Multitarget Simultaneous Localisation and Mapping of a Sensor Network

Ángel F. García-Fernández, Mark R. Morelande, Jesús Grajal

Abstract

This paper addresses the problem of simultaneously localising multiple targets and estimating the positions of the sensors in a sensor network using particle filters. We develop a new technique called Multitarget Simultaneous Localisation and Mapping (MSLAM) that has better performance than the well-known FastSLAM when there are several targets in the surveillance area. The proposed algorithm is based on the Parallel Partition particle filter, especially designed for multiple target tracking, and the Truncated Unscented Kalman Filter for updating the sensors' positions.

Index Terms

Multitarget SLAM, Particle filters, Sensor Networks, Tracking, Truncated Unscented Kalman filter.

I. Introduction

There has been a significant interest in wireless sensor networks over the past few years due to a broad range of applications related to homeland security, environmental monitoring and biological species tracking among other fields of use [1]. These networks are composed of inexpensive sensors that collect data and usually report it to a fusion center through a wireless channel [2], [3].

When the network is deployed in the surveillance area, the positions of the sensors might not be accurately known. For instance, in battlefield surveillance, the sensor network might have to be deployed

Ángel F. García-Fernández and Jesús Grajal are with Departamento de Señales, Sistemas y Radiocomunicaciones, ETSI Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain. email: {agarcia, jesus}@gmr.ssr.upm.es

Mark R. Morelande is with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, Victoria 3010, Australia. email: mrmore@unimelb.edu.au

quickly and, therefore, there could be large uncertainty in the positions of the sensors unless they have a built-in GPS. In most cases, this option is not feasible because of its power consumption and the increase in the price of the sensors.

The main objective of this paper is to develop a centralised algorithm for tracking multiple targets in such a scenario. However, this aim can be achieved more accurately by simultaneously estimating the sensors' positions. Thus, we address the problem of simultaneously estimating the positions of multiple targets and localising the sensors based on the sensors' measurements from a Bayesian point of view. This problem has previously been addressed for a single target in [4], [5] and is also related to the Simultaneous Localisation and Mapping (SLAM) problem in robotics in which the robot estimates a map of the environment, i.e., the positions of the so-called landmarks, and its position based on its own measurements [6]–[8]. In the following discussion, we will use the acronyms R-SLAM for SLAM in robotics and SN-SLAM for SLAM in sensor networks, while in the rest of the paper, we will use SLAM instead of SN-SLAM as context permits.

There have been two main approaches to tackle the R-SLAM problem: Kalman-filter-type algorithms (KSLAM) [7], [9] and particle-filter-type algorithms (FastSLAM or FSLAM) [10], [11]. KSLAM estimates the state of the robot at the current time and the positions of the landmarks based on previous measurements using a Kalman filter (KF). However, this algorithm has two main problems. The first one is that it is based on Kalman filtering so its result is optimal for linear/Gaussian models but good performance and convergence are not guaranteed for nonlinear/non-Gaussian models. Moreover, the posterior correlation among landmarks' positions should be taken into account to build a more accurate map [12] as the current robot state and landmarks' positions conditioned on the past measurements are correlated. The computational expense of doing this is quadratic in the number of landmarks. On the contrary, FSLAM aims to estimate the whole robot trajectory, rather than the current robot state, using a particle filter [13]. It can be shown that, conditioned on the whole robot trajectory, the landmarks' positions are independent. Then, FSLAM uses approximate Rao-Blackwellisation [14] so that the posterior pdf of the map conditioned on a given particle is approximated by a set of independent Gaussians. This results in linear complexity in the number of landmarks, rather than the quadratic complexity that would be necessary if the landmarks' positions were correlated [7].

Information filtering [8] and smoothing [15] have also been applied to R-SLAM. These methods allow sparsification of the information matrix to be exploited in reducing computation. Information filtering estimates the state of the robot at the current time, as does KSLAM, while smoothing estimates the whole robot trajectory to exploit factorisation as in FSLAM with the difference that [15] uses batch processing.

However, these methods are based on the linearisation of the process and measurement equations and, then, their performance is expected to be poor in highly nonlinear scenarios. Another approach to dealing with the R-SLAM problem is via finite set statistics in which the map is represented by a random set [16]–[18]. Within this framework, an approximation to the posterior pdf of the map conditioned on the robot trajectory based on the Gaussian mixture Probability Hypothesis Density (PHD) filter [19] outperforms FSLAM using real data in environments with high clutter [16]–[18]. The use of this formulation to represent the map is also adopted in [20] although assuming the robot pose is known.

SN-SLAM is equivalent to R-SLAM considering that now the entities that sense (sensors/robots) are static and the ones which are measured (targets/landmarks) are dynamic. If we solved this problem via KSLAM, the KF recursive equations would be similar but using the new dynamic models. Conversely, the direct application of FSLAM to SN-SLAM, i.e., approximating the sensors' positions by particles and the targets' states by Gaussians, is not appropriate as particle filters are not easily applied to the estimation of static parameters [21]. In addition, the number of sensors in a sensor network is usually high. Then, the positions of the sensors is a vector of large dimension and the particle filter will vastly suffer from the curse of dimensionality [22]. Therefore, it is better to approximate the whole trajectories of the targets by particles, as they are the dynamic part of the model, and conditioned on them, the sensors' positions approximated by a set of independent Gaussians. Thus, in principle, we can easily adapt the algorithms for R-SLAM to SN-SLAM. We should also note that the formulations that use information smoothing [15] or finite sets statistics [16]–[18] can also deal with moving landmarks although their generalisations have not been done yet.

There is an important difference between both problems using particle-filter-type algorithms when there are multiple robots/targets. In both SN-SLAM and R-SLAM, it is assumed that the sensing entities measure the sensed entities independently of other sensing entities. Then, in R-SLAM, a robot's measurement only depends on its (dynamic) state and the landmarks, where the posterior of the state is represented by single-entity particles. Then, a particle filter in R-SLAM could approximate the dynamic states using single-entity particles even in a multiple robot set-up. This is because a robot can build a map without considering other robots although merging individual maps would improve map estimation [23]. In SN-SLAM, a sensor's measurement depends only on its position and the states of all the targets. As a result, targets cannot be considered separately and the posterior of the (dynamic) state must be represented by multiple-entity particles. To summarise, the primary difference between R-SLAM and SN-SLAM is that single-entity particles cannot be used in SN-SLAM as we have to account for all the targets to build the map.

The FSLAM method has two shortcomings: one applies to both R-SLAM and SN-SLAM while the other applies only to its generalisation to multitarget SN-SLAM. The first problem is that accurately estimating the whole robot/target trajectory using a particle filter is impractical as we are trying to estimate a quantity of increasing dimension. Therefore, it does not produce consistent estimates in the long-term because of the degeneracy of the particles [24]. Nevertheless, FSLAM has been shown to work well in most real and simulated scenarios without having to propagate a covariance matrix of large dimensions [11]. The second problem arises because FSLAM was originally designed to deal with single-entity particles. As such, its application to multitarget SN-SLAM does not readily permit the use of techniques which have been found to be essential for computationally efficient approximation of the posterior in multitarget situations. One such technique is subparticle crossover [25]–[29]. It takes advantage of the structure of multiple-entity particles and it is not compatible with FSLAM as explained in Section III. Subparticle crossover allows the propagation of the multitarget particles, which are made of subparticles, each one representing a target state, by mixing subparticles of different particles.

In this paper, we develop a new SLAM algorithm for sensor networks based on particle filtering when there are multiple targets. As mentioned in the previous paragraph, a well-known important element of efficient multitarget particle filtering is subparticle crossover [25]–[29]. Therefore, we develop an algorithm referred to as Multitarget SLAM (MSLAM) which permits its application. The key feature of this algorithm is that the distribution of the sensor positions is computed conditional on only the most recent multitarget state rather than the trajectories of multitarget states, as in FSLAM. The result of this is a map mixing step in which the map from each of the particles are combined. Computationally efficient implementation of this idea requires certain approximations, to be described in Section III.

A further contribution of this paper arises in the approximation of the map posterior. Traditionally, KF approximations such as the Extended Kalman Filter (EKF) [7], [10], [11] or the Unscented Kalman Filter (UKF) [9] have been used. It is known that measurement nonlinearities become significant when the measurement noise variance is small compared to the prior variance [30]. In this case, a KF does not provide a reliable, accurate approximation of the posterior. However, applying a KF to a modified prior, which is a mixture of the original prior and a truncated version of it, could provide a vast improvement in the approximation to the posterior [31]. This algorithm is called Truncated Kalman filter (TKF) and its aim is to reduce the variance of the prior so that measurement nonlinearities become milder and the KF applied to the modified prior approximates the posterior properly. In practice, the TKF can be approximately realised using the Truncated Unscented Kalman Filter (TUKF) [31], [32]. It will be shown that MSLAM combined with TUKF provides a vast improvement in performance compared to conventional techniques in multitarget situations.

This paper is organised as follows. The target dynamic and measurement model are explained in Section II. In Section III, the differences between FSLAM and MSLAM are outlined. How to sample the positions of the targets, update the targets' velocities and calculate the particles' weights is addressed in Section IV. The Bayes' rule approximation to update the sensors' positions is shown in Section V. Simulation results are provided in Section VI. Finally, conclusions are drawn in Section VI.

II. Problem statement

We assume there are a fixed and known number of targets, t, in the surveillance area. The state of target i is described at time k by the four-dimensional vector $\mathbf{x}_i^k = \left[x_i^k,\, \dot{x}_i^k,\, y_i^k,\, \dot{y}_i^k\right]^T$ where $\left[x_i^k,\, y_i^k\right]^T$ is the position vector and $\left[\dot{x}_i^k,\, \dot{y}_i^k\right]^T$ is the velocity vector and the superscript T means transpose. The multitarget state vector is formed by stacking the individual target state vectors into one vector $\mathbf{X}^k = \left[\left(\mathbf{x}_1^k\right)^T,\, \left(\mathbf{x}_2^k\right)^T, ..., \left(\mathbf{x}_t^k\right)^T\right]^T$. We also assume there are M stationary sensors. The position of sensor i is described by the two-dimensional vector $\mathbf{m}_i = \left[m_{x,i},\, m_{y,i}\right]^T$. The multisensor state vector is formed by stacking the individual sensor positions into one vector $\mathbf{m} = \left[\mathbf{m}_1^T,\, \mathbf{m}_2^T, \ldots, \mathbf{m}_M^T\right]^T$. Therefore, our aim is the recursive approximation of $p\left(\mathbf{X}^k,\, \mathbf{m}\left|\mathbf{z}^{1:k}\right.\right)$ where $\mathbf{z}^{1:k} = \left(\mathbf{z}^1,\, \mathbf{z}^2, ..., \mathbf{z}^k\right)$ refers to the collection of measurements up to time k, and $\mathbf{z}^i$ is the set of measurements at time i. To do so, firstly we develop the dynamic model of targets and the measurement model of the sensors.

The dynamic model of the targets is the nearly-constant velocity model [33]:

$$p\left(\mathbf{X}^{k+1}\left|\mathbf{X}^k\right.\right) = \prod_{j=1}^{t} \mathcal{N}\left(\mathbf{x}_j^{k+1};\, \mathbf{F}\cdot\mathbf{x}_j^k,\, \mathbf{Q}\right) \tag{1}$$

$$\mathbf{F} = \mathbf{I}_2 \otimes \begin{pmatrix} 1 & \tau \\ 0 & 1 \end{pmatrix} \tag{2}$$

$$\mathbf{Q} = \sigma_u^2 \mathbf{I}_2 \otimes \begin{pmatrix} \tau^3/3 & \tau^2/2 \\ \tau^2/2 & \tau \end{pmatrix} \tag{3}$$

where it is assumed that the movements of the targets are independent, $\mathcal{N}\left(\mathbf{x};\, \overline{\mathbf{x}},\, \mathbf{Q}\right)$ is the Gaussian pdf evaluated at $\mathbf{x}$ with mean $\overline{\mathbf{x}}$ and covariance matrix $\mathbf{Q}$, $\mathbf{I}_m$ is the $m \times m$ identity matrix, $\otimes$ is the Kronecker product and σ_u^2 is the continuous-time process noise intensity [33].

The vector $\mathbf{z}^k = \left[\left(\mathbf{z}_1^k\right)^T,\, \left(\mathbf{z}_2^k\right)^T, \ldots, \left(\mathbf{z}_M^k\right)^T\right]^T$ contains the measurements from the sensors at time k and $\mathbf{z}_j^k$ is the measurement from sensor j at time k. Measurements are independent in each sensor and do not depend on targets' velocities:

$$p\left(\mathbf{z}^k \left|\mathbf{X}^k,\, \mathbf{m}\right.\right) = \prod_{r=1}^{M} p\left(\mathbf{z}_r^k \left|\mathbf{P}^k,\, \mathbf{m}_r\right.\right) \tag{4}$$

where $\mathbf{P}^k$ is the vector with all the targets' positions.

We also assume that the sensors are equipped with N_a antennas aiming at different known directions $[\theta_1,\, \theta_2,\, \ldots,\, \theta_{N_a}]^T$ which measure the amplitude of the signal coming from those directions independently [34]. Therefore, the measurement in sensor r at time k is represented by a vector $\mathbf{z}_r^k = \left[z_{r,1}^k,\, z_{r,2}^k,\, \ldots,\, z_{r,N_a}^k\right]^T$ where each component represents the measurement in an antenna. We assume Gaussian noise so that

$$p\left(\mathbf{z}_r^k \left|\mathbf{P}^k,\, \mathbf{m}_r\right.\right) = \prod_{l=1}^{N_a} \mathcal{N}\left(z_{r,l}^k;\, \overline{z}_{r,l}^k, \sigma_z^2\right) \tag{5}$$

where $\sigma_z^2 = 1$ without loss of generality and the expected value of the amplitude is:

$$\overline{z}_{r,l}^k = \sqrt{\sum_{j=1}^{t} SNR\left(d_{j,r,k}\right) \cdot \exp\left(-\beta\left(g\left(\vartheta_{j,r,k},\, \theta_l\right)\right)^2\right)} \tag{6}$$

where

$$g\left(\vartheta,\, \theta\right) = \operatorname{mod}\left(\vartheta - \theta + \pi, 2\pi\right) - \pi \tag{7}$$

$$SNR\left(d\right) = \begin{cases} SNR_0 & d \le d_0 \\ \frac{SNR_0 d_0^2}{d^2} & d > d_0 \end{cases} \tag{8}$$

$$d_{j,r,k} = \sqrt{\left(x_j^k - m_{x,r}\right)^2 + \left(y_j^k - m_{y,r}\right)^2} \tag{9}$$

$$\vartheta_{j,r,k} = \arctan\frac{y_j^k - m_{y,r}}{x_j^k - m_{x,r}} + \begin{cases} 0 & x_j^k - m_{x,r} \ge 0 \\ \pi & y_j^k - m_{y,r} \ge 0,\, x_j^k - m_{x,r} < 0 \\ -\pi & y_j^k - m_{y,r} < 0,\, x_j^k - m_{x,r} < 0 \end{cases} \tag{10}$$

and $g\left(\vartheta,\, \theta\right)$ is a modular subtraction of angles ϑ and θ that accounts for angle wrapping [35], β is a parameter that controls the radiation pattern of the antenna, SNR_0 is the maximum signal-to-noise ratio produced by a single target in a sensor, and d_0 is the saturation distance. We should note that $d_{j,r,k}$ is the distance from target j to sensor r at time k, $\vartheta_{j,r,k}$ is the angle formed by the target j in relation to sensor r and the positive side of the x axis at time k, $SNR\left(d_{j,r,k}\right)$ is the maximum signal-to-noise ratio produced by target j in sensor r at time k and the exponential term in (6) indicates the radiation pattern of the antenna.

We also assume that we have some prior knowledge about the targets and sensors. That is, we assume that the prior at time 0 for target j is $p\left(\mathbf{x}_j^0\right)$ and for sensor r is:

$$p\left(\mathbf{m}_r\right) = \mathcal{N}\left(\mathbf{m}_r;\, \overline{\mathbf{m}}_r^0,\, \mathbf{\Theta}_r^0\right) \tag{11}$$

where $\overline{\mathbf{m}}_r^0$ and $\mathbf{\Theta}_r^0$ are the prior mean and covariance matrix of sensor r.

III. FSLAM VERSUS MULTITARGET SLAM

In this section, we explain the characteristics of FSLAM, the reasons why it is not appropriate to deal with a multitarget scenario and we introduce MSLAM to address its deficiencies.

A. *FSLAM*

FSLAM is based on the approximation of the pdf

$$p\left(\mathbf{X}^{0:k},\, \mathbf{m}\,\middle|\,\mathbf{z}^{1:k}\right) = p\left(\mathbf{P}^{0:k}\,\middle|\,\mathbf{z}^{1:k}\right) p\left(\mathbf{V}^{0:k},\, \mathbf{m}\,\middle|\,\mathbf{P}^{0:k},\, \mathbf{z}^{1:k}\right) \tag{12}$$

where $\mathbf{X}^{0:k}$ refers to the collection of the multitarget states from time 0 to k, $\mathbf{V}^{0:k}$ represents the targets' velocities from time 0 to k and $\mathbf{P}^{0:k}$ represents the targets' positions from time 0 to k. The posterior pdf of the targets' positions is approximated using a particle filter [36]. We should note that, according to our model, the relationship between the targets' velocities and positions is linear and Gaussian and the measurements do not depend on the velocity. Then, we can apply Rao-Blackwellisation removing the velocities from the particle filter and calculating their pdf conditioned on the positions by a KF [14], [27]. In addition, FSLAM approximates the pdf of the positions of the sensors conditioned on the complete trajectories of the targets and the measurements by set of independent Gaussians, with linear complexity [10], [11]. Thus, the pdf of the multitarget positions from time 0 to k is represented by particles and the conditional pdfs of the positions of the sensors and the velocities are represented using Rao-Blackwellisation via KFs applied to each particle. However, sampling the complete multitarget trajectory in a proper way is impractical because of its increasing dimension and sampling degeneracy. This problem is more pronounced in a multitarget set-up, as it is even more difficult to sample the complete trajectory because the dimension of the state vector is larger and, consequently, particle degeneracy is more acute.

A representation of the propagation of the particles using FSLAM for multiple targets is shown in Fig. 1 where the last three time instants have been depicted for two particles. As we are sampling the complete trajectory, to form particle i at a time we must propagate simultaneously all the targets of the same particle at the previous time. This implies that we cannot mix different subparticles (part of the

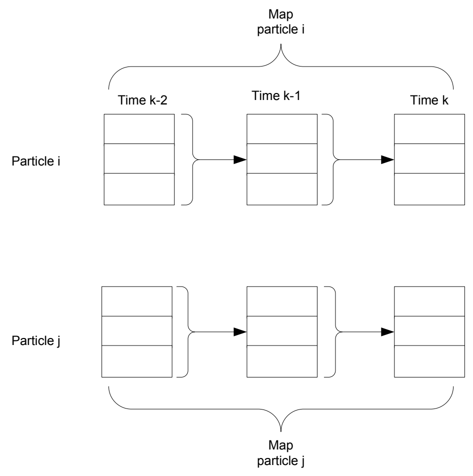


Figure 1 – FSLAM adapted to multiple targets. Each particle is represented by three boxes. Each box represents the part of the particle (subparticle) corresponding to a target. The whole multitarget particle has to be propagated at once as indicated by the arrows. Also, each particle has its own map independent of the maps of the rest of the particles.

particle that represent one target) to form the new particles, as we will do in the following subsection. This hinders the process of sampling efficiently because the more targets there are, the more difficult it is to sample adequately as the dimension of the state increases [22]. In general, any sampling procedure that propagates all the targets within a particle at once will be compatible with FSLAM and we will be able to approximate (12). For example, particles are propagated using the prior in [10], while in [11], it is done using an EKF approximation to the optimal importance density. In Fig. 1, it is also illustrated that each particle has its own map conditioned on the whole past trajectory of that particle that is independent of the maps of the rest of the particles.

Another problem with FSLAM is that it suffers map degeneracy. This arises because the map posterior is computed conditional on the trajectory of the target states. Resampling ensures that all particles at the current time have a common ancestor at some previous time. Sensors that have not been approached by any target since this time will have the same posterior means and covariance matrices for all particles, see Fig. 2. Some techniques proposed to alleviate this problem deal with the resampling step [37], [38] or use parameter estimation rather than Rao-Blackwellisation for the map [39]. Instead, we have developed the algorithm MSLAM, which is explained in the next subsection, to lower the effect of these two shortcomings.

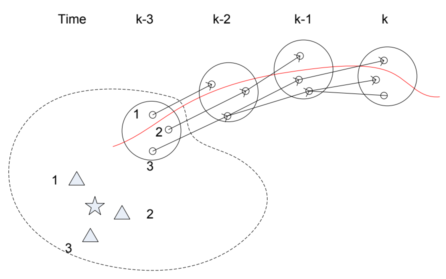


Figure 2 – Map degeneracy in FSLAM: The red line indicates the target trajectory. The target state is represented by three particles at each time step. The arrows indicate who is the parent of a particle at the previous time step. The star denotes a real sensor position and the triangles denote its mean position for a particle. The dotted line indicates the coverage of the sensor. Therefore, the sensor position is only updated at time $k-3$ remaining unaltered thereafter. At time $k-3$, each particle has a different mean position for the sensor. However, at time k all the particles have the same mean position for the sensor that is one of the initial means as the three particles have a common ancestor and, thus, map degeneracy appears.

B. *Multitarget SLAM*

The algorithm we develop here, MSLAM, was designed to propagate the particles of the targets more efficiently than FSLAM and to mitigate the map degeneracy problem. To this end, we use the Parallel Partition method (PP) to propagate the particles [28]. This method was designed to properly deal with multitarget tracking with low computational burden. The use of this method implies that we need to resort to a map mixing step that reduces the effect of map degeneracy as we will explain hereafter.

The PP method uses a technique called subparticle crossover. This refers to the mixing of parts of different particles to form new ones in the propagation step. This way the effect of particle degeneracy is lowered and the filter can work with fewer particles in a multitarget tracking set-up [25]–[29]. However, carrying out subparticle crossover implies that we are not sampling the complete trajectories of all the targets as we get the best subparticles of each target to represent the new state. This implies that FSLAM, approximating (12), cannot be used as we are not sampling the complete multitarget trajectory. The way the PP method propagates the particles is shown in Fig. 3. As we mix different parts of the particles at time $k-1$ to form the new particles at time k, it is obvious that we also need to mix the maps of the particles at time $k-1$ when we get the particles at time k. This map mixing step is one of the novelties of our algorithm compared to FSLAM.

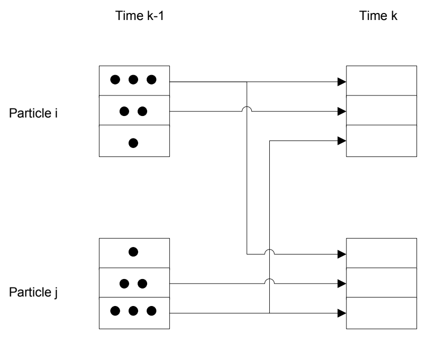


Figure 3 – Subparticle crossover for three targets: A particle at time k can be made up of the propagation of the tracks that belonged to different particles at time $k-1$. The circles inside each subparticle stand for the first-stage weight of the subparticle $b_{j,i}$ in (24). At time k, we tend to form particles with subparticles that have a high weight.

As opposed to FSLAM, MSLAM aims to compute

$$p\left(\mathbf{X}^{k},\,\mathbf{m}\left|\mathbf{z}^{1:k}\right.\right)=p\left(\mathbf{P}^{k}\left|\mathbf{z}^{1:k}\right.\right)p\left(\mathbf{V}^{k},\,\mathbf{m}\left|\mathbf{P}^{k},\,\mathbf{z}^{1:k}\right.\right) \tag{13}$$

It can be seen that the main difference between MSLAM and FSLAM is that in (13) we approximate the pdf of the current multitarget state, while in (12), we approximate the pdf of the complete multitarget trajectories. As in FSLAM, we use a particle filter approximation for the target positions such that (13) becomes:

$$p\left(\mathbf{X}^{k},\,\mathbf{m}\left|\mathbf{z}^{1:k}\right.\right)\approx\sum_{i=1}^{N_{par}}w_{i}^{k}\delta\left(\mathbf{P}^{k}-\mathbf{P}_{i}^{k}\right)p\left(\mathbf{V}^{k},\,\mathbf{m}\left|\mathbf{P}_{i}^{k},\,\mathbf{z}^{1:k}\right.\right) \tag{14}$$

where N_{par} is the number of particles, $\delta(\cdot)$ is the Dirac delta, $\mathbf{P}_{i}^{k}$ is particle i that represents the positions of the targets at time k and w_{i}^{k} is its weight. We assume that the pdf $p\left(\mathbf{V}^{k},\,\mathbf{m}\left|\mathbf{P}_{i}^{k},\,\mathbf{z}^{1:k}\right.\right)$ is:

$$p\left(\mathbf{V}^{k},\,\mathbf{m}\left|\mathbf{P}_{i}^{k},\,\mathbf{z}^{1:k}\right.\right)=\prod_{j=1}^{t}\mathcal{N}\left(\mathbf{v}_{j}^{k};\overline{\mathbf{v}}_{j,i}^{k},\boldsymbol{\Sigma}_{j,i}^{k}\right)\cdot\prod_{r=1}^{M}\mathcal{N}\left(\mathbf{m}_{r};\overline{\mathbf{m}}_{r,i}^{k},\boldsymbol{\Theta}_{r,i}^{k}\right) \tag{15}$$

where $\mathbf{v}_{j}^{k}$ is the velocity of target j at time k, $\overline{\mathbf{v}}_{j,i}^{k}$ and $\boldsymbol{\Sigma}_{j,i}^{k}$ are the mean and covariance matrix approximation of the velocity of target j for particle i and, $\overline{\mathbf{m}}_{r,i}^{k}$ and $\boldsymbol{\Theta}_{r,i}^{k}$ are the mean and covariance matrix approximation of sensor r position for particle i.

The imposition of posterior independence and Gaussianity on the velocities and sensors' positions in (15) involve some approximations which will be described in the following sections. Nevertheless, we are going to demonstrate in the simulations of Section VI that for a multitarget scenario making these approximations has much better results than trying to estimate the complete trajectories of the targets with FSLAM.

There are two main reasons for this. The first one is that MSLAM can be applied with the PP method, which was specifically designed to reduce the number of particles in multiple target tracking. Recall that PP method cannot be applied with FSLAM because FSLAM requires sampling of the complete trajectories of all targets. Then, the approximation of $p\left(\mathbf{P}^{k}\left|\mathbf{z}^{1:k}\right.\right)$ using MSLAM is expected to be better than the approximation of $p\left(\mathbf{P}^{k}\left|\mathbf{z}^{1:k}\right.\right)$ using FSLAM when there are multiple targets. The second one is, as the dimension of $\mathbf{P}^{k}$ is much lower than the dimension of $\mathbf{P}^{0:k}$, which is a vector of increasing dimension, the particle filter approximation of the pdf $p\left(\mathbf{P}^{k}\left|\mathbf{z}^{1:k}\right.\right)$ will be considerably better than the approximation of $p\left(\mathbf{P}^{0:k}\left|\mathbf{z}^{1:k}\right.\right)$ [22]. The difficulty of approximating $p\left(\mathbf{P}^{0:k}\left|\mathbf{z}^{1:k}\right.\right)$ gives rise to the map degeneracy problem which was introduced in subsection III-A and illustrated in Figure 2. Then, as the posterior pdf of the map in MSLAM depends on the particle approximation of $p\left(\mathbf{P}^{k}\left|\mathbf{z}^{1:k}\right.\right)$ rather than $p\left(\mathbf{P}^{0:k}\left|\mathbf{z}^{1:k}\right.\right)$, sample degeneracy is much less likely to adversely affect estimation of the map than in FSLAM. A better approximation of the posterior pdfs of current position of the targets and the map implies that MSLAM should perform better than FSLAM in a multiple target set-up.

We need to address two issues to completely define MSLAM. The first one is how to propagate the particles that represent the positions, which is done using the PP method, and the second is how to update the conditional pdfs of the targets' velocities and sensors' positions. In other words, we need an approximation of the posterior pdf at time $k+1$ in a similar form of (14) and (15) based on the pdf at time k, (14) and (15), and the model description in Section II. These subjects are addressed in the next sections.

IV. Approximation of the target state posterior

In this section, we explain how to sample the targets' positions, how to update the targets' velocities and how to calculate the weights of the particles.

A. *Sampling the targets' positions*

As we stated in the introduction, we use the PP method to sample the targets' positions. The PP method works with a posterior pdf for the multitarget state that can be factorised as [28]:

$$p\left(\mathbf{X}^{k}\left|\mathbf{z}^{1:k}\right.\right)\propto\prod_{j=1}^{t}\sum_{i=1}^{N_{par}}\delta\left(\mathbf{p}_{j}^{k}-\mathbf{p}_{j,i}^{k}\right)p\left(\mathbf{v}_{j}^{k}\left|\mathbf{p}_{j,i}^{k},\,\mathbf{z}^{1:k}\right.\right)\tag{16}$$

where, according to (15),

$$p\left(\mathbf{v}_j^k \left|\mathbf{p}_{j,i}^k,\, \mathbf{z}^{1:k}\right.\right) = \mathcal{N}\left(\mathbf{v}_j^k; \overline{\mathbf{v}}_{j,i}^k, \mathbf{\Sigma}_{j,i}^k\right) \tag{17}$$

and $\mathbf{p}_j^k$ is the position of target j at time k, $\mathbf{p}_{j,i}^k$ is the position of target j in particle i and $\propto$ indicates proportionality. It should be noted that (16) does not necessarily hold for a group of closely spaced targets as their positions conditioned on the measurements could be dependent. However, assumptions of this kind are quite common in multitarget tracking algorithms such as the well-known Joint Probabilistic Data Association Filter (JPDAF) [40].

Using (1), integrating out the states of the targets at time k and applying the Bayes' rule in (16), the updated pdf for the positions (the pdf we want to obtain samples from) is

$$p\left(\mathbf{P}^{k+1} \left|\mathbf{z}^{1:k+1}\right.\right) \propto p\left(\mathbf{z}^{k+1} \left|\mathbf{P}^{k+1}, \mathbf{z}^{1:k}\right.\right) \prod_{j=1}^{t} \sum_{i=1}^{N_{par}} \mathcal{N}\left(\mathbf{p}_j^{k+1}; \overline{\mathbf{p}}_{j,i}^{k+1|k}, \mathbf{\Xi}_{11,j,i}^{k+1}\right) \tag{18}$$

where

$$\mathcal{N}\left(\mathbf{p}_j^{k+1}; \overline{\mathbf{p}}_{j,i}^{k+1|k}, \mathbf{\Xi}_{11,j,i}^{k+1}\right) = \int p\left(\mathbf{p}_j^{k+1} \left|\mathbf{p}_{j,i}^k,\, \mathbf{v}_j^k\right.\right) \mathcal{N}\left(\mathbf{v}_j^k; \overline{\mathbf{v}}_{j,i}^k, \mathbf{\Sigma}_{j,i}^k\right) \mathrm{d}\mathbf{v}_j^k \tag{19}$$

and $\overline{\mathbf{p}}_{j,i}^{k+1|k} = \overline{\mathbf{p}}_{j,i}^k + \tau \overline{\mathbf{v}}_{j,i}^k$ and $\mathbf{\Xi}_{11,j,i}^{k+1} = \tau^2 \mathbf{\Sigma}_{j,i}^k + \mathbf{Q}_p$ are the predicted mean and covariance matrix, and $\mathbf{Q}_p$ is the process noise covariance matrix for the position elements according to (3).

The PP method samples an auxiliary variable for each target such that

$$p\left(\mathbf{P}^{k+1},\, \mathbf{a} \left|\mathbf{z}^{1:k+1}\right.\right) \propto p\left(\mathbf{z}^{k+1} \left|\mathbf{P}^{k+1}, \mathbf{z}^{1:k}\right.\right) \prod_{j=1}^{t} \mathcal{N}\left(\mathbf{p}_j^{k+1}; \overline{\mathbf{p}}_{j,a_j}^{k+1|k}, \mathbf{\Xi}_{11,j,a_j}^{k+1}\right) \tag{20}$$

where $\mathbf{a} = [a_1, ..., a_t]^T$, with $a_j \in \{1, ..., N_{par}\}$, indicates that we are choosing the component a_j on the mixture given by the posterior, equation (18), for target j [28]. The idea of sampling in a higher dimension has traditionally been used, for instance, in the auxiliary particle filter, and can report a lot of benefits [13], [41]. We should note that integrating out the auxiliary variables in (20), we get (18) so this definition of auxiliary variables is sound. Then, if we draw a sample from the joint density (20) and then discard the part of the sample that corresponds with $\mathbf{a}$, then we are producing a sample from (18) as required.

If we draw $\mathbf{P}_i^{k+1}$, $\mathbf{a}_i = \left[a_1^i,\, \ldots,\, a_t^i\right]^T$ from an importance density $q(\cdot)$ for $i = 1,\, \ldots,\, n$, the updated weights are calculated as:

$$w_i^{k+1} \propto \frac{p\left(\mathbf{z}^{k+1} \left|\mathbf{P}_i^{k+1},\, \mathbf{z}^{1:k}\right.\right) \prod_{j=1}^{t} \mathcal{N}\left(\mathbf{p}_{j,i}^{k+1}; \overline{\mathbf{p}}_{j,a_j^i}^{k+1|k}, \mathbf{\Xi}_{11,j,a_j^i}^{k+1}\right)}{q\left(\mathbf{P}_i^{k+1},\, \mathbf{a}_i \left|\mathbf{z}^{1:k+1}\right.\right)} \tag{21}$$

The importance density in the PP filter is [28]:

$$q\left(\mathbf{P}^{k+1},\,\mathbf{a}\left|\mathbf{z}^{1:k+1}\right.\right)=\prod_{j=1}^{t}b_j\left(\mathbf{p}_j^{k+1}\right)\mathcal{N}\left(\mathbf{p}_j^{k+1};\overline{\mathbf{p}}_{j,a_j}^{k+1|k},\boldsymbol{\Xi}_{11,j,a_j}^{k+1}\right) \tag{22}$$

where $b_j\left(\mathbf{p}_j^{k+1}\right)$ are the first-stage weights for target j. The PP method suggests that $b_j\left(\mathbf{p}_j^{k+1}\right)$ should be selected proportional to [28]:

$$p\left(\mathbf{z}^{k+1}\left|\mathbf{p}_j^{k+1},\hat{\mathbf{P}}_{-\{j\}}^{k+1|k},\mathbf{z}^{1:k}\right.\right)=\int p\left(\mathbf{z}^{k+1}\left|\mathbf{p}_j^{k+1},\hat{\mathbf{P}}_{-\{j\}}^{k+1|k},\,\mathbf{m}\right.\right)p\left(\mathbf{m}\left|\mathbf{p}_j^{k+1},\hat{\mathbf{P}}_{-\{j\}}^{k+1|k},\mathbf{z}^{1:k}\right.\right)\mathrm{d}\mathbf{m} \tag{23}$$

where $\hat{\mathbf{P}}_{-\{j\}}^{k+1|k}$ represents the predicted positions of the targets averaged over all the particles except for target j. The first-stage weights are used to determine which single target sub-particles will be used to construct multitarget particles. These weights should be such that sub-particles which well-represent a given target are selected with high probability. At the same time, the weight assigned to a sub-particle should account for the influence of neighbouring targets. Eq. (23) fulfils both these requirements by using the likelihood of a multitarget state composed of a sample from the transition prior for the sub-particle in question and the particle filter approximation of the predicted positions of the remaining targets.

The problem with (23) is that it is computationally demanding to calculate. For every $\mathbf{p}_j^{k+1}$, the approximation to the posterior pdf of the map $p\left(\mathbf{m}\left|\mathbf{p}_j^{k+1},\hat{\mathbf{P}}_{-\{j\}}^{k+1|k},\mathbf{z}^{1:k}\right.\right)$ changes and we need to approximate an integral based on it. Then, to lower the computational burden, we propose to use:

$$b_j\left(\mathbf{p}_j^{k+1}\right)\propto\prod_{r=1}^{M}\int p\left(\mathbf{z}_r^{k+1}\left|\mathbf{p}_j^{k+1},\hat{\mathbf{P}}_{-\{j\}}^{k+1|k},\,\mathbf{m}_r\right.\right)\mathcal{N}\left(\mathbf{m}_r;\overline{\mathbf{m}}_r^k,\,\boldsymbol{\Theta}_r^k\right)\mathrm{d}\mathbf{m}_r \tag{24}$$

where $\left\{\overline{\mathbf{m}}_1^k,\,\boldsymbol{\Theta}_1^k,\,\ldots,\overline{\mathbf{m}}_M^k,\,\boldsymbol{\Theta}_M^k\right\}$ refers to the set of means and covariance matrices that represent an independent Gaussian approximation to the marginal pdf of the map at time k, which is obtained by integrating out the positions and velocities in (14). Finally, the integral in (24) is approximated using the unscented transformation [42]. The steps of the approximate sampling procedure of the PP method are shown in Table I and a thorough explanation is given in [28].

B. Update of the targets' velocities

Once we have drawn the particles of the positions at time $k+1$, we explain how to update the velocities of the targets. It is shown in Appendix I that:

$$p\left(\mathbf{V}^{k+1}\left|\mathbf{P}^{k+1},\,\mathbf{z}^{1:k}\right.\right)\propto\prod_{j=1}^{t}\sum_{i=1}^{N_{par}}\mathcal{N}\left(\mathbf{u}_j^{k+1};\overline{\mathbf{u}}_{j,i}^{k+1},\boldsymbol{\Xi}_{j,i}^{k+1}\right) \tag{25}$$

where $\mathbf{u}_j^{k+1}=\left[\mathbf{p}_j^{k+1},\mathbf{v}_j^{k+1}\right]^T$ and

$$\overline{\mathbf{u}}_{j,i}^{k+1}=\left[\overline{\mathbf{p}}_{j,i}^k+\tau\overline{\mathbf{v}}_{j,i}^k,\overline{\mathbf{v}}_{j,i}^k\right]^T \tag{26}$$

Table I – Parallel Partition Subroutine for Multitarget SLAM

PP Subroutine for target j

- For each particle $i = 1, ..., N_{par}$:
  - Get a sample for the position $\mathbf{p}^*_{j,i} \backsim N\left(\mathbf{p}_j^{k+1}, \overline{\mathbf{p}}_{j,i}^{k+1|k}, \mathbf{\Xi}_{11,j,i}^{k+1}\right)$.
  - Calculate $b_j\left(\mathbf{p}^*_{j,i}\right)$ using (24).
- Normalise $b_j\left(\mathbf{p}^*_{j,i}\right)$ to sum to one over $i = 1, ...N_{par}$.
- Resampling step for each track. For each particle $i = 1, ..., N_{par}$:
  - Sample an index p from the distribution defined by $b_j\left(\mathbf{p}^*_{j,l}\right)$ over $l = 1, ..., N_{par}$.
  - Set $\mathbf{p}_{j,i}^{k+1} = \mathbf{p}^*_{j,p}$ and the parent's index $a_j^i = p$.

$$\mathbf{\Xi}_{j,i}^{k+1} = \begin{pmatrix} \mathbf{\Xi}_{11,j,i}^{k+1} & \mathbf{\Xi}_{12,j,i}^{k+1} \\ \mathbf{\Xi}_{21,j,i}^{k+1} & \mathbf{\Xi}_{22,j,i}^{k+1} \end{pmatrix} = \begin{pmatrix} \tau^2\mathbf{\Sigma}_{j,i}^k + \mathbf{Q}_p & \tau\mathbf{\Sigma}_{j,i}^k + \mathbf{Q}_{pv} \\ \tau\mathbf{\Sigma}_{j,i}^k + \mathbf{Q}_{vp} & \mathbf{\Sigma}_{j,i}^k + \mathbf{Q}_v \end{pmatrix} \tag{27}$$

where $\mathbf{Q}_p$, $\mathbf{Q}_v$ are the covariance matrices of the positions and velocities, and $\mathbf{Q}_{pv}$ is the covariance matrix between the positions and velocities according to (3).

Equation (25) can be written as

$$p\left(\mathbf{V}^{k+1}\left|\mathbf{P}^{k+1}, \mathbf{z}^{1:k}\right.\right) = \prod_{j=1}^{t}\sum_{i=1}^{N_{par}} \rho_{1,i}\left(\mathbf{p}_j^{k+1}\right)\mathcal{N}\left(\mathbf{v}_j^{k+1}; \overline{\mathbf{v}}_{j,i}^{k+1|k}\left(\mathbf{p}_j^{k+1}\right), \mathbf{\Sigma}_{j,i}^{k+1|k}\right) \tag{28}$$

where

$$\tilde{\rho}_{1,i}\left(\mathbf{p}_j^{k+1}\right) = \mathcal{N}\left(\mathbf{p}_j^{k+1}; \overline{\mathbf{p}}_{j,i}^{k+1|k}, \mathbf{\Xi}_{11,j,i}^{k+1}\right) \tag{29}$$

and $\rho_{1,i}\left(\mathbf{p}_j^{k+1}\right)$ are the normalised $\tilde{\rho}_{1,i}\left(\mathbf{p}_j^{k+1}\right)$ so that they sum to one in the variable i and, $\overline{\mathbf{v}}_{j,i}^{k+1|k}\left(\mathbf{p}_j^{k+1}\right)$ and $\mathbf{\Sigma}_{j,i}^{k+1|k}$ are calculated using the fundamental equations of linear estimation [33]:

$$\overline{\mathbf{v}}_{j,i}^{k+1|k}\left(\mathbf{p}_j^{k+1}\right) = \overline{\mathbf{v}}_{j,i}^{k} + \mathbf{\Xi}_{12,j,i}^{k+1}\left(\mathbf{\Xi}_{11,j,i}^{k+1}\right)^{-1}\left(\mathbf{p}_j^{k+1} - \overline{\mathbf{p}}_{j,i}^{k+1|k}\right) \tag{30}$$

$$\mathbf{\Sigma}_{j,i}^{k+1|k} = \mathbf{\Xi}_{22,j,i}^{k+1} - \mathbf{\Xi}_{12,j,i}^{k+1}\left(\mathbf{\Xi}_{11,j,i}^{k+1}\right)^{-1}\mathbf{\Xi}_{21,j,i}^{k+1} \tag{31}$$

Then, according to (28), the posterior of the targets' velocities is a product of Gaussian mixtures. As for the next step of the recursion, see (16) and (17), we assume the velocity of a target is approximated by a Gaussian pdf, we approximate each Gaussian mixture in (28) by a Gaussian pdf [13]:

$$p\left(\mathbf{V}^{k+1}\left|\mathbf{P}^{k+1}, \mathbf{z}^{1:k}\right.\right) \approx \prod_{j=1}^{t}\mathcal{N}\left(\mathbf{v}_j^{k+1}; \overline{\mathbf{v}}_{j,i}^{k+1}, \mathbf{\Sigma}_{j,i}^{k+1}\right) \tag{32}$$

where

$$\overline{\mathbf{v}}_{j,i}^{k+1} = \sum_{p=1}^{N_{par}} \rho_{1,p}\left(\mathbf{p}_j^{k+1}\right)\overline{\mathbf{v}}_{j,p}^{k+1|k}\left(\mathbf{p}_j^{k+1}\right) \tag{33}$$

$$\boldsymbol{\Sigma}_{j,i}^{k+1} = \sum_{p=1}^{N_{par}} \rho_{1,p}\left(\mathbf{p}_j^{k+1}\right) \cdot \left[\boldsymbol{\Sigma}_{j,p}^{k+1|k} + \left(\overline{\mathbf{v}}_{j,p}^{k+1|k}\left(\mathbf{p}_j^{k+1}\right) - \overline{\mathbf{v}}_{j,i}^{k+1}\right)\left(\overline{\mathbf{v}}_{j,p}^{k+1|k}\left(\mathbf{p}_j^{k+1}\right) - \overline{\mathbf{v}}_{j,i}^{k+1}\right)^T\right] \tag{34}$$

It should be mentioned that approximating a Gaussian mixture by independent Gaussian pdfs has traditionally been done in the tracking community, for instance, in the well-known JPDAF [40].

C. Calculation of the particles' weights

In subsection IV-A, we explained how to propagate the particles using the PP method. Here, we show how to calculate their weights. Substituting (22) into (21), the particle weights are:

$$w_i^{k+1} \propto \frac{p\left(\mathbf{z}^{k+1}\left|\mathbf{P}_i^{k+1}, \mathbf{z}^{1:k}\right.\right)}{\prod_{j=1}^{t} b_j\left(\mathbf{p}_{j,i}^{k+1}\right)} \tag{35}$$

In order to calculate (35), we need to obtain

$$p\left(\mathbf{z}^{k+1}\left|\mathbf{P}_i^{k+1}, \mathbf{z}^{1:k}\right.\right) = \int p\left(\mathbf{z}^{k+1}\left|\mathbf{P}_i^{k+1}, \mathbf{m}\right.\right) \cdot p\left(\mathbf{m}\left|\mathbf{P}_i^{k+1}, \mathbf{z}^{1:k}\right.\right) \mathrm{d}\mathbf{m} \tag{36}$$

The pdf $p\left(\mathbf{m}\left|\mathbf{P}^{k+1}, \mathbf{z}^{1:k}\right.\right)$ is calculated in Appendix II:

$$p\left(\mathbf{m}\left|\mathbf{P}^{k+1}, \mathbf{z}^{1:k}\right.\right) = \sum_{i=1}^{N_{par}} \rho_{2,i}\left(\mathbf{P}^{k+1}\right) \prod_{r=1}^{M} \mathcal{N}\left(\mathbf{m}_r; \overline{\mathbf{m}}_{r,i}^{k}, \boldsymbol{\Theta}_{r,i}^{k}\right) \tag{37}$$

where

$$\tilde{\rho}_{2,i}\left(\mathbf{P}^{k+1}\right) = \prod_{j=1}^{t} \mathcal{N}\left(\mathbf{p}_j^{k+1}; \overline{\mathbf{p}}_{j,i}^{k+1|k}, \boldsymbol{\Xi}_{11,j,i}^{k+1}\right) \tag{38}$$

and $\rho_{2,i}\left(\mathbf{P}^{k+1}\right)$ are the normalised $\tilde{\rho}_{2,i}\left(\mathbf{P}^{k+1}\right)$ so that they sum to one. Then, (37) indicates that the distribution of the map given the new positions of the targets at time $k+1$ and without taking into account the measurements at time $k+1$ is a mixture of independent Gaussian distributions that represent the maps of the particles at time k. This mixture of maps has one positive effect and one drawback. The positive effect is that it tends to reduce map degeneracy because the maps are shared among the particles in the following way. The map for a certain position at time $k+1$ is a mixture of Gaussians determined by the values of $\rho_{2,i}\left(\mathbf{P}^{k+1}\right)$. The value $\rho_{2,i}\left(\mathbf{P}^{k+1}\right)$ indicates the probability that the position $\mathbf{P}^{k+1}$ at time $k+1$ has been drawn from particle i at time k. This way if a particle at time $k+1$ matches only one particle at time k, it will inherit its map. On the contrary, if there are several particles at time k that match the new particle, the new particle will inherit a map that is a combination of those previous maps. The drawback is that the new map is not Gaussian distributed any more and that the sensors' positions become correlated. Therefore, in order to be able to estimate recursively the positions of the sensors using (15), we approximate the probability in (37) by a set of independent Gaussian distributions whose mean and covariance matrix are equal to the Gaussian mixture in (37). Using the mean and covariance matrix of a mixture of Gaussians [13]:

$$p\left(\mathbf{m}\left|\mathbf{P}_i^{k+1},\mathbf{z}^{1:k}\right.\right)\approx\prod_{r=1}^{M}\mathcal{N}\left(\mathbf{m}_r;\,\overline{\mathbf{m}}_{r,i}^{k+1|k},\,\mathbf{\Theta}_{r,i}^{k+1|k}\right)\tag{39}$$

where

$$\overline{\mathbf{m}}_{r,i}^{k+1|k}=\sum_{p=1}^{N_{par}}\rho_{2,p}\left(\mathbf{P}^{k+1}\right)\overline{\mathbf{m}}_{r,p}^{k}\tag{40}$$

$$\mathbf{\Theta}_{r,i}^{k+1|k}=\sum_{p=1}^{N_{par}}\rho_{2,p}\left(\mathbf{P}^{k+1}\right)\cdot\left[\mathbf{\Theta}_{r,p}^{k}+\left(\overline{\mathbf{m}}_{r,p}^{k}-\overline{\mathbf{m}}_{r,i}^{k+1|k}\right)\left(\overline{\mathbf{m}}_{r,p}^{k}-\overline{\mathbf{m}}_{r,i}^{k+1|k}\right)^T\right]\tag{41}$$

Finally, we substitute (39) into (36) and we approximate the integral in (36) by applying the unscented transformation to each sensor position [42] to obtain an approximation to (35).

V. Sensors' positions update

So far we have explained how to sample the positions, update the velocities and calculate the weights of the particles. In this section, we explain the last step to completely define the MSLAM recursion, i.e., how we use the sensors' measurements to update the sensors' positions. Applying the Bayes' rule for the map of particle i to (39) and (4):

$$p\left(\mathbf{m}\left|\mathbf{P}_i^{k+1},\,\mathbf{z}^{1:k+1}\right.\right)\propto\prod_{r=1}^{M}p\left(\mathbf{z}_r^{k+1}\left|\mathbf{P}_i^{k+1},\,\mathbf{m}_r\right.\right)p\left(\mathbf{m}_r\left|\mathbf{P}_i^{k+1},\,\mathbf{z}^{1:k}\right.\right)\tag{42}$$

In this section, we drop the indexes that indicate the particle and time for the sake of clarity but noting that the Bayes' rule for the sensors' positions needs to be performed for each particle and sensor at each time. Hence, the algorithm described in this section has to be carried out particle by particle and sensor by sensor. Usually the Bayesian recursion for the map is approximated using the EKF or UKF [7], [9]. However, it is known that measurement nonlinearities become significant when the measurement noise variance is small compared to the prior variance [30]. In this case, conventional KF-type algorithms, such as the EKF or UKF, perform badly and the recently proposed Truncated Unscented Kalman Filter (TUKF) was designed to overcome their deficiencies [31]. The TUKF is based on the idea that a more accurate approximation of the first two moments of the posterior can be obtained by applying a KF to a modified prior that is a mixture of the original prior plus a truncated version of it. The aim is to reduce the prior variance so that nonlinearities become milder and the KF performs well.

The Single Point TUKF (SP-TUKF)[1], which is the approximation of the TKF we employ in this paper, uses the same set of sigma-points as the UKF plus one extra sigma point $\tilde{\mathbf{m}}_r(\mathbf{z})$ that is chosen

[1]In this paper we refer to the SP-TUKF in [31] simply as TUKF as context permits.

according to the likelihood [31], [32]. Then, what TUKF proposes is to use another distribution for the prior $p_2(\mathbf{m}_r)$, rather than the real prior $p_0(\mathbf{m}_r)$, approximated by [32]:

$$p_2(\mathbf{m}_r) \approx \alpha_{sp}\delta(\mathbf{m}_r - \tilde{\mathbf{m}}_r(\mathbf{z})) + (1-\alpha_{sp})\tilde{p}_0(\mathbf{m}_r) \tag{43}$$

where $\tilde{\mathbf{m}}_r(\mathbf{z})$ is the extra sigma point that represents the truncated prior, α_{sp} is the weight of the extra sigma point $\tilde{\mathbf{m}}_r(\mathbf{z})$ and, $\tilde{p}_0(\mathbf{m}_r)$ is the unscented approximation of the prior $p_0(\mathbf{m}_r)$ [42]. The unscented approximation of $p_0(\mathbf{m}_r)$ can be written as [43], [44]:

$$p_0(\mathbf{m}_r) \approx \tilde{p}_0(\mathbf{m}_r) = \sum_{i=1}^{N_s} w_s^i \delta(\mathbf{m}_r - \mathbf{m}_{r,i}) \tag{44}$$

where N_s is the number of sigma points, $\mathbf{m}_{r,i}$ is the sigma point i for sensor r, and w_s^i is the weight of sigma point i. The number N_s of sigma points depends on the dimension of the state and the number of moments of the distribution we want to approximate. The selection of the sigma points and the way to perform a KF update stage using them are beyond the scope of this paper and can be found in [42]. There are many possible choices of sigma points for representing $p_0(\mathbf{m}_r)$. We use the unscented transformation as described in Section IV of [42] in which the weight of the sigma point located on the mean is 1/3.

A. Selection of α_{sp}

We use the value of α_{sp} proposed in [31], [32]:

$$\alpha_{sp} = \alpha_{max}\frac{\gamma \operatorname{tr}(\mathbf{P}_{p,0})}{\gamma \operatorname{tr}(\mathbf{P}_{p,0}) + (1-\gamma)\operatorname{tr}(\mathbf{P}_{p,1})} \tag{45}$$

where $\mathbf{P}_{p,0}$ is the covariance matrix of the prior for the particle we are considering, given by (41), $\mathbf{P}_{p,1}$ is the covariance matrix of the approximated truncated prior, $\gamma \in [0,\, 1]$ is a parameter that controls the weights of the traces of the covariance matrices to select α_{sp}, $\alpha_{max} < 1$ is the maximum value α_{sp} can take, and $\operatorname{tr}(\mathbf{A})$ denotes the trace of matrix $\mathbf{A}$. Following a similar procedure as in [31], when the measurement vector dimension is different to the state vector dimension, $\mathbf{P}_{p,1}$ can be approximated as

$$\mathbf{P}_{p,1} = \left(\tilde{\mathbf{H}}^T\tilde{\mathbf{H}}\right)^{-1} \tag{46}$$

where we have assumed that the measurement noise covariance matrix is the identity matrix and

$$\tilde{\mathbf{H}} = \left[\nabla_{\mathbf{m}_r}\mathbf{h}^T(\mathbf{m}_r)\right]^T\Big|_{\mathbf{m}_r=\tilde{\mathbf{m}}_r(\mathbf{z})} \tag{47}$$

is the Jacobian of $\mathbf{h}(\mathbf{m}_r)$ evaluated at $\tilde{\mathbf{m}}_r(\mathbf{z})$, $\mathbf{h}(\mathbf{m}_r)$ is given by (6) particularised for the current multitarget state and sensor.

The motivation of using (45) is the following. The approximation of the truncated prior by a single sigma point improves when the trace of the covariance matrix of the approximated truncated prior decreases. Additionally, KF applied to $p_0(\cdot)$ is expected to perform worse when the prior uncertainty is large so the trace of the covariance matrix of $p_0(\cdot)$ is high. Then, α_{sp} should increase when the trace of the covariance matrix of the truncated prior decreases and when the trace of the covariance matrix of $p_0(\cdot)$ increases.

Additionally, we set a threshold, Γ_E. If the sum of the squared measurements in the antennas is under this threshold, we set $\alpha_{sp} = 0$. There are two reasons for using this threshold. If the energy of the received signal is not high enough, the value of α_{sp} will be low, then, the TUKF tends to be the UKF. Therefore, we can avoid calculating the extra sigma point and α_{sp} and, then, reduce computational expense. Another reason is that if the energy is not high enough, for example, the received signal is only noise, then, the extra sigma point will be placed practically at random and, therefore, there is no point in using it.

B. Selection of the extra sigma point

The extra sigma point $\tilde{\mathbf{m}}_r(\mathbf{z})$ for Gaussian noise should minimise $(\mathbf{z}_r - \mathbf{h}(\mathbf{m}_r))^T \mathbf{R}^{-1} (\mathbf{z}_r - \mathbf{h}(\mathbf{m}_r))$ so that the likelihood attains its maximum [31] where $\mathbf{R} = \mathbf{I}_{N_a}$ is the measurement noise covariance matrix for sensor r and $\mathbf{z}_r$ is the measurement of sensor r. Then, we could use Nonlinear Least Squares to do so. However, as we also have some prior knowledge about the location of the sensor, equation (11), we use a penalty function to increase the chance that the extra sigma point is in a region where the prior is non-negligible and improve the convergence of the algorithm. We define:

$$\mathbf{L}(\mathbf{m}_r) = \left[(\mathbf{z}_r - \mathbf{h}(\mathbf{m}_r))^T, g(\mathbf{m}_r)\right]^T \tag{48}$$

where

$$g(\mathbf{m}_r) = \begin{cases} 0 & \left(\mathbf{m}_r - \overline{\mathbf{m}}_r^0\right)^T \left(\boldsymbol{\Theta}_r^0\right)^{-1} \left(\mathbf{m}_r - \overline{\mathbf{m}}_r^0\right) < \Gamma_p \\ c_p \left(\left(\mathbf{m}_r - \overline{\mathbf{m}}_r^0\right)^T \left(\boldsymbol{\Theta}_r^0\right)^{-1} \left(\mathbf{m}_r - \overline{\mathbf{m}}_r^0\right) - \Gamma_p\right) & \text{otherwise} \end{cases} \tag{49}$$

and $\overline{\mathbf{m}}_r^0$ and $\boldsymbol{\Theta}_r^0$ are the initial mean and covariance matrix of the position of sensor r, $c_p > 0$ is a parameter of the penalty function and Γ_p sets the region where the penalty function applies. Then, we aim to minimise the objective function:

$$J(\mathbf{m}_r) = \mathbf{L}^T(\mathbf{m}_r)\,\mathbf{L}(\mathbf{m}_r) \tag{50}$$

We also assume that $N_a > 2$ (there are more than two antennas) so that the problem is overdetermined and we use the Gauss-Newton method to solve it [45]. It should be noted that we need to minimise

Table II – Subroutine for updating the sensors' positions

- For each sensor to update (based on Γ_{sens})
  - Sum the squared signal for all the antennas in that sensor $E_s = \sum_{i=1}^{N_a} z_{r,i}^2$ where $z_{r,i}$ is the ith component of $\mathbf{z}_r$.
    - IF $E_s > \Gamma_E$
      - Calculate the extra sigma point and α_{sp} for each particle according to Section V.
      - Apply the TUKF for that sensor in each particle.
    - ELSE
      - Set $\alpha_{sp} = 0$, i.e., apply the UKF for that sensor in each particle.
    - END

(50) for every particle which is very time-consuming. Then, what we propose is a two-phase procedure in which we minimise (50) globally, i.e., for the mean positions of the targets and then, we refine the estimate for each particle.

1) Global minimisation: In this step of the algorithm, we minimise the objective function, equation (50), for the mean positions of the targets (considering all the particles), i.e., we use these positions to calculate function $\mathbf{h}(\cdot)$. Then, the Gauss-Newton method uses the following iteration to obtain the sensor position $\mathbf{m}_r$ that minimises the objective function, equation (50) [45]:

$$\tilde{\mathbf{m}}_{r,k+1} = \tilde{\mathbf{m}}_{r,k} - \gamma_g \left(\mathbf{L'}_k^T \mathbf{L'}_k\right)^{-1} \mathbf{L'}_k^T \mathbf{L}\left(\tilde{\mathbf{m}}_{r,k}\right) \tag{51}$$

where $\tilde{\mathbf{m}}_{r,k}$ is the position of sensor r at iteration k, $\mathbf{L'}_k$ is the Jacobian of $\mathbf{L}(\mathbf{m}_r)$ evaluated at $\tilde{\mathbf{m}}_{r,k}$ and γ_g is a parameter that controls the convergence rate. The starting point of the iteration is the mean of the sensor at the current time step, equation (40).

Finally, the convergence criterion of the recursion (51) is

$$\left|\frac{J\left(\tilde{\mathbf{m}}_{r,k+1}\right) - J\left(\tilde{\mathbf{m}}_{r,k}\right)}{J\left(\tilde{\mathbf{m}}_{r,k}\right)}\right| < 10^{-5} \tag{52}$$

or if the number of steps of the recursion reaches N_{rg}.

2) Local minimisation: The output of the global minimisation is used as the input of the local minimisation. In this step, we calculate a refinement of the position of the sensor for each particle. Thus, for each particle we perform again iteration (51) bearing in mind that function $\mathbf{h}(\cdot)$ is different for each particle as it depends on the targets' positions, which are different for each particle. The convergence criterion of the recursion is again (52) or if the number of steps is equal to N_{rl}.

Additionally, in the implementation of the algorithm, we do not update all the sensors' positions and we do not consider all the sensors to propagate the particles at every time step because it is very time-

Table III – Proposed algorithm for Multitarget SLAM

- Calculate the sensors to update, using Γ_{sens}.
- For each target, sample its position using PP subroutine, see Table I.
- Map and velocity mixing step: mix the maps of the sensors via (39) and the velocities via (32).
- Update the weights of each particle using (35).
- Update the positions of the sensors using (42) following the steps given in Table II and Section V.
- Perform resampling to obtain an evenly weighted particle set.

consuming. We calculate the set of sensors that is used to do all the calculations based on a threshold Γ_{sens}. If the distance from any target to a sensor is less than Γ_{sens} then the sensor is considered in the current step of the algorithm. To sum up, the steps of the subroutine for updating the sensors' position using the TUKF and the steps of the complete algorithm for multitarget SLAM are shown in Tables II and III, respectively.

In addition, if the surveillance area were large enough we could consider applying target clustering as in [26]–[28]. We have not introduced it here for the clarity of presentation and because in the scenario analysed in the following section, the targets are close to each other and the likelihood does not factorise over the clusters as in [27].

VI. Simulation results

In this section, we compare MSLAM with FSLAM and with a filter that uses the PP method for tracking but without updating the sensors' positions. The last method is a suboptimal approach that takes into account the initial uncertainty in the positions of the sensors but does not update them. This method is analysed to show the improvement that can be made by estimating simultaneously the positions of the targets and the sensors. The Bayes' rule for updating the conditional posterior pdf of the sensors' positions for both SLAM methods is performed using the UKF and the TUKF.

The FSLAM algorithm for multitarget SLAM cannot perform subparticle crossover as we stated previously, so we use an auxiliary particle filter [41] to propose the new particles jointly for all the targets. In our implementation of the auxiliary particle filter, the predicted mean of the multitarget state is used to compute the first-stage weights.

We use three targets to evaluate the performances of the algorithms with the trajectories shown in Fig. 4. The sampling period of the trajectories is $\tau = 0.5\,\mathrm{s}$ and there are $l = 99$ time steps in the simulation. The surveillance area is a square whose side is 150 m. We will study the algorithms varying the initial uncertainty of the sensors' positions for a multiple target and a single target set-up. The initial mean

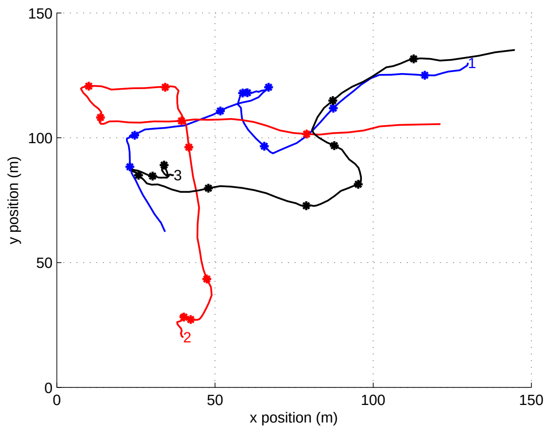


Figure 4 – Scenario of the simulations: Each target is identified at the beginning of its trajectory by a number. The trajectories have an asterisk to indicate the target position every ten time steps.

Table IV – Model parameters

Parameter	σ_u	SNR_0	d_0	β	N_a	σ_0
Value	$1.8\ \mathrm{m/s}^{3/2}$	32 dB	1 m	1	4	0.1 m

positions of the sensors form a grid whose side is 10 m. The initial covariance matrix of each sensor position is

$$\boldsymbol{\Theta}_j^0 = \sigma_s^2 \mathbf{I}_2 \tag{53}$$

The prior pdf for the target's position is:

$$\mathbf{p}_i^0 \backsim \mathcal{N}\left(\mathbf{p}_i^0;\, \overline{\mathbf{p}}_i^0,\, \sigma_0^2 \mathbf{I}_2\right) \tag{54}$$

for $i \in \{1,\, ...,\, t\}$ where $\overline{\mathbf{p}}_i^0$ is the expected position of target i at time 0 and $\sigma_0^2\mathbf{I}_2$ is the covariance matrix of the initial position at time 0 which is assumed to be the same for all the targets for simplicity. In each Monte Carlo run, $\overline{\mathbf{p}}_i^0$ is drawn from a Gaussian distribution whose mean is the real position of target i at time 0 and whose covariance matrix is $\sigma_0^2\mathbf{I}_2$. In addition, the initial target velocity is taken as in [27]. For each target in each particle, we draw two integers η_1, η_2 such that $\mathrm{P}\left(\eta_i = j\right) = 1/2$ for $j \in \{-1,\, 1\}$ $i \in \{1,\, 2\}$ and then select the initial distribution of the target velocity to be Gaussian with mean $[\eta_1,\, \eta_2]^T$ and covariance matrix $\mathbf{I}_2$. The parameters of the model used in the simulations are those shown in Table IV. The sensor's antennas are aiming at directions $(0^{\mathrm{o}},\, 90^{\mathrm{o}},\, 180^{\mathrm{o}},\, 270^{\mathrm{o}})$. The parameters of the algorithms used in the simulations are those shown in Table V.

Table V – Algorithm parameters

Parameter	Γ_{sens}	Γ_E	Γ_p	c_p	N_{rg}	N_{rl}	γ_g	α_{max}	γ	N_{par}
Value	25 m	40	9	100	100	100	0.1	0.8	0.1	300

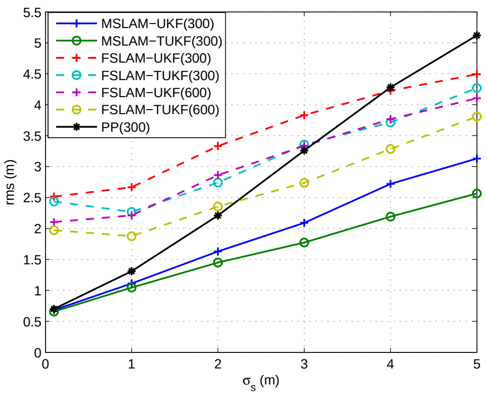
We evaluate the tracking performance calculating the root mean square (RMS) of the position error for the targets and the sensors by a Monte Carlo simulation of $m = 50$ realisations. To assess the tracking performance, we make a *Track-to-truth Assignment Table* [46] in which each track is associated with one target or with none at each time step depending on the distance from the track to the real position of the target and a threshold that is taken to be $\Lambda = 10$ m. If the distance from a track to a target exceeds that threshold, the track cannot be associated with that target.

Let $\hat{\mathbf{p}}_i^k(j)$, $i = 1, \dots, t$, $k = 1, \dots, l$, $j = 1, \dots, m$ denote the position estimate of the ith target on the jth Monte Carlo realisation at time k, $e_i^k(j) = \left\|\mathbf{p}_i^k - \hat{\mathbf{p}}_i^k(j)\right\|_2$ denote the position error for the ith target on the jth Monte Carlo realisation at time k and $\chi_{i,j,k} = 1$ if $e_i^k(j) < \Lambda$ and zero otherwise. The variable $\chi_{i,j,k}$ indicates if the ith target is in track at time k on the jth Monte Carlo realisation. The RMS position error E for the targets is then calculated by

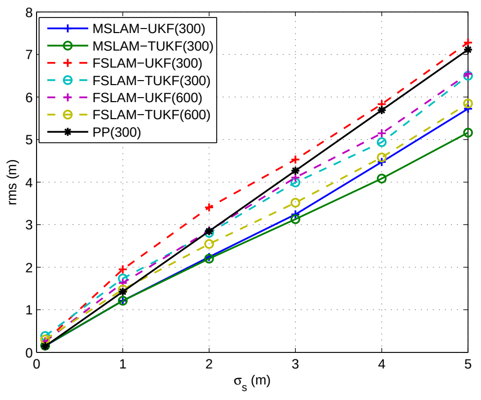
$$E = \sqrt{\frac{\sum_{j=1}^{m}\sum_{k=1}^{l}\sum_{i=1}^{t}\left(e_i^k(j)\right)^2 \cdot \chi_{i,j,k}}{\sum_{j=1}^{m}\sum_{k=1}^{l}\sum_{i=1}^{t}\chi_{i,j,k}}} \tag{55}$$

The RMS position error for the targets and sensors are shown in Fig. 5. To be fair with the comparisons, we also include the results for FSLAM with 600 particles as it has roughly the same computational burden in our implementation as MSLAM with 300 particles. The RMS position error for the sensors for the PP method corresponds to the average initial error as the sensors' positions are not updated, see Fig. 5b. It has been obtained by simulation although we know its theoretical value is $\sqrt{2}\sigma_s$. Therefore, the distance of this line to the line of another algorithm indicates the improvement or deterioration of the knowledge of the sensors' positions given by the algorithm. One should note that it may seem that the localisation of the sensors for these algorithms is poor by looking at Fig. 5 but this error is averaged over all the sensors and not all the sensors are updated because the targets do not approach them.

There are three characteristics of the results in Fig. 5 that are of interest: MSLAM versus FSLAM, the UKF versus the TUKF and, SLAM versus PP method with only tracking. Firstly, it can be seen that FSLAM performance is rather poor compared to MSLAM. This is due to the advantage of using subparticle crossover and a lower map degeneracy of MSLAM. Even if we double the number of particles of FSLAM, it is still far from the performance of MSLAM. Secondly, the improvement of



(a)



(b)

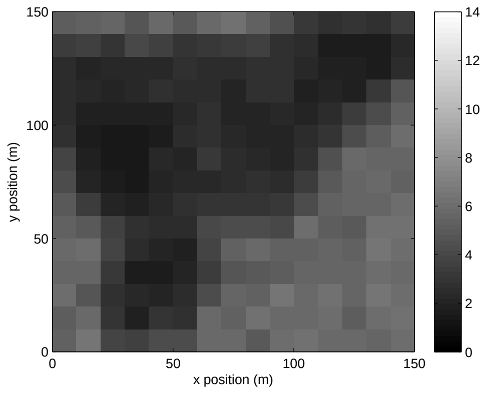
Figure 5 – Performance of the algorithms for three targets: (a) RMS position error for the targets, (b) RMS position error for the sensors. In both cases, MSLAM-TUKF outperforms the rest of the algorithms.

the estimate given by the TUKF over the UKF is more significant as σ_s increases. This happens because, as σ_s increases, the prior knowledge about the sensors' positions is increasingly vague and, then, the measurement carries more information compared to the prior. In this case, the TUKF clearly outperforms the UKF [31]. When the prior knowledge is very accurate compared with the information of the measurement, α_{sp} tends to zero, equation (45), and the TUKF comes down to the UKF. Therefore, for low σ_s, both methods have the same performance. Thirdly, if SLAM is not carried out properly, i.e., FSLAM is used in a multiple target set-up, it is better to use the PP method without updating the sensors' positions for $\sigma_s \leq 2\,\mathrm{m}$. In this case for FSLAM, simultaneously estimating the map and the targets' positions is worse than just doing tracking. For MSLAM, simultaneously estimating the map and the targets' positions is never worse than just doing tracking as expected. As σ_s grows, both SLAM methods provide an advantage over only tracking. However, when the uncertainty in the sensors' positions is very low, $\sigma_s = 0.1\,\mathrm{m}$, MSLAM and PP performances are approximately the same. In that case, it might not be worth updating the sensors' positions as the improvement is negligible.

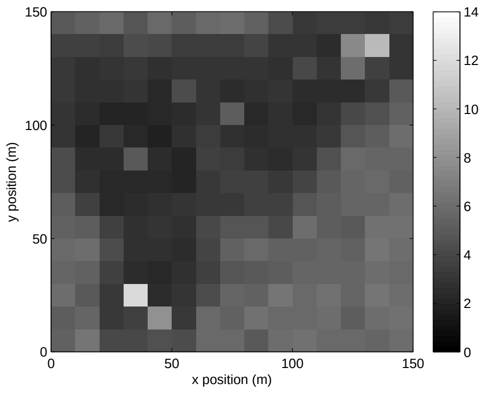
We now analyse more deeply the estimate of the map for $\sigma_s = 4\,\mathrm{m}$. In Fig. 6, we show the final RMS position error map for the sensors for the algorithms. Every pixel of the figure represents the RMS for the sensor whose prior expected position at time 0 is located in that part of the surveillance area. The axes are labelled according to the initial mean positions of the sensors. We suggest having a look at the trajectories of the targets in Fig. 4 to understand Fig. 6 better. It is clear that the estimates of the positions of the sensors that are located far from the trajectories of the targets are not improved as they acquire no useful information regarding their positions. The main difference between using the UKF and the TUKF lies in the estimate of the sensors' positions near the initial positions of the targets. The UKF fails to provide a good estimate of the positions of such sensors. This affects the tracking and localisation at future times. This occurs because, initially, the information given by the measurements could be much higher than the information of the prior and, therefore, the UKF does not work properly [31]. The main difference between FSLAM-TUKF and MSLAM-TUKF are the estimates of the sensors' positions that are updated after a certain time has passed. This is because FSLAM does not work well in a multiple target set-up and cannot track the targets properly. As tracking performance is poor, map estimation in FSLAM-TUKF is poor too. Then, it is clear the algorithm that was previously available in the literature, FSLAM-UKF, is vastly outperformed by the algorithm we propose, MSLAM-TUKF because of two reasons: the use of an efficient particle filter for multitarget tracking and a more accurate approximation of the posterior of the map given by the TUKF.

The RMS position error for the targets when only target number two is present in the scenario of Fig. 4, is shown in Figure 7. In this case, the best algorithm is FSLAM-TUKF. MSLAM that was designed to deal with a multitarget set-up is worse than FSLAM for one target. The reason for this is that we cannot take advantage of subparticle crossover as there is only one target and, therefore, it cannot be performed. We recall that the good properties of our method that are subparticle crossover and a map mixing step occur at the expense of approximating Gaussian mixtures by independent Gaussian, equation (39). However, for a one-target scenario it is better to estimate the whole target trajectory and potentially incur map degeneracy than to estimate the current state of the target without map degeneracy and make the approximations in Section III.

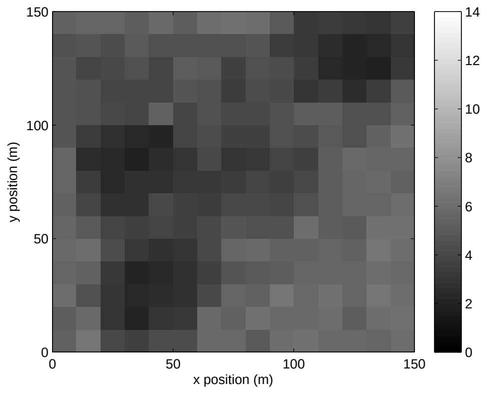
It is interesting to compare the results of Fig. 7 for one target with the results of Fig. 5 for three targets. This provides an insight into how the PP method and MSLAM deal with the problem of dimensionality. While FSLAM severely suffers the curse of dimensionality, when there are three targets the performance slumps, PP and MSLAM have roughly the same performance regardless of the number of targets because of subparticle crossover. Moreover, when $\sigma_s \geq 4$, the performance of MSLAM for three targets is better than for one target. This is because MSLAM is able to localise more sensors improving tracking performance. Then, this suggests that MSLAM-TUKF is an appropriate way to address Multitarget SLAM. Finally, the execution times for the algorithms implemented in Matlab with C-MEX subroutines on a Pentium IV with a 2.4-GHz processor for this scenario using 300 particles are shown in Table VI.



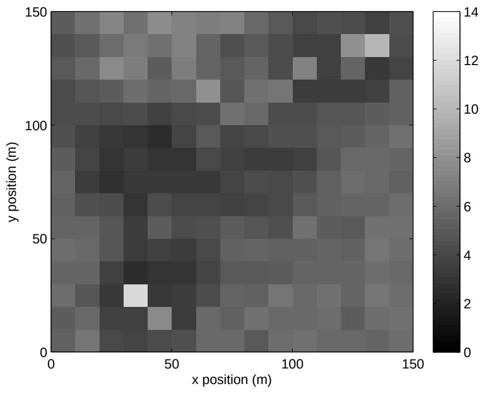
(a) MSLAM-TUKF (300 particles)



(b) MSLAM-UKF (300 particles)



(c) FSLAM-TUKF (600 particles)



(d) FSLAM-UKF (600 particles)

Figure 6 – Final RMS position error map for the sensors for $\sigma_s = 4\,\mathrm{m}$. MSLAM combined with the TUKF provides the best estimate of the map.

Table VI – Execution times in seconds for $\sigma_s = 4\,\mathrm{m}$.

Algorithm (N_{par})	One target	Three targets
MSLAM-UKF (300)	38	153
MSLAM-TUKF (300)	39	184
FSLAM-UKF (600)	43	160
FSLAM-TUKF (600)	44	213
FSLAM-UKF (300)	21	79
FSLAM-TUKF (300)	22	106
PP (300)	14	99

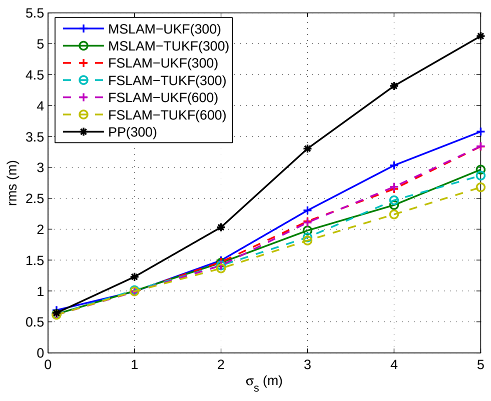


Figure 7 – RMS position error for the targets with one target (target two in Fig. 4). FSLAM-TUKF provides a slightly better estimate than MSLAM-TUKF.

VII. Conclusions

We have developed a new algorithm for Multitarget Simultaneous Localisation and Mapping of a sensor network that vastly outperforms the well-known FSLAM in multitarget scenarios. The are two reasons for this improvement. Firstly, we use an efficient particle filter for multiple target tracking that allows subparticle crossover. Secondly, the TUKF provides a more accurate approximation of the posterior of the map than the UKF.

It is shown that a big improvement in the tracking of multiple targets can be achieved by simultaneously estimating the multitarget state and the sensors' positions compared to only estimating the multitarget state. This was expected as we use the measurements to update the pdf of all the variables of the model rather than just the multitarget state.

The lines of future research are manifold. Firstly, calculating the posterior Cramér-Rao lower bound would be interesting to see how far this algorithm is from this bound. Secondly, although superior results were obtained for a multitarget situation, our algorithm was slightly outperformed by FSLAM in a single target scenario. Presumably this is because posterior correlations in the sensors' positions are removed to save computational expense. This suggests that accounting for sensors' positions correlations will create a better estimate of the map. However, how to do it in a computationally efficient manner for MSLAM-TUKF is well-worth exploring. We have assumed that we know the aiming directions of the antennas included in the sensors. In a more realistic scenario, these would be unknown so a generalisation of the algorithm for this case is also a topic of interest. In addition, we have assumed that the number of targets is known. Provided the measurements permit identifiability of all the unknown parameters, including the number of targets, a modification of the algorithm in [28] could be applied.

VIII. Acknowledgements

Ángel F. García-Fernández is supported by an FPU Fellowship from Spanish MEC. This work was supported in part by the Spanish National Research and Development Program under Projects TEC2008-02148 and Comonsens (Consolider-Ingenio 2010, CSD2008-00010). Part of this work was done when Ángel F. García-Fernández was a visiting researcher at The University of Melbourne in 2009.

Appendix I

In this appendix, we provide an important result used to calculate $p\left(\mathbf{V}^{k+1}\left|\mathbf{P}^{k+1},\, \mathbf{z}^{1:k}\right.\right)$ based on $p\left(\mathbf{V}^{k}\left|\mathbf{P}^{k},\, \mathbf{z}^{1:k}\right.\right)$:

$$\begin{aligned} p\left(\mathbf{V}^{k+1}\left|\mathbf{P}^{k+1},\, \mathbf{z}^{1:k}\right.\right) &= \int p\left(\mathbf{V}^{k+1},\, \mathbf{X}^{k}\left|\mathbf{P}^{k+1},\, \mathbf{z}^{1:k}\right.\right) \mathrm{d}\mathbf{X}^{k} \\ &= \int p\left(\mathbf{V}^{k+1}\left|\mathbf{P}^{k+1},\, \mathbf{X}^{k}\right.\right) p\left(\mathbf{X}^{k}\left|\mathbf{P}^{k+1},\, \mathbf{z}^{1:k}\right.\right) \mathrm{d}\mathbf{X}^{k} \\ &= \int \prod_{j=1}^{t} p\left(\mathbf{v}_j^{k+1}\left|\mathbf{p}_j^{k+1},\, \mathbf{x}_j^{k}\right.\right) \frac{p\left(\mathbf{P}^{k+1}\left|\mathbf{X}^{k}\right.\right)}{p\left(\mathbf{P}^{k+1}\left|\mathbf{z}^{1:k}\right.\right)} p\left(\mathbf{X}^{k}\left|\mathbf{z}^{1:k}\right.\right) \mathrm{d}\mathbf{X}^{k} \end{aligned} \tag{56}$$

Substituting (16) into (56):

$$p\left(\mathbf{V}^{k+1}\left|\mathbf{P}^{k+1},\, \mathbf{z}^{1:k}\right.\right) \propto$$

$$\int \prod_{j=1}^{t} p\left(\mathbf{v}_j^{k+1}\left|\mathbf{p}_j^{k+1},\, \mathbf{x}_j^{k}\right.\right) \prod_{j=1}^{t} p\left(\mathbf{p}_j^{k+1}\left|\mathbf{x}_j^{k}\right.\right) \cdot \prod_{j=1}^{t} \sum_{i=1}^{N_{par}} \delta\left(\mathbf{p}_j^{k} - \mathbf{p}_{j,i}^{k}\right) p\left(\mathbf{v}_j^{k}\left|\mathbf{p}_{j,i}^{k},\, \mathbf{z}^{1:k}\right.\right) \mathrm{d}\mathbf{X}^{k} \tag{57}$$

$$= \prod_{j=1}^{t} \sum_{i=1}^{N_{par}} \int p\left(\mathbf{v}_j^{k+1}\left|\mathbf{p}_j^{k+1},\, \mathbf{p}_{j,i}^{k},\, \mathbf{v}_j^{k}\right.\right) \cdot p\left(\mathbf{p}_j^{k+1}\left|\mathbf{p}_{j,i}^{k},\, \mathbf{v}_j^{k}\right.\right) p\left(\mathbf{v}_j^{k}\left|\mathbf{p}_{j,i}^{k},\, \mathbf{z}^{1:k}\right.\right) \mathrm{d}\mathbf{v}_j^{k} \tag{58}$$

$$= \prod_{j=1}^{t} \sum_{i=1}^{N_{par}} \int p\left(\mathbf{p}_j^{k+1},\, \mathbf{v}_j^{k+1}\left|\mathbf{p}_{j,i}^{k},\, \mathbf{v}_j^{k}\right.\right) \cdot p\left(\mathbf{v}_j^{k}\left|\mathbf{p}_{j,i}^{k},\, \mathbf{z}^{1:k}\right.\right) \mathrm{d}\mathbf{v}_j^{k} \tag{59}$$

$$= \prod_{j=1}^{t} \sum_{i=1}^{N_{par}} \mathcal{N}\left(\mathbf{u}_j^{k+1}; \overline{\mathbf{u}}_{j,i}^{k+1}, \boldsymbol{\Xi}_{j,i}^{k+1}\right) \tag{60}$$

where $\mathbf{u}_j^{k+1} = \left[\mathbf{p}_j^{k+1}, \mathbf{v}_j^{k+1}\right]^T$ and $\overline{\mathbf{u}}_{j,i}^{k+1}$ and $\boldsymbol{\Xi}_{j,i}^{k+1}$ are given by (26) and (27), respectively, and we have used (1) and (17). It should be noted that $\mathbf{u}_j^{k+1}$ is the vector $\mathbf{x}_j^{k+1}$ but swapping its components so that it is easier to express the result of integral (59), which corresponds with a prediction step.

Appendix II

In this appendix, we provide a recursive way to calculate $p\left(\mathbf{m}\left|\mathbf{P}^{k+1},\,\mathbf{z}^{1:k}\right.\right)$ based on $p\left(\mathbf{m}\left|\mathbf{P}^{k},\,\mathbf{z}^{1:k}\right.\right)$:

$$\begin{aligned} p\left(\mathbf{m}\left|\mathbf{P}^{k+1},\,\mathbf{z}^{1:k}\right.\right) &= \int p\left(\mathbf{m},\,\mathbf{X}^{k}\left|\mathbf{P}^{k+1},\,\mathbf{z}^{1:k}\right.\right)\mathrm{d}\mathbf{X}^{k} \\ &= \int \frac{p\left(\mathbf{P}^{k+1}\left|\mathbf{m},\,\mathbf{X}^{k},\,\mathbf{z}^{1:k}\right.\right)}{p\left(\mathbf{P}^{k+1}\left|\mathbf{z}^{1:k}\right.\right)} p\left(\mathbf{m},\,\mathbf{X}^{k}\left|\mathbf{z}^{1:k}\right.\right)\mathrm{d}\mathbf{X}^{k} \\ &\propto \int p\left(\mathbf{P}^{k+1}\left|\mathbf{X}^{k}\right.\right) p\left(\mathbf{m},\,\mathbf{X}^{k}\left|\mathbf{z}^{1:k}\right.\right)\mathrm{d}\mathbf{X}^{k} \end{aligned} \tag{61}$$

Substituting (14) and (15) into (61):

$$\begin{aligned} p\left(\mathbf{m}\left|\mathbf{P}^{k+1},\,\mathbf{z}^{1:k}\right.\right) &\propto \int p\left(\mathbf{P}^{k+1}\left|\mathbf{X}^{k}\right.\right)\sum_{i=1}^{N_{par}}\delta\left(\mathbf{P}^{k}-\mathbf{P}_{i}^{k}\right)\cdot \\ &\cdot\prod_{j=1}^{t}\mathcal{N}\left(\mathbf{v}_{j}^{k};\overline{\mathbf{v}}_{j,i}^{k},\boldsymbol{\Sigma}_{j,i}^{k}\right)\prod_{r=1}^{M}\mathcal{N}\left(\mathbf{m}_{r};\overline{\mathbf{m}}_{r,i}^{k},\boldsymbol{\Theta}_{r,i}^{k}\right)\mathrm{d}\mathbf{P}^{k}\mathrm{d}\mathbf{V}^{k} \\ &= \int\sum_{i=1}^{N_{par}} p\left(\mathbf{P}^{k+1}\left|\mathbf{P}_{i}^{k},\,\mathbf{V}^{k}\right.\right)\prod_{j=1}^{t}\mathcal{N}\left(\mathbf{v}_{j}^{k};\overline{\mathbf{v}}_{j,i}^{k},\boldsymbol{\Sigma}_{j,i}^{k}\right)\cdot\prod_{r=1}^{M}\mathcal{N}\left(\mathbf{m}_{r};\overline{\mathbf{m}}_{r,i}^{k},\boldsymbol{\Theta}_{r,i}^{k}\right)\mathrm{d}\mathbf{V}^{k} \end{aligned} \tag{62}$$

Using the fact that the dynamics of the targets are independent, equation (1):

$$p\left(\mathbf{m}\left|\mathbf{P}^{k+1},\,\mathbf{z}^{1:k}\right.\right)\propto\sum_{i=1}^{N_{par}}\prod_{j=1}^{t}\int p\left(\mathbf{p}_{j}^{k+1}\left|\mathbf{p}_{j,i}^{k},\,\mathbf{v}_{j}^{k}\right.\right)\mathcal{N}\left(\mathbf{v}_{j}^{k};\overline{\mathbf{v}}_{j,i}^{k},\boldsymbol{\Sigma}_{j,i}^{k}\right)\mathrm{d}\mathbf{v}_{j}^{k}\prod_{r=1}^{M}\mathcal{N}\left(\mathbf{m}_{r};\overline{\mathbf{m}}_{r,i}^{k},\boldsymbol{\Theta}_{r,i}^{k}\right) \tag{63}$$

Substituting (19) into (63) gives

$$\begin{aligned} p\left(\mathbf{m}\left|\mathbf{P}^{k+1},\,\mathbf{z}^{1:k}\right.\right) &\propto \sum_{i=1}^{N_{par}}\prod_{j=1}^{t}\mathcal{N}\left(\mathbf{p}_{j}^{k+1};\overline{\mathbf{p}}_{j,i}^{k+1|k},\boldsymbol{\Xi}_{11,j,i}^{k+1}\right)\cdot\prod_{r=1}^{M}\mathcal{N}\left(\mathbf{m}_{r};\overline{\mathbf{m}}_{r,i}^{k},\boldsymbol{\Theta}_{r,i}^{k}\right) \\ &= \sum_{i=1}^{N_{par}}\tilde{\rho}_{2,i}\left(\mathbf{P}^{k+1}\right)\prod_{r=1}^{M}\mathcal{N}\left(\mathbf{m}_{r};\overline{\mathbf{m}}_{r,i}^{k},\boldsymbol{\Theta}_{r,i}^{k}\right) \end{aligned} \tag{64}$$

where $\tilde{\rho}_{2,i}\left(\mathbf{P}^{k+1}\right)$ is given by (38). Then, we can write (64) as:

$$p\left(\mathbf{m}\left|\mathbf{P}^{k+1},\,\mathbf{z}^{1:k}\right.\right)=\sum_{i=1}^{N_{par}}\rho_{2,i}\left(\mathbf{P}^{k+1}\right)\prod_{r=1}^{M}\mathcal{N}\left(\mathbf{m}_{r};\overline{\mathbf{m}}_{r,i}^{k},\boldsymbol{\Theta}_{r,i}^{k}\right) \tag{65}$$

where $\rho_{2,i}\left(\mathbf{P}^{k+1}\right)$ are the normalised $\tilde{\rho}_{2,i}\left(\mathbf{P}^{k+1}\right)$ so that they sum to one.

References

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, pp. 2292–2330, Aug. 2008.

[2] R. Niu and P. K. Varshney, "Performance analysis of distributed detection in a random sensor field," *IEEE Transactions on Signal Processing*, vol. 56, pp. 339–349, Jan. 2008.

[3] A. F. García-Fernández and J. Grajal, "Asynchronous particle filter for tracking using non-synchronous sensor networks," *Signal Processing*, vol. 91, pp. 2304–2313, October 2011.

[4] N. Kantas, S. S. Singh, and A. Doucet, "Distributed online self-localization and tracking in sensor networks," in *5th International Symposium on Image and Signal Processing and Analysis ISPA*, pp. 498–503, Sept. 2007.

[5] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *The Fifth International Conference on Information Processing in Sensor Networks, IPSN.*, pp. 27–33, 2006.

[6] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 229 –241, Jun. 2001.

[7] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics Automation Magazine*, vol. 13, pp. 99–110, June 2006.

[8] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics Automation Magazine*, vol. 13, pp. 108–117, Sept. 2006.

[9] C. Kim, R. Sakthivel, and W. K. Chung, "Unscented FastSLAM: A robust and efficient solution to the SLAM problem," *IEEE Transactions on Robotics*, vol. 24, pp. 808–820, Aug. 2008.

[10] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, 2002.

[11] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the International Conference on Artificial Intelligence (IJCAI)*, pp. 1151–1156, 2003.

[12] J. A. Castellanos, J. D. Tardos, and G. Schmidt, "Building a global map of the environment of a mobile robot: the importance of correlations," in *Proceedings IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1053–1059, Apr. 1997.

[13] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications.* Artech House, 2004.

[14] C. Andrieu and A. Doucet, "Particle filtering for partially observed Gaussian state space models," *Journal of the Royal Statistical Society: Series B*, vol. 64, pp. 827–836, 2002.

[15] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, pp. 1181–1203, December 2006.

[16] J. Mullane, B.-N. Vo, M. D. Adams, and B.-T. Vo, "A random-finite-set approach to Bayesian SLAM," *IEEE Transactions on Robotics*, vol. 27, pp. 268–282, April 2011.

[17] J. Mullane, B.-N. Vo, M. Adams, and W. Wijesoma, "A random set formulation for Bayesian SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1043–1049, Sept. 2008.

[18] J. Mullane, B.-N. Vo, and M. Adams, “Rao-Blackwellised PHD SLAM,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5410–5416, May 2010.

[19] B.-N. Vo and W.-K. Ma, “The Gaussian mixture probability hypothesis density filter,” *IEEE Transactions on Signal Processing*, vol. 54, pp. 4091–4104, Nov. 2006.

[20] C. Lundquist, L. Hammarstrand, and F. Gustafsson, “Road intensity based mapping using radar measurements with a probability hypothesis density filter,” *IEEE Transactions on Signal Processing*, vol. 59, pp. 1397–1408, April 2011.

[21] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[22] F. Daum and J. Huang, “Curse of dimensionality and particle filters,” in *Proceedings IEEE Aerospace Conference*, vol. 4, pp. 1979–1993, 8-15 2003.

[23] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, “Distributed multirobot exploration and mapping,” *Proceedings of the IEEE*, vol. 94, pp. 1325–1339, July 2006.

[24] T. Bailey, J. Nieto, and E. Nebot, “Consistency of the FastSLAM algorithm,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA*, pp. 424 –429, May 2006.

[25] M. Orton and W. Fitzgerald, “A Bayesian approach to tracking multiple targets using sensor arrays and particle filters,” *IEEE Transactions on Signal Processing*, vol. 50, pp. 216–223, Feb. 2002.

[26] C. Kreucher, K. Kastella, and A. O. Hero III, “Multitarget tracking using the joint multitarget probability density,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 1396–1414, Oct. 2005.

[27] M. R. Morelande, C. M. Kreucher, and K. Kastella, “A Bayesian approach to multiple target detection and tracking,” *IEEE Transactions on Signal Processing*, vol. 55, pp. 1589–1604, May. 2007.

[28] A. F. García-Fernández, J. Grajal, and M. R. Morelande, “Two-layer particle filter for multiple target detection and tracking,” *submitted to IEEE Transactions on Aerospace and Electronic Systems*, 2011.

[29] A. F. García-Fernández and J. Grajal, “Multitarget tracking using the joint multitrack probability density,” in *12th International Conference on Information Fusion*, pp. 595–602, July 2009.

[30] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[31] A. F. García-Fernández, M. R. Morelande, and J. Grajal, “Truncated unscented Kalman filtering,” *submitted to IEEE Transactions on Signal Processing*, 2010.

[32] A. F. García-Fernández, M. R. Morelande, and J. Grajal, “Nonlinear filtering update phase via the single point truncated unscented Kalman filter,” in *14th International Conference on Information Fusion*, 2011.

[33] Y. Bar-Shalom, T. Kirubarajan, and X. R. Li, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2002.

[34] C.-L. Yang, S. Bagchi, and W. Chappell, “Topology insensitive location determination using independent estimates through semi-directional antennas,” *IEEE Transactions on Antennas and Propagation*, vol. 54, pp. 3458–3472, Nov. 2006.

[35] N. Nikolaidis and I. Pitas, “Nonlinear processing and analysis of angular signals,” *IEEE Transactions on Signal Processing*, vol. 46, pp. 3181–3194, Dec. 1998.

[36] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, Feb. 2002.

[37] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with Rao-Blackwellized particle filters,” *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, Feb. 2007.

[38] N. Kwak, G.-W. Kim, and B.-H. Lee, “A new compensation technique based on analysis of resampling process in FastSlam,” *Robotica*, vol. 26, pp. 205–217, March 2008.

[39] R. Martinez-Cantin, N. de Freitas, and J. A. Castellanos, “Analysis of particle methods for simultaneous robot localization and mapping and a new algorithm: Marginal-SLAM,” in *IEEE International Conference on Robotics and Automation*, pp. 2415–2420, April 2007.

[40] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, “Sonar tracking of multiple targets using joint probabilistic data association,” *IEEE Journal of Oceanic Engineering*, vol. 8, pp. 173 –184, Jul. 1983.

[41] M. K. Pitt and N. Shephard, “Filtering via simulation: Auxiliary particle filters,” *Journal of the American Statistical Association*, vol. 94, pp. 590–599, Jun. 1999.

[42] S. J. Julier and J. K. Uhlmann, “Unscented filtering and nonlinear estimation,” *Proceedings of the IEEE*, vol. 92, pp. 401–422, Mar. 2004.

[43] M. Briers, *Improved Monte Carlo Methods for State-Space Models*. PhD thesis, University of Cambridge, 2007.

[44] I. Arasaratnam, S. Haykin, and T. Hurd, “Cubature Kalman filtering for continuous-discrete systems: Theory and simulations,” *IEEE Transactions on Signal Processing*, vol. 58, pp. 4977–4993, Oct. 2010.

[45] R. Kelley, *Iterative Methods for Optimization*. SIAM, 1999.

[46] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.

Ángel F. García-Fernández received his degree in Telecommunication Engineering (with honours) from Universidad Politécnica de Madrid, Spain, in 2007. Since 2007 he has been working towards the Ph.D. degree at the Department of Signals, Systems and Radiocommunications of the same university. His research activities and interests are in the area of target tracking, wireless sensor networks, and radar signal processing.

Mark Morelande received the B.Eng. degree in aerospace avionics from Queensland University of Technology, Brisbane, Australia in 1997 and the Ph.D. in electrical engineering from Curtin University of Technology, Perth, Australia in 2001. In 2001 he was a Postdoctoral Fellow at the Centre for Eye Research, Queensland University of Technology. From 2002-2005 he was a Research Fellow at the Cooperative Research Centre for Sensor, Signal and Information Processing, University of Melbourne. He is now a Senior Research Fellow in the Melbourne Systems Laboratory, also at The University of Melbourne. His research interests include non-stationary signal analysis and target tracking with particular emphasis on multiple target tracking and sequential Monte Carlo methods.

Jesús Grajal was born in Toral de los Guzmanes (León), Spain, in 1967. He received the Ingeniero de Telecomunicación and the Ph.D. degrees from the Technical University of Madrid, Madrid, Spain in 1992 and 1998, respectively. Since 2001 he has been an Associate Professor at the Signals, Systems, and Radiocommunications Department of the Technical School of Telecommunication Engineering of the same University. His research activities are in the area of hardware-design for radar systems, radar signal processing and broadband digital receivers for radar and spectrum surveillance applications.