Distributed Target Tracking for Sensor Networks with Only Local Communication

Antonio Petitti, Donato Di Paola, Alessandro Rizzo, Grazia Cicirelli

Abstract—In this paper the problem of distributed target tracking is considered. A network of agents is used to observe a mobile target and, at each iteration, all the agents agree about the estimate of the target position, despite the fact that they only have local interactions and only a small percentage of them can sense the target. The proposed approach, named Consensus-based Distributed Target Tracking (CDTT), is a fully distributed iterative tracking algorithm. At each iteration our method applies two phases. During the perception phase the target position is obtained either as a measure or as a prediction; subsequently, in the consensus phase a consensus algorithm is applied in order to let all the agents agree on the target position. As a result, the estimated trajectories are identical for all the agents. Numerical simulations are carried out to show the effectiveness and feasibility of our approach.

I. INTRODUCTION

In the last decade, the availability of powerful, low cost, (mostly wireless) sensor networks, has fostered research in the field of distributed estimation. In this paper we focus our attention on the distributed target tracking problem: a pool of networked agents try to track a mobile target which moves on a given field. Each agent, if possible, performs an estimate of the target position; then a global estimate is carried out through computing and communicating among the nodes of the network. All the sensors have a limited sensing range, and consequently, at a given time, the target is sensed by a subset of the sensors in the network. Even though a local connection scheme is assumed among agents, a lot of target tracking algorithms rely on some form of centralization. Some algorithms make use of a data fusion center [1], some others on local filtering associated with all-to-all communication schemes, which can introduce an heavy communication overhead [2], [3], [4]. An important step towards the realization of distributed approaches has been made in [5], [6]. Even though these algorithms join a good performance with an acceptable communication overload, they keep some aspects of centralized computation.

In this paper we present a target tracking strategy which is fully distributed. The sensors have limited communication and sensing ranges, and only a fraction of them can sense the target at a given time. Each iteration of the algorithm consists of two phases. In the first phase, each network node measures the position of the target, if possible, or, alternatively, try to predict the target position by means of a suitable motion model. In the second phase, the nodes estimates converge on a common value by means of a suitable consensus strategy. It is well known that consensus-based techniques [7] are used to solve different cooperative problems such as formation control [8], flocking [9], rendezvous [10], attitude alignment [11] and sensor networks applications [12]. In our application, we suitably apply the discrete-time consensus algorithm such that the consensus value (i.e., the global estimate of the target status) is mostly affected by those nodes which hold the most significant information about the target position. To achieve this, a perception confidence value quantifies the quality of the information possessed by a single node.

II. BACKGROUND

The communication structure of a sensor network can be represented by a graph. A graph \( G \) is a pair \((\mathcal{I}, \mathcal{E})\), where \( \mathcal{I} = \{1, \ldots, n\} \) is a finite nonempty set of nodes and \( \mathcal{E} \subseteq \mathcal{I} \times \mathcal{I} \) is a set of ordered pairs of nodes, called edges. An edge \((i, j) \in \mathcal{E}\) denotes that node \( j \) can obtain information from node \( i \), but not necessarily vice versa. If the pairs of nodes are ordered the graph is said directed, otherwise if node \( i \) and \( j \) can always obtain information from each other, that is pairs of nodes are unordered, the graph is said undirected. Let \( G = (\mathcal{I}, \mathcal{E}) \) be a non-weighted undirected graph with \( n \) nodes, with node set \( \mathcal{I} = \{1, \ldots, n\} \), edge set \( \mathcal{E} \subseteq \mathcal{I} \times \mathcal{I} \), and \( n = |\mathcal{I}| \). Each node is a discrete-time dynamical system with state variable \( x_i(t) \in \mathbb{R} \), with \( i \in \mathcal{I} \). The set of neighbors of node \( i \) is denoted by \( \mathcal{N}_i = \{j \in \mathcal{I} : (j, i) \in \mathcal{E}\} \).

For fixed or switching topology and zero communication time delay, the discrete-time consensus protocol is given by:

\[
x_i(k+1) = \sum_{j \in \mathcal{N}_i \cup \{i\}} \beta_{ij}(k)x_j(k),
\]

where \( \sum_{j \in \mathcal{N}_i \cup \{i\}} \beta_{ij}(k) = 1, \forall k \), and \( \beta_{ij}(k) > 0, \forall j \in \mathcal{N}_i \cup \{i\}, \forall k \). In matrix form, (1) becomes:

\[
x(k + 1) = P(k)x(k).
\]

From the constraints on \( \beta_{ij} \) results that \( P(k) \) in (2) is a row stochastic matrix [13], which has 1 as an eigenvalue with an associated eigenvector \( \mathbf{1} \). For the discrete-time consensus algorithm (2), Gershgorin’s Disc Theorem implies that all eigenvalues of \( P \) are either in the open unit disk or at 1. As shown in [14], if 1 is a simple eigenvalue of \( P \), then \( P^k \to \mathbf{1}\nu^T \), as \( k \to \infty \), where \( \nu \) is an \( n \times 1 \) nonnegative
left eigenvector of $P$ associated with the eigenvalue $1$ and satisfies $\nu^T \mathbf{1} = 1$. As a result, $x(k) = P^k x(0) \to \nu^T x(0)$, as $k \to \infty$, which implies that, for all $i$, $x_i(k) \to \nu^T x(0)$, as $k \to \infty$, and thus \[ |x_i(k) - x_j(k)| \to 0, \text{ as } k \to \infty. \]

It is also easy to verify [7], thanks to the well-known Perron-Frobenius Theorem, that, under a time-invariant communication topology, system (2) achieves consensus if and only if either the direct communication topology has a directed spanning tree or the undirected communication topology is connected. Although the convergence is asymptotic, many studies have been carried out to assess the convergence rate and time of the consensus algorithms [15], [16].

III. Problem Formulation

In this paper we deal with the distributed tracking of a maneuvering target, performed by a sensor network in which each node has limited sensing and communication ranges, that is to say, only a subset of the nodes in the network can sense the target, and that each node is directly connected to a limited fraction of the nodes in the network. The aim of distributed tracking is to estimate and track the state of a target by using a distributed algorithm involving message-passing between a node and all of its neighbors over a network. The algorithm that we present in this paper is fully distributed, that is to say, only communication between a node and its direct neighbors is allowed, and there is no centralization of the computation in any phase of the algorithm (i.e., the algorithm does not require a phase of sensor fusion).

We consider a sensor network composed by $n$ nodes linked in a connection scheme described by an undirected graph $G = \{\mathcal{I}, \mathcal{E}\}$, which has the aim to track a target moving in the environment $E = [L/2, L/2] \times [-L/2, L/2] \subset \mathbb{R}^2$, where $L > 0$, that is a square field with side length $L^1$. In the following, we will refer to the node of the sensor network with the term agent, to emphasize the fact that each node of the sensor network must have sensing, computation and communication capabilities. We assume a one-hop communication among agents, that is each agent can send and receive messages only with its direct neighbors.

The target moves according to a piece-wise linear trajectory which can be described by a linear switching system in [6]. In practice, it moves on a blend of random linear trajectories which can be described by a linear switching system or a network. The algorithm that we present in this paper is that of one-hop communication among agents, that is each agent can perform a computation and communication task, and thus each agent can move on a blend of random linear trajectories which can be described by a linear switching system one-hop communication among agents, that is each agent can perform a computation and communication task, and thus each agent can move on a blend of random linear trajectories which can be described by a linear switching system
directed spanning tree or the undirected communication topology is connected. Although the convergence is asymptotic, many studies have been carried out to assess the convergence rate and time of the consensus algorithms [15], [16].

The target moves according to a piece-wise linear trajectory which can be described by a linear switching system in [6]. In practice, it moves on a blend of random linear trajectories which can be described by a linear switching system or a network. The algorithm that we present in this paper is that of one-hop communication among agents, that is each agent can perform a computation and communication task, and thus each agent can move on a blend of random linear trajectories which can be described by a linear switching system one-hop communication among agents, that is each agent can perform a computation and communication task, and thus each agent can move on a blend of random linear trajectories which can be described by a linear switching system
directed spanning tree or the undirected communication topology is connected. Although the convergence is asymptotic, many studies have been carried out to assess the convergence rate and time of the consensus algorithms [15], [16].

In this paper we deal with the distributed tracking of a maneuvering target, performed by a sensor network in which each node has limited sensing and communication ranges, that is to say, only a subset of the nodes in the network can sense the target, and that each node is directly connected to a limited fraction of the nodes in the network. The aim of distributed tracking is to estimate and track the state of a target by using a distributed algorithm involving message-passing between a node and all of its neighbors over a network. The algorithm that we present in this paper is fully distributed, that is to say, only communication between a node and its direct neighbors is allowed, and there is no centralization of the computation in any phase of the algorithm (i.e., the algorithm does not require a phase of sensor fusion).

We consider a sensor network composed by $n$ nodes linked in a connection scheme described by an undirected graph $G = \{\mathcal{I}, \mathcal{E}\}$, which has the aim to track a target moving in the environment $E = [L/2, L/2] \times [-L/2, L/2] \subset \mathbb{R}^2$, where $L > 0$, that is a square field with side length $L^1$. In the following, we will refer to the node of the sensor network with the term agent, to emphasize the fact that each node of the sensor network must have sensing, computation and communication capabilities. We assume a one-hop communication among agents, that is each agent can send and receive messages only with its direct neighbors.

The target moves according to a piece-wise linear trajectory which can be described by a linear switching system in [6]. In practice, it moves on a blend of random linear trajectories, and when it reaches the boundary of the region, it is pushed back by a force which is orthogonal to the region boundary. We will refer to the discrete-time target trajectory as $\xi(k)$, where $\xi(k) \in E$ and $k$ is the discrete time.

An agent $i \in \mathcal{I}$ is described by the tuple:

\[ \langle \mathbf{p}_i, \xi_i(k), \mathbf{M}_i, \gamma_i(k) \rangle, \]

where the vector $\mathbf{p}_i \in E$ denotes the position of the agent in the environment; the vector $\xi_i(k) \in E$ is the estimated position of the target at time $k$; the entity $\mathbf{M}_i$ is the agent’s memory, which contains all the global estimates of the target state over time. The size of $\mathbf{M}_i$ is equal to the number of time instants during which the target is tracked, $k_f$. In the following we will indicate with $\mathbf{M}_i(k)$ the global estimate of the target state performed by the network at time $k$. Note that the difference between $\xi_i(k)$ and $\mathbf{M}_i(k)$ is that the former is the target state estimation performed individually by agent $i$ at time $k$, whereas the latter is the estimate of the target state performed globally over the sensor network and stored in agent $i$ at time $k$. The value $\gamma_i(k) \in \mathbb{N}_0$ is called perception confidence value, and quantifies the quality of the information about the estimate of the target position $\xi_i(k)$. Its working principle will be clarified later in the paper.

Furthermore, let us denote the sensing and communication range of each agent with $r_s$ and $r_c$ respectively. Each agent senses the target at time $k$ if and only if $||\mathbf{p}_i - \xi_i(k)|| \leq r_s$. Two agents, $i$ and $j \in \mathcal{I}$, can communicate, i.e. a link between them exists, if and only if $||\mathbf{p}_i - \mathbf{p}_j|| \leq r_c$.

Finally, among the many possible aspects of the performance assessment in the target tracking framework [17], in this work we will focus our attention on the tracking accuracy, by evaluating the mean square error between estimated and actual target trajectory. A further interesting aspect that we investigate on is the detriment of the tracking accuracy with respect both to the sensing coverage and to the average percentage of sensing nodes in the network.

IV. Consensus-based Distributed Target Tracking

The proposed CDTT algorithm is an iterative strategy that, at each iteration, consists of two main phases: perception phase and consensus phase. During the perception phase each agent either senses or predicts the target position, depending on the fact that the target is within its sensing range or not. We will use the term sensing agent for those nodes for which the target lays in their sensing range; whereas we will use the term predicting agent for the remainder of the nodes.

In the second phase, a consensus algorithm is applied such that all the agents converge to a common value for the target position. The CDTT algorithm, running at each iteration the two phases, guarantees the agreement of the whole network on the target position over time, exploiting the convergence property of the distributed consensus algorithm, making our approach fully distributed. Our algorithm requires the synchronization of node clocks to rely on a common discrete timeline. This need does not invalidate the distributed nature of our approach in real applications, because the sensor network itself can be effectively exploited to achieve the global synchronization of agent’s clocks [18]. Let us now describe in detail the two phases of each iteration. We describe the computation of iteration at time $k + 1$ assuming, therefore, that each agent has available the information in the tuple (3) up to time $k$.

A. Phase 1: Perception Phase

In this phase each agent estimates the target position. How this information is obtained depends on the role of the agent $i$, i.e. whether it is a sensing or a predicting agent. Algorithm 1 describes the perception phase for agent $i$ at
where the uncertainty which depends on their relative distance is measured by the sensing system. We assume that each agent individual estimate will be in practice ineffective for computing the global estimate.

Algorithm 1 CDTT: Phase 1 for agent \( i \) at iteration \( k + 1 \)

1. if \( d(k + 1) \leq r_s \) then
2. \( \xi_i(k + 1) = z_i(k + 1) \)
3. \( \gamma_i(k + 1) = 0 \)
4. \( \nu = 0 \)
5. else
6. if \( \nu = 0 \) then
7. \( \tilde{x}(0) = [M(k)^T \tilde{v}(0)^T]^T \)
8. end if
9. \( \tilde{x}(\nu + 1) = A \tilde{x}(\nu) \)
10. \( \xi_i(k + 1) = \xi(\nu + 1) \)
11. if \( \gamma_i(k + 1) > \gamma^* \) then
12. \( \gamma_i(k + 1) = \gamma^* \)
13. end if
14. \( \nu = \nu + 1 \)
15. end if
16. end if

iteration \( k + 1 \). Notice that one iteration consists of a single run of both phases, perception and consensus.

If the agent \( i \) is a sensing agent, the position of the target is measured by the sensing system. We assume that each sensing agent measures the position of the target with an uncertainty which depends on their relative distance \( d_i(k + 1) = ||p_i - \xi(k + 1)|| \). Let us denote with \( z_i(k + 1) \) the position of the target measured by the agent \( i \) at time \( k + 1 \) as follows:

\[
z_i(k + 1) = Hx(k + 1) + \eta_i(k + 1),
\]

where \( x(k + 1) = [\xi^{(1)}(k + 1) \xi^{(2)}(k + 1) v^{(1)}(k + 1) v^{(2)}(k + 1)]^T \) is the target state vector containing the actual position and velocity in the environment at time \( k + 1 \), \( H \) is the output matrix

\[
H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},
\]

and \( \eta_i(k + 1) \) is a stochastic component, which is defined as

\[
\eta_i(k + 1) = N_0(0, \sigma^2_{\eta_i}(k + 1)),
\]

where \( N_0 \) is a zero-mean white Gaussian noise with variance \( \sigma^2_{\eta_i}(k + 1) \) obtained by the relation

\[
\sigma^2_{\eta_i}(k + 1) = \frac{2d_i(k + 1)}{r_s}.
\]

Thus, the estimated position of the target at time \( k + 1 \) made by a sensing agent \( i \), is \( \xi_i(k + 1) = z_i(k + 1) \). The quality of the estimated position could be improved by any algorithm which make use of past measurements and the motion model, as for example, a Kalman Filter. Obviously, such a choice goes in the direction of increasing the computational complexity for each agent, therefore a trade off between these two issues must be set.

If the target is not within the sensing range of the \( i \)-th node, that is \( i \) is a predicting agent, the target motion is estimated through a simple linear target motion model, widely used in literature to cope with this kinds of problem [6]. The target motion is represented by the linear process

\[
x(\nu + 1) = A x(\nu),
\]

where \( A = \begin{bmatrix} 1 & \epsilon \\ 0 & 1 \end{bmatrix} \otimes I_2 \), where \( A \) is the target state transition matrix in \( \nu \)-discrete steps, \( I_2 \) is the 2 \( \times 2 \) identity matrix, and the symbol \( \otimes \) denotes the Kronecker product of matrices. Thus, the estimated position of the target at time \( k + 1 \) made by a predicting agent \( i \), is \( \xi_i(k + 1) = \xi_i(\nu + 1) \).

Phase 1 ends up with each agent possessing an individual estimate of the target position, \( \xi_i(k + 1) \), obtained either by measurement or by prediction. Phase 2 will make all agents agree on a common estimate of the target position, via a consensus-based algorithm. The perception confidence value \( \gamma_i(k + 1) \) is conveniently set during Phase 1 for each agent, in order to quantify the reliability of the estimate of each agent, which will determine the influence of the single agent estimate on the final outcome of Phase 2. In particular, all sensing agents have their perception confidence value set to \( \gamma_i(k + 1) = 0 \). Each predicting agent, on the other hand, sets its \( \gamma_i(k + 1) \) to the number of consecutive iterations during which it performs a prediction (instead of a measurement), i.e., \( \gamma_i(k + 1) = \nu + 1 \), up to a threshold \( \gamma^* \), for which the agent individual estimate will be in practice ineffective for computing the global estimate.

B. Phase 2: Consensus Phase

The second phase of CDTT is the consensus phase. Here agents make use of a discrete-time consensus protocol to agree on a global estimate of the target position. This allows trajectory coherence, at each iteration, over all agents in the network.

This phase can be further divided into two steps. In the first step each agent communicates with its neighbors in order to share information useful to set up suitable weights for the consensus algorithm, while in the second step a round of consensus is run.

In the communication step, agent \( i \) receives a message from each of its neighbors, that is \( \gamma_i = [\xi_i(k + 1), \gamma_j(k + 1)] \) from \( j \in N_i \), which contains the estimate of the target
the readability of further formulas.

Fig. 1. Evolution of CDTT: A simulation of a network of 9 agents and $\epsilon = 20\%$ is shown. For each iteration the two phases are shown. Moreover, time windows concerning the Phase 2 (consensus phase) are highlighted on the plot of coordinates of target position individual estimates $\zeta_i^{[1]}$ and $\zeta_i^{[2]}$.

position and the perception confidence value of the neighbor $j$. Using this information each agent can determine the weights to be used in the next step. Let us define $w^\ast_{ij}(k+1)$ as:

$$w^\ast_{ij}(k+1) = \begin{cases} \beta n^{(k+1)} & \text{if } j \in N_i \\ 0 & \text{otherwise} \end{cases},$$

where $\beta < 1$ is a constant factor for all agents in the network. The effect of the value $w^\ast_{ij}$ on the consensus protocol is that the more outdated is the value of the prediction of the target position, the less significant is the information coming from the agent $j$ for the computation of the global estimate. Indeed, the weights that will be used in the subsequent consensus step are the normalized perception weights, defined as:

$$w_{ij}(k+1) = \frac{w^\ast_{ij}(k+1)}{\sum_{j=1}^{n} w^\ast_{ij}(k+1)},$$

so that the matrix $W = [w_{ij}]$, $W \in \mathbb{R}^{n \times n}$ is row-stochastic.3

After the weight initialization step, the consensus process starts. Each agent sets its initial value for starting the consensus procedure from its current individual estimate of the target position, that is to say, $\zeta_i(0) = \hat{\xi}_i(k+1)$. Thus the consensus is performed by the following local rule

$$\zeta_i(\kappa + 1) = \sum_{j=1}^{n} w_{ij} \zeta_j(\kappa),$$

where $\kappa$ is a discrete time index for which the associated time step must be smaller enough than that associated with index $k$, as it will be clarified later. Equation (9) can be written in matrix form as

$$\Psi(\kappa + 1) = \Omega \Psi(\kappa),$$

where $\Psi \in \mathbb{R}^{2n}$,

$$\Psi(\kappa) = [\zeta_1^{[1]}(\kappa), \zeta_2^{[1]}(\kappa), \ldots, \zeta_n^{[1]}(\kappa), \zeta_1^{[2]}(\kappa), \zeta_2^{[2]}(\kappa), \ldots, \zeta_n^{[2]}(\kappa)],$$

and $\Omega \in \mathbb{R}^{2n \times 2n}$ is given by

$$\Omega = I_2 \otimes W = \begin{bmatrix} W & 0_{n,n} \\ 0_{n,n} & W \end{bmatrix},$$

where $0_{n,n}$ is the $n \times n$ null matrix.

It straightforward to verify that, as $W$ is row-stochastic, also $\Omega$ is row-stochastic and, as recalled in Section II, the associated consensus protocol (10) converges asymptotically. From a practical point of view, we can assume that consensus is achieved in a finite number of steps $\kappa_c$. At the convergence of the consensus protocol, that is at $\kappa = \kappa_c$, each agent stores the final common value of the consensus phase $\bar{\zeta}_i$ in its own memory, that is, for the $i$-th agent, $\hat{M}_i(k+1) = \bar{\zeta}_i$. We remark that this value will be more affected by the sensing agents, and less affected by the predicting agents in a way that the greater the perception confidence value of a given predicting agent is, the lesser will be the influence of that agent on the final consensus value. This is due to the weight choice strategy defined in (7-8).

C. Convergence Issues

In the case under analysis, it is possible to bound the convergence time $\kappa_c$. By construction, in fact, $W$ is a particular type of row-stochastic matrix representative of an equal-neighbor, time-invariant, bidirectional model on a connected graph with $N$ nodes, for which the convergence time is $T_c(\varepsilon_T) = O(n^3 \log(n/\varepsilon_T))$, where $\varepsilon_T$ is the tolerance for which consensus can be considered achieved [16], [19].

It is easy to prove that our algorithm converges under two hypotheses: 1) the sensor network is connected; 2) denoting with $\varepsilon$ the iteration time of the consensus protocol (10) and $\varepsilon$ that of the main CDTT algorithm, the latter must be greater than $t_p$, the time needed by the agents to perform the perception phase, plus $\kappa_c \varepsilon$, the time needed for the convergence of the consensus algorithm. That is to say, $\varepsilon > t_p + \kappa_c \varepsilon$.

Fig. 1 shows three complete iterations of the CDTT algorithm, where $\varepsilon$ has been fixed such that $\varepsilon = 20\%$, so that to satisfy hypothesis 2). The two pictures in Fig. 1 show the evolution of $\bar{\zeta}_i^{[1]}$ and $\bar{\zeta}_i^{[2]}$, i.e., the evolution of the individual estimates of the target position made by each single node over time.

V. Numerical Results

The performances of the CDTT are analyzed by running a number of nontrivial simulations in a generic scenario. We simulate a sensor network tracking the position of a maneuvering target, with a motion model as described in Section III, moving inside a square field $E$ with side length $L = 90$. For each agent we set the communication radius as $r_c = (3\sqrt{n}) + 2$. The sensing radius $r_s$ varies in order to obtain a specific value of the coverage sensing area in the environment $E$, as detailed in the following. In Phase

3we omit in the following the time dependence of matrix $W$ to increase the readability of further formulas.

4the superscript into brackets indicates the component of vector $\zeta_i(\kappa)$.
For each agent, we set the initial value of the estimated target position \( \hat{\xi}_t(0) = [0, 0]^T \). Moreover, the variance of the random process used to generate the initial target velocity \( \hat{v}(0) \) in the target motion prediction model is initialized using \( \sigma^2_0 = 2 \). By trial and error, we fixed the tracking algorithm parameters as \( \gamma^* = 10 \), \( \beta = 0.9 \).

In order to evaluate global performance of the CDTT, we define two additional parameters. The first, \( \rho \), represents the ratio between the coverage sensing area and the total area of the field \( E \). It is straightforward to verify that, given a fixed network topology and spatial node distribution, \( \rho \) is a function of the sensing radius \( r_s \). The second, \( \varphi \), represents the average percentage of sensing agents during a single run. As a metric of target tracking accuracy, the following mean square error (in norm) is computed:

\[
\alpha = \frac{1}{k_f} \sum_{k=1}^{k_f} ||M(k) - \xi(k)||^2, \quad (12)
\]

where \( k_f \) is the length (in time samples) of the target trajectory, \( M(k) \) and \( \xi(k) \) are the estimated and actual target positions, respectively, at time \( k \), and \( || \cdot || \) is the Euclidean norm in \( \mathbb{R}^2 \).

We now evaluate the performance of the CDTT algorithm in different setups of the simulation scenario. Table I lists the results obtained in different simulations, varying \( \rho \) and \( n \). As expected, the performance better as \( \rho \) and \( n \) increase. Two cases of the aforementioned simulation campaign are depicted in Fig. 2. In particular, two trajectories estimated by CDTT for a network of 25 agents are considered for two different values of the coverage percentage. As can be seen, performance is better with a higher coverage percentage.

A. Comparison with the KCF algorithm

In order to evaluate the tracking performance of our CDTT algorithm we compare it to the hybrid architecture of the Kalman Consensus Filter with message passing (KCF) as described in [6]. This method is based on distributed microfilters with message-passing between agents and the use of a high-level fusion center for aggregating the estimates of a suitably selected set of agents, specifically, those who show the best agreement in the trajectory estimate (this is done by evaluating the covariance matrix of the local estimates). As detailed in [6], we set the following KCF parameters: \( k_u = 3 \), \( k_w = 5 \), \( R_i = k_w^2 I_2 \), \( Q = k_w^2 I_4 \), \( P_0 = 10 k_w^2 I_4 \).

In order to compare the tracking performance of CDTT and KCF, 20 random test trajectories have been generated for different values of \( \rho \). More specifically, the average value of \( \alpha \) over the test trajectories has been computed for each value of \( \rho \) and compared with the correspondent performance parameter obtained through the KCF algorithm trajectories.

The results of this comparison are depicted in Fig. 3. For lower values of \( \rho \) the KCF algorithm outperforms our CDTT. This is due to the fact that the centralized data fusion strategy can sensibly select the best estimates and get rid of the worst ones. In fact, in the case of a small number of sensing agents, bad estimates worsen the performance of a totally distributed approach. On the other hand, for values of \( \rho \) greater than 62, CDTT outperforms KCF, with the advantage of a complete distribution of the computation. Finally, for values of \( \rho \) close to 100\%, the performance of the two algorithms are comparable. This is due to the fact that, concerning CDTT algorithm, at each time instant many agents are sensing ones, and the global estimate is strongly pushed by the consensus.
algorithm towards reliable estimates. On the other hand, KCF performs very well because it can operate a sensor fusion by relying on very accurate measurements made by several agents, and therefore nearly no bad estimates are included in the computation. At this stage, the centralization or the distribution of the computation become irrelevant.

VI. Conclusion

In this paper we have addressed the problem of distributed target tracking by a network of sensory agents. The agents have only local interactions and only a small percentage of them can sense the target. Each agent estimates the target state either by a direct measurement, if the target is in its sensing range, or by a prediction, according to a linear target motion model, if not. Then, at each iteration, a consensus algorithm is applied to reach the agreement about the target state. We have proved the convergence of the proposed algorithm towards a unique estimate of the target trajectory. The novel aspect of our approach is the full distribution of the computation become irrelevant.

VI. C O N C L U S I O N

In this paper we have addressed the problem of distributed target tracking by a network of sensory agents. The agents have only local interactions and only a small percentage of them can sense the target. Each agent estimates the target state either by a direct measurement, if the target is in its sensing range, or by a prediction, according to a linear target motion model, if not. Then, at each iteration, a consensus algorithm is applied to reach the agreement about the target state. We have proved the convergence of the proposed approach to a unique estimate of the target trajectory. The novel aspect of our approach is the full distribution of the algorithm, with respect to a partially distributed method with which we provided some numerical comparisons. Moreover, the performance is good despite the fact that a few agents can actually sense the target at each iteration, and the environment is not fully covered by the sensor range.

Further developments of this work will deal with the inclusion of mobile agents to exploit mobility in order to improve the tracking performance (consequently, a time-varying topology of the network will be considered) and the possibility of having heterogeneous sensing agents, possibly with more limited sensing capabilities.

ACKNOWLEDGEMENTS

This work is partially supported by “APQ Regione Puglia Distretto della Meccatronica” MEDIS DM01.

REFERENCES


TABLE I

<table>
<thead>
<tr>
<th>n</th>
<th>ρ</th>
<th>φ</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>50%</td>
<td>2.43%</td>
<td>177.894</td>
</tr>
<tr>
<td></td>
<td>75%</td>
<td>4.77%</td>
<td>15.053</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>71.30%</td>
<td>0.078</td>
</tr>
<tr>
<td>50</td>
<td>50%</td>
<td>1.24%</td>
<td>99.248</td>
</tr>
<tr>
<td></td>
<td>75%</td>
<td>1.97%</td>
<td>14.117</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>35.10%</td>
<td>0.073</td>
</tr>
<tr>
<td>75</td>
<td>50%</td>
<td>0.78%</td>
<td>67.322</td>
</tr>
<tr>
<td></td>
<td>75%</td>
<td>1.40%</td>
<td>12.228</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>28.84%</td>
<td>0.062</td>
</tr>
</tbody>
</table>

Fig. 3. Comparison between CDTT and KCF: Values of α as function of the parameter ρ (n = 25).