Sequential Monte Carlo Filtering with Long Short-Term Memory Prediction

Steffen Jung
Sensor Data and Information Fusion
Fraunhofer FKIE
Wachtberg, Germany
steffen.jung@fkie.fraunhofer.de

Isabel Schlangen
Sensor Data and Information Fusion
Fraunhofer FKIE
Wachtberg, Germany
isabel.schlangen@fkie.fraunhofer.de

Alexander Charlish
Sensor Data and Information Fusion
Fraunhofer FKIE
Wachtberg, Germany
alexander.charlish@fkie.fraunhofer.de

***Abstract*—Recursive Bayesian estimation builds on the use of suitable mathematical models in order to accurately describe the evolution of an object based on imperfect sensor data. Sequential Monte Carlo (SMC) techniques have been developed to overcome the need of restrictive models, however more sophisticated motion models can require high-dimensional particle states, resulting in a need for many particles and hence greater computation. Additionally, most conventional prediction methods assume low-order Markovian transitions and therefore tend to ignore the majority of the target history that may be relevant to the prediction. In this article, a variation of the Particle Filter (PF) is introduced which uses a Long Short-Term Memory (LSTM) neural network to perform the prediction step. In contrast to existing techniques, the network is trained to predict the relative displacement between object states rather than absolute positions to gain rotation and translation invariance. Experiments on simulated trajectories show that the approach is more robust against low probability of detection and occlusion than the conventional PF.**

***Index Terms*—Sequential Monte Carlo, Bootstrap Filter, Particle Filter, Long Short-Term Memory, Recurrent Neural Network, Target Tracking**

I. Introduction

Conventional Bayesian estimation algorithms depend on two central mathematical models, namely the *motion model* of the target under observation and the *measurement model* which describes the quality of the incoming sensor data. Given that the observed measurements are received at discrete time steps, the Bayesian estimator first predicts where the target might go next based on the present state. This prediction is then corrected using the new measurement and the model of the sensor accuracy. Since the general Bayes recursion is computationally intractable, it is necessary to introduce further assumptions, e.g. using a linear Gaussian model leading to the Kalman filter [1]; however, the method might fail if the true model is far away from the chosen assumption. Small modelling inaccuracies can be compensated up to a certain degree by the introduction of process noise, however high amounts of uncertainty might "blur" the prediction so much that the choice of motion model becomes insignificant.

The introduction of Sequential Monte Carlo (SMC) techniques like the Particle Filter (PF) in the 1990's made it possible to describe arbitrary (and possibly multi-modal) target distributions by approximating them with a set of weighted random samples, called *particles* [2], [3]. Such a formulation also facilitates arbitrarily complex motion models but a sufficient amount of samples is necessary in order to represent the underlying distribution appropriately, especially while working with high-dimensional state spaces. Also, the prediction step in a particle filter usually assumes low-order Markovian transitions and consequently the predicted particle states are only dependent on the particle states at the previous few time steps. Consequently, the estimated state history is normally mostly ignored, even though it may be relevant to the prediction.

It is possible to avoid the formulation of an explicit target motion model altogether if Recurrent Neural Networks (RNNs) [4] are used to perform the prediction. RNNs are especially designed to process sequences of data and to learn an underlying pattern, which fits perfectly to the task of replacing the motion model in the case of Bayesian filtering. Due to the intimate connection between the fields of computer vision and machine learning, the concept of visual tracking has sparked many approaches using neural nets for detecting and tracking objects in image sequences, e.g. [5]–[7], using deep Convolutional Neural Networks (CNNs) and other architectures to train the propagation of bounding boxes in image sequences. Recent literature also developed more general approaches that do not depend on the association of image patches from frame to frame but rather interpret targets as discrete points in a d_x-dimensional space. For example, the approach in [8] first extracts the centroids of some point-like objects and learns the data association across successive frames using an RNN with Long Short-Term Memory (LSTM) [9]. The work in [10] proposes to learn all Kalman filter parameters and therefore relies on an underlying linear Gaussian model. The authors of [11] and [12], on the other hand, replace the use of a Kalman filter or PF completely by proposing end-to-end trackers that train one LSTM for the single-target prediction and one for the update. The approach of [13] goes one step further by training multiple RNNs for object creation and deletion, data association and single-target tracking.

In the fashion of [12], a few other attempts have been made to combine a PF approach with machine learning. In [14], for example, an Expectation-Maximisation (EM) algorithm was combined with an LSTM network using a particle representation of the state, and [15] interprets the particles as the hidden states of the RNN. Note that all these approaches predict absolute positions; such techniques might be useful to learn about absolute traffic routes in a surveillance area, but they are error-prone with respect to rotation and translation.

In contrast to previous methods, this article introduces a PF that predicts the *change* in target motion instead of absolute positions using a pre-trained LSTM network. This approach is especially useful to describe characteristic target behaviour independent of its location in the real world. It is shown in simulation that this technique is preferable to using a conventional PF in complex scenarios since it performs better for a very low probability of detection and especially in the presence of occlusions over longer periods of time. Moreover, while the network needs to be trained beforehand, the computational overhead of using an LSTM prediction on-line is not significant; the improvement in performance even facilitates the use of low-dimensional state spaces which reduces the computational load.

The paper is organised as follows. Sec. II gives a small revision of the used SMC (or PF) method and shows how the LSTM prediction is incorporated in this framework. The resulting pseudo-code is shown in Sec. III and the method is demonstrated and evaluated in simulation in Sec. IV. The paper concludes in Sec. V.

II. A PF WITH A LEARNED MOTION MODEL

In the following, let $x_t \in \mathbb{R}^{d_x}$ be the d_x-dimensional state of an object at time $t \in \mathbb{N}$. A time series between time steps $t \leq u \in \mathbb{N}$ is written as $x_{t:u} = (x_t, x_{t+1}, \ldots, x_{u-1}, x_u)$; similarly, we write $\mathrm{d}x_{t:u} = \mathrm{d}x_t \mathrm{d}x_{t+1} \ldots \mathrm{d}x_u$. Measurements are represented by the d_z-dimensional vector $z_t \in \mathbb{R}^{d_z}$, and denote by $Z^t = z_{0:t}$ the collection of all measurements received up to time t. The current state estimate will be denoted by $x_t^\star$, which is chosen to be the weighted average of the particle positions in this article.[1]

First, let us revise the concept of SMC methods for filtering before describing the incorporation of an LSTM network into the algorithm.

A. Sequential Monte Carlo filtering

The general Bayes recursion consists of a prediction stage that estimates the current state of an object based on its history and a correction stage that incorporates a current measurement to validate the predicted belief. One possible formulation is as follows:

$$p_{t|0:t-1}(x_t|Z^{t-1}) \\ = \int f_{t|0:t-1}(x_t|x_{0:t-1}, Z^{t-1}) p_{t-1}(x_{0:t-1}|Z^{t-1}) \mathrm{d}x_{0:t-1} \tag{1a}$$

$$\approx \int f_{t|0:t-1}(x_t|x_{0:t-1}, Z^{t-1}) p_{t-1}(x_{t-1}|Z^{t-1}) \mathrm{d}x_{0:t-1}, \tag{1b}$$

$$p_t(x_t|Z^t) = \frac{g_t(x_t|z_t) p_{t|0:t-1}(x_t|Z^{t-1})}{\int g_t(x|z_t) p_{t|0:t-1}(x|Z^{t-1}) \mathrm{d}x}. \tag{1c}$$

Here, $p_{t|0:t-1}$ is the predicted probability distribution with $f_{t|0:t-1}$ being the time transition function to time t given a history of states $x_{0:t-1}$. Furthermore, p_{t-1} and p_t denote the prior and posterior single-target distributions at time t, respectively, and $g_t(x|z)$ is the (possibly time-dependent) association likelihood of x given z. Note that in Eq. (1b) it is implicitly assumed that the prior is constant with respect to the past states $x_{0:t-2}$ so that the recursion can be closed.

Since the Bayes recursion is not tractable in its most general form, it is necessary to make certain additional assumptions (like that in Eq. (1b)) to retrieve a more specific but computationally feasible algorithm. A widely used special case is the well-known Kalman filter [1] which describes the target state and the association likelihood with Gaussian distributions and uses a linear, Markovian transition function; under these restrictive assumptions, it is an optimal solution, however many common applications are highly non-linear and/or non-Gaussian. For mildly non-linear problems the extended and unscented Kalman filters were introduced which approximate the transition function through linearisation or the propagation of sigma points [16].

[1]Alternatively, the Maximum A Posteriori (MAP) estimate could be used instead which is the particle position with the highest weight.

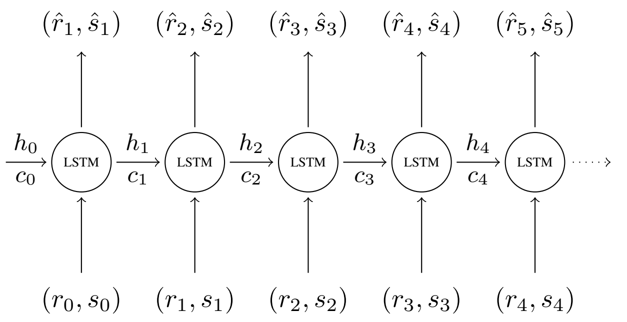


Fig. 1: Graphical representation of the LSTM architecture used for learning the target motion. The h_i and c_i represent the hidden states of the LSTM network and each LSTM node is implemented according to [9]. Instead of predicting the absolute target states, the network is trained to predict the current rotation and speed.

The advancements in computational power since the 1990s facilitated the development of SMC techniques that propagate the state distribution through time [2] using a set of N weighted samples with states $x_t^{(i)}$ and weights $w_t^{(i)}$ for $i \in \{1, \ldots, N\}$ as follows:

$$p(x_t|Z^t) = \sum_{i=1}^{N} w_t^{(i)} \delta\left(x_t - x_t^{(i)}\right) \tag{2}$$

with δ denoting the Dirac delta function. These weighted samples are commonly called *particles*. The accuracy of SMC filters increases with increasing numbers of particles. It has the advantage that it is not dependent on specific forms of the underlying distribution or the type of transition function.

B. Learning the motion model: an LSTM approach

A common choice for the time transition function $f_{t|0:t-1}$ is a first-order Markovian model which forgets about the target history $x_{0:t-2}$, i.e. $f_{t|0:t-1}(x_t|x_{0:t-1}) = f_{t|t-1}(x_t|x_{t-1})$. While this approximation reduces the complexity of the algorithm in the classical formulation by removing the integrals with respect to $x_{0:t-2}$, it might also come with a loss of accuracy since temporal context information can help identify characteristic target behaviour, especially in the presence of occlusions.

To alleviate the restrictiveness of the Markov assumption, we propose to use an LSTM network [9] for the motion prediction step. RNNs – especially those with Long Short-Term Memory – are designed to process temporal sequences of data while incorporating context information about previously seen measurements. This idea can be especially useful in cases where all objects in the field of view follow

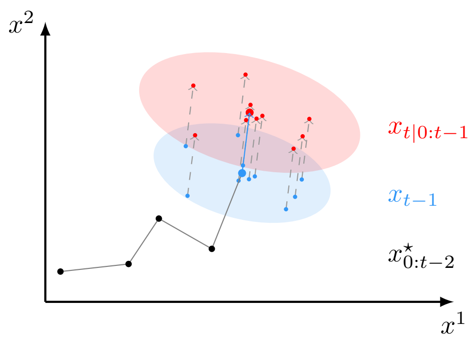


Fig. 2: Schematic concept of the proposed prediction. First, the next mean/MAP particle state (●) is predicted based on the history of state estimates (●—●—●). Then, the predicted motion is applied equally to all prior states (·) and white noise is added to the predicted states (·), thus increasing the spread of the particle cloud.

a common behavioural pattern. Moreover, in contrast to previous methods, the proposed predictor does not predict the absolute target position but rather the *displacement* (r_t, s_t), i.e. the rotation r_t between the past and current object heading and the current speed s_t which is the length of the displacement vector.

To get a relative time series of data (r_t, s_t) from a time series $x_{0:T}$ of absolute target positions $x_t = (x_t^1, x_t^2) \in \mathbb{R}^2$, the initial rotation and speed are first set to $r_0 = 0\,\mathrm{rad}$ and $s_0 = 0\,\mathrm{m\,s^{-1}}$, respectively. While iterating over the absolute positions, the velocity vector v_t between x_{t-1} and x_t is calculated as

$$v_t = x_t - x_{t-1}, \tag{3}$$

which defines a speed via $s_t = \|v_t\|$ and a course via $c_t = \operatorname{atan2}(v_t^2, v_t^1)$. With these, it is possible to calculate a rotation $r_t = c_t - c_{t-1}$ as long as there is a previous course c_{t-1}, i.e. if $v_{t-1} \neq (0, 0)$.

As shown in Fig. 1, the network is trained on previously recorded trajectories from which it predicts a tentative current target rotation $\hat{r}_t$ and speed $\hat{s}_t$ based on the previous rotation r_{t-1} and speed s_{t-1}. The loss is calculated from the distance between the tentative displacement $(\hat{r}_t, \hat{s}_t)$ and the true displacement (r_t, s_t) obtained from the ground truth. To avoid over-fitting, learning is performed using dropout, meaning that the input activations are individually dropped with a common dropout probability during training.

Once the model is trained, the LSTM predictor is applied to the trajectory formed by the respective average prior particles $x^\star_{0:t-1}$; the obtained predicted tuple $(\hat{r}_t, \hat{s}_t)$ at time t is then applied equally to *all* particle states $x^{(i)}_{t-1}$ to produce the predicted particle positions $x^{(i)}_{t|0:t-1}$. Additional white noise with standard deviation σ_p is added to account for errors in the prediction. The full prediction mechanism is illustrated in Fig. 2.

C. Update and resampling

Since the LSTM network is only concerned with the prediction of the target state in the proposed architecture, the update and resampling steps of the filter are chosen according to a common SMC approach like the bootstrap filter [17]. On receiving a new measurement z_t, the particle update is performed using the weight update

$$w_t^{(i)} = \frac{w_{t-1}^{(i)} g_t(x^{(i)}_{t|0:t-1} | z_t)}{\sum_{j=1}^{N} w_{t-1}^{(j)} g_t(x^{(j)}_{t|0:t-1} | z_t)} \tag{4}$$

according to Eq. (1c) with association likelihood g_t, which incorporates white measurement noise with standard deviation σ_m. If the effective sample size $N_\text{eff} = \left(\sum_i (w_t^{(i)})^2\right)^{-1}$ falls below a certain threshold, e.g. $\frac{N}{2}$, the samples are rearranged according to their weights using a low-variance sampler and each weight is reset to N^{-1} [18].

III. Implementation

The pseudo-code for the LSTM PF is given in Algorithm 1. Initialisation is implemented by drawing N random samples with equal weights N^{-1} from a Gaussian distribution centred around an initial position $x^\star_0$ and a covariance Σ_0. The process noise is modelled as white noise with standard deviation σ_p in each state dimension.

In contrast to conventional filtering with a fixed model, the proposed algorithm requires prior training of the neural network. However this is performed only once and does not affect the on-line computation time of the tracking algorithm. The training time is highly dependent on different factors such as the amount of training data, the available hardware resources (i.e. Graphics Processing Units (GPUs) are faster than Central Processing Units (CPUs)), the complexity of the target movement, etc. For the experiments below, the network was trained on a conventional CPU with a small amount of training data representing a target with relatively simple sinusoidal motion model. In this case, the training time was of the order of hours.

```
Input: $N$ particles with weights $N^{-1}$;
while tracking do
    LSTM prediction:
    Predict common displacement;
    foreach particle do
        Add displacement and noise to state vector;
    end
    Update:
    if measurement received then
        foreach particle do
            Update weight using (4);
        end
    end
    Current estimate = weighted average of particles;
    Resampling:
    if effective sample size too small then
        Resample particles using [18];
    end
end
```

Algorithm 1: Pseudo-code for the proposed SMC filter with an LSTM predictor.

It can be seen from Algorithm 1 that the on-line computational complexity of the filter is $\mathcal{O}(N + H)$, where H denotes the number of hidden units in the LSTM network. In other words, it differs from a filter with standard prediction only by $\mathcal{O}(H)$ [9].

IV. Evaluation

To evaluate the proposed LSTM predictor we compare its tracking performance to that of a standard PF with a Brownian motion model in a set of simulated experiments. In the following, all estimation errors are computed in terms of the mean (and possibly the variance) of the errors between each particle and the ground truth.

A. LSTM training

The LSTM predictor was first trained on an i7-6700 CPU with the Java deep learning framework *deeplearning4j* with an initial learning rate of 10^{-5} and using 25 hidden units for 4500 epochs.[2] The true trajectory $x_{0:T}$ is defined to be a scaled sine curve with 25 periods given by the two-dimensional states

$$x_t = \begin{pmatrix} t \cdot \Delta \\ \sin(t \cdot \Delta) \end{pmatrix}, \tag{5}$$

where Δ is the step size in the direction of the first dimension (which will be denoted by x^1). For all following experiments, a step size of $\Delta = 0.2$ is chosen for consistency. Note that all standard deviations of the measurement and process noises σ_m and σ_p in this paper are given in multiples of the step size Δ.

A sine curve was chosen for the subsequent experiments since it consists of near-constant velocity segments and segments of high curvature which makes it an interesting case study. It is also important to note that the network was trained on the true trajectory without any noise but using dropout with a probability of 0.25.

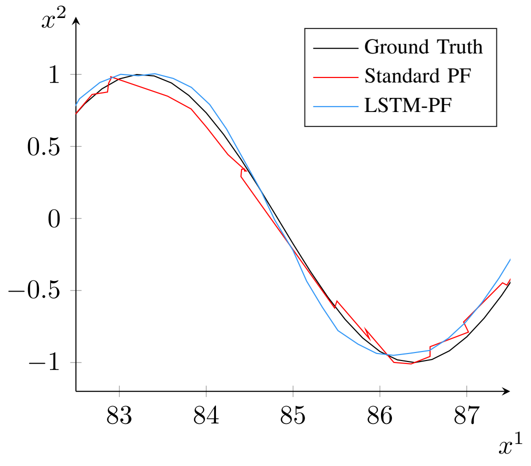
B. Experiments

In the following five experiments, the PF with LSTM prediction model is compared against a standard PF with Brownian motion model using a standard deviation of $1.0 \cdot \Delta$. The standard PF uses the exact same implementation as in Algorithm 1 but with a Brownian prediction model instead of the LSTM predictor. Note that a Brownian model was used to keep a state space of dimension 2; for more complex motion models like near-constant velocity, the filter would need twice as many dimensions which would increase the necessary amount of particles significantly. Also note that all figures showing actual tracks are in the two-dimensional Euclidean state space $\mathbb{R}^2$ with coordinates x^1 and x^2.

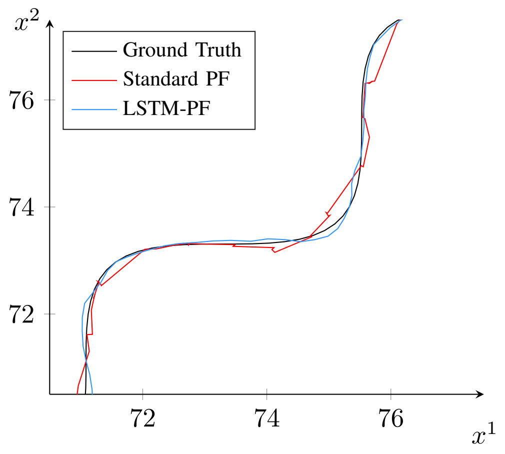
The following parameters are analysed:

- *Number of particles, N*: It shall be analysed how the number of samples influences the tracking results using a standard or LSTM-based prediction.
- *Probability of detection, p_d*: In most realistic scenarios, the sensor might miss measurements from time to time. It will be shown how low detection rates affect the performance of the proposed method, and moreover how the algorithm deals with occlusions in the data.
- *Process noise, σ_p*: Additional noise is used to account for imperfections in the prediction step. It will be examined how much process noise is needed for both filter versions.

[2]For documentation see deeplearning4j.org.



(a) Track of a sine curve without rotation.

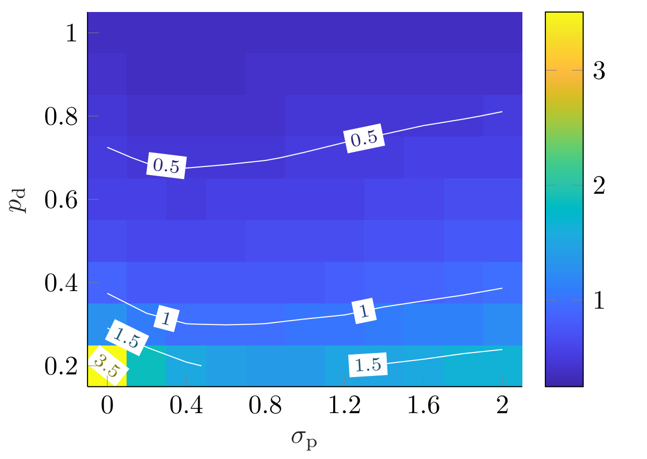


(b) Track of a sine curve rotated around the origin by 45°.

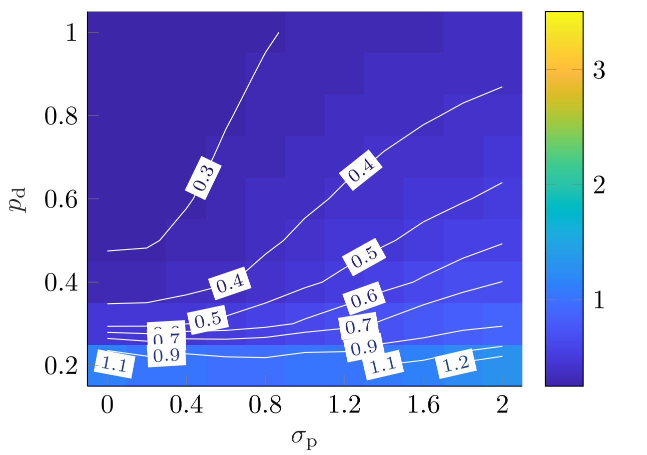
Fig. 3: Details of example tracks estimated by the PFs with standard and LSTM predictor using 100 particles, detection rate $p_\mathrm{d} = 0.5$ and $\sigma_\mathrm{p} = \sigma_\mathrm{m} = 0.5$.

1) Experiment 1: To give an intuition about the track quality obtained by the PF with LSTM-based prediction, let us first consider the estimated trajectories on a standard sine curve and one rotated by 45°, see Fig. 3. For this experiment, the filters were initialised with $N = 100$ particles and $\sigma_\mathrm{p} = \sigma_\mathrm{m} = 0.5$, while measurements were provided with a probability $p_\mathrm{d} = 0.5$. The plots demonstrate the rotation invariance of the approach since the proposed method produces smooth trajectories close to the ground truth in both cases. The standard filter yields noisier trajectories due to the underlying Brownian motion and the low detection rate.

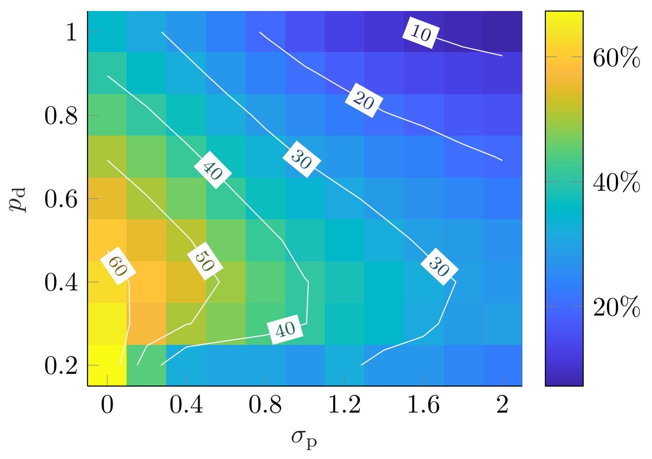
2) Experiment 2: This experiment analyses the performance of both methods with regard to different probabilities of detection and process noise, using $\sigma_\mathrm{m} = 0.5$ and 100 particles. Fig. 4 shows the results of the two filters in terms of the estimation error averaged over all time steps and across 100 Monte Carlo (MC) runs. As Fig. 4b shows, the proposed method can cope better with low probabilities of detection while needing just a small process noise, whereas the standard filter (Fig. 4a) works best for high detection rates and medium process noise. In terms of the performance gain, it can be seen that the proposed method outperforms the standard PF by more than 60 % for low process noise and detection rate; this shows that using an LSTM-based



(a) Standard predictor.



(b) LSTM predictor.



(c) Percentage of accuracy gained from using a learned motion model over a standard predictor.

Fig. 4: Errors for Experiment IV-B2 with varying detection rates and process noise, averaged over 100 MC runs. The measurement noise has a standard deviation $\sigma_m = 0.5$ and the number of particles is 100.

prediction gives an especially high performance gain if the probability of detection is low, however it always performs better than the standard filter. Note that the performance of both filters degrades for increasing process noise since high noise makes the influence of the motion model less significant.

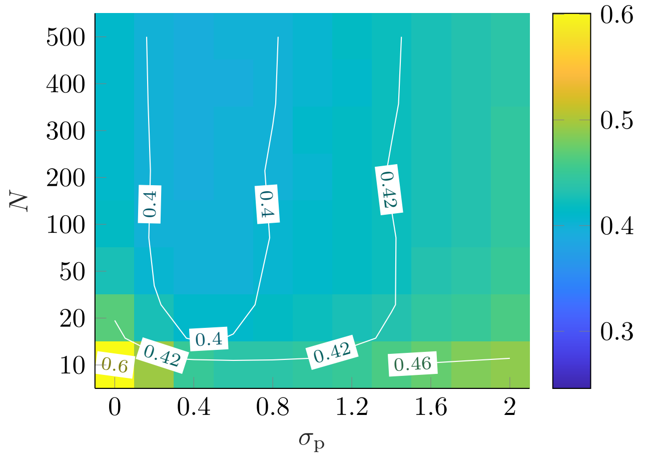
3) Experiment 3: The third experiment compares different combinations of process noise and numbers of particles, assuming $p_d = 0.9$ and $\sigma_m = 0.5$. Results can be seen in Fig. 5, again plotted in terms of the average errors over time and 100 MC runs. Similarly to section IV-B2, it can be seen that high amounts of process noise deteriorate the tracking performance of both filters, still the proposed LSTM-PF consistently produces lower errors than the standard PF irrespective of σ_p or N. Moreover, it can be seen in Fig. 5c that the performance gain does not depend much on the number of particles. It could be argued that more sophisticated motion models should be used to improve the performance of the standard PF, however this would require the use of a higher-dimensional state space and therefore necessarily a higher amount of particles.

4) Experiment 4: This experiment is intended to show the robustness of the proposed LSTM-based method against target occlusion. Instead of omitting targets according to a given detection probability, all measurements are obtained for the first 100 iterations; then, the target is miss-detected for n consecutive iterations before receiving new measurements.

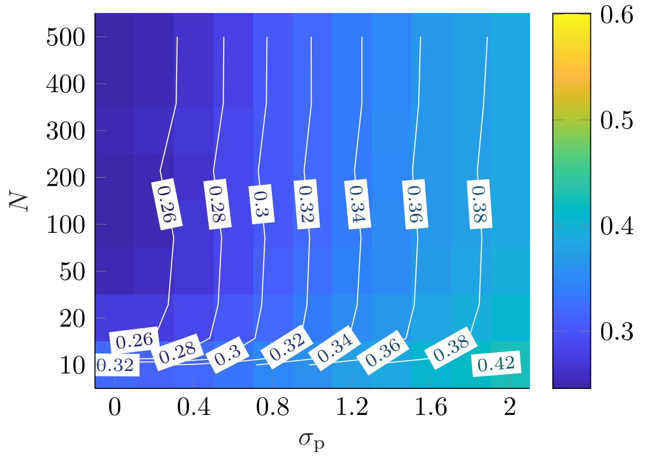
Fig. 6a displays the tracking results in the presence of an occlusion interval of length $n = 25$. It can be seen that the standard PF appears to stop at the location of the last measurement since the Brownian motion forces the particles to spread without moving the average particle position too much. This can also be observed in the increased variance in Fig. 6b. Only after the next measurement is received, the filter is able to jump back onto the true trajectory. The LSTM-based filter, on the other hand, predicts a smooth sine curve based on its learned model and hence draws a continuous trajectory without any major jumps.

Fig. 7a displays the estimation error for different occlusion intervals, averaged over time and 100 MC runs. The detection rate outside the occlusions is $p_d = 1.0$ and the standard deviations of the measurement and process noises are $\sigma_m = \sigma_p = 0.5$. We define a track as lost if the average estimation error exceeds 5; all lost tracks are omitted from the estimation plot in Fig. 7a. The plot shows that for occlusions longer than 80 time steps, the PF with a standard motion model cannot recover the true trajectory in any of the MC runs, whereas the proposed method still produces valid tracks and increases in error more slowly. This stems from the fact that the LSTM may be accumulating a bias over long periods without measurements, but its predictions still help recover the true trajectory more easily once new measurements are received. Also note the steep climb in the rate of tracks lost for the standard PF when increasing the occlusion lengths. Compared to the standard PF, the LSTM-PF also loses tracks less often and the loss rate climbs shallower.

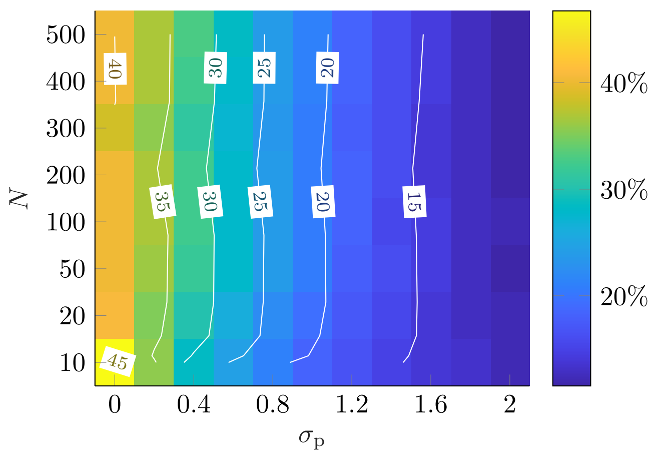
5) Experiment 5: The final experiment can be seen as a general discussion on the proposed filter's limitations. As with any motion model, the trained LSTM's performance is limited by the motion it describes. This requires a set of training data which represents the underlying motion of the target sufficiently. Therefore the same LSTM as above was also tested on a piecewise linear trajectory, as shown in Fig. 8. It can be observed that the sinusoidal predictions are negatively impacting the overall tracking performance, especially when the target is miss-detected. This shows



(a) Standard predictor.

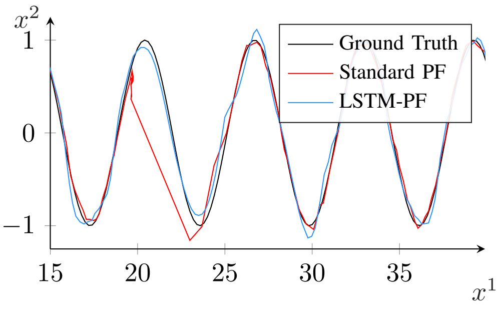


(b) LSTM predictor.

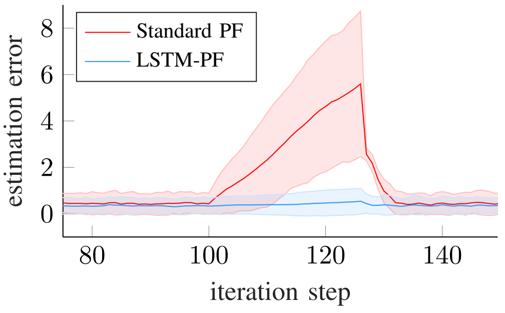


(c) Percentage of accuracy gained from using a learned motion model over a standard predictor.

Fig. 5: Errors for Experiment IV-B3 with varying process noise and numbers of particles, averaged over 100 MC runs. The detection rate is $p_\mathrm{d} = 0.9$ and the standard deviation of the measurement noise is $\sigma_\mathrm{m} = 0.5$.

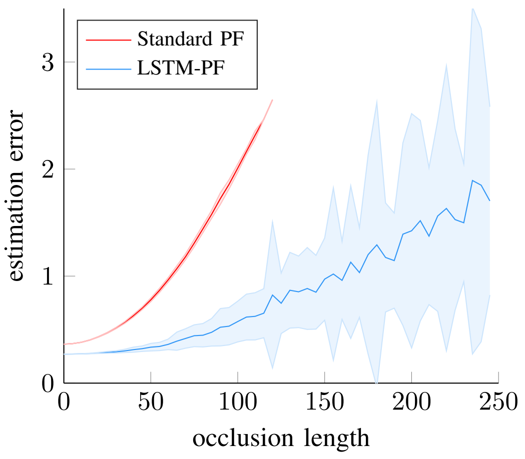


(a) After 100 iterations (corresponding to $x = 20$), the object is miss-detected for 25 consecutive time steps.

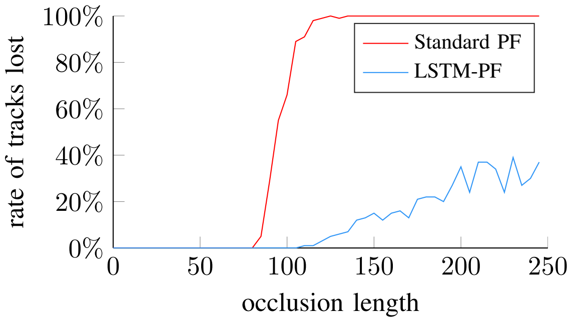


(b) Average distance between particle states and the ground truth (with variance over all particles) for the experiment in Fig. 6a.

Fig. 6: Experiment IV-B4: example of tracking an occluded target.



(a) Average estimation error for different occlusion lengths over 100 MC runs.



(b) Percent of tracks lost for different occlusion lengths over 100 MC runs.

Fig. 7: Experiment IV-B4, showing the effect of occlusions of different length.

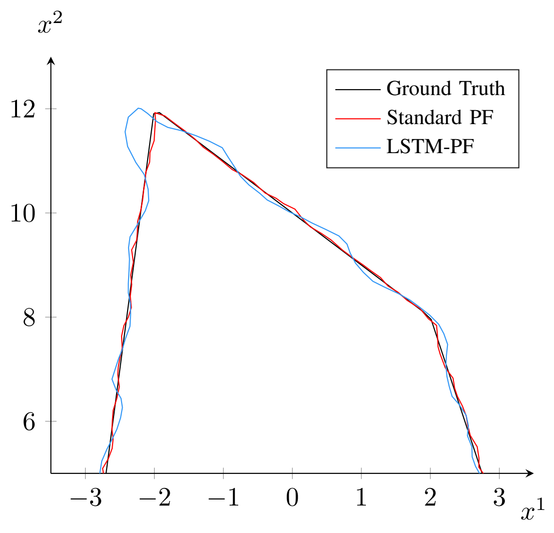


Fig. 8: Tracking a target on a piecewise linear trajectory with an LSTM trained on a sinusoidal motion. The step size was kept at $\Delta = 0.2$ for the linear motion, while $\sigma_\mathrm{m} = \sigma_\mathrm{p} = 0.5$ and a probability of detection of $p_\mathrm{d} = 0.5$.

that the ability of an LSTM predictor to extrapolate from its motion model fundamentally depends on what it has learned. However, such difficulties could be overcome by incorporating multiple motion models.

V. Conclusion

This paper presented a Particle Filter that uses a pre-trained Recurrent Neural Network with Long Short-Term Memory to predict the current target motion, i.e. the displacement between the prior and the predicted object states, in terms of rotation and speed. The computational overhead by the proposed LSTM predictor is insignificant since it does not depend on the number of particles but only on the hidden states of the network. The demonstrated experiments show that this method has a clear advantage to standard low-order Markovian prediction models (using the same state space dimension) in the presence of occlusions or very low detection rates since it includes the full state history into the time update. Furthermore, since the filter predicts relative displacement instead of absolute positions, it is invariant to rotation and translation and therefore more versatile than absolute predictors. This shows that the presented machine learning filter facilitates the use of computationally efficient low-dimensional state spaces while achieving a considerable improvement in tracking accuracy. Further work will explore the proposed filter's performance on real world data and improve upon the current neural network architecture.

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[2] A. Doucet, N. De Freitas, and N. Gordon, "An introduction to sequential Monte Carlo methods," in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 3–14.

[3] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[5] J. Son, M. Baek, M. Cho, and B. Han, "Multi-object tracking with quadruplet convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5620–5629.

[6] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 366–382.

[7] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran, "Detect-and-track: Efficient pose estimation in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 350–359.

[8] Y. Yao, I. Smal, and E. Meijering, "Deep neural networks for data association in particle tracking," in *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*. IEEE, 2018, pp. 458–461.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari, "Long short-term memory kalman filters: Recurrent neural estimators for pose regularization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5524–5532.

[11] D. Iter, J. Kuck, and P. Zhuang, "Target tracking with Kalman filtering, KNN and LSTMs," 2016.

[12] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," *arXiv preprint arXiv:1805.11122*, 2018.

[13] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks." in *AAAI*, vol. 2, 2017, p. 4.

[14] X. Zheng, M. Zaheer, A. Ahmed, Y. Wang, E. P. Xing, and A. J. Smola, "State space LSTM models with particle MCMC inference," *arXiv preprint arXiv:1711.11179*, 2017.

[15] Y. J. Choe, J. C. Shin, and N. A. Spencer, "Probabilistic interpretations of recurrent neural networks," 2017.

[16] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068. International Society for Optics and Photonics, 1997, pp. 182–194.

[17] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings F-radar and signal processing*, vol. 140, no. 2. IET, 1993, pp. 107–113.

[18] S. Thrun, W. Burgard, D. Fox *et al.*, *Probabilistic robotics*. MIT press Cambridge, 2005, vol. 1.