Distributed Nonlinear Estimation for Robot Localization

Andrea Simonetto, Tamás Keviczky and Robert Babuška

Abstract—Distributed linear estimation theory has received increased attention in recent years due to several promising industrial applications. Distributed nonlinear estimation, however, is still a relatively unexplored field despite the need in numerous practical situations for techniques that can handle nonlinearities. This paper presents a unified way of describing distributed implementations of three commonly used nonlinear estimators: the Extended Kalman Filter, the Unscented Kalman Filter and the Particle Filter. Leveraging on the presented framework, we propose new versions of these methods, which are shown to outperform the few published ones in two robot localization test cases.

I. INTRODUCTION

Driven by the numerous potential applications in sensor networks, distributed estimation techniques have been studied with growing interest in the past few years. So far, the research has been mainly focused on state estimation of linear dynamical systems. References [1], [2], [3] and [4] give a comprehensive overview of the field.

There are many situations in which such a framework cannot be applied, due to the nonlinearities in the dynamical system or in the sensing equation or both. An important example of these scenarios is the localization of a moving object via range-only measurements. This particular problem arises in applications such as indoor robot localization [5], [6], [7], underwater sensor networks [8], [9], [10] and space exploration [11] among others.

Although there are a few cases in the literature in which distributed nonlinear estimation has been addressed, a clear performance evaluation is still missing. For instance in [3] a distributed Extended Kalman Filter is suggested but not implemented, and in [12] a distributed Particle Filter is proposed without extensive analysis.

The aim of this paper is to propose effective distributed algorithms for nonlinear estimation. To achieve this, we introduce a unified framework, in which we design improved versions of both distributed Extended Kalman Filters and distributed Particle Filters. Moreover, we propose an algorithm for the distributed Unscented Kalman filter. The core of the framework is a merging mechanism based on a weighted consensus procedure similar to the ones developed in [13]. The main difference is the chosen consensus matrix, which ensures local optimality at each step.

The paper is organized as follows: Section II defines the problem, in Section III and IV the merging mechanism and the estimation algorithms are proposed, Section V presents two test cases and in Section VI the conclusions are drawn.
and a distributed merging mechanism which aggregates the different estimates.

Remark 1: The graph \( G \) is assumed to be fixed in time. However, with weak technical conditions, we could extend the present algorithms to situations in which the graph \( G \) is time-varying, see for instance [13].

C. Distributed localization

Distributed localization via range-only measurements is one example of distributed nonlinear estimation where the process is locally unobservable by the individual sensors. Let \( x_p \) be the position of a generic object, a robot for instance. Let \( s_i, i = 1, \ldots, N \) be the position of the fixed sensors. The measurement equation can be written as:

\[
Z_i = ||x_p(k) - s_i|| + v_i(k), \quad i = 1, \ldots, N
\]  

(3)

Since the state is locally unobservable, the sensors have to communicate with each other. In the following section we will present some possibilities for combining their estimates.

III. CONSENSUS ALGORITHMS

If requirement R1) is satisfied by the choice of a suitable local nonlinear filter, requirement R2) has to be imposed using a consensus algorithm which merges the different estimates.

Consensus algorithms make use of linear maps between some local variables, for example the estimates \( x_j(k) \)’s with \( j \in J_i \), and their weighted nodal average \( \bar{x}_i(k) \) as:

\[
\bar{x}_i(k) = \sum_{j \in J_i} W_{ij} x_j(k)
\]

or, in matrix-vector notation

\[
\bar{x}(k) = W \otimes I_n x(k)
\]  

(4)

where \( \otimes \) represents the Kronecker product. Repeating the update equation (4) constitutes a consensus iteration. It is straightforward to note that the \( \tau \)-th iteration is computed as \( W^\tau \otimes I_n x(k) \). The matrix \( W \) is required to satisfy:

\[
\lim_{\tau \to \infty} W^\tau = \frac{1}{N} 11^T
\]  

(5)

so that the consensus iterations not only converge but they also give the mean of the initial values. This property has been used in the last few years [14] as a means of averaging the different estimates \( \hat{x}_j(k) \), see also [3] and [12]. A typical form for \( W \) is

\[
W = I_N - \epsilon \mathcal{L}
\]

where \( \mathcal{L} \) is the weighted Laplacian associated with the graph \( G \), \( \epsilon \) is a positive constant which has to be less than one to ensure convergence. In its typical implementation, a consensus algorithm merges the different \( \bar{x}_i(k) \) as

\[
\bar{x}_i = \bar{x}_i + \epsilon/(N_i - 1) \sum_{j \in N_i} (\bar{x}_j - \bar{x}_i)
\]  

(6)

delivering for \( \tau \to \infty \), \( \forall i \):

\[
\bar{x} = \bar{x}_i = \frac{1}{N} \sum_{j \in V} \bar{x}_j
\]

Quite often, the number of consensus iterations is finite (\( \tau \ll \infty \)), or even 1 for instance in [3]. This reduces drastically the communication among the sensors, losing however the convergence property. Detailed analysis of such interleaved schemes is still an open problem; see [15] for a stability/convergence proof applied to a particular case.

We propose to use a weighted version of the typical algorithm (6), similar to [13]. First we note that:

Lemma 1 ([13]): Given a set of independent and unbiased estimates, \( \hat{x}_j \), with associated covariance matrices, \( P_j \), where \( i \in V \), the following weighted averaging:

\[
\hat{x} = \left( \sum_{j \in V} P_j^{-1} \right)^{-1} \sum_{j \in V} P_j^{-1} \hat{x}_j
\]

\[
P^{-1} = \sum_{j \in V} P_j^{-1}
\]

gives the linear minimum-variance unbiased estimate of \( x \).

Second, the previous weighted averaging can be seen as

\[
\hat{x} = Y^{-1} Z \quad P^{-1} = NY
\]

where \( Z_i = P_i^{-1} \hat{x}_i, Y_i = P_i^{-1}, Z = (1/N) \sum_{j \in V} Z_j, Y = (1/N) \sum_{j \in V} Y_j \). Therefore \( Z \) and \( Y \) can be calculated as standard averaging. Hence we can implement a consensus iteration, in the form:

\[
\bar{Z}_i = (1/N_i) \sum_{j \in J_i} Z_j
\]
\[
\bar{Y}_i = (1/N_i) \sum_{j \in J_i} Y_j
\]  

(7)

which has iteration matrix \( W \), where \( W_{ij} = 1/N_i \) if and only if \( j \in J_i \), and \( W_{ij} = 0 \) otherwise. \( W \) can be proven to satisfy (5).

As a final step, we note that, for this particular \( W \), even for \( \tau = 1 \), the couple \( (\bar{x}_i(k), \bar{P}_i(k)) \) will be optimal in the sense of Lemma 1, even though only w.r.t. the neighborhood. Hence we can assure a local optimality of the scheme. Algorithm 1 shows the method for each sensor \( i \).

Algorithm 1 MERGE algorithm

1: Input: \( \hat{x}_j(k), P_j(k), j \in J_i \)
2: Merge:

\[
\bar{x}_i(k) = \left( \sum_{j \in J_i} P_j^{-1}(k) \right)^{-1} \sum_{j \in J_i} P_j^{-1}(k) \hat{x}_j(k)
\]
\[
\bar{P}_i^{-1}(k) = \sum_{j \in J_i} P_j^{-1}(k)
\]
3: Output: \( \bar{x}_i(k), \bar{P}_i(k) \)

IV. DISTRIBUTED NONLINEAR ESTIMATION

Given the premises of the previous sections, our general distributed nonlinear estimation framework will consist of \( N \) different local filters, connected to the merge/consenus
A. Distributed Extended Kalman Filters

Let $F_i$, $G_i$ and $H_i$ be respectively:
\[
F_i = \frac{\partial f(x(k), u(k), w(k))}{\partial x(k)} \mid _{(x_i(k), u(k), 0)},
\]
\[
G_i = \frac{\partial f(x(k), u(k), w(k))}{\partial v(k)} \mid _{(x_i(k), u(k), 0)},
\]
\[
H_i = \frac{\partial g(x(k))}{\partial x(k)} \mid _{(x_i(k))}.
\]

Let $Q = E[w(k)w^T(k)]$, $R_i = E[v_i(k)v_i^T(k)]$. Define the weighted predicted observation vector $O_i$, the weighted true observation vector $y_i$, and their nodal aggregates:
\[
O_i \triangleq H_i^T R_i^{-1} g(F_i \hat{x}_i(k)),
\]
\[
\overline{O}_i \triangleq \sum_{j \in \mathcal{J}_i} O_j
\]
\[
y_i \triangleq H_i^T R_i^{-1} z_i(k+1),
\]
\[
\overline{y}_i \triangleq \sum_{j \in \mathcal{J}_i} y_j
\]

whose differences $(O_i - y_i)$ and $(\overline{O}_i - \overline{y}_i)$ represent the mismatch between the prediction and the measurements. Define the inverse-covariance matrix $S_i$ and its nodal aggregate:
\[
S_i \triangleq H_i^T R_i^{-1} H_i, \quad \overline{S}_i \triangleq \sum_{j \in \mathcal{J}_i} S_j
\]

The distributed Extended Kalman Filter algorithm of [3] implements a typical consensus iteration. For each sensor it contains the following two steps:

**Step 1. (Prediction):**
\[
\begin{align*}
\hat{x}_i(k+1) &= F_i \hat{x}_i(k) \\
\hat{P}_i(k+1) &= F_i \hat{P}_i(k) F_i^T + G_i Q G_i^T
\end{align*}
\]

**Step 2. (Update):**
\[
\begin{align*}
(\hat{P}_i(k+1))^{-1} &= (\hat{P}_i(k+1))^{-1} + \overline{S}_i \\
\hat{x}_i(k+1) &= \hat{x}_i(k+1) + P_i(k+1) (\overline{y}_i - \overline{O}_i) + \hat{x}_i(k+1) - \hat{x}_i(k+1)
\end{align*}
\]

The proposed version has a different update step and makes use of Algorithm 1:

**Step 1. (Prediction):** same as in (8)

**Step 2. (Modified Update):**
\[
\begin{align*}
(\hat{P}_i(k+1))^{-1} &= (\hat{P}_i(k+1))^{-1} + S_i \\
\hat{x}_i(k+1) &= \hat{x}_i(k+1) + P_i(k+1) (y_i - O_i)
\end{align*}
\]

**Step 3. (Merge):**
\[
(\hat{x}_i(k+1), \hat{P}_i(k+1)) = \text{MERGE}(\hat{x}_j(k+1), P_j(k+1)), \quad j \in \mathcal{J}_i
\]

Note that the consensus algorithm in (9) has a similar form as (6). In addition, $y_i$, $O_i$, and $S_i$ are used in the modified update step rather than $\overline{y}_i$, $\overline{O}_i$, and $\overline{S}_i$. The reasons for this are two-fold: we aim to decrease the amount of data that the sensors send to each other, and more importantly, we need to have independent estimates $\hat{x}_i$ in order to apply Lemma 1. Hence, the final new algorithm consists of a modified local prediction-update step, followed by a merge step.

B. Distributed Unscented Kalman Filters

Let
\[
(\hat{x}_i(k+1), P_i(k+1)) = \text{UKF}_i(\hat{x}_i(k), \hat{P}_i(k), z_i(k+1))
\]

be the local Unscented Kalman Filter, whose formulation can be found for example in [16]. Then, the Distributed UKF algorithm is shown in Algorithm 2.

**Algorithm 2 DUKF algorithm**

1. $\hat{x}_i(0), P_i(0)$
2. while new data exists do
3. Local UKF:
   \[
   (\hat{x}_i(k+1), P_i(k+1)) = \text{UKF}_i(\hat{x}_i(k), \hat{P}_i(k), z_i(k+1))
   \]
4. Merge:
   \[
   (\hat{x}_i(k+1), \hat{P}_i(k+1)) = \text{MERGE}(\hat{x}_j(k+1), P_j(k+1)), \quad j \in \mathcal{J}_i
   \]
5. end while

As it can be seen, the structure fits in the same framework as before, using a local UKF filter and a merge step which processes the different estimates and their covariances.
C. Distributed Particle Filters

Distributed Particle Filters are a rather unexplored research field. The main idea behind them is that local particle filters agree upon a common proposal distribution, from which to draw the particles. A comprehensive overview of standard Particle Filters can be found in [17], while a survey of recent developments in distributed Particle Filters is given in [18]. For our purposes the formulation of [12] can be considered as the state-of-the-art in this field.

The key concept is to use a Gaussian proposal distribution such as:

\[ q(x(k+1)|x(k), z(k+1)) = \mathcal{N}(\mu(k), \Sigma(k)) \]

where \( \mathcal{N} \) represents a normal distribution with mean \( \mu(k) \) and covariance \( \Sigma(k) \). The couple \((\mu(k), \Sigma(k))\) is calculated by propagating \((\hat{x}(k), P(k))\) via an Unscented Transformation, UT, as in [12]:

\[ (\mu(k), \Sigma(k)) = UT(\hat{x}(k), P(k)) \]

whereas the couple \((\hat{x}(k), P(k))\) can be approximated via consensus using the local couples \((\hat{x}_i(k), P_i(k))\). These can be computed by each local Particle Filter (at the preceding time instant) in the following way:

\[
\hat{x}_i(k) = \sum_{j=1}^{m} \omega_{i,j} x_i(k)^j \tag{12}
\]

\[
P_i(k) = \sum_{j=1}^{m} \omega_{i,j} x_i(k)^j - \hat{x}_i(k) (x_i(k)^j - \hat{x}_i(k))^T \tag{13}
\]

where \( j \) represents the particle index, \( m \) the number of particles, \( x_i(k)^j \) and \( \omega_{i,j} \) respectively the state and the weight of the \( j \)-th particle for the \( i \)-th sensor at time \( k \). In the formulation of [12], the global couple \((\hat{x}(k), P(k))\) is approximated via a consensus algorithm in the form of (4):

\[
\hat{x}_i(k) = \hat{x}_i(k) + \epsilon/(N_i - 1) \sum_{j \in N_i} (\hat{x}_j(k) - \hat{x}_i(k))
\]

\[
P_i(k) = P_i(k) + \epsilon/(N_i - 1) \sum_{j \in N_i} (P_j(k) - P_i(k))
\]

Therefore, after the consensus iteration each local PF has the new proposal distribution:

\[ q(x(k+1)|x(k), z(k+1)) = \mathcal{N}(\mu_i(k), \Sigma_i(k)) \]

with

\[ (\mu_i(k), \Sigma_i(k)) = UT(\hat{x}_i(k), P_i(k)) \]

In our formulation, we will use the \textsc{Merge} algorithm instead of the standard consensus algorithm, thus:

\[ (\hat{x}_i(k), P_i(k)) = \textsc{Merge}(\hat{x}_j(k), P_j(k)), \quad j \in J_i \]

Algorithm 3 presents the proposed modified method. Note that PF indicates the local Particle Filter.

Remark 2: Note that in the paper of [12] the choice of \( \tau \) for the consensus algorithm is left to the user as a parameter. We will assume \( \tau = 1 \) to compare it with our scheme.

Algorithm 3 DPF algorithm

1: \( x_i(0), P_i(0) \)
2: while new data exists do
3: \quad \text{LOCAL FILTER} \{
4: \quad \text{a. Propagation of } (\hat{x}_i(k), P_i(k)): \quad (\mu_i(k), \Sigma_i(k)) = UT(\hat{x}_i(k), P_i(k))
5: \quad \text{b. Local PF with proposal distribution } \mathcal{N}(\mu_i(k), \Sigma_i(k)): \quad (x_i(k+1)^j, \omega_{i,k+1}^j) = \text{PF}(\mu_i(k), \Sigma_i(k), z_i(k+1))
6: \quad \text{c. Compute } (\hat{x}_i(k+1), P_i(k+1)) \text{ via (12) - (13)}
7: \quad \text{d. Merge:} \quad (\hat{x}_i(k+1), P_i(k+1)) = \text{Merge}(\hat{x}_j(k+1), P_j(k+1), j \in J_i)
8: \} \text{ end while}

V. Test Cases

In the next sections we will present two different test cases to analyze the proposed algorithms. First, we consider the localization problem of a unicycle robot in a 2D environment. This is representative of an experimental setup currently under development by the authors. Second, we estimate the state of an autonomous underwater vehicle, which can represent a scaled model of many existing underwater robotic platforms, see for example [9]. In both cases we define the error of the sensor \( i \) at time \( k \), \( e_i(k) \), as the distance between the true position at that time and the one estimated by the sensor \( i \). Let the mean error \( e_m \) and the maximum error \( e_M \) be:

\[ e_m = \frac{1}{NT} \sum_{i=1}^{N} \sum_{k=0}^{T} e_i(k), \quad e_M = \max_{i} \frac{1}{T} \sum_{k=0}^{T} e_i(k) \]

where \( T \) is the final time of simulation. Let the average error and the average maximum error be respectively the mean error averaged on a number of different simulations and the maximum error averaged on the same number of different simulations.

A. Unicycle robot

The state of the unicycle robot is chosen as \( x = (x^p, \theta)^T = (\xi, \eta, \theta)^T \), where \( x^p \in \mathbb{R}^2 \) is the position and \( \theta \) is the orientation. The dynamical equations are:

\[ \begin{align*}
\xi(k+1) &= \xi(k) + \frac{\hat{v}}{\rho} (\sin(\theta(k) + \hat{\delta}t) - \sin(\theta(k))) \\
\eta(k+1) &= \eta(k) - \frac{\hat{\delta}}{\rho} (\cos(\theta(k) + \hat{\delta}t) - \cos(\theta(k))) \\
\theta(k+1) &= \theta(k) + \hat{\delta}t + \gamma t \\
\hat{v} &= v + n_v \\
\hat{\delta} &= r + n_r
\end{align*} \]

where \( v \) and \( r \) are the velocity and the angular velocity control inputs, respectively, and \( n_v, n_r, \text{ and } \gamma \) are noise terms. More details about the model can be found in [19]. We assume to have \( N = 15 \) sensors sparsely distributed in the environment. The simulation parameters are \( \Delta = 1 \text{ s}, \)
Furthermore, in this situation, the proposed DUKF is more accurate than the DPF of [12]. This can be exploited in cases where we need both high sampling frequency and high accuracy.

B. Autonomous underwater vehicle

As a second test case we simulate the localization problem of an autonomous underwater vehicle (AUV). This is a relevant example because its main features introduce more difficulties in the nonlinear estimation problem. First of all, the sensors are usually located on the sea floor, a configuration in which the error in the depth direction is considerably greater than in the other directions. This can easily cause the divergence of estimation techniques. In addition, the environment is more noisy than in the unicycle case. The state of the AUV is chosen as $x = (x_p^T, v_p^T)\text{T}$, where $x_p \in \mathbb{R}^3$ is the position and $v_p \in \mathbb{R}^3$ is the velocity. The dynamical equations are:

\[
\begin{align*}
x_p(k+1) &= x_p(k) + v_p(k)\Delta t \\
v_p(k+1) &= v_p(k) + \frac{\Delta t}{m} (\hat{u} - \alpha \|v_p(k)\| v_p(k)) \\
\hat{u} &= u + n_u
\end{align*}
\]

where $m$ is the mass of the vehicle, $\alpha$ is a drag coefficient, and $n_u$ is a noise term. We assume to have $N = 25$ sensors sparsely distributed at varying heights from a plane surface. The different heights simulate an uneven seafloor. We take $\Delta t = 1$ s, $T = 130$ s, $m = 1$ kg, $\alpha = 1$ kg/s, $\|\text{mean}(u)\| \sim 0.5$ N and $\text{std}(n_u) = (0.05, 0.05, 0.025)^T$ N, where $\text{mean}(\cdot)$ is the mean operator and $\text{std}(\cdot)$ is the standard deviation. We assume that the measurement error in equation (2) is $\text{std}(v_i) = 0.1$ m, for all the sensors. We consider 500 particles for the DPF. We run 400 different simulations, varying randomly the position of the sensors. For the chosen communication range, in mean, each sensor has 4 neighbors. In Figure 2 an example of the simulation results is shown. In Figure 3 the comparison between the proposed algorithms and the ones in the literature is depicted. The average error is based on the 200 different simulations. The I bars represent the standard deviation. The squares represent the average maximum error of the sensors. The results show that the proposed algorithms outperform those in the literature.
the average error of the proposed algorithms versus the second smallest eigenvalue of the graph Laplacian (also called the algebraic connectivity). The average error is based on 400 simulations. The second smallest eigenvalue, denoted as $\lambda_2$, or its normalized counterpart, $\lambda_2/\lambda_{2,\text{max}}$, dictate the convergence rate of any consensus algorithms. Values of $\lambda_2/\lambda_{2,\text{max}}$ close to 0 represent graphs which are not highly connected, leading to more distributed estimation problems. Values of $\lambda_2/\lambda_{2,\text{max}}$ near 1 reveal highly connected graphs, thus estimation problems close to the centralized case. Here $\lambda_{2,\text{max}}$ is the maximum over the graphs from the 400 simulations. In Figure 5 a dot at the coordinate $(\phi, \psi)$, represents that the graphs with $\lambda_2/\lambda_{2,\text{max}} \in (\phi - 0.05, \phi + 0.05)$ have an average error of $\psi$. The shaded areas show the standard deviation of these errors. Note that the DEKF estimations are not depicted here because they do not converge. The DUKF is shown without the standard deviation to make the graph more readable, its value is on the order of 0.2 m. The results show that the proposed algorithms outperform the DPF found in the literature. In particular, they show that, in this case, the DUKF has similar average error with the literature DPF. This is important because DUKF is less computationally expensive than DPF and therefore this fact is crucial in the context of designing fast yet accurate algorithms. Moreover, the proposed DPF has the minimum achievable average error even when $\lambda_2/\lambda_{2,\text{max}} \rightarrow 0$.

VI. CONCLUSIONS

We proposed an effective scheme to distribute the non-linear estimation problem among different sensing units. We applied the method to a localization with range-only measurements, designing distributed Extended Kalman Filters, distributed Unscented Kalman Filters and distributed Particle Filters. We have shown how the proposed algorithms outperform the ones found in the literature.

As future work we plan to implement the scheme in a real robotic testbed which is currently under development. More-over we will extend the formulation to multi-robot settings, in which some of the sensors are moving with the robots themselves. Another interesting idea that can be investigated is the case in which the local filters are different: some of them can be UKFs and others PFs. This is especially practical when the sensors have different computational capabilities.

REFERENCES


