Distributed Robust Consensus using RANSAC and Dynamic Opinions

Eduardo Montijano, Sonia Martínez and Carlos Sagues

Abstract— **Sensor networks must be able to fuse nodes' perceptions in a reliable way in order to reach a trustworthy consensus. Data association mistakes and measurement outliers are some of the factors that can contribute to incorrect perceptions and considerably affect consensus values. In this paper, we present a novel distributed scheme for robust consensus in autonomous sensor networks. The proposed method builds on random sampling consensus to exploit measurement redundancy, and enables the network to determine outlier observations with local communications. To do this, different hypotheses are generated and voted for using distributed averaging. In our approach, nodes can change their opinion as the hypotheses are computed, making the voting process dynamic. Assuming that enough hypotheses are generated to have at least one composed exclusively by inliers, we show that the method converges to the maximum likelihood of all the inlier observations under some natural conditions. We present several simulations and examples with real information that demonstrate the good performance of the proposed algorithm.**

Index Terms - Robust Data Fusion, Distributed Consensus, Outlier rejection, RANSAC.

I. Introduction

Recent advances in communication, computation and control are leading to a new generation of autonomous systems that can execute complex tasks while interacting over dynamic networks. To exploit their full potential, an intensive research is being devoted to the development of decentralized algorithms that enable the scalable management of these networks. Within the control and robotics communities, a canonical problem that has received much attention is that of consensus; see e.g., [1] and references therein. In this problem, the goal is to devise a distributed algorithm that allows a group of agents to agree upon some specific information. Several algorithms have been proposed to achieve consensus in different situations such as directed communications [2], time-varying topologies [3], [4], or fast convergence [5]. One of the major drawbacks of the current approaches is their lack of robustness to outliers. That is, the consensus value can be severely affected by wrong sensor measurements and information completely unrelated to the quantity that the robots would like to measure. In this manuscript, we address this issue by means of random sampling techniques.

This work was supported by the projects DPI2009-08126, DPI2012-32100, CUD2013-05, NSF-0712746, NSF-0930919 and AFOSR-11RSL548.

E. Montijano is at Centro Universitario de la Defensa (CUD) and Instituto de Investigación en Ingeniería de Aragón (I3A), Zaragoza, Spain, emonti@unizar.es

S. Martínez is at the Department of Mechanical and Aerospace Engineering, Univ. of California, San Diego, soniamd@ucsd.edu

C. Sagues is at Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain, csagues@unizar.es

Faulty agents implementing a consensus algorithm have been considered in [6]–[10]. In these papers, failing nodes introduce additive errors in the consensus algorithm, either because they are unable to update it correctly or because they are malicious. These works then propose the use of linear detection filters that the well-behaving nodes can employ to identify the faulty ones. The algorithms are guaranteed to work provided the number of well-behaving nodes is sufficiently large, and the connectivity among them is high enough. A main drawback of these approaches is the requirement of a global knowledge of the network topology by all nodes, which has to be essentially non-switching. In this way, each node keeps an estimate of the state of all other nodes in the network, which does not scale well if the network size is very large. In [11], this assumption is dropped by introducing several network operators who supervise a subnetwork each. However, the precise detection of arbitrarily faulty agents still requires the implementation of a non-scalable number of filters, one per subset of potentially failing nodes. Finally, these papers do not adequately address the problem of how to remove the influence of "spurious" initial conditions from the final consensus value, which is a main focus of this work.

Distributed robustness to outliers has been recently studied in [12] and the problem of robust acoustic localization with non-gaussian noise is analyzed in [13]. These works present distributed outlier detection solutions that exploits a distributed optimization problem formulation. The solutions are distributed and scalable as agents are not required to know the network topology, and is implementable over switching communication graphs. However, these algorithms can not handle properly multi-modal distributions, which can lead to different clusters of good measurements.

Finally, a connection can be made to works dealing with noise and quantized messages, e.g., [14]–[16]. These solutions analyze the influence of sending messages that have finite precision and how the exact final consensus value can still be reached. Unlike these previous algorithms, our iteration does not employ quantized information feedback from neighbors. Instead of that, we exploit the quantized nature of vote counting to propose a consensus iteration that converges in a finite number of iterations to the exact number of votes.

In this paper, we depart from the above problem scenarios and techniques in the following way. First, nodes are considered to be faulty or "outliers" (as opposed to "inliers") if their initial measurements are out of pattern by the rest of

the data.[1] Other than this, all nodes are able to implement consensus protocols correctly and with infinite precision, are not malicious, and collaborate together to arrive at a good average observation value. Thus, the need of keeping an estimate of other nodes' states can be avoided. Additionally, we exploit the quantized nature of the initial conditions to reach the average in finite time.

The main contribution of this paper is a novel and scalable robust distributed consensus method which identifies outlier information and achieves consensus by discarding it. Our approach integrates random sampling principles [17] with those of distributed averaging [18]. In this way, we propose two algorithms which can be implemented by agents without knowledge of the network topology, and tolerate switching network interactions. The first one, DISTRIBUTED STATIC-OPINION RANSAC algorithm integrates the creation of best hypotheses, voting process, and creation of a better model sequentially. Due to random sampling, the method has a high tolerance to large number of outliers and can be adapted to deal with multi-modal distributions. In the DISTRIBUTED DYNAMIC-OPINION RANSAC algorithm, nodes are further allowed to change their opinion online, making the voting process dynamic and leading to the simultaneous execution of the aforementioned steps for the first algorithm. We present reasonable sufficient conditions that guarantee the dynamic process converges and preserves the good properties of random sampling, which, in particular, do not depend on how the outliers or inlier nodes are connected among them. More precisely, our algorithm scalability and success depend on the probability of randomly picking a node that is an "inlier"; i.e. the *inlier ratio*, which is independent of the number of nodes in the network.

Preliminary versions of this paper were presented in [19], [20]. In this version we include the proofs of all the theoretical results, omitted in the conference versions, and we present three other contributions: (i) An extension of the averaging rule to guarantee convergence in finite time when the consensus value has finite precision. (ii) A distributed averaging primitive to compute the number of active nodes in a network. (iii) Analysis of the performance of the algorithm on the following relevant application scenarios: distributed sensor-network event localization and distributed people identification by camera networks. In this way, our algorithm expands the utility of random sampling by adding a new set of applications where it can be used by sensor networks such as robust localization [21], face recognition [22], distributed labeling [23], multi-robot mapping [24], or collaborative sensor-bias calibration [25].

The remainder of the paper is organized as follows: Section II briefly reviews a consensus algorithm for Maximum Likelihood approximation and presents the main objective of this paper. A distributed algorithm considering static opinions is described in Section III. In Section IV we extend this algorithm to allow the nodes change their opinion. A finite time rule to count the number of votes is explained in Section V. Simulation results are reported in Section VI and finally, the conclusions of the work are presented in Section VII.

[1]These are data with arbitrarily large errors. A more precise definition will be given in Section II

II. PRELIMINARIES AND OBJECTIVE

We consider a network of N nodes labeled by $i \in \mathcal{V} = \{1, \dots, N\}$. At each time $t > 0$, communications among nodes are modeled by an undirected graph $\mathcal{G}(t) = \{\mathcal{V}, \mathcal{E}(t)\}$, where $\mathcal{E}(t) \subset \mathcal{V} \times \mathcal{V}$ represents the edge set at time t. Thus, nodes i and j can communicate at time t if and only if $(i, j) \in \mathcal{E}(t)$. The neighbors of node $i \in \mathcal{V}$ at time $t > 0$ are those that can directly communicate with it; i.e., $\mathcal{N}_i(t) = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}(t)\}$.

***Assumption** 2.1 (Periodic Connectivity):* There exists an integer $T > 0$ such that, for any instant of time $t \geq 0$, the graph $\mathcal{G}(\mathcal{V}, \mathcal{E}(t) \cup \mathcal{E}(t+1) \cup \ldots \cup \mathcal{E}(t+T))$ is connected.

Let $\boldsymbol{\theta} \in \mathbb{R}^d$ be a set of attributes that represent an object of interest for the network. These attributes can be, for example, the position of the object in a world coordinate frame, its shape and color, or a set of descriptors that identify it. Each node makes some noisy measurement of $\boldsymbol{\theta}$, say $\mathbf{x}_i \in \mathbb{R}^d$, subject to uncertainty characterized by the symmetric, semi-definite positive covariance matrix $\boldsymbol{\Lambda}_i \in \mathbb{R}^{d \times d}$, for $i \in \{1, \dots, N\}$. We denote by $CH(\mathcal{V})$, the convex hull of the observations $\mathbf{x}_i$.

The maximum likelihood (ML) of $\boldsymbol{\theta}$, $\boldsymbol{\theta}_{\text{ML}}$, is estimated using a weighted least-squares approximation from the node measurements as

$$\boldsymbol{\theta}_{\text{ML}} = (\Sigma_{i=1}^{N} \boldsymbol{\Lambda}_i^{-1})^{-1} \Sigma_{i=1}^{N} \boldsymbol{\Lambda}_i^{-1} \mathbf{x}_i. \tag{1}$$

A distributed algorithm to compute the ML can be found in [18]. By means of this, each node first initializes two state variables, $\mathbf{P}_i$ and $\mathbf{q}_i$, as

$$\mathbf{P}_i(0) = \boldsymbol{\Lambda}_i^{-1}, \quad \mathbf{q}_i(0) = \boldsymbol{\Lambda}_i^{-1} \mathbf{x}_i. \tag{2}$$

Then, at each iteration, they update these variables using

$$\begin{aligned} \mathbf{P}_i(t+1) &= \mathbf{P}_i(t) + \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t)(\mathbf{P}_j(t) - \mathbf{P}_i(t)), \\ \mathbf{q}_i(t+1) &= \mathbf{q}_i(t) + \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t)(\mathbf{q}_j(t) - \mathbf{q}_i(t)), \end{aligned} \tag{3}$$

where $\mathbf{A}(t) = [a_{ij}(t)]$ are the weight matrices associated with $\mathcal{G}(t)$, satisfying the following assumption:

***Assumption** 2.2 (Symmetry and Row Stochasticity):* The matrices $\mathbf{A}(t)$ satisfy the following conditions $\forall t$:

$$\begin{cases} a_{ij}(t) \in \{0\} \cup [\alpha, 1], \\ a_{ii}(t) = 1 - \sum_{j \neq i} a_{ij}(t) \geq \alpha, \quad \forall i, j, t, \\ \mathbf{1}^T \mathbf{A}(t) = \mathbf{1}^T, \quad \mathbf{A}(t)\mathbf{1} = \mathbf{1}, \end{cases}$$

where α is some positive constant and $\mathbf{1} = (1, \dots, 1)^T \in \mathbb{R}^d$. In [18] it is shown that under the Periodic Connectivity Assumption 2.1 and the Symmetry and Row-stochasticity Assumption 2.2, all the nodes in the network reach the average of the initial conditions:

$$\lim_{t \to \infty} \mathbf{P}_i(t) = \frac{1}{N} \sum_{j \in \mathcal{V}} \mathbf{P}_j(0), \quad \lim_{t \to \infty} \mathbf{q}_i(t) = \frac{1}{N} \sum_{j \in \mathcal{V}} \mathbf{q}_j(0),$$
$$\lim_{t \to \infty} \mathbf{P}_i^{-1}(t) \mathbf{q}_i(t) = \boldsymbol{\theta}_{\text{ML}}, \quad \forall i \in \{1, \dots, N\}.$$

The covariance associated to the ML estimate is $\mathbf{\Lambda}_{\text{ML}} = \frac{1}{N}\lim_{t\to\infty}\mathbf{P}_i^{-1}(t)$. Note that the inverse of $\mathbf{P}_i^{-1}(t)$ is always well defined because of the properties of $\mathbf{\Lambda}_i$ and eq. (3).

The algorithm relies only on local communications and is robust to changes in the network topology. Moreover, the intermediate estimates, $\boldsymbol{\theta}_i(t) = \mathbf{P}_i^{-1}(t)\mathbf{q}_i(t)$, are unbiased; that is, $\text{E}[\boldsymbol{\theta}_i(t)] = \boldsymbol{\theta}_{\text{ML}}, \forall t \geq 0$. However, if some of the initial measurements contain extreme noise or spurious information about the attributes of the measured object, the final estimation will be erroneous and unreliable.

In order to give a formal definition of what an outlier is we use the Mahalanobis distance of the nodes' measurements to the real value of the quantity they are measuring:

$$d(\mathbf{x}_i, \boldsymbol{\theta}, \mathbf{\Lambda}_i) = \sqrt{(\mathbf{x}_i - \boldsymbol{\theta})^T \mathbf{\Lambda}_i^{-1}(\mathbf{x}_i - \boldsymbol{\theta})}. \tag{4}$$

***Definition** 2.1: (Inlier, Outlier):* A node is said to be an inlier if

$$d(\mathbf{x}_i, \boldsymbol{\theta}, \mathbf{\Lambda}_i) \leq \chi^2_{d,p}, \tag{5}$$

where $\chi^2_{d,p}$ is the value of the Chi-square distribution for d degrees of freedom, equal to the dimension of $\boldsymbol{\theta}$, and confidence probability p. Respectively, one node is considered as an outlier if eq. (5) is not true. We denote by $\mathcal{V}_{\text{in}} \subset \mathcal{V}$ the subset of nodes with inlier information.

The Mahalanobis distance is a standard tool in multivariate analysis to detect and discard outliers. In this case, it provides statistical information about the similarity of nodes' observations to the actual value. The confidence probability, p, is a parameter that needs to be fixed by the network (or its user) prior the execution of the algorithm, usually with standard values of 95% or 99%. The observations that do not pass the chi-square test will be considered to be highly uncorrelated with the real feature, and therefore, they should not be merged in the distributed computation of the ML in (3).

The goal of this paper is to identify this set of nodes with incorrect information and discard their influence in the computation of the maximum likelihood, all in a distributed fashion. In other words, the goal is to estimate the ML of the observations of the nodes in $\mathcal{V}_{\text{in}}$ in a distributed way.

III. Distributed RANSAC with Static Opinions

To filter out outliers, we propose an algorithm following the random sampling consensus (RANSAC) approach [17]. The algorithm in [17] is divided in three steps. First, subsets of nodes are randomly chosen to compute different possible consensus solutions using only their observations. Each one of these possible solutions is called a hypothesis in the RANSAC algorithm. Secondly, hypotheses are ranked by a voting procedure in which the votes are given by contrasting the observations of the nodes with each hypothesis. As a consequence, the hypothesis with the most number of votes is considered to be the best solution, and the nodes who voted for it are categorized as a *set of inliers*. Finally, the consensus value is computed using only the information of this set of inlier nodes. We emphasize the assumption that nodes are not considered to be malicious, and so the steps of our algorithm can be performed correctly. In the following, we explain how to make the process distributed, leading to our Distributed Static-Opinion RANSAC algorithm.

A. Distributed Generation of Hypotheses

The first step of the RANSAC algorithm consists of fixing the number K of hypothetical values of the real quantity, $\boldsymbol{\theta}$. Taking into account that the final consensus value is computed as the maximum likelihood of a set of uncertain observations, a hypothesis is defined as the maximum likelihood of a subset of the observations of the network.

Following the RANSAC principles, we assume that the observation of each node has equal probability of being a good observation, p_{in}, independent of the probability of the rest of observations. This probability represents the ratio between inliers and outliers. Given a desired probability of succeeding to generate a hypothesis only of inliers, p_{suc}, the number of required hypotheses is:

$$K = \frac{\log(1 - p_{\text{suc}})}{\log(1 - p_{\text{in}}^c)},$$

where c is the number of observations used to generate a hypothesis, see [17].

Noting that K is an increasing function of c, the smaller this value is, the less hypotheses will be required. Therefore, here we choose $c = 1$, meaning that each hypothesis will be the observation of a single node. Observe that the number of hypotheses does not depend on the number of nodes. For example, for $p_{\text{in}} = 0.7$ and $p_{\text{suc}} = 0.99$, the network will only generate $K = 4$ hypotheses, independently of the number of nodes N. The probability p_{in}, is in general not known and, therefore, it is based on an estimated conservative value. Typically, p_{in} is based on the *largest set of observed inliers*, or employs a priori information about this probability.

By setting $c = 1$, we can use a max-consensus algorithm to compute the set of hypotheses in finite time [1], [26]. We describe the process for a single hypothesis, noting that it should be run in parallel for all hypotheses.

To generate the h^{th} hypothesis each node initializes a random number, aux_i^h. The hypothesis is defined by the mean and the covariance of the node with the maximum value of aux_i^h,

$$(\boldsymbol{\theta}_{\text{ML}}^h, \mathbf{\Lambda}_{\text{ML}}^h) = \{(\mathbf{x}_i, \mathbf{\Lambda}_i) \mid i = \arg\max \text{aux}_i^h\}. \tag{6}$$

Algorithm 1 shows the distributed computation of (6) by the nodes. It is basically a max-consensus computation of the values aux_i^h while assigning the corresponding values to $\boldsymbol{\theta}_{\text{ML}}^h$ and $\mathbf{\Lambda}_{\text{ML}}^h$. After the execution of the algorithm, which is known to finish in a finite number of communication rounds, all the nodes in the network have the information of the hypothesis, contained in the variables $\boldsymbol{\theta}_{\text{ML}}^h$ and $\mathbf{\Lambda}_{\text{ML}}^h$.

B. Distributed Voting of Hypotheses

Next, agents vote for the different hypotheses in a decentralized fashion. The number of votes on the hypotheses can be counted in a distributed manner using distributed averaging. To do so, each node initializes a voting vector, $\mathbf{v}_i \in \mathbb{R}^K$, with as many elements as hypotheses to be voted for.

Algorithm 1 Computation of one hypothesis - Node i

aux_i^h = rand; $\boldsymbol{\theta}_{\text{ML}}^h = \mathbf{x}_i$; $\boldsymbol{\Lambda}_{\text{ML}}^h = \boldsymbol{\Lambda}_i$
for $it = 1 \dots \text{Diam}(\mathcal{G})$ **do**
  Send $(\text{aux}_i^h, \boldsymbol{\theta}_{\text{ML}}^h, \boldsymbol{\Lambda}_{\text{ML}}^h)$ to all $j \in \mathcal{N}_i$
  Receive $(\text{aux}_j^h, \boldsymbol{\theta}_{\text{ML}}^j, \boldsymbol{\Lambda}_{\text{ML}}^j)$ from all $j \in \mathcal{N}_i$
  if $\text{aux}_j^h > \text{aux}_i^h$ **then**
    $(\text{aux}_i^h, \boldsymbol{\theta}_{\text{ML}}^h, \boldsymbol{\Lambda}_{\text{ML}}^h) = (\text{aux}_j^h, \boldsymbol{\theta}_{\text{ML}}^j, \boldsymbol{\Lambda}_{\text{ML}}^j)$
  end if
end for

For every hypothesis h, the h^{th} component of the voting vector of node i, $\mathbf{v}_i^h$, is initialized as

$$\mathbf{v}_i^h(0) = \begin{cases} 1, & \text{if } d(\mathbf{x}_i, \boldsymbol{\theta}_{\text{ML}}^h, \boldsymbol{\Lambda}_i^h) \leq \chi^2_{d,p}, \\ 0, & \text{if } d(\mathbf{x}_i, \boldsymbol{\theta}_{\text{ML}}^h, \boldsymbol{\Lambda}_i^h) > \chi^2_{d,p}, \end{cases} \tag{7}$$

where

$$d(\mathbf{x}_i, \boldsymbol{\theta}_{\text{ML}}^h, \boldsymbol{\Lambda}_i^h) = \sqrt{(\mathbf{x}_i - \boldsymbol{\theta}_{\text{ML}}^h)^T (\boldsymbol{\Lambda}_i^h)^{-1} (\mathbf{x}_i - \boldsymbol{\theta}_{\text{ML}}^h)}. \tag{8}$$

For other nodes whose observations were not used to generate the hypothesis, the covariance, $\boldsymbol{\Lambda}_i^h$, is set equal to

$$\boldsymbol{\Lambda}_i^h = \boldsymbol{\Lambda}_{\text{ML}}^h + \boldsymbol{\Lambda}_i, \tag{9}$$

as $\mathbf{x}_i$ and $\boldsymbol{\theta}_{\text{ML}}^h$ are uncorrelated. For the node whose observation generated the hypothesis, $d(\mathbf{x}_i, \boldsymbol{\theta}_{\text{ML}}^h, \boldsymbol{\Lambda}_i^h)$ is set to zero as $\mathbf{x}_i$ and $\boldsymbol{\theta}_{\text{ML}}^h$ are equal.

After this, the nodes use distributed averaging to reach a consensus about the number of votes of the different hypotheses. That is,

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t)(\mathbf{v}_j(t) - \mathbf{v}_i(t)). \tag{10}$$

Since initially $\sum_i \mathbf{v}_i^h(0)$ equal the total number of votes, eventually, by the Periodic Connectivity Assumption 2.1 and the Symmetry and Row-stochasticity Assumption 2.2, the vectors of all nodes will converge to the total number of votes divided by the number of nodes in the network. Therefore, the hypothesis with the maximum value will correspond to the hypothesis with the most votes, which we denote by $h^* = \arg\max_h \mathbf{v}_i^h$. Additionally, since the nodes know if they voted or not for the hypothesis, they are also aware if they are inliers or outliers.

C. Distributed Computation of the ML of the Inliers

The last step computes the ML of those observations that voted for the best hypothesis, the set of inliers.

Let $\mathcal{V}_{\text{in}}$ be the subset of inlier nodes. The maximum likelihood of the observations of this set is equal to

$$\boldsymbol{\theta}_{\text{ML}} = (\sum_{i \in \mathcal{V}_{\text{in}}} \boldsymbol{\Lambda}_i^{-1})^{-1} \sum_{i \in \mathcal{V}_{\text{in}}} \boldsymbol{\Lambda}_i^{-1} \mathbf{x}_i. \tag{11}$$

***Proposition** 3.1 (Distributed ML Computation):* The variable $\boldsymbol{\theta}_i(t) = \mathbf{P}_i^{-1}(t)\mathbf{q}_i(t)$, updated using (3) with initial conditions

$$[\mathbf{P}_i(0), \mathbf{q}_i(0)] = \begin{cases} [\boldsymbol{\Lambda}_i^{-1}, \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i] \text{ if } i \in \mathcal{V}_{\text{in}}, \\ [\mathbf{0}, \mathbf{0}] \text{ otherwise,} \end{cases} \tag{12}$$

asymptotically converges to (11) for all $i \in \mathcal{V}$.

Proof. As stated in [18], $\mathbf{P}_i(t)$ and $\mathbf{q}_i(t)$ converge to the average of the initial values of all the $\mathbf{P}_j(t)$ and $\mathbf{q}_j(t), j \in \mathcal{V}$. However, the initial values for any node $j \notin \mathcal{V}_{\text{in}}$ are zero by eq. (12), therefore, for all $i \in \mathcal{V}$, it holds that

$$\begin{aligned} \lim_{t\to\infty} \boldsymbol{\theta}_i(t) &= \lim_{t\to\infty} \mathbf{P}_i^{-1}(t)\mathbf{q}_i(t) = \\ &= (\frac{1}{N}\sum_{j\in\mathcal{V}} \boldsymbol{\Lambda}_j^{-1})^{-1} \frac{1}{N} (\sum_{j\in\mathcal{V}} \boldsymbol{\Lambda}_j^{-1}\mathbf{x}_i) = \\ &= (\frac{1}{N}\sum_{j\in\mathcal{V}_{\text{in}}} \boldsymbol{\Lambda}_j^{-1})^{-1} \frac{1}{N} (\sum_{j\in\mathcal{V}_{\text{in}}} \boldsymbol{\Lambda}_j^{-1}\mathbf{x}_i) = \boldsymbol{\theta}_{\text{ML}}. \end{aligned}$$

■

The covariance associated to the ML will be $\boldsymbol{\Lambda}_{\text{ML}} = \frac{1}{N}\lim_{t\to\infty} \mathbf{P}_i^{-1}(t)$.

Note that, compared to a centralized implementation of RANSAC, our DISTRIBUTED STATIC-OPINION RANSAC algorithm offers the same performance guarantees. It generates the same number of hypotheses with random subsets of observations, it uses the same ranking function that centralized RANSAC would use and therefore, it returns the same voting count. Finally, Proposition 3.1 ensures the computation of the desired result for the best hypothesis. Thus, our algorithm offers the same behavior but in a fully distributed fashion.

IV. DISTRIBUTED RANSAC WITH DYNAMIC OPINIONS

The previous DISTRIBUTED STATIC-OPINION RANSAC requires three different consensus steps, one for the generation of the hypotheses, one for the distributed voting process, and a last one to compute the ML of the inlier observations. Here, we propose a DISTRIBUTED DYNAMIC-OPINION RANSAC algorithm where the nodes vote for (or not) a hypothesis as soon as their observations pass the Chi-square test. In this way, we reduce the amount of information exchanged in the hypotheses generation step and we merge the voting and the computation of the ML of the inliers into a single step. In order to make the presentation clearer, we describe the algorithm just for one hypothesis, omitting the superscript h.

First, nodes initialize a random number, aux_i^h, as in the DISTRIBUTED STATIC-OPINION RANSAC algorithm and perform a max-consensus to decide which node is the generator of the hypothesis. For the sake of legibility, we will denote by $i^* = \arg\max \text{aux}_i^h$. The difference is that in this version of the algorithm the nodes do not send the values of $\mathbf{x}_i$ and $\boldsymbol{\Lambda}_i$ to their neighbors, therefore reducing the amount of communications of this step. Also recall that this step requires a finite number of communication rounds.

After this, nodes initialize their states as follows

$$[\mathbf{P}_i(0), \mathbf{q}_i(0), \mathbf{v}_i(0)] = \begin{cases} [\boldsymbol{\Lambda}_i^{-1}, \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i, 1] \text{ if } i = i^*, \\ [\mathbf{0}, \mathbf{0}, 0] \text{ otherwise.} \end{cases} \tag{13}$$

In contrast with the previous approach, now the nodes start to vote without knowing the hypotheses. The ML computation of the positive votes also starts at the same time as the voting

itself. The new local updates for each node are

$$\begin{aligned}\mathbf{P}_i(t+1) &= \mathbf{P}_i(t) + \sum_{j\in\mathcal{N}_i(t)} a_{ij}(t)(\mathbf{P}_j(t)-\mathbf{P}_i(t)) + \mathbf{u}_i^{\mathbf{P}}(t),\\ \mathbf{q}_i(t+1) &= \mathbf{q}_i(t) + \sum_{j\in\mathcal{N}_i(t)} a_{ij}(t)(\mathbf{q}_j(t)-\mathbf{q}_i(t)) + \mathbf{u}_i^{\mathbf{q}}(t),\\ \mathbf{v}_i(t+1) &= \mathbf{v}_i(t) + \sum_{j\in\mathcal{N}_i(t)} a_{ij}(t)(\mathbf{v}_j(t)-\mathbf{v}_i(t)) + \mathbf{u}_i^{\mathbf{v}}(t),\end{aligned} \tag{14}$$

where $\mathbf{u}_i^{\mathbf{P}}(t), \mathbf{u}_i^{\mathbf{q}}(t)$ and $\mathbf{u}_i^{\mathbf{v}}(t)$ are dynamic inputs to the consensus rule.

In order to decide the inputs, each node executes the Chi-square test with the current value of $\boldsymbol{\theta}_i(t) = \mathbf{P}_i^{-1}(t)\mathbf{q}_i(t)$. For brevity, we will denote

$$d_i(t) = d(\mathbf{x}_i, \boldsymbol{\theta}_i(t), \boldsymbol{\Lambda}_i). \tag{15}$$

Note that the inverse of $\mathbf{P}_i(t)$ is not always well defined. For the time instants t for which $\mathbf{P}_i^{-1}(t)$ does not exist, we cannot compute the Mahalanobis distance. However, at these instants we assign the distance $d_i(t)$ a value larger than $\chi^2_{d,p}$.

Note that compared to (9), the covariance matrix in this case does not include the term caused by the hypothesis, $\boldsymbol{\Lambda}^h_{\text{ML}}$. This is a conservative solution adopted because in the first iterations the estimation of $\mathbf{P}_i(t)$ is highly unreliable. Usually, at these times $\mathbf{P}_i(t)$ is multiplied by weights very close to zero, resulting in large covariances due to the inverse $\mathbf{P}_i^{-1}(t)$. When this happens, the Mahalanobis distances are close to zero and votes from the outliers appear in the algorithm. Thus, by not considering this term, even when some inliers might not vote for the hypothesis, we ensure that the outliers do not vote for it, which is the most important part of the process.

Denote the set of time instants in which node i changes its opinion as follows:

$$\begin{aligned}\mathcal{T}_i^+ &= \{t\in\mathbb{N} \mid d_i(t) \le \chi^2_{d,p} \ \wedge \ d_i(t-1) > \chi^2_{d,p}\},\\ \mathcal{T}_i^- &= \{t\in\mathbb{N} \mid d_i(t) > \chi^2_{d,p} \ \wedge \ d_i(t-1) \le \chi^2_{d,p}\}.\end{aligned} \tag{16}$$

The control inputs of node i are given by:

$$[\mathbf{u}_i^{\mathbf{P}}(t), \mathbf{u}_i^{\mathbf{q}}(t), \mathbf{u}_i^{\mathbf{v}}(t)] = \begin{cases}[\boldsymbol{\Lambda}_i^{-1}, \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i, 1] & \text{if } t\in\mathcal{T}_i^+,\\ -[\boldsymbol{\Lambda}_i^{-1}, \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i, 1] & \text{if } i\in\mathcal{T}_i^-,\\ [\mathbf{0},\mathbf{0},0] & \text{otherwise.}\end{cases} \tag{17}$$

***Proposition** 4.1: (Convergence to ML with Dynamic Opinions):* If $\mathcal{T}_i^+$ and $\mathcal{T}_i^-$ are finite for all $i\in\mathcal{V}$ then the rule (14) with control inputs (17) converges to

$$\lim_{t\to\infty} \boldsymbol{\theta}_i(t) = \Big(\sum_{j\in\mathcal{V}_{\text{con}}} \boldsymbol{\Lambda}_j^{-1}\Big)^{-1} \sum_{j\in\mathcal{V}_{\text{con}}} \boldsymbol{\Lambda}_j^{-1}\mathbf{x}_j, \tag{18}$$

$$\lim_{t\to\infty} \mathbf{v}_i(t) = \frac{|\mathcal{V}_{\text{con}}|}{N}. \tag{19}$$

where $\mathcal{V}_{\text{con}} = \{i\in\mathcal{V} \mid d_i(t_{\max}) \le \chi^2_{d,p}\}$ and $t_{\max} = \max_t(\mathcal{T}_i^+, \mathcal{T}_i^-), \forall i\in\mathcal{V}$.

Proof. The sets $\mathcal{T}_i^+$ and $\mathcal{T}_i^-$ are finite. This means that there is some time instant, $t_{\max}$, that upper bounds $\mathcal{T}_i^+$ and $\mathcal{T}_i^-, \forall i\in\mathcal{V}$. Moreover, $\forall t > t_{\max}$ and $i\in\mathcal{V}$, the sign of $d_i(t) - \chi^2_{d,p}$ remains constant, which means that the nodes do not change their opinion after $t_{\max}$.

Let us analyze the evolution of $\mathbf{P}_i(t)$. After $t_{\max}$, the iteration rule (14) behaves like (3) because $\mathbf{u}_i^{\mathbf{P}}(t) = \mathbf{0}$ for all i and t, and therefore, $\mathbf{P}_i(t)$ will converge to $\frac{1}{N}\sum_{i\in\mathcal{V}}\mathbf{P}_i(t_{\max})$. The sum of the values of $\mathbf{P}_i(t_{\max})$ can be written as

$$\sum_{i\in\mathcal{V}} \mathbf{P}_i(t_{\max}) = \boldsymbol{\Lambda}_{i^*}^{-1} + \sum_{i\in\mathcal{V}}\sum_{t=0}^{t_{\max}} \mathbf{u}_i^{\mathbf{P}}(t). \tag{20}$$

The sum of the inputs for each node is

$$\sum_{t=0}^{t_{\max}} \mathbf{u}_i^{\mathbf{P}}(t) = \sum_{t\in\mathcal{T}_i^+} \boldsymbol{\Lambda}_i^{-1} - \sum_{t\in\mathcal{T}_i^-} \boldsymbol{\Lambda}_i^{-1} = (|\mathcal{T}_i^+| - |\mathcal{T}_i^-|)\boldsymbol{\Lambda}_i^{-1}.$$

By the eq. (16), for any node i, $-1 \le |\mathcal{T}_i^+| - |\mathcal{T}_i^-| \le 1$. At the beginning, for the node i^* it holds that $d_{i^*}(0) \le \chi^2_{d,p}$, because $\boldsymbol{\theta}_{i^*}(0) = \mathbf{x}_{i^*}$. Therefore,

$$\sum_{t=0}^{t_{\max}} \mathbf{u}_{i^*}^{\mathbf{P}}(t) = \begin{cases}-\boldsymbol{\Lambda}_{i^*}^{-1} & \text{if } d_{i^*}(t_{\max}) > \chi^2_{d,p},\\ \mathbf{0} & \text{otherwise,}\end{cases} \tag{21}$$

The rest of the nodes initially cannot compute the distance because the inverse of $\mathbf{P}_i(0)$ is not defined; but this situation in our algorithm is equivalent to having a distance larger than $\chi^2_{d,p}$. Then we have that for any $i\neq i^*$,

$$\sum_{t=0}^{t_{\max}} \mathbf{u}_i^{\mathbf{P}}(t) = \begin{cases}\boldsymbol{\Lambda}_i^{-1} & \text{if } d_i(t_{\max}) \le \chi^2_{d,p},\\ \mathbf{0} & \text{otherwise,}\end{cases} \tag{22}$$

Putting together (20), (21) and (22), in the limit we have

$$\lim_{t\to\infty} \mathbf{P}_i(t) = \frac{1}{N}\sum_{i\in\mathcal{V}} \mathbf{P}_i(t_{\max}) = \frac{1}{N}\sum_{i\in\mathcal{V}_{\text{con}}} \boldsymbol{\Lambda}_i^{-1}, \tag{23}$$

Applying the same argument to $\mathbf{q}_i(t)$, we obtain

$$\lim_{t\to\infty} \mathbf{q}_i(t) = \frac{1}{N}\sum_{i\in\mathcal{V}} \mathbf{q}_i(t_{\max}) = \frac{1}{N}\sum_{i\in\mathcal{V}_{\text{con}}} \boldsymbol{\Lambda}_i^{-1}\mathbf{x}_i, \tag{24}$$

and then eq. (18) holds. Finally, following the same reasoning with $\mathbf{v}_i(t)$ eq. (19) is obtained and the proof is complete. ■

If the nodes are not indefinitely changing their vote, then the algorithm will achieve convergence to the ML of the subset of nodes that have voted for it. If we consider all the hypotheses at the same time the nodes execute the same process with all of them simultaneously. At the end the hypothesis with the larger value of $\mathbf{v}_i^h$ will be the one selected by all the nodes as the good one. Note that with this approach, once the hypothesis is selected there is no need to compute additional ML estimates.

What remains to be done now is to determine conditions that guarantee the covergence to the ML of the inliers.

A. Conditions to reach the ML of $\mathcal{V}_{\text{in}}$

We derive a set of reasonable conditions such that, if $i^*\in\mathcal{V}_{\text{in}}$ for one hypothesis, then the assumptions in Proposition 4.1 are met and $\mathcal{V}_{\text{con}} = \mathcal{V}_{\text{in}}$ for that hypothesis.

First, we impose a condition on the inlier observations. Since they are different measurements of the same vector, they have to be close to each other.

***Condition** 1 (Relative Inlier location):* For any pair of nodes $i, j\in\mathcal{V}_{\text{in}}$ it holds that $d(\mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\Lambda}_i) \le \chi^2_{d,p}$.

***Lemma** 4.1 (Mahalanobis distance to the Convex Hull):* Let $CH(\mathcal{V}_{\text{in}})$ be the convex hull of the inlier observations. If Condition 1 is satisfied, then, for any node $i \in \mathcal{V}_{\text{in}}$ and any $\mathbf{x} \in CH(\mathcal{V}_{\text{in}})$, we have $d(\mathbf{x}_i, \mathbf{x}, \mathbf{\Lambda}_i) \leq \chi^2_{d,p}$.

Proof. Let us note that $\mathbf{x}_i \in CH(\mathcal{V}_{\text{in}})$ for all i. For any point in the convex hull, the maximum distance to points inside the hull is achieved at one of the corner points. Since these points are observations that belong to $\mathcal{V}_{\text{in}}$

$$d(\mathbf{x}_i, \mathbf{x}, \mathbf{\Lambda}_i) \leq \max_{\mathbf{x}_j \in \mathcal{V}_{\text{in}}} d(\mathbf{x}_i, \mathbf{x}_j, \mathbf{\Lambda}_i) \leq \chi^2_{d,p}.$$

■

This means that we have a set of points voted for by all the inliers, which leads to a second condition:

***Condition** 2 (Location of the ML of Inliers):* For any subset $\mathcal{S} \subseteq \mathcal{V}_{\text{in}}$, the maximum likelihood estimate of $\mathcal{S}$ is inside $CH(\mathcal{V}_{\text{in}})$.

The lemma also suggests a restriction to impose to the outlier observations.

***Condition** 3 (Location of Outliers):* For all $\mathbf{x} \in CH(\mathcal{V}_{\text{in}})$ and $k \notin \mathcal{V}_{\text{in}}$ it holds that $d(\mathbf{x}_k, \mathbf{x}, \mathbf{\Lambda}_k) > \chi^2_{d,p}$.

However, let us note that the above conditions are not enough to ensure that one hypothesis instantiated with inliers will end up with all the inliers voting for it and all the outliers rejecting it. It could be possible that some outliers vote for it at some intermediate estimation or that one or more inliers constantly change their vote and there is no convergence. An additional condition to ensure that these two situations do not occur is imposed.

Let us notice that $\boldsymbol{\theta}_i(t)$ and the Mahalanobis distance, $d_i(t)$, can be written as functions of a vector $\mathbf{w} = (w_1, \ldots, w_N) \in [0,1]^N$, which represents the weights of the linear combination of the different observations.

$$\boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{P}_i^{-1}(\mathbf{w})\mathbf{q}_i(\mathbf{w}) = (\sum_{i \in \mathcal{V}} w_i \mathbf{\Lambda}_i^{-1})^{-1} (\sum_{i \in \mathcal{V}} w_i \mathbf{\Lambda}_i^{-1} \mathbf{x}_i),$$

$$d_i(\mathbf{w}) = d(\mathbf{x}_i, \boldsymbol{\theta}_i(\mathbf{w}), \mathbf{\Lambda}_i) = \\ = \sqrt{(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))^T \mathbf{\Lambda}_i^{-1} (\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))}.$$

Without loss of generality, let us assume that the nodes are ordered so that we can separate the different elements of $\mathbf{w}$ in $\mathbf{w}_{\text{in}} \in [0,1]^{|\mathcal{V}_{\text{in}}|}$, the corresponding to the inliers and $\mathbf{w}_{\text{out}}$ the components of the outliers, $\mathbf{w} = [\mathbf{w}_{\text{in}}, \mathbf{w}_{\text{out}}]$.

***Condition** 4 (Distance Minimization by Inliers):* For any $i \in \mathcal{V}$, the partial derivatives of $d_i(\mathbf{w})$,

$$\frac{\partial d_i(\mathbf{w})}{\partial w_j} = \frac{(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))^T \mathbf{\Lambda}_i^{-1} \mathbf{P}_i^{-1}(\mathbf{w}) \mathbf{\Lambda}_j^{-1} (\mathbf{x}_j - \boldsymbol{\theta}_i(\mathbf{w}))}{d_i(\mathbf{w})}, \tag{25}$$

satisfy, $\forall j \in \mathcal{V}$ and $\mathbf{w}_{\text{out}} = \mathbf{0}$, that

$$\frac{\partial d_i(\mathbf{w})}{\partial w_j} = 0 \Leftrightarrow \begin{cases} \boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{x}_i, \mathbf{x}_j, & \text{if } i,j \in \mathcal{V}_{\text{in}}, \\ \boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{x}_i, & \text{if } i \in \mathcal{V}_{\text{in}}, j \notin \mathcal{V}_{\text{in}}, \\ \boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{x}_j, & \text{if } i \notin \mathcal{V}_{\text{in}}, j \in \mathcal{V}_{\text{in}}. \end{cases} \tag{26}$$

The derivation of (25) is included in the Appendix.

The following Theorem demonstrates that under the conditions given in the section, all the inliers vote for a good hypothesis and none of the outliers vote for it at any time:

***Theorem** 4.1 (Convergence to the ML of the Inliers):* Under Conditions 1-4 for any $\mathcal{V}_h \subseteq \mathcal{V}_{\text{in}}$, the following holds:

- All inliers eventually vote for the hypothesis

$$\exists\, t^+ \mid \forall t > t^+, \forall i \in \mathcal{V}_{\text{in}},\ d_i(t) \leq \chi^2_{d,p}. \tag{27}$$

- Outliers do not vote for the hypothesis at any time

$$\forall k \notin \mathcal{V}_{\text{in}}, d_k(t) > \chi^2_{d,p},\ \forall t > 0. \tag{28}$$

This means that, by Proposition 4.1, (14) will converge and that $\mathcal{V}_{\text{con}} = \mathcal{V}_{\text{in}}$.

Proof. The main idea of the proof is to analyze the behavior of $d_i(\mathbf{w})$ for outliers and inliers in order to demonstrate that for outliers it is always greater than $\chi^2_{d,p}$, whereas for inliers it goes below that value and remains there.

Let us denote by C_ε, the set that contains values of $\mathbf{w}$ with the weights associated to the outliers equal to zero and at least one inlier with weight different than zero, i.e.,

$$C_\varepsilon = \{\mathbf{w} \mid \mathbf{w}_{\text{out}} = \mathbf{0} \text{ and } \mathbf{w}_{\text{in}} \in [0,1]^{|\mathcal{V}_{\text{in}}|} \setminus [0,\varepsilon)^{|\mathcal{V}_{\text{in}}|}\} \tag{29}$$

with $\varepsilon > 0$ an arbitrarily small value. In the values of $\mathbf{w}_{\text{in}}$, we remove the relatively open set $[0,\varepsilon)^{|\mathcal{V}_{\text{in}}|}$ in $[0,1]^{|\mathcal{V}_{\text{in}}|}$ to avoid the possibility of $\mathbf{w} = \mathbf{0}$, since that would mean that $d_i(\mathbf{w}) > \chi^2_{d,p}$ and any agent would vote for the hypothesis at any time. The subtraction of the relatively open set $[0,\varepsilon)^{|\mathcal{V}_{\text{in}}|}$ ensures that C_ε remains a closed set. Denoting by $\mathbf{e}_i$ the i^{th} vector of the canonical basis, the set of corners (the end values) of C_ε is characterized as

$$\mathbf{w}^* = \{\sum_{i \in \mathcal{S}} \mathbf{e}_i, \mathcal{S} \subseteq \mathcal{V}_{\text{in}}\} \bigcup \{\varepsilon \sum_{i \in \mathcal{S}} \mathbf{e}_i, \mathcal{S} \subseteq \mathcal{V}_{\text{in}}\}, \tag{30}$$

with $\mathcal{S}$ any possible subset of $\mathcal{V}_{\text{in}}$.

Note that $\boldsymbol{\theta}_i(\mathbf{e}_i) = \mathbf{x}_i$, $d_i(\mathbf{w}) = d_i(\varepsilon\mathbf{w})$, and if $\mathbf{w}$ is equal to the sum of several vectors of the canonical basis, then $\boldsymbol{\theta}_i(\mathbf{w})$ is equal to the maximum likelihood estimate of the corresponding nodes' observations. Combining these equalities with Condition 4, we conclude that the extreme values of $d_i(\mathbf{w})$, with $\mathbf{w} \in C_\varepsilon$ are reached at the corners of C_ε [27].

Let us consider any outlier node $k \notin \mathcal{V}_{\text{in}}$. First, $\mathbf{P}_k(0) = \mathbf{0}$ because $i^* \in \mathcal{V}_{\text{in}}$, and therefore k does not vote for the hypothesis. Let us denote now by k the first outlier for which $\mathbf{P}_k(t) \neq \mathbf{0}$. At this moment $\mathbf{w} \in C_\varepsilon$ for a small enough ε. Condition 2 together with Condition 3 implies that

$$d_k(\sum_{i \in \mathcal{S}} \mathbf{e}_i) = d_k(\varepsilon \sum_{i \in \mathcal{S}} \mathbf{e}_i) > \chi^2_{d,p}, \quad \forall \mathcal{S} \subseteq \mathcal{V}_{\text{in}}.$$

For all the corner points of C_ε, the Mahalanobis distance is larger than $\chi^2_{d,p}$. Since the corners represent the extreme values of the Mahalanobis distance, there is no value of $\mathbf{w} \in C_\varepsilon$ for which $d_k(\mathbf{w}) \leq \chi^2_{d,p}$. Therefore, the outlier will not vote for the hypothesis. As successive outliers have $\mathbf{P}_k(t) \neq \mathbf{0}$, still the same argument applies. This means that none of the outliers will vote the hypotheses and the condition in (28) is true.

A similar argument can be applied for the inliers. We already know that $\mathbf{w}_{\text{out}} = \mathbf{0}$ for any time instant. Looking at the corners of C_ε, by Conditions 1 and 2, for any $i \in \mathcal{V}_{\text{in}}$

$$d_i(\sum_{j \in \mathcal{S}} \mathbf{e}_j) = d_i(\varepsilon \sum_{j \in \mathcal{S}} \mathbf{e}_j) \leq \chi^2_{d,p}, \quad \forall \mathcal{S} \subseteq \mathcal{V}_{\text{in}},$$

then for any $\mathbf{w}_{\text{in}} \in C_\varepsilon$, $d_i(\mathbf{w}_{\text{in}}) \leq \chi^2_{d,p}$. This means that once $\mathbf{P}_i(t) \neq \mathbf{0}$, node i will vote for the hypotheses and will keep doing so. By taking t^+ the first time instant for which for all $i \in \mathcal{V}_{\text{in}}$ $\mathbf{P}_i(t) \neq \mathbf{0}$, the condition in (27) is satisfied and the result holds. ■

From the above theorem one can also conclude that if $p_{\text{in}} \geq 0.5$, then not only all the inliers will vote for the hypothesis, but they will also represent the majority, which leads to the choice of this hypothesis as the most voted. Also, if for any other subset $\mathcal{S} \in \mathcal{V}$ different from $\mathcal{V}_{\text{in}}$ for which Conditions 1-4 are satisfied it holds that $|\mathcal{S}| < |\mathcal{V}_{\text{in}}|$, then the network will choose as the best hypothesis the correct one, i.e., the one formed by the inliers.

Let us note that the above sufficient conditions are relatively easy to hold in practice. Condition 1, on the relative inlier location, is easy to hold as it corresponds to good measurements of the same object. Condition 2, on the location of the ML of inliers, might be violated for a small number of inliers (two or three). However, if the number of inliers is sufficiently large, the ML behaves more and more as a weighted average, resulting in the fulfillment of Condition 2. Condition 3, on the outlier location, is a natural condition, otherwise it may be argued that these measurements are not really outliers, as they would be posing a good observation of the feature.

Finally, Condition 4 essentially states that the Mahalanobis distance is minimized by inlier observations, when the contribution of the outliers to the maximum likelihood of the observations is zero. Note that, for any i, a global minimum of d_i is obtained when $\boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{x}_i$, and the Mahalanobis distance d_i becomes zero. This naturally holds for inliers, since they must vote for the hypotheses. In contrast, the satisfaction of this property by outliers would pose a problem. However, it is hard that a combination of observations satisfying Conditions 1-3 can return such an outlier. An extreme can also be given by a $\mathbf{w}$ such that $(\mathbf{x}_k - \boldsymbol{\theta}(\mathbf{w}))$ is orthogonal to all $(\mathbf{x}_i - \boldsymbol{\theta}(\mathbf{w}))$ with respect to $\boldsymbol{\Lambda}_k^{-1}\mathbf{P}_k^{-1}(\mathbf{w})\boldsymbol{\Lambda}_i^{-1}$. Nevertheless, we have not encountered this situation in any of the simulations we have carried out and, provided that this extreme satisfies that $d_k(\mathbf{w}) > \chi^2_{d,p}$, the algorithm would still converge.

Additionally, let us remark that the algorithm may still converge to the desired result if some of these conditions are not met. In Section VI we provide evidence of this situation.

To conclude, the algorithm presented in this section, compared to RANSAC, might not necessarily have the same output on a general instance due to dynamic opinions. Nevertheless, the hypothesis generation step is performed in the same fashion as in Section III, which implies the same probability of generating one hypothesis composed by inliers. This fact combined with the result given in Theorem 4.1 leads to the conclusion both algorithms will have the same performance.

V. Vote Counting in Finite Time

In this section, we propose a modification of the standard averaging rule that allows the network to compute the number of votes of the different hypotheses in a finite number of iterations. In this way, the nodes can identify in less time which hypotheses have low rankings and discard them, reducing the amount of communications. We divide this section in three parts: first, we present the modified averaging rule to reach the value of a given function in finite time. In the second part we explain how this new rule can be applied to the problem of counting the votes in finite-time. Since vote counting in finite time requires the knowledge of the total number of active nodes in the network, the last part of the section presents a distributed primitive to compute this number.

A. Distributed Averaging in Finite-Time

For general graph-switching sequences, the asymptotic convergence of the distributed averaging rule achieves convergence only asymptotically. However, when consensus involves integers, as it happens with voting vectors, or with real numbers with finite precision, this is possible as we see next.

***Definition** 5.1 (Finite-precision Set):* We will say that $\Phi \subseteq \mathbb{R}^D$ is a φ-set, $\varphi \in \mathbb{R}$, $D \in \mathbb{N}$ if $\forall\ \mathbf{x},\ \mathbf{x}' \in \Phi$

$$\frac{|x_k - x'_k|}{\varphi} \in \mathbb{N}, \quad \forall k = 1, \ldots, D, \tag{31}$$

with x_k, x'_k being the k^{th} component of $\mathbf{x}$, $\mathbf{x}'$ respectively. Considering the examples above mentioned, the set of integers, $\mathbb{Z}$, is defined as a 1-set and the set of reals with s decimal digits as a 10^{-s}-set.

We introduce next a modification of the standard consensus algorithm when dealing with elements in a φ-set. The idea of the new rule is to keep the distributed update in (3) and combine it with a rounding rule that returns the closest element in Φ. If the average, or the value of some function evaluated in the average, belongs to Φ, then the algorithm will reach the exact consensus value for all the nodes in finite time.

***Proposition** 5.1 (Distributed Averaging in Finite Time):* Let $g(\mathbf{x})$ be a continuous function, $g : \mathbb{R}^d \to \mathbb{R}^D$ and $\mathbf{x}_i(0) \in \mathbb{R}^d$ be initial conditions with $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i\in\mathcal{V}} \mathbf{x}_i(0)$ their average. Given Φ as a known φ-set, if $g(\bar{\mathbf{x}}) \in \Phi$, then the iteration

$$\begin{cases} \mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \displaystyle\sum_{j\in\mathcal{N}_i(t)} a_{ij}(t)(\mathbf{x}_j(t) - \mathbf{x}_i(t)), \\ \mathbf{y}_i(t+1) \in \arg\min_{\mathbf{x}\in\Phi} \|\mathbf{x} - g(\mathbf{x}_i(t))\|_2, \end{cases} \tag{32}$$

leads to the finite-time convergence of the variables $\mathbf{y}_i$ to the consensus value $g(\bar{\mathbf{x}})$. That is,

$$\exists t^* > 0 \mid \forall t > t^*,\ \mathbf{y}_i(t) = \mathbf{y}_i(t^*) = g(\bar{\mathbf{x}}),\ \forall i \in \mathcal{V}.$$

Proof. By Assumption 2.1 on Periodic Connectivity and Assumption 2.2 on Symmetry and Row Stochasticity, we know that $\mathbf{x}_i(t)$ in (32) will asymptotically converge to $\bar{\mathbf{x}}$ for all the nodes. Given $\varepsilon > 0$, the asymptotic convergence allows us to find t^* such that

$$\forall t > t^* \quad \|\bar{\mathbf{x}} - \mathbf{x}_i(t)\|_2 < \varepsilon, \quad \forall i \in \mathcal{V}.$$

In addition, the continuity of g in $\mathbf{x}$ implies that for all $\psi > 0$ there exists $\varepsilon > 0$ such that

$$\|\bar{\mathbf{x}} - \mathbf{x}_i(t)\|_2 < \varepsilon \Rightarrow \|g(\bar{\mathbf{x}}) - g(\mathbf{x}_i(t))\|_2 < \psi.$$

Considering again the definition of the φ-set and that $g(\bar{\mathbf{x}}) \in \Phi$, by choosing $\psi < \frac{\varphi}{2}$, there exists t^* depending on ε, and thus on ψ, such that for all $t > t^*$

$$\|g(\bar{\mathbf{x}}) - g(\mathbf{x}_i(t))\|_2 < \frac{\varphi}{2},$$

and therefore

$$\arg\min_{\mathbf{x}\in\Phi} \|\mathbf{x} - g(\mathbf{x}_i(t))\|_2 < \frac{\varphi}{2}.$$

Now, let us suppose that there exists some $\mathbf{x} \in \Phi, \mathbf{x} \neq g(\bar{\mathbf{x}})$, such that for some $t > t^*$

$$\|\mathbf{x} - g(\mathbf{x}_i(t))\|_2 < \|g(\bar{\mathbf{x}}) - g(\mathbf{x}_i(t))\|_2.$$

This would imply

$$\|\mathbf{x} - g(\bar{\mathbf{x}})\|_2 \leq \|\mathbf{x} - g(\mathbf{x}_i(t))\|_2 + \|g(\mathbf{x}_i(t)) - g(\bar{\mathbf{x}})\|_2 < \varphi,$$

which by definition of Φ is not possible. Then

$$g(\bar{\mathbf{x}}) = \arg\min_{\mathbf{x}\in\Phi} \|\mathbf{x} - g(\mathbf{x}_i(t))\|_2. \tag{33}$$

■

The proposed rule converges in finite time to the same value for all the nodes. In addition to the knowledge of g, nodes require to know φ, which is easy to find out before running the application. The following corollary, provides a bound on the number of iterations in terms of the initial error with respect to the average:

***Corollary** 5.1 (Bound on the total iterations):* Let us define $e(t) = \|\mathbf{x}(t) - \bar{\mathbf{x}}\mathbf{1}\|_\infty$, the maximum difference of the nodes' estimations with respect to the average at iteration t. Then, under Assumptions 2.1, on the network periodic connectivity, and 2.2, on symmetry and row stochasticity, $\mathbf{y}_i(t) = g(\bar{\mathbf{x}})$ for all t such that

$$t > T\frac{\log\left(\frac{\varphi}{2e(0)}\right)}{\log\lambda_2} \tag{34}$$

with

$$\lambda_2 = \sup_t \rho\left(\prod_{t_1=0}^{T} \mathbf{A}(t+T-t_1) - \frac{\mathbf{1}^T\mathbf{1}}{N}\right), \tag{35}$$

and $\rho(\cdot)$ the spectral radius operator.

Proof. By Assumptions 2.1 and 2.2, the product $\prod_{t_1=0}^{T} \mathbf{A}(t+T-t_1)$ returns a doubly stochastic matrix associated to a connected communication graph for all t. Thus, it has one eigenvalue equal to one, and the rest of the eigenvalues strictly smaller than one (in modulus). The parameter λ_2 represents the supreme of all these eigenvalues, and it can be shown that this parameter is also strictly smaller than one (in modulus). The error, $e(t)$, at iteration t can be bounded by

$$e(t) = \|\mathbf{x}(t) - \bar{\mathbf{x}}\mathbf{1}\|_\infty \leq \|\mathbf{x}(t) - \bar{\mathbf{x}}\mathbf{1}\|_2 < \lambda_2^{\lfloor \frac{t}{T} \rfloor} e(0).$$

When $e(t) < \varphi/2$, then eq. (33) holds. Combining this with the last equation and operating the result is demonstrated. ■

B. Distributed Vote Counting in finite-time

Let us see now how we can apply the previous iteration rule to the problem of counting the number of votes in the distributed RANSAC algorithms.

Basically the problem consists on finding an appropriate function, $g(x)$ such that when it is evaluated in the different components of the voting vectors, $\mathbf{v}_i(t)$, it returns a suitable output, in the sense that it contains the information about the number of votes and it belongs to some φ-set. Such a function is given in the following proposition:

***Proposition** 5.2 (Distributed Voting in Finite Time):* For any initial voting, the rule (32) with $g(\mathbf{x}) = N\mathbf{x}$ converges to the total number of votes in finite time.

Proof. First, let us notice that the initial voting belongs to the 1-set, because all the votes are integers. Also the sum of the initial conditions belongs to the 1-set. Finally note that $\sum_{i\in\mathcal{V}} \mathbf{v}_i(t) = N\bar{\mathbf{v}} = g(\bar{\mathbf{v}})$. The function satisfies all the conditions of Proposition 5.1 and then convergence in finite time follows. ■

The application of the rule in the static voting solution is straightforward. Letting $\mathbf{x}_i(t)$ in eq. (32) be the value of the voting vectors, $\mathbf{v}_i(t)$, in eq. (10), the vectors $\mathbf{y}_i(t)$ converge to the number of votes in finite time.

In the dynamic scenario, the update given in eq. (32) is slightly different than the one for the voting vectors given in eq. (14) because of the dynamic input. Indeed, depending on the changes of opinions of the nodes, there might be some hypotheses for which convergence does not occur. However, under Conditions 1-4, recalling Theorem 4.1 we know that there exists some time instant for which all the inliers vote a good hypothesis and keep the positive opinion. At this point the dynamic inputs of the votes remain equal to zero all the time and the algorithm behaves as a standard static consensus algorithm. Thus, the application of the finite-time rule also guarantees convergence to the number of votes in finite-time.

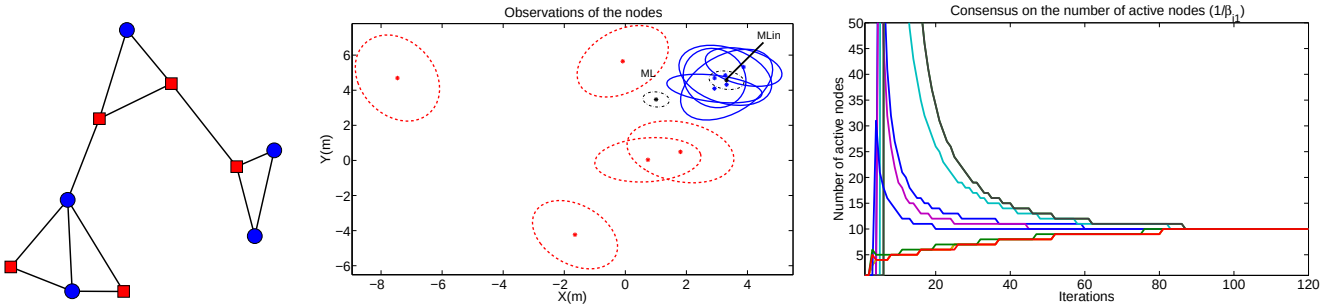
Finally, the chosen function, $g(x) = Nx$, to compute the exact number of votes in finite time requires the network to know how many nodes are participating in the process. This information can be very useful in many situations, but that may not always be available. In the following, we present a distributed procedure to obtain this information.

C. Distributed Primitive for Node Counting

We propose a distributed algorithm that allows a network to compute the number of nodes participating in the consensus process. The algorithm is based on distributed averaging combined with a max-consensus rule. The idea of the algorithm is to make the network evolve in such a way that the sum of the initial conditions is equal to one. Since the average value is divided by the total number of nodes involved in the computation, the system will tend to $1/N$ and every node will be able to know how many nodes are participating.

Let $N_i(t), i \in \mathcal{V}$, be the estimated value of N that node i has at time instant t. In order to make $N_i(t) \to N$ the node exchange a 2-dimensional vector, $\boldsymbol{\beta}$, initialized as

$$\boldsymbol{\beta}_i(0) = (\mathrm{ID}_i, 1)^T, \tag{36}$$



(a) Communication Graph (b) Observations of the nodes (c) Distributed Node Counting

Fig. 1. Robust sensor fusion of a bi-dimensional feature observed by ten nodes. (a) Communication network. Blue circles represent the nodes with inlier information and red squares the outliers. (b) Observations of the ten nodes. Five of them have a good observation of the feature (blue crosses and solid ellipses) whereas five nodes have observed different features (red crosses and dashed ellipses). This implies a 50% of outlier observations. If all the measurements are considered in the computation of the Maximum Likelihood, the obtained result is the black cross and dash-dot ellipse with the ML mark. The good maximum likelihood is the one in the middle of the inlier observations (MLin). (c) Distributed computation of the number of active nodes. Each line of the graphic represents the estimation of one node about the total number of nodes participating. In a finite number of iterations all the nodes know that there are 10 active nodes participating in the algorithm.

where ID_i is the identifier of node i in the network.

***Assumption** 5.1 (Node Identifiability):* Each node $i \in \mathcal{V}$ can be univocally identified.

At each time step the nodes update their values as follows:

$$\beta_{i1}(t+1) = \max(\beta_{i1}(t), \beta_{j1}(t)), \quad j \in \mathcal{N}_i(t), \tag{37}$$

$$\beta_{i2}(t+1) = \beta_{i2}(t) + \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t)(\beta_{j2}(t) - \beta_{i2}(t)) + u_i(t+1), \tag{38}$$

denoting $\beta_{i1}(t)$ and $\beta_{i2}(t)$ as the first and second component of $\boldsymbol{\beta}(t)$ that node i has at time instant t and $a_{ij}(t)$ satisfying Assumption 2.2 on Symmetry and Row Stochasticity.

In eq. (38), the initial input is set to zero, $u_i(0) = 0$, and for the rest of time instants is defined as

$$u_i(t+1) = \begin{cases} -1, & \text{if } \beta_{i1}(t) \neq \mathrm{ID}_i \text{ and } \sum_{s=0}^{t} u_i(s) = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{39}$$

***Proposition** 5.3 (Distributed Node Counting):* If all the nodes have initial values, $\boldsymbol{\beta}$, defined in (36) and update their states using (37) and (38), then, for all $i \in \mathcal{V}$,

1) $\beta_{i1}(t) \to \max_{i \in \mathcal{V}} \mathrm{ID}_i$, in finite time.
2) $\beta_{i2}(t) \to \frac{1}{N}$ as $t \to \infty$.
3) The iteration rule

$$N_i(t) = \arg\min_{n \in \mathbb{N}} |n - \frac{1}{\beta_{i2}(t)}|, \tag{40}$$

converges to N in finite time.

Proof. 1) The rule in (37) is a max consensus update rule. Under Assumption 2.1, the max consensus algorithm is proved to converge in a finite number of iterations for all the nodes in the network [26].

2) The sum of the initial conditions is $\sum_i \beta_{i2}(0) = N$. In order to compute the sum of the inputs, taking into account 1) and Assumption 5.1 on Node Identifiability, we have that after some time instant $t^* < \infty$, $\beta_{i1} \neq \mathrm{ID}_i, \forall i \setminus \max_{i \in \mathcal{V}} \mathrm{ID}_i$. Since the initial input is equal to zero for all the nodes, then, by (39), every node but the leader (the one with ID equal to the max consensus) will have an input $u_i(t) = -1$ for some t within the interval $[2, t^*]$. After that, the sum of the previous inputs will be different from zero and, therefore, the future inputs will also be equal to zero. Thus,

$$\sum_{t=0}^{t^*} \sum_{i \in \mathcal{V}} u_i(t) = -N + 1. \tag{41}$$

Then, $\sum_i \beta_{i2}(t^*) = 1$ and $u_i(t) = 0$, $\forall\, i \in \mathcal{V}, t > t^*$, which means that (38) behaves like (3) and $\beta_{i2}(t) \to \frac{1}{N}$ as $t \to \infty$.

3) Choosing $g(x) = \frac{1}{x}$, which is continuous for any $x \neq 0$ and satisfies that $g(\bar{x}) = N \in \mathbb{N}$, by direct application of Proposition 5.1 the result is proved. ■

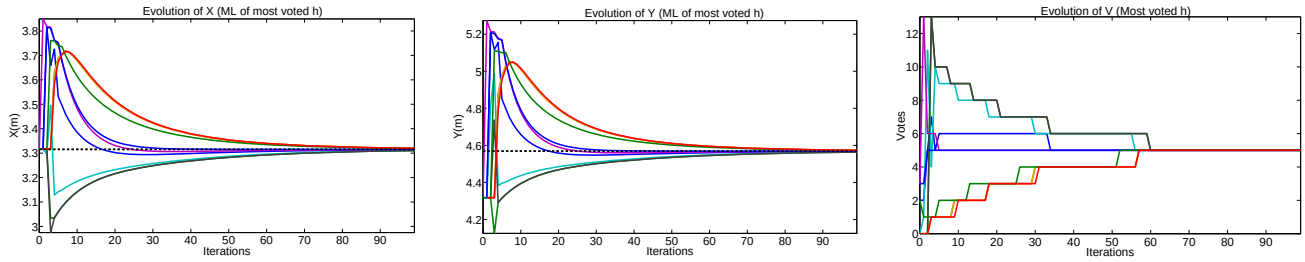
VI. Simulation Results

A. Illustrative Example

We first show the behavior of the dynamic voting algorithm with an illustrative example. In Fig. 1 (b), we show the observations of a two-dimensional feature by a network composed by 10 nodes. We have chosen a two-dimensional feature in order to have a good visualization of the results, however, we note that the algorithm is not restricted to this case and can be used with descriptors of any dimension. The communication network is fixed (Fig. 1 (a)) and we have used Metropolis Weights [18] to ensure the Symmetry and Row-Stochasticity Assumption 2.2. Five of the nodes have good observations of the feature (blue crosses and solid ellipses) and other five nodes have outlier information (red crosses and dashed ellipses).

If all the measurements are considered in the computation of the Maximum Likelihood, the obtained result is the black cross and dash-dotted ellipse with the ML mark at value $(1.00, 3.47)$ while the ML of the inlier observations is $(3.31, 4.56)$ (MLin). In this example, the conditions stated in section IV-A are satisfied, ensuring convergence to the ML of the inliers if one hypothesis is instantiated by nodes in $\mathcal{V}_{\text{in}}$.

Initially, the nodes do not know how many other sensors are active due to failures, initialization errors, etc. Fig. 1 (c) shows the evolution on the consensus about the number of active nodes using the finite-time rule. The final value is 10, the exact number of active sensors, for all the nodes in the network. Once this computation is done, the nodes decide the



(a) (b) (c)

Fig. 2. Evolution of the Maximum Likelihood and the number of votes of the most voted hypothesis. In figures (a) and (b) we show the evolution of $\boldsymbol{\theta}_i(t)$ for the hypothesis that has obtained the most number of votes in the end. The dashed black line is the value of the ML of the nodes with inlier information whereas the colored lines represent the estimations of the different nodes. It is observed that in both coordinates the values of $\boldsymbol{\theta}_i(t)$ asymptotically converge to the dashed black line, meaning that all the nodes reach the correct value. The evolution of the number of votes is depicted in (c). It converges to 5 for all the nodes, which is exactly the number of nodes with inlier information.

hypotheses using max consensus. In this experiment, we have set the probability of being an inlier to 0.6 and the probability of success in RANSAC to 0.99, resulting in a total of $K = 5$ hypotheses. In Fig. 2 (a), (b), we show the evolution of the two coordinates of $\boldsymbol{\theta}_i(t)$ for the most voted hypothesis. The values of the different nodes converge to the value of the ML of all the inliers (depicted in black dashed line in the graphics). In Fig. 2 (c), the evolution of the number of votes for the same hypothesis is depicted. Eventually, all the nodes reach the value 5, which is exactly the number of nodes with good information. It is also remarkable that the number of iterations in which the nodes change their opinion is considerably small. In less than 10 iterations, the plots do not show discontinuities due to the inputs (14). After that point, the algorithm behaves as a static consensus algorithm and the number of iterations required to converge is of the same order.

B. Monte Carlo Simulations

We have also run a Monte Carlo simulation considering more general situations where the conditions of Section IV-A do not always hold to analyze the performance of our algorithm.

1) Influence of p_{in}: In a first experiment, we analyze the performance of our algorithms with static and dynamic opinions (algorithms in Sections III and IV respectively), considering different percentages of inlier information (p_{in}). We run 1000 trials in which 20 nodes are considered. For each trial we randomly generate a 2-dimensional feature that is used as ground truth to measure the final errors in the estimations of the network. For each node we have generated a different measurement of the feature. The inlier nodes have measurements of the feature with zero-mean gaussian error and standard deviation of 1 meters. For the outlier nodes we have assigned a deviation of 10 meters. Covariance matrices are randomly generated with eigenvalues of mean 0.5 and standard deviation 0.5. Regarding the communication topology we have changed it, also in a random fashion, at each iteration of each trial without affecting our results.

The results are reported in Table I. The row "Total Outliers" contains the total number of outlier nodes generated in each experiment. As expected, the smaller p_{in}, the larger the number of outliers. The row "Outliers Detected" represents the number of nodes that were correctly classified as outliers whereas the row "Inliers Discarded" shows the number of inlier nodes that were mistakenly considered as outliers by our algorithm. Even in the worst-case scenario, with only a 20% of inlier nodes ($p_{\text{in}} = 0.2$) both approaches were able to discover almost of the outliers, 14101 out of 15034 in the static case, and 14534 out of 15034 in the dynamic case. The last row, "% Failure", shows the percentage of occasions that the most voted solution did not correspond to a solution voted by inliers only. Again, even in the worst case, in only the 18.4% and 12.9% of the situations our algorithms failed. In the last row we analyze the number of iterations required for each of the two algorithms to reach the final results. The static voting algorithm requires in all the cases 220 iterations per trial, which are divided in 20 iterations for the max-consensus, 100 iterations to vote for the hypothesis and 100 iterations in order to compute the ML of the inliers. The dynamic voting algorithm executes the voting and the computation of the ML at the same time, and therefore, it only requires 120 iterations.

TABLE I
ROBUSTNESS FOR DIFFERENT VALUES OF p_{in}

Static Voting (1000 trials)				
p_{in}	1	0.75	0.5	0.2
Total Outliers	0	5059	9913	15034
Outliers Detected	0	4797	9390	14101
Inliers Discarded	10	98	111	154
% Failure	0	1.4	4.1	18.4
Iterations per Trial	220	220	220	220
Dynamic Voting (1000 trials)				
p_{in}	1	0.75	0.5	0.2
Total Outliers	0	5059	9913	15034
Outliers Detected	0	4952	9680	14534
Inliers Discarded	126	185	201	306
% Failure	0	0.9	2.0	12.9
Iterations per Trial	120	120	120	120

2) Comparison with other approaches: We have also compared the results obtained by our method with the distributed consensus algorithm proposed in [18] to compute the ML of all the observations and the robust approach in [12] considering the $L_{1.5}$ loss norm. The conditions of the experiment are the same as in the previous scenario considering a fixed probability for each node to have inlier information equal to 0.8, leading to the generation of only 3 hypotheses in each trial.

TABLE II
COMPARISON OF THE DIFFERENT ALGORITHMS

Method	ML [18]	Optim. [12]	Static	Dynamic
Trials	1000			
False Positive Votes	4004	-	488	341
False Negative Votes	0	-	191	163
Avrg. Norm of Error	2.07	0.46	0.47	0.39
Std. Deviation of Error	2.36	0.27	0.60	0.51
Iterations per Trial	100	100	220	120
Data per iteration	6	6	21	21

The results obtained in the simulation can be seen in Table II. In the table a false positive represents a node with outlier information that has given a positive vote to the best hypothesis, i.e., a node that should not have voted the hypothesis but has done so. In contrast, a false negative is a node who had inlier information but did not vote for the best hypothesis. The results using the non-robust algorithm [18] are in the first column. In this case all the outliers participate in the different trials (a total of 4004 false positives). As a consequence, the average error in the estimation, computed using the Euclidean distance to the ground truth feature, is large (2.07 meters). If we use our approach with static opinions (third colum), the results are improved because only 488 false positive votes appear and the average error is reduced to 0.47 meters, with only 191 inliers thinking they have outlier information. The use of [12], optimizing a robust loss function is in the second column. In this case we cannot count the number of false positives and false negatives because all the nodes participate in some way in the final consensus value. The consensus solution given by this approach has a slightly smaller mean Euclidean distance with respect to the ground truth than our static-voting algorithm. On the other hand, the obtained results by means of dynamic voting are better, with an average error of 0.39 meters. This can be explained by the violation of Condition 1 (*Relative Inlier location*) in the experiment. While two inlier observations might not be close to each other in practice, it is very likely, as shown by the results, that both of them are sufficiently close to the ML of the set of inliers to eventually give a positive vote to the hypothesis. Moreover, since the nodes are voting a dynamic observation which tends to the good ML, a fewer number of false negatives is registered (163). This is explained by the violation of Condition 3 (*Location of Outliers*), similarly as we have done to explain the violation of Condition 1.

In terms of communications, the dynamic voting only requires as additional iterations the max-consensus to initialize the hypothesis, which we have upper-bounded by the number of nodes in the experiment, leading to 120 iterations compared to the 100 iterations of existing techniques. The size of the messages is shown in the last row of the table. In this example, using [12], [18] the nodes need to send at each iteration 6 floats (2 floats for the coordinates of the feature plus 4 floats for the covariance matrix) whereas using our method, for each hypothesis we require to send the same 6 floats plus an additional one for the vote counting. Since we are handling 3 hypotheses, we require to send 21 floats per iteration.

In order to show another important advantage of our approach, in Table III we repeat the previous comparison, but considering that nodes produce a measurement of either of two different features. In principle, nodes are not aware that there can be two different events, and this can lead to large deviations in final consensus values. In our simulations, each feature is observed by a 40% of the nodes, while the remaining 20% has outlier observations. In this way, since the observations are on both features, the total percentage of outliers will amount to a 60% of the total number of nodes. Since in our approach the nodes can consider multiple hypotheses in parallel, it is possible to handle both features simultaneously as follows: once the best hypothesis has been identified, nodes will select the next most voted hypothesis such that

- They did not vote for it if they were inliers.
- They voted for it if they were outliers.

Since the experiment is repeated 1000 times, the total number of features generated in the simulation is equal to 2000. The row (Features Detected) in Table III shows how many of the real features the different algorithms were able to identify, computing the number of final consensus values (reached after an algorithm run) located at an Euclidean distance smaller than 0.75 meters of the ground truth value of the features of that run. Since the approaches in [12], [18] only output one value for each trial, in the best case they are able to identify half of the real features, though they are only able to detect 20 and 28 out of 2000, respectively. In [12], this is explained because the algorithm looks for a value that optimizes the $L_{1.5}$ norm of the error, and therefore most trials return a value halfway of the two features, leading to a very small number of localized events. The mean error in these methods is calculated considering the Euclidean distance of their solutions to the closest event. Using our algorithms, static or dynamic, although there are more false positives than in the previous experiment (1852 and 1080 respectively), a larger number of real features are correctly identified (1268 and 1656), also leading to a smaller number of false negatives (1256 and 1036), which means that they are also robust to the presence of multiple features mixed in the same consensus process. The number of iterations and information exchanged by the nodes is the same as in the previous experiment.

TABLE III
COMPARISON OF THE DIFFERENT ALGORITHMS (2 EVENTS)

Method	ML [18]	Optim. [12]	Static	Dynamic
Trials	1000			
Total features	2000			
Features Detected	20	28	1268	1656
False Positive Votes	4000	-	1852	1080
False Negative Votes	0	-	1256	1036
Avrg. Norm of Error	16.98	16.24	5.92	3.60
Std. Deviation of Error	8.87	8.58	8.81	6.83
Iterations per Trial	100	100	220	120
Data per iteration	6	6	21	21

We show an example of this situation in Fig. 3, where the solutions of the different algorithms can be visualized. The non-robust solution (ML) and the optimization solution (Optim) return solutions approximately half way between the

two events. For our algorithms, we show the two most voted hypotheses, which correspond to the locations of the two events.

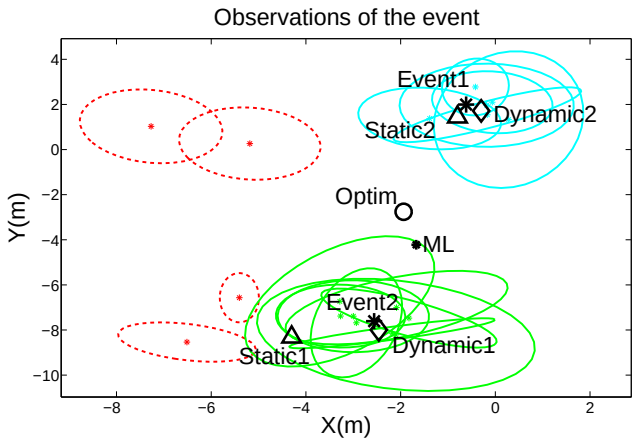


Fig. 3. Simulation considering two events and the robust distributed estimation of their locations. The network is composed by 20 nodes, some of them have seem Event 1 (cyan ellipses), others have seen Event 2 (green ellipses) and some have not observed any useful information (red ellipses). The non robust solution (ML) and the optimization solution (Optim) return solutions approximately half way between the two events whereas the dynamic consensus algorithm (Dynamic), due to the handling of different hypotheses, is able to detect both events with more precision.

C. People Identification in Camera Networks

A challenging problem in computer vision is the identification of people across multiple views. In camera networks with limited communications, the complexity of this problem is increased by the lack of information provided by all the cameras. We present an example to show the possibilities of our algorithm in this context. We consider a network composed by six cameras, A to F, forming a ring topology. Each camera acquires a picture of the scene, not necessarily in the same time instant, and extracts the faces using a Haar classifier with the implementation available in Open CV [28]. For a better interpretation, we have manually assigned the faces identifiers $\in \{1, 2, 3, 4\}$. The patches containing the faces are resized to a fixed dimension of 100x100 pixels. Therefore, each face is described by a 10000 dimensional vector containing the intensities of each of the pixels (values from 0 to 255). Since we are not considering a database of faces to recognize the extracted ones, neighboring cameras match their observations computing the absolute differences between pairs of descriptors. That is,

$$d(I_i, I_j) = \sum_{x=1}^{10000} |I_i(x) - I_j(x)|, \tag{42}$$

where $I_i(x)$ and $I_j(x)$ are the intensities of the pixel x for the faces i and j respectively. Then, two faces of two different images are matched if and only if the distance between their descriptors is smaller than the distance of these faces to any other face.

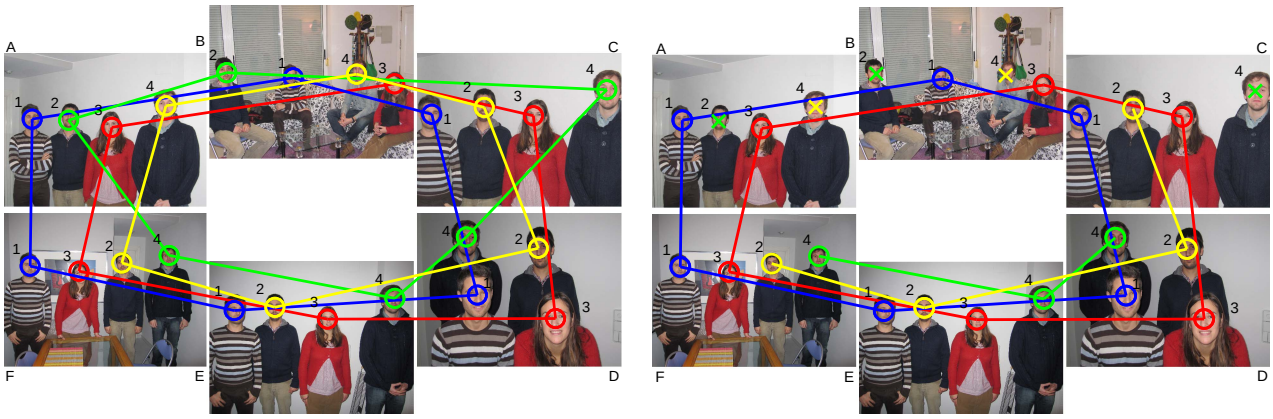
This simple matching algorithm has been chosen for simplicity, but more sophisticated methods as in [29] could be used. Figure 4 (a) shows the initial correspondences of the network. Each color in the figure represents one association set that ideally, should include the same face in the six cameras. However, as we can see, there are some mistakes in the global correspondences. The yellow association, which corresponds to face number 2, contains two entries of face 4 in cameras A and B and the green association, which should contain only entries of face 4, has two entries of face 2 in cameras A and B. This is caused by mistakes matching faces 2 and 4 between cameras A and F and cameras B and C.

We can use our approach to discard the mismatches and improve the data association process. Since our algorithm considers only one datum per camera, the algorithm is executed in parallel 4 times, one for each of the four association sets. Each execution considers as local information the face descriptor belonging to the corresponding association set. For example, when the algorithm is run with the green set, the descriptor of face 2 is used by cameras A and B and the descriptor of face 4 by the rest of the cameras. Assuming the initial data association is relatively good, we set p_{in} to 0.9, which implies that each camera only generates one hypothesis. Since this example does not contain covariance matrices, instead of using the Mahalanobis distance to vote for the hypotheses, we have used the same error function used to match the faces, eq. (42). The threshold to vote for one hypothesis has been empirically set and the robust average of the descriptors has only been used to identify the outliers.

The results of the application of our method are shown in Fig. 4 (b). The algorithm has been able to detect that faces 2 and 4 were wrongly matched in the green and yellow sets respectively, and has removed these faces in cameras A and B. Also, for the blue and red executions, where faces 1 and 3 were correctly matched in all the cameras, the algorithm has not removed any match. On the other hand, the algorithm has also considered as an outlier in the green set face 4 in camera C, which was a correct association (false negative). This happens because of choosing a restrictive threshold for the voting. A possible way to improve this could be to use a smaller threshold in the error function to vote for the hypotheses. However, in real applications it is better to eliminate all the spurious matches rather than missing some good correspondences.

VII. Conclusions

This paper introduces new static and dynamic algorithms that allow a sensor network to compute robust consensus values when nodal information includes outliers. Our approach combines the random sampling consensus algorithm with consensus-algorithmic tools, which enables decentralized, network-consistent implementations, and the handling of large number of outliers. Our algorithms allow nodes to detect locally if their measurement is an outlier or not, and have convergence guarantees under mild assumptions. In particular, the dynamic approach is based on a modification of the standard averaging iteration that converges in finite time. This can be used by nodes to compute the total number of active sensors in the network, and the total number of votes, speeding



Initial correspondences Robust correspondences

Fig. 4. People Identification in Camera Networks. For a better interpretation we have manually assigned the faces identifiers and colors to the association sets. In the initial correspondences (a) faces 2 and 4 have been wrongly matched in the yellow and green sets in cameras A and B. After applying our algorithm in (b) these faces have been removed from the association sets.

up the whole process. Simulation results confirm the good performance of our approach in practice, even in situations in which which not all the requirements are satisfied.

The current algorithm considers hypotheses that require a single observation to be generated. The static voting method could also handle hypotheses generated by several observations by appropriately modifying the first step, letting the nodes gather the information required to generate the different hypotheses. Whether this can be done with dynamic opinions is an open question. Another limitation of the current approach is that it only works with static features. We leave for future work the extension of the algorithms for multi-target tracking problems, with the features representing the time-varying positions of the possible targets, and the investigation of how the method depends on fixed parameters or random sampling.

REFERENCES

[1] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[2] W. Ren. Consensus tracking under directed interaction topologies: Algorithms and experiments. *IEEE Transactions on Control Systems Technology*, 18(1):230–237, January 2010.

[3] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[4] M. Zhu and S. Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, February 2010.

[5] E. Montijano, J. I. Montijano, and C. Sagués. Chebyshev polynomials in distributed consensus applications. *IEEE Transactions on Signal Processing*, 61(3):693–706, March 2013.

[6] M. Franceschelli, M Egersdedt, and A. Giua. Motion probes for fault detection and recovery in networked control systems. In *American Control Conference*, pages 4358–4363, June 2008.

[7] E. Franco, R. Olfati-Saber, T. Parisini, and M. M. Polycarpou. Distributed fault diagnosis using sensor networks and consensus-based filters. In *IEEE Int. Conference on Decision and Control*, pages 386–391, December 2006.

[8] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterations in the presence of malicious agents - part i: Attacking the network. In *American Control Conference*, pages 1350–1356, June 2008.

[9] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterations in the presence of malicious agents - part ii: Overcoming malicious behavior. In *American Control Conference*, pages 1357–1362, June 2008.

[10] F. Pasqualetti, A. Bicchi, and F. Bullo. On the security of linear consensus networks. In *IEEE Int. Conference on Decision and Control*, pages 4894–4901, Shanghai, China, December 2009.

[11] F. Pasqualetti, F. Dorfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58(11):2715–2729, November 2012.

[12] J. Li, E. Elhamifar, I-J Wang, and R. Vidal. Consensus with robustness to outliers via distributed optimization. In *IEEE Conference on Decision and Control*, pages 2111–2117, 2010.

[13] Y. Liu, Y. H. Hu, and Q. Pan. Distributed, robust acoustic source localization in a wireless sensor network. *IEEE Transactions on Signal Processing*, 60(8):4350–4359, August 2012.

[14] R. Carli, F. Bullo, and S. Zampieri. Quantized average consensus via dynamic coding/decoding schemes. *International Journal of Robust and Nonlinear Control*, 20(2):156–175, January 2010.

[15] T. C. Aysal, M. J. Coates, and M. G. Rabbat. Distributed average consensus with dithered quantization. *IEEE Transactions on Signal Processing*, 56(10):4905–4918, October 2008.

[16] S. Kar and J. M. F. Moura. Distributed Consensus Algorithms in Sensor Networks: Quantized Data and Random Link Failures. *IEEE Transactions on Signal Processing*, 58(3):1383–1400, March 2010.

[17] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[18] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedins of the International Conference on Information Processing in Sensor Networks*, pages 63–70, 2005.

[19] E. Montijano, S. Martínez, and C. Sagués. *De-RANSAC*: Robust consensus for robot formations. In *Network Science and Systems in Multi-Robot Autonomy, Workshop at the IEEE International Conference on Robotics and Automation 2010*, May 2010.

[20] E. Montijano, S. Martínez, and C. Sagues. Distributed robust data fusion based on dynamic voting. In *IEEE Int. Conference on Robotics and Automation*, pages 5893–5898, May 2011.

[21] M. Schwager, P. Dames, D. Rus, and V. Kumar. A multi-robot control policy for information gathering in the presence of unknown hazards. In *Proceedings of the International Symposium on Robotics Research*, August 2011.

[22] R. Tron and R. Vidal. Distributed face recognition via consensus on se(3). In *Workshop on Omnidirectional Vision*, 2008.

[23] E. Kokiopoulou and P. Frossard. Distributed classification of multiple observation sets by consensus. *IEEE Transactions on Signal Processing*, 59(1):104–114, January 2011.

[24] E. Montijano and C. Sagues. Distributed multi-camera visual mapping using topological maps of planar regions. *Pattern Recognition*, 44(7):1528–1539, July 2011.

[25] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak. A collaborative approach for in-place sensor calibration. In *IPSN*, volume 2634 of *Lecture Notes in Computer Science*, pages 301–316. Springer-Verlag, 2003.

[26] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann publishers, 1997.

[27] J. Marsden and A. Weinstein. *Calculus, Volume III, Chapter 16*. Applied Mathematics Series. Springer, 1985.
[28] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, pages 511–518, 2001.
[29] R. Garg, D. Ramanan, S. Seitz, and N. Snavely. Where's waldo: Matching people in images of crowds. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1793–1800, 2011.

APPENDIX

A. Computation of the derivative of the Mahalanobis distance

The Mahalanobis distance can be rewritten as

$$\begin{aligned} d_i(\mathbf{w}) &= \sqrt{d_2(\mathbf{w})}, \\ d_2(\mathbf{w}) &= (\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))^T \boldsymbol{\Lambda}_i^{-1} (\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w})), \\ \boldsymbol{\theta}_i(\mathbf{w}) &= \mathbf{P}_i^{-1}(\mathbf{w}) \mathbf{q}_i(\mathbf{w}). \end{aligned} \tag{43}$$

We compute the partial derivative applying the chain rule.

The partial derivative of $\boldsymbol{\theta}_i(\mathbf{w})$ with respect to w_j is a function of $\mathbf{P}_i(\mathbf{w})$ and $\mathbf{q}_i(\mathbf{w})$, whose partial derivatives are

$$\frac{\partial \mathbf{P}_i(\mathbf{w})}{\partial w_j} = \boldsymbol{\Lambda}_j^{-1}, \ \frac{\partial \mathbf{q}_i(\mathbf{w})}{\partial w_j} = \boldsymbol{\Lambda}_j^{-1} \mathbf{x}_j. \tag{44}$$

By passing the inverse matrix to the left member we have

$$\frac{\partial \mathbf{P}_i(\mathbf{w})}{\partial w_j} \boldsymbol{\theta}_i(\mathbf{w}) + \mathbf{P}_i(\mathbf{w}) \frac{\partial \boldsymbol{\theta}_i(\mathbf{w})}{\partial w_j} = \frac{\partial \mathbf{q}_i(\mathbf{w})}{\partial w_j}. \tag{45}$$

Clearing $\partial \boldsymbol{\theta}_i(\mathbf{w}) / \partial w_j$ in (45) and plugging (44) yields

$$\frac{\partial \boldsymbol{\theta}_i(\mathbf{w})}{\partial w_j} = \mathbf{P}_i^{-1}(\mathbf{w}) \boldsymbol{\Lambda}_j^{-1} (\mathbf{x}_j - \boldsymbol{\theta}_i(\mathbf{w})). \tag{46}$$

The partial derivative of $d_2(\mathbf{w})$ with respect to $\boldsymbol{\theta}_i(\mathbf{w})$ is obtained as

$$\begin{aligned} &\frac{\partial (\mathbf{x}_i - \boldsymbol{\theta}_i(w))^T \boldsymbol{\Lambda}_i^{-1} (\mathbf{x}_i - \boldsymbol{\theta}_i(w))}{\partial \boldsymbol{\theta}_i(w)} = \\ &(\mathbf{x}_i - \boldsymbol{\theta}_i(w))^T \boldsymbol{\Lambda}_i^{-T} + (\mathbf{x}_i - \boldsymbol{\theta}_i(w))^T \boldsymbol{\Lambda}_i^{-1} = \\ &2(\mathbf{x}_i - \boldsymbol{\theta}_i(w))^T \boldsymbol{\Lambda}_i^{-1}. \end{aligned} \tag{47}$$

Finally, computing the partial of $d_i(\mathbf{w})$ with respect to $d_2(\mathbf{w})$ and applying the chain rule we get

$$\frac{\partial d_i(\mathbf{w})}{\partial w_j} = \frac{(\mathbf{x}_i - \boldsymbol{\theta}_i(\mathbf{w}))^T \boldsymbol{\Lambda}_i^{-1} \mathbf{P}_i^{-1}(\mathbf{w}) \boldsymbol{\Lambda}_j^{-1} (\mathbf{x}_j - \boldsymbol{\theta}_i(\mathbf{w}))}{d_i(\mathbf{w})}.$$

The derivative is well defined at any point but the set of points such that $\boldsymbol{\theta}_i(\mathbf{w}) = \mathbf{x}_i$, because $d_i(\mathbf{w}) = 0$. However, at this points we already know that $d_i(\mathbf{w})$ has a global minimum because the distance is always positive (or zero). Therefore, they do not affect our analysis in Theorem 4.1.

Eduardo Montijano (M'12) received the M.Sc. and Ph.D. degrees from the Universidad de Zaragoza, Spain, in 2008 and 2012 respectively. He has been a visiting scholar at University of California San Diego, University of California Berkeley and Boston University in the United States and at Royal Institute of Technology, in Stockholm, Sweden. He is currently a Professor at Centro Universitario de la Defensa, in Zaragoza, Spain. His main research interests include distributed algorithms, cooperative control and computer vision. His Ph.D. obtained the extraordinary award of the Universidad de Zaragoza in the 2012-2013 academic year.

Sonia Martínez is an associate professor at the Mechanical and Aerospace Engineering department at UC San Diego. She received her Ph.D. degree in Engineering Mathematics from the Universidad Carlos III de Madrid, Spain, in May 2002. Following a year as a Visiting Assistant Professor of Applied Mathematics at the Technical University of Catalonia, Spain, she obtained a Postdoctoral Fulbright fellowship and held positions as a visiting researcher at UIUC and UCSB. Dr Martínez main research interests include nonlinear control theory, cooperative control and networked control systems. In particular, her work has focused on the modeling and control of robotic sensor networks, the development of distributed coordination algorithms for groups of autonomous vehicles, and the geometric control of mechanical systems. She is also interested in distributed control applications in energy and demand-response systems. For her work on the control of underactuated mechanical systems she received the Best Student Paper award at the 2002 IEEE Conference on Decision and Control. She was the recipient of a NSF CAREER Award in 2007. For the paper Motion coordination with Distributed Information, coauthored with Francesco Bullo and Jorge Cortes, she received the 2008 Control Systems Magazine Outstanding Paper Award.

Carlos Sagüés (M'00, SM'11) received the M.Sc. and Ph.D. degrees from the Universidad de Zaragoza, Spain. During the course of his Ph.D. he worked on force and infrared sensors for robots. Since 1994 he has been Associate Professor and, since 2009 Full Professor with the Departamento de Informática e Ingenieía de Sistemas, Universidad de Zaragoza, where he has also been Head Teacher. His current research interest includes control systems, computer vision, visual robot navigation and multi-vehicle cooperative control.