A MODIFIED RAO-BLACKWELLISED PARTICLE FILTER

Frédéric Mustière , Miodrag Bolić, Martin Bouchard

School of Information Technology and Engineering, University of Ottawa
800 King Edward Ave., Ottawa, ON, Canada, K1N 6N5
Email:{mustiere,mbolic,bouchard}@site.uottawa.ca

ABSTRACT

Rao-Blackwellised Particle Filters (RBPFs) are a class of Particle Filters (PFs) that exploit conditional dependencies between parts of the state to estimate. By doing so, RBPFs can improve the estimation quality while also reducing the overall computational load in comparison to original PFs. However, the computational complexity is still too high for many real-time applications. In this paper, we propose a modified RBPF that requires a single Kalman Filter (KF) iteration per input sample. Comparative experiments show that while good convergence can still be obtained, computational efficiency is always drastically increased, making this algorithm an option to consider for real-time implementations.

1. INTRODUCTION

Particle Filters (PF) constitute a family of solutions to the so-called state estimation problem represented by the following coupled equations:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{w}_k) \quad (1)$$

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k) \quad (2)$$

where $\mathbf{x}_k$ represents the state vector at instant k, $\mathbf{z}_k$ is the vector of observations, $\mathbf{w}_k$ and $\mathbf{v}_k$ are the process and the measurement noises, f is a signal transition function and g is a measurement function (both assumed to be known). The goal is to estimate the state based on all available measurements up to time k, which we denote $\mathbf{Z}_k = \mathbf{z}_{0:k}$. The PFs introduce an approximate recursive solution to (1) and (2) for very weak assumptions: f and h may be non-linear, $\mathbf{v}$ and $\mathbf{w}$ may be non-Gaussian, at the cost of a more computationally expensive implementation [1]. PFs exist in many versions [2][3][4], and should be tailored to the particular dynamics of the problem that they address. Rao-Blackwellised Particle Filters (RBPF) are a branch of PFs applicable under certain state dynamics, aiming at improving convergence and efficiency [2][5].

The increase in popularity of PFs is still inhibited by their computational load [1]. We present here a version of RBPF which can further improve efficiency, making this method more attractive for real-time implementations. In the next paragraphs, a brief presentation of the generic PF algorithm will be shown. Then, a section on regular RBPFs will follow. Next, the modification of the RBPF algorithm along with its rationale will be introduced, followed by experimental results obtained from the application of this algorithm to tracking and signal processing examples. We will conclude on the pros and cons of using the modified algorithm.

Thanks to NSERC for funding this work.

2. GENERIC PARTICLE FILTERS

We refer the reader to [1] for a complete derivation of the PF algorithm summarized here. We use the subscripts k, i to denote the i^{th} particle at instant k. Note that the algorithm requires a prior choice of an importance density $q(\mathbf{x}_k|\mathbf{x}_{k-1,i}, \mathbf{z}_k)$, and an initialization step (see [1]).

PF algorithm

for every k, do the following:
∘ For every $i \in \{1, 2, .., N\}$

- Draw $\mathbf{x}_{k,i} \sim q(\mathbf{x}_k|\mathbf{X}_{k-1,i}, \mathbf{Z}_k)$
  and set $\mathbf{X}_{k,i} = \{\mathbf{x}_{k,i}\ ; \mathbf{X}_{k-1,i}\}$
- Compute the unnormalized weights
  $\tilde{w}_{k,i} = w_{k-1,i} \frac{p(\mathbf{z}_k|\mathbf{X}_{k,i}, \mathbf{Z}_{k-1}) p(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{Z}_{k-1})}{q(\mathbf{x}_{k,i}|\mathbf{X}_{k-1,i}, \mathbf{Z}_k)}$

∘ Compute the normalizing factor $\sum_i \tilde{w}_{k,i}$ and obtain normalized weights $w_{k,i}$.
∘ If needed, resample particles

The estimates of $\mathbf{x}_k$ can be computed at time instant k from $p(\mathbf{x}_k|\mathbf{Z}_k) \simeq \sum_i w_{k,i} \delta(\mathbf{x}_k - \mathbf{x}_{k,i})$. Even though, in theory, particle filtering offers the most flexibility, as mentioned before it is also a computationally expensive approach. In many situations, in order to obtain reliable estimates for the states, a large number of particles is required, especially when the dimension of the state vector is large [1]. RBPFs are an important modification of the generic PF, applicable to a certain class of state-space models. Using RBPFs, fewer particles are needed to achieve similar convergence [1][2][5].

3. RAO-BLACKWELLISED PARTICLE FILTERS

The idea is to try to exploit the structure of the state vector. If some conditional dependencies between elements of the state vector can be analytically explicited, then there is no need to draw samples from the entire state space. Specifically, suppose that $\mathbf{x}_k = \{\mathbf{x_1}_k \ ; \mathbf{x_2}_k\}$ represents the state vector, for which $p(\mathbf{x_2}_k|\mathbf{X_1}_k, \mathbf{Z}_k)$ can be evaluated. This way, it is possible to run a PF on $\mathbf{X_1}_k$ while updating the corresponding particles of $\mathbf{x_2}_k$ using $p(\mathbf{x_2}_k|\mathbf{X_1}_k, \mathbf{Z}_k)$. RBPFs are in practice mostly used when part of the state evolves in a linear-Gaussian fashion, conditioned upon the other part of the state. In this context, $p(\mathbf{x_2}_k|\mathbf{X_1}_k, \mathbf{Z}_k)$ is Gaussian and one can use a set of N KFs to update particles of $\mathbf{x_2}_k$. Note here that N KF iterations are required at each step for an implementation using N particles.

The RBPF procedure has been shown to sharply reduce the variance of the error estimates [2][5]. Moreover, since the dimension of the part of the state on which the PF is running is smaller, we can expect to outperform, in efficiency, the regular PF: even though more operations are performed per particle, fewer particles are needed to achieve a given convergence [5]. We now present formally the RBPF algorithm:

RBPF algorithm

For every *k*, do the following:
Simulation part (PF)
∘ Run a PF on the sub-state $\mathbf{x_1}_k$
Exact part (KF)
∘ For every *i*, update $p(\mathbf{x_2}_k|\mathbf{X_1}_{k,i}, \mathbf{Z}_k)$ using $p(\mathbf{x_2}_{k-1}|\mathbf{X_1}_{k-1,i}, \mathbf{Z}_{k-1})$, $\mathbf{x_1}_{k,i}$, $\mathbf{x_1}_{k-1,i}$, and $\mathbf{z}_k$.

The RBPF algorithm is very similar to the regular PF, except for the so-called exact part.

4. A MODIFICATION OF THE RBPF ALGORITHM

4.1. Presentation of the algorithm

The main difference from the RBPF approach lies in the way that the particles $\{\mathbf{x_2}_{k,i}\}_{i=1}^N$ are propagated to the next iteration of the algorithm. Let us first describe how the algorithm operates, and how the two "subfilters" – Particle and Kalman – interact. At step k, the PF will compute its own estimate, $\hat{\mathbf{x}}_{\mathbf{1}k}$. This estimate is passed to a single KF to update the mean $\hat{\mathbf{x}}_{\mathbf{2}k}$ and error covariance matrix $\mathbf{K}_k$ of an estimate for $\mathbf{x_2}_k$. In turn, $\{\hat{\mathbf{x}}_{\mathbf{2}k}; \mathbf{K}_k\}$ are passed to the PF at the beginning of the next step, which still carries the particles $\mathbf{x_1}_{k,i}$. A new procedure then takes place: a set of N new samples $\mathbf{x_2}_{k,i} \sim \mathcal{N}(\mathbf{x_2}_k|\hat{\mathbf{x}}_{\mathbf{2}k}, \mathbf{K}_k)$ are generated and coupled to the particles $\{\mathbf{x_1}_{k+1,i}\}_{i=1}^N$ that are drawn at the beginning of the simulation part. With this completed set of particles, the PF can now carry on, and the algorithm continues.

We now formally present the modified RBPF algorithm:

Modified RBPF algorithm

Initialization
∘ Obtain N initial particles of $\mathbf{x_1}_0$
∘ Choose initial mean $\hat{\mathbf{x}}_{\mathbf{2}0}$ and error covariance matrix $\mathbf{K}_0$ for an estimate of $\mathbf{x_2}_0$.

For every *k*, do the following:
Simulation part (PF)
∘ For every $i \in 1, 2, .., N$

- Draw $\mathbf{x_2}_{k-1,i} \sim \mathcal{N}(\mathbf{x_2}_{k-1}|\hat{\mathbf{x}}_{\mathbf{2}k-1}, \mathbf{K}_{k-1})$
- Draw $\mathbf{x_1}_{k,i} \sim q(\mathbf{x_1}_k|\mathbf{X_1}_{k-1,i}, \mathbf{Z}_k)$ and set $\mathbf{X_1}_{k,i} = \{\mathbf{x_1}_{k,i} \ ; \mathbf{X_1}_{k-1,i}\}$
- Compute the unnormalized weights $\tilde{w}_{k,i} = \frac{p(\mathbf{z}_k|\mathbf{X_1}_{k,i}, \mathbf{Z}_{k-1}) p(\mathbf{x_1}_{k,i}|\mathbf{X_1}_{k-1,i}, \mathbf{Z}_{k-1})}{q(\mathbf{x_1}_{k,i}|\mathbf{X_1}_{k-1,i}, \mathbf{Z}_k)}$

∘ Compute the normalizing factor $\sum_i \tilde{w}_{k,i}$ and obtain normalized weights $w_{k,i}$
∘ Resample **only** the particles $\mathbf{x_1}_{k,i}$, with replacement.
Exact part (KF)
∘ Compute $\hat{\mathbf{x}}_{\mathbf{1}k} = N^{-1} \sum_{i=0}^N \mathbf{x_1}_{k,i}$
∘ Obtain $\mathcal{N}(\mathbf{x_2}_k|\hat{\mathbf{x}}_{\mathbf{2}k}, \mathbf{K}_k) = p(\mathbf{x_2}_k|\mathbf{X_1}_k = \hat{\mathbf{X}}_{\mathbf{1}k}, \mathbf{Z}_k)$ using a Kalman Filter and $\hat{\mathbf{x}}_{\mathbf{1}k}$, $\hat{\mathbf{x}}_{\mathbf{2}k-1}$, $\mathbf{K}_{k-1}$ and $\mathbf{z}_k$.

Note that in the exact part, we are updating the distribution $p(\mathbf{x_2}_k|\mathbf{X_1}_k = \hat{\mathbf{X}}_{\mathbf{1}k}, \mathbf{Z}_k)$, and we are taking its mean as our estimate for $\mathbf{x_2}_k$. In contrast, in the regular RBPF, we must update $p(\mathbf{x_2}_k|\mathbf{X_1}_k = \mathbf{X_1}_{k,i}, \mathbf{Z}_k)$ for all i (all particles), take each of their means to form $\{\mathbf{x_2}_{k,i}\}_{i=1}^N$, and only then are we able to form an estimate for $\mathbf{x_2}_k$, using the weights computed by the PF. In the simulation part, the weight computation is identical to that carried out during regular RBPF, but in particular, we have $\forall i, \mathbf{K}_{k,i} = \mathbf{K}_k$ (should these matrices intervene in the computation of the weights). Finally, we consider that resampling is done at each step, so that the available particles of $\mathbf{x_1}_{k,i}$ are as relevant as possible (only a few will have negligible weight). It would not serve any purpose to resample the $\mathbf{x_2}_{k,i}$ particles, however the $\mathbf{x_1}_{k,i}$ are still subject to degeneracy.

4.2. Advantages and drawbacks

As the main advantage, since only 1 KF is used instead of N, the efficiency is drastically increased (as observed by the execution time), especially if the number of particles used is large. Another advantage is that there is less memory usage. In the regular RBPF, the N means and covariance matrices used in the KFs must be stored. If $\mathbf{x_2}_k \in \mathbb{R}^n$, then $N(n+n^2)$ memory locations must be used, instead of $n + n^2$ for the algorithm presented here (since only $\mathbf{K}_k$ must be stored).

On the other hand, we can expect that there will be a loss in performance. Specifically, for the modified algorithm, we are only passing a single estimate of $\mathbf{x_1}_k$ at step k to a single KF, and not the entire approximated distribution to a bank of KFs. If this estimate is an expectation, then the algorithm will not perform as well if the current measure $\{\mathbf{x_1}_{k,i}\ ,\ w_{k,i}\}_{i=1}^{N}$ is not unimodal. Similarly, if we decide to only return the particle with maximum weight to the KF, then we are taking the risk of selecting an outlier. Moreover, in the regular RBPF, the analytical relationship that exists between every particle of $\mathbf{x_1}$ and $\mathbf{x_2}$ (and their history) is only approximated in the modified algorithm. Indeed, at the beginning of the simulation part, each $\mathbf{x_2}_{k-1,i}$ is drawn from a Gaussian distribution instead of being the "exact" match for $\mathbf{x_1}_{k-1,i}$ (i.e., the value that would be returned by a KF iterated on $\mathbf{x_1}_{k-1,i}$). For these reasons, we also expect that the difference of performance between the two algorithms will increase with larger measurement and process noises. Intuitively, since the single estimate of $\mathbf{x_1}_k$ is the only source of information that the KF can rely on, repeated inaccuracies could potentially gear the KF in wrong directions, damaging in turn the quality of estimates of $\mathbf{x_2}_k$.

5. APPLICATIONS AND PERFORMANCE

5.1. A first example: maneuvering target tracking

Consider the problem of tracking a maneuvering target, whose position and velocity at instant k are given by a continuous random vector $\mathbf{x_2}_k \in \mathbb{R}^{n-1}$, and where the maneuver/regime of the target is represented by the discrete random variable $\mathbf{x_1}_k \in \mathbb{R}$. The state to be estimated is $\mathbf{x}_k = \{\mathbf{x_1}_k\ ;\ \mathbf{x_2}_k\}$. The model is as follows:

$$\mathbf{x_2}_k = \mathbf{F}\mathbf{x_2}_{k-1} + \mathbf{B}\mathbf{x_1}_k + \mathbf{w}_k \quad (3)$$

$$\mathbf{z}_k = \mathbf{C}\mathbf{x_2}_k + \mathbf{v}_k \quad (4)$$

Additionally, $p(\mathbf{x_1}_k|\mathbf{x_1}_{k-1})$ is given, $\mathbf{w}$ and $\mathbf{v}$ are zero-mean Gaussian noises, with covariance matrices $\mathbf{Q}$ and $\mathbf{R}$. This problem can be solved using RBPFs, since given $\mathbf{X_1}_k$, the dynamics of $\mathbf{x_2}_k$ are linear-Gaussian. In our model, we use $\mathbf{x_2}_k = [x_k\ y_k\ \dot{x}_k\ \dot{y}_k]^T$ where (x, y) is the position of the target in a cartesian plane. We take:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0.3 & 0 \\ 0 & 1 & 0 & 0.3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 1.25 \\ -1.25 \\ 0.25 \\ -0.25 \end{bmatrix}$$

We only observe the noisy position (x, y) of the object. For the sake of simplicity we suppose that $\mathbf{x_1}_k \in \{-1, 0, 1\}$. Let $p(\mathbf{x_1}_k|\mathbf{x_1}_{k-1}) = 0.8$ if $\mathbf{x_1}_k = \mathbf{x_1}_{k-1}$, and $p(\mathbf{x_1}_k|\mathbf{x_1}_{k-1}) = 0.1$ otherwise. We take $q(\mathbf{x_1}_k|\mathbf{X_1}_{k-1,i}, \mathbf{Z}_k) = p(\mathbf{x_1}_k|\mathbf{x_1}_{k-1})$. The weight update equation is thus $\tilde{w}_{k,i} = p(\mathbf{z}_k|\mathbf{X_1}_{k,i}, \mathbf{Z}_{k-1})$.

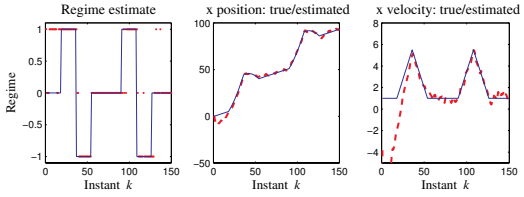
Using the letter $\mathbf{K}$ for error covariance matrices, we can show that:

$$p(\mathbf{z}_k|\mathbf{X_1}_{k,i}, \mathbf{Z}_{k-1}) = \mathcal{N}(\mathbf{z}_k|\mathbf{C}(\mathbf{B}\mathbf{x_1}_{k,i} + \mathbf{F}\mathbf{x_2}_{k-1,i}), \mathbf{T})$$
$$\text{where} \quad \mathbf{T} = \mathbf{R} + \mathbf{C}[\mathbf{Q} + \mathbf{F}\mathbf{K}_{k-1,i}\mathbf{F}^T]\mathbf{C}^T$$

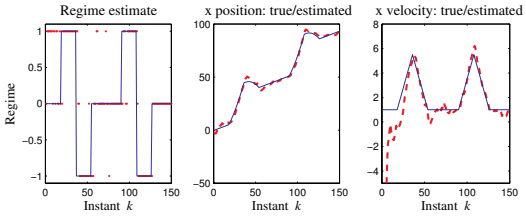
Both the regular and modified RBPF algorithms were implemented in MATLAB. Examples of results are shown on Figure 1. To compare the performance, a range of values for the covariance of $\mathbf{w}$ and $\mathbf{v}$ was defined. For a given pair $\{\sigma_\mathbf{w}, \sigma_\mathbf{v}\}$, where $\mathbf{Q} = \sigma_\mathbf{w}^2\mathbf{I}$ and $\mathbf{R} = \sigma_\mathbf{v}^2\mathbf{I}$, we generated a sequence of true states and measurements shared by both algorithms. Initially the particles for the regular RBPF are drawn from a specific normal distribution (which remains the same throughout all tests), from which the modified RBPF also draws its initial state estimates. To measure the performance, we use the following metric:

- For the sub-state $\mathbf{x_1}_k$, we add the absolute value of the differences between the true and estimated values, and divide the result by the number of observations.
- For the sub-state $\mathbf{x_2}_k$, we compute $\sum_k |\mathbf{x_2}_{k\ \text{(true)}} - \hat{\mathbf{x}}_{\mathbf{2}k}|$, which we then divide elementwise by the empirical standard deviation of the true vector $\mathbf{x_2}_k$. Finally, we add the 4 values obtained.

The average performance computed over 200 experiments for each pair $\{\sigma_\mathbf{w}, \sigma_\mathbf{v}\}$, using 120 particles is shown in Figure 2. Execution time was found to be of the order of 50% higher for the regular RBPF. The modified algorithm offers a performance that is close to the regular algorithm, but with increasing degradation when both noises become large, as conjectured in Section 4.2.



(a) Regular RBPF Algorithm



(b) Modified RBPF Algorithm

Fig. 1. Tracking example results, with $\sigma_\mathbf{v}^2 = 10$ and $\sigma_\mathbf{w}^2 = 0.1$. The estimates are in dotted lines.

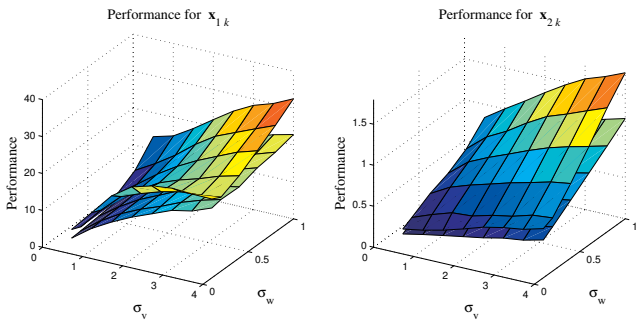


Fig. 2. Tracking example, performance comparison. The performance for the modified algorithm is the upper surface.

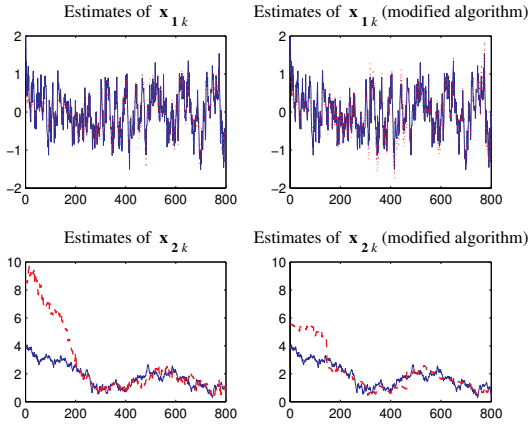


Fig. 3. Signal processing example. The estimates are the dotted lines.

5.2. Second example

We now take an example that can be applied in signal processing. Supposing (without loss of generality) that $\mathbf{x_{1}}_k$ and $\mathbf{x_{2}}_k$ have the same dimension:

$$\mathbf{x_{1}}_k = f(\mathbf{x_{1}}_{k-1}) + \mathbf{w_{1}}_k \quad (5)$$

$$\mathbf{x_{2}}_k = \mathbf{F}_k \mathbf{x_{2}}_{k-1} + \mathbf{w_{2}}_k \quad (6)$$

$$\mathbf{z}_k = {\mathbf{x_{1}}_k}^T \mathbf{x_{2}}_k + \mathbf{v}_k \quad (7)$$

The sub-state $\mathbf{x_{2}}_k$ can be seen as the coefficients of a time-varying filter. Let $\mathbf{w_{1}}_k \sim \mathcal{N}(0, \mathbf{Q_1})$, $\mathbf{w_{2}}_k \sim \mathcal{N}(0, \mathbf{Q_2})$, and $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R})$. We can solve this problem using RBPFs. With $q(\mathbf{x_{1}}_k | \mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) = p(\mathbf{x_{1}}_k | \mathbf{x_{1}}_{k-1})$, we have again $\tilde{w}_{k,i} = p(\mathbf{z}_k | \mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1})$, and we can show that:

$$p(\mathbf{z}_k | \mathbf{X_{1}}_{k,i}, \mathbf{Z}_{k-1}) = \mathcal{N}({\mathbf{x_{1}}_{k,i}}^T \mathbf{F}_k \mathbf{x_{2}}_{k-1,i}, \mathbf{T})$$
$$\text{where } \mathbf{T} = \mathbf{R} + {\mathbf{x_{1}}_{k,i}}^T (\mathbf{Q_2} + \mathbf{F}_k \mathbf{K}_{k-1,i} \mathbf{F}_k^T) \mathbf{x_{1}}_{k,i}$$

We present the simple case where $\{\mathbf{x_{1}}_k, \mathbf{x_{2}}_k\} \in \mathbb{R}^2$, $f() = \arctan()$, and $\mathbf{F}_k = \mathbf{I}$. We thus have an unknown time varying gain applied to an unknown signal, evolving in a known non-linear fashion. An example of results from the implementation of both algorithms is shown in Figure 3. We note again a significant decrease in execution time (of the order of 30%), at the cost of a potential loss in performance, especially in the presence of noises with large amplitude. We also note that some cases have been found to perform seemingly better for both examples, although it is commonly not the case. For instance, observe the specific results of Figure 3: the gain is tracked faster when the modified RBPF algorithm is used.

6. DISCUSSION AND CONCLUSION

We presented in this paper a modification to Rao Blackwellised Particle Filters. From the observation of the algorithm and through two examples, we have found that this alteration constitutes a tradeoff between performance and efficiency. The modified algorithm always executes much faster, but is in counterpart more sensitive to noise. The presented algorithm is a radical simplification, but we are presently considering a more advanced approach, in which a given number of KFs is specified before the algorithm execution. At the end of the simulation part, the estimate for the distribution of $\mathbf{x_{1}}_k$ is regularized and approximated by a mixture of Gaussians. The number of KFs is thus equal to the order of the Gaussian mixture. The computational load is still reduced, and we can expect a better performance than that obtained in the modified algorithm of this paper.

7. REFERENCES

[1] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: particle filters for tracking applications*, Artech House, London, Feb. 2004.

[2] A. Doucet, J.F.G. de Freitas, and N. Gordon, Eds., *Sequential monte carlo methods in practice*, Springer-Verlag, New York, 2001.

[3] J.H. Kotecha and P.M. Djuric, "Gaussian particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2592–601, Oct 2003.

[4] R. van der Merwe, A. Doucet, J.F.G. de Freitas, and E. Wan, "The unscented particle filter," in *Advances in Neural Information Processing Systems 13*, T. K. Leen et. al., Ed. 2001, pp. 584–590, MIT Press.

[5] A. Doucet, J.F.G. de Freitas, K.P. Murphy, and S.J. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 2000, pp. 176–183, Morgan Kaufmann Publishers Inc.