RBMCDAbox - Matlab Toolbox of Rao-Blackwellized Data Association Particle Filters

Jouni Hartikainen and Simo Särkkä
Department of Biomedical Engineering and Computational Science,
Helsinki University of Technology,
P.O.Box 9203, FIN-02015 TKK, Espoo, Finland
jmjharti@lce.hut.fi, simo.sarkka@hut.fi

February 26, 2008

Version 1.0

Abstract

In this paper we present a documentation for Matlab toolbox consisting of Rao-Blackwellized particle filtering based algorithms, which can be used in solving data association problems frequently occuring in the context of multiple target tracking. The provided algorithms can be used for fixed as well as unknown and time-varying number of targets. The mathematical background of the provided algorithms as well as the relevant estimation methods are briefly reviewed. The properties and the usage of the provided methods are illustrated with several tracking examples.

Contents

1 Introduction

In multiple target tracking (MTT) we are estimating the states of several targets trough measurements, which are typically generated by some kind of radars. If we know the targets which produce each measurement the problem reduces to single target tracking and we can use a standard filtering algorithm (e.g. Kalman or extended Kalman filter) for estimating the states of the targets independently. Unfortunately, such knowledge is very rarely available in practice, and in many cases some measurements might be also due to clutter, so one is forced to solve the problem of data association. The problem is further complicated if we do not know the exact number of targets and if that number varies trough time. After we have estimated the number of targets and associated measurements to individual targets we can apply the standard filtering techniques for estimating the target states.

Methods for solving the data association problems are typically divided into two classes: *Unique-neighbor data association* (such as *multiple hypothesis tracking* (MHT)) and *All-neighbors data association* (such as *joint probabilistic data association* (JPDA)) methods. The main difference between these two are that in the former each measurement is associated with one of the previously created tracks, and in the latter all measurements are used for updating all the tracks.

In MHT [3, 4, 5] each measurement is associated with an existing track, or a new track is formed according to some criterion. Instead of only one track configuration several hypotheses are simultanously formed and maintained as the data associations are not necessarily unique. Likelihoods of the measurements and the posterior probabilities of the hypotheses are calculated, and according to these only the most probable hypotheses are stored. Several variants of this algorithm can be obtained by applying various heuristic techniques, such as gating, hypothesis merging and clustering.

In this paper we provide a documentation for a Matlab toolbox, which consists of algorithms aimed for solving data association problems, in which the number of targets can be *known* or *unknown and time-varying*. In the provided algorithm we treat the target states, data associations and the births and deaths of targets as hidden stochastic processes, which are observed through (possibly noisy and indirect) measurements. The joint tracking and data association is formulated as a Bayesian estimation problem and the inference is done with *Sequential Monte Carlo* (SMC) methods (also referred as *particle filtering* methods), which give Monte Carlo approximations to the posterior distributions, that is, they are represented as a discrete set of samples. The SMC based MTT methods can be considered as generalizations of MHT as particles are used for representing multiple different data association hypothesis. Furthermore, the accuracy and efficiency of the algorithm is enchanced with the application of *Rao-Blackwellization*, which allows us to integrate over the target states and use SMC only for estimating the data associations. This leads to a mixture Gaussian representation of the joint posterior distribution, which has smaller variance than the pure particle representation.

The main purpose of this software is to illustrate the functioning and the properties of the implemented MTT methods rather than being highly optimized software, which can be directly incorporated into a real tracking platform. Practical implementations of these methods certainly require more engineering effort.

The software of this toolbox uses many of the functions provided in our Kalman filtering toolbox EKF/UKF [7], so it's highly recommended to get familiarized with it before getting deeper into the code provided in this toolbox.

This documentation is organized as follows:

- In section 2 we review the filtering methods in which the provided MTT methods relies on.
- Section 3 reviews the provided data association algorithm for the case, in which number of targets is known and remains constant. The software implementation of the algorithm is also reviewed by two demonstrations and function descriptions.
- In the section 4 the RBMCDA algorithm for unknown and time-varying number of targets is reviewed, and the software implementation of the algorithm is also in this case illustrated with two demonstrations and function descriptions.

- In the section 5 all the functions of the toolbox are listed. Also, the descriptions of provided miscallenous functions (which are not directly related to the provided MTT methods) are given in this section.

2 Filtering Methods

In this section we shall briefly review the filtering methods relevant to methods provided by this toolbox.

2.1 Optimal Filtering of Target States

The model for the system (which in this context means individual targets) can be written in (discrete-time) state space form as

$$\mathbf{x}_k \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}) \quad (1)$$
$$\mathbf{y}_k \sim p(\mathbf{y}_k|\mathbf{x}_k) \quad (2)$$

where

- $\mathbf{x}_k \in \Re^n$ is the *state* of the target on time step k, which can be, for example, the position, the velocity or the acceleration of the target.
- $\mathbf{y}_k \in \Re^d$ is the *measurement* on time step k, which can be, for example, position of the target, distance to the target or the relative angle between the target and the sensor.
- $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the *dynamic model* of the targets, which describe the stochastic dynamics of the target.
- $p(\mathbf{y}_k|\mathbf{x}_k)$ is the *measurement model*, which describe how the measurements are distributed given the current state of the target.

The purpose of *optimal filtering* is to compute the estimate for the current state given all the measurements obtained so far, that is, we want to recursively compute the *marginal posterior distribution*

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}), \quad (3)$$

which is given as follows:

- Initialize the recursion with the prior distribution $p(\mathbf{x}_0)$.
- Compute the *predictive distribution* of the state $\mathbf{x}_k$ on time step k by solving the *Chapman-Kolmogorov equation*

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})d\mathbf{x}_{k-1}. \quad (4)$$

- After obtaining the measurement $\mathbf{y}_k$ on time step k compute the posterior distribution of the state $\mathbf{x}_k$ by the Bayes' rule as

$$p(\mathbf{x}_k|\mathbf{y}_k) = \frac{1}{Z_k}p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1}), \quad (5)$$

  where Z_k is a normalization constant given as

$$Z_k = \int p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})d\mathbf{x}_k. \quad (6)$$

These integrals (also commonly referred as *optimal Bayesian filtering equations*) can be solved in analytic form when the dynamic and measurement models are of linear and Gaussian form, i.e,

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathrm{N}(\mathbf{x}_k|\mathbf{A}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}) \quad (7)$$
$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathrm{N}(\mathbf{y}_k|\mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k) \quad (8)$$

and the solution is also of Gaussian form. The algorithm for calculating it is commonly referred as the *Kalman filter* introduced by R.E Kalman in his seminal paper [15]. The solutions for slightly non-linear models, e.g., models of form

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathrm{N}(\mathbf{x}_k|f(\mathbf{x}_{k-1}, k-1), \mathbf{Q}_{k-1}) \quad (9)$$
$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathrm{N}(\mathbf{x}_k|h(\mathbf{x}_{k-1}, k-1), \mathbf{R}_k), \quad (10)$$

can be approximated with Gaussians by using the extended Kalman filter (EKF) [12, 2] or the Unscented Kalman filter (UKF) [13]. The EKF replaces the non-linear model with a *locally linearized* model using the Taylor series expansion whereas UKF determines the mean and covariance of the Gaussian by using the *unscented transformation*. The details of these methods are covered, for example, in the manual of our Kalman filtering toolbox EKF/UKF [7].

The toolbox also contains methods for computing smoothed state estimates of previous time steps given the measurements obtained so far. In other words, we are interested in computing the marginal posterior distribution

$$p(\mathbf{x}_k \,|\, \mathbf{y}_{1:T}), \quad (11)$$

where $T > k$. As with the filtering equations above also in this case the formal solution can expressed as a set of recursive Bayesian *smoothing* equations (see, e.g. [23]):

$$\begin{aligned} p(\mathbf{x}_{k+1} \,|\, \mathbf{y}_{1:k}) &= \int p(\mathbf{x}_{k+1} \,|\, \mathbf{x}_k)\, p(\mathbf{x}_k \,|\, \mathbf{y}_{1:k})\, \mathrm{d}\mathbf{x}_k \\ p(\mathbf{x}_k \,|\, \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \,|\, \mathbf{y}_{1:k}) \int \left[\frac{p(\mathbf{x}_{k+1} \,|\, \mathbf{x}_k)\, p(\mathbf{x}_{k+1} \,|\, \mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1} \,|\, \mathbf{y}_{1:k})} \right] \mathrm{d}\mathbf{x}_{k+1}. \end{aligned} \quad (12)$$

Naturally, these integrals have analytic solutions only in certain cases. With linear Gaussian models the algorithm for calculating the smoothed estimates is commonly referred as *Kalman smoother* (see, e.g., [9, 2]).

2.2 Particle Filtering

In case of more complex (i.e. highly non-linear and non-Gaussian) models Kalman filter based methods are often inadequate for providing sufficiently accurate estimates of the system, and therefore other methods should be considered. *Particle filtering* is a very effective estimation framework, which can be used in estimating arbitrary state space models of form (2). It is based on representing the distributions of interest (i.e. filtering distribution (3)) as a discrete set of samples (*particles*). In principle, any system can be estimated to a desired accuracy with a particle filter by using sufficiently large amount of particles, but of course the practical limitations in computational resources sets the lower bound for it.

2.2.1 Sequential Importance Resampling Filter

Generally it is impossible to directly draw samples from the filtering distribution (3). One way of avoiding this difficulty is to use a recursive version of importance sampling, in which we draw samples from a chosen *importance distribution* and calculate a weight for each sample according to their posterior probabilities. A commonly used algorithm based on this idea is called the *sequential importance resampling* (SIR) [10, 16, 6, 19], in which an additional *resampling* procedure is introduced for avoiding the degeneracy of the algorithm by removing particles with too small weights and duplicating particles with large weights. Resampling introduces some variance to the estimates, but it can reduced by choosing the actual resampling method appropriately. *Stratified resampling* [16] is optimal in terms of variance.

Sequential importance resampling

Given an importance distribution $\pi(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k})$ and a set of particles $\{w_{k-1}^{(i)}, \mathbf{x}_{k-1}^{(i)} : i = 1, \ldots, N\}$ from the previous time step, a set of particles $\{w_k^{(i)}, \mathbf{x}_k^{(i)} : i = 1, \ldots, N\}$ is obtained as follows:

1. Draw $\mathbf{x}_k^{(i)}$ for each particle from the importance distribution:

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}), \quad i = 1, \ldots, N. \tag{13}$$

2. Calculate new (unnormalized) weights as

$$w_k^{*(i)} \propto w_{k-1}^{*(i)} \frac{p(\mathbf{y}_k|\mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k})}, \quad i = 1, \ldots, N. \tag{14}$$

3. Normalize the weights to sum to unity as

$$w_k^{(i)} = \frac{w_k^{*(i)}}{\sum_{j=1}^{N} w_k^{*(i)}}. \tag{15}$$

4. Estimate the effective number of particles as

$$n_{\text{eff}} \approx \frac{1}{\sum_{i=1}^{N} \left(w_k^{(i)}\right)^2}. \tag{16}$$

   If the effective number of particles is too low (i.e. $n_{\text{eff}} < N/10$), perform resampling. This resampling scheme is called the *adaptive resampling*, which monitors the effective number of particles by estimating the variance of the particle weights.

After the particles have been obtained the filtering distribution (3) can be approximated as

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}), \tag{17}$$

where $\delta(\cdot)$ is the Dirac delta function.

The choice of importance distribution is usually critical. Firstly, it should be of functional from which it is easy to draw samples and from which it is possible to evaluate the probability densities of the samples. Secondly, it should be as close as possible to *the optimal importance distribution* [6, 19]

$$\pi(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_{1:k}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_{1:k}), \tag{18}$$

for which the variance is minimal. If it is not possible to directly use this one can obtain good importance distributions by *local linearization* where a mixture of extended Kalman filters (EKF) or unscented Kalman filters (UKF) is used as the importance distribution.

A simple variation of SIR is obtained by using the dynamic model $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ as the importance distribution. This is called the *bootstrap filter* [10], which is really easy to implement, but can also need a very large number of particles to be accurate enough, as it doesn't use the information provided by measurements when sampling from the importance distribution.

2.2.2 Rao-Blackwellized Particle Filter

If the model is of certain form the efficiency of SIR can be enhanced by using the theorem of Rao-Blackwell. The theorem states that if $g(\mathbf{X})$ is any kind of estimator of some parameter θ and $T(\mathbf{X})$ some sufficient statistic, then the conditional expectation of $g(\mathbf{X})$ given $T(\mathbf{X})$ is usually better estimate of θ and never worse. This kind of estimator transformation is often referred as *Rao-Blackwellization*. *The Rao-Blackwellized particle filter* (RBPF) [1, 6, 19] is based on the idea that sometimes it is possible to compute some of the filtering equations in closed form and the other with Monte Carlo sampling instead of using sampling methods for all equations. According to Rao-Blackwell theorem this leads to estimators with less variance, which can be interpret as replacing a finite set of particles with a infinite set, which is always more accurate than any finite set.

In this context the Rao-Blackwellized particle filtering refers to marginalized filtering of Markov models, which are conditionally Gaussian of form

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{x}_{k-1}, \lambda_{k-1}) &= \mathrm{N}(\mathbf{x}_k|\mathbf{A}_{k-1}(\lambda_{k-1})\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}(\lambda_{k-1})) && (19)\\ p(\mathbf{y}_k|\mathbf{x}_k, \lambda_k) &= \mathrm{N}(\mathbf{y}_k|\mathbf{H}_k(\lambda_k)\mathbf{x}_k, \mathbf{R}_k(\lambda_k)) && (20)\\ p(\lambda_k|\lambda_{k-1}) &= (\text{any given form}), && (21) \end{aligned}$$

where $\mathbf{x}_k$ is the state, $\mathbf{y}_k$ the measurement, and λ_k an arbitrary latent variable. $\mathbf{M}(\lambda)$ denotes that matrix $\mathbf{M}$ is conditioned on the value of latent variable λ. If the prior distribution of $\mathbf{x}_k$ is also Gaussian the state variables $\mathbf{x}_k$ can be integrated out in closed form and only the latent variables λ_k need to be sampled. The resulting algorithm is summarized below.

Conditionally Gaussian Rao-Blackwellized particle filter

Given an importance distribution $\pi(\lambda_k|\lambda_{1:k-1}^{(i)}, \mathbf{y}_{1:k})$, set of particles $\{w_{k-1}^{(i)}, \lambda_{k-1}^{(i)}, \mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)} \;:\; i = 1, \ldots, N\}$, and measurement $\mathbf{y}_k$, the set of particles $\{w_k^{(i)}, \lambda_k^{(i)}, \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)} \;:\; i = 1, \ldots, N\}$ is processed as follows [6]:

1. Perform Kalman filter predictions for the means $\mathbf{m}_{k-1}^{(i)}$ and the covariances $\mathbf{P}_{k-1}^{(i)}$ of particles $i = 1, \ldots, N$ conditional on the previously drawn latent variable values $\lambda_{k-1}^{(i)}$ as

$$\begin{aligned} \mathbf{m}_k^{-(i)} &= \mathbf{A}_{k-1}(\lambda_{k-1}^{(i)})\, \mathbf{m}_{k-1}^{(i)} \\ \mathbf{P}_k^{-(i)} &= \mathbf{A}_{k-1}(\lambda_{k-1}^{(i)})\, \mathbf{P}_{k-1}^{(i)}\, \mathbf{A}_{k-1}^T(\lambda_{k-1}^{(i)}) + \mathbf{Q}_{k-1}(\lambda_{k-1}^{(i)}). \end{aligned} \tag{22}$$

2. Draw new latent variables $\lambda_k^{(i)}$ for each particle in $i = 1, \ldots, N$ from the corresponding importance distributions

$$\lambda_k^{(i)} \sim \pi(\lambda_k \mid \lambda_{1:k-1}^{(i)}, \mathbf{y}_{1:k}). \tag{23}$$

3. Calculate new (unnormalized) weights as follows:

$$w_k^{*(i)} \propto w_{k-1}^{*(i)} \frac{p(\mathbf{y}_k \mid \lambda_{1:k}^{(i)}, \mathbf{y}_{1:k-1})\; p(\lambda_k^{(i)} \mid \lambda_{k-1}^{(i)})}{\pi(\lambda_k^{(i)} \mid \lambda_{1:k-1}^{(i)}, \mathbf{y}_{1:k})}, \tag{24}$$

where the likelihood term is the marginal measurement likelihood of the Kalman filter

$$\begin{aligned} &p(\mathbf{y}_k \mid \lambda_{1:k}^{(i)}, \mathbf{y}_{1:k-1}) \\ &\quad = \mathrm{N}\left(\mathbf{y}_k \,\middle|\, \mathbf{H}_k(\lambda_k^{(i)})\, \mathbf{m}_k^{-(i)}, \mathbf{H}_k(\lambda_k^{(i)})\, \mathbf{P}_k^{-(i)}\, \mathbf{H}_k^T(\lambda_k^{(i)}) + \mathbf{R}_k(\lambda_k^{(i)})\right). \end{aligned} \tag{25}$$

such that the model parameters in the Kalman filter are conditioned on the drawn latent variable value $\lambda_k^{(i)}$.

4. Normalize the weights to sum to unity as

$$w_k^{(i)} = \frac{w_k^{*(i)}}{\sum_{j=1}^{N} w_k^{*(i)}}. \tag{26}$$

5. Perform Kalman filter updates for each of the particles conditional on the drawn latent variables $\boldsymbol{\theta}_k^{(i)}$

$$\begin{aligned}
\mathbf{v}_k^{(i)} &= \mathbf{y}_k - \mathbf{H}_k(\lambda_k^{(i)})\, \mathbf{m}_k^{-} \\
\mathbf{S}_k^{(i)} &= \mathbf{H}_k(\lambda_k^{(i)})\, \mathbf{P}_k^{-(i)}\, \mathbf{H}_k^T(\lambda_k^{(i)}) + \mathbf{R}_k(\lambda_k^{(i)}) \\
\mathbf{K}_k^{(i)} &= \mathbf{P}_k^{-(i)}\, \mathbf{H}_k^T(\lambda_k^{(i)})\, \mathbf{S}_k^{-1} \\
\mathbf{m}_k^{(i)} &= \mathbf{m}_k^{-(i)} + \mathbf{K}_k^{(i)}\, \mathbf{v}_k^{(i)} \\
\mathbf{P}_k^{(i)} &= \mathbf{P}_k^{-(i)} - \mathbf{K}_k^{(i)}\, \mathbf{S}_k^{(i)}\, [\mathbf{K}_k^{(i)}]^T.
\end{aligned} \tag{27}$$

6. If the effective number of particles (16) is too low, perform resampling.

After the set of particles have been acquired the filtering distribution can approximated as

$$p(\mathbf{x}_k, \lambda_k \,|\, \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)}\, \delta(\lambda_k - \lambda_k^{(i)})\, \mathrm{N}(\mathbf{x}_k \,|\, \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}). \tag{28}$$

3 RBMCDA with Known Number of Targets

Next we review the Rao-Blackwellized Monte Carlo data association (RBMCDA) algorithm for known number of targets originally presented in [21] and later modified a bit in [22]. First we shall introduce the filtering model of RBMCDA and how it's incorporated into the RBPF framework reviewed above. Then we shall

3.1 Filtering Model

The filtering model of the RBMCDA algorithm is the following:

- *The number of targets* T is known and constant.
- *Target state priors* for targets $j = 1, \ldots, T$ can be expressed as a set of weighted importance samples

$$p(\mathbf{x}_{0,j}) = \sum_i w^{(i)}\, \mathrm{N}(\mathbf{x}_{0,j} | \mathbf{m}_{0,j}^{(i)}, \mathbf{P}_{0,j}^{(i)}). \tag{29}$$

- *The dynamic models* of targets are linear Gaussian

$$p(\mathbf{x}_{k,j} | \mathbf{x}_{k-1,j}) = \mathrm{N}(\mathbf{x}_{k,j} | \mathbf{A}_{k-1,j}\mathbf{x}_{k-1,j}, \mathbf{Q}_{k-1,j}), \tag{30}$$

  where the transition matrices $\mathbf{A}_{k-1,j}$ and process noise covariance matrices $\mathbf{Q}_{k-1,j}$ can be different for each target. Targets with non-linear dynamics of form (9) can be modeled by forming a Gaussian approximation by EKF or UKF. The motions of individual targets are a priori independent.

- *The measurement models* of targets are linear Gaussian

$$p(\mathbf{y}_k | \mathbf{x}_{k,j}, c_k = j) = \mathrm{N}(\mathbf{y}_k | \mathbf{H}_{k,j}\mathbf{x}_{k,j}, \mathbf{R}_{k,j}), \tag{31}$$

where c_k is the data association indicator, which has value $c_k = 0$ for clutter and $c_k = j$ for targets $j = 1, \ldots, T$. The measurement model matrices $\mathbf{H}_{k,j}$ and covariance matrices $\mathbf{Q}_{k,j}$ in (31) can be different for each target. Non-linear measurement models can used similarly as non-linear dynamic models, that is, by using EKF or UKF for making Gaussian approximations.

- *Clutter or false alarm measurements* can be modeled by using any probability density

$$p(\mathbf{y}_k|c_k = 0), \tag{32}$$

which is independent of the target states $\mathbf{x}_k = (\mathbf{x}_{k,1} \ \cdots \ \mathbf{x}_{k,T})^T$. For example, it can be a uniform distribution over the measurement space of volume V

$$p(\mathbf{y}_k|c_k = 0) = 1/V. \tag{33}$$

- *The priors of data association indicators* are known and can be modeled as an mth order Markov chain

$$p(c_k|c_{k-1}, \ldots, c_{k-m}). \tag{34}$$

By allowing the data associations priors to depend on previous data associations we can, for example, restrict the amount of associations per target to one on a scan, that is, on a sequence of multiple measurements on single time instance. This is demonstrated, for example, in section 3.6. The associations can also be completely independent ($m = 0$).

In the article [22] it was shown that the RBMCDA algorithm can be incorporated directly into the Rao-Blackwellized Particle Filtering framework reviewed in section 2.2.2 when the latent values λ_k are defined to contain the data association event indicators as

$$\lambda_k = c_k. \tag{35}$$

It was also noted, that conditional on the data associations c_k the targets will remain independent during tracking. Due to this the computations in RBPF are simplified in such a way that individual targets can be processed separately in each particle. That is, we can perform Kalman filter prediction and update steps and measurement likelihood evaluations for each target separately rather than jointly processing state of the system. Furthermore, the KF measurement update is actually performed only to one target in each particle.

3.2 Data Representation

The state of the RBMCDA algorithm described above consists of a set of N particles, where each particle i at time step k contains the following:

$$\{c^{(i)}_{k-m+1:k}, \mathbf{m}^{(i)}_{k,1}, \ldots, \mathbf{m}^{(i)}_{k,j}, \ldots, \mathbf{m}^{(i)}_{k,T}, \mathbf{P}^{(i)}_{k,1}, \ldots, \mathbf{P}^{(i)}_{k,j}, \ldots, \mathbf{P}^{(i)}_{k,T}, w^{(i)}_k\}, \tag{36}$$

where

- $c^{(i)}_{k-m+1:k}$ are the data association indicators of time steps $k-m+1, \ldots, k$ with integer values $0, \ldots, T$, where T is the number of targets. If the data association prior model is an mth order Markov model, then m previous data associations should be stored. If the data association prior is time independent, the data association indicators do not need to be stored at all.

- $\mathbf{m}^{(i)}_{k,j}$,$\mathbf{P}^{(i)}_{k,j}$ are the mean and covariance of the target j, and they are conditional on the data association history $c^{(i)}_{1:k}$.

- $w^{(i)}_k$ is the importance weight of the particle.

In the actual software implementation a single particle is represented as a structure array with the following fields:

`M`	Cell array of size $1 \times T$ of T target means.
`P`	Cell array of size $1 \times T$ of T target covariances.
`W`	Importance weight of the particle.

The data association indicators $c^{(i)}_{k-m+1:k}$ are not actually stored in the particle structures and the indicators of the current time step are returned by the RBMCDA routine (see the description of software implementation in section 3.5) separately. The purpose of this is to give the programmer a freedom of using any data structure he wants when computing the target and clutter priors in each time step.

New particle structures can be initialized with the following function:

`mcda_init`

Initializes a cell array containing N particle struct arrays. Each target can be given a separate prior for each particle, or same prior can be used. The prior particle weights can also be given. If not given, however, uniform distribution is used.

Syntax: `[S] = mcda_init(np,M0,P0,[W])`

Input:	`np`	Number of particles.	
	`M0`	Cell array of target prior means. The size is $T \times N$ if separate is used for each particle, otherwise $T \times 1$.	
	`P0`	Cell array of target prior covariances. The size is $T \times N$ if separate is used for each particle, otherwise $T \times 1$.	
	`W`	Prior particle weights.	default: uniform
Output:	`S`	$1 \times N$ cell array containing N particle structures.	

3.3 Optimal Importance Distribution

To use the RBPF framework we must specify a importance distribution $\pi(c_k \mid \mathbf{y}_{1:k}, c^{(i)}_{1:k-1})$ for the algorithm. Fortunately, we can use the optimal importance distribution, which can be decomposed by the Bayes' rule as

$$\begin{aligned} & p(c_k \mid \mathbf{y}_{1:k}, c^{(i)}_{1:k-1}) \\ & \quad \propto p(\mathbf{y}_k \mid c_k, \mathbf{y}_{1:k-1}, c^{(i)}_{1:k-1}) \\ & \quad \times p(c_k \mid c^{(i)}_{k-m:k-1}), \end{aligned} \tag{37}$$

where we have used the fact that association c_k depends only on the m previous associations $c_{k-m:k-1}$ if the order of the Markov model is m. The marginal measurement likelihood can be computed as [21]

$$\begin{aligned} & p(\mathbf{y}_k \mid c_k, \mathbf{y}_{1:k-1}, c^{(i)}_{1:k-1}) \\ & \quad = \begin{cases} 1/V & \text{if } c_k = 0 \\ \mathrm{KF}_{lh}(\mathbf{y}_k, \mathbf{m}^{-(i)}_{j,k}, \mathbf{P}^{-(i)}_{j,k}, \mathbf{H}_{j,k}, \mathbf{R}_{j,k}) & \text{if } c_k = j \end{cases} \end{aligned} \tag{38}$$

where $j = 1, \ldots, T$ and $\mathrm{KF}_{lh}(\cdot)$ denotes the Kalman filter measurement likelihood evaluation. $\mathbf{H}_{j,k}$ and $\mathbf{R}_{j,k}$ are the measurement model matrix and the measurement covariance matrix of the target j, respectively. For $j = 1, \ldots, T$ we have

$$[\mathbf{m}^{-(i)}_{j,k}, \mathbf{P}^{-(i)}_{j,k}] = \mathrm{KF}_p(\mathbf{m}^{(i)}_{j,k-1}, \mathbf{P}^{(i)}_{j,k-1}, \mathbf{A}_{j,k-1}, \mathbf{Q}_{j,k-1}), \tag{39}$$

where $\mathrm{KF}_p(\cdot)$ denotes the Kalman filter prediction step, and $\mathbf{m}^{(i)}_{j,k-1}, \mathbf{P}^{(i)}_{j,k-1}$ are the mean and the covariance of target j in particle i, which is conditioned on the state history $c^{(i)}_{1:k-1}$. $\mathbf{A}_{j,k-1}$ and

$\mathbf{Q}_{j,k-1}$ are the transition matrix of dynamic model and the process noise covariance matrix of the target j, respectively.

We can acquire samples from the optimal importance distribution as follows:

1. Compute the unnormalized clutter association probability

$$\begin{aligned}\hat{\pi}_0^{(i)} &= p(\mathbf{y}_k \mid c_k^{(i)} = 0, \mathbf{y}_{1:k-1}, c_{1:k-1}^{(i)}) \\ &\quad \times p(c_k^{(i)} = 0 \mid c_{k-m:k-1}^{(i)}).\end{aligned} \tag{40}$$

2. Compute the unnormalized target association probabilities for each target $j = 1, \ldots, T$

$$\begin{aligned}\hat{\pi}_j^{(i)} &= p(\mathbf{y}_k \mid c_k^{(i)} = j, \mathbf{y}_{1:k-1}, c_{1:k-1}^{(i)}) \\ &\quad \times p(c_k^{(i)} = j \mid c_{k-m:k-1}^{(i)}).\end{aligned} \tag{41}$$

3. Normalize the importance distribution:

$$\pi_j^{(i)} = \frac{\hat{\pi}_j^{(i)}}{\sum_{j'=0}^{T} \hat{\pi}_{j'}^{(i)}}, \quad j = 0, \ldots, T. \tag{42}$$

4. The probabilites $\hat{\pi}_j^{(i)}, j = 0, \ldots, T$ now represent a discrete probability distribution in each particle, from which the data association indicators $c_k^{(i)}, i = 1, \ldots, N$ can be easily sampled as follows:
   - Draw $c_k^{(i)} = 0$ with probability $\pi_0^{(i)}$
   - Draw $c_k^{(i)} = 1$ with probability $\pi_1^{(i)}$.
   - Draw $c_k^{(i)} = 2$ with probability $\pi_2^{(i)}$.
   - ...
   - Draw $c_k^{(i)} = T$ with probability $\pi_T^{(i)}$.

3.4 Software Implementation

Conceptually the RBMCDA algorithm can be divided into three main parts: prediction, update and resampling. Although these are provided as ready Matlab functions the writing of main program, which ties the different parts of the estimation procedure together, is left for the end user.

The structure of the main program can be roughly summarized as follows:

1. Set up the model parameters and the needed data structures used in the estimation. See the demonstrations or the EKF/UKF toolbox [7] for more details on how this can be done. Data structures for particles can be initialized with the function **`mcda_init`** introduced in section 3.2.

2. Load the measurement data. If real data is not available simulated data can be loaded/generated here.

3. Create the filtering loop, which processes each time step as follows:
   (a) Call the prediction function (**`kf_mcda_predict`**, **`ekf_mcda_predict`** or **`ukf_mcda_predict`**).
   (b) Each measurement on the current time step are processed as follows:
      i. Set up the target and clutter priors for the update function conditional on the previous data associations for each particle. This can be done in any way the user wants. See the demos for some examples.

ii. Call the update function (**kf_mcda_update**, **ekf_mcda_update** or **ukf_mcda_update**) for each measurement on the current time step.

(c) Perform resampling (with function **resample**), if needed. The actual criterion on when to perform resampling is left for user to decide. If adaptive resampling is used the effective number of particles can be estimates with the function **eff_particles**.

(d) Save the estimates of the current time step. The particles in each time step are stored as a $1 \times N_p$ cell array of N_p particle structures, so a natural structure for saving them is a $N_t \times N_p$ sized cell array, where N_t is the number of time steps.

4. Smooth the estimates (with the function **kf_mcda_smooth**, **ekf_mcda_smooth** or **ukf_mcda_smooth**), if needed.

5. Save and plot the overall estimates.

Of course, the procedure is not by any means restricted to have this form. The purpose is only the give an illustration on how it can be used. The descriptions of the functions highlighted above are given below.

kf_mcda_predict

Computes Kalman filter prediction (eq. (22)) for each target in each particle separately.

Syntax: `S = kf_mcda_predict(S,A,Q)`

Input:	`S`	$1 \times N$ cell array containing particle structures.
	`A`	State transition matrix which can be a same numeric matrix for every target or a TxN cell array containing separate matrices for each target in each particle.
	`Q`	Process noise covariance matrix which can be a same numeric matrix for every target or a TxN cell array containing separate matrices for each target in each particle
Output:	`S`	$1 \times N$ cell array containing the struct arrays of predicted particles.

kf_mcda_update

Forms the optimal importance distribution in each particle by calculating the likelihoods of targets and clutter (eqs. (38)) and combining them with prior target detection and clutter probabilities (eqs. (41) and (40), respectively). Then, new data association indicators are drawn according to procedure reviewed in section 3.4. After that the Kalman filter update (eqs. (27)) is done for targets, whose values were drawn from the importance distribution in each particle. Finally, the weights of particles are updated and normalized to sum to unity.

Syntax: `[S,C] = kf_mcda_update(S,Y,H,R,TP,CP,CD)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	Measurement as $D \times 1$ vector.	
	`H`	Measurement model matrix.	
	`R`	Measurement noise covariance as $D \times D$ matrix.	
	`TP`	$T \times 1$ vector of prior probabilities for measurements hitting each of the targets. Can also be a $T \times N$ matrix containing separate priors for every particle.	
	`CP`	Prior probability of a measurement being due to clutter. Can be a scalar (same prior for all particles) or a 1xN vector (different for each particle).	default: zero
	`CD`	Probability density of clutter measurements, which could be for example $1/V$, where V is the volume of clutter measurement space.	default: 0.01
Output:	`S`	$1 \times N$ cell array containing the struct arrays of updated particles.	
	`C`	$1 \times N$ vector of contact incidences $0, \ldots, T$, where 0 means clutter.	

kf_mcda_smooth

Smooths the filtered particles produced by the RBMCDA algorithm
with RTS smoother.

Syntax: `[S,SM] = kf_mcda_smooth(S,A,Q)`

Input:	`S`	$N_t \times N_p$ cell array containing N_p particle structures for N_t time steps.
	`A`	State transition matrix which can be a same numeric matrix for every target or a $T \times N_p$ cell array containing separate matrices for each target in each particle.
	`Q`	Process noise covariance matrix which can be a same numeric matrix for every target or a $T \times N_p$ cell array containing separate matrices for each target in each particle.
Output:	`S`	$N_t \times N_p$ cell array containing the struct arrays of predicted particles.
	`SM`	$1 \times T$ cell array containing smoothed means for each target as a DxN_t matrix.

ekf_mcda_predict

Computes EKF prediction (eq. (22)) for each target in each particle separately.
If the dynamic models are non-linear EKFs are used instead of standard KFs.

Syntax: `S = ekf_mcda_predict(S,A,Q,a,AW,p)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`A`	Derivative of dynamic model w.r.t. to state as matrix if common for all targets or as cell array of size $T \times N$ for each target in each particle. Can also be a inline function or name of function in form `A(x,i,p)`, where `i` is the index of the target `x`. If the model is linear state transition matrices are given instead of Jacobians.	default: eye
	`Q`	Process noise of discrete model as matrix if common for all targets, or as cell array of size $T \times 1$ for all targets separately.	default: zero
	`a`	Dynamic model as cell array of size $T \times N$ for each target in each particle, or inline function or name of function in form `a(x,i,p)`, where `i` is the index of the target `x`.	default: `A*X`
	`AW`	Derivative of `a` w.r.t. noise as matrix if common for all targets or as cell array of size $T \times N$ for each target in each particle. Can also be a inline function or name of function in form `AW(x,i,p)`, where `i` is the index of the target `x`.	default: eye
	`p`	Parameters of `A`, `a` and `AW`.	
Output:	`S`	$1 \times N$ cell array containing the struct arrays of predicted particles.	

ekf_mcda_update

Same as `kf_mcda_update` with expection that standard KF is replaced with EKF.

Syntax: `[S,C] = ekf_mcda_update(S,Y,H,R,h,V,TP,CP,CD,p)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	Measurement as $D \times 1$ vector.	
	`H`	Derivative of h w.r.t. state as matrix, inline function or name of function in form `H(x,p)`. In case of linear measurements the Jacobian is replaced with measurement model matrix.	
	`R`	Measurement noise covariance as $D \times D$ matrix.	
	`h`	Mean prediction (innovation) as vector, inline function or name of function in form `h(x,p)`.	
	`V`	Derivative of `h` w.r.t. noise as matrix, inline function or name of function in form `V(x,i,p)`	default: eye
	`TP`	$T \times 1$ vector of prior probabilities for measurements hitting each of the targets. Can also be a $T \times N$ matrix containing separate priors for every particle.	
	`CP`	Prior probability of a measurement being due to clutter. Can be a scalar (same prior for all particles) or a 1xN vector (different for each particle).	default: zero
	`CD`	Probability density of clutter measurements, which could be for example $1/V$, where V is the volume of clutter measurement space.	default: 0.01
	`p`	Parameters of `H`, `h` and `V`.	
Output:	`S`	$1 \times N$ cell array containing the struct arrays of updated particles.	
	`C`	$1 \times N$ vector of contact incidences $0, \ldots, T$, where 0 means clutter.	

ekf_mcda_smooth

Smooths the filtered particles produced by the RBMCDA algorithm with ERTS smoother.

Syntax: `[S,SM] = ekf_mcda_smooth(S,A,Q,a,AW,p,s_p)`

Input:	`S`	$N_t \times N_t$ cell array containing N_p particle structures for N_t time steps.	
	`A`	Derivative of dynamic model w.r.t. to state as matrix if common for all targets or as cell array of size $T \times N_p$ for each target in each particle. Can also be a inline function or name of function in form `A(x,i,param)`, where `i` is the index of the target `x`. If the model is linear state transition matrices are given instead of Jacobians.	default: eye
	`Q`	Process noise of discrete model as matrix if common for all targets, or as cell array of size $T \times 1$ for all targets separately.	default: zero
	`a`	Dynamic model as cell array of size $T \times N_p$ for each target in each particle, or inline function or name of function in form `a(x,i,p)`, where `i` is the index of the target `x`.	default: `A*X`
	`AW`	Derivative of `a` w.r.t. noise as matrix if common for all targets or as cell array of size $T \times N_p$ for each target in each particle. Can also be a inline function or name of function in form `AW(x,i,p)`, where `i` is the index of the target `x`.	default: eye
	`p`	Parameters of `A`, `a` and `AW`.	
	`s_p`	1 if the same parameters should be used for every time step.	
Output:	`S`	$N_t \times N_p$ cell array containing smoothed particle structures for each time step.	
	`SM`	$1 \times T$ cell array containing smoothed means for each target as a $D \times N_t$ matrix.	

ukf_mcda_predict

Computes UKF prediction (eq. (22)) for each target in each particle separately.

Syntax: `S = ukf_mcda_predict(S,A,Q,a,AW,p)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`a`	Dynamic model as cell array of size $T \times N$ for each target in each particle, or inline function or name of function in form `a(x,i,p)`, where `i` is the index of the target `x`.	default: `A*X`
	`Q`	Process noise of discrete model as matrix if common for all targets, or as cell array of size $T \times N$ for all targets and particles separately.	default: zero
	`p`	Parameters of `a`.	
Output:	`S`	$1 \times N$ cell array of predicted particle structures.	

ukf_mcda_update

Same as `kf_mcda_update` with expection that standard KF is replaced with UKF.

Syntax: `[S,C] = ukf_mcda_update(S,Y,h,R,TP,CP,CD,p)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	Measurement as $D \times 1$ vector.	
	`h`	Measurement model function as a inline function, function handle or name of function in form `h(x,p)`.	
	`R`	Measurement noise covariance.	
	`TP`	$T \times 1$ vector of prior probabilities for measurements hitting each of the targets. Can also be a $T \times N$ matrix containing separate priors for every particle.	
	`CP`	Prior probability of a measurement being due to clutter. Can be a scalar (same prior for all particles) or a 1xN vector (different for each particle).	default: zero
	`CD`	Probability density of clutter measurements, which could be for example $1/V$, where V is the volume of clutter measurement space.	default: 0.01
	`p`	Parameters of `h`.	
Output:	`S`	$1 \times N$ cell array containing the struct arrays of updated particles.	
	`C`	$1 \times N$ vector of contact incidences $0, \ldots, T$, where 0 means clutter.	

ukf_mcda_smooth

Smooths the filtered particles produced by the RBMCDA algorithm with URTS smoother.

Syntax: `[S,SM] = ukf_mcda_smooth(S,a,Q,p,s_p)`

Input:	`S`	$N_t \times N_t$ cell array containing N_p particle structures for N_t time steps.	
	`a`	Dynamic model as cell array of size $T \times N$ for each target in each particle, or inline function or name of function in form `a(x,i,p)`, where `i` is the index of the target `x`.	default: `A*X`
	`Q`	Process noise of discrete model as matrix if common for all targets, or as cell array of size $T \times 1$ for all targets separately.	default: zero
	`p`	Parameters of `a`.	
	`s_p`	1 if the same parameters should be used for every time step.	
Output:	`S`	$N_t \times N_p$ cell array containing smoothed particle structures for each time step.	
	`SM`	$1 \times T$ cell array containing smoothed means for each target as a $D \times N_t$ matrix.	

3.5 Demonstration: Tracking a Single Object with Cluttered Measurements

As a first example we shall consider a scenario, in which we are tracking a single target in two dimensions having cluttered measurements. The dynamic model of the object is described by the

discretized Wiener process velocity model (see, e.g., [2]), where the state of the target can be written as

$$\mathbf{x}_k = \begin{pmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{pmatrix}^T, s \tag{43}$$

where (x_k, y_k) is the object's position and $(\dot{x}_k, \dot{y}_k)$ the velocity in two dimensional cartesian coordinates. The discretized dynamics can be expressed with a linear, time-invariant plant equation

$$\mathbf{x}_k = \underbrace{\begin{pmatrix} 1 & 0 & \Delta\mathrm{t} & 0 \\ 0 & 1 & 0 & \Delta\mathrm{t} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{:=\mathbf{A}} \mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \tag{44}$$

where $\mathbf{q}_{k-1}$ is discrete Gaussian white process noise having moments

$$\mathrm{E}[\mathbf{q}_{k-1}] = \mathbf{0}$$

$$\mathrm{E}[\mathbf{q}_{k-1}\mathbf{q}_{k-1}^T] = \begin{pmatrix} \frac{1}{3}\Delta\mathrm{t}^3 & 0 & \frac{1}{2}\Delta\mathrm{t}^2 & 0 \\ 0 & \frac{1}{3}\Delta\mathrm{t}^3 & 0 & \frac{1}{2}\Delta\mathrm{t}^2 \\ \frac{1}{2}\Delta\mathrm{t}^2 & 0 & \Delta\mathrm{t} & 0 \\ 0 & \frac{1}{2}\Delta\mathrm{t}^2 & 0 & \Delta\mathrm{t} \end{pmatrix} q.$$

The size of time step is set to $\Delta\mathrm{t} = 0.1$, and the power spectral density of process noise to $q = 0.1$.

To model the clutter measurements the data association indicator c_k is now defined to have value 0 if the measurement is clutter, and value 1 if the measurement is from the actual target. The likelihood of clutter measurements is defined to be uniform in space $[-5, 5] \times [-4, 4]$. The measurement model for the actual target is linear with additive Gaussian noise. Thus, we can express the joint measurement likelihood as

$$p(y_k \mid \mathbf{x}_k, c_k) = \begin{cases} 1/80 & , \quad \text{if } c_k = 0 \\ \mathrm{N}(y_k \mid \mathbf{H}\,\mathbf{x}_k, \mathbf{R}) & , \quad \text{if } c_k = 1, \end{cases}$$

where the measurement model matrix is

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{45}$$

and the noise covariance matrix

$$\mathbf{R} = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.05 \end{pmatrix}. \tag{46}$$

The data association priors are completely independent on previous data associations, and can be written as

$$p(c_k) = \begin{cases} 0.5 & , \quad \text{if } c_k = 0 \\ 0.5 & , \quad \text{if } c_k = 1, \end{cases}$$

which means that half of the measurements are from the real target and the other half is clutter.

The system is simulated 240 time steps and the target is given slightly randomized accelerations such that is achieves a curved trajectory, which is plotted in panel a) of figure 1. It also shows the simulated (real and clutter) measurements. The software code for generating the trajectory and measurements can be found in the beginning of file containing the demo's main program (`clutter_demo.m`).

To illustrate the usage of RMBCDA-algorithm we shall now go through the essential parts of the demo's main program. Firstly, we must create the needed data structures for particles. This can be done with the following code segment:

```
% Space for prior estimates
MM1 = cell(1,NT);
```

```
PP1 = cell(1,NT);
% Prior for target 1
MM1{1} = X_r(:,1);
PP1{1} = diag([0.1 0.1 0.1 0.1]);

% Generate the particle structures
S = mcda_init(N1,MM1,PP1);

% Space for RBMCDA estimates
MM = zeros(m,n);
SS = cell(n,N1);
```

In the first two lines (excluding comments and empty lines) we create cell arrays for containing the prior estimates of the targets (that is, for only one target in this case), and in the following two lines the prior estimate for the target 1 is assigned to the cell array. Next we call the particle structure initialization function **`mcda_init`**, for which we pass the prior estimates and the number of particles. In the last two lines we reserve space for the filtered mean estimates and particle structures for each time step.

The actual filtering can be done with following loop:

```
for k=1:size(Y,2)
    % Track with RBMCDA
    S = kf_mcda_predict(S,A,Q);
    [S,C] = kf_mcda_update(S,Y(:,k),H,R,TP,CP,CD);

    M = calculate_mean(S,1);
    MM(:,k) = M;
```

The first line in the loop calls the prediction function and the second the update function of the RBMCDA-algorithm. The variables `A` and `Q` contain the transition and noise covariance matrices of the dynamic model as regular Matlab matrices. Similarly, variables `H` and `R` contains the correspoding model parameters of the measurement model. Variable `Y` contains the measurements as $d \times n$ array, where in this case $d = 2$ and $n = 240$. Variables `TP`, `CP` and `CD` contains the target prior, clutter prior and clutter density values as scalars. The variable `S` contains the particle structures, which cycled between prediction and update functions. The update function also outputs the data assosiation indicators of the current time step in variable `C`, which are now unused as the indicators are independent. For visualization purposes in the last two lines we also calculate and store the mean estimate for the target's state (the actual plotting commands are here omitted and can be found in the original source code).

The essential part of the filtering is the resampling prosedure, which can be done as follows:

```
    n_eff = eff_particles(S);
    if n_eff < N1/4
        ind = resample(S);
        S = S(:,ind);
        % Set weights to be uniform
        W  = ones(1,N1)/N1;
        S = set_weights(S,W);
    end
    % Save the particle structures after resampling
    SS(k,:) = S;
end
```

First we calculate the effective number of particles $n_{\texttt{eff}}$ with the function `eff_particles`. Then the resampling is done, if $n_{\texttt{eff}}$ is smaller than the fourth of number of particles. After the actual

resampling is done (by calling the function `resample`) we set the weight of the particles to be uniform. In the last line we save the current particle structures, which concludes the filtering loop. Possible function calls for animating the target movement can be added at the end of the loop. The full software code of this example can found be in file `clutter_demo.m`.

In panel b) of figure 1 we plotted the filtering results of RBMCDA algorithm using only 10 particles. As the particles represent a mixture of Gaussians distribution (which is hard to visualize directly) the estimates are visualized with samples, which are drawn from the corresponding distributions. The samples can be drawn from Gaussian mixtures with the provided function **`gmm_rnd`**. It can be seen, that the algorithm has no problems following the target trajectory despite the fairly high noise and clutter values.

The smoothing of filtered estimates can be done with only a one line of code:

```
[SM,S_EST1] = kf_mcda_smooth(SS,A,Q);
```

The output variables `SM` and `S_EST1` contains the smoothed particle structured and the smoothed means for each time step, correspondigly.

The smoothing results are displayed in panel c) of figure 1. It can be seen that the estimates are significantly more accurate. The RMSE of filtering estimates are approximately two times greater than the RMSE of smoothed results.

3.6 Demonstration: Bearings Only Tracking of Two Targets

In this demonstration we shall review a scenario frequently arising in the context of passive radar tracking, in which we track multiple targets using bearings-only measurements produced by stationary sensors. Furthermore, the measurements are corrupted by clutter.

The dynamics of the objects are modeled using the discretized Wiener velocity model described in the previous demonstration. The measurement model, however, is now changed to bearings-only model, in which we observe only the angles θ between the targets and the sensors. That is, we can express the measurement from sensor i in the time step k as

$$\theta_k^i = \arctan\left(\frac{y_k - s_y^i}{x_k - s_x^i}\right) + r_k^i, \tag{47}$$

where $(x_{j,k}, y_{j,k})$ is the position of target j, (s_x^i, s_y^i) the position of sensor i and $r_k^i \sim N(0, \sigma^2)$, with $\sigma = 0.02$ radians. As this model is non-linear we replace the Kalman filter with EKF in the RBMCDA-algorithm. The target detection probability is set to $p_d = 80\%$.

The clutter measurements are thought of being generated from an imaginary sensor located at origin, whose measurements are uniformly distributed on range $[-\pi, \pi]$. The number of measurements at a single time instance is Poisson-distributed with rate 5, that is, multiple (varying) number of measurements are produced on each time step. The simulated measurements are plotted in figure 2.

The data-assosiations are defined similarly as previously, that is, $c_k = j$ if the measurement θ_k^i is from target j, and $c_k = 0$ if it's clutter. The priors for data associations are now defined such that for each time step we can get only a one (or zero) detections from each target, and the rest is clutter. Assume now that on time instance t_{scan} we obtain m measurements $\mathbf{y}_k, \ldots, \mathbf{y}_{k+m-1}$. The joint prior for the data associations can be expanded as

$$\begin{aligned} &p(c_{k+m-1}, \ldots, c_k) \\ &= \prod_{j=1}^{m} p(c_{k+j} \mid c_k, \ldots, c_{k+j-1}). \end{aligned} \tag{48}$$

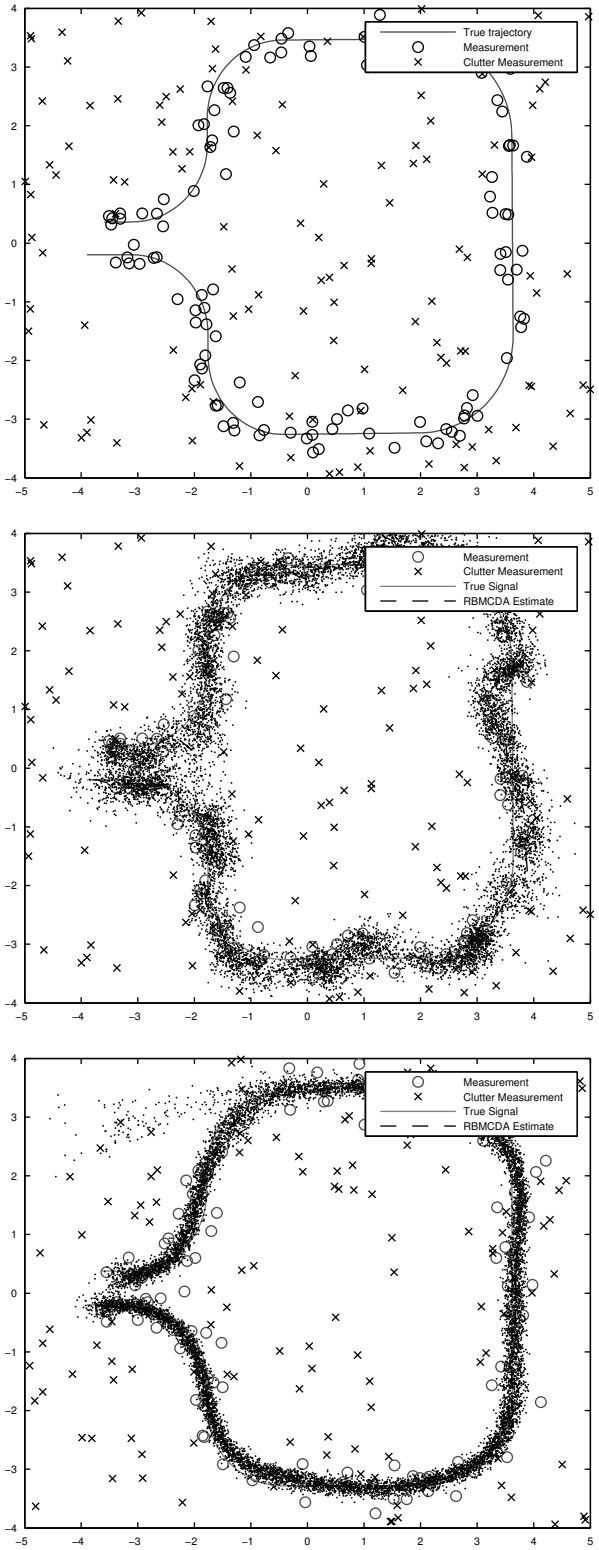

Figure 1: Tracking a Single Object with Cluttered Measurements demonstration: a) Target trajectory and the simulated measurements, b) Filtering results, c) Smoothing results.

Now, by defining *target detection indicators* as

$$\begin{aligned}\delta_1(j) &= \begin{cases} 1, & \text{if there is target 1 detection in } c_{k+j-1}...c_k \\ 0, & \text{otherwise} \end{cases} \\ \delta_2(j) &= \begin{cases} 1, & \text{if there is target 2 detection in } c_{k+j-1}...c_k \\ 0, & \text{otherwise.} \end{cases}\end{aligned} \tag{49}$$

the data association priors for measurements $\mathbf{y}_{k+j}$ can be solved by computing each of the possible target detection indicator combinations:

$$\begin{aligned} &p(c_{k+j} \mid \delta_1(j) = 0 \text{ and } \delta_2(j) = 0) \\ &p(c_{k+j} \mid \delta_1(j) = 0 \text{ and } \delta_2(j) = 1) \\ &p(c_{k+j} \mid \delta_1(j) = 1 \text{ and } \delta_2(j) = 0) \\ &p(c_{k+j} \mid \delta_1(j) = 1 \text{ and } \delta_2(j) = 1). \end{aligned} \tag{50}$$

In this demonstration these probabilities are computed as follows:

$$\begin{aligned} &p(c_{k+j} = 0 \mid \delta_1(j) = 0 \text{ and } \delta_2(j) = 0) = (1-p_d)^2 + p_d^2\frac{r-2}{r} + 2p_d(1-p_d)\frac{r-1}{r} \\ &p(c_{k+j} = 1 \mid \delta_1(j) = 0 \text{ and } \delta_2(j) = 0) = p_d(1-p_d)/r + p_d^2/r \\ &p(c_{k+j} = 2 \mid \delta_1(j) = 0 \text{ and } \delta_2(j) = 0) = p_d(1-p_d)/r + p_d^2/r \\ &p(c_{k+j} = 0 \mid \delta_1(j) = 0 \text{ and } \delta_2(j) = 1) = (1-p_d) + p_d\frac{r-1}{r} \\ &p(c_{k+j} = 1 \mid \delta_1(j) = 0 \text{ and } \delta_2(j) = 1) = p_d/r \\ &p(c_{k+j} = 2 \mid \delta_1(j) = 0 \text{ and } \delta_2(j) = 1) = 0 \\ &p(c_{k+j} = 0 \mid \delta_1(j) = 1 \text{ and } \delta_2(j) = 0) = (1-p_d) + p_d\frac{r-1}{r} \\ &p(c_{k+j} = 1 \mid \delta_1(j) = 1 \text{ and } \delta_2(j) = 0) = 0 \\ &p(c_{k+j} = 2 \mid \delta_1(j) = 1 \text{ and } \delta_2(j) = 0) = p_d/r \\ &p(c_{k+j} = 0 \mid \delta_1(j) = 1 \text{ and } \delta_2(j) = 1) = 1 \\ &p(c_{k+j} = 1 \mid \delta_1(j) = 1 \text{ and } \delta_2(j) = 1) = 0 \\ &p(c_{k+j} = 2 \mid \delta_1(j) = 1 \text{ and } \delta_2(j) = 1) = 0, \end{aligned} \tag{51}$$

where $r = m - j$ is the number of unprocessed measurements left in the sequence. This same idea can be used for constructing similar data association priors for any number of targets.

The initial estimates for the targets are chosen such that all four crossing of measurements from the two sensors contain equal probability mass. This makes the prior distributions of both targets to be bi-modal, as can be seen as shown in figure 3. As was the case in previous example also now we visualize the distributions with samples, which are drawn from the corresponding Gaussian mixtures.

The essential parts of the filtering loop in this case can be written as follows:

```
for k=1:length(Y)
    % Predict all targets
    PS = ekf_mcda_predict(PS,A,Q);

    % Space for data assosiations in the current time step
    % for both targets in each particle
    D1 = zeros(1,NP);
    D2 = zeros(1,NP);

    % Space for clutter and target priors
    CP = zeros(1,NP);
    TP = zeros(2,NP);
```

The measurements are now stored in a cell array Y, whose kth element is matrix containing all the measurements of the time step k. Firstly we call the RBMCDA prediction step function (`ekf_mcda_predict`), which applies the EKF prediction step for each target in each particle. After that we reserve space for data associations in this time step, which are used in calculating the clutter and target priors, for which the space is reserved in the last lines.

Next we form an inner loop, which processes each measurement of the current time step. This can be done as follows:

```
    % Loop over all measurement in the current time step
    for kk=1:length(Y{k})
         % Number of measurements not processed yet
          r = length(Y{k}) - kk + 1;

          % Construct priors for each particle for the current
          % measurement here and store them to variables TP and CP.

          % RBMCDA-update for the current measurement
          [PS,C] = ekf_mcda_update(PS,Y{k}(:,kk),dh_dx_func,R,h_func,...
                                   [],TP,CP,CD,S{k}(:,kk));

          % Store the data associations to arrays D1 and D2
          ind = find(C==1);
          D1(ind) = 1;

          ind = find(C==2);
          D2(ind) = 1;
    end
    % Do resampling, saving and plotting here
end
```

First we calculate how many unprocessed measurements there is left in the current time step and store that result on the variable r. After that we construct the clutter and target priors for each particle according to the equations described above (Matlab code is omitted here and can found in the actual m-file `ekf_bot_tt_demo.m`). Given the constructed priors we call the update step function of the EKF based RBMCDA algorithm (`ekf_mcda_update`). After that we store the data-associations of targets 1 and 2 computed by the update function to arrays `D1` and `D2`, correspondingly. After the inner loop we can apply the resampling and saving commands as was described in the previous example. Possible animation commands can also be added to this section of the loop. The full software code of this example can found in file `bot_tt_demo.m`.

The filtering results are shown in figure 4 by using $N = 100$ particles. The multimodality of the estimates in the beginning of the track can be seen easily. The high uncertainty of position estimates on the line connecting the two sensors can also be seen.

The figure 4 shows the smoothed estimates of the target trajectories. Clearly the smoothing has removed the bimodality, which existed in the track's beginning. Also the position uncertainties in estimates on the line between the sensors has also been resolved by the smoother. In software the smoothing is done exactly the same as in the previous example.

4 RBMCDA with Unknown Number of Targets

In this section we shall review the extension for the RMBCDA allowing to track unknown and time varying number of targets, which was originally presented in [23].

Theoretically we shall assume that there is always a (very large) constant number of targets T_∞, from which only a portion of targets are visible (or alive), which are actually being tracked. The

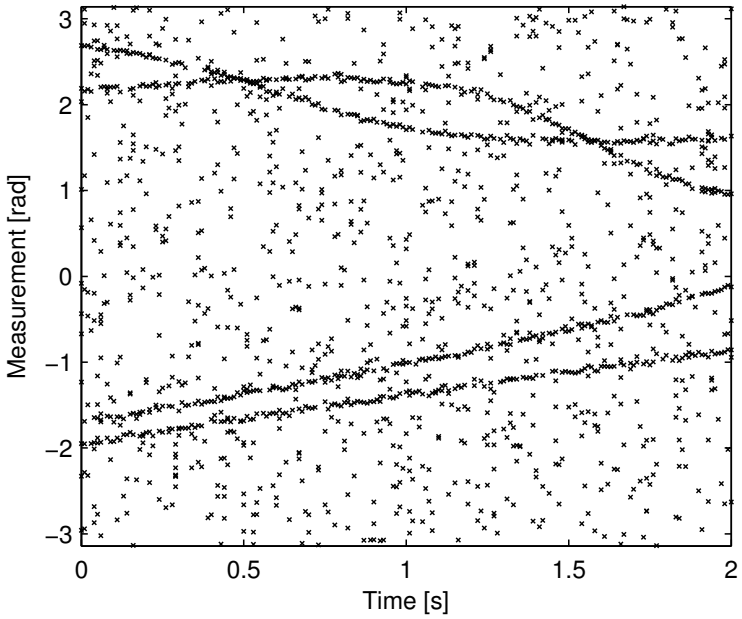

Figure 2: Bearings Only Tracking of Two Targets demonstration: The actual measurements made with two sensors of the two targets. Also the clutter measurements are plotted.

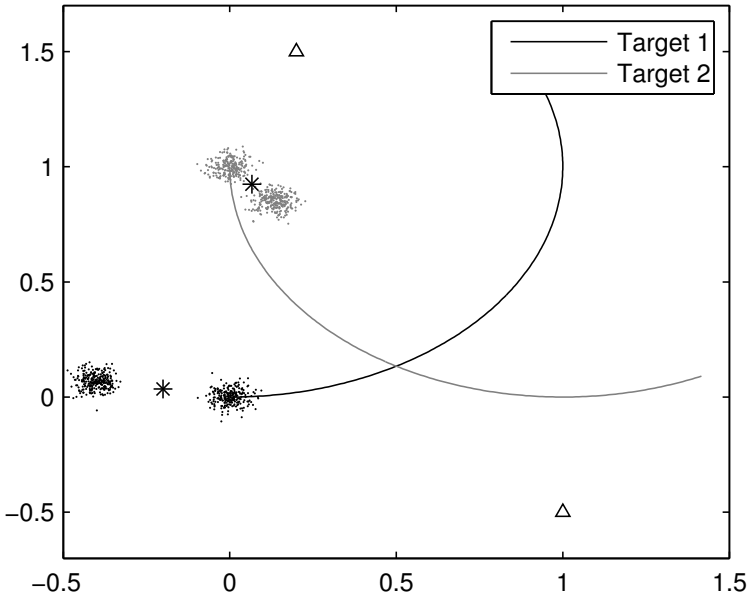

Figure 3: Bearings Only Tracking of Two Targets demonstration: Bimodal prior distributions of the targets and their real trajectories. Also the positions of the sensors are shown as black triangles.

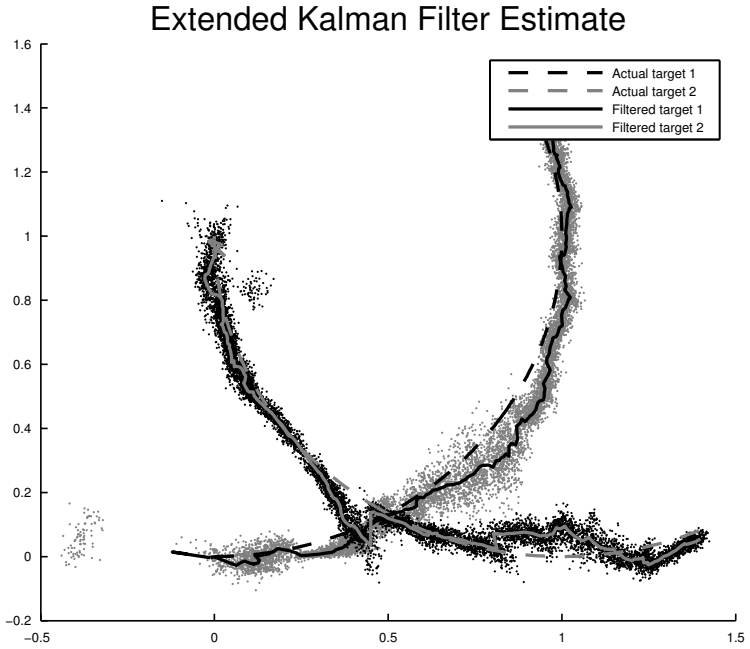

Figure 4: Bearings Only Tracking of Two Targets demonstration: Filtering results of the RBMCDA algorithm. Notice the higher position uncertainty in the beginning of the track and on the line between the two sensors.

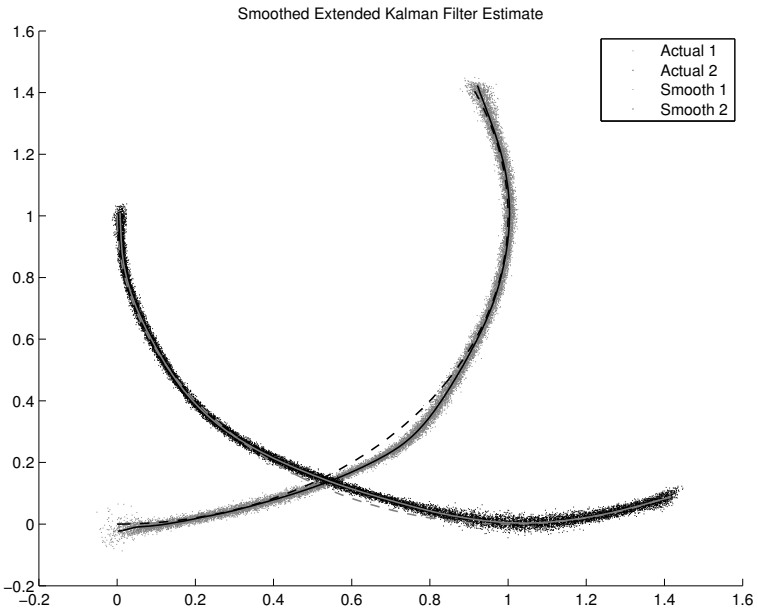

Figure 5: Bearings Only Tracking of Two Targets demonstration: Smoothing results of the RBM-CDA algorithm. The

purpose of this assumption is to guarantee that the joint distribution of the target states is a well defined (Gaussian) probability distribution. The model is constructed in such a way that we do not need to know the actual number of targets T_∞, and that the states of invisible targets do not need to be stored.

The births and deaths of the targets are modeled as stochastic processes in such a way, that the joint filtering model can be directly incorporated into the RBPF framework reviewed in section 2.2.2.

4.1 Filtering Model

1. *The dynamics of targets* and *target measurements* are linear Gaussian as in the RBMCDA model with known and constant number of targets. Non-linear models can be handled with EKF and UKF.

2. *Clutter or false alarm measurements* have the same specification as in the RBMCDA model with known and constant number of targets.

3. The data association priors for targets ($c_k = j$) and clutter ($c_k = 0$), in the case that births and deaths do not occur at the current time step, are known and can be modeled as an mth order Markov chain $p(c_k \mid c_{k-m:k-1}, T_{k-m:k-1})$, where $T_{k-m:k-1}$ contains the number of targets at time steps $k-m, \ldots, k-1$. For example, we might have a uniform prior over targets and clutter:
$$p(c_k \mid c_{k-m:k-1}, T_{k-m:k-1}) = \frac{1}{1+T_{k-1}}. \tag{52}$$

4. Target births may happen only when a measurement is obtained, and in that case a birth happens with probability p_b. For simplicity, the model is defined such that a birth may happen only jointly with an association event, so that if there is no association to a newborn target, there is no birth.

   This is equivalent to stating that the target state prior remains constant until the first measurement is associated, that is, the dynamic model does not affect the target state before the first measurement has been associated to the target. This indicates that it is sufficient to consider the time of the first associated measurement as the actual birth moment.

5. After associating a measurement with a target, the life time t_d (or time to death) of the target has a known probability density
$$t_d \sim p(t_d), \tag{53}$$
   which is in this software package set to be the gamma distribution
$$t_d \sim \text{gamma}(\theta|\alpha,\beta) = \theta^{(\alpha-1)}\frac{\beta^\alpha e^{\frac{-\theta}{\beta}}}{\Gamma(\alpha)}. \tag{54}$$

6. At the time of birth each target has a known Gaussian prior distribution
$$p(\mathbf{x}_{k_0,j}) = \mathrm{N}(\mathbf{x}_{0,j} \mid \mathbf{m}_{0,j}, \mathbf{P}_{k_0,j}). \tag{55}$$

4.2 Probabilities of Birth and Death

If a birth has occurred, it is assumed to be certain that the current measurement is associated to the newborn target:
$$p(c_k \mid \text{birth}) = \begin{cases} 1 & , \quad \text{if } c_k = T_{k-1}+1 \\ 0 & , \quad \text{otherwise.} \end{cases} \tag{56}$$
In the case of no birth, the Markov model for the data associations applies:
$$p(c_k \mid \text{no birth}) = p(c_k \mid c_{k-m:k-1}). \tag{57}$$
The data association and birth events can be divided into the following cases with different probabilities:

1. A target is born and the measurement is associated with the newborn target:

$$\begin{aligned} b_k &= \text{birth} \\ c_k &= T_{k-1} + 1. \end{aligned} \tag{58}$$

2. A target is not born and the measurement is associated with one of the existing targets or with clutter:

$$\begin{aligned} b_k &= \text{no birth} \\ c_k &= j, \qquad j = 0, \ldots, T_k. \end{aligned} \tag{59}$$

3. Other events have zero probability.

Thus, given the associations $c_{k-m:k-1}$ on the m previous steps, the joint distribution of the event $b_k \in \{\text{no birth}, \text{birth}\}$ and the association c_k is given as

$$p(b_k, c_k \mid c_{k-m:k-1}) = \begin{cases} p_b & \text{in case (1)} \\ (1 - p_b)\, p(c_k \mid c_{k-m:k-1}) & \text{in case (2)} \\ 0 & \text{in case (3),} \end{cases} \tag{60}$$

where p_b is the prior probability of birth.

The restriction to one data association per target at single time instance in the case of unknown number of targets can be handled in the same manner as in the case of known number of targets. We simply assume that there is positive probability of detecting a newborn target on each step. That is, in addition to the existing targets we model the possibility detection of a new target which has the detection probability p_b. This probability of detecting a new target is equivalent to the probability of birth, because we have defined the birth to be the event of detecting the target for the first time. However, the difference to the restriction of one association per target is that births may occur as many times as there are measurements on scan, not only once per scan.

The filtering model presented in this section states that after associating a measurement with a target, the life time t_d of the target has the known probability density (53). Thus if the last association with target j was at time $\tau_{k,j}$, and on the previous time step t_{k-1} we sampled a hypothesis that the target is alive, then the probability that the target is dead at current time step t_k is

$$\begin{aligned} &p(\text{death of } j \mid t_k, t_{k-1}, \tau_{k,j}) \\ &\quad = P(t_d \in [t_{k-1} - \tau_{k,j}, t_k - \tau_{k,j}] \mid t_d \geq t_{k-1} - \tau_{k,j}). \end{aligned} \tag{61}$$

4.3 Relationship to RBPF

The RBMCDA algorithm with an unknown number of targets fits to the RBPF framework, if the latent variable $\boldsymbol{\lambda}_k$ contains the visibility indicator $\mathbf{e}_k$ and the data association indicator c_k at the current time step

$$\boldsymbol{\lambda}_k = \{\mathbf{e}_k, c_k\}. \tag{62}$$

The visibility indicators and the data associations implicitly define the number of (visible) targets T_k at each time step.

Given that the targets are a priori unordered, there is a high permutation symmetry in the posterior distributions of the target states, visibility indicators and data association indicators. We can change the indices of any two targets, including the visibility indicators and data associations, and the probability of the configuration will remain the same. For this reason, we shall select one of the permutations arbitrarily and use it for representing all the permutations. This permutation is based on the times of the first associations with the targets. This does not change the model, because this is not a priori ordering, but is merely a way of selecting a compact representation for a very high number of redundant permutations.

1. The joint state $\mathbf{x}_k$ contains the states of the T_∞ targets

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{x}_{k,1} \\ \cdots \\ \mathbf{x}_{k,T_\infty} \end{pmatrix}. \tag{63}$$

2. At the initial time step the targets have Gaussian prior distributions

$$\mathrm{N}(\mathbf{x}_{0,j} \mid \mathbf{m}_0, \mathbf{P}_0). \tag{64}$$

   The model is constructed such that the invisible targets at any time step k (indicated by $\mathbf{e}_k$) do not have a dynamic model. This means that the targets which have not yet become visible (have not been born yet) at any time step k have independent Gaussian prior distributions $\mathrm{N}(\mathbf{x}_{k,j} \mid \mathbf{m}_0, \mathbf{P}_0)$. If we denote the sets of not visible and visible target indices with $\mathcal{J}_0$ and $\mathcal{J}_1$, respectively, the joint prior distribution of all targets is of the form

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = &\prod_{j \in \mathcal{J}_1} \mathrm{N}(\mathbf{x}_{k,j} \mid \mathbf{m}_{k,j}, \mathbf{P}_{k,j}) \\ &\times \prod_{j' \in \mathcal{J}_0} \mathrm{N}(\mathbf{x}_{k,j'} \mid \mathbf{m}_0, \mathbf{P}_0). \end{aligned} \tag{65}$$

   That is, the distribution of the visible targets is completely independent of the distributions of the invisible targets and thus it suffices to store only the states of the visible targets instead of all T_∞ targets. The joint distribution of all targets is still always theoretically Gaussian distribution of dimension T_∞.

3. When a target birth occurs, that is, a new target becomes visible (i.e., produces the first measurement) a new item in the indicator vector $\mathbf{e}_k$ is set and the corresponding target prior distribution is updated (initialized) by the measurement. Because we only need to store one possible permutation from a high number of equivalent target permutations, we can add the new target to the first empty place in the indicator vector $\mathbf{e}_k$.

4. When a target dies, that is, becomes invisible again, the target distribution again becomes the prior and the target state is moved to the end of the joint state vector $\mathbf{x}_k$ and indicator vector $\mathbf{e}_k$. The targets in the vectors can be shifted such that the visible targets always remain in the beginning of the vectors.

5. The target dynamics, target measurements and clutter measurements are modeled in the same way as in the RBMCDA model of Section **??** and thus they fit into the RBPF framework easily.

6. By constructing a prior model for births and deaths, we get the distribution

$$p(\mathbf{e}_k \mid \mathbf{e}_{k-1}), \tag{66}$$

   which defines the dynamics of births and deaths. The data association model is of the form

$$p(c_k \mid c_{k-m:k-1}, \mathbf{e}_k), \tag{67}$$

   and thus these two models together give a joint Markov chain model for the indicators:

$$\begin{aligned} &p(\mathbf{e}_k, c_k \mid c_{k-m:k-1}, \mathbf{e}_{k-m:k-1}) \\ &= p(c_k \mid c_{k-m:k-1}, \mathbf{e}_k)\, p(\mathbf{e}_k \mid \mathbf{e}_{k-1}), \end{aligned} \tag{68}$$

   which is the form required by the RBPF model.

Similarly to the case of RBMCDA with known number of targets, because the targets are a priori independent, conditional on data associations c_k and indicators $\mathbf{e}_k$, the targets will also remain independent during tracking. This means that exactly the same simplifications to RBPF apply to the case of an unknown number of targets as to a known number of targets.

4.4 Evaluating and Sampling from the Optimal Importance Distribution

The possible events between two measurements $\mathbf{y}_{k-1}$ and $\mathbf{y}_k$ and at the association of measurement $\mathbf{y}_k$ are:

1. Targets may die (indicated by elements of $\mathbf{e}_k$):
   (a) none of the targets dies
   (b) one or more targets die
2. $\mathbf{y}_k$ is associated with (indicated by c_k):
   (a) clutter
   (b) one of the existing targets
   (c) a newborn target

Death events are independent of the measurements. However, the two event families are related such that a new measurement $\mathbf{y}_k$ can be associated only to the targets that have not died between the measurements $\mathbf{y}_{k-1}$ and $\mathbf{y}_k$.

As was stated in the original article [23] the number of different events grows exponentially with the number of targets due to the exponential number of possible combinations of target deaths. This problem was avoided by restricting the possible number of deaths to one at each time step, so the number of possible event combinations grows quadratically with the number of targets. Thus, the approximate prior distribution of births, deaths and associations can be constructed as follows:

1. Enumerate all possible combinations of joint birth, (zero or one) deaths, and association events and compute probabilities for each of the combinations.
2. Normalize the list of events such that their probabilities sum to one.

For each combination of birth, death and association events there is a transition pair $(\mathbf{e}_{1:k-1}, c_{1:k-1}) \rightarrow (\mathbf{e}_k, c_k)$ with a probability given by the above procedure. That is, we have an approximate representation of the distribution

$$p(\mathbf{e}_k, c_k \mid \mathbf{e}_{1:k-1}, c_{1:k-1}). \tag{69}$$

The likelihood term $p(\mathbf{y}_k \mid \mathbf{e}_k, c_k)$ can be computed similarly as in the case of a known number of targets (see Equation (38)). By multiplying each of the birth, death, and association combinations with the measurement likelihood and normalizing, we can form the optimal importance distribution similarly as in Section 3.3.

4.5 Data Representation

The algorithm state consists of a set of N particles, where each particle i at time step k contains the following:

$$\{c^{(i)}_{k-m+1:k}, \mathbf{e}^{(i)}_k, \mathbf{m}^{(i)}_{k,1}, \ldots, \mathbf{m}^{(i)}_{k,j}, \ldots, \mathbf{m}^{(i)}_{k,T}, \mathbf{P}^{(i)}_{k,1}, \ldots, \mathbf{P}^{(i)}_{k,j}, \ldots, \mathbf{P}^{(i)}_{k,T}, w^{(i)}_k\}, \tag{70}$$

where

- $c^{(i)}_{k-m+1:k}$ are the data association indicators of the time steps $k-m+1, \ldots, k$.
- $\mathbf{e}^{(i)}_k$ is the life-indicator, which is a binary vector of length T_∞ indicating which of the targets are alive at current time step.
- $\mathbf{m}^{(i)}_{k,j}$,$\mathbf{P}^{(i)}_{k,j}$ are the mean and covariance of the target j, and they are conditional on the data association history $c^{(i)}_{1:k}$.

- $w_k^{(i)}$ is the importance weight of the particle.

The following information is also implicitly or explicitly stored for each particle:

$$\{T_k^{(i)}, \tau_{k,j}^{(i)}, \mathrm{id}_{k,j}^{(i)}, \}, \tag{71}$$

where

- $T_k^{(i)}$ is the number of targets.
- $\tau_{k,j}^{(i)}$ is the time of the last measurement associated with target j.
- $\mathrm{id}_{k,j}^{(i)}$ is a unique integer valued identifier, unique over all targets in all particles, which is assigned at the birth of the target.

In the software implementation the structure used in the particles is now extended to have the following fields:

`M`	Cell array of size $1 \times T$ of T target means.
`P`	Cell array of size $1 \times T$ of T target covariances.
`W`	Importance weight of the particle.
`M0`	Prior mean for targets.
`P0`	Prior covariance for targets.
`T`	Times of last associated measurements.
`B`	Birth indicator.
`D`	Death index, zero mean none.
`dt`	Elapsed time from last measurement.

A cell array containing particle structures of the above kind can be initialized with the following function:

`nmcda_init`

Initializes a cell array containing N particle struct arrays. All the invisible targets have the same Gaussian prior. The prior particle weights can also be given. If not given, however, uniform distribution is used.

Syntax: `S = nmcda_init(np,M0,P0,dt,[W])`

Input:	`np`	Number of particles.	
	`M0`	$D \times 1$ vector containing the prior mean of invisible targets.	
	`P0`	$D \times D$ matrix containing the prior covariance of invisible targets.	
	`dt`	Time between measurements.	
	`W`	Prior particle weights.	default: uniform
Output:	`S`	$1 \times N$ cell array containing N particle structures.	

4.6 Software Implementation

The concept of dividing the data association algorithm into prediction, update and resampling phases is also used in the extended version of the RBMCDA algorithm. Thus, the main program can be constructed similarly as was described in the section 3.4.

The functions of the extended RBMCDA algorithm are described below.

kf_nmcda_predict

Computes Kalman filter prediction (eq. (22)) for each visible target in each particle separately.

Syntax: `S = kf_nmcda_predict(S,A,Q,B,u)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`A`	Transition matrix of the dynamic model.	
	`Q`	Process noise covariance matrix of the dynamic model.	
	`B`	Input effect matrix.	default: identity
	`u`	Constant input.	default: empty
Output:	`S`	$1 \times N$ cell array containing the struct arrays of predicted particles.	

kf_nmcda_update

Processes the target births and deaths as well as the data associations in the current time step.
The (approximate) optimal importance distribution is formed by numerating all the possible events.
After that new latent values are drawn, and based on the drawn values the particles are updated accordingly.

Syntax: `[S,EV_STRS] = kf_nmcda_update(S,Y,t,H,R,CP,CD,pb,alpha,beta)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	$D \times 1$ measurement vector.	
	`t`	Time stamp of the measurement.	
	`H`	Measurement matrix.	
	`R`	Measurement noise covariance.	
	`CP`	Prior probability of a measurement being due to clutter.	default: zero
	`CD`	Probability density of clutter measurements.	default: 0.01
	`pb`	Prior probability of birth.	default:
	`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
	`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
Output:	`S`	$1 \times N$ cell array of updated particle structures.	
	`EV_STRS`	Comment strings of happened events.	

kf_nmcda_update_sa

Same as `kf_nmcda_update` with the exception that the maximum number of associations per target is restricted to be one in each time step.

Syntax: `[SS,EV_STRS] = kf_nmcda_update_sa(S,Y,t,H,R,CP,CD,pb,alpha,beta)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	$D \times M$ measurement matrix.	
	`t`	Time stamp of the measurement.	
	`H`	Measurement matrix.	
	`R`	Measurement noise covariance.	
	`pd`	Target detection probability.	default: one
	`CD`	Probability density of clutter measurements.	default: 0.01
	`pb`	Prior probability of birth.	default:
	`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
	`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
Output:	`SS`	$M \times N$ cell array of updated particle structures.	
	`EV_STRS`	Comment strings of happened events.	

ekf_nmcda_predict

Computes Kalman filter prediction (eq. (22)) for each visible target in each particle separately. If the dynamic models are non-linear EKFs are used instead of standard KFs.

Syntax: `S = ekf_nmcda_predict(S,A,Q,a,AW,param)`

Input:

`S`	$1 \times N$ cell array containing particle structures.	
`A`	Derivative of dynamic model w.r.t. to state as matrix if common for all targets or as cell array of size $T \times N$ for each target in each particle. Can also be a inline function or name of function in form `A(x,i,param)`, where `i` is the index of the target `x`. If the model is linear state transition matrices are given instead of Jacobians.	default: eye
`Q`	Process noise of discrete model as matrix if common for all targets, or as cell array of size $T \times 1$ for all targets separately.	default: zero
`a`	Dynamic model as cell array of size $T \times N$ for each target in each particle, or inline function or name of function in form `a(x,i,param)`, where `i` is the index of the target `x`.	default: `A*X`
`AW`	Derivative of `a` w.r.t. noise as matrix if common for all targets or as cell array of size $T \times N$ for each target in each particle. Can also be a inline function or name of function in form `AW(x,i,param)`, where `i` is the index of the target `x`.	default: eye
`param`	Parameters of `A`, `a` and `AW`.	

Output: `S` $1 \times N$ cell array containing the struct arrays of predicted particles.

ekf_nmcda_update

Does the same as `kf_nmcda_update` with the exception that KF is replaced with EKF.

Syntax: `[S,ES] = ekf_nmcda_update(S,Y,t,H,R,h,V,CP,CD,pb,alpha,beta,param)`

Input:

`S`	$1 \times N$ cell array containing particle structures.	
`Y`	$D \times 1$ measurement vector.	
`t`	Time stamp of the measurement.	
`H`	Derivative of `h` with respect to state as matrix, inline function or name of function in form `H(x,param)`.	
`R`	Measurement noise covariance.	
`h`	Mean measurement prediction as vector, inline function or name of function in form `h(x,param)`.	
`V`	Derivative of `h` w.r.t to noise as matrix, inline function or name of function in form `V(x,param)`.	
`CP`	Prior probability of a measurement being due to clutter.	default: zero
`CD`	Probability density of clutter measurements.	default: 0.01
`pb`	Prior probability of birth.	default:
`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
`param`	Parameters of `H`,`h` and `V`.	

Output:

`S`	$1 \times N$ cell array of updated particle structures.
`ES`	Comment strings of happened events.

ekf_nmcda_update_sa

Does the same as `kf_nmcda_update_sa` with the exception that KF is replaced with EKF.

Syntax: `[SS,ES] = ekf_nmcda_update_sa(S,Y,t,H,R,h,V,CP,CD,pb,alpha,beta,param)`

Input:

`S`	$1 \times N$ cell array containing particle structures.	
`Y`	$D \times M$ measurement matrix.	
`t`	Time stamp of the measurement.	
`H`	Derivative of `h` with respect to state as matrix, inline function or name of function in form `H(x,param)`.	
`R`	Measurement noise covariance.	
`h`	Mean measurement prediction as vector, inline function or name of function in form `h(x,param)`.	
`V`	Derivative of `h` w.r.t to noise as matrix, inline function or name of function in form `V(x,param)`.	
`pd`	Target detection probability.	default: one
`CD`	Probability density of clutter measurements.	default: 0.01
`pb`	Prior probability of birth.	default:
`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
`param`	Parameters of `H`,`h` and `V`.	

Output:

`S`	$M \times N$ cell array of updated particle structures.
`ES`	Comment strings of happened events.

ukf_nmcda_predict

Computes Kalman filter prediction (eq. (22)) for each visible target in each particle separately.
If the dynamic models are non-linear UKFs are used instead of standard KFs.

Syntax: `S = ukf_nmcda_predict(S,a,Q,param)`

Input:

`S`	$1 \times N$ cell array containing particle structures.	
`a`	Dynamic model as cell array of size $T \times N$ for each target in each particle, or inline function or name of function in form `a(x,i,param)`, where `i` is the index of the target `x`.	default: `A*X`
`Q`	Process noise of discrete model as matrix if common for all targets, or as cell array of size $T \times 1$ for all targets separately.	default: zero
`param`	Parameters of `a`.	

Output:

`S`	$1 \times N$ cell array containing the struct arrays of predicted particles.

ukf_nmcda_update

Does the same as `kf_nmcda_update` with the exception that KF is replaced with UKF if the measurement model is non-linear.

Syntax: `[S,ES] = ukf_nmcda_update(S,Y,t,h,R,CP,CD,pb,alpha,beta,param)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	$D \times 1$ measurement vector.	
	`t`	Time stamp of the measurement.	
	`h`	Mean measurement prediction as vector, inline function or name of function in form `h(x,param)`.	
	`R`	Measurement noise covariance.	
	`CP`	Prior probability of a measurement being due to clutter.	default: zero
	`CD`	Probability density of clutter measurements.	default: 0.01
	`pb`	Prior probability of birth.	default:
	`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
	`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
	`param`	Parameters of `H`,`h` and `V`.	
Output:	`S`	$1 \times N$ cell array of updated particle structures.	
	`ES`	Comment strings of happened events.	

ukf_nmcda_update_sa

Does the same as `kf_nmcda_update_sa` with the exception that KF is replaced with UKF if the measurement model is non-linear.

Syntax: `[SS,ES] = ukf_nmcda_update_sa(S,Y,t,h,R,CP,CD,pb,alpha,beta,param)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	$D \times M$ measurement matrix.	
	`t`	Time stamp of the measurement.	
	`h`	Mean measurement prediction as vector, inline function or name of function in form `h(x,param)`.	
	`R`	Measurement noise covariance.	
	`pd`	Target detection probability.	default: one
	`CD`	Probability density of clutter measurements.	default: 0.01
	`pb`	Prior probability of birth.	default:
	`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
	`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
	`param`	Parameters of `H`,`h` and `V`.	
Output:	`SS`	$M \times N$ cell array of updated particle structures.	
	`ES`	Comment strings of happened events.	

4.7 Demonstration: Tracking Unknown Number of 1D Signals

In this section we shall demonstrate the RBMCDA algorithm for unknown number of targets by estimating multiple and time-varying number of one dimensional Gaussian signals from noisy measurements, which corrupted by noise and clutter measurements.

The dynamic model is the same as in the previous demonstrations with the exception that there is new only one dimension instead of two. Thus, the dynamics can be written in form

$$\begin{pmatrix} x_k \\ \dot{x}_k \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{pmatrix} + \mathbf{q}_{k-1}, \tag{72}$$

where $x_k = x(t_k), \dot{x}_k = \dot{x}(t_k)$, the step size $\Delta t = 1/100$, and the spectral density of process noise is $q = 1/10$. The signal conditioned measurements are modeled (and simulated) as measurements

Signal	*Appears*	*Disappears*
$x^{(1)}(t)$	$t = 0$	$t = 8$
$x^{(2)}(t)$	$t = 0$	$t = 15$
$x^{(3)}(t)$	$t = 1$	$t = 4$
$x^{(4)}(t)$	$t = 2$	$t = 5$
$x^{(5)}(t)$	$t = 5.5$	$t = 10$
$x^{(6)}(t)$	$t = 6$	$t = 15$

of the signal plus a white Gaussian noise component

$$y_{k,j} = x^{(j)}(t_k) + r_k, \tag{73}$$

where $r_k \in \mathrm{N}(0, 1/5^2)$, given that the measurement is from signal j. Every measurement has an equal chance of originating from each of the visible signals and 1% change of being a clutter measurement uniformly distributed on the area $[-5, 5]$. The prior distribution for a new born signal was a Gaussian distribution with mean $\mathbf{m}_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$ and covariance $\mathbf{P}_0 = \mathrm{diag}(100, 10)$. The prior probability of birth is $p_b = 1/100$.

The real number of signals is six, and their appearances and disappearances follow the times listed in the table 4.7. The signals and the simulated measurements are plotted in figure 6.

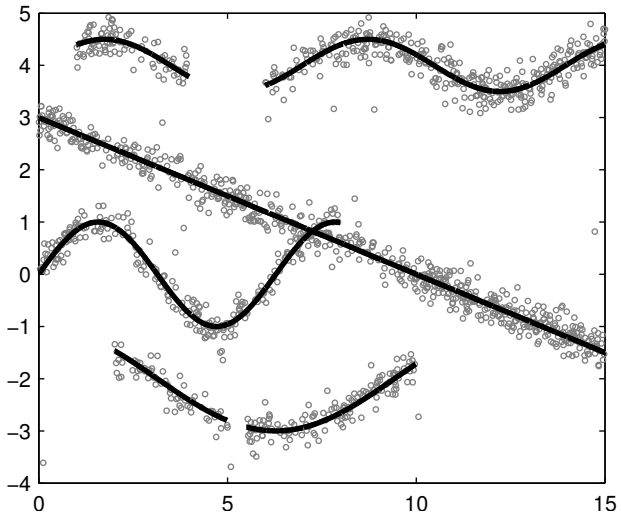

Figure 6: Simulated data of *Tracking Unknown Number of 1D Signals* demonstration.

The filtering loop the extended RBMCDA algorithm is in this case very easy to write:

```
for k=size(Y,2)
    % Filter with RBMCDA
    S = kf_nmcda_predict(S,A,Q);
    S = kf_nmcda_update(S,Y(:,k),T(k),H,R,cp,cd,[],alpha,beta);

    % Perform resampling, saving and plotting of the particles here
end
```

The variable `S` is assumed to contain particle structures, which can be initialized with function `nmcda_init`. In addition to the parameters of dynamic and measurement models (variables `A`, `Q`,

H and R) we also need pass the measurements (Y), time of the current sampling period (T), clutter prior (cp), clutter density (cd) and death model parameters (alpha and beta) for the RBMCDA functions. The prior probability of the birth p_b is not passed (empty vector [] in the argument list) to the update function as we use default value $p_b = 1/100$. The end of the loop (that is, resampling, saving and plotting of the particles) is done similarly as in the previous demonstration cases.

The target trajectories can now be collected from the $N_t \times N_p$ sized cell array SS containing N_p filtered particles structures on N_t time steps by the following function call:

```
[FM,FP,SM,SP,Times] = kf_nmcda_collect(SS,A,Q);
```

The function returns the filtered means and covariances for each and particle in variables FM and FP. The function also smooths the target trajectories (hence the parameters A and Q are passed to the function) and the results are returned in variables SM and SP. Also the the time instances of each target's lifetimes are returned in variable Times. The full software code of this example can found in file multisignal_demo.m.

The left panel of figure 7 shows the filtering results using $N = 10$ particles and parameter values $\alpha = 2$ and $\beta = 1$ in the gamma distribution modeling the target deaths. Clearly the signal death model is now too slow as the lowest signal stays alive during the gap in the signal. There is also notable delays in the disappearances of all signals. This can also be seen in figure 8, which shows the estimated and real number of alive signals in each sampling period. The smoothed estimates are shown in right panel of figure 7. The estimates are quite close to the true signal trajectories expect for the mentioned delays in signal disappearances.

In figure 9 we have plotted the filtering and smoothing results using $N = 10$ particles and death model parameters $\alpha = 2$ and $\beta = 0.1$, which means that signals disappear almost ten times faster. This removes the delays from the signal disappearings, but also causes them to disappear and reappear in time instances, when there is random gaps in signals due to uneven measurement times.

The results above would suggest that the value of β should lie somewhere between the two tested values. As a quick test in figure 11 we have plotted the filtering and smoothing results by using parameters $\alpha = 2$ and $\beta = 0.4$ with $N = 100$ particles. The signal appereances and disapperances are now very close to right values.

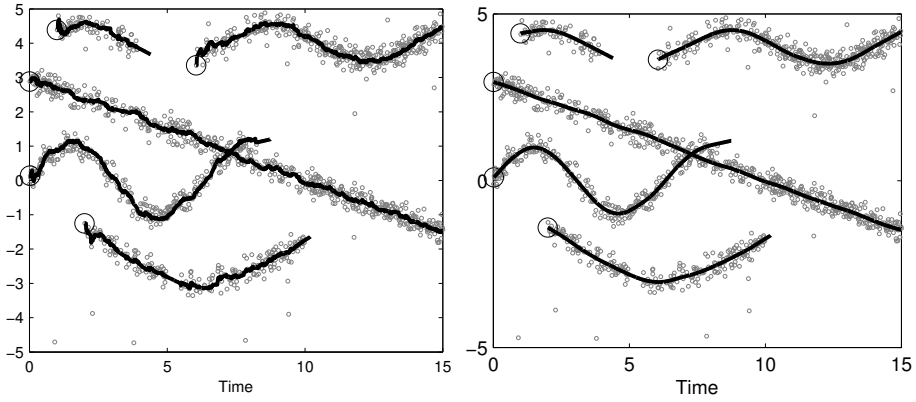

Figure 7: Filtering (left) and smoothing (right) results of the 1D scenario with an unknown number of signals and parameters $\alpha = 2$ and $\beta = 1$ with $N = 10$ particles. The circles represent the estimated starting points of the signals.

4.8 Demonstration: Tracking Unknown Number of Targets in 2D

In this section we shall extend the previous demonstration to have the targets moving in two dimensional space. The usage of data association priors for restricting the number of target measurements

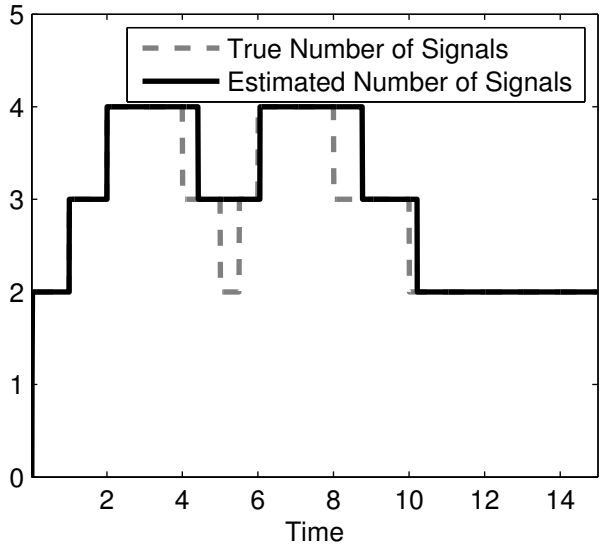

Figure 8: Estimated number of signals in the 1D scenario with an unknown number of signals and parameters $\alpha = 2$ and $\beta = 1$ with $N = 10$ particles.

on each scan to one is illustrated

The dynamic model for the targets is the same as in the sections 3.5 and 3.6, that is, two dimensional discretized Wiener velocity model with process noise spectral density $q = 0.1$. There are four targets, which appear and disappear on different times. The trajectories of the targets are shown in figure **??**, which were generated by giving the targets slightly randomized accelerations similarly as was done in the demonstration in section 3.5.

Also the measurement model is the same as in the section 3.5, but now we allow the possibility of acquiring multiple measurements on a single time step. This means that we perform RBMCDA prediction step once per time step and update step for each measurement on that step. Thus, the filtering loop can be written as

```
t = 0;
for k=1:size(Y,2)
    if T(k) - t > 0
        S = kf_nmcda_predict(S,A,Q);
    end
    t = T(k);
    [S,E] = kf_nmcda_update(S,Y(:,k),T(k),H,R,cp,cd,[],alpha,beta);

    % Perform resampling, saving and plotting of the particles here
end
```

First we check whether the time stamp of the kth measurement (stored in variable `T`) is bigger than the time of the previously processed measurement, which is stored in variable `t`. If true, we call the RBMCDA prediction function. After that we update variable `t` with new value and call the RBMCDA update function normally. As previously, the resampling, saving and plotting commands goes to the end of the loop. Also the smoothing is done similarly as in the previous demonstration.

The figure 13 shows the filtering and smoothing results, when the measurements were generated on each time step such that only one measurement origins from a randomly selected visible target and the rest is clutter on area $[-2, 2] \times [-2, 2]$. The number of clutter measurements is drawn from a Poisson distribution with mean 0.1. The parameters of the death model was $\alpha = 2$ and $\beta = 0.5$ and

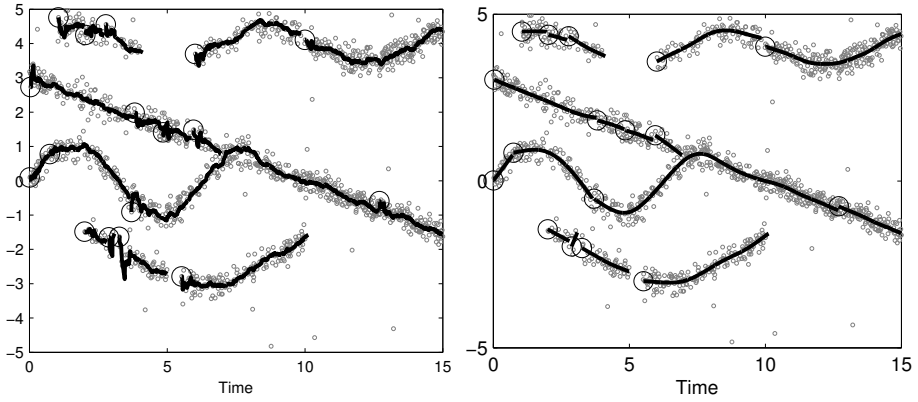

Figure 9: Filtering (left) and smoothing (right) results of the 1D scenario with an unknown number of signals and parameters $\alpha = 2$ and $\beta = 0.1$ with $N = 10$ particles.

the number of used particles was $N = 100$. The full software code used to produce these estimates can be found in file `kf_mt_demo1.m`.

In figure 14 we have plotted the filtering results when the number of clutter measurements on each time step were drawn from a Poisson distribution with mean 1. The target measurements are also changed such that on each sampling period each target has the probability $p_d = 0.95$ of generating a measurement. Clearly the higher clutter count makes the estimation harder, which can be seen as false targets.

As the amount target measurements per each sampling period is one we can enhance the estimation performance by constructing this information to the data association priors, as was done the section 3.6. In this case we have ready functions for doing this, for which the filtering loop can be written as follows:

```
SS = {};
t = 0;
k = 1;
count = 0;
while k <= size(Y,2)
    % Perform prediction similarly as above

    % Gather all the measurements of the current
    % sampling period into a matrix y.
    y = [];
    while t == T(k)
        y = [y Y(:,k)];
        k = k+1;
        if k > size(Y,2)
            break;
        end
    end
```

Before going into the loop we shall first initialize the needed variables. After that in the while loop the prediction step is done similarly as in the previous case. For the update step, however, we must first gather all the measurements having the current time stamp into a single matrix. Then we can call the update step function as follows:

```
    [SA,E] = kf_nmcda_update_sa(S,y,t,H,R,pd,cd,[],alpha,beta);
```

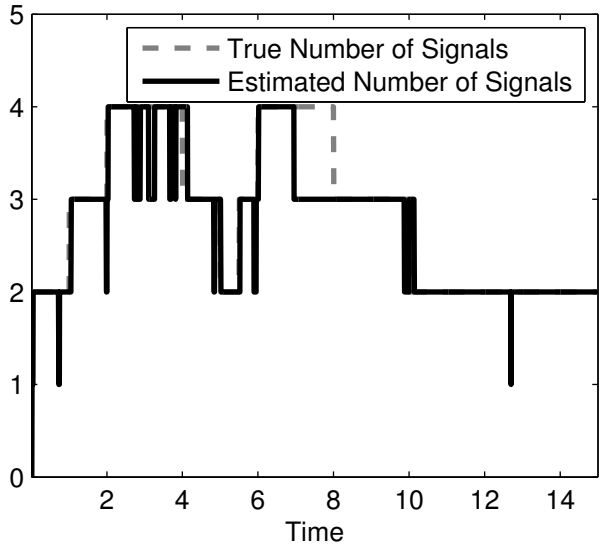

Figure 10: Estimated number of signals in the 1D scenario with an unknown number of signals and parameters $\alpha = 2$ and $\beta = 0.1$ with $N = 10$ particles.

```
    % Store the estimates after processing each measurement
    for j=1:length(SA)
        count = count+1;
        S = SA{j};
        SS(count,:) = S;
    end

    % Perform resampling and plotting here
end
```

The modified update function `kf_nmcda_update_sa` produces particle structures for each measurement in the current time step, so must save all of them individually. As previously, the filtering loop can be concluded with resampling and plotting. The full software code of this example can be found in file `kf_mt_demo2.m`.

The filtering and smoothing results using the restrictions of data associations is shown in figure 15 when the amount of clutter measurements per each time step were drawn from a Poisson distribution with mean 10. Despite the heavy amount of clutter the algorithm is able to estimate the target trajectories correctly. Without the restriction even with $N = 1000$ particles the estimates are much worse.

4.9 Alternative Formulation for Target Deaths

We shall now present a new modification to the RBMCDA algorithm reviewed in this section, in which the target deaths are processed in the prediction step of the algorithm rather than processing them jointly with other possible events in the update step. This is possible as the target deaths are independent of the measurements. The advatage of this formulation is that the number of different event combinations is reduced as we first process the deaths of T_{k-1} targets and after that the data associations of T_k targets (that is, association to any existing target, clutter or to a newly born target). Hence, the number of different events grows linearly.

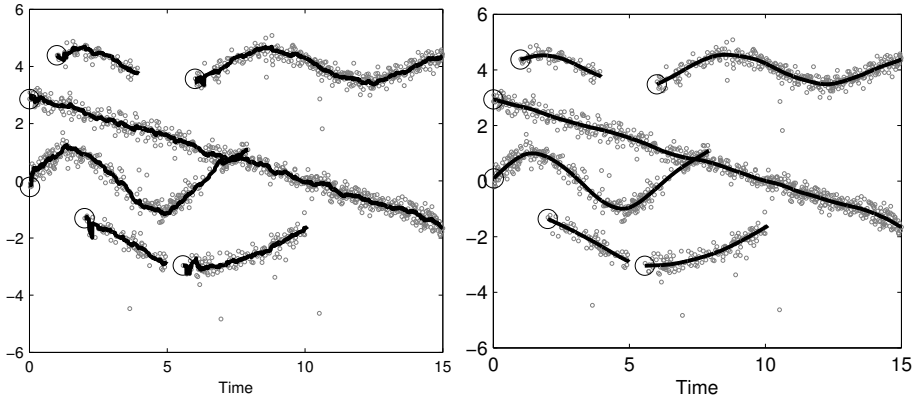

Figure 11: Filtering (left) and smoothing (right) results of the 1D scenario with an unknown number of signals and parameters $\alpha = 2$ and $\beta = 0.4$ with $N = 100$ particles.

We could now also remove the restriction of having only one target death on each time step as the deaths of targets are processed independently. This would actually help the processing of data associations as number of different association events is smaller as there are smaller amount of target left the more targets die in the prediction step. However, this feature is not currently included in the toolbox.

The functions for the modified algorithm are described below. They can be recongnized from the suffix `_dp`. The difference between these functions and those described above is that now the parameters α and β of death model are moved from update functions to predict functions. The time of the current sampling period must also be passed now to the predict function for processing the target deaths.

`kf_nmcda_predict_dp`

Computes Kalman filter prediction (eq. (22)) for each visible target in each particle separately. Processes the target deaths.

Syntax: `[S,EV_STRS] = kf_nmcda_predict_dp(S,A,Q,B,u,t,alpha,beta)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`A`	Transition matrix of the dynamic model.	
	`Q`	Process noise covariance matrix of the dynamic model.	
	`B`	Input effect matrix.	default: identity
	`u`	Constant input.	default: empty
	`t`	Time of the prediction step	
	`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
	`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
Output:	`S`	$1 \times N$ cell array containing the struct arrays of predicted particles.	
	`EV_STRS`	Comment strings of happened events.	

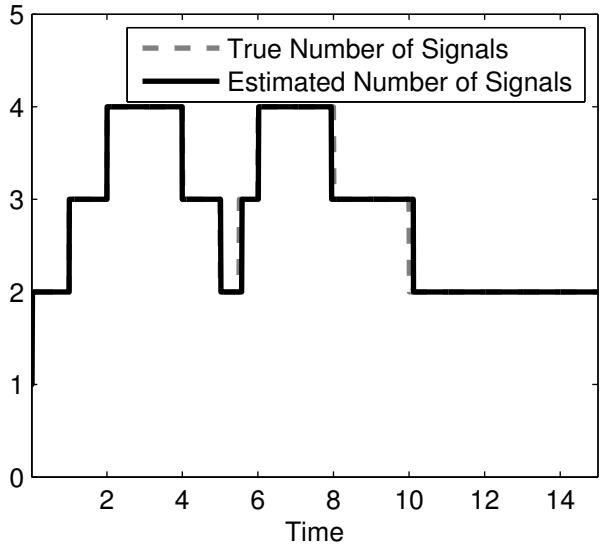

Figure 12: Estimated number of signals in the 1D scenario with an unknown number of signals and parameters $\alpha = 2$ and $\beta = 0.4$ with $N = 100$ particles.

kf_nmcda_update_dp			
Processes the target births and the data associations in the current time step. The (approximate) optimal importance distribution is formed by numerating all the possible events. After that new latent values are drawn, and based on the drawn values the particles are updated accordingly.			
Syntax: `[S,EV_STRS] = kf_nmcda_predict_dp(S,Y,t,H,R,CP,CD,pb)`			
Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	$D \times 1$ measurement vector.	
	`t`	Time stamp of the measurement.	
	`H`	Measurement matrix.	
	`R`	Measurement noise covariance.	
	`CP`	Prior probability of a measurement being due to clutter.	default: zero
	`CD`	Probability density of clutter measurements.	default: 0.01
	`pb`	Prior probability of birth.	default:
Output:	`S`	$1 \times N$ cell array of updated particle structures.	
	`EV_STRS`	Comment strings of happened events.	

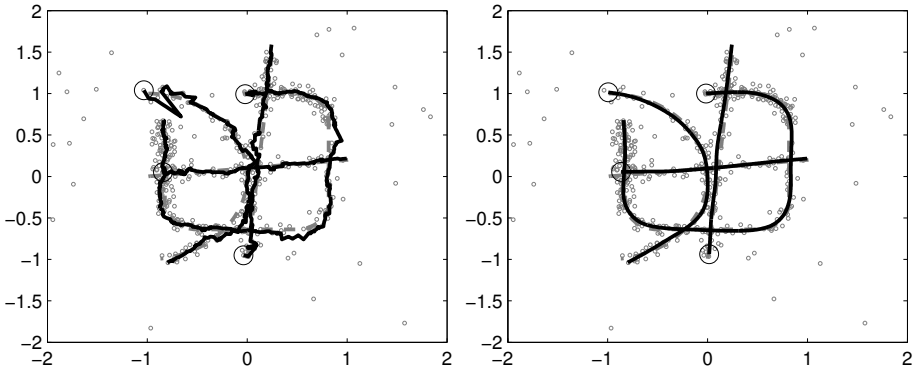

Figure 13: Filtering (left) and smoothing (right) results of the 2D scenario with an unknown number of targets and parameters $\alpha = 2$ and $\beta = 0.5$ with $N = 100$ particles and low amount of clutter measurements.

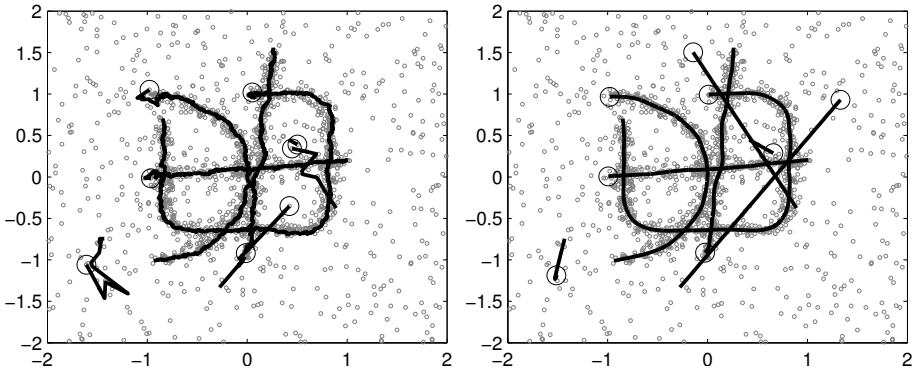

Figure 14: Filtering (left) and smoothing (right) results of the 2D scenario with an unknown number of targets and parameters $\alpha = 2$ and $\beta = 0.5$ with $N = 100$ particles and high amount of clutter measurements.

ekf_nmcda_predict_dp Computes Kalman filter prediction (eq. (22)) for each visible target in each particle separately. Processes the target deaths. If the dynamic models are non-linear EKFs are used instead of standard KFs.			
Syntax: `[S,EV_STRS] = ekf_nmcda_predict_dp(S,A,Q,a,AW,param,t,alpha,beta)`			
Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`A`	Derivative of dynamic model w.r.t. to state as matrix if common for all targets or as cell array of size $T \times N$ for each target in each particle. Can also be a inline function or name of function in form `A(x,i,param)`, where `i` is the index of the target `x`. If the model is linear state transition matrices are given instead of Jacobians.	default: eye
	`Q`	Process noise of discrete model as matrix if common for all targets, or as cell array of size $T \times 1$ for all targets separately.	default: zero
	`a`	Dynamic model as cell array of size $T \times N$ for each target in each particle, or inline function or name of function in form `a(x,i,param)`, where `i` is the index of the target `x`.	default: `A*X`
	`AW`	Derivative of `a` w.r.t. noise as matrix if common for all targets or as cell array of size $T \times N$ for each target in each particle. Can also be a inline function or name of function in form `AW(x,i,param)`, where `i` is the index of the target `x`.	default: eye
	`param`	Parameters of `A`, `a` and `AW`.	
	`t`	Time of the prediction step	
	`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
	`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
Output:	`S`	$1 \times N$ cell array containing the struct arrays of predicted particles.	
	`EV_STRS`	Comment strings of happened events.	

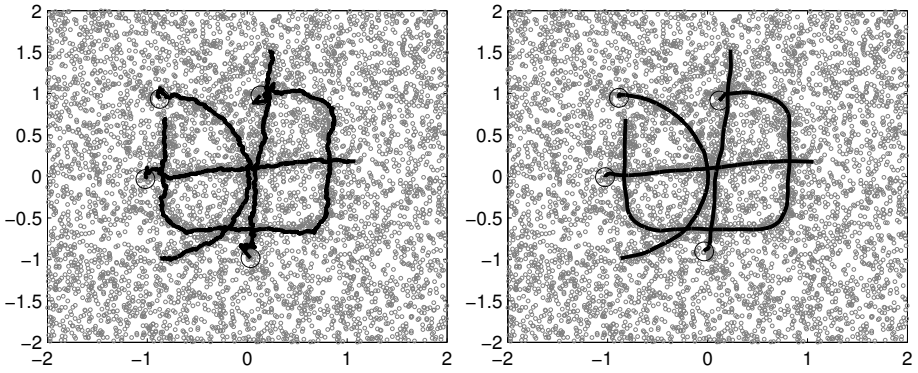

Figure 15: Filtering (left) and smoothing (right) results of the 2D scenario with an unknown number of targets and parameters $\alpha = 2$ and $\beta = 0.5$ with $N = 100$ particles.

ekf_nmcda_update_dp			
Does the same as `kf_nmcda_update_dp` with the exception that KF is replaced with EKF.			
Syntax: `[S,ES] = ekf_nmcda_update_dp(S,Y,t,H,R,h,V,CP,CD,pb,param)`			
Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	$D \times 1$ measurement vector.	
	`t`	Time stamp of the measurement.	
	`H`	Derivative of `h` with respect to state as matrix, inline function or name of function in form `H(x,param)`.	
	`R`	Measurement noise covariance.	
	`h`	Mean measurement prediction as vector, inline function or name of function in form `h(x,param)`.	
	`V`	Derivative of `h` w.r.t to noise as matrix, inline function or name of function in form `V(x,param)`.	
	`CP`	Prior probability of a measurement being due to clutter.	default: zero
	`CD`	Probability density of clutter measurements.	default: 0.01
	`pb`	Prior probability of birth.	default:
	`param`	Parameters of `H`,`h` and `V`.	
Output:	`S`	$1 \times N$ cell array of updated particle structures.	
	`ES`	Comment strings of happened events.	

ukf_nmcda_predict_dp

Computes Kalman filter prediction (eq. (22)) for each visible target in each particle separately.
Processes target deaths.
If the dynamic models are non-linear UKFs are used instead of standard KFs.

Syntax: `[S,EV_STRS] = ukf_nmcda_predict_dp(S,a,Q,param,t,alpha,beta)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`a`	Dynamic model as cell array of size $T \times N$ for each target in each particle, or inline function or name of function in form `a(x,i,param)`, where `i` is the index of the target `x`.	default: `A*X`
	`Q`	Process noise of discrete model as matrix if common for all targets, or as cell array of size $T \times 1$ for all targets separately.	default: zero
	`param`	Parameters of `a`.	
	`t`	Time of the prediction step	
	`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
	`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
Output:	`S`	$1 \times N$ cell array containing the struct arrays of predicted particles.	
	`EV_STRS`	Comment strings of happened events.	

ukf_nmcda_update_dp

Does the same as `kf_nmcda_update` with the exception that KF is replaced with UKF
if the measurement model is non-linear.

Syntax: `[S,ES] = ukf_nmcda_update_dp(S,Y,t,h,R,CP,CD,pb,alpha,beta,param)`

Input:	`S`	$1 \times N$ cell array containing particle structures.	
	`Y`	$D \times 1$ measurement vector.	
	`t`	Time stamp of the measurement.	
	`h`	Mean measurement prediction as vector, inline function or name of function in form `h(x,param)`.	
	`R`	Measurement noise covariance.	
	`CP`	Prior probability of a measurement being due to clutter.	default: zero
	`CD`	Probability density of clutter measurements.	default: 0.01
	`pb`	Prior probability of birth.	default:
	`alpha`	Parameter α for the gamma distribution model for time to death.	default: 1
	`beta`	Parameter β for the gamma distribution model for time to death.	default: 10
	`param`	Parameters of `H`,`h` and `V`.	
Output:	`S`	$1 \times N$ cell array of updated particle structures.	
	`ES`	Comment strings of happened events.	

5 Functions In the Toolbox

All the functions of the toolbox are listed in the table 1 with the page numbers, where they can be found in this document.

5.1 Particle Handling Functions

In addition to the functions of actual RBMCDA algorithm some functions for handling the particle structures are included in the toolbox. These are described below:

Table 1: Functions provided with the toolbox.

calculate_mean	
Calculates the mean of the given target from particle structures.	
Syntax: `M = calculate_mean(S,T)`	
Input:	`S` $N_t \times N_p$ cell array containing particles structures. `T` Index of the target.
Output:	`M` $D \times N_p$ matrix containing the mean of the target.

eff_particles	
Estimates the effective number of particles from a particle structures.	
Syntax: `N_eff = eff_particles(W)`	
Input:	`W` Importance weights as a numeric array or a cell array containing particle structures.
Output:	`N_eff` Estimated number of effective particles.

get_weights	
Extracts an array of weights from particles structures.	
Syntax: `W = get_weights(S)`	
Input:	`S` $N_t \times N_p$ cell array containing particle structures.
Output:	`W` $N_t \times N_p$ matrix containing particle weights.

normalize_weights	
Normalizes the weights in the given particle structures.	
Syntax: `S = normalize_weights(S)`	
Input:	`S` $1 \times N$ cell array containing particle structures.
Output:	`S` Updated particle structures.

resample	
Performs simple resampling for particle filtering.	
Syntax: `ind = resample(W,alg)`	
Input:	`W` Positive weights in a numeric array or in cell array containing particle structures. `alg` Algorithm index. Currently only index 1 is defined.
Output:	`ind` Indices of resampled particles.

set_weights	
Sets the given weights to particle structures.	
Syntax: `W = set_weights(S,W)`	
Input:	`S` $N_t \times N_p$ cell array containing particle structures. `w` $N_t \times N_p$ matrix containing the particle weights.
Output:	`S` $N_t \times N_p$ cell array of updated particle structures.

5.2 Distribution Functions

For convience some useful functions related to probability distributions are also included in the toolbox.

bin_pdf

Probability density function of a Binomial distribution.

Syntax: `p = bin_pdf(x,rho,n)`

Input:	`x`	Location where to evaluate the PDF.
	`rho`	"Probability" parameter.
	`n`	"Sample size" parameter.
Output:	`p`	Density at the given location.

categ_rnd

Draws samples from a given one dimensional discrete distribution.

Syntax: `C = categ_rnd(P,N)`

Input:	`P`	Discrete distribution, which can be a numeric array or a cell array of particle structures, whose weights respresent the distribution.
	`N`	Number of samples to be drawn.
Output:	`C`	Samples in a $N \times 1$ vector.

gamma_cdf

Cumulative density function of a Gamma distribution.

Syntax: `p = gamma_cdf(x,alpha,beta,mu)`

Input:	`x`	Locations where to evaluate the CDF.
	`alpha`	α parameter of the distribution.
	`beta`	β parameter of the distribution.
	`mu`	Mean of the distribution.
Output:	`p`	Cumulative density at the given locations.

gamma_pdf

Probability density function of a Gamma distribution.

Syntax: `p = gamma_pdf(x,alpha,beta,mu)`

Input:	`x`	Locations where to evaluate the PDF.
	`alpha`	α parameter of the distribution.
	`beta`	β parameter of the distribution.
	`mu`	Mean of the distribution.
Output:	`p`	Density at the given locations.

gmm_pdf

PDF of a multivariate mixture of Gaussians distribution.

Syntax: `[P,PJX] = gmm_pdf(X,M,S,PJ)`

Input:	`X`	Locations where to evaluate the PDF as $D \times N$ matrix.
	`M`	$D \times N_c$ matrix of means.
	`S`	$D \times D \times N_c$ matrix of covariances.
	`PJ`	Probabilities of mixture components.
Output:	`P`	Densities at the given locations.
	`PJX`	Probabilities of each component at the given locations.

gmm_rnd			
Draws random samples from a multivariate mixture of Gaussians distribution.			
Syntax: `X = gmm_rnd(M,S,PJ,N)`			
Input:	`M`	$D \times N_c$ matrix of means.	
	`S`	$D \times D \times N_c$ matrix of covariances.	
	`PJ`	Probabilities of mixture components.	
	`N`	Number of samples.	default: 1
Output:	`X`	$D \times N$ matrix of samples.	

poisson_rnd			
Draws random samples from a Poisson distribution.			
Syntax: `X = poisson_rnd(lambda,N)`			
Input:	`lambda`	Rate parameter of the distribution.	
	`N`	Number of samples.	default: 1
Output:	`X`	$1 \times N$ vector of samples.	

References

[1] Akashi, H. and Kumamoto, H. (1977). *Random sampling approach to state estimation in switching environments.* Automatica, 13:429-434.

[2] Bar-Shalom,Y., Li, X.-R., and Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation.* Wiley Interscience.

[3] Bar-Shalom,Y., Li, X.-R., and Kirubarajan, T. (1995). *Multitarget-Multisensor Tracking: Principles and Techniques.* YBS.

[4] Blackman,S., Popoli, R. (1999). *Design and Analysis of Modern Tracking Systems.* Artech House Radar Library.

[5] Stone, L. D., Barlow, C. A., Corwin, T. L. (1999). *Bayesian Multiple Target Tracking.* Artech House, Boston, London.

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

[7] Hartikainen, J., and Särkkä, S. (2008). *EKF/UKF Toolbox for Matlab*. Available at `http://www.lce.hut.fi/research/mm/ekfukf/`.

[8] Fraser, D. C. and Potter, J. E. (1969) *The Optimum Linear Smoother as a Combination of Two Optimum Linear Filters.* IEEE Transactions on Automatic Control AC-14, 387-390.

[9] Gelb, A., editor (1974). *Applied Optimal Estimation.* The MIT Press.

[10] Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). *Novel approach to nonlinear/non-Gaussian Bayesian state estimation* in IEEE Proceedings on Radar an Signal Processing, volume 140, pages 107-113.

[11] Grewal, M. S. and Andrews, A. P. (2001). *Kalman Filtering, Theory and Practice Using MATLAB.* Wiley Interscience.

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

[13] Julier, S. J. and Uhlmann, J. K. (2004). *Unscented filtering and nonlinear estimation.* In Proceedings of the IEEE, 92(3):401-422.

[14] Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. (1995). *A new approach for filtering nonlinear systems.* In *Proceedings of the 1995 American Control, Conference, Seattle, Washington*, pages 1628-1632.

[15] Kalman, R. E. (1960). *A new approach to linear filtering and prediction problems.* Transactions of the ASME, Journal of Basic Engineering, 82:34-45.

[16] Kitagawa, G. (1996). *Monte Carlo filter and smoother for non-Gaussian nonlinear state space models.* Journal of Computational and Graphical Statistics, 5:1-25.

[17] Kotecha, J. H., and Djuric, P. M. (2003). *Gaussian Particle Filtering.* IEEE Transactions on Signal Processing, 51(10):2592-2600.

[18] Maybeck, P. (1982). *Stochastic Models, Estimation and Control, Volume 2.* Academic press.

[19] Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyond the Kalman Filter.* Artech House.

[20] Sage, A. P. and Melsa, J. L. (1971). *Estimation Theory with Applications to Communications and Control.* McGraw-Hill Book Company.

[21] Särkkä, S., Vehtari, A., and Lampinen, J. (2004). *Rao-Blackwellized Monte Carlo data association for multiple target tracking*. In Proceedings of the Seventh International Conference on Information Fusion, volume I, pages 583-590.

[22] Särkkä, S. (2006). *Recursive Bayesian Inference on Stochastic Differential Equations.* Doctoral dissertation, Helsinki University of Technology.

[23] Särkkä, S., Vehtari, A. and Lampinen, J. (2007) *Rao-Blackwellized Particle Filter for Multiple Target Tracking.* Information Fusion, 8(1):2-15.

[24] Särkkä, S. (2006) *Unscented Rauch-Tung-Striebel Smoother.* Submitted.

[25] Wan, E. A. and van der Merwe, R. (2001). *The unscented Kalman filter.* In Haykin, S., editor *Kalman filtering and Neural Networks*, chapter 7. Wiley.