A NON-PARAMETRIC METHOD BASED ON NEURAL NETWORKS AND PARTICLE FILTERING FOR CAMERA LENS DISTORTION ESTIMATION

Younes Boutiyarzist[1,2,3], *Sinan Yildirim*[4], *Jean-Yves Tourneret*[2,5], *François Vincent*[1], *Cheng Cheng*[6] *and Philippe Salmon*[3]

[1] ISAE-SUPAERO, Univ. Toulouse, Toulouse, France. firstname.lastname@isae-supaero.fr
[2] TéSA laboratory, Toulouse, France. firstname.lastname@tesa.prd.fr
[3] Collins Aerospace, Blagnac, France. firstname.lastname@collins.com
[4] Sabancı University, Istanbul, Turkey. sinanyildirim@sabanciuniv.edu
[5] IRIT-ENSEEIHT, Univ. Toulouse, Toulouse, France. firstname.lastname@toulouse-inp.fr
[6] Northwestern Polytechnical University, Xi'an, China. cheng.cheng@nwpu.edu.cn

ABSTRACT

A non-parametric method is introduced to estimate the measurement model of dynamical systems. The method uses a neural network trained in an unsupervised manner integrated into a particle filter framework. The network learns the measurement likelihood directly from the distribution of particles. The performance of the resulting neural network particle filter is first evaluated on synthetic data with a known measurement model showing a very interesting performance. The particle filter is then applied to sensor calibration with a specific focus on camera distortion estimation. Experimental results show that the method provides a reliable alternative to traditional parametric calibration techniques.

***Index Terms*—** Tracking, particle filter, neural networks, camera distortion estimation, pinhole model.

1. INTRODUCTION

Sequential Monte Carlo methods and particle filters (PFs) are widely used for state estimation in complex systems. Non-linear and non-Gaussian dynamics are handled by propagating a set of weighted particles that approximate the posterior state distribution over time [1]. Many state-estimation problems with non-linear state or/and measurement dynamics and non-Gaussian noises can be solved using PFs [2]. However, the performance of PFs strongly depends on the accuracy of the measurement model, which defines how the system's state relates to the observations. In many real-world scenarios, this model is partially known or completely unknown, which complicates the filtering process. These scenarios include inertial sensor calibration [3], air-quality sensor calibration [4], and indoor localisation [5].

The measurement model used for cameras can be difficult to define. Indeed, real lenses bend light unevenly across the field of view and small sensor misalignments introduce additional non-linearities, causing three-dimensional points to deviate from the ideal pinhole projection onto the image plane, making the observation model difficult to express analytically [6]. Traditional calibration methods take into account these distortions by using fixed parametric models such as radial–tangential or Brown–Conrady formulations [7]. However, they require large sets of labelled calibration images captured from multiple viewpoints and the estimated parameters may become invalid after lens refocusing, temperature changes, or camera replacement, which raises problems for real-world deployments [8].

When the state or/and measurement models are only partially known, typical approaches rely on parametric approximations and update the unknown parameters during inference. Two notable strategies in this context are online expectation-maximisation (EM) [9] and score-based gradient methods [10, 11]. The EM algorithm alternates between estimating latent states and updating model parameters. Gradient methods compute the gradient of the marginal log-likelihood with respect to parameters using PF approximations of the score.

More recent research includes differentiable PFs, which treat the entire pipeline as a differentiable computation graph: NNs are used to define a proposal distribution, the measurement likelihood, or the resampling weights, and parameters are learned directly from data through back-propagation [12–14]. However, many differentiable PF frameworks are first trained offline using supervised learning. This stage relies on ground-truth hidden states and the learned parameters are then fixed while the model performs online inference on new data streams [15].

This work concerns unsupervised and non-parametric learning of the measurement equation of a dynamical system by modeling the equation with an NN, with a particular interest in the problem of camera-lens distortion estimation. Recently, PF-based techniques [10] that exploit Fisher's identity have been introduced to obtain estimates for the score function, which is the gradient of the log-likelihood in a state-space model with respect to the model parameters [16]. This, in turn, can be used to build a gradient-based algorithm to estimate the parameters of the NN. This work pursues a similar line to develop an unsupervised framework for camera-lens distortion estimation by integrating a PF and an NN. Since the NN is "trained" directly using the particles within the PF, this approach removes the need for building ground-truth state vectors. Specifically, NN is employed to learn the *residual* between the nominal (distortion-free) and true (distorted) measurement models directly from the particle set. Its parameters are updated via a score-based estimate of the log-likelihood gradient [10]. This strategy removes the need for an explicit analytical distortion model, which can be interesting for practical applications.

The paper is structured as follows: The proposed method is studied in Section 2. Simulation results are presented in Section 3 and conclusions are provided in Section 4.

2. METHODOLOGY

2.1. The model: A state-space model with an NN

A discrete-time nonlinear state-space model is defined by a hidden state process $\{x_t \in \mathbb{R}^{d_x}\}_{t\geq 0}$ and an observation process $\{z_t \in \mathbb{R}^{d_z}\}_{t\geq 1}$ for some $d_x, d_z \geq 1$. At time $t \geq 1$, the system evolves according to the following state and measurement equations:

$$x_t = f(x_{t-1}) + u_t, \tag{1}$$

$$z_t = h(x_t) + e_t, \tag{2}$$

where $u_t \in \mathbb{R}^{d_x}$ and $e_t \in \mathbb{R}^{d_z}$ are random processes and measurement noises with zero mean and known covariance matrices. The functions $f : \mathbb{R}^{d_x} \mapsto \mathbb{R}^{d_x}$ and $h : \mathbb{R}^{d_x} \mapsto \mathbb{R}^{d_z}$ represent the (possibly) nonlinear transition dynamics and the unknown measurement model. For $t \geq 1$, the joint probability distribution of the states $x_{0:t} = (x_0, ..., x_t)^\top$ and the observations $z_{1:t} = (z_1, ..., z_t)^\top$ is:

$$p\left(x_{0:t}, z_{1:t}\right) = p(x_0) \prod_{k=1}^{t} p\left(x_k \mid x_{k-1}\right) \prod_{k=1}^{t} p\left(z_k \mid x_k\right), \tag{3}$$

where $p\left(x_t \mid x_{t-1}\right)$ and $p\left(z_t \mid x_t\right)$ are the transition and observation densities resulting from (1) and (2) and $p(x_0)$ is the initial distribution for x_0.

In many real-world applications, while the state transition function f can be defined with a reasonable accuracy, the measurement function h may be partially known or entirely unknown, rendering the use of a fixed likelihood model inadequate. In such cases, we propose to replace h in (2) with a neural network h_θ, leading to the following measurement model:

$$z_t = h_\theta(x_t) + e_t, \tag{4}$$

where the vector $\theta \in \Theta$, for some suitable Θ, contains the parameters of the NN that approximates the measurement function. This choice is motivated by the fact that NNs such as multi-layer perceptrons can serve as universal functional approximations modelling complex non-linear relationships in data. Once modelled as such, "learning" h_θ from a given set of observations $z_{1:T}$ until time $T \geq 1$ can be formulated as a maximum likelihood problem for θ, in which one maximizes the marginal (log-)likelihood of the observations:

$$\theta^* = \arg\max_{\theta\in\Theta} \log p_\theta\left(z_{1:T}\right).$$

The parameter θ^* is found with gradient ascent, which updates the estimate of θ at the jth iteration as:

$$\theta_{j+1} = \theta_j + \eta \nabla \log p_{\theta_j}\left(z_{1:T}\right), \tag{5}$$

where η is the learning rate and the superscript $j = 0, \ldots, n$ indicates the training iteration.

2.2. Particle filtering (PF)

The learning problem addressed in this work involves the calculation of the posterior distribution $p(x_{0:t} \mid z_{1:t})$ sequentially. However, in many practical problems, where the state-space dynamics are nonlinear and/or non-Gaussian, the joint distribution (3) cannot be marginalized in closed form,making $p(x_{0:t} \mid z_{1:t})$ intractable. To address this difficulty, PFs are widely used to approximate the posterior using a set of $N > 1$ weighted particles as:

$$p(x_{0:t}|z_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta(x_{0:t} - \mathbf{x}_t^{(i)}), \tag{6}$$

where $\delta(\cdot)$ is the Dirac delta function, $\mathbf{x}_t^{(i)} := [x_{t,0}^{(i)}, x_{t,1}^{(i)}, \ldots, x_{t,t}^{(i)}]$ is the i-th *path particle* at time t, and $w_t^{(i)}$ is its normalized weight computed using the observation density.

2.3. Neural Network Particle Filter Approximation

For a given $\theta \in \Theta$, the gradient of the log-likelihood $\nabla \log p_{\theta_j}\left(z_{1:T}\right)$, which is also known as the *score* vector, cannot be calculated exactly, for the same reasons that make the posterior distribution intractable. To approximate the score vector with PF, the methodology in [10] is used, which is based on the Fisher's identity:

$$\nabla \log p_\theta\left(z_{1:T}\right) = \int \nabla \log p_\theta\left(x_{0:T}, z_{1:T}\right) p_\theta\left(x_{0:T} \mid z_{1:T}\right) dx_{0:T}. \tag{7}$$

Since only the measurement function (4) depends on θ, one has:

$$\nabla_\theta \log p_\theta(x_{0:T}, z_{1:T}) = \sum_{t=1}^{T} \nabla_\theta \log p_\theta(z_t \mid x_t). \tag{8}$$

Substituting (8) into (7), and using the PF approximation (6) for $p_\theta\left(x_{0:T} \mid z_{1:T}\right)$ leads to:

$$\nabla \log p_\theta\left(z_{1:T}\right) \approx \sum_{i=1}^{N} w_T^{(i)} \sum_{t=1}^{T} \nabla \log p_\theta\left(z_t \mid x_{T,t}^{(i)}\right), \tag{9}$$

where it is recalled that $x_{T,t}^{(i)}$ is the "time t" component of the ith path particle at time T. The methodology in [10] is also used in [16] to approximate the score function for a state-space model involving NN, however, with modifications toward fixed-lag smoothing. Unlike [16], a direct application of the "$\mathcal{O}(N)$" particle path approximation of the Fisher identity is considered in this work.

The approximation of $\nabla \log p_\theta\left(z_{1:T}\right)$ in (9) can be calculated for each $T \geq 1$ recursively, in an algorithm that integrates PF with gradient calculations for NN. The algorithm operates through three main stages at each iteration: the prediction step uses the motion model, the update step incorporates the measurement model, and the resampling step maintains particle diversity. For recursive computation of the score function, each particle path $\mathbf{x}_t^{(i)}$ is assigned a gradient vector $\alpha_t^{(i)}$ defined as the particle's local gradient:

$$\alpha_t^{(i)} := \sum_{k=1}^{t} \nabla_{\theta_j} \log p_{\theta_j}(z_k \mid x_{t,k}^{(i)}).$$

In the prediction step, particles are propagated according to the system dynamics (1):

$$x_t^{(i)} \sim p(x_t \mid x_{t,t-1}^{(i)}), \quad \mathbf{x}_t^{(i)} = [\mathbf{x}_{t-1}^{(i)}, x_t^{(i)}], \tag{10}$$

so that $x_{t,t}^{(i)} := x_t^{(i)}$ for $i = 1, \ldots, N$. After receiving the observation z_t, the particle weights are calculated according to (2) as:

$$w_t^{(i)} \propto p_{\theta_j}(z_t|x_t^{(i)}). \tag{11}$$

This likelihood depends on $h_{\theta_j}(x_t^{(i)})$, i.e., the NN output obtained from the forward pass of particle $x_t^{(i)}$ [17]. Each vector $\alpha_t^{(i)}$ accumulates local gradient contributions and is recursively updated as:

$$\alpha_t^{(i)} = \alpha_{t-1}^{(i)} + \nabla \log p_{\theta_j}(z_t|x_t^{(i)}). \tag{12}$$

Assuming e_t is zero-mean Gaussian noise with covariance matrix Σ_e, the gradient is given by

$$\nabla \log p_{\theta_j}(z_t \mid x_t^{(i)}) = \Sigma_e^{-1} \left(z_t - h_{\theta_j}(x_t^{(i)}) \right) \odot \nabla h_{\theta_j}(x_t^{(i)}),$$

where $\nabla h_{\theta_j}(x_t^{(i)})$ is obtained by backpropagation through the NN [17]. During resampling, both the particles and the associated α-vectors, $\{(x_t^{(i)}, \alpha_t^{(i)})\}_{i=1}^N$, are resampled according to the respective particle weights $\{w_t^{(i)}\}_{i=1}^N$. This ensures that the particle population remains diverse and representative of the posterior distribution.

After processing the entire observation sequence, the score vector is computed as a weighted sum of the accumulated gradients:

$$\nabla \log p_{\theta_j}(z_{1:T}) \approx \sum_{i=1}^{N} w_T^{(i)} \alpha_T^{(i)}. \tag{13}$$

Finally, the vector θ is updated using the gradient descent step (5). The proposed method for learning h_θ, referred to as neural network particle filter (NNPF), is provided in Algorithm 1.

Algorithm 1 NNPF: Neural Network Particle Filter

Input: Initial particle set $\{x_0^{(i)} \sim p(x_0)\}_{i=1}^N$, NN model and initial parameter vector θ_0, measurements $z_{1:T}$, number of training iterations n, number of particles N
Output: neural network parameter vector θ_n
for $j = 0$ to n **do**
  for $t = 1$ to T **do**
    for $i = 1$ to N **do**
      Sample from the motion model: $x_t^{(i)} \sim p(x_t \mid x_{t-1}^{(i)})$
      Compute particle weights: $w_t^{(i)} = p_{\theta_j}(z_t \mid x_t^{(i)})$
    Normalize weights: $w_t^{(i)} \leftarrow \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}}$, $i = 1, \ldots, N$.
    Estimate the state: $\hat{x}_t = \sum_{i=1}^N w_t^{(i)} x_t^{(i)}$
    Update the particle-specific score vectors:

$$\alpha_t^{(i)} = \alpha_{t-1}^{(i)} + \nabla \log p_{\theta_j}(z_t | x_t^{(i)})$$

    Resample $\{x_t^{(i)}, \alpha_t^{(i)}\}_{i=1}^N$ based on weights $\{w_t^{(i)}\}_{i=1}^N$.
  Compute the score: $\nabla \log p_{\theta_j}(z_{1:T}) \approx \sum_{i=1}^N w_T^{(i)} \alpha_T^{(i)}$.
  Update: $\theta_{j+1} = \theta_j + \eta \sum_{i=1}^N w_T^{(i)} \alpha_T^{(i)}$.

3. SIMULATION RESULTS

This section first studies the performance of NNPF for a synthetic 1D scenario with a known measurement model. NNPF is used to estimate the measurement function, which is compared to the ground truth. A camera calibration problem is then investigated: the measurement model is initialised with a pinhole model, and NNPF is used to estimate the camera distortion.

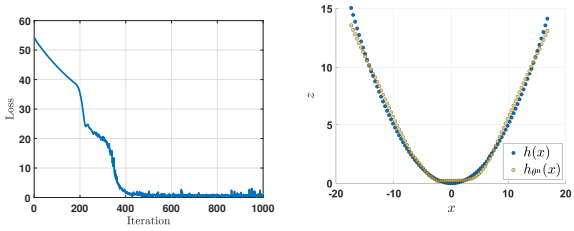
3.1. One-dimensional non linear filtering problem

The following nonlinear state space model is considered [18]:

$$x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{1 + x_{t-1}^2} + 8\cos(1.2(t-1)) + u_t, \tag{14}$$

$$z_t = x_t^2/20 + e_t, \tag{15}$$

where $u_t \sim \mathcal{N}(0, \sigma_u^2)$ and $e_t \sim \mathcal{N}(0, \sigma_e^2)$ are zero-mean Gaussian white noises with variances σ_u^2 and σ_e^2. The measurements $z_{1:T}$ are generated using the state and observation equations with $\sigma_u^2 = 0.1$ and $\sigma_e^2 = 0.1$ and a trajectory length $T = 200$. The NN architecture was chosen to obtain a reasonable problem complexity related to the ground truth measurement function $h(x_t) = x_t^2/20$, i.e., 3 hidden layers with 3 neurons. The Adam optimizer was initialized for the NN parameters with a learning rate $\eta = 0.01$ and an L2 regularization term equal to 0.01 [19]. The number of iterations is $n = 1000$ and the number of particles is $N = 100$. All these parameters were tuned by cross validation to obtain the best results.



(a) Loss vs iteration number.

(b) Ground truth and estimated measurement function.

Fig. 1: Performance measures for the synthetic data.

Figures 1a and 1b illustrate the performance of NNPF. The evolution of the loss versus the number of iterations is displayed in Fig. 1a (left), showing that 400 iterations are sufficient to obtain a reasonable performance. The estimated measurement function displayed in Fig. 1b (right) is in good agreement with the actual quadratic measurement function, with a mean square error (MSE) equal to 0.20 for state values evaluated on a regular grid in the interval $[-20, 20]$.

3.2. Camera Calibration

In this experiment, the state vector $x_t = [\, p_t^x,\ p_t^y,\ v_t^x, v_t^y \,]^\top$ contains the planar position and velocity at time t of a simulated camera. The camera is rigidly mounted 1 m above the ground, with its optical axis orthogonal to a 5×5 chessboard, with a square side length of 50 mm (Fig. 2). During a 60 s sequence sampled at 1 Hz, the camera undergoes a planar translation with a constant-velocity state model:

$$x_t = \begin{bmatrix} I_2 & (\Delta t) I_2 \\ 0_2 & I_2 \end{bmatrix} x_{t-1} + u_t,\ u_t \sim \mathcal{N}\Big(0, \begin{bmatrix} \sigma_p^2 I_2 & 0_2 \\ 0_2 & \sigma_v^2 I_2 \end{bmatrix}\Big). \tag{16}$$

The camera translates along the x-axis at a constant speed of 1 cm/s, with process-noise standard deviations $\sigma_p = \sigma_v = 10^{-3}$. The observation at time t contains measurements on $n_c = 36$ chessboard corners, where $z_{t,i}$, $i = 1, \ldots, n_c$, where $z_{t,i} \in \mathbb{R}^2$ is the image coordinate vector of the i-th chessboard corner, defined as:

$$z_{t,i} = h\big(K, x_{t,i}^{(c)}\big) + e_{t,i}, \quad e_{t,i} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_e^2 I_2), \quad i = 1, \ldots, n_c,$$

where $h(\cdot)$ denotes the camera projection function that integrates both the intrinsic parameters and the lens distortion model, using radial and tangential distortion coefficients $[0.1,\ -0.2,\ 5{\times}10^{-4},\ 5{\times}10^{-4}]$. The 3×3 matrix K encodes the camera's intrinsic calibration parameters and $x_{t,i}^{(c)}$ is the position of the ith corner in the camera frame at time t, computed from x_t. The measurement dataset for this example has $T = 60$ time steps, generated with $\sigma_e = 1$.
To estimate the camera distortion, the available motion information of the camera is exploited within the NNPF algorithm to learn the

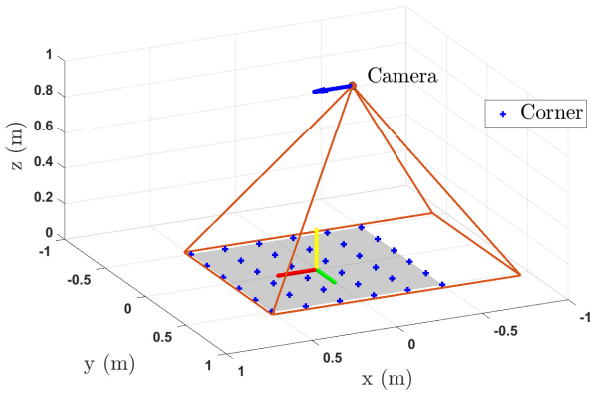


Fig. 2: Initial 3D scene showing the camera pose. Blue crosses indicate the corners of the chessboard, and the blue arrow represents the direction of camera motion.

relationship between 3D points and their corresponding 2D pixel positions, thereby avoiding the need for a traditional calibration procedure that requires paired 3D–2D labeled data. Every corner position $x_{t,i}^{(c)}$ is tracked using the constant-velocity state model (16). Due to the limited size of the dataset, prior knowledge is incorporated by modeling the measurement function as the sum of an ideal pinhole projection h_{m} [20] and a neural-network correction h_θ:

$$z_{t,i} = h_{\mathrm{m}}\left(K, x_{t,i}^{(c)}\right) + h_\theta\left(x_{t,i}^{(c)}\right) + e_{t,i}, \tag{17}$$

where $[\cdot]_3$ refers to the third component of a vector and

$$h_{\mathrm{m}}\left(K, x_{t,i}^{(c)}\right) = K x_{t,i}^{(c)} / [x_{t,i}^{(c)}]_3. \tag{18}$$

Note that the 3rd component of $h_{\mathrm{m}}(\cdot)$ is always 1.

The network architecture consists of five hidden layers, each comprising five neurons. The number of iterations is $n = 10^5$, and the number of particles is $N = 50$. Again all these parameters were chosen empirically to provide the best results. Figure 3 compares the undistorted pinhole image, the ground-truth distorted image, and the corrected image produced by the network. All observations were generated using the same camera motion described by (16), combined with three measurement functions: h_{m} for the pinhole model, h for the ground truth, and $\hat{h} = h_{\mathrm{m}} + \hat{h}_\theta$ for the learned function. Under the pinhole model and the specified camera motion, six parallel lines of observations appear in the image plane. The ground-truth function introduces visible distortion at the image periphery. The estimated function corresponds to the learned correction using the proposed method. The mean squared reprojection error is significantly reduced, from 7.52 with the pinhole model to 1.12×10^{-1} using the estimated function.

Figure 4 shows the tracking performance of a specific corner (bottom-left) for each measurement function. The plot highlights the tracking error of $\mathbf{x}_{\mathrm{c}}^i$ introduced by the pinhole model, as well as its correction achieved by the NN. Finally, Table 1 provides quantitative results for each measurement function allowing the tracking accuracy to be appreciated.

Table 1: Mean-squared reprojection error (expressed in pixels), for the different measurement models.

Model	h (true)	h_m (pinhole)	$\hat{h}$ (ours)
MSE	2.2×10^{-6}	3.4×10^{-4}	4.4×10^{-5}

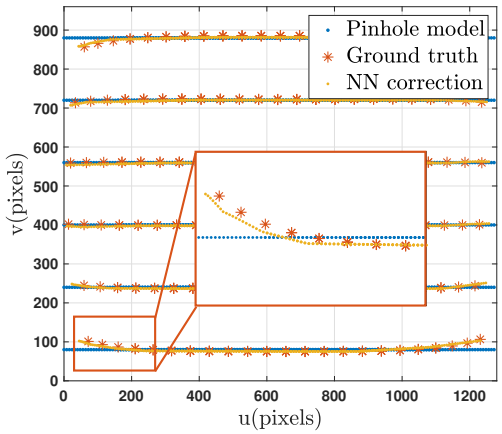


Fig. 3: Distortion map with the pinhole model (no distortion), the ground truth with the perfect distortion model and the NN correction

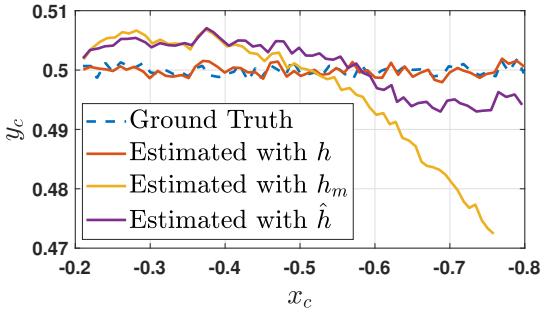


Fig. 4: Tracking of the bottom-left corner with three different measurement functions: h_{m} (pinhole model), h (ground truth), and $\hat{h}$ (estimated function)

4. CONCLUSION

This work studied a non-parametric approach based on neural networks to jointly estimate the measurement model and the system state within a particle filter framework. Simulation results showed that the proposed method can learn the measurement function accurately, significantly reducing the discrepancy between the estimated and ground-truth functions. In particular, it demonstrated strong potential for camera distortion estimation, enabling accurate calibration without relying on explicit distortion models. This data-driven strategy provides a flexible alternative to traditional parametric methods and holds promise for broader applications in state estimation for complex non-linear systems. Future work will explore the integration of this framework into real-time sensor calibration pipelines, addressing challenges such as temporal constraints and real-world conditions.

5. REFERENCES

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

[2] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[3] T. Zheng, A. Xu, X. Xinchao, and M. Liu, "Modeling and Compensation of Inertial Sensor Errors in Measurement Systems," *Electronics*, vol. 12, p. 2458, May 2023.

[4] J. Dong, N. Goodman, A. Carre, and P. Rajagopalan, "Calibration and validation-based assessment of low-cost air quality sensors," *Science of the Total Environment*, vol. 977, p. 179364, 2025.

[5] J. Yoo and J. Park, "Indoor localization based on wi-fi received signal strength indicators: Feature extraction, mobile fingerprinting, and trajectory learning," *Applied Sciences*, vol. 9, p. 3930, Sept. 2019.

[6] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[7] J. Wang, F. Shi, J. Zhang, and Y. Liu, "A new calibration model and method of camera lens distortion," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Beijing, China), pp. 5713–5718, 2006.

[8] P. Podbreznik and B. Potočnik, "Influence of temperature variations on calibrated cameras," *Int. Journal of Computer and Information Engineering*, vol. 2, pp. 546–551, Jan. 2008.

[9] C. Andrieu, A. Doucet, and V. Tadic, "On-line parameter estimation in general state-space models," in *Proc. IEEE Conference on Decision and Control*, (Seville, Spain), pp. 332–337, 2005.

[10] G. Poyiadjis, A. Doucet, and S. S. Singh, "Particle approximations of the score and observed information matrix in state space models with application to parameter estimation," *Biometrika*, vol. 98, no. 1, pp. 65–80, 2011.

[11] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin, "On particle methods for parameter estimation in state-space models," *Statistical Science*, pp. 328–351, Aug. 2015.

[12] X. Chen and Y. Li, "An overview of differentiable particle filters for data-adaptive sequential Bayesian inference," *Foundations of Data Science*, 2023.

[13] P. Karkus, D. Hsu, and W. S. Lee, "Particle filter networks with application to visual localization," in *Proc. Conf. on Robot Learning*, (Zurich, Switzerland), pp. 169–178, 2018.

[14] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," *ArXiv*, vol. abs/1805.11122, 2018.

[15] J. Li, X. Chen, and Y. Li, "Learning differentiable particle filter on the fly," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, (Pacific Grove, CA, USA), pp. 1355–1361, Oct. 2023.

[16] A. Singh, O. Makhlouf, M. Igl, J. Messias, A. Doucet, and S. Whiteson, "Particle-Based Score Estimation for State Space Model Learning in Autonomous Driving," in *Proc. Conf. Robot Learn (CoRL)*, (Atlanta, USA), pp. 1168–1177, 2023.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[18] 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, pp. 107–113, IET, 1993.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv*, 2014.

[20] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge university press, 2000.