De-RANSAC: Decentralized RANSAC for Sensor Networks

E. Montijano, S. Martínez, and C. Sagues

Abstract— **This paper studies the problem of distributed consensus in the presence of spurious sensor information. We propose a new method, *De-RANSAC*, which allows a multi-agent system to detect outliers—erroneous measurements or incorrect hypotheses—when the sensed information is gathered in a distributed way. The method is an extension of the RANSAC (RANdom SAmple Consensus) algorithm, which leads to a consensus result on the goodness of a set of measurements with certain probability. In order to execute the full process in a decentralized way, we propose a distributed voting policy valid for fixed and switching topologies. Simulations of real applications are provided showing the reliability of the proposed method.**

Index Terms - Robust consensus, distributed algorithms, sensor networks.

I. Introduction

Recent advances in communication, computation and control are leading to a new generation of autonomous systems that can be capable of complex tasks while interacting over dynamic networks. To exploit their full potential, much research is being devoted to the development of new algorithms and methods that can be executed in a decentralized way by these networks.

Within the control and robotics communities, a canonical problem that has received much attention is that of achieving consensus; see e.g. [1] and references therein. Here the goal is to devise a distributed algorithm that allows a group of agents to agree upon some specific measurement. Several algorithms have been proposed in order to achieve this; however, most of them are not robust to the presence of outliers in the network. If some of the initial measurements are wrong; e.g., they include extreme values of sensor noise, so will be the final consensus value. An example of this kind of situation could be given by a team of robots exploring and mapping the environment. Some of the measurements can be wrong due to sensor failures or wrong data association, leading to inaccurate final consensus values. In this situation least squares cannot be used because it does not discard the wrong data. This paper focuses precisely on this aspect and aims to find a distributed algorithm that can solve this problem.

In centralized scenarios, one of the most widely used algorithms for robust estimation is RANSAC (RANdom SAmple Consensus) [5]. RANSAC is a non-deterministic algorithm designed to fit a set of data to a given mathematical model by selecting inliers from the set. The algorithm generates different random hypotheses that are voted for by the whole set of samples. The most voted hypothesis is then the selected with certain probability. As more hypotheses are tested by the algorithm, the probability of choosing a correct one increases. The algorithm has been used in different applications such as computer vision [7] or robot navigation [3]. In the last few years, new variations to the basic algorithm have been presented in [2], [10], aiming to reduce computational time or to improve the algorithm effectiveness. However, none of these algorithms are implementable in a distributed way over a sensor network. The closest approach to a distributed scenario can be found in [17]. In this case the information to be fitted is separated in several non-overlapping subsets and the hypotheses are generated choosing data from them. However, the whole process is still centralized.

Different approaches have been proposed to cope with noisy information in distributed consensus algorithms. For instance, [16] considers a stochastic model to deal with the uncertainty of the measurements. Here, the authors prove convergence to the least-mean-square deviation value using a distributed iteration rule. In [13] stochastic link failures are considered and convergence is defined in terms of the variance deviation. These works are focused on the optimal mix of the measures minimizing the error but do not consider the situations in which some of the measurements are spurious and must be discarded in the fusion.

Other works consider the case in which one or more nodes of the network are faulty or malicious. In [6], different “motion probes” to detect malicious nodes are proposed. Specifically the motion probes detect static and divergent nodes and suggest possible techniques to recover from the influence of these agents once they have been detected. A more general approach is considered in [14] for a fixed graph. In this case, the malicious agent is able to update its value in an arbitrary manner. The paper shows that if the number of malicious agents is less than half the connectivity of the network, which is the maximum number of different paths that connect any two nodes of the network, it is possible to detect the malicious agents and compute the desired consensus function using the right initial values [15]. In [12] this bound is improved, increasing the number of possible malicious agents to the connectivity of the network minus one. The situation we consider here is somewhat different. We assume that all the nodes in the network are cooperative,

This work was supported by the projects DPI2006-07928, DPI2009-08126, IST-1-045062-URUS-STP.

E. Montijano and C. Sagues are at DIIS - I3A, University of Zaragoza, emonti@unizar.es, csagues@unizar.es

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

however, some of them have spurious information and there is no knowledge about which nodes are outliers.

Our contribution is a new robust distributed consensus method, *De-RANSAC*, in which the outlier nodes are identified. The present approach is an extension of RANSAC to a distributed setup. The agents reach an agreement about the information they have, which can be a non-linear function computed from the individual information that each agent has. It is worth noticing that just sharing all the information over the network and applying the classic RANSAC can make each node reach different final solutions due to the non-deterministic component of the algorithm. *De-RANSAC*, explained in detail in section III, solves this problem and can be divided in two steps. First, each node shares its local information with its neighbors and generates a number k_i of different hypotheses to be evaluated. The total number of hypotheses is proved to be enough to achieve the same levels of reliability as in the centralized case. In the second step, all the hypotheses are voted for in a distributed way, so that each node can determine which model fits most of the information and therefore, whether it is an inlier or not. Several simulations, presented in section IV, show the reliability and the applications of our proposal.

II. Preliminary Concepts

Here, we briefly introduce the classic RANSAC algorithm and pose its extension to a distributed scenario. We also introduce some notation and definitions used in the sequel.

The RANSAC algorithm was proposed in [5] to estimate the parameters of a model that can be used to explain sampled data. In particular, this data set can be contaminated from a large set of *outliers*. An outlier is a datum that lies outside a pattern of a distribution; this is in opposition to an *inlier*, which is considered to be truthful information [9]. If the pattern is described by a mathematical model, then an error threshold can be used to quantify acceptable deviations and outliers with respect to the model.

More precisely, assume that we have a set of mathematical models $\mathcal{H}$ which are instantiated by specifying different parameter values. Let S be a set of samples. Then, given a threshold $\tau > 0$, and an error function $e : S \times \mathcal{H} \to \mathbb{R}_{\geq 0}$, we say that $s \in S$ is an outlier with respect to $h \in \mathcal{H}$ if $e(s,h) > \tau$; otherwise it is an inlier with respect to $h \in \mathcal{H}$.

Suppose that at least c samples are required in order to determine a possible model. A model is considered to be good if it is generated by c truthful samples. Instead of trying all the possible combinations to generate models, RANSAC generates a subset of the models and chooses one based on a voting system. The algorithm follows the next steps:

1) Consider a set of putative samples S with $|S| = n > c$.
2) Choose $K > 0$ and determine $H_\ell \subset S$ with $|H_\ell| = c$, for all $\ell = 1, \ldots, K$.
3) For each H_ℓ, find a model $h_\ell \in \mathcal{H}$ that best fits the samples in H_ℓ. For example, one can use the least squares procedure:
$$h_\ell \in \arg\min_{h \in \mathcal{H}} \sum_{s_\ell \in H_\ell} \text{dist}(s_\ell, h)^2.$$
Here, $\text{dist}(s_\ell, h)^2$ is a positive number quantifying the (squared) distance between the sample point s_ℓ and an associated datum generated by $h \in \mathcal{H}$.
4) Rank the models h_ℓ, $\ell = 1, \ldots, K$, according to how well they fit all of the samples in S. This is done through a "voting process" of the $s \in S$ onto the h_ℓ, for all $\ell = 1, \ldots, K$. In other words, given a threshold value $\tau > 0$, a vote is generated as:
$$\text{vote}(s, h_\ell) = \begin{cases} 1, & e(s, h_\ell) \leq \tau, \\ 0, & e(s, h_\ell) > \tau. \end{cases} \tag{1}$$
5) Choose the most voted model,
$$h^* = \arg\max_{\ell=1,\ldots,K} (\sum_{s \in S} \text{vote}(s, h_\ell)) \tag{2}$$
6) Given the subset of inliers for h^*; i.e., $I_{h^*} = \{s \in S \,|\, e(s, h^*) \leq \tau\}$, determine a better fit from $\mathcal{H}$. This can be done by least squares:
$$h^{**} = \arg\min_{h \in \mathcal{H}} \sum_{s \in I_{h^*}} \text{dist}(s, h)^2. \tag{3}$$

RANSAC uses a stochastic approach to decide the number of hypothetical sample subsets, H_ℓ, (also called hypotheses) to be evaluated. If the probability that a sample s is an inlier to the true model is w, then the probability of one hypothesis to be composed only by inliers is w^c. Therefore, the probability that one hypothesis contains at least one outlier is $1 - w^c$. Then, the number of hypotheses, K, required to have a probability P that one of them is made all by inliers is:
$$K = \frac{\log(1-P)}{\log(1-w^c)}. \tag{4}$$
When the algorithm is run by a central unit that has access to the all the sensor data, the hypotheses can be assumed to have the same probability of being chosen. This is no longer reasonable for nodes in a sensor network that try to run a similar algorithm with access only to local samples.

In the sequel, we will consider a network with undirected communications formed by N agents labeled by $i \in \mathcal{V} = \{1, \ldots, N\}$, $N > c \geq 2$. Each node i produces a sample s_i, $i \in \mathcal{V}$. Communications among agents are defined according to a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represents the edge set. In this way, nodes i and j can communicate if and only if $(i,j) \in \mathcal{E}$. The neighbors of a node i are the sets of nodes that can directly communicate with the node i, $\mathcal{N}_i = \{j \in \mathcal{V} \setminus (i,j) \in \mathcal{E}\} \cup \{i\}$. Although node i does not explicitly communicate with itself, we will consider that $i \in \mathcal{N}_i$ because it has access to the information measured by its sensor. Considering this, the maximum number of possible edges of $\mathcal{G}$ is $|\mathcal{E}_{max}| = \frac{(N)(N-1)}{2}$. The density of the graph, δ, is defined to be $\delta = \frac{|\mathcal{E}|}{|\mathcal{E}_{max}|} \leq 1$, where $|\mathcal{E}|$ is the actual number of edges in the graph.

***Assumption** 2.1:* $\mathcal{G}$ is connected. Therefore $|\mathcal{E}| \geq N-1$ and $\delta \geq \frac{2}{N-2}$.

We will denote $\mathcal{N}_i^{in}$ and $\mathcal{N}_i^{out}$ as the set of neighbors with inlier and outlier information respectively. Obviously $\mathcal{N}_i = \mathcal{N}_i^{in} \cup \mathcal{N}_i^{out}$.

III. De-RANSAC Algorithm

This section describes our proposed algorithm. In a first phase, the random hypotheses are generated in a distributed way. The second phase deals with the problem of voting the generated hypotheses by all the nodes in the network.

A. Generation of the hypotheses

The first issue to address is how to create the hypotheses over the network so that the probability of having one hypothesis made all by inliers is the same as in the centralized case. The required number of hypotheses to have a probability of success equal to P is the same as in the centralized case, eq. (4). However, each node has access to the information gathered by itself and its neighbors, and then their hypotheses are limited to combinations of that set.

Since the information available to every node is limited to its neighbors, the topology of the network will play a fundamental role in *De-RANSAC*. One sensor producing spurious measurements and connected to others in a critical way can make the whole set of hypotheses to have outliers in their composition. For example, consider the family of spanning tress in which the root of the tree has up to $c-1$ sons and every other node has a number of neighbors less than or equal to $c-2$. The only nodes that are able to generate hypotheses in this situation are the root and its sons. Let us notice that all the hypotheses will contain the sample provided by the root. If this sample is the spurious element, none of the hypotheses will be free of outliers and the algorithm will fail. On the other hand if $\exists i \in \mathcal{V}$ such that $|\mathcal{N}_i^{in}| \geq c$ and $|\mathcal{N}_i^{out}| = 0$ then *De-RANSAC* will generate one good combination with probability 1 independently of the configuration of the rest of the nodes.

The following lemma gives a restriction on the topology of the network so that the distributed sampling can work.

Lemma 3.1: If a graph G is able to generate a hypothesis made all by inliers then

$$\exists\, i \in \mathcal{V} \ni |\mathcal{N}_i^{in}| \geq c.$$

Proof. Let us assume that $|\mathcal{N}_i^{in}| < c\ \forall i$. Let $\mathcal{H}_i \subseteq \mathcal{H}$ be the set of hypotheses that the node i can create. It is true that $\forall h \in \mathcal{H}_i\ \exists\, s_j \in \mathcal{N}_i$ such that $s_j \in \mathcal{N}_i^{out}$, and therefore, none of the hypotheses generated by the node i will be composed all by inliers. Since this holds for all i the method will not be able to create any good combination. ∎

Let us note that if one node is able to generate one combination made by inliers then the algorithm is able to generate one combination made by inliers. However, due to its stochastic component, this does not mean that the algorithm is always going to generate such combination, as happens in the centralized version. From now on it is assumed that lemma 3.1 holds so that the probability of generating at least one combination made all by inliers does not vanish.

The restriction could be lifted up by allowing the nodes to exchange more messages in this first stage. Using a flooding algorithm during t steps, $t \in 1,\ldots,\mathrm{Diam}(\mathcal{G})$, the nodes would receive more samples acquired by farther nodes at each time step until $t = \mathrm{Diam}(\mathcal{G})$. At that point all the agents in the network would have access to all the samples, being able to generate the hypotheses as in the centralized case. However, due to the random component of the algorithm, the hypotheses generated by each node would be different and so would be the final result achieved by each node. Since the nodes will also require to share the hypotheses with the whole network in order to vote them, we have decided to limit the number of messages in this stage in order to reduce the communications.

Each node i must generate a set $\mathcal{H}_i$, with $|\mathcal{H}_i| = k_i$ different hypotheses in such a way that $\sum_i k_i \geq K$. The maximum number of hypotheses that one node can generate is $\binom{|\mathcal{N}_i|}{c}$. It is clear that nodes with less than c neighbors will not be able to generate any hypothesis. The following assumption is made:

Assumption 3.1: The number of possible hypotheses that the network can generate is greater than K, that is

$$\sum_{i \in \mathcal{V}} \binom{|\mathcal{N}_i|}{c} > K.$$

The assumption guarantees that it is possible to generate at least K hypotheses. However, the previous sum can be much larger than K. We propose a better distribution of k_i that still ensures K or more hypotheses. It is assumed that agents know the total number of nodes in the network, N, and have some estimation of the graph density, $\frac{2}{N} \leq \hat{\delta} \leq 1$. A node can then estimate its number of neighbors as:

$$\hat{\mathbf{E}}[\mathcal{N}] = 1 + \frac{2\hat{\delta}}{N}|\mathcal{E}_{max}| = 1 + (N-1)\hat{\delta}, \tag{5}$$

where the additional neighbor comes from the assumption that $i \in \mathcal{N}_i$. If all the nodes had the same number of neighbors it would make sense that $k_i = \frac{K}{N}$, $\forall i$. Since this is not easily verifiable and will not hold in most situations, an alternative number of hypotheses for each node is proposed:

$$k_i = \begin{cases} \left\lceil \dfrac{K|\mathcal{N}_i|}{c\,\hat{\mathbf{E}}[\mathcal{N}]} \right\rceil, & \text{if } \left\lceil \dfrac{K|\mathcal{N}_i|}{c\,\hat{\mathbf{E}}[\mathcal{N}]} \right\rceil \leq \dbinom{|\mathcal{N}_i|}{c}, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where $\lceil \cdot \rceil$ is the closest upper integer.

Let $\mathcal{V}_1 \subset \mathcal{V}$ be the set of nodes that generate hypotheses; that is, $i \in \mathcal{V}_1$ if and only if $k_i > 0$ using rule (6). Then, we have:

Proposition 3.1: A sufficient condition for having $\bar{K} \geq K$ hypotheses using (6) is that there are $\hat{\mathbf{E}}[\mathcal{N}]$ nodes in the network with $\alpha = \min(N, c\sqrt[c-1]{\frac{K}{\hat{\mathbf{E}}[\mathcal{N}]}+1})$ neighbors each.

Proof. The total number of hypotheses generated by the network is

$$\bar{K} = \sum_{i \in \mathcal{V}_1} k_i = \sum_{i \in \mathcal{V}_1} \left\lceil \frac{K|\mathcal{N}_i|}{c\,\hat{\mathbf{E}}[\mathcal{N}]} \right\rceil \geq \frac{K}{c\hat{\mathbf{E}}[\mathcal{N}]} \sum_{i \in \mathcal{V}_1} |\mathcal{N}_i|.$$

Any node belonging to $\mathcal{V}_1$ will have at least c neighbors, therefore

$$\bar{K} \geq |\mathcal{V}_1| \frac{K}{\hat{\mathbf{E}}[\mathcal{N}]}, \tag{7}$$

which is greater than K if $|\mathcal{V}_1| \geq \hat{\mathbf{E}}[\mathcal{N}]$.

With respect to the number of neighbors required to generate hypotheses, taking into account again that $|\mathcal{N}_i| \geq c$ then

$$\left\lceil \frac{K|\mathcal{N}_i|}{c\,\hat{\mathbf{E}}[\mathcal{N}]} \right\rceil \leq \frac{K|\mathcal{N}_i|}{c\,\hat{\mathbf{E}}[\mathcal{N}]} + 1 \leq \frac{K|\mathcal{N}_i|}{c\,\hat{\mathbf{E}}[\mathcal{N}]} + \frac{|\mathcal{N}_i|}{c}. \quad (8)$$

The combinatoric number can be lower bounded by

$$\binom{|\mathcal{N}_i|}{c} = \frac{|\mathcal{N}_i|}{c} \prod_{k=1}^{c-1} \frac{|\mathcal{N}_i| - k}{c-k} \geq \left(\frac{|\mathcal{N}_i|}{c}\right)^c. \quad (9)$$

The bound comes from the property that $(|\mathcal{N}_i| - k)c \geq (c-k)|\mathcal{N}_i|, \ \forall\ 1 \leq k \leq c-1$. From (8) and (9) a sufficient condition in the number of neighbors to belong to $\mathcal{V}_1$ is

$$|\mathcal{N}_i| \geq c \sqrt[c-1]{\frac{K}{\hat{\mathbf{E}}[\mathcal{N}]} + 1} = \alpha. \quad (10)$$

Combining the sufficient conditions in (7) and (10) we obtain the sufficient condition for rule (6) to generate more than K hypotheses. ■

Since $|\mathcal{N}_i|$ and $|\mathcal{V}_1|$ are integers but α and $\mathbf{E}[\hat{\mathcal{N}}]$ are reals we also take the closest upper integers, which also satisfy the bounds in (7) and (10).

An additional problem to be analyzed in the distributed scenario with respect to the centralized one is the appearance of repeated hypotheses in different nodes. Since the nodes do not know the hypotheses created by other nodes, it is possible that two nodes generate equal hypotheses. A way to overcome this is to generate additional hypotheses to keep the probability of having a good one similar to the probability in the centralized case.

From now on, we will assume that every edge in the graph can exist with equal probability; thus, every combination can be made by any node. This kind of random graphs were proposed by Erdos and Renyi [4]. Considering this, the problem of computing repeated hypotheses generated by different nodes is a problem of counting of Pólya. There is a set of $\binom{N}{c}$ labeled combinations and each node chooses randomly k_i combinations from the set. Although the number of hypotheses generated is $\bar{K}$, in order to keep the process decentralized the nodes assume that the total number of hypotheses is K. The probability of repeating one of the $K - k_i$ combinations chosen by the rest of the nodes is equal to

$$p_r = \frac{K - k_i}{\binom{N}{c}},$$

and therefore, the probability of creating a new different combination is $1 - p_r$.

It is assumed that each node is able to pick the combinations in such a way that all of them are different but may be equal to the combinations generated by other nodes. The probability of having at least k_i new combinations after node i creates x_i combinations is

$$C(x_i, k_i, p_r) = \sum_{m=0}^{x_i - k_i} \binom{x_i}{m} (1-p_r)^{x_i - m} p_r^m. \quad (11)$$

The probability desired for a node to create k_i hypotheses different to the hypotheses generated by the rest of nodes is denoted as P_{dif}. Given this probability, the final number of hypotheses created by the node must satisfy

$$C(x_i, k_i, p_r) \geq P_{dif}. \quad (12)$$

Since it is not possible to obtain the exact analytic solution for (12), an approximated bound is found.

***Theorem** 3.1:* Consider $k_i > 1$ and the probabilities p_r and P_{dif}, defined above. Taking

$$x_i \geq e^{-(a+1)+\sqrt{(a+1)^2 + 2(b-1)}}, \quad (13)$$

with

$$a = \frac{k_i}{\log p_r} \text{ and } b = \frac{\log(1 - P_{dif})}{\log p_r} + k_i - 1, \quad (14)$$

(12) is satisfied.

Proof. Using the binomial theorem for $1 = ((1-p_r)+p_r)^{x_i}$ we can rewrite (11) as

$$\begin{aligned} C(x_i, k_i, p_r) &= \sum_{m=0}^{x_i - k_i} \binom{x_i}{m} (1-p_r)^{x_i - m} p_r^m \\ &= 1 - \sum_{m=x_i-k_i+1}^{x_i} \binom{x_i}{m} (1-p_r)^{x_i-m} p_r^m. \end{aligned}$$

Changing the indices in the sum, we obtain

$$C(x_i, k_i, p_r) = 1 - \sum_{m=0}^{k_i - 1} \binom{x_i}{m} (1-p_r)^m p_r^{x_i - m},$$

and then (12) can be stated as

$$\sum_{m=0}^{k_i-1} \binom{x_i}{m} (1-p_r)^m p_r^{x_i-m} \leq 1 - P_{dif}. \quad (15)$$

The left-hand side of the above equation can be rewritten as:

$$\sum_{m=0}^{k_i-1} \binom{x_i}{m} (1-p_r)^m p_r^{x_i-m} =$$

$$\frac{x_i! p_r^{x_i-k_i+1}}{(k_i-1)!} \sum_{m=0}^{k_i-1} \frac{(k_i-1-m)!}{(x_i-m)!} \binom{k_i-1}{m} (1-p_r)^m p_r^{k_i-1-m}.$$

The expression is upper bounded considering the maximum of $(k_i - 1 - m)!$, attained by $m = 0$, and the minimum of $(x_i - m)!$, by $m = k_i - 1$. Both factors are taken out of the sum and the binomial theorem is used again

$$\sum_{m=0}^{k_i-1} \binom{x_i}{m} (1-p_r)^m p_r^{x_i-m} < p_r^{x_i-k_i+1} \frac{x_i!}{(x_i-k_i+1)!}.$$

The quotient can be upper bounded by x_i^k, yielding

$$p_r^{x_i-k_i+1} \frac{x_i!}{(x_i-k_i+1)!} < p_r^{x_i-k_i+1} x_i^{k_i}.$$

We can guarantee that (15) holds if the following is true:

$$p_r^{x_i-k_i+1} x_i^{k_i} \leq 1 - P_{dif}$$

Taking logarithms at both sides of the above equation lead to:

$$(x_i - k_i + 1)\log(p_r) + k_i \log(x_i) \leq \log(1 - P_{dif}),$$

and, after some computations, the following inequality is obtained

$$x_i + a\log(x_i) - b \geq 0, \tag{16}$$

with $a < 0$ and $b > 0$ as in eq. (14). The change of variable $x_i = e^c$ is then applied

$$e^c + ac - b = \sum_{m=0}^{\infty} \frac{c^m}{m!} + ac - b > \sum_{m=0}^{2} \frac{c^m}{m!} + ac - b.$$

By setting the right-hand side of the above equation greater or equal to zero, we obtain the inequality:

$$c^2 + 2(a+1)c + 2(1-b) \geq 0.$$

Since the coefficient of c^2 is positive, the previous inequality is true in the interval $(-\infty, c_1] \cup [c_2.\infty)$, with c_1 and c_2 the roots of the polynomial. The smallest root of the polynomial is negative for $k_i \geq 2$. Because of that, we take the positive root of the polynomial:

$$c_2 = -(a+1) + \sqrt{(a+1)^2 + 2(b-1)},$$

By taking $x_i \geq e^{c_2}$ we guarantee the result for all k_i. ∎
Although this is a valid bound, it can be overly conservative. A more precise bound to determine the minimum x_i can be found by using; e.g. the Newton's method. To do this, we will employ (16), which is continuous and differentiable, instead of (11), which is discrete. Let $f(x) = x + a\log(x) - b$ be the function defined in (16). A classical result on the global convergence on an interval $\mathcal{I}$ of the Newton's method [11] states that if

$$\begin{cases} f'(x_i) = 1 + \frac{a}{x_i} \neq 0 \text{ for all } x_i \in I, \\ \text{sign}(f''(x_i)) = \text{sign}(-a/x_i^2) \text{ constant for all } x_i \in I, \end{cases} \tag{17}$$

then the method converges for any $x_i(0) \in \mathcal{I}$ such that $f(x_i(0))f''(x_i(0)) \geq 0$. Eq. (17) are satisfied for all $x_i > -a$ and therefore the method is convergent to the root of $f(x)$. New approximations of the solution are computed using

$$x_i(n+1) = x_i(n) - \frac{f(x_i(n))}{f'(x_i(n))}, \tag{18}$$

where the initial term $x_i(0)$ can be the one in (13). Let us notice that the first derivative of f is close to a constant for large values of x_i. This means that f behaves like a line when x_i is large, and therefore, convergence will be obtained in few steps.

B. Voting the hypotheses

With all the hypotheses created, the nodes must then vote for them in order to decide which one is the best one. This means that the hypotheses must be transmitted over the network so that every node receives them. We propose a general procedure to vote the samples on a switching topology network. The algorithm is based on a distributed average consensus technique.

Let $\mathcal{H}_\mathcal{G} = \bigcup_{i\in\mathcal{V}} \mathcal{H}_i$, $|\mathcal{H}_\mathcal{G}| = \bar{K}$ be total set of hypotheses generated by the network. Before starting to vote, the nodes share all the hypotheses with all the other nodes [8]. With all the hypotheses available a distributed average consensus is used to decide the most voted one. Assumption 2.1 is relaxed to:

***Assumption** 3.2:* There exists a positive integer T 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.
Instead of reaching the average of the measurements, the agreement is performed with respect to the number of votes of the different hypotheses. Every node creates two state vectors $\mathbf{v}_i(t) \in \mathbb{N}^{\bar{K}}$, which counts the number of votes that each hypothesis has, and $\boldsymbol{\gamma}_i(t) \in \mathbb{R}^{\bar{K}}$, used for the averaging. An important aspect to consider is how the hypotheses are sorted in the vector so that for all the nodes the l^{th} component of $\mathbf{v}_i$ and $\boldsymbol{\gamma}_i$, denoted as v_{il} and γ_{il} respectively, corresponds to the same hypothesis for all $i \in \mathcal{V}$. Since the hypotheses are instantiated by a set of different parameters, these parameters can be used to define a global order function, $o : \mathcal{H}_\mathcal{G} \to \{1, \ldots, \bar{K}\}$, among the set of hypotheses.

At the beginning, for every $l = 1, \ldots, \bar{K}$, the l^{th} component of $\boldsymbol{\gamma}_i$, is initialized as

$$\gamma_{il}(0) = \begin{cases} 1, & e(s_i, h_\ell) \leq \tau, \\ 0, & e(s_i, h_\ell) > \tau, \end{cases} \tag{19}$$

where $h_l = \{h \in \mathcal{H}_\mathcal{G} \mid o(h) = l\}$ and $\tau > 0$ a given threshold. The voting vectors are initialized by multiplying by N the averaging vectors, $\mathbf{v}_i(0) = N\boldsymbol{\gamma}_i(0)$.

After that the nodes exchange messages averaging the values of the $\boldsymbol{\gamma}_i$. The local updating rule, executed synchronously by every node, is

$$\mathbf{v}_i(t+1) = \arg\min_{\mathbf{v}\in\mathbb{N}^{\bar{K}}} ||\mathbf{v} - N\boldsymbol{\gamma}_i(t)||_2, \tag{20}$$

$$\boldsymbol{\gamma}_i(t+1) = \boldsymbol{\gamma}_i(t) + \sum_{j\in\mathcal{N}_i} a_{ij}(t)(\boldsymbol{\gamma}_j(t) - \boldsymbol{\gamma}_i(t)), \tag{21}$$

where $\mathbf{A}(t) = \{a_{ij}(t)\}$ are the weighted matrices, which have the property of being doubly stochastic.

***Theorem** 3.2:* The iteration rule in (20) converges to the number of votes of each hypothesis in finite time, that is

$$\exists t_0 > 0 \ni \forall t > t_0,\ \mathbf{v}_i(t) = \mathbf{v}_i(t_0) = \mathbf{v}^*,\ \forall i \in \mathcal{V}$$

Proof. For any $l = 1, \ldots, \bar{K}$, γ_{il} has initial values equal to 0 or 1; hence

$$\sum_{i=1,\ldots,N} \gamma_{il}(0) \in \{0, \ldots, N\}, \quad \forall l = 1, \ldots, \bar{K}. \tag{22}$$

Taking into account that the $\mathbf{A}(t)$ are doubly stochastic and assumption 3.2 we know that (21) will asymptotically converge to the average of the initial values for all the nodes,

$$\boldsymbol{\gamma}_i(t) = \frac{1}{N}\sum_{j\in\mathcal{V}} \boldsymbol{\gamma}_j(t) = \bar{\boldsymbol{\gamma}}, \text{ as } t \to \infty \quad \forall i \in \mathcal{V}. \tag{23}$$

We refer the reader to [1] for a proof of this convergence. The asymptotic convergence allows us to find t_0 such that

$$\exists\, \epsilon > 0 \ni \forall t > t_0 \quad ||\bar{\boldsymbol{\gamma}} - \boldsymbol{\gamma}_i(t)||_2 < \epsilon, \quad \forall i \in \mathcal{V}.$$

Considering (22) and (23), the set of possible consensus values of each component of $\bar{\boldsymbol{\gamma}}$ is $\{0, \frac{1}{N}, \frac{2}{N}, .., 1\}$, which is finite. Let us note that $N\bar{\boldsymbol{\gamma}}$ represents the total number of

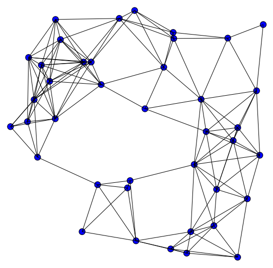

Fig. 1. Graph used for the robust average simulations.

votes that the hypotheses have. By choosing $\epsilon < \frac{1}{2N}$ we see that for any $t > t_0$

$$\begin{aligned}
||\bar{\gamma} - \gamma_i(t)||_2 &< \frac{1}{2N} \Rightarrow \\
||N\bar{\gamma} - N\gamma_i(t)||_2 &< \frac{1}{2} \Rightarrow \\
||N\bar{\gamma} - N\gamma_i(t)||_2 &< ||\mathbf{v} - N\gamma_i(t)||_2, \forall \mathbf{v} \in \mathbb{N}^{\bar{K}} \setminus N\bar{\gamma},
\end{aligned}$$

and therefore $N\bar{\gamma} = \arg\min_{\mathbf{v}\in\mathbb{N}^{\bar{K}}} ||\mathbf{v} - N\gamma_i(t)||_2 = \mathbf{v}_i(t+1), \forall t > t_0$, and $i \in \mathcal{V}$. In other words, the iteration rule in (20) converges to $\mathbf{v}^* = N\bar{\gamma}, \forall t > t_0$, and $i \in \mathcal{V}$. ∎

After the iteration, the largest value of the vector $\mathbf{v}_i$ will correspond to the hypothesis with the most number of votes. Since the hypotheses are sorted in the vector, if there exists a tie between two or more hypotheses every node will keep the one with the smallest index,

$$\begin{aligned}
l^* &= \min\left(\arg\max_{l=1,\dots,\bar{K}} v_{il}\right) \\
h^* &= \{h \in \mathcal{H}_{\mathcal{G}} \mid o(h) = l^*\} \qquad (24)
\end{aligned}$$

The De-RANSAC algorithm is summarized in Algorithm 1.

```
Algorithm 1 De-RANSAC scheme - Agent $i$
Require: $N, c, w, P, P_{dif}, \hat{\delta}, \tau, o, e$ and synchronicity
Ensure: $h^*$ is the most voted hypothesis and equal $\forall i \in \mathcal{V}$
1: Compute $K = \frac{\log(1-P)}{\log(1-w^c)}$
2: Receive the samples $s_j$ for $j \in \mathcal{N}_i$
3: Compute $\hat{\mathbf{E}}[\mathcal{N}] = 1 + (N-1)\hat{\delta}$, and
     $k_i = \begin{cases} \left\lceil \frac{K|\mathcal{N}_i|}{c\,\hat{\mathbf{E}}[\mathcal{N}]} \right\rceil, & \text{if } \left\lceil \frac{K|\mathcal{N}_i|}{c\,\hat{\mathbf{E}}[\mathcal{N}]} \right\rceil \le \binom{|\mathcal{N}_i|}{c}, \\ 0, & \text{otherwise}, \end{cases}$
4: Initialize $x_i(0) = e^{-(a+1)+\sqrt{(a+1)^2+2(b-1)}}$ with
     $a = \frac{k_i}{\log p_r}, b = \frac{\log(1-P_{dif})}{\log p_r} + k_i - 1$
   and $p_r = \frac{K-k_i}{\binom{N}{c}}$
5: Compute a better bound of $x_i$ by
     $x_i(n+1) = x_i(n) - \frac{f(x_i(n))}{f'(x_i(n))}$,
6: Share the hypotheses with the network [8]
7: Initialize $\mathbf{v}_i$ and $\gamma_i$,
     $\gamma_{il}(0) = \begin{cases} 1, & e(s_i, h_\ell) \le \tau, \\ 0, & e(s_i, h_\ell) > \tau, \end{cases}$ and $v_{il}(0) = N\gamma_{il}(0)$,
   $l = 1\dots,\bar{K}$ and $h_l = \{h \in \mathcal{H}_{\mathcal{G}} \mid o(h) = l\}$
8: Decentralized voting
     $\begin{cases} \mathbf{v}_i(t+1) = \arg\min_{\mathbf{v}\in\mathbb{N}^{\bar{K}}} ||\mathbf{v} - N\gamma_i(t)||_2, \\ \gamma_i(t+1) = \gamma_i(t) + \sum_{j\in\mathcal{N}_i} a_{ij}(t)(\gamma_j(t) - \gamma_i(t)), \end{cases}$
9: Select $h^*$
     $\begin{cases} l^* = \min\left(\arg\max_{l=1,\dots,\bar{K}} v_{il}\right), \\ h^* = \{h \in \mathcal{H}_{\mathcal{G}} \mid o(h) = l^*\} \end{cases}$
```

IV. Applications of De-RANSAC

The *De-RANSAC* method can be used in different scenarios. We present here some possible applications of the algorithm.

A. Robust Average Consensus

Let us consider a network like the one in Fig. 1. The network is composed by $N = 40$ nodes, each one with a different measurement of some magnitude (whose real value in the example is 5). All the nodes have some noise in their measurements, some of them small (inliers) and some of them very large (outliers). The network intends to compute a robust average of the measurements. Several experiments, two of which are reported, have been done with different probabilities of having a small error; in the first case w is set to 0.8 and in the second $w = 0.3$. The values of the nodes for the first case are collected in the Table I where we have marked in red the 8 spurious measurements. In a similar way, measurements for the second experiment are in Table II. The probabilities to decide the total number of hypotheses, eqs. (4) and (12), have been both set to $P = P_{dif} = 0.99$. The threshold for voting one hypothesis, τ, has been set to 0.5. The results of the two simulations are shown in Table III. The Avrg column shows the average of the information provided by all the nodes, which has in both cases a large error caused by the measurements of the outliers. Robust_Avrg represents the value of the most voted hypotheses. Let us notice that in both cases the error is smaller than the tolerance fixed. K_cent is the number of hypotheses required for the algorithm in the centralized case whereas K_dist is the number of hypotheses generated in the distributed scenario. In both cases the decentralized algorithm generates enough hypotheses and all the outliers have been identified.

B. Analysis of the number of hypotheses

It is also interesting to analyze how the generation of the hypotheses works in the distributed scenario. Choosing $E[\hat{\mathcal{N}}] = 34$ it holds that there are 34 agents with $\alpha = 5 \le |\mathcal{N}_i|$, and therefore, the number of hypotheses will be bigger than K. Table IV shows some results in this regard. Max_k_i shows the maximum k_i among the nodes, Sum_k_i represents the total number of hypotheses generated using

TABLE I

VALUES OF THE NODES CASE 1 ($w = 0.8$. OUTLIERS IN RED)

Values							
8.75	9.01	4.95	4.96	5.10	5.12	5.11	4.91
4.91	9.00	9.90	5.06	5.11	8.60	5.09	5.02
5.09	4.87	4.88	4.87	9.43	4.96	5.04	4.89
5.05	8.98	5.06	5.07	4.95	4.98	5.10	6.25
4.89	4.93	5.12	5.06	4.92	4.99	5.11	4.92

TABLE II

VALUES OF THE NODES CASE 2 ($w = 0.3$. OUTLIERS IN RED)

Values							
5.11	6.17	5.58	9.98	7.47	5.16	9.40	6.15
4.92	6.50	9.30	8.39	8.28	9.36	5.12	8.11
7.72	9.50	4.98	6.85	9.63	8.36	8.47	5.50
7.51	9.72	8.75	6.93	5.00	6.68	8.16	8.98
4.90	5.00	4.97	5.11	4.92	7.50	8.95	7.00

eq. (6) and $\bar{k}$ is the average number of hypotheses. Max_x_i is the maximum number of additional hypotheses created, Sum_x_i is the total number of additional hypotheses to avoid repeated combinations and $\bar{x}$ is the average. The maximum number of hypotheses that the graph can generate is 1205, which is greater than the number of hypotheses generated with our formula (6). In the first case, since the number of inliers is high only one hypotheses is required in each node without additional combinations. The second case requires many extra hypotheses. Most of the nodes generate all their possible combinations because the probability of repeating combinations is higher and δ is small.

C. *Formation of robots*

Another application of *De-RANSAC* can be the formation of teams of robots. Each robot has access to its own position and the positions of the robots inside its communication or perception range. The robots must deploy in a specific configuration; e.g., a circumference. The least squares solution to the problem requires all the robots to know all the positions of all others. Moreover, depending on these positions the least squares solution can lead to an impossible configuration. Each robot generates different hypotheses of formations using only the information it can measure. After sharing the models with the rest of the robots in the network the distributed averaging procedure starts. At each step the robots move towards the current most voted hypothesis.

TABLE III

RESULTS OF DE-RANSAC

	w	Avrg	Robust_Avrg	Votes	K_cent	K_dist
Case 1	0.8	5.75	4.99	32	5	40
Case 2	0.3	7.15	5.25	13	49	558

TABLE IV

ANALYSIS OF THE GENERATION OF THE HYPOTHESES

	Max_k_i	Sum_k_i	$\bar{k}$	Max_x_i	Sum_x_i	$\bar{x}$
Case 1	1	40	1	0	0	0
Case 2	8	173	6.92	16	385	15.4

Fig. 2 (a) shows an example of this problem. Ten robots with limited communications must be uniformly distributed on a circle. The least squares solution to the initial positions (black dotted circle) would make the robots collide with each other. The red-dashed circumference is the most voted hypothesis by the robots, the green circles represent the robots that vote the hypothesis whereas the black circles are the ones that do not vote. A proportional controller has been implemented that makes each robot moving towards a position in its current most voted hypothesis. The controller considers the position of the robot and also the positions of its neighbors in order to distribute the team uniformly in the circumference. The final positions of the robots are shown in Fig. 2 (b). The continuous lines show the trajectories followed by each robot. Instead of waiting until the whole voting process is complete, at each step, the robots move towards the current most voted hypothesis. This motion generates changes in the topology. Fig. 2 (c) shows the evolution of the number of votes (bottom) and of γ (top) for the most voted hypothesis. The number of votes converges in finite time whereas γ does it asymptotically. Let us notice that the final value is equal to 7, the same number of robots that supports the hypothesis.

V. CONCLUSIONS

We have presented a decentralized method, *De-RANSAC*, for consensus problems when the initial set of information contains some outliers. The proposed method is based on a new distributed approach of the RANSAC algorithm. Both the hypotheses generation and the voting process are completely distributed. The method ensures convergence to the most voted hypothesis in finite time. The number of generated hypotheses has been proved to be at least the same as in the centralized case and the additional problem of repeated combinations has also been taken into account. With respect to the voting process we have presented a distributed method based on averaging, valid for switching topologies and that is proved to reach the final solution in finite time. The simulations show the reliability of the method in different distributed applications.

REFERENCES

[1] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. June 2008. Manuscript preprint. Electronically available at http://coordinationbook.info.

[2] O. Chum and J. Matas. Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1–11, 2008.

[3] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point RANSAC for EKF-based structure from motion. In *International Conference on Intelligent Robots and Systems*, pages 3498–3504, St. Louis, 2009.

[4] P. Erdos and A. Renyi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[5] 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*, pages 381–395, 1981.

[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.

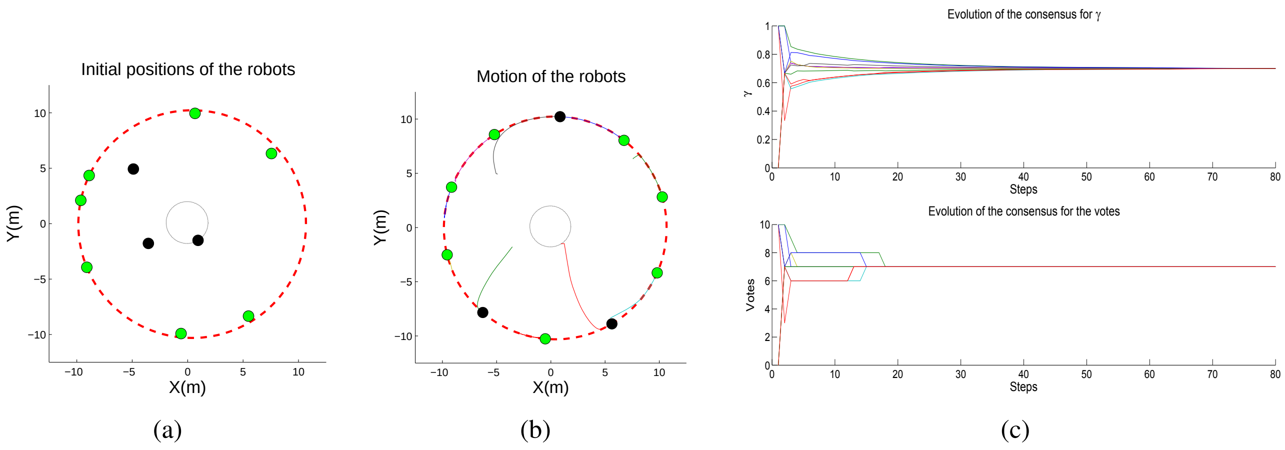

Fig. 2. *De-RANSAC* applied to robust formations. A set of 10 robots must move to form a circle. The initial positions of the robots are shown in (a). The final positions and the motion of the robots are depicted in (b). The red dashed circle is the one with most votes. The small circle is the solution using least squares with all the positions, which is an impossible configuration to fit the 10 robots. Green circles represent the robots that initially supported the hypothesis whereas black ones are the ones that did not support it. The evolution of $\mathbf{v}$ and $\boldsymbol{\gamma}$ for the most voted hypothesis is in (c). The values of $\mathbf{v}$ converge in finite time whereas the convergence of $\boldsymbol{\gamma}$ is asymptotic. Note that the final value of $\mathbf{v}$ is 7, the number of robots that voted the hypothesis.

[7] X. Li, Y. Liu, Y. Wang, and D. Yan. Computing homography with ransac. In *Proceedings of the SPIE*, pages 109–112, 2005.

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

[9] D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics*. New York: W. H. Freeman, 3 edition, 1999.

[10] D. Nister. Preemtive ransac for live structure and motion estimation. *Machine Vision and Application*, 16(5):321–329, 2005.

[11] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Classics in Applied Mathematics. 2000. ISBN 0-89871-461-3.

[12] F. Pasqualetti, A. Bicchi, and F. Bullo. On the security of linear distributed iterations. In *48th IEEE Conference on Decision and Control*, pages 4894–4901, 2009.

[13] S. Patterson, B. Bamieh, and A. El Abbadi. Distributed average consensus with stochastic communication failures. In *46th IEEE Conference on Decision and Control*, pages 4215–4220, 2007.

[14] 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.

[15] 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.

[16] L. Xiao, S. Boyd, and S. J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, pages 33–46, 2007.

[17] M. Xu and M. Petrou. Distributed RANSAC for 3D Reconstruction. In *Proceedings of the SPIE*, pages 1–9, 2008.