF with IF is 7.1 times computationally more demanding than the PF, and hence, the PF is still a better choice. Other filters that exhibit good accuracies in the position prediction were the GMSPPF and SPPF. However, these do not perform as well as the CDKF, SR-UKF or SR-CDKF for angle predictions 4 or 8-frames ahead. On the other hand, it is to be noted that the computational demand of the PF is about five times greater per prediction cycle than that of the simpler EKF with IF. The latter offers the best trade-off between accuracy and speed. This is because the EKF with IF was ranked second or third in the 1, 4 or 8-frame prediction in terms of the position RMSE, yet it was ranked first in terms of computational speed.

Figure 7 shows one of the six noise-free reference trajectories (see Fig. 5) employed in the simulations, while Fig. 8 shows a noisy measurement corresponding to the trajectory in Fig. 7. The 4-frame ahead predictions resulting from the application of the EKF without the IF and with the IF are shown in Figs. 9 and 10, respectively. It is possible to clearly see from the comparison of Figs. 9 and 10 that the prediction using the EKF with the IF to reject non-Gaussian noise is substantially better, while disastrous results are obtained with the EKF alone. In terms of the position and heading RMSE values presented in Table IV, the PF

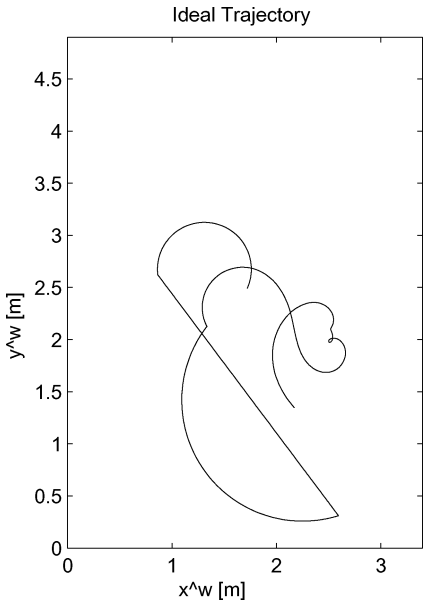


Fig. 7. Noise-free reference trajectory.

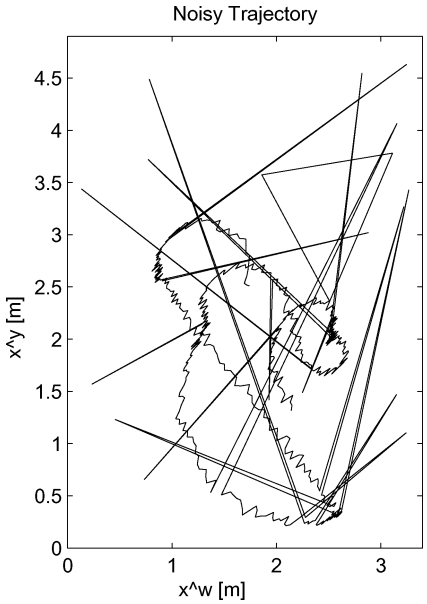


Fig. 8. Measured trajectory.

is slightly better than the EKF with IF, however the PF is not capable of generating reasonably smoothed trajectories as can be appreciated from the comparison of Figs. 9 and 11.

The EKF with the IF was chosen for validation using the robots in a real court because the EKF demonstrated a reasonable accuracy and smallest computational cost. Figure 12 shows the results of one of the tests in which the robot was programmed to follow a simple straight line parallel to the $\mathbf{y}^{\mathbf{w}}$ axis back and forth. The motion was repeated four times and lasted approximately 11 seconds. From Fig. 12 it is possible to observe that the filter is capable of overcoming the non-Gaussian noise and track the programmed reference trajectory. It is to be noted that the zigzagging of the estimation about the reference trajectory is mainly due to the fact that the robot used for testing had low

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

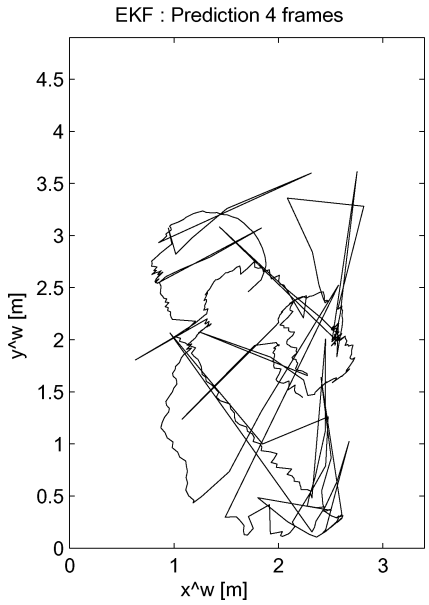


Fig. 9. Predicted trajectory 4-frames ahead using the EKF without the IF.

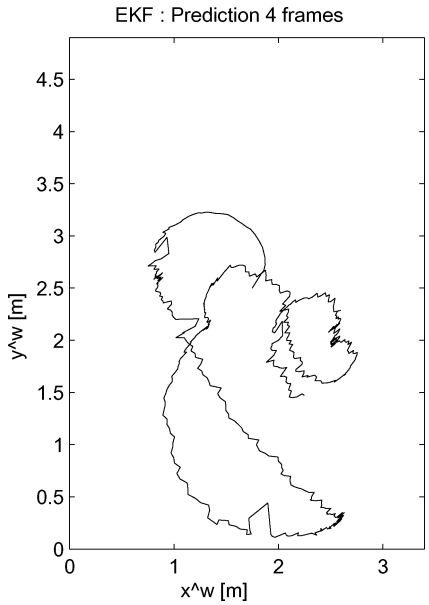


Fig. 10. Predicted trajectory 4-frames ahead using the EKF with the IF.

cost gearboxes and wheels, which generated a jerky motion difficult to control.

6. Conclusions and Future Work

The performance of 10 different Bayesian filters was evaluated in terms of their ability to handle non-Gaussian noise and to make accurate state predictions. This work's contribution is in the identification of the best filtering techniques capable of handling non-Gaussian disturbances or noise with a comparatively low computational effort in order to make accurate state predictions. The ability to make a good prediction of a system's state, rather than to simply obtain a good state estimate at the current time, is fundamental in many autonomous systems applications

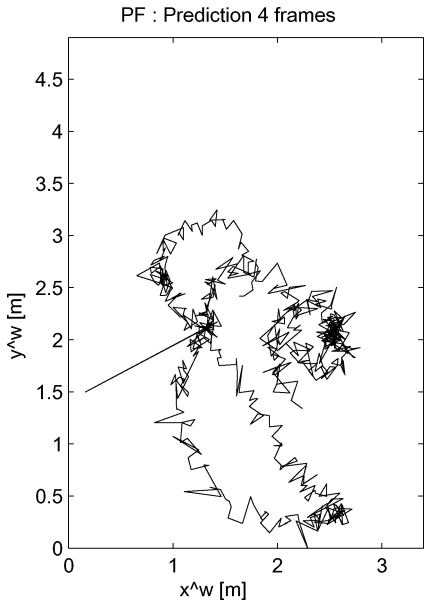


Fig. 11. Predicted trajectory 4-frames ahead using the PF.

that involve expensive computations to perform perception, control and decision tasks. Such expensive computations introduce delays that limit response times of the closed-loop system. Therefore, accurate state predictors become essential to cope with the delays.

The filtering techniques evaluated have been proposed during the last decades, however no exhaustive comparison was found in the literature. Some of the techniques belong to the family of the Kalman filter and its modifications, while other to the family of Sequential Monte Carlo sampling schemes. A third group is composed by hybrid filters, which combine the Kalman filtering approaches with Sequential Monte Carlo techniques.

The results obtained indicate that in general both the simple EKF with the IF and the PF yield accurate predictions. However, the PF requires a significantly larger computational effort. This larger computational cost could nevertheless be decreased by sacrificing accuracy if the number of particles in the PF implementation is reduced. The PF implemented in the simulations used 600 particles and had a slightly better accuracy than that of the EKF with the IF, however the PF was about 5 times slower than the EKF with IF per sampling period. This means that for prediction horizons of n sampling periods, the PF would take $5n$ times longer than the simpler EKF with IF. Some other methods evaluated also satisfy the accuracy requirements of the application presented, but their computational complexity and comparatively smaller accuracy makes them a less reasonable choice. In particular, the GMSPPF and SPPF are computationally expensive and less suitable for applications requiring online filtering. Future availability of more powerful computers may allow to reduce the computation time required by these methods, however their advantages in the context of tracking multiple robots is not clear.

It is also worth noting that the inclusion of the Improbability Filter allows for significant prediction improvements in all the different filters implemented that assume Gaussian noises (all filters except for the Particle

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

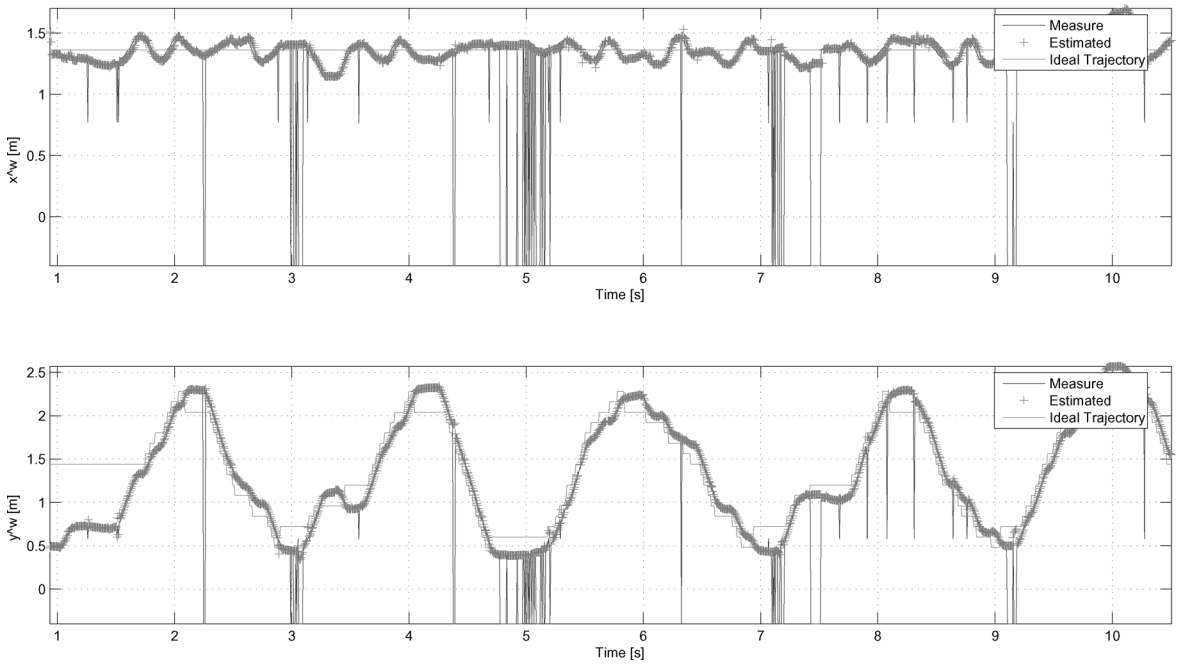


Fig. 12. Trajectory components x^w and y^w of a real F-180 robot predicted 4-frames ahead using the EKF with the IF.

Filter). A minor drawback of the Improbability Filter is that it requires the appropriate threshold value to be found empirically. On the other hand, the right tuning of the hybrid SPKF combined with SMC sampling methods may be very difficult because a slight modification of the filter parameters may result in non positive-definite state covariance matrices. This is because the sigma points are deterministically calculated in terms of the square root of the weighted state covariance matrix.[10] The approaches preserve the first two statistical moments of a not necessarily Gaussian distribution and ensure that the predicted covariance stays semi-positive definite, unlike previous methods which could yield negative definite predicted covariances. However, the choice of the filter parameters and the numerical round-off errors that accumulate throughout the simulation may render the predicted covariance matrix to be not strictly positive definite. In such a case, it may not be possible to compute the matrix square root, even if the numerically efficient Cholesky factorization[39] is employed. To overcome this problem, the state covariance matrix was reset whenever it became non-positive definite. However, this action also limits the performance of the hybrid approaches.

Future work considers the implementation of multiple hypothesis tracking methods to increase the robustness of the planning and control system that drives the team of robots, the development of techniques for automatically adjusting the IF parameters so as to maximize the rejection of false positives, and the comparison of the Bayesian methods with non-Bayesian techniques, such as approaches based on neural networks.

Acknowledgements

This project has been supported by the National Commission for Science and Technology Research of Chile (Conicyt) under Fondecyt Grant 11060251.

References

1. S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics* (MIT Press, Cambridge, MA, 2005).
2. B. Ristic, S. Arulampalam and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications* (Artech House, Norwood, MA, 2004).
3. Special Issue on Sequential State Estimation, *Proceedings of the IEEE* **92**(3), (2004).
4. M.-C. Tsai, K.-Y. Chen, M.-Y. Cheng and K. C. Lin, "Implementation of a real-time moving object tracking system using visual servoing," *Robotica* **21**(6), 615–625 (2003).
5. C. Hue, J.P. Le-Cadre and P. Perez, "Sequential Monte Carlo methods for multiple target tracking and data fusion," *IEEE Trans. Signal Processing* **50**(2), 309–325 (2002).
6. M. Orton and W. Fitzgerald, "A Bayesian approach to tracking multiple targets using sensor arrays and particle filters," *IEEE Trans. Signal Processing* **50**(2), 216–223 (2002).
7. A. Doucet, S. Godsill and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Stat. Comput.* **10**(3), 197–208 (2000).
8. Y. Bar-Shalom and X. Rong Li, *Estimation and Tracking: Principles, Techniques, and Software* (Ybs Publishing, Storrs, CT, USA, 1998).
9. F. Askary and N. T. Sullivan, "Importance of measurement accuracy in statistical process control," *Proceedings of SPIE Metrology, Inspection, and Process Control for Microlithography XIV* (N. T. Sullivan ed.) **3998**, Santa Clara, CA, USA (Feb. 28– Mar. 2, 2000) pp. 546–554.
10. R. van der Merwe, E. A. Wan and S. Julier, "Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion: Applications to Integrated Navigation", *Proceedings of the AIAA Guidance, Navigation and Control Conference* (2004) pp. 5120.
11. A. Gloye, M. Simon, A. Egorova, F. Wiesel, O. Tenchio, M. Schreiber, S. Behnke and R. Rojas, *Predicting Away Robot Control Latency* (Technical Report B-08-03, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 2003).
12. D. Bouvet and G. Garcia, "GPS latency identification by Kalman filtering," *Robotica* **18**(5), 475–485 (2001).
13. F. Donoso-Aguirre, J.-P. Bustos-Salas, M. Torres-Torriti and A. Guesalaga, "Mobile robot localization using the Hausdorff distance," *Robotica, to appear (published online by Cambridge University Press, 12 Jul 2007).*

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press

14. B. D. O. Anderson and J. B. Moore, *Optimal Filtering* (Dover Publications, Mineola, NY, 2005).
15. R. E. Kalman, "New approach to linear filtering and prediction problems," *Trans. ASME* **82**, 34–45 (1960).
16. A. H. Jazwinski, *Stochastic Processes and Filtering Theory* (Academic Press, 1970).
17. S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, Orlando, Florida (1997) pp. 182–193.
18. K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *Proceedings of the IEEE International Conference on Robotics and Automation* **45**(5), San Francisco, CA, USA, (Apr. 24–28, 2000) pp. 910–927.
19. M. Nørgaard, N. K. Poulsen and O. Ravn, "New developments in state estimation for nonlinear systems," *Automatica* **36**(11), 1627–1638 (2000).
20. R. Van-der-Merwe and E. Wan, "The Square-Root Unscented Kalman Filter for state and parameter estimation," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* **6**, Salt Lake City, UT, USA, (May 7–11, 2001) pp. 3461–3464.
21. R. Van-der-Merwe, A. Doucet, N. de Freitas and E. Wan, "The Unscented Particle Filter," **In:** *Advances in Neural Information Processing Systems* (T. K. Leen, T. G. Dietterich and V. Tresp eds.), (MIT Press, Cambridge, MA, 2000) pp. 584–590 .
22. R. Van-der-Merwe and E. Wan, "Gaussian mixture Sigma-Point Particle Filters for sequential probabilistic inference in dynamic state-space models," *Proceedings of the International Conference on Acoustic, Speech and Signal Processing*, Hong Kong, SAR China, (Apr. 6–10, 2003) pp. 701–704.
23. J. H. Kotecha and P. M. Djuric, "Gaussian sum Particle Filtering for dynamic state space models," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing 2001*, Salt lake City, UT, USA, (May 7–11, 2001). pp. 3465–3468.
24. N. Cui, L. Hong and J. R. Layne, "A comparison of nonlinear filtering approaches with an application to ground target tracking," *Signal Process., Elsevier*, **85**(8), 1469–1492 (2005).
25. R. Wright, S. R. Maskell and M. Briers, "Comparison of Kalman-based methods with Particle Filters for raid tracking," *Pract. Bayesian Stat. 5* (2003).
26. D. C. K. Yuen and B. A. MacDonald, "A comparison between Extended Kalman Filtering and Sequential Monte Carlo techniques for simultaneous localisation and map-building," *Proceedings of the 2002 Australasian Conference on Robotics and Automation*, Auckland, Australia, (Nov. 27–29 2002) pp. 111–116.
27. E. V. Cuevas, D. Zaldívar and R. Rojas, *Kalman Filter for Vision Tracking* (Technical Report B-05-12, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 2005).
28. E. V. Cuevas, D. Zaldívar and R. Rojas, *Particle Filter for Vision Tracking* (Technical Report B-05-13, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 2005).
29. N. Metropolis and S. Ulam, "The Monte Carlo method," *J. Am. Stat. Assoc.* **44**(247), 335–341 (1949).
30. N. J. Gordon, D. J. Salmond and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proc. F* **140**(2), 107–113 (1993).
31. M. J. Goris, D. A. Gray and I. M. Y. Mareels, "Reducing the computational load of a Kalman filter," *IEE Electron. Lett.* **33** (18), 1539–1541 (1997).
32. R. Karlsson, T. Shön and F. Gustafsson, "Complexity analysis of the marginalized particle filter," *IEEE Trans. Signal Process.* **53**(11), 4408–4411 (2005).
33. B. Browning, M. Bowling and M. Veloso, "Improbability filtering for rejecting false positives," *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, USA, (May 11–15, 2002) pp. 3038–3043.
34. Y. Liu, X. Wu, J. Zhu and J. Lew, "Omni-directional mobile robot controller design by trajectory linearization," *Proceedings of the 2003 American Control Conference* **4**, Denver, Co, USA, (June 4–6 2002) pp. 3423–3428.
35. J. Angeles, *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*, 3rd ed. (Springer, 2006).
36. R. van der Merwe, *ReBEL: Recursive Estimation Bayesian Library* (OGI School of Science & Engineering, Oregon Health & Science University (OHSU), 2002–2006), http://choosh.cse.ogi.edu/rebel/, visited on February 2006.
37. M. Nørgaard, *Kalmtool: version 2*, http://www.iau.dtu.dk/research/control/kalmtool.html Ole Ravn and Niels Kjølstad Poulsen, *KalmTool: versions 3, 4*, http://server.oersted.dtu.dk/www/or/kalmtool/, visited February 2006, (Department of Mathematical Modelling, Department of Automation, Ørsted - Danmarks Tekniske Universitet, 1998–2006).
38. K. Gadeyne, *BFL: Bayesian Filtering Library* (Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium, 2001–2006), http://www.orocos.org/bfl, visited on February 2006.
39. W. H. Press, S. A. Teukolsky, W. T. Vetterling and E. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. (Cambridge University Press, MA, 1992).

https://doi.org/10.1017/S0263574708004153 Published online by Cambridge University Press