Delay-Tolerant Data Fusion for Underwater Acoustic Tracking Networks

Mohammadreza Alimadadi, Milica Stojanovic and Pau Closas
Department of Electrical and Computer Engineering, Northeastern University, Boston, USA
{alimadadi, millitsa, closas}@ece.neu.edu

2021 Fifth Underwater Communications and Networking Conference (UComms) | 978-1-7281-9315-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/UCOMMS50339.2021.9598132

***Abstract*—We consider a network of distributed underwater sensors whose task is to monitor the movement of objects across an area. The sensors measure the strength of signals emanated by the objects and convey the measurements to the local fusion centers. Multiple fusion centers are deployed to cover an arbitrarily large area. The fusion centers communicate with each other to achieve consensus on the estimated locations of the moving objects. We introduce two efficient methods for data fusion of distributed partial estimates when delay in communication is not negligible. We concentrate on the minimum mean squared error (MMSE) global estimator, and evaluate the performance of these fusion methods in the context of multiple-object tracking via extended Kalman filtering. Numerical results show the superior performance compared to the case when delay is ignored.**

***Index Terms*—data fusion, statistical inference, Kalman filter, object tracking, sensor network, random access, underwater acoustic communication, delay.**

I. Introduction

We consider the problem of multiple object tracking within the setting of an underwater sensor network, where multiple distributed sensor nodes communicate their field measurements acoustically to local fusion centers (FCs). The FCs are in turn connected over a separate infrastructure, which does not share the same communication bandwidth with the sensors. In underwater acoustic scenarios, for instance, the FCs can be connected by radio links (if close to shore) or by satellite links (if remote). The FCs use this separate infrastructure to exchange information and achieve large area coverage. The specific problem we address within this framework is that of communication delay between the FCs.

A mathematically equivalent problem is often found in statistical signal processing, where models parameters need to be inferred from a large data-set [1]. Performing statistical inference in these situations imposes computational/memory/storage constraints that make global estimation impractical [2]. A common solution is to divide the data among multiple partial estimator (PE) and to combine their estimates together to approximate the global one. The goal is to find an optimal combination that achieves a performance as close as possible to the global estimator that has access to all the information.

Data fusion of PEs has been studied in many different contexts including signal processing, digital communication, control and sensor networks. A general procedure to combine estimators in the multiple parameter case has been proposed in [3]. Time series forecasting [4], distributed estimation in wireless sensor networks [5]–[7], optimal linear fusion for multi-dimensional case [8], distributed fusion by adapting methods for graphical models [9], as well as different consensus, gossip or diffusion algorithms [10]–[12] are some of the previous works in this area. In the consensus-oriented distributed tracking in underwater acoustic sensor networks, nodes are arranged in groups and each group has an FC that runs an independent filter [13], [14], while exchanging information with its neighbors iteratively. As a result, FCs can reach a consensus [10], with the benefits of network cooperation in terms of improved performance and better robustness and resilience to failure.

These approaches have certain limitations. Firstly, in most applications, FCs are either the cores in a multi-core worker or machines in a multiple-machine setup. In either case, communication happens over a wired platform and the delay is considered negligible. This is not a reasonable assumption in underwater networks where the delay can be significant. Moreover, unlike regular distributed fusion, synchronization is not easy to achieve in underwater situations. For these reasons, regular distributed inference methods are in general not suitable for underwater scenarios.

In this paper, we introduce two efficient data fusion methods for the fusion of distributed estimators, where delay in communication is not negligible. We concentrate on the minimum mean squared error (MMSE) global estimator, but the developed framework is general and can be used to combine any type of unbiased partial estimates.

The rest of the paper is organized as follows. Sec. II presents the system model. In Sec. III the proposed fusion methods are introduced. Numerical results are presented in Sec. IV. Finally, we provide concluding remarks in Sec. V.

II. System Model

In [15] we addressed a scenario in which networked sensor nodes measure the strength of the field generated by a number of moving objects (less or equal to M) and transmit their measurements to an FC in a random access manner for final reconstruction of the objects' trajectories. To ensure scalability, the total observation area is divided into design units, each of which is assigned an FC that runs an extended Kalman filter to produce its partial estimate. The neighboring FCs then communicate with each other to exchange current status, thus

This work was supported by one or more of the following grants: ONR N00014-15-1-2550, NSF CNS-1726512, CNS-1815349 and ECCS-1845833.

allowing state fusion whereby the Kalman filters adjust their local estimates at the end of each updating interval, yielding a system design that is scalable across a large coverage area without an increase in complexity. Fig. 1 illustrates the concept. The proposed methods are specially useful in underwater scenarios with limited bandwidth and power restrictions. While [15] treated an ideal case in which communication between FCs occurs with no delay, here we take into account the fact that the information exchanged between the FCs may be outdated.

In this paper we consider a similar scenario as in [15]. The location and the velocity of the m-th object at time interval k are denoted by

$$\mathbf{x}_k(m) = [x_k(m),\ y_k(m),\ \dot{x}_k(m),\ \dot{y}_k(m)]^\top \tag{1}$$

The location of objects follows a stationary uniform process with density ρ_o objects/m^2. Each object emits a signal of certain amplitude. The signal decays with distance d based on a known signature function $\mathrm{f}(d)$. It is worth mentioning that here we do not strive to discriminate the objects from one another. The objects are not cooperating, i.e. they do not send their IDs to the sensors, and the sole job of the system is to track their location based on the signatures they emanate.

A number of sensor nodes are distributed uniformly across the observation area with density ρ_s nodes/m^2. Nodes are divided into zones and an FC is assigned to each zone. At time interval k the received signal at the n-th sensor node is generically modeled as

$$v_k(n) = \sum_m \mathrm{f}\left(d_k(m,n)\right) + w_k(n) \tag{2}$$

where $d_k(m,n)$ is the distance between the m-th object and the n-th sensor at time k, and $w_k(n)$ is the zero-mean Gaussian noise.

After sensing the field, each node encodes its measurement $v_k(n)$ into a digital data packet, adds its ID, and transmits the packet to its FC. Each FC collects the packets transmitted in one time interval of duration T_c, which is chosen short enough that the objects' locations can be considered fixed over it. Some packets will be dropped because of collisions and noise. After discarding the erroneous packets, the FC is left with useful packets. These intermittent observations are then used as an input to the tracking algorithm to estimate the location of the objects.

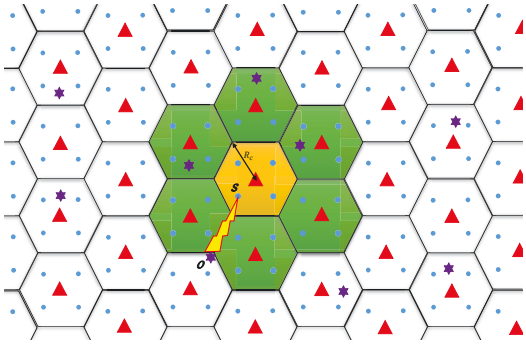

Fig. 1: The entire area is divided into sensing cells, and an FC is assigned to each cell. Sensor nodes are shown as circles, FCs as triangles, and moving objects as stars [15].

We define the state of the system at time kT_c, as

$$\mathbf{x}_k = [\mathbf{x}_k^\top(1),\ \mathbf{x}_k^\top(2),\ \ldots,\ \mathbf{x}_k^\top(M)]^\top \tag{3}$$

where T_c is the collection interval, $k \in \mathbb{Z}^+$, M is the maximum number of objects to be tracked by one FC and $\mathbf{x}_k(m)$ is defined in (1). We assume that the state vector evolves linearly as

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{q}_k \tag{4}$$

where $\mathbf{A}$ is the state transition matrix, and $\mathbf{q}_k$ is the process noise, which is modeled as zero-mean Gaussian with covariance $\mathbf{Q}$. We assume that L FCs try to make estimates of the state vector based on the set of observations they receive during the k-th collection interval. These observations are used as an input to the FCs to estimate the state of the system. Given a general state-space model (4), we use the modified lossy extended Kalman filter (MLEKF) introduced in [16] as the estimation method for the locations of objects.

III. Fusion Methods

The goal of data fusion is to combine the partial estimates provided by a set of L FCs, along with the corresponding uncertainty measures characterized by the estimated covariance matrices, in a manner that provides performance as close as possible to that of the global estimator that has access to all the information.

The optimal linear MMSE (L-MMSE) fusion rule is given by [1]

$$\widehat{\mathbf{x}}_k^{(\mathrm{L-MMSE})} = \left[\sum_{\ell=1}^{L}\left(\boldsymbol{C}_k^{(\ell)}\right)^{-1}\right]^{-1}\sum_{\ell=1}^{L}\left(\boldsymbol{C}_k^{(\ell)}\right)^{-1}\widehat{\mathbf{x}}_k^{\ell} \tag{5}$$

where $\widehat{\mathbf{x}}_k^{\ell}$ and $\boldsymbol{C}_k^{(\ell)}$ are the partial estimate of the state vector and covariance matrix of the ℓ-th FC at time k, respectively. Note that (5) holds regardless of the approach used to derive the partial estimates.

The performance of the linear estimator can be improved by overwriting the state vector of each FC with the final estimate [15], [17]. In other words, the final estimate of the ℓ-th FC is determined as

$$\widehat{\mathbf{x}}_k^{(\ell,F)} = \widehat{\mathbf{x}}_k^{(\mathrm{L-MMSE})}, \quad \ell = 1, \ldots, L \tag{6}$$

This state vector is then used to initialize the next round of estimates in all FCs.

The results in [15] show that overwriting offers a substantial improvement in performance in terms of the localization mean squared error (MSE). However, the overwriting process requires each FC to wait for the state vector of all other FCs before running the next round of estimation. In other words, the underlying assumption in the approaches proposed in [15] is that the exchange of information between the FCs happens instantaneously. In practice, however, there may be a delay that is not negligible with respect to the collection interval,

and thus proper adjustments need to be made to accommodate for this effect.

Clearly, communication between the FCs adds to the overall delay, affecting the way in which the system operates. To be specific, let us distinguish between the collection interval T_c during which the observations are collected and the delay T_d during which the information is exchanged between the FCs. Without loss of generality, we assume that the processing times needed to update the various FCs and perform the fusion are negligible; if not, they can be subsumed into T_c and/or T_d. If T_d is not negligible with respect to T_c, adjustments need to be made to the way in which the information from neighboring FCs is fused and the system is updated. In the sequel, we introduce two delay-tolerant data fusion methods that can address this issue.

A. Soft Fusion with Delay-and-Wait principle (SoftDW)

One way to address the issue of delay is to have the local FC wait for the feedback from neighbors and then perform fusion. In this approach, an FC collects the observations during T_c, updates its local estimate using what existing final state estimate it has, and sends that local estimate to the neighbors. It then waits for T_d. Let us say that $T_d = D \times T_c$, i.e. the delay equals a certain number of collection intervals. Once the neighbors' partial estimates have arrived at the end of the waiting interval, the local FC fuses them with its own. As the local estimate is delayed by the same amount as the neighbors' estimates, there is no timing discrepancy. The local FC thus has an accurate information about the state of the system T_d seconds ago. This information constitutes its final estimate, which will be used at the end of the next collection interval that starts immediately upon fusion. The cycle then repeats: observations are collected during the next T_c, local estimate is updated and sent to the neighbors, feedback is received after T_d seconds and fusion is performed. Fig. 2 illustrates the process of SoftDW, where communication between the ℓ-th FC and its neighbor is shown. The duration of the full cycle is now $T_c + T_d$, and hence the system is updated as often. In other words, estimates are available only every $T_c + T_d$ seconds. Therefore, the equivalent state-transition matrix is $\mathbf{A}' = \mathbf{A}^{D+1}$. The updating rate is thus reduced compared to the case of no delay ($D = 0$). However, the estimation accuracy is not affected, other than by the fact that the system is observed less frequently. One last thing to note is that in order to implement the process, the communication between FCs needs to be synchronous.

B. Soft Fusion with Predict-and-Go principle (SoftPG)

While the above procedure may be well suited to many applications, the question remains open as to whether it is possible to avoid the waiting (during which information is lost), to keep updating the estimates as often as possible, and to produce an estimate of the current state, not the delayed state. The answer is yes, but a more elaborate procedure has to be put in place. Specifically, each FC will now continue to make the partial estimates every T_c seconds, but at the time when fusion is performed, the information available from the neighboring FCs will be outdated. An intelligent fusion scheme will take this fact into account by giving proportionately less importance to the outdated estimates. Equivalently, the estimates available from the neighbors will first be turned into D-step predictions, and these predictions will then be used in fusion. The fusion algorithms thus remain the same, except that they operate with predictions made on the neighbors' estimates and its corresponding covariances. Once the fusion is complete, each FC sends its new information out to the neighbors. The process thus continues in cycles of duration T_c. The estimates now reflect the current state of the system, and the only effect of delay is through the prediction error. Fig. 3 illustrates the process of SoftPG. Note also that communication between the FCs no longer needs to be synchronized.

At the k-th updating interval, the ℓ-th FC uses the measurements received during this interval and generates a partial estimate $\widehat{\mathbf{x}}_k^\ell$ and covariance matrix $\boldsymbol{C}_k^{(\ell)}$. The FCs then exchange the information. We assume that it takes $T_d = D \times T_c$ seconds (or D collection intervals) for the partial estimate to reach the other FCs. The final estimate of the ℓ-th FC at the k-th updating interval is now defined as

$$\widehat{\mathbf{x}}_k^{(\ell,\mathrm{F})} = \left[\left(\boldsymbol{C}_k^{(\ell)}\right)^{-1} + \sum_{i=1,i\neq\ell}^{L}\left(\widehat{\boldsymbol{C}}_k^{(i)}\right)^{-1}\right]^{-1} \times \left(\left(\boldsymbol{C}_k^{(\ell)}\right)^{-1}\widehat{\mathbf{x}}_k^\ell + \sum_{i=1,i\neq\ell}^{L}\left(\widehat{\boldsymbol{C}}_k^{(i)}\right)^{-1}\widehat{\widehat{\mathbf{x}}}_k^i\right) \tag{7}$$

where $\widehat{\widehat{\mathbf{x}}}_k^\ell$ and $\widehat{\boldsymbol{C}}_k^{(\ell)}$ are the D-step prediction of the ℓ-th partial estimate and its covariance matrix at time k, respectively. These are the results of the prediction process which is described in Algorithm 1. As an alternative to the recursive process, we can calculate the values directly from

$$\widehat{\widehat{\mathbf{x}}}_k^\ell = \mathbf{A}^D\widehat{\mathbf{x}}_{k-D}^\ell \tag{8}$$

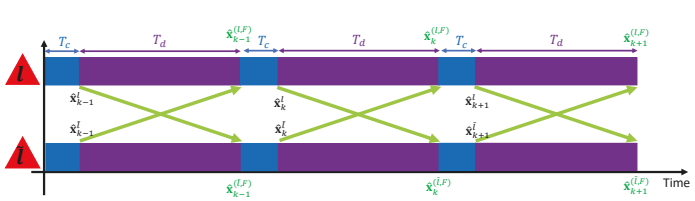

Fig. 2: Soft Fusion with Delay-and-Wait principle.

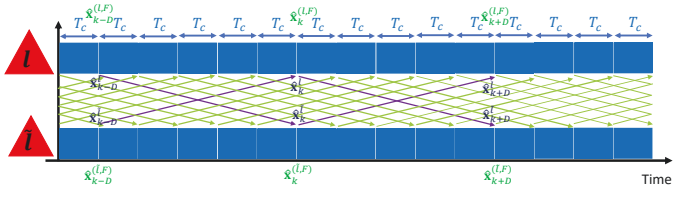

Fig. 3: Soft Fusion with Predict-and-Go principle.

$$\widehat{\boldsymbol{C}}_k^{(\ell)} = \mathbf{A}^D \boldsymbol{C}_{k-D}^{(\ell)} \mathbf{A}^{D^\top} + \sum_{i=0}^{D-1} \mathbf{A}^i \mathbf{Q} \mathbf{A}^{i^\top} \tag{9}$$

IV. Numerical Results

The system parameters are chosen to reflect an underwater scenario with limited acoustic bandwidth. To illustrate the results, we assume an acoustic path loss (see [18]) between objects and sensor nodes with $f = 1\text{kHz}$, $k = 1.5$, $d_{ref} = 100\text{m}$.

The effect of waiting is quantified in Fig. 4. This figure shows the system performance (localization MSE in the delayed estimate) as a function of normalized delay D.

When delay is small enough, SoftDW outperforms SoftPG. In order to understand the reason behind this, we need to specify the sources of error in each method. In SoftDW the source of error is the increased process noise covariance, estimation process as $\mathbf{Q}' = \sum_{i=0}^{D} \mathbf{A}^i \mathbf{Q} \mathbf{A}^{i^\top} \succ \mathbf{Q}$, where for two square matrices $\mathbf{A}$ and $\mathbf{B}$, we say that $\mathbf{A} \succ \mathbf{B}$ if $\mathbf{A} - \mathbf{B}$ is positive definite. In other words, the variance of the process noise increases with D. In SoftPG, on the other hand, the source of error is the prediction process which yields the state vectors and covariance matrices that deviate from the actual values. Depending on which of these sources dominates, one method would perform better than the other.

We note that the effect of delay is not detrimental even for SoftDW for a considerable range of the values D. Moreover, the difference between the performance of the proposed methods and what achieved with no fusion can be as large as 35dB. This proves the superior performance of the proposed methods.

Fig. 5 shows the MSE as a function of ρ_o for various fusion methods. Note that the SoftDW provides new estimates every $T_c + T_d$ compared with SoftPG which generates estimates every T_c seconds. SoftPG performance is almost independent from the density of objects, while the MSE of SoftDW increases slowly with ρ_o.

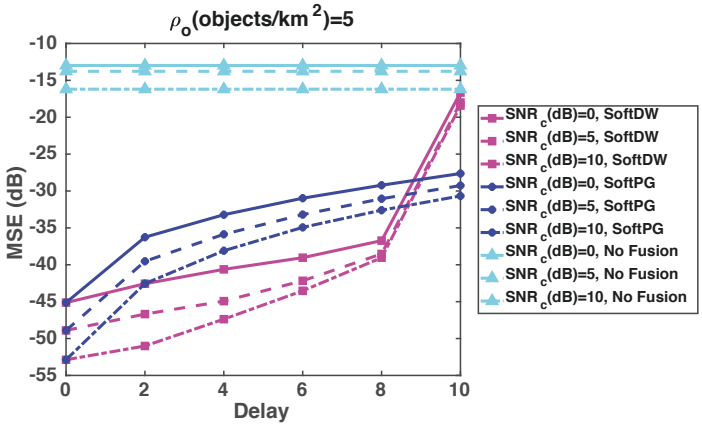

Fig. 4: MSE of different fusion methods as a function of delay for various SNR_c's. $T_c = 1s$. Perfect communication (no packet loss) is considered. The average speed is 5.1m/s. The MSE achieved with no fusion is shown as the benchmark.

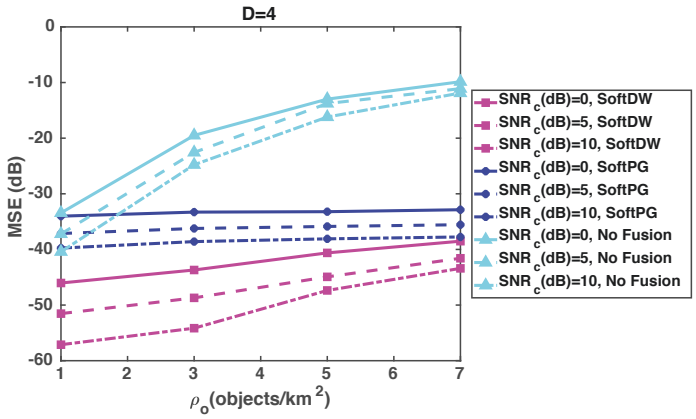

Fig. 5: MSE of different fusion methods as a function of the object density for various SNR_c's. $T_c = 1s$. Perfect communication (no packet loss) is considered. The average speed is 5.1m/s. The MSE achieved with no fusion is shown as the benchmark.

V. Conclusion

We addressed the issue of delay in acoustic sensor networks for object tracking. Specifically, we introduced two data fusion methods for combining the partial estimates of local fusion centers. In the method termed Delay-and-Wait, FCs exchange their state vectors and covariance matrices and wait until they receive all the information from neighboring FCs. They then perform fusion and produce the final estimate. In the method termed Predict-and-Go, FCs constantly update their estimate based on the latest observations and fuse it with the most current estimates available from the neighbors. Delay-and-Wait requires synchronization between FCs and has better performance at smaller values of delay. Predict-and-Go, on the other hand, does not need synchronization and performs better at larger delays. Both methods are able to provide good performance even at delay values as large as ten seconds which is well above the average latency of the geostationary satellite communication ($240-700$ms) that could be a plausible choice for connecting the FCs in practice.

Algorithm 1 Prediction process at time k

Inputs: $\widehat{\mathbf{x}}_{k-D}^{\ell}$ and $\boldsymbol{C}_{k-D}^{(\ell)}$ for all $\ell = 1, \ldots, L$
for $\ell = 1, \ldots, L$ **do**
  $\widehat{\widehat{\mathbf{x}}}_k^{\ell} \leftarrow \widehat{\mathbf{x}}_{k-D}^{\ell}$
  $\widehat{\boldsymbol{C}}_k^{(\ell)} \leftarrow \boldsymbol{C}_{k-D}^{(\ell)}$
  $j \leftarrow 0$
  while $j < D$ **do**
    $\widehat{\widehat{\mathbf{x}}}_k^{\ell} \leftarrow \mathbf{A} \widehat{\widehat{\mathbf{x}}}_k^{\ell}$
    $\widehat{\boldsymbol{C}}_k^{(\ell)} \leftarrow \mathbf{A} \widehat{\boldsymbol{C}}_k^{(\ell)} \mathbf{A}^\top + \mathbf{Q}$
    $j \leftarrow j + 1$
end
Outputs: $\widehat{\widehat{\mathbf{x}}}_k^{\ell}$ and $\widehat{\boldsymbol{C}}_k^{(\ell)}$ for all $\ell = 1, ..., L$

REFERENCES

[1] D. Luengo, L. Martino, V. Elvira, and M. Bugallo, "Efficient linear fusion of partial estimators," *Digital Signal Processing*, vol. 78, pp. 265–283, 2018.

[2] G. B. Giannakis, F. Bach, R. Cendrillon, M. Mahoney, and J. Neville, "Signal processing for big data [from the guest editors]," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 15–16, 2014.

[3] F. Lavancier and P. Rochet, "A general procedure to combine estimators," *Computational Statistics & Data Analysis*, vol. 94, pp. 175–192, 2016.

[4] K. F. Wallis, "Combining forecasts–forty years later," *Applied Financial Economics*, vol. 21, no. 1-2, pp. 33–41, 2011.

[5] T. Li, H. Fan, J. García, and J. M. Corchado, "Second-order statistics analysis and comparison between arithmetic and geometric average fusion: Application to multi-sensor target tracking," *Information Fusion*, vol. 51, pp. 233–243, 2019.

[6] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006.

[7] J.-J. Xiao, A. Ribeiro, Z.-Q. Luo, and G. B. Giannakis, "Distributed compression-estimation using wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 27–41, 2006.

[8] A. Swami, Q. Zhao, Y.-W. Hong, and L. Tong, *Wireless sensor networks: signal processing and communications perspectives*. John Wiley & Sons, 2007.

[9] M. Cetin, L. Chen, J. W. Fisher, A. T. Ihler, R. L. Moses, M. J. Wainwright, and A. S. Willsky, "Distributed fusion in sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 42–55, 2006.

[10] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[11] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[12] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE transactions on signal processing*, vol. 58, no. 3, pp. 1035–1048, 2009.

[13] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[14] R. P. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic systems*, vol. 39, no. 4, pp. 1152–1178, 2003.

[15] M. Alimadadi, M. Stojanovic, and P. Closas, "Object tracking in random access networks: A large scale design," *To appear in IEEE Internet of Things Journal*, 2020.

[16] ——, "Object tracking using modified lossy extended Kalman filter," in *Proc. Int. Conf. Underwater Netw. & Syst.* ACM, 2017, p. 7.

[17] ——, "Object tracking in random access sensor networks: Extended Kalman filtering with state overlapping," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 927–932.

[18] P. Qarabaqi and M. Stojanovic, "Statistical characterization and computationally efficient modeling of a class of underwater acoustic communication channels," *IEEE Journal of Oceanic Engineering*, vol. 38, no. 4, pp. 701–717, 2013.