Cooperative Fault-Tolerant Target Tracking in Camera Sensor Networks

C. Laoudias, P. Tsangaridis, M. Polycarpou, C. Panayiotou, C. Kyrkou, T. G. Theocharides
KIOS Research Center for Intelligent Systems and Networks
Department of Electrical and Computer Engineering, University of Cyprus

Abstract—**Camera Sensor Networks (CSN) are becoming increasingly popular in a variety of security and safety-critical applications including public space surveillance, monitoring of attack-sensitive facilities, and critical infrastructure protection. Cameras in such networks are equipped with high-resolution visual sensors and on-board processors, while featuring wireless communication capabilities. These features enable the execution of various tasks, such as area coverage, activity recognition and target tracking, in a cooperative fashion. However, the performance of CSN may be compromised when faults occur, either due to unintentional software and hardware faults or as the result of a malicious attack. Paving the way for fault tolerance in CSN-based target tracking, we introduce a flexible fault model that can be used to generate different types of erroneous behaviour, thus simulating realistic faults in CSN. We also propose a fault-tolerant decentralized solution for tracking a target that passes through the area monitored by the CSN. Our simulation results indicate that the proposed solution is able to track the target reliably despite the presence of faults.**

I. Introduction

Target tracking is a popular application in Wireless Sensor Networks (WSN) owing to the ease of deploying a large number of cheap sensor nodes inside the monitoring area. Due to the requirement for uninterrupted and reliable network operation, even in hostile environments or in case of unpredictable events, fault tolerance is of utmost importance. Therefore, decentralized or distributed tracking approaches are usually preferred, instead of centralized solutions, because they avoid the use of a central processing unit (sink) that introduces a single point of failure to the system.

Distributed approaches [1] are robust to faults because target tracking information is maintained in several nodes across the network. However, they usually require more powerful nodes in terms of processing power and communication capabilities due to the consensus algorithms that ensure agreement among neighbouring nodes. Hence, decentralized solutions [2], [3] may be preferred for some application scenarios because they provide a good trade-off between resources and resilience to faults. In such approaches, a cluster is formed when a target is detected and all nodes in the vicinity of the target forward their observations to the elected leader (i.e., cluster head) that is responsible for the tracking task.

Recently, emerging Camera Sensor Networks (CSN) offer advanced sensing (i.e., high-resolution visual information) and collaboration capabilities that facilitate the development of cooperative tracking solutions [4]. However, existing decentralized approaches and underlying leader election protocols assume an omnidirectional sensing model, i.e., the target can be detected within a circular disc with radius equal to the node's sensing range. Thus, they are not directly applicable to CSN tracking due to the directional sensing and limitations in the coverage area (i.e., camera field of view). This necessitates the development of appropriate leader election protocols for CSN tracking applications.

With respect to fault tolerance, still in decentralized solutions individual nodes are likely to suffer faults because of harsh environmental conditions (e.g., high temperatures) or exhibit erroneous behaviour due to a malicious attack. For example, if a camera fails to detect and identify a specific target, due to software errors, miscalculations in the detection and data association modules or hardware faults in the lens or mechanical parts, then that camera would falsely report that the target is not present, while it moves inside its field of view.

This scenario highlights that fault tolerance is a highly desirable property, however, only a few works including [5], have addressed this issue in CSN applications. Motivated by the key importance of fault tolerance, our contribution in this work is twofold. First, we propose a flexible fault model to generate noisy observations and inject different types of realistic faults, which affect camera sensing capabilities and may degrade tracking accuracy. To our knowledge, this is the first attempt to develop a model for CSN faults. Second, we design a decentralized tracking solution featuring a novel distributed leader election protocol, which is suitable for the directional sensing nature of CSN. This protocol is combined with a robust voting-based target localization algorithm, and a location smoothing component based on Kalman filter.

The rest of this paper is structured as follows. Section II briefly overviews related works. In Section III we formulate the problem of target tracking in CSN and introduce our network, sensing and fault models. Section IV presents the details of our decentralized tracking solution. In Section V we evaluate the fault tolerance of the proposed solution. Finally, Section VI provides concluding remarks and discusses directions for future work.

II. Related Work

Decentralized tracking solutions involve a cooperative leader election protocol for dynamically electing a new leader node, while the target moves or new targets enter the field. Such solutions have been applied successfully to WSN, including binary networks [3], [6] where sensor nodes report whether they detect a target or not.

The deployment of CSN enables a wide variety of cooperative applications, including area coverage [7] and activity recognition [8]. Regarding tracking applications, authors in [4] discuss distributed and decentralized multicamera solutions, while an online approach to control camera parameters for distributed tracking is presented in [9]. However, most existing CSN systems assume fault-free conditions and do not take into account possible malfunctions or software errors that may disrupt normal operation. For instance, authors in [5] focus on errors in the horizontal orientation (i.e., pan) of cameras during target tracking, due to initial calibration inaccuracies (modelled as Gaussian noise) and external effects that cause the camera orientation to take arbitrary values, i.e., Byzantine faults (modelled as random orientation bias).

III. Modelling and Problem Formulation

For our target tracking application in CSN we define the following environment and models.

A. Camera Network Model

1) A square field $\mathcal{A}$ that is discretized using grid $\mathcal{G}$ of N_g grid points with coordinates $\mathbf{g}_n = (x_n, y_n),\ n = 1, \ldots, N_g$, while the grid resolution is G_s.
2) A network of N_c static cameras $\mathcal{C} = \{c_1, \ldots, c_{N_c}\}$ that are spread over $\mathcal{A}$. Their positions $\mathbf{p}_i = (x_i, y_i)$ and orientations $\theta_i,\ i = 1, \ldots, N_c$ are known[1].
3) All cameras have the same limited communication range[2]. This range, denoted R_c, defines the set of neighbours $\mathcal{N}_i$ of camera c_i such that $\mathcal{N}_i = \{c_n : d(i,n) \leq R_c\}$, where $d(i,n) = ||\mathbf{p}_i - \mathbf{p}_n||$ is the Euclidean distance between cameras c_i and c_n.
4) A set of N_t targets $\mathcal{T} = \{t^1, \ldots, t^{N_t}\}$ moving inside $\mathcal{A}$. At time step k each target is located at $\mathbf{p}^j(k) = (x^j(k), y^j(k)),\ j = 1, \ldots, N_t$, while the number of targets may change over time.

B. Camera Sensing Model

All cameras have the same sensing range R_s that depends on the imaging capabilities of the hardware. In the following we define different areas of interest that depend on R_s.

The *Field of View* of camera c_i, denoted $\mathcal{F}_i \subseteq \mathcal{G}$, is the subset of grid points $\mathbf{g}_n \in \mathcal{G}$ where c_i can sense the presence of a target. Typically, by projecting the 3D visual sensing cone of camera c_i onto the 2D field $\mathcal{A}$, $\mathcal{F}_i$ can be approximated by a triangle with height equal to R_s and the angle of $\mathcal{F}_i$ is 2α; see the illustration in Fig. 1.

However, camera c_i may not be able to determine the exact location of a target inside $\mathcal{F}_i$. This is typically the case with cheap cameras, which may suffer manufacturing impairments in the visual sensor or inaccuracies in the image processing algorithms that introduce uncertainty (noise) in target location.

[1]For example, they can be computed shortly after network deployment by using a distributed camera calibration algorithm [10].

[2]This assumption can be relaxed without compromising the validity of the model because it would only affect the cardinality of $\mathcal{N}_i$.

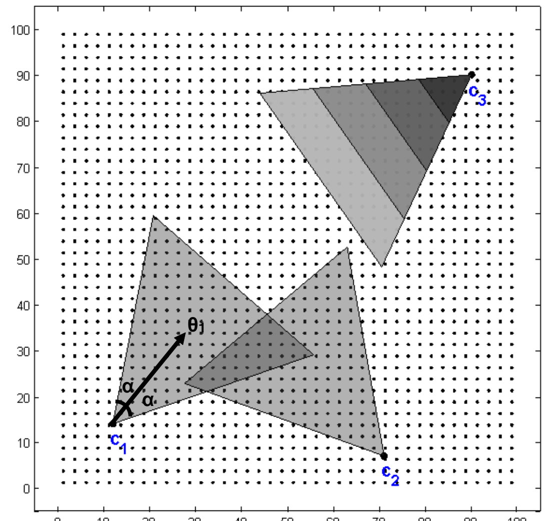


Fig. 1. Three cameras $c_i,\ i = 1, \ldots, 3$ are deployed inside $\mathcal{A}$ ($G_s = 2.5$ units) and $\mathcal{F}_i$ is equilateral triangle (i.e., $\alpha = 30°$) with height $R_s = 40$ units (shaded area). The zones of camera c_3 ($N_z = 4$) are shown with different shades of gray.

Hence, such cameras usually report a subregion (zone) inside $\mathcal{F}_i$ where the target resides, rather than a single location.

The *Zone of View* of camera c_i, denoted $\mathcal{Z}_{i,m} \subseteq \mathcal{F}_i$, where $m = 1, \ldots, N_z$, is the subset of grid points $\mathbf{g}_n \in \mathcal{F}_i$ where c_i can sense the presence of a target. In this sense, $\mathcal{F}_i$ is divided into a number of non-overlapping sensing zones such that $\mathcal{F}_i = \bigcup_{m=1}^{N_z} \mathcal{Z}_{i,m}$. The notion of zones is depicted in Fig. 1 for camera c_3.

We assume that camera c_i can consistently identify each target within $\mathcal{F}_i$ by employing appropriate distributed data association mechanisms; see [11] and references therein. Subsequently, it indicates the corresponding $\mathcal{Z}_{i,m}$ where target t^j lies. In this sense, the detection status $A^j_{i,m}(k)$ of camera c_i pertaining to target t^j at time step k is given by

$$A^j_{i,m}(k) = \begin{cases} 1 & \text{if } \mathbf{p}^j(k) \in \mathcal{Z}_{i,m} \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

If camera c_i detects target t^j inside $\mathcal{Z}_{i,m}$ then $A^j_{i,m}(k) = 1$ and sends a message to all neighbouring cameras $c_n \in \mathcal{N}_i$; otherwise, it remains silent. The detection statuses $A^j_{i,m}$ are forwarded to a camera in the vicinity of the target acting as leader; see Section IV-A for the details of our distributed directional leader election protocol. The leader combines relevant information from its set of neighbours to estimate the current location of target t^j, as discussed in Section IV-B.

In real-life applications, however, camera c_i may not always be able to reliably determine the correct zone where a target resides. For instance, it may report that the target is located inside a neighbouring zone due to faults in the target detection module. In another case, camera c_i may falsely report that a target is not present, e.g., when the target passes behind an obstacle while the target is actually there, or a detection message may be dropped due to network operation reasons.

Alternatively, camera c_i may wrongly report that a target is present inside $\mathcal{F}_i$, e.g., by confusing one target with another one due to errors in the image processing algorithms.

All such scenarios degrade sensing accuracy and may introduce faults in camera observations, which affect the decisions that are made based on these observations.

C. Fault Model

We employ a probabilistic fault model to capture and simulate the undesired effects described above. Our model disturbs the ideal measurements of camera c_i pertaining to target t^j, i.e., the detection statuses $A^j_{i,m}$, according to probability vector $\mathbf{P}_{ij} = [P^c_{ij}\ P^w_{ij}\ P^{fn}_{ij}\ P^r_{ij}\ P^{fp}_{ij}]$.

In case target t^j is located *inside* $\mathcal{Z}_{i,m}$, then we have the following possibilities:

- P^c_{ij}: $\mathbf{P}[c_i$ senses t^j in the *correct* zone $m]$. In this event $A^j_{i,m} = 1$.
- P^w_{ij}: $\mathbf{P}[c_i$ senses t^j in the *wrong* zone $m']$. In this event $A^j_{i,m} = 0$ and $A^j_{i,m'} = 1$ for another zone $m' \neq m$.
- P^{fn}_{ij}: $\mathbf{P}[c_i$ *falsely* does not sense $t^j]$. In this event $A^j_{i,m} = 0,\ \forall m$, i.e., *false negative* observation.

Alternatively, in case target t^j is located *outside* $\mathcal{F}_i$, then we have the following possibilities:

- P^r_{ij}: $\mathbf{P}[c_i$ *correctly* does not sense $t^j]$. In this event $A^j_{i,m} = 0,\ \forall m$.
- P^{fp}_{ij}: $\mathbf{P}[c_i$ *falsely* senses $t^j]$. In this event $A^j_{i,m} = 1$ for a randomly selected zone m, i.e., *false positive* observation.

In this model, $\mathbf{P}[]$ denotes the probability of the corresponding event, while $P^c_{ij} + P^w_{ij} + P^{fn}_{ij} = 1$ and $P^r_{ij} + P^{fp}_{ij} = 1$. During normal operation, when no faults have occurred, P^c_{ij} and P^r_{ij} will be equal to 1, while P^w_{ij}, P^{fn}_{ij}, and P^{fp}_{ij} will be equal to 0.

Our sensing model has low computational cost and is able to simulate different types of faulty behaviour. For instance, increasing P^{fn}_{ij}, while decreasing P^c_{ij} accordingly, generates random *temporary* faults where the faulty cameras do not report the presence of a target. Note that such behaviour can be also attributed to random failures in the communication links, which lead to dropped detection messages.

Setting $P^{fn}_{ij} = 1$ for a subset of cameras in the field would simulate *permanent* faults of this type, e.g., in case target detection or data association modules do not work properly and the identification of some or all targets fails permanently. Alternatively, this behaviour can be caused by a malicious attacker who might place a decoy picture of a target-free environment in front of the camera to prevent intruder detection. In addition, we can simulate the same erroneous behaviour for neighbouring cameras, thus generating *spatially correlated* faults. For instance, in case there is a fire in a part of the field and smoke degrades the target detection capabilities of the cameras in the immediate vicinity.

Similarly, we may adjust P^w_{ij}, e.g., to simulate the performance degradation of the target detection module owing to harsh environmental conditions, optical calibration errors, or due to prolonged operation time. In another scenario, by controlling P^{fp}_{ij} we can generate false detections across the field, e.g., due to errors in the target detection and identification modules (a target is detected when it is not actually within the field of view or target t^j is mistaken for another target $t^{j'}$) or in case an opponent unleashes a malign attack by releasing decoy targets around the field. As another example, the orientation fault considered in [5] can be simulated by increasing both P^{fn}_{ij} and P^{fp}_{ij} for that camera.

The above scenarios demonstrate the effectiveness of the proposed camera sensing and fault models to simulate diverse erroneous behaviours. Note that the evaluation of real scenarios would require realistic probabilities. These can be determined by offline analysis of publicly available video tracking datasets or by allowing the CSN system to run for some time and comparing the camera readings against the ground truth (i.e., human operator).

IV. Decentralized Target Tracking

Our decentralized target tracking solution integrates three components, namely a directional leader camera election protocol, a target localization algorithm and a location smoothing algorithm. Next, we present these components in detail.

A. Directional Leader Camera Election Protocol

When target t^j traverses the monitored area a subset of cameras in proximity detect it. The objective of our directional Leader Camera Election Protocol (LCEP) is to elect a leader c_l among those cameras in a distributed fashion that will be responsible for handling the localization and tracking tasks.

Note that LCEP is invoked occasionally to elect a new leader dynamically as the target moves inside the field. Moreover, to reduce leadership changes or possible leader oscillations and prevent unnecessary computations, current leader c_l maintains leadership while t^j is within its field of view $\mathcal{F}_l$. When t^j is about to leave $\mathcal{F}_l$, c_l notifies its neighbours to invoke LCEP again for electing a new leader.

For convenience, before proceeding with the description of LCEP, we define the *effective* neighbours of a camera.

Definition 1: Camera c_n is an *effective* neighbour of camera c_i that detects target t^j ($A^j_{i,m} = 1$), if the following two conditions hold: (i) c_n is a neighbour of c_i, i.e., $c_n \in \mathcal{N}_i$, and (ii) the field of view of c_n overlaps with the detection zone of c_i, i.e., $\mathcal{F}_n \cap \mathcal{Z}_{i,m} \neq \emptyset$.

The above definition introduces the notion of *effective* set of neighbours of camera c_i, denoted $\mathcal{N}'_i$, which is the subset of cameras $c_n \in \mathcal{N}_i$ that sense target t^j, including camera c_i itself. For instance, if camera c_n is within communication range R_c from c_i, but orientation θ_n does not allow camera c_n to sense target t^j, i.e., $A^j_{n,m} = 0,\ \forall m$, then $c_n \notin \mathcal{N}'_i$. In this sense, LCEP incorporates the inherent directional sensing of CSN in the leader election process. Obviously, $\mathcal{N}'_i \subseteq \mathcal{N}_i$.

We assume that during CSN deployment, every camera c_i shares with its neighbours $c_n \in \mathcal{N}_i$ information about the area that it covers, i.e., the grid points that belong to its zones. This is accomplished by broadcasting an *initialization* message *init-msg*$\{i, \mathbf{p}_i, \theta_i, \mathcal{Z}_{i,1}, \ldots, \mathcal{Z}_{i,N_z}\}$. Note that these information are shared across the network once in a single round prior to target tracking. In this fashion, camera c_i also receives similar messages and checks for overlapping grid points between $\mathcal{Z}_{i,m}$ and $\mathcal{Z}_{n,m}$, $\forall m$, $\forall c_n \in \mathcal{N}_i$. This information in conveniently stored locally in a look-up table to facilitate the identification of *effective* set of neighbours during LCEP execution.

Initially, camera c_i that senses target t^j broadcasts a *detection* message *det-msg*$\{i, j, m\}$ to all neighbouring cameras $c_n \in \mathcal{N}_i$; see Algorithm 1. In the same way, based on the received messages, c_i is able to determine $\mathcal{N}_i'$. Next, c_i computes the number of *effective* neighbours, denoted $N_i^{eff} \equiv |\mathcal{N}_i'|$, and the number of *effective* neighbours detecting target t^j, denoted N_i^{det}, where $N_i^{det} \leq N_i^{eff}$. Note that if c_i does not receive a *detection* message from a neighbour this implies that the latter did not sense t^j.

Subsequently, c_i broadcasts a *candidate leader* message *cand-msg*$\{i, j, N_i^{det}, N_i^{eff}\}$ and waits for a period of time $T_w = f(1/N_i^{det})$, where $f(x)$ is a strictly increasing function. During this time, c_i receives *candidate leader* messages from neighbouring cameras and compares the corresponding values with its own local value. If $N_i^{det} > N_n^{det}$, $\forall c_n \in \mathcal{N}_i'$ then it becomes leader ($c_l \equiv c_i$) and broadcasts a *leader* message *ldr-msg*$\{l, j\}$ to its neighbours. If $N_i^{det} = N_n^{det}$, $c_n \in \mathcal{N}_i'$, then to resolve the tie c_i checks whether $N_i^{eff} > N_n^{eff}$. If this is the case, then it undertakes leadership ($c_l \equiv c_i$) and broadcasts a *leader* message *ldr-msg*$\{l, j\}$ to its neighbours. If camera c_i receives a *candidate leader* message from camera c_n with $N_i^{det} < N_n^{det}$ or alternatively $N_i^{det} = N_n^{det}$ and $N_i^{eff} < N_n^{eff}$, then c_i pauses the leader election process and cannot become leader until the protocol runs again.

Current leader c_l maintains leadership while t^j moves within $\mathcal{F}_l$ by periodically retransmitting *ldr-msg*$\{l, j\}$. When t^j is about to leave $\mathcal{F}_l$, c_l broadcasts *ldr-msg*$\{null, j\}$ so that all neighbouring cameras can invoke LCEP again for electing a new leader $c_l \neq c_i$ to continue tracking. In case target t^j leaves $\mathcal{F}_l$ but goes into an area that is not covered by any cameras, then our system terminates the associated track. Later on, when t^j moves into a covered area, another subset of cameras will detect it and invoke LCEP for starting a new track.

Algorithm 1: Leader Camera Election Protocol (LCEP)

```
Input: Set of cameras detecting a target.
Output: Elected leader camera $c_l$.
 1: Broadcast det-msg$\{i, j, m\}$ to all cameras $c_n \in \mathcal{N}_i$.
 2: Use the received det-msg$\{n, j, m\}$, $\forall c_n \in \mathcal{N}_i$ to determine $\mathcal{N}_i'$ and compute $N_i^{eff}$ and $N_i^{det}$.
 3: Broadcast cand-msg$\{i, j, N_i^{det}, N_i^{eff}\}$.
 4: Wait for time $T_w = f(1/N_i^{det})$, while receiving cand-msg$\{n, j, N_n^{det}, N_n^{eff}\}$, $\forall c_n \in \mathcal{N}_i'$.
 5: If $N_i^{det} > N_n^{det}$, $\forall c_n \in \mathcal{N}_i'$ then assume leadership ($c_l \equiv c_i$) and broadcast ldr-msg$\{l, j\}$ to neighbours.
 6: ElseIf $N_i^{det} < N_n^{det}$, $\forall c_n \in \mathcal{N}_i'$ then STOP until ldr-msg$\{null, j\}$ is received.
 7: Else // $N_i^{det} = N_n^{det}$ for some camera $c_n$
 8:     If $N_i^{eff} > N_n^{eff}$ then $c_l \equiv c_i$ and broadcast ldr-msg$\{l, j\}$ to neighbours.
 9:     Else STOP until ldr-msg$\{null, j\}$ is received.
10:     EndIf
11: EndIf
```

B. Voting-based Localization and Tracking

For localization, we employ a voting-based scheme where current leader c_l aggregates the decisions of its *effective* neighbours to determine target location.

In particular, camera c_l maintains a $|\mathcal{S}| \times |\mathcal{N}_l'|$ voting matrix $\mathbf{V}^j$ for target t^j to store the decisions of cameras $c_n \in \mathcal{N}_l'$, where $|\cdot|$ denotes the cardinality of the corresponding set and $\mathcal{S} = \bigcup_{c_n \in \mathcal{N}_l'} \mathcal{F}_n$ is the subset of the grid points g_q that are covered by all *effective* neighbours. The elements of the voting matrix $\mathbf{V}^j(q, n)$ are formally given by

$$\mathbf{V}^j(q,n) = \begin{cases} +1 & g_q \in \mathcal{Z}_{n,m} \text{ AND } A^j_{n,m} = 1 \\ -1 & \forall m',\ g_q \in \mathcal{Z}_{n,m'} \text{ AND } A^j_{n,m'} = 0 \\ 0 & g_q \notin \mathcal{F}_n \end{cases}, \tag{2}$$

where q is the index of associated grid points and n is the index of related neighbouring cameras.

The main idea is that if camera c_n senses target t^j, then it is certain about its presence inside $\mathcal{F}_n$. Thus, c_n upvotes $(+1)$ the grid points that fall inside its detection zone $\mathcal{Z}_{n,m}$, while it downvotes (-1) the grid points of other zones. Similarly, if camera c_n does not sense a target, then it downvotes all the grid points inside $\mathcal{F}_n$ because $A^j_{n,m'} = 0$, $\forall m'$. Finally, camera c_n is uncertain about the presence of a target outside $\mathcal{F}_n$, thus contributing 0 to the corresponding grid points. Using ± 1 values in such voting scheme was shown to be effective in target localization and tracking applications, while tolerating a large number of faulty binary sensor observations [12].

Note that while t^j moves inside $\mathcal{F}_l$, current leader c_l still receives *det-msg*$\{n, j, m\}$ from neighbouring cameras and may need to update $\mathcal{N}_l'$ according to definition 1. Consequently, the voting matrix $\mathbf{V}^j$ may need to be updated in case t^j is sensed in a different zone. However, such an update can be implemented quickly by employing the local look-up table.

Subsequently, for each grid point g_q, c_l sums the contributions of cameras $c_n \in \mathcal{N}_l'$ pertaining to target t^j to obtain the voting result V_q^j as

$$V_q^j = \sum_{c_n \in \mathcal{N}_l'} \mathbf{V}^j(q, n). \tag{3}$$

Target t^j is localized at the grid point with the maximum voting value given by

$$\hat{\mathbf{p}}^j = \arg\max_{g_q} V_q^j, \tag{4}$$

or at the centroid of the maximum value grid points.

Subsequently, the leader runs a Kalman filter algorithm that first predicts future target location according to a mobility model (e.g., random force) and then updates this prediction by leveraging the output of the localization algorithm to get the refined location estimate; see [6] for the iterative Kalman filter algorithm. When a new leader is to be elected, as discussed previously, current leader passes on to the next leader the required information to continue computations within the Kalman filter, thus ensuring uninterrupted target tracking.

V. Simulation Results

In our simulation setup, $\mathcal{A}$ is a 100×100 square CSN field with grid resolution $G_s = 2$ units. There are $N_c = 150$ randomly deployed cameras for tracking a single target t^1 that traverses the field with constant speed by following a staircase path, where $(10, 50)$ is the start point and $(90, 50)$ is the end point[3]. Cameras have the same hardware characteristics and their field of view is equilateral triangle, as shown in Fig. 1, with height equal to the sensing range $R_s = 30$ units. When a camera detects t^1 it reports its presence inside a specific zone with $N_z = 5$. This information is shared with neighbouring cameras located within communication range $R_c = 45$ units for collaboratively tracking the target.

We employ the fault model discussed in Section III-C, and assume that it is the same for all cameras[4], i.e., $\mathbf{P}_{ij} = \mathbf{P} = [P^c\ P^w\ P^{fn}\ P^r\ P^{fp}]$. We start with $\mathbf{P} = [1\ 0\ 0\ 1\ 0]$ to simulate fault-free conditions and then investigate the fault tolerance of the proposed target tracking solution by varying one of the probabilities P^w, P^{fn} and P^{fp}, while keeping others constant. In particular, we report the mean tracking error pertaining to the target path when faults are present. Essentially, a tracking solution is considered fault-tolerant if it exhibits smooth performance degradation as the probability of camera faults increases. Results are averaged 20 runs, where each run uses a random camera deployment.

To gain some insight about the properties of our decentralized target tracking solution we start by investigating the performance of the proposed directional leader election protocol. In particular, we compare the following solutions:

1) Centralized tracking, referred to as CEN, where all camera observations are forwarded to the sink that runs the localization and tracking algorithms.
2) Decentralized tracking, referred to as LCEP, which employs the proposed LCEP discussed in Section IV-A for electing a leader camera close to the target that undertakes the localization and tracking tasks.
3) Decentralized tracking, referred to as RLEP, which employs a Random Leader Election Protocol (RLEP). The main difference from LCEP is that RLEP elects a leader randomly among cameras that detect the target, i.e., without taking into consideration the number of *effective* neighbours for each candidate leader.

[3]Relative performance of various solutions does not vary much with more or less cameras and the results are omitted due to space limitations, however tracking multiple targets is not trivial and is part of our ongoing work.

[4]For brevity we also drop subscript j because only target t^1 is considered.

The tracking error in the presence of false positive faults is depicted in Fig. 2a. Results indicate that when our LCEP election protocol is employed, accuracy degrades smoothly and the tracking error is considerably lower compared to the case of RLEP, as the number of faulty cameras is increased. Importantly, the accuracy achieved by our decentralized tracking solution is close to the accuracy of CEN, which has a global view of the field.

Next, we study the fault tolerance of the proposed tracking system and compare it against alternative solutions that rely on LCEP for leader election, but employ different localization algorithms instead of our voting-based algorithm. Note that these algorithms were originally introduced for target tracking in binary WSN. Thus, to provide a fair comparison with our solution, we have properly adapted them to the CSN setup by taking into account the directional sensing of cameras. Specifically, we consider the following counterparts:

1) The Centroid Estimator (CE) [13], which localizes the target at the centroid of the grid points that fall into the detection zones of those cameras that detect the target in the neighbourhood of the current leader.
2) The Closest Point Approach (CPA) [2], which localizes the target at the centroid of the grid points that fall into the detection zone of the current leader.
3) The Maximum Likelihood (ML) estimator [14].

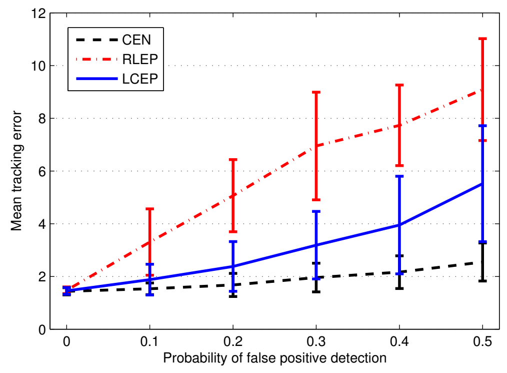
The tracking error for increasing probability of detecting the target in the wrong zone (P^w) is depicted in Fig. 2b. We observe that in the fault-free case (i.e., $P^w = 0$) the proposed solution and ML achieve the lowest tracking error, which is considerably better than CE and CPA. As P^w is increased the tracking error of our system increases gradually, but it still outperforms CE that is resilient to this type of fault.

The system that employs ML for localization is severely affected in case faulty cameras do not to report the presence of t^1 even though it is actually there, as shown in Fig. 2c. This is attributed to the fact that the likelihood of a target to reside in a specific area is greatly reduced even if a single camera reports that it does not detect the target there. In contrast, the proposed solution degrades smoothly as P^{fn} increases. Notably, the tracking error remains well below CE and CPA, which are both not affected by this type of fault because they consider only those cameras that detect the target. On the other hand, CE and CPA are extremely sensitive to cameras that falsely report that they have detected the target, as shown in Fig. 2d. We observe, however, that the proposed solution can handle this type of fault and is able to tolerate a significant number of false positive detections as probability P^{fp} increases.

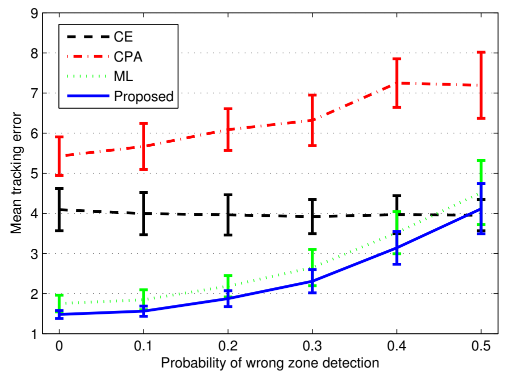
To summarize, combining LCEP with voting-based localization seems to be a promising solution for target tracking in CSN when faulty cameras are considered. Our findings suggest that it can tolerate different types of faults, thus keeping tracking error low as more cameras become faulty.

VI. Conclusions

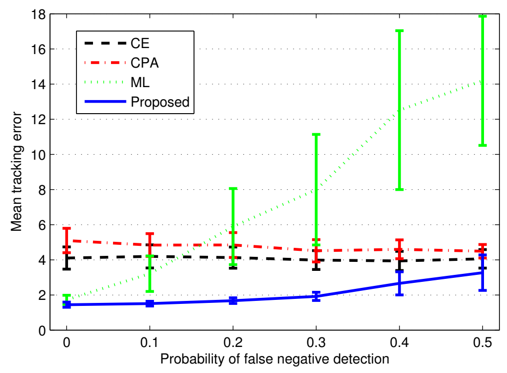
In this work, we have introduced a flexible fault model that is capable to simulate different types of faults that may com-



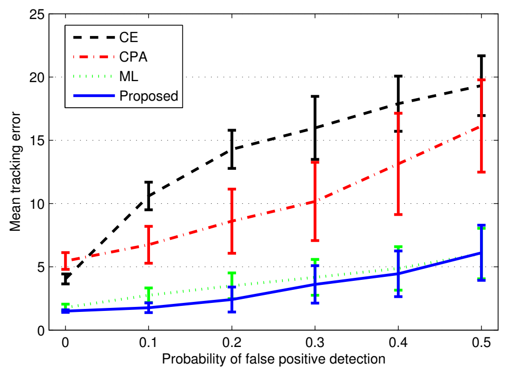
(a) Performance of LCEP for varying probability P^{fp}.



(b) Sensing the target in the wrong zone with probability P^{w}.



(c) Falsely not sensing the target with probability P^{fn}.



(d) Falsely sensing the target with probability P^{fp}.

Fig. 2. Tracking error of the decentralized target tracking solution in the presence of different types of camera faults.

promise tracking accuracy in CSN. This model was employed to assess the fault tolerance of the proposed decentralized target tracking system. Simulation results indicate that our solution can mitigate the effect of erroneous observations and is resilient to faulty camera behaviours. As part of future work, we also plan to explore fault detection mechanisms to identify and isolate faulty cameras, thus further enhancing the fault accommodation capabilities of our target tracking system.

Acknowledgement

This research has been funded by the European Research Council under the ERC Advanced Grant ERC-2011-ADG-291508 "Fault-Adaptive Monitoring and Control of Complex Distributed Dynamical System" (FAULT-ADAPTIVE).

References

[1] H. Chen and K. Sezaki, "Distributed target tracking algorithm for wireless sensor networks," in *IEEE International Conference on Communications (ICC)*, 2011, pp. 1–5.

[2] J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed group management in sensor networks: Algorithms and applications to localization and tracking," *Telecommunication Systems*, vol. 26, no. 2-4, pp. 235–251, 2004.

[3] J. Teng, H. Snoussi, and C. Richard, "Decentralized variational filtering for target tracking in binary sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 10, pp. 1465–1477, 2010.

[4] M. Taj and A. Cavallaro, "Distributed and decentralized multicamera tracking," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 46–58, 2011.

[5] M. Karakaya and H. Qi, "Collaborative localization in visual sensor networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 2, pp. 1–24, 2014.

[6] M. Michaelides, C. Laoudias, and C. Panayiotou, "Fault tolerant localization and tracking of multiple sources in WSNs using binary data," *IEEE Transactions on Mobile Computing*, vol. 13, no. 6, pp. 1213–1227, June 2014.

[7] P. Zhou, J. Wu, and C. Long, "Probability-based optimal coverage of PTZ camera networks," in *IEEE International Conference on Communications (ICC)*, 2012, pp. 218–222.

[8] B. Song, A. Kamal, C. Soto, C. Ding, J. Farrell, and A. Roy-Chowdhury, "Tracking and activity recognition through consensus in distributed camera networks," *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2564–2579, 2010.

[9] A. Kamal, J. Farrell, and A. Roy-Chowdhury, "Information weighted consensus filters and their application in distributed camera networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3112–3125, 2013.

[10] D. Devarajan and R. J. Radke, "Calibrating distributed camera networks using belief propagation," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, pp. 221–231, 2007.

[11] J. Wan and L. Liu, "Distributed data association in smart camera networks using belief propagation," *ACM Transactions on Sensor Networks*, vol. 10, no. 2, pp. 1–24, 2014.

[12] M. P. Michaelides and C. G. Panayiotou, "SNAP: Fault tolerant event location estimation in sensor networks using binary data," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1185–1197, 2009.

[13] M. Ding, F. Liu, A. Thaeler, D. Chen, and X. Cheng, "Fault-tolerant target localization in sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, no. 1, pp. 19–19, 2007.

[14] R. Niu and P. Varshney, "Target location estimation in sensor networks with quantized data," *IEEE Transactions on Signal Processing*, vol. 54, no. 12, pp. 4519–4528, 2006.