Using nonlinear observers as virtual sensors in hydraulically actuated equipment in Industrial IoT systems

Victor Zhidchenko, Yashar Shabbouei Hagh, Egor Startcev, Heikki Handroos

Mechanical Engineering Department, LUT University, Lappeenranta, Finland
victor.zhidchenko@lut.fi

Abstract.

Real-time state observation of hydraulically actuated equipment requires powerful computational resources. The reason is the high sampling rate needed to capture dynamics of the hydraulic systems and additional computational complexity introduced by nonlinear estimators. In Industrial IoT applications, this problem is exacerbated by network delays. This paper considers the problem of improving the computational performance of the two popular nonlinear observers—Unscented Kalman Filter and Particle Filter—in IoT applications for hydraulic equipment. For the example case of a mobile log crane, the paper demonstrates several ways of building remote observers for IoT applications. Multi-body model of the crane coupled with the model of its hydraulic system allows capturing crane dynamics with different sets of sensors. Parallel computing on multi-core CPUs and GPUs is used to speed up calculations. Experimental results demonstrate the performance of the observers executed on a desktop computer and on an edge-computing device. The paper concludes with general guidelines for implementing nonlinear observers in these execution environments. The results can be applied in the remote and autonomous operation of hydraulically actuated equipment and maintenance activities for such machines.

Keywords. Observation, observer, Unscented Kalman Filter, Particle Filter, hydraulics, Internet of Things.

1. INTRODUCTION

Recent advancements in sensor and telecommunication technology allowed obtaining continuous information about remote equipment state. The benefits of processing this information within the Industrial Internet of Things (IIoT) and Digital Twin concepts are limited by the availability and cost of computing resources that can be allocated to each piece of equipment. At the time of writing, statistical and machine learning methods dominated in equipment fleet management systems. Physics-based methods, being widely used in simulation in the machine design phase, were not so popular in the operation and

maintenance phase. The reasons include the difficulty of precise model creation for each instance of equipment and the computational complexity of such methods, especially when a small time step is required. Since physics-based methods can enrich information by calculating unmeasured values and providing confident results, their integration into IIoT and Digital Twin concepts is an important task.

Observation techniques facilitate the application of physics-based models by decreasing the modeling errors related to the non-perfect fit of the model to a real object [1, 2]. They can also extend virtual sensor capabilities by parameter estimation by finding the values for which explicit formulas are not present in the model. Due to nonlinearity of models used for hydraulically actuated equipment simulation, nonlinear filters should be used to observe such equipment. This paper considers two widely used nonlinear filters, the Unscented Kalman Filter (UKF) and the Particle Filter (PF). Applying these methods for hydraulically actuated machines further increases the high computational complexity of the physics-based models. However, the capability of the model to provide results in real time is essential, even if there is no real-time connection with the machine. For IIoT and digital twin applications, it ensures that the model results are provided concurrently with the machine operation and no latency accumulates over time.

The goal of the presented work is to study real-time capabilities of computer programs that implement nonlinear filtering for the hydraulic models coupled with multi-body models of heavy equipment. Such programs can be used in IIoT systems to build digital twins of mobile working machines. Using a hydraulic log crane as an example, this paper considers real-time application of UKF and PF and demonstrates some techniques for improving their computational performance. The paper contributes to the development of IIoT and Digital Twin concepts by presenting possible ways of integrating physics-based models into the information processing pipelines for hydraulic equipment. For hydraulic cranes, such integration can facilitate crane operation [3] (especially remote and autonomous operation), remote surveillance, predictive maintenance, and fatigue life estimation [4].

The rest of the paper is organized as follows. Section II gives a brief literature review on the topic of the paper. The methodology used in this paper is addressed in Section III. Section IV introduces the experimental setup of the considered log crane. The results are discussed in Section V. Conclusions are presented in the final section.

2. RELATED WORK

Due to computational complexity, nonlinear filters have traditionally been used in application areas where the sampling interval is relatively large. For example, in position detection and some manufacturing and chemical processes, the measurements can be made once in several seconds [5]. The works considering nonlinear filtering in multi-body dynamics are more recent.

The computational efficiency of a program implementing model-based nonlinear filtering for multi-body systems depends primarily on the time needed to calculate a system state at each time step. The computational complexity of different multi-body formulations varies from $O(N^4)$ to $O(N)$, where N is the number of bodies [6]. Sometimes, it is impossible to choose the formulation because it is prescribed by the software used for multi-body modeling. If there is an opportunity to choose, the formulation with lower computational

complexity will give more freedom for the efficient filter implementation. Another important aspect is the possibility to represent the multi-body model in a state-space form, which is required for applying the filtering techniques.

The application of the nonlinear Kalman filters, such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF), in real-time state estimation of different nonlinear systems has gained significant attention in the past years. For instance, [7, 8] proposed a new version of the EKF algorithm named indirect EKF based on multi-body models. Using different methods to calculate the Jacobian matrix, it has been shown that the proposed EKF could achieve the same level of accuracy as UKF but with less computational cost. However, by reducing the sampling rate of sensors, the quality of the estimations degrades faster than with UKF. Nevertheless, computing the Jacobian matrix might not be an easy task, especially for systems with severe nonlinearity. Furthermore, [9] applied the nonlinear Kalman filters state estimation using multi-body models. To this end, EKF, UKF, and spherical simplex UKF (SSUKF) are applied to a 5-bar linkage mechanism. Emphasizing on the computational cost of the UKF-based filters, these filters resulted in more accurate estimations than the EKF. On the other hand, it has been mentioned that the easy implementation of the UKF and its ability to use parallel computing are the reasons that make UKF capable to run in real time.

Representing a subset of Monte Carlo methods, particle filters support parallelization. A generic parallel algorithm for particle filter was presented in [10]. Experiments on different computers using the Message Passing Interface (MPI) technology demonstrated an ability to implement the algorithm in real time for the simple synthetic tasks of position estimation. A survey on early developments in distributed estimation with particle filtering can be found in [11]. The paper considered the particle filter a viable alternative for real-time implementations in the near future. Nonetheless, a decade later, the method was still used primarily in low-sampling rate applications.

Attempts to improve the computational efficiency of particle filtering by using GPUs appeared immediately with the rise of this technology. One of the early works [12] considered a general approach for parallelizing the particle filter on GPUs. In more recent works, GPU-accelerated particle filters were applied to the tasks of controlling the robotic arm [13] and manufacturing processes [5].

3. METHODOLOGY

3.1. Multi-body Model of the Crane

This paper studies the application of nonlinear filtering to the multi-body system state estimation using a simple real machine. Mobile hydraulic log crane PATU655 installed in the laboratory of intelligent machines of LUT University was used as a test system (Figure 3.1). The crane consists of a pillar, a main boom, and a jib boom. A grapple was not attached to the crane for safety reasons. Constrained space in the laboratory did not allow to rotate the pillar. Therefore, the crane could only move in two dimensions by rotating the main and the jib booms. Two hydraulic cylinders actuated the booms. The jib boom was connected to the cylinder by a four-bar mechanism.

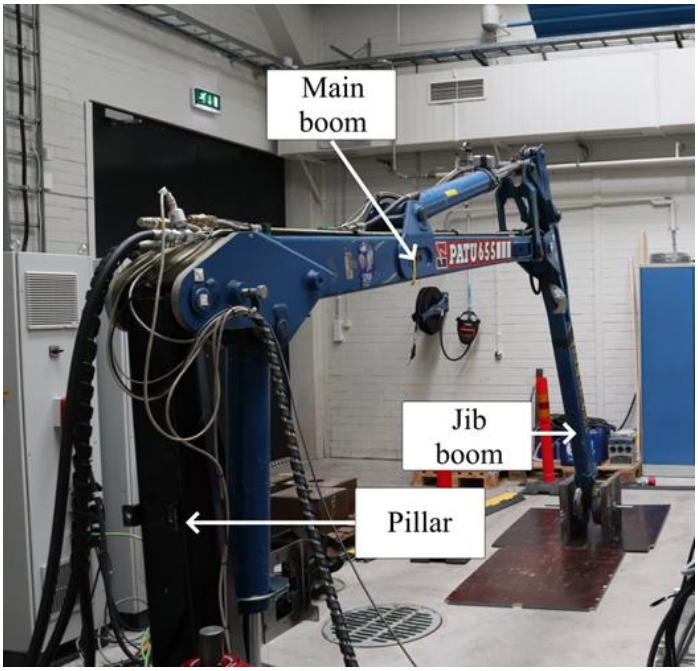


Figure 3.1. The mobile hydraulic log crane considered in the paper

The multi-body model based on the Iterative Newton-Euler Formulation (INEF), which was developed and described in [3] and [4], was used to simulate the crane dynamics. The model considers four states: the angle of the main boom, the angular velocity of the main boom, the angle between the main and the jib boom, and the angular velocity of the jib boom relative to the joint between the booms. The forces produced by the hydraulic cylinders are calculated from the pressure in the cylinder chambers, the geometry of the cylinders, and the speed of the cylinder motion.

For state estimation problems, the state-space representation of the dynamic model offers a general framework. Thus, in this paper, the state-space formulation is utilized to introduce the filtering algorithms.

Consider a discrete-time nonlinear system as:

$$\begin{aligned} x_k &= \mathcal{F}(x_{k-1}, u_{k-1}) + w_{k-1} \\ y_k &= \mathcal{H}(x_k) + v_k \end{aligned} \tag{3.1}$$

in which x_k denotes the state vector of the system with the dimension of n_x, and u_k represents the input control signal. $\mathcal{F}(x_{k-1}, u_{k-1})$ represents the dynamic system function. The observation is denoted by y_k with dimension of n_y and measurement sensitivity matrix of $\mathcal{H}(x_k)$. The nonlinear system is under an additive Gaussian process and measurement noise. $w \sim \mathrm{N}(0, \mathcal{Q})$ and $v \sim \mathrm{N}(0, \mathcal{R})$ represent the process and the measurement noises with zero-mean and the covariance matrix of $\mathcal{Q}$ and $\mathcal{R}$, respectively.

3.2. Unscented Kalman Filter

As an alternative to the conventional EKF with deficiencies due to linearization, which can be found in the literature [14], the UKF algorithm was proposed to address these drawbacks [15]. This algorithm is based on a sampling technique named Unscented Transform, which generates a set of sample points known as sigma-points to capture the true mean and covariance of the random variable [16, 17].

The algorithm is initialized with $k = 0$ values of the error covariance matrix as $P_{0|0} = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$ and the state estimation as $\hat{x}_{0|0} = E[x_0]$. Based on these assumptions the sigma-points are calculated for $k = 1$ as:

$$\chi^{(j)}_{k-1|k-1} = \left[\hat{x}_{k-1|k-1}, \hat{x}_{k-1|k-1} + \sqrt{n_x + \Gamma}\sqrt{P_{k-1|k-1}}, \hat{x}_{k-1|k-1} - \sqrt{n_x + \Gamma}\sqrt{P_{k-1|k-1}}\right] \tag{3.2}$$

in which $j = 0, \dots, 2n_x$, $\sqrt{\cdot}$ denotes the square root of the Cholesky decomposition using its lower triangular matrix and $\Gamma = \alpha^2(\mathrm{n_x} + \kappa) - \mathrm{n_x}$. α and κ are the scaling parameters in which $0 \leq \alpha \leq 1$ controls the spread of the sigma-points and $\kappa \geq 0$ guarantees positive semi-definiteness of the covariance matrix. Then, based on the dynamic system function defined in Eq.(3.1), the generated sigma-points should be propagated as:

$$\gamma^{(j)}_{k|k-1} = \mathcal{F}\left(\chi^{(j)}_{k-1|k-1}, u_{k-1}\right), j = 0,1,\dots,2n_x \tag{3.3}$$

Now, using the predefined set of weights as $\omega_0^{(m)} = \Gamma/(n_x + \Gamma)$, $\omega_0^{(c)} = \omega_0^{(m)} + (1 - \alpha^2 + \beta)$, $\omega_j^{(c)} = \omega_j^{(m)} = 1/2(n_x + \Gamma)$, the *a priori* estimation of the mean and the covariance can be calculated using the following formulations:

$$\begin{aligned} \hat{x}_{k|k-1} &= \textstyle\sum_{j=0}^{2n_x} \omega_j^{(m)} \gamma^{(j)}_{k|k-1} \\ \mathrm{P}_{k|k-1} &= \textstyle\sum_{\mathrm{j}=0}^{2\mathrm{n_x}} \left\{\omega_{\mathrm{j}}^{(c)} \left(\gamma^{(j)}_{k|k-1} - \hat{x}_{k|k-1}\right)\left(\gamma^{(j)}_{k|k-1} - \hat{x}_{k|k-1}\right)^{\mathrm{T}}\right\} + \mathcal{Q}_{\mathrm{k\text{-}1}} \end{aligned} \tag{3.4}$$

in which β as the secondary scaling factor is set to be 2 as an optimal choice for Gaussian noise distributions. In order to correct the *a priori* estimations, a new set of sigma-point should be generated as:

$$\chi^{(j)}_{k|k-1} = \left[\hat{x}_{k|k-1}, \hat{x}_{k|k-1} + \sqrt{n_x + \Gamma}\sqrt{P_{k|k-1}}, \hat{x}_{k|k-1} - \sqrt{n_x + \Gamma}\sqrt{P_{k|k-1}}\right] \tag{3.5}$$

then, each column of the generated sigma-points will propagate through the measurement sensitivity matrix as:

$$\mathcal{Y}^{(j)}_{k|k-1} = \mathcal{H}\left(\chi^{(j)}_{k|k-1}\right), \quad j = 0,1,\dots,2\mathrm{n}_x \tag{3.6}$$

Based on the propagated sigma-points the predicted measurements, innovation covariance, and the cross-covariance of the predicted measurement can then be computed as follows:

$$\begin{aligned} \hat{\mathcal{Y}}_{k|k-1} &= \textstyle\sum_{\mathrm{j}=0}^{2\mathrm{n}_x} \omega_j^{(m)} \mathcal{Y}^{(j)}_{k|k-1} \\ P_k^{xy} &= \textstyle\sum_{\mathrm{j}=0}^{2\mathrm{n}_x} \omega_j^{(c)} \left(\gamma^{(j)}_{k|k-1} - \hat{x}_{k|k-1}\right)\left(\mathcal{Y}^{(j)}_{k|k-1} - \hat{\mathcal{Y}}_{k|k-1}\right)^{\mathrm{T}} \\ P_k^{yy} &= \textstyle\sum_{\mathrm{j}=0}^{2\mathrm{n}_x} \left\{\omega_j^{(c)} \left(\mathcal{Y}^{(j)}_{k|k-1} - \hat{\mathcal{Y}}_{k|k-1}\right)\left(\mathcal{Y}^{(j)}_{k|k-1} - \hat{\mathcal{Y}}_{k|k-1}\right)^{\mathrm{T}}\right\} + \mathcal{R}_k \end{aligned} \tag{3.7}$$

Based on P_k^{xy} and P_k^{yy}, the Kalman gain K_k can be calculated, which is used to calculate the *a posteriori* mean state vector $\hat{x}_{k|k}$ and the covariance matrix $P_{k|k}$ as:

$$\begin{aligned} K_k &= P_k^{xy}\left(P_k^{yy}\right)^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k\text{-}1} + K_k\left(y_k - \hat{y}_{k|k-1}\right) \\ P_{k|k} &= P_{k|k\text{-}1} - K_k P_k^{yy} K_k^{\mathrm{T}} \end{aligned} \tag{3.8}$$

The equations from Eq. (3.2) to Eq. (3.8) are calculated for each new time step. This loop is repeated until the stop criteria are met.

3.3. Particle Filter

The particle filter algorithm, which is based on the Monte Carlo method, uses a set of randomly chosen samples (particles) with associated weights to estimate the posterior probability density function (PDF) [18, 19]. The idea of representing the posterior PDF with a set of samples and weights is referred to as the sequential importance sampling (SIS) algorithm. This posterior PDF can be approximated as $p[x_k|y_k] \approx \sum_{i=1}^{N} w_k^i\, \delta\left[x_k - x_k^i\right]$, in which x_k^i is the set of random samples, N is the number of samples, w_k^i are the associated weights, and $\delta[\cdot]$ is the Dirac delta measure. This sampling procedure from the true posterior PDF is usually impossible, hence an importance density as $q\left[x_k^i|x_{k-1}^i, y_k\right]$ is used. Therefore, at each sampling instant, a sample is drawn from the importance density rather than the true posterior PDF. To compensate for the difference between these two densities, the associated weights can be calculated in a sequential case as:

$$\tilde{w}_k^i = \tilde{w}_{k\text{-}1}^i \frac{p\left[y_k|x_k^i\right]p\left[x_k^i|x_{k-1}^i\right]}{q\left[x_k^i|x_{k-1}^i, y_k\right]} \implies w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^{N} \tilde{w}_k^j} \tag{3.9}$$

One major issue with the SIS filter is known as weight degeneracy. In this case, after some iterations, only a few particles have non-zero importance weights. In other words, after some iterations, some particles will have zero contribution to the approximation of $p[x_k|y_k]$. A resampling step is introduced to resolve this drawback, which is based on the sampling importance resampling particle filter (SIR-PF) algorithm. This step discards the samples with low importance using different methods such as: residual sampling [20], systematic sampling [18], stratified sampling [21]. However, owing to the simplicity and efficiency of systematic resampling, this is a common choice in an SIR method. The algorithm of this resampling method is given in [22]. The SIR-particle filter algorithm can be summarized as follows:

- Step (1): Initialization: For $k = 0$ generate initial particles x_0^i and the associated weights w_0^i for $i = 1, \ldots, N$ from the *a priori* distribution;
- Step (2): Importance sampling: Generate prior particles, $x_k^{i,-}$, from the importance sampling density function;
- Step (3): Weighting: Calculate the weights of each particle once a new measurement is available using Eq. (3.9);
- Step (4): Resampling: Based on the calculated weightings and resampling method of systematic resampling, calculate the *a posteriori* particles x_k^i;

- Step (5): Estimation: Estimate the state by calculating $\hat{x}_k = \sum_{i=1}^{N} w_k^i \, x_k^i$. Set $k = k + 1$ and go back to Step (2).

3.4. Particle Filter Implementation on a GPU

This study used Nvidia CUDA technology to accelerate particle filter implementation. Three features of Nvidia GPUs were utilized to minimize calculation time: a substantial amount of device memory, asynchronous memory transfers, and streams. Since the function $\mathcal{F}(x, u)$ derived from INEF formulation was implemented as a custom C function, it was converted to a kernel function to generate particles concurrently at step (2) of the particle filter algorithm. Another kernel function performed weighting at step (3) of the algorithm.

The most time-consuming operations in a particle filter are the calculation of the system function $\mathcal{F}(x, u)$ and the generation of the pseudo-random numbers. The latter operation was performed in a batch mode when many random values were generated by a single function call and stored in the device memory as a lookup table. Instead of calling a time-consuming random generator at each time step, the values from the table were used across several time steps to create particles. When the lookup table with random numbers was consumed, it was generated again. System function calculation and random number generation were performed in different streams. These measures allowed to implement real-time particle filter for the crane model with 1024 particles. The accuracy of the filter was considered sufficient, and the resampling step was not performed on the GPU.

4. EXPERIMENTAL SETUP

The log crane used in the experiments was instrumented with pressure sensors for the piston side and the rod side of each cylinder, position sensors measuring the length of each cylinder, and the gyroscopes attached to each boom. The sensors were connected to the dSPACE 1005 data acquisition system, which transmitted the data to the computer installed in the lab using the local area network connection. A Python script was used to save the data on disk and send them to a remote computer in LUT University by the UDP protocol.

Two sets of experiments were performed. In the first set, the accuracy and computational performance of the UKF and particle filters were evaluated. The particle filter was implemented by a custom-made C program using Nvidia CUDA technology for GPU computing. The C program implementing the UKF filter was automatically generated from a custom MATLAB script using the MATLAB Coder tool. Both programs read the measurement data from a CSV file, estimated the system state using a full or constrained set of sensors, and saved the results to another CSV file. Measurement data consisted of the boom angles calculated from the cylinder linear displacement measurements, angular velocity of the booms measured by gyroscopes, and pressure in the cylinder chambers measured by pressure sensors. Calculations were performed on a desktop computer equipped with the Intel Core i9-9820X 3.3GHz CPU and NVIDIA GeForce RTX 2080 SUPER GPU. The program implementing the UKF filter was also compiled and executed on a Raspberry Pi 4 Model B microcomputer. Optimizing compilation was not used in the experiments. Each program version was executed 30 times on the same data set. Between the executions, 15 seconds breaks were made to decrease the influence of other programs on CPU load. Execution time was measured using the omp_get_wtime() function of the OpenMP library. Other capabilities of the OpenMP technology were not used in the experiments.

In the second set of experiments, the nonlinear filters were used for the real-time visualization of the crane motion. A remote computer in LUT University connected to the crane measurement system by the local area network represented an IoT system. CAD drawings of the crane were used to visualize its motion in the Unity environment. The programs implementing UKF and particle filters were modified to receive the input data by UDP protocol instead of reading them from a CSV file. The Python script executed on the computer located near the dSPACE system transmitted the measurement data to the remote computer. The sensor data were transmitted with a sampling rate of 1000 samples per second. The programs implementing the filtering were calculating the boom angles and sending them to the Unity program 40 times per second. The user interface of the Unity program allowed to choose the measured states by setting the checkboxes in real time, during the program execution. A 3D model of the crane was displayed in real time on-top of the video stream from a camera installed in the laboratory near the crane. While one researcher was operating the crane, the second researcher watched the concurrent motion of the crane and its 3D model on a video wall located in another laboratory of LUT University. The second researcher was also able to select the measured states in real time using the checkboxes. The experimental environment is presented in Figure 4.1.

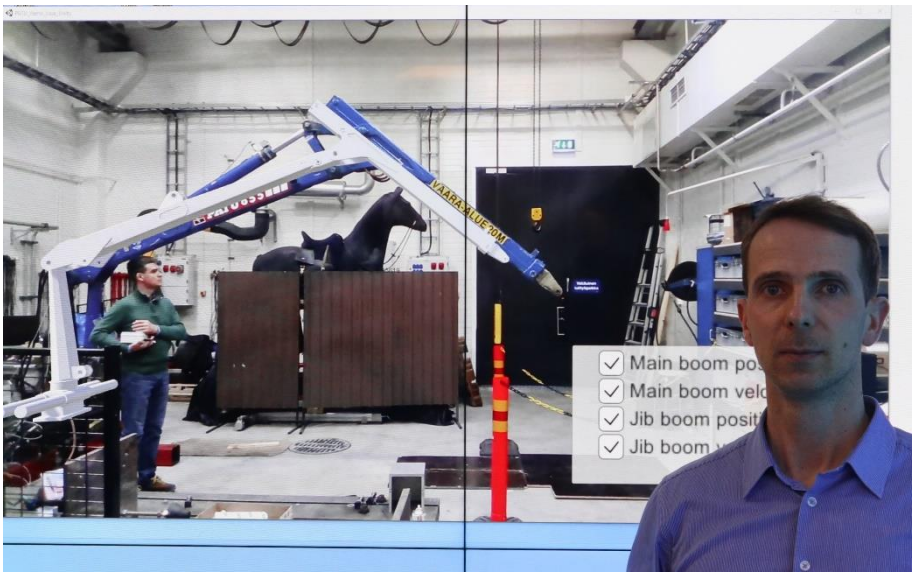
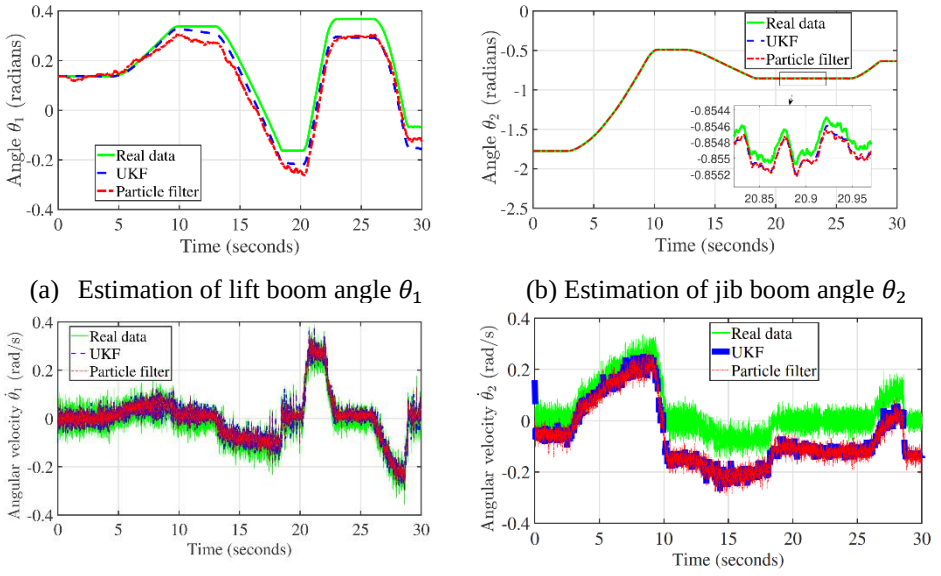


Figure 4.1. Experimental environment

5. RESULTS

The results of the first set of experiments are presented in Table 1 and Figure 5.1. Table 1 presents the execution time of different programs analyzing the data set obtained during the 39.992 s of measurements. The execution duration of the program implementing the UKF filter was 14.7 times smaller than the duration of the analyzed period, even on the Raspberry Pi microcomputer (denoted in the table as RPi). The program implementing the particle filter was able to fit the analyzed period duration using GPU and 1024 particles.



(a) Estimation of lift boom angle θ_1

(b) Estimation of jib boom angle θ_2

(c) Estimation of lift boom velocity $\dot{\theta}_1$

(d) Estimation of jib boom velocity $\dot{\theta}_2$

Figure 5.2. Estimation of the hydraulically actuated log crane states using the UKF and Particle filter

Figure 5.1 shows the measured and estimated states using the UKF and particle filter for the same input data that was used in the first set of experiments. In this example, the angular velocity of the main boom $\dot{\theta}_1$ and the angle of the jib boom θ_2 calculated from the position of the jib cylinder were used as measurements. The angle of the main boom θ_1 and the angular velocity of the jib boom $\dot{\theta}_2$ were estimated by the filters.

As for the estimation accuracy of the two approaches, Figure 5.1 indicates that both methods are capable to estimate the true states of the model. To have a better metric on how these methods perform, root mean square error (RMSE) is calculated for all four states. The RMSE of the UKF are: $\theta_1 = 0.2174\,rad$, $\theta_2 = 0.0203\,rad$, $\dot{\theta}_1 = 0.0001\,rad/s$, $\dot{\theta}_2 = 0.1401\,rad/s$ and the RMSE of the particle filter method are: $\theta_1 = 0.2684\,rad$, $\theta_2 = 0.0212\,rad$, $\dot{\theta}_1 = 0.0001\,rad/s$, $\dot{\theta}_2 = 0.1448\,rad/s$. It can be seen that the performance of the UKF method is slightly better than the particle filter.

Table 1: Execution time of different observer implementations

	UKF	**Particle filter**		
	PC	**RPi**	**GPU**	**CPU**
Total time (s)	0.654	2.718	13.844	80.510
Reading from file (s)	0.505	2.062	1.354	1.359
Calculations (s)	0.086	0.480	12.176	78.840
Writing to file (s)	0.046	0.105	0.312	0.309

6. CONCLUSION

The presented study demonstrated the possibility of using nonlinear observers for the real-time state estimation of hydraulically actuated equipment in Industrial IoT systems. The capability of the observation techniques to estimate unmeasured states allows using them as virtual sensors and decreasing the number of sensors installed on the equipment. It should be noted that due to the nonlinear nature of the hydraulically actuated systems, observer parameter tuning might be required for each type of equipment, taking into account its operating conditions.

The experiments performed in the current study showed that the UKF-based observer was able to run up to 142 times faster than the particle filter implemented on GPU while providing similar estimation accuracy. The result favors the use of UKF for virtual sensor implementation in Industrial IoT systems.

A combination of UKF and PF in a single observer for an IIoT system could be considered for future research. For the most common operating conditions, a fast UKF-based estimator could be used. For the periods with high nonlinearity detected using additional information on a particular machine, a PF-based estimator could be applied. Another research topic would be an improvement of an observer's performance by using the measurement data for a period of the machine operation. In the presented experiments, each sample was transmitted separately, while in practice, it is more efficient to transmit a sequence of samples in an IIoT system at once. Processing such sets of data could improve observer performance while preserving its real-time capabilities.

7. REFERENCES

[1] D. Simon, Optimal state estimation: Kalman, H infinity, and nonlinear approaches. Nashville, TN: John Wiley & Sons, 2006.

[2] J. V. Candy, Bayesian signal processing: Classical, modern, and particle filtering methods, 2nd ed., ser. Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control. Nashville, TN: John Wiley & Sons, 2016.

[3] I. Malysheva, H. Handroos, V. Zhidchenko, and A. Kovartsev, “Faster than real-time simulation of a hydraulically actuated log crane,” in 2018 Global Fluid Power Society PhD Symposium (GFPS). IEEE, 2018, pp. 1–6.

[4] V. Zhidchenko, H. Handroos, and A. Kovartsev, “Application of digital twin and IoT concepts for solving the tasks of hydraulically actuated heavy equipment lifecycle management,” International Journal of Engineering Systems Modelling and Simulation, vol. 11, no. 4, pp. 194–206, 2020.

[5] F. Lopez, L. Zhang, J. Beaman, and A. Mok, “Implementation of a particle filter on a GPU for nonlinear estimation in a manufacturing remelting process,” in 2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. IEEE, 2014, pp. 340–345.

[6] R. Featherstone and D. E. Orin, “Robot dynamics: equations and algorithms,” Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), vol. 1, pp. 826–834 vol.1, 2000.

[7] E. Sanjurjo, D. Dopico, A. Luaces, and M. Á. Naya, “State and force observers based on multibody models and the indirect Kalman filter,” Mechanical Systems and Signal Processing, vol. 106, pp. 210–228, 2018.

[8] A. J. Rodríguez, E. Sanjurjo, R. Pastorino, and M. Á. Naya, “State, parameter and input observers based on multibody models and Kalman filters for vehicle dynamics,” Mechanical Systems and Signal Processing, vol. 155, p. 107544, 2021.

[9] R. Pastorino, D. Richiedei, J. Cuadrado, and A. Trevisani, “State estimation using multibody models and non-linear Kalman filters,” International Journal of Non-Linear Mechanics, vol. 53, pp. 83–90, 2013.

[10] O. Brun, V. Teuliere, and J.-M. Garcia, “Parallel particle filtering,” Journal of Parallel and Distributed Computing, vol. 62, no. 7, pp. 1186– 1202, 2002.

[11] A. Simonetto and T. Keviczky, “Recent developments in distributed particle filtering: towards fast and accurate algorithms,” IFAC Proceedings Volumes, vol. 42, no. 20, pp. 138–143, 2009.

[12] G. Hendeby, R. Karlsson, and F. Gustafsson, “Particle filtering: the need for speed,” EURASIP Journal on Advances in Signal processing, vol. 2010, pp. 1–9, 2010.

[13] M. Chitchian, A. Simonetto, A. S. van Amesfoort, and T. Keviczky, “Distributed computation particle filters on GPU architectures for Real-Time control applications,” IEEE Trans. Control Syst. Technol., vol. 21, no. 6, pp. 2224–2238, Nov. 2013.

[14] T. Fiorenzani, C. Manes, G. Oriolo, and P. Peliti, “Comparative study of unscented Kalman filter and extended Kalman filter for position/attitude estimation in unmanned aerial vehicles,” Inst. for Systems Analysis and Computer Science (IASI-CNR), Rome, Italy, Rept, pp. 08–08, 2008.

[15] S. J. Julier and J. K. Uhlmann, “Unscented filtering and nonlinear estimation,” Proceedings of the IEEE, vol. 92, no. 3, pp. 401–422, 2004.

[16] Y. S. Hagh, R. M. Asl, and V. Cocquempot, “A hybrid robust fault tolerant control based on adaptive joint unscented Kalman filter,” ISA transactions, vol. 66, pp. 262–274, 2017.

[17] M. Mohammadi, Y. S. Hagh, X. Yu, H. Handroos, and A. Mikkola, “Determining the State of a Nonlinear Flexible Multibody System using an Unscented Kalman Filter,” IEEE Access, 2022.

[18] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” IEEE Transactions on signal processing, vol. 50, no. 2, pp. 174–188, 2002.

[19] X. Han and X. Li, “An evaluation of the nonlinear/non-Gaussian filters for the sequential data assimilation,” Remote Sensing of environment, vol. 112, no. 4, pp. 1434–1449, 2008.

[20] J. S. Liu and R. Chen, “Sequential Monte Carlo methods for dynamic systems,” Journal of the American statistical association, vol. 93, no. 443, pp. 1032–1044, 1998.

[21] P. Fearnhead et al., “Sequential Monte Carlo methods in filter theory,” University of Oxford, 1998.

[22] A. Tulsyan, R. B. Gopaluni, and S. R. Khare, “Particle filtering without tears: A primer for beginners,” Computers & Chemical Engineering, vol. 95, pp. 130–145, 2016.

Biographies

Victor Zhidchenko received the D.Sc. (Tech.) degree from LUT University, Lappeenranta, Finland, in 2019, and the Candidate of Sciences degree from Samara National Research University, Russia. He is currently working as a Postdoctoral Researcher at LUT University. His research interests include computer simulation, cyber-physical systems, digital twins for heavy equipment, big data, cloud computing, and the Internet of Things.

Yashar Shabbouei Hagh received his M.Sc. degree in Control Engineering from the University of Tabriz, Iran, in 2016. In 2023, he obtained his doctoral degree in Mechanical Engineering from LUT University, Finland. At the time of the conference, he was a Junior Researcher at the Laboratory of Intelligent Machines, Department of Mechanical Engineering, LUT University. His research focuses on synergetic control theory and the state and parameter estimation of multibody dynamic systems, with applications to digital twins. His interests include fault detection and diagnosis systems, fault-tolerant controls, sliding mode controllers, synergetic control, nonlinear Kalman filtering, and robotic manipulators.

Egor Startcev received an Engineering degree in computing machines, complexes, systems, and networks from Samara State Technical University, Russia, in 2003. He is a former Microsoft Most Valuable Professional (MVP) for Data Protection Manager in 2011 and Cloud and Datacenter Management from 2012 to 2013. He has 15 years of experience in managing the IT department of a large construction holding. He is currently working as a Junior Researcher at LUT University. His research interests include cloud computing, digital twins, virtualization technologies, and the Internet of Things.

Heikki Handroos received the M.Sc. (Eng.) and D.Sc. (Tech.) degrees from the Tampere University of Technology in 1985 and 1991, respectively. He has been a Professor of machine automation at LUT University since 1992. He has also been a Visiting Professor at the University of Minnesota, Minneapolis, MN, USA; Peter the Great St. Petersburg Polytechnic University, Saint Petersburg, Russia; and the National Defense Academy, Japan. He has published about 250 international scientific papers and supervised around 20 D.Sc. (Tech.) theses. He has led several important domestic and international research projects. His research interests include modeling, design, and control of mechatronic transmissions to robotics and virtual engineering. He has been an Associate Editor of the Journal of Dynamic Systems, Measurement, and Control since 2014.