Moving Target Detection and Tracking in Wireless Sensor Networks

Stefano Di Cairano, Alberto Bemporad, and Angelita Caldelli

Abstract— **Wireless sensor networks (WSNs) are receiving an increasing interest because of their ease of deployment, autonomous re-configuration capabilities, and small cost. They can be employed in a large variety of applications, from military and surveillance to environmental monitoring. Since on board batteries only provide a limited energy resource to nodes and often they cannot be replaced, when designing WSN-based applications a fundamental issue is the reduction of power consuming operations. In this paper we exploit a WSN for detecting and tracking moving targets. Such a network would waste energy if all nodes were kept constantly active, since at every time instant the target is in the range of only a subset of them. For this reason an autonomous target detection algorithm and a cooperative selective-activation tracking algorithm are proposed, and the use of Kalman filtering techniques for motion estimation is discussed. In particular, the problem of fusing the data obtained from different sensors is considered. Simulations results in different scenarios are presented.**

I. Introduction

A wireless sensor network (WSN) [1] is composed by a possibly large set of small electronic devices equipped with different sensors, communication capabilities and with limited data processing system [2], [3]. The nodes take measurements from different sensing devices, perform simple processing operations and share the obtained information through wireless links. WSNs can be employed in a large variety of applications [1], since the nodes are cheap and installation costs are very small, they are able to automatically re-configure when nodes die or are added, and they can be deployed even in human-unaccessible locations. Small node size and long battery life are colliding objectives. Typically, once exhausted the battery cannot be replaced, either because it is physically impossible, or just because it is not economically convenient. For this reason it is necessary to adopt ad-hoc strategies to exploit energy resources in the most efficient way, in order to increase the WSN life and reduce costs.

Past studies [4] have shown that communication is the most expensive operation for a WSNs in terms of power consumption. Hence, the life-cycle of a WSN, is maximized by minimizing the communications between sensor nodes, for instance by adopting data fusion techniques [5].

Besides power consumption, there are other aspects to consider when designing a WSN:

- *Latency*: after a node senses an interesting event, the information may take a long time before it reaches the base station. A compromise between information delays and power consumption is needed.
- *Fault tolerance*: sensor nodes can stop working because they have exhausted their energy supply or they have been damaged. The network functionality must not change significantly when a small subset of nodes stops working [6]. Fault tolerance is a measure of the capability of the WSN to maintain its functionality when a part of its nodes is damaged or not utilizable.
- *Scalability*: the number of nodes that composes a network changes accordingly to the applications and to the area over which it is deployed [7]. Algorithms designed for WSNs must be suitable for networks composed by any number of nodes.

In this paper we consider a WSN composed by sensor nodes randomly deployed over a region to detect and track objects moving across the area. A similar problem has been treated in [8], where the power consumption and the tracking performance of different node activation strategies are compared. The tracking algorithm of this paper is an implementation of the so called "selective activation" approaches described in [8].

Section II defines the WSN scenario and the target model. In Section III a target detection scheme is presented, which defines some of the network parameters, namely the number of nodes that form the WSN and the mean activation frequency of the nodes. Section IV deals with the tracking algorithm and the data fusion. Finally, in Section V simulations for different scenarios are presented.

II. Scenario

The purpose of the WSN is to detect an object that is moving inside the monitored area and, once detected, to keep track of its position and velocity. This must be accomplished with a reduced power consumption, thus keeping the number of active nodes and the number of communications small.

A. Target model

In order to exploit model-based filtering techniques, a dynamical model of the moving target is needed. We use a general model because we do not make any assumption about the target nature. In case more information on the target is available, a more detailed model can be used. We consider a discrete-time model of the object motion in the x-axis and y-axis of the Cartesian plane. We assume the target moves with an initial unknown velocity vector $v(k)$, and it is subject to random accelerations modelled by a Gaussian white noise vector $w(k)$, with zero mean and covariance matrix Q.

Work (partially) done in the framework of the HYCON Network of Excellence, contract number FP6-IST-511368

The authors are with Dipartimento di Ingegneria dell'Informazione, Università di Siena, dicairano,bemporad,caldelli@dii.unisi.it

ISBN: 978-960-89028-5-5

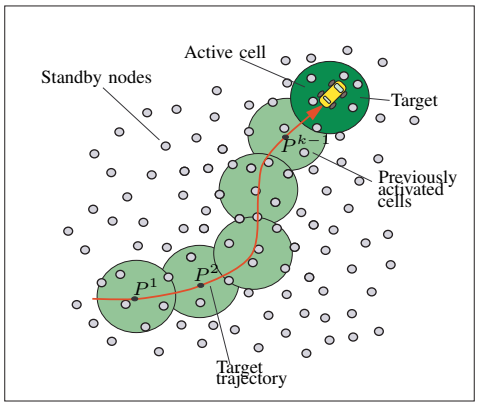


Fig. 1. Detection and tracking scenario

Let $x(k) = [p_x(k)\ v_x(k)\ p_y(k)\ v_y(k)]^T$, be the overall state vector, where $p_x(k)$, $v_x(k)$, $p_y(k)$ and $v_y(k)$ are the object position and velocity with respect to the x and to the y axis, respectively, at time k. The target dynamics are

$$x(k+1) = Ax(k) + Bw(k), \tag{1}$$

where

$$A = \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{2}T_s^2 & 0 \\ T_s & 0 \\ 0 & \frac{1}{2}T_s^2 \\ 0 & T_s \end{bmatrix}, \tag{2}$$

and where T_s is the sampling time. The acceleration components are independent, $w(k) \triangleq \mathcal{N}(0, Q)$, where $Q = \begin{bmatrix} q_x & 0 \\ 0 & q_y \end{bmatrix}$.

B. Sensor node and sensor network

We consider a physically static network composed by N nodes with a dynamic communication configuration, ensuring robustness against node damages.

It is assumed that each node i knows its position $p_i^s \in \mathbb{R}^2$ in Cartesian coordinates with respect to a common reference system. This can be obtained through a GPS on board of each node, or through the use of self-localization algorithms [9] after the network deployment, or simply by deploying each sensor in a prespecified position. Each node measures the relative position of the tracking object along the x and y axes, and the absolute position of the object is computed. The sensing range of each node is limited and equal to r_s.

Instead of keeping all the nodes constantly in a fully operating mode, we propose an approach in which the nodes are awaken by events of interest. A sensor node can be in three operating modes. *Active*, in which it can send and receive messages and it can perform communication. *Standby*, in which its computation, sensing and transmission capabilities are disabled, so that no energy is wasted by the sensing process or by outgoing communications, but it is still able to receive to be possibly awaken. *Alert*, in which sensing and reception are enabled, while transmission and computation are disabled, and which is basically an intermediate state between the previous two. The optimal situation would be that the sensor is in active mode only when the target is in its sensing range and it can take valid measurements. This is practically impossible because no sensor would be able to detect a new object on the field. Each node has a scheduled "wake-up" instant at which it automatically switches to the alert mode, searches for new targets in its range, and, if it does not find anyone, returns to the standby mode.

III. Sensor Deployment and Target Detection

With the aim of minimizing the communication between nodes, we designed a target detection scheme in which each sensor works autonomously until a target is acquired. The detection scheme is the described in Algorithm III.1.

1. Scan the detection range
2. if new target detected
   2.1. then go to 5.
3. schedule next wake-up instant and switch to standby mode
4. at the scheduled instant, switch to alert mode then go to 1.
5. stop detection and start the tracking algorithm

Algorithm III.1: Detection algorithm

The only choice we have to perform in Algorithm III.1 is the scheduling policy of "wake-up" instants. Since we want to minimize communications, we do not allow any information exchange between nodes before the target is discovered. In addition, we do not have any information about the target, neither about its presence nor about its position, until we have detected it. Thus, we use a stochastic scheduling approach: The next wake-up instant is defined by summing an exponentially distributed random variable with distribution parameter λ_W to the current time. Note that λ_W is the mean frequency of wake-up events and λ_W^{-1} is the mean standby period. This approach, given a certain monitored area, allows us to design the parameter λ_W and the number of sensor deployed N so that an object which moves in the monitored area for a period larger than T is detected with a certain probability.

The following assumptions simplify computations.

Assumption 1: Until detected, the object can be at any location in the monitored area with equal probability.

Assumption 2: The sensors are randomly deployed on the monitored area with uniform probability distribution.

Assumption 3: The detection period is infinitesimal.

Assumption 1 states that no a-priori information is available on the object position until it has been detected. Assumption 2 states that the probability of the object to be in the detection range of a node is uniform on the monitored area, and thus independent from its position. Assumption 3 decouples the detecting action of different sensors, because it states that the probability that two nodes are scanning at the same instant is zero, and it holds if the detection period is much smaller than the mean standby period.

Consider a single sensor node. The "wake-up" (W) event instants are exponentially distributed with (mean) frequency

λ_W. However, the instants at which the sensor wakes up and finds a new target are only a subset of those. In particular, the probability that an active sensor detects an object crossing the monitored area is equal to the probability of the object being in the detection range. Thus, by Assumption 1, the probability is $p_{d|a} = \frac{a_s}{A_m}$, where A_m is the total monitored area and a_s is the detection area of a node. Since there is no relation between the wake-up events and the object position, the "wake-up and detected" (WD) event instants of a single node are exponentially distributed events with mean frequency $\lambda_{WD} = \lambda_W\, p_{d|a}$.

Because of Assumption 3, the "network's wake-up and detected" (WDN) event instants are exponentially distributed with parameter

$$\lambda_{WDN} = \lambda_W\, p_{d|a}\, N = \lambda_W \frac{a_s}{A_m} N, \tag{3}$$

and thus the probability density function that a WDN event occurs at time ξ after the previous one (or after the algorithm initialization) is

$$pdf_{WDN}(\xi) = \lambda_{WDN} e^{-\lambda_{WDN}\xi} = \lambda_W \frac{a_s}{A_m} N e^{-\lambda_W \frac{a_s}{A_m} N \xi}. \tag{4}$$

Proposition 1: Given a sensor network composed by N sensors having sensing range a_s and mean activation frequency λ_W, an object which moves in the monitored area A_m is detected within a period T with probability $\hat{P}$ satisfying

$$\frac{A_m}{a_s T} \ln\left(\frac{1}{1-\hat{P}}\right) = \lambda_W N. \tag{5}$$

Proof: The probability that a target crossing the monitored area is detected within a time period T is equal to the probability that a WDN event occurs within T time units, that is $\mathbf{P}[\xi \le T]$. Thus,

$$\mathbf{P}[\xi \le T] = \int_0^T \lambda_W \frac{a_s}{A_m} N e^{-\lambda_W \frac{a_s}{A_m} N \xi} \mathrm{d}\xi = \left(1 - e^{-\lambda_W \frac{a_s}{A_m} N T}\right). \tag{6}$$

By taking logarithms, (5) follows. ∎

Equation (5) can be used to design the network characteristics (λ_W, N) for a given T and a given confidence interval $\hat{P}$. The parameter T is related to the object motion characteristics, for example it can be the minimum time the object takes to cross the monitored area, and to the tolerated delay of detection. The parameter $\hat{P}$ is related to the required confidence probability of the detection. Note that $\mathbf{P}[\xi \le T] \ge \hat{P}$ is satisfied if $\frac{A_m}{a_s T} \ln\left(\frac{1}{1-\hat{P}}\right) \le \lambda_W N$.

Equation (5) states a natural trade off between the number of sensors and the frequency of node activation for a desired detection performance. The first parameter will be related to the mean-life of the sensors, the second to the network cost.

We performed tests to validate this approach. A squared monitored area of 23.2×23.2 m^2 is considered, nodes having a circular detection area of 8 m^2 (i.e., $r_d = 1.6$ m), and a desired maximum detection period $T = 3.8$ s. The sensors are randomly distributed with uniform probability on the monitored area in a way that the sensor detection areas lie inside the monitored area. The network must detect an object that starts moving from a position $(\bar{x} + \eta, 0)$, where $\bar{x} = 8$ m and η is a random variable uniformly distributed in $[0, 2]$ m. The object initial velocity is zero, and the random acceleration components are independently uniformly distributed in $[0, 3]$ m/s^2 each.

We suppose the node activation mean frequency $\lambda_W = 0.5$ s^{-1} fixed, and consider three different numbers of nodes in order to satisfy three different confidence probabilities. The confidence probability $\hat{P}$, the corresponding number of nodes N computed from (5), and its discrete approximation $\tilde{N}$ are reported in Table I. We set up 200 different detection fields by changing the random sensor distribution on the monitored area and we performed 100 detection tests for each field. In Table I the experimental mean detection probabilities D over the total 20000 tests for the three cases are reported. The results are close to the confidence probability required, despite the assumptions we made.

$\hat{P}$	N	$\tilde{N}$	D
0.95	106.07	106	94.72
0.90	81.53	82	90.61
0.70	42.63	43	71.81

TABLE I

DETECTION RESULTS FOR DIFFERENT NETWORK DESIGNS.

IV. TRACKING ALGORITHM

After a new target has been discovered by a node through Algorithm III.1, the nodes must start cooperating to track the target [10]. We assume here that at most one object may be crossing the area, while noting that the proposed strategies can be extended to multitarget tracking, for instance by exploiting the results in [11], [12].

With the aim of minimizing the power consumption, we do not want to activate all the nodes for tracking, but only the ones that give significant contribution to the position estimation, that is, the ones whose sensing range contains the target. Such nodes form a sensor cell. Let $S = \{s_1, \ldots, s_N\}$ be the sensor nodes, p_i^s be the position of the i^{th} node. Given a cell center c and a cell radius r_c, a sensor cell is the set of nodes $SC(c, r_c) = \{s_i \in S : \|c - p_i^s\|_2 \le r_c\}$. Thus, a sensor cell is composed by those sensors which are at a distance smaller than r_c from the cell center c.

We propose a tracking algorithm in which sensor cells are activated for taking measurements and for estimating target position and velocity. Sensor cells are dynamically managed: a cell remains active for a certain period, then it is released and a new cell, possibly containing different nodes, is formed. The cell life, that is the number of sampling periods K the cell remains active, is fixed in the algorithm considered here, even though the same strategies can be applied to the case of a dynamically varying K. Once the cell life expires, the actual estimate of the target state is used for choosing the nodes that will possibly form the next cell.

During the cell life all the sensors in the cell measure the target position. As a consequence, it is required that the target

remains within the detection range r_s of the each sensor in the cell. Hence, the choice of the algorithm parameters r_c and K, depends on r_s and on the target motion characteristics. For instance, the larger K, the larger is the distance that the target can cover before a cell switch. Thus, the larger must be r_s and the small r_c, in order to ensure that all the node can take measurements. On the other hand, the larger r_c, the larger is the number of measurements available for estimation, while the smaller K, the more frequent are cell switches, which require additional communications.

The simpler position at which the cell center can be placed is the position of the node that detects the object. However, in this way it is impossible to ensure that the target remains in the sensing range (r_s) of each node if $r_d = r_s$. Suppose the target is detected at a distance r_d, then at next step it can be out from the sensing range, at least of the master. Thus, it is required that $r_d < r_s$. If the maximum distance $X(K)$ the target can cover in K steps ($X(K) \leq r_s$) is available with a confidence probability $\tilde{P}$, then the range r_c of a cell centered on the node that detected the target must satisfy

$$r_c + r_d + X(K) \leq r_s. \tag{7}$$

Since the distance d_0 of the target when detected is smaller than r_d and the distance between a node and the center is smaller than r_c, the initial distance of each node from the target is smaller than $r_d + r_c$. Thus, there is at least a confidence probability $\tilde{P}$ that all the sensors can take measurements during K steps. A better position to place the cell center is close to the detected target position. However, this requires additional communication and hence will not be considered here.

Because of Assumption 3 there is no conflict on the master choice, since only one node detects the target. However, in case it is needed, a simple master selection policy that resolves conflicts can be based on a fixed-priority rule, e.g. by using the node ID.

Once a node has detected a new target, the *selective activation tracking* Algorithm IV.1 is executed.

1. node sends a "wake-up" signal to the nodes within a range r_c
2. active nodes form a sensor cell, the one that has awaken the others become the master.
3. for $i = 1 : K$
   3.1. active nodes collect measurements and send them to the master.
   3.2. master updates the estimation by using the collected measurements.
   3.3. if target exited from the monitored area or it has been lost
      3.3.1. master release the target, then go to 7.
   end
4. active nodes switch to standby mode.
5. master alerts all the nodes within a fixed distance from the final estimation of target position, then switches to standby mode.
6. one of the alerted nodes detects the target, continue from 1.
7. master broadcasts the order to stop tracking and to activate detection.

Algorithm IV.1: Tracking Algorithm

In the current implementation of Algorithm IV.1 K and r_c are fixed, and chosen to satisfy (7). Thus, in order to wake up the nodes, the cell master only needs to broadcast the wake up signal together with the center position of the new cell. In case of conflicts, the master is selected by a static priority rule based on the node ID. In case the target is lost the Algorithm III.1 is activated to possibly detect it once again. The estimation is performed by Kalman filters, as described next.

A. Estimation

The estimation of the target motion is performed through a Kalman filter located at the master. The filter remains active during the cell life, until the cell changes and a filter in the new master is started.

We assume that each sensor measures the absolute position of the target in Cartesian coordinates. The estimation is based on model (1) coupled with the output equation

$$y(k) = Cx(k) + \nu(k), \tag{8}$$

for each sensor, where $y \in \mathbb{R}^p$, $C = \left[\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{smallmatrix}\right]$, $\nu(k) \triangleq \mathcal{N}(0, R)$, and the measurement error covariance R is assumed to be equal for each sensor. In the estimation phase the measurements coming from all the active sensors are exploited at the same time by the filter. Thus, the output matrix and the covariance matrix considered by the filter are, respectively,

$$C_f = \begin{bmatrix} C \\ \vdots \\ C \end{bmatrix}, \quad R_f = \begin{bmatrix} R & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & R \end{bmatrix}, \tag{9}$$

where $C_f \in \mathbb{R}^{n_a p \times n}$ and $R_f \in C \in \mathbb{R}^{n_a p \times n_a p}$, and n_a is the number of active sensors in the cell. A standard Kalman filter is considered, where the prediction step is

$$\begin{array}{rcl} \hat{x}(k|k-1) & = & A\hat{x}(k-1|k-1) + Bu(k), \\ P(k|k-1) & = & AP(k-1|k-1)A^T + Q, \end{array} \tag{10}$$

the estimation step is

$$\begin{array}{rcl} \hat{x}(k|k) & = & x(k|k-1) + M(k)[y(k) - C_f\hat{x}(k|k-1)], \\ P(k|k) & = & [I - M(k)C_f^T]P(k|k-1), \end{array} \tag{11}$$

and the time-varying Kalman gain is

$$M(k) = P(k|k-1)C_f^T[C_f P(k|k-1)C_f^T + R_f]^{-1}. \tag{12}$$

The filter is initialized by $\hat{x}(0|-1) = [s_x \ 0 \ s_y \ 0]^T$, where $[s_x \ s_y]$ is the position of the master node. Note that this strategy can be extended to the case of nonlinear target dynamics by using the Extended Kalman Filter.

B. Data fusion

The proposed approach represents the simplest data fusion, in which the filter fuses all the available data simply by augmenting the model used in the estimation and by considering a larger set of data [13]. However, such an approach is computationally expensive, because the dimension of the matrix to be inverted increases with the number of active sensors, hence the algorithm scalability is limited. In [14]

it has been shown that there is a functional equivalence between this filter and the one which operates on data that have been fused in advance [15]. In our case all the sensors are equal (they measure the same quantity and have the equal covariance R), hence a data-fusion Kalman filter which exploits two sensors has the same behavior than the fused-data Kalman filter which uses the averages of the measurements and a measurement covariance equal to $\frac{1}{2}R$.

The extension to many nodes has been derived in [14] through the information form of the Kalman filter. However, here we consider the case in which the nodes are homogenous, thus the results simplify largely.

Corollary 1: In a homogenous sensor network, the filter obtained by fusing n measurements with covariance R behaves as the one obtained by using the average of the n measurements and measurement error covariance $\frac{1}{n}R$.

Proof: Instead of applying the result in [14], we can prove the proposition by mathematical induction. The nodes are homogenous, thus they all have the same measurement error covariance R. Suppose that the filter that fuses h data is equivalent to the one that uses the average of h data and measurement covariance $\frac{1}{h}R$. Consider the case of the filter with $h+1$ data and assume that the first h data are pre-fused, and the last one is included by augmenting the model. Let $\tilde{E}_1$ be the average of prediction errors obtained from the first h sensors, $\tilde{E}_1 = \rho(E_1 + \dots E_h)$, $\rho^{-1} = h$, and $E_2 = E_{h+1}$ be the prediction error of sensor $h+1$. Accordingly, the covariance matrixes are $R_1 = \rho R$ and $R_2 = R$. Then

$$
\begin{aligned}
&M(k)\begin{bmatrix}\tilde{E}_1\\ \tilde{E}_2\end{bmatrix} = \\
&= PC^T\left[(1+\rho)CPC^TR + \rho R^2\right]^{-1}\left[R\tilde{E}_1 + \rho R\tilde{E}_2\right] = \\
&PC^T\left[CPC^T + \tfrac{\rho}{1+\rho}R\right]^{-1}\left[\tfrac{1}{1+\rho}\tilde{E}_1 + \tfrac{\rho}{1+\rho}\tilde{E}_2\right] = \\
&PC^T\left[CPC^T + \tfrac{1}{1+h}R\right]^{-1}\left[\tfrac{1}{1+h}\textstyle\sum_{i=1}^{h}E_i + \tfrac{1}{1+h}E_{h+1}\right],
\end{aligned}
\tag{13}
$$

which is the estimation update of a filter that uses the average of $h+1$ prediction errors and a measurement error covariance which is $\frac{1}{h+1}$ the original one. The estimation covariance update follows directly from (13). By induction, the proposition follows. ■

In this strategy the data are fused before the filtering process. The advantages are that the matrix to be inverted is smaller, thus a lower computational burden is required, saving energy and time, and that it is easier to reconfigure the estimation. If the number of nodes that measure the target changes, it only needs to change the measurement error covariance by considering the new number of nodes. If a stationary Kalman filter is used, the gains can be computed in advance for certain numbers h of active sensors[1] and dynamically selected.

V. Simulation Example

Consider a target crossing an area of 20×20 m^2, in which 600 sensor nodes have been randomly deployed with uniform distribution. The target moves accordingly to model (1), with initial velocities $v_x(0), v_y(0) \in [0,5]$ m/s. The sampling period is 10^{-2} s and the acceleration is a random vector $w(k) \triangleq \mathcal{N}(0, 3I)$.

The detection range used for master activation is $r_d = 1$ m, the cell radius is $r_c = 1.5$ m which causes an average sensor/cell number of 10.6. The cell life is 20 steps, thus a new master is selected every 0.2 seconds. It is assumed that the sensing range is such that (7) is satisfied ($r_s = 3.5$ m would be required, considering only the velocity, for such a conservative approach). The sensor measurements are affected by Gaussian disturbances $\nu(k) \in \mathcal{N}(0, 0.1 \cdot I)$ [m/s^2].

In addition to the process noise, on each period of 20 steps there is a probability of 0.2 that the target makes a turn. The step at which the turn occurs is random, with uniform distribution. When the target turns, its velocities on both axes are randomly reset to $v_x, v_y \in [0,5]$ m/s.

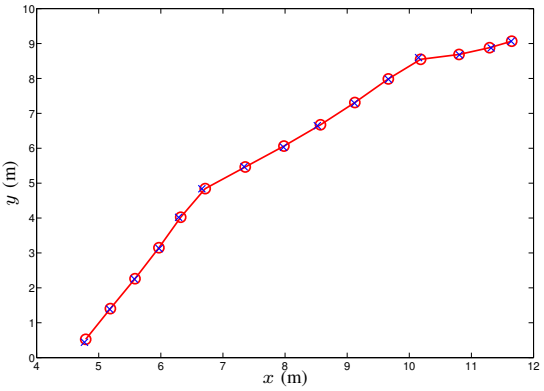
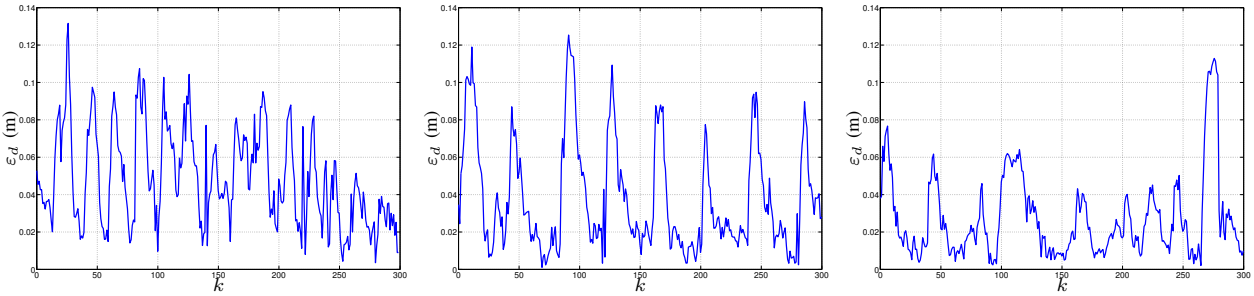


Fig. 2. Target positions (circles) and estimated positions (crosses) at the cell switch instants

The results of 300 steps of simulation are reported in Figure 2. The target position at the end of each cell life period is reported as a solid line marked by circles and the estimated states at such instants are reported as crosses. Figure 3(a) shows the distance error profile $\varepsilon_d(k)$ along the simulation, where $\varepsilon_d(k) = \sqrt{(p_x(k) - \hat{p}_x(k))^2 + (p_y(k) - \hat{p}_y(k))^2}$, $(p_x(k), p_y(k))$ is the target position at step k, and $(\hat{p}_x(k), \hat{p}_y(k))$ is the estimated position at step k. The peaks in the error are at the cell switch instants, since the estimation is reset and, in particular, no information on the target velocity is available. Additional peaks are due to target turns. The errors due to cell switches could be removed if when cell switches the previous master sends the current estimate and the current covariance to the next master. However, this would require additional communications, with the consequent use of battery energy.

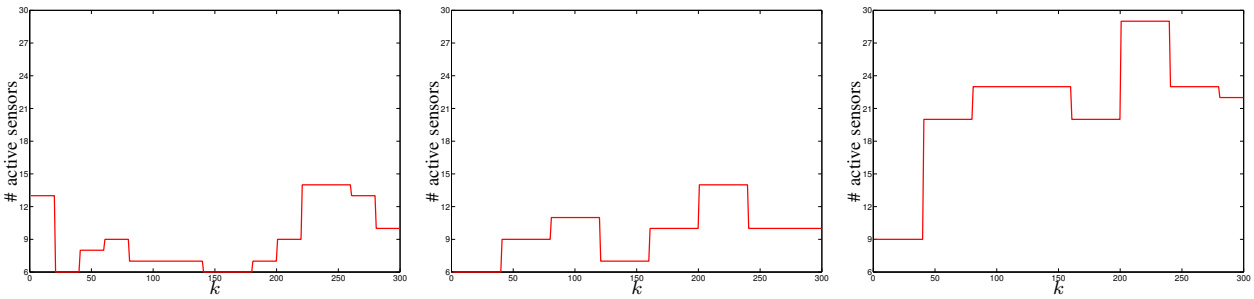
In Figure 3(b), the $\varepsilon_d(k)$ profile is reported for a different simulation with the same duration, and in which the cell life is increased to 40 steps; the sensing range of the node is increased accordingly. The error is lower in average, but the estimation error after the cell switches and the turns is still large. Finally, Figure 3(c) reports a simulation with the same

[1]For uniformly deployed sensors, the average number of active sensors is $\overline{n} = \frac{\pi r_c^2}{A_m}N$.



(a) Cell life 20 steps, 600 nodes (b) Cell life 40 steps, 600 nodes (c) Cell life 40 steps, 1200 nodes

Fig. 3. Distance error between real and estimated position



(a) Cell life 20 steps, 600 nodes (b) Cell life 40 steps, 600 nodes (c) Cell life 40 steps, 1200 nodes

Fig. 4. Number of active sensors during simulation

duration, in which the cell life is 40 steps and the number of sensor is increased to 1200. Since the mean number of active sensors in the cell is increased, as shown in Figure 4, the effect of measurement noise is decreased, accordingly to (13).

VI. Conclusions

We have considered the problem of detecting and tracking a moving target by exploiting a wireless sensor network. The proposed algorithms are fault tolerant, since node deaths only affect tracking performance without halting the estimation procedure, and scalable, because the algorithm does not depends on the number of nodes, except for the performance. A design technique based on detection confidence probability for some network parameters and a selective-activation tracking algorithm have been proposed. Such techniques are designed to minimize the communication between nodes, since those are the most power consuming operations and battery duration is critical in WSN applications. An experimental validation of the proposed algorithm is currently planned at the Automatic Control Laboratory of Siena, based on a WSN of Telos T-mote Sky motes.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and W. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, pp. 393–422, 2002.

[2] K. Sohrabi, J.G., V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, vol. 7, no. 5, pp. 16–27, Oct. 2000.

[3] B. Sinopoli, C. Sharp, S. Schaffert, L. Schenato, and S. Sastry, "Distributed control applications within sensor networks," in *Proc. IEEE*, vol. 91, Aug. 2003, pp. 1235–1246.

[4] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, May 2000.

[5] D. Atherton, J. Bather, and A. Briggs, "Data fusion for several kalman filters tracking a single target," *IEE Proc.-Radar Sonar Navigation*, vol. 152, no. 5, pp. 372–376, Oct. 2005.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *In Proc. of ACM MobiCom'00*, Boston, MA, 2000, pp. 56–67.

[7] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: application driver for wireless communications technology," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 2, pp. 20–41, Apr. 2001.

[8] S. Pattem, S. Poduri, and B. Krishnamachari, "Energy-quality tradeoffs for target tracking in wireless sensor networks," in *The 2nd Workshop on Information Processing in Sensor Networks (IPSN 2003)*, Palo Alto, CA, Apr. 2003, pp. 32–46.

[9] R. Moses, D. Krishnamurthy, and R. Patterson, "A self-localization method for wireless sensor networks," *EURASIP Journal on Applied Signal Processing*, vol. 4, pp. 348–358, Mar. 2003, special Issue on Sensor Networks.

[10] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for target tracking," *IEEE Signal Processing Magazine*, vol. 19, no. 2, Mar. 2002.

[11] C. Chong, F. Zhao, S. Mori, and S. Kumar, "Distributed tracking in ad hoc sensor networks," in *Proc. 6th Int. Conf. Information Fusion*, 2003, pp. 431–438.

[12] R. Popp, T. Kirubarajan, and K. Pattipati, "Survey of assignment techniques for multitarget tracking," in *Multitarget/Multisensor Tracking: Applications and Advances III*, Y. Bar-Shalom and W. D. Blair, Eds. Artech House, 2000.

[13] J. Manyika and H. Durrant-White, *Data fusion and sensor management: a decentralized information-theoretic approach*. Prentice-Hall, 1994.

[14] Q. Gan and C. Harris, "Comparison of two measurement fusion methods for Kalman-filter-based multisensor data fusion," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 37, no. 1, pp. 273–280, Jan. 2001.

[15] J. Roecker and C. McGillem, "Comparison of two-sensor tracking methods based on state vector fusion and measurement fusion," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 24, no. 4, pp. 447–479, 1988.