Estimating the Spatiotemporal Evolution Characteristics of Diffusive Hazards using Wireless Sensor Networks

Dimitris V. Manatakis, *Student Member, IEEE,* and Elias S. Manolakos, *Senior Member, IEEE*

Abstract—There is a fast growing interest in exploiting Wireless Sensor Networks (WSNs) for tracking the boundaries and predicting the evolution properties of diffusive hazardous phenomena (e.g. wildfires, oil slicks etc.) often modeled as “continuous objects”. We present a novel distributed algorithm for estimating and tracking the local evolution characteristics of continuous objects. The hazard’s front line is approximated as a set of line segments, and the spatiotemporal evolution of each segment is modeled by a small number of parameters (orientation, direction and speed of motion). As the hazard approaches, these parameters are re-estimated using ad-hoc clusters (triplets) of collaborating sensor nodes. Parameters updating is based on algebraic closed-form expressions resulting from the analytical solution of a Bayesian estimation problem. Therefore, it can be implemented by microprocessors of the WSN nodes, while respecting their limited processing capabilities and strict energy constraints. Extensive computer simulations demonstrate the ability of the proposed distributed algorithm to estimate accurately the evolution characteristics of complex hazard fronts under different conditions by using reasonably dense WSNs. The proposed in-network processing scheme does not require sensor node clocks synchronization and is shown to be robust to sensor node failures and communication link failures, which are expected in harsh environments.

Index Terms—Environmental hazard; continuous object; predictive modeling; distributed estimation; Bayesian estimation; Wireless Sensor Networks

1 Introduction

Tracking objects (i.e. determining their location over time) has been a fundamental problem with numerous applications (surveillance, aviation, military etc.). Apart from determining the trajectory of the objects, it is also important to estimate their motion characteristics (e.g. direction and speed) in real-time, since this information can be used to predict their future locations and understand their overall spatiotemporal behavior.

During the last decade there has been a fast growing interest in exploiting the capabilities of Wireless Sensor Networks (WSNs) in a variety of application domains (health, military, IoT, etc.). WSNs have also been used for single and multiple target tracking [1]–[3], and due to their rapidly dropping cost they are also gaining popularity in environmental monitoring applications [4]. Recently, sensor network-based methods have been proposed for detecting the boundaries of diffusive hazardous phenomena [5]–[12], modeled as “continuous objects”. However, traditional target tracking algorithms cannot be applied for continuous object tracking, since the two problems are fundamentally different. Continuous objects (such as wildfires, oil spills, diffusing biochemical materials etc.) tend to occupy a large area and their size and shape is continuously changing (albeit smoothly) with time. To be able to track their boundaries, a large number of cooperating sensor nodes is needed, increasing faster than linearly with the area occupied by the continuous object. In contrast, discrete targets (such as vehicles, animals, humans etc.) have a very small size compared to the WSN’s deployment area and a much smaller number of sensor nodes usually suffices to track their trail.

The key idea behind WSN-based continuous object tracking methods has been to identify over time the sensor nodes located closest to the evolving object’s front line (boundary nodes). Although these methods can estimate implicitly the boundaries of an evolving hazard, they require unrealistic network densities (thousands of deployed sensor nodes per km^2) which renders them impractical even for small scale environmental monitoring applications. Furthermore, the reported algorithms (with few exceptions as those in [11], [12]) do not provide information about the spatiotemporal evolution characteristics (e.g. direction and speed) of the diffusing phenomenon and therefore cannot be exploited directly to make valuable predictions.

The front of an evolving hazard can be approximated as a piecewise linear curve. Each segment of this curve (to be called the local front) can be adequately characterized by a small set of parameters, namely an orientation angle and the direction and speed of the segment’s propagation. In [13], [14] we have shown that the spatiotemporal evolution of each local front can be modeled by a modified 2D Gaussian function and that it is possible to track the front by updating the model parameters using a distributed processing scheme which solves a Kullback-Leibler divergence minimization problem.

In contrast with existing works [5]–[12] which try to delineate the area affected by the diffusive hazard, we present in this paper a novel decentralized algorithm which can estimate with accuracy, using dynamically formed clusters (triplets) of cooperating sensor nodes, the local evolution characteristics (orientation, direction and speed) of a continuous object. The updating of the evolution parameters is based on a Bayesian probabilistic modeling approach which relatively to our prior work ([13], [14]) : (i) Casts the problem in a framework allowing us to account for the sensing mechanism uncertainties expected in harsh environments and also characterize the uncertainty of the estimated parameters, (ii) Improves the accuracy of the obtained local model parameter estimates, (iii) Leads to simpler algebraic expressions for updating these parameters that can be easily implemented by the commonly used processing- and power-constraint embedded microprocessors of WSN nodes, (iv) Takes into account the possibility of imperfect sensor nodes which may fail to communicate since the approaching hazard may impair their functionality.

With respect to related work on predictive modeling ([11], [12]) our approach exhibits the following advantages: It can track accurately the time varying characteristics of a local front line even if, (i) the WSN is not dense, (ii) the sensor node clocks are not synchronized, (iii) the sensing mechanism is imperfect, (iv) nodes and communication links may fail as the hazard approaches. The ability to track the spatiotemporal evolution characteristics of the local front enables making predictions about its future location, a feature that is found only in hazard specific WSN schemes [15], [16].

The rest of the paper is organized as follows: In Section 2 we introduce preliminaries needed to understand the proposed modeling approach and the associated collaborative in-network processing algorithm. The Bayesian approach used to update the model parameters is presented in Section 3. In Section 4 we describe in detail the steps of the collaborative in-network algorithm. Extensive validation results are presented and discussed in Section 5. Finally, our findings are summarized and work in progress is outlined in Section 6.

2 Preliminaries

The key idea of the proposed in-network collaborative algorithm is the following: As soon as the deployed sensor nodes detect the evolving front line of a propagating hazard they are dynamically organized into ad-hoc local clusters of 3 nodes (*triplets*). Each triplet consists of a Master node (S_i^M) that initiates cluster formation and two Helper nodes $\{S_j^H, S_k^H\}$ that the Master selects among the nodes in its neighborhood and uses (without them knowing it!) to update its current (prior) local front evolution belief model in a Bayesian manner. The parameters of the updated (posterior) model are then propagated forward to other sensor nodes residing in the region that the evolving phenomenon is moving into.

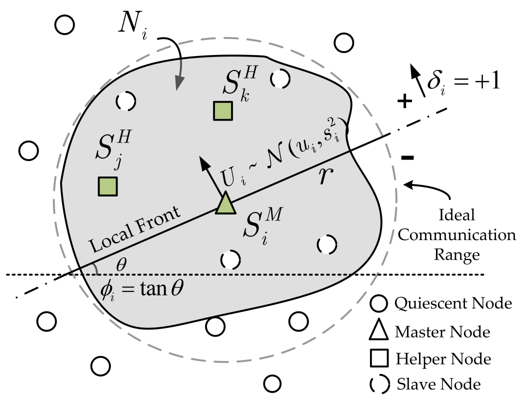

Fig. 1: The neighborhood (grey area) and local front model characteristics of sensor S_i^M.

2.1 Sensor Network Assumptions

We assume that sensor nodes are stationary and their locations are known. A sensor S_i can communicate directly only with nodes located within its neighborhood N_i that may change dynamically (grey shaded area in Figure 1) and is a subset of node S_i's ideal communication range (a disk of radius r). We have to emphasize that the proposed scheme has been designed to tolerate communication link failures, since in real WSN deployments we expect that factors such as physical obstacles, adverse local conditions created by propagating hazards etc., may affect the operation and/or communication capabilities of deployed sensor nodes. We assume that each sensor node is aware of: (i) its own location, (ii) the location of its neighbors, and (iii) a parametric model capturing its prior belief about the local front line's evolution characteristics (see Section 2.3 for details). In a valid WSN deployment, each sensor node is assumed to have at least two neighbors. The local clocks of sensor nodes *do not* need to be synchronized to a global clock reference. A sensor node may fail at any time due to the hazard's propagation. Once a node fails we assume that it cannot communicate with its neighbors. Sensor node failures may be either permanent or intermittent.

2.2 Sensor Node Status

A sensor node S_i (subscripts will be used to uniquely identify a sensor node when necessary) may assume one of the following statuses:

Quiescent (S_i^Q): Default and initial status.

Master (S_i^M): A node that has become responsible for updating the local front's model.

Master Candidate (S_i^C): This transitional state is entered while a node checks if it can satisfy the necessary conditions to become a Master (details are provided in Section 4).

A status transition will be denoted using the right arrow symbol ($\rightarrow$). e.g. $S_i^Q \rightarrow S_i^C$ denotes that the Quiescent node S_i^Q becomes a Master Candidate S_i^C.

Slave (S_i^L): A Slave node is responsible for monitoring the phenomenon upon receiving a request from a Master. A Slave may serve more than one Masters at the same time.

As we see in Figure 1 the neighborhood N_i of Master node S_i^M is partitioned into two half planes (positive and negative) by the local front line (a line segment of length equal to the diameter of S_i^M's ideal communication range). To refer to the neighbors of S_i^M located in the positive (negative) half plane we will use the notation N_i^+ (N_i^-) respectively. It holds that $N_i = N_i^+ \cup N_i^-$. Furthermore, N_i^0 is the subset of neighbors of S_i that have not detected the phenomenon yet. It holds that $N_i^0 = N_i^{+0} \cup N_i^{-0}$, where $N_i^{+0} \subseteq N_i^+$ ($N_i^{-0} \subseteq N_i^-$) are the subsets of neighbors that have not detected the phenomenon yet and are residing in the positive (negative) half plane respectively. Similarly we will use $N_i^H = \{S_j^H, S_k^H\}$ to refer to two special Slaves of S_i^M, called *Helpers*, that the Master S_i^M selects as assistants in the process of updating its local front model; S_j^H is assumed to be the first and S_k^H the second selected Helper of S_i^M. (Details on how the Helpers are selected among the Master's neighbors are provided in Section 4).

2.3 Local Front Models and Parameters

Each sensor node S_i uses a parametric model to represent its belief about the local front's evolution characteristics, namely its orientation, direction and speed. This model approximates the local front as an evolving line segment of length equal to the diameter (*2r*) of the sensor's circular communication range (see Figure 1).

For Master node S_i^M the *Prior Model* (before a model update) and the *Posterior model* (after a model update) will be denoted as $m_i = \{\phi_i, \delta_i, u_i, s_i\}$ and $m_i^* = \{\phi_i^*, \delta_i^*, u_i^*, s_i^*\}$ respectively, where:

- ϕ_i (*Orientation*): Is the tangent of the angle formed between the local front's line and the x-axis (see Figure 1).
- δ_i (*Direction*): It is assumed to be always perpendicular to the local front's line segment. The direction coefficient may take one of the following values: 0, if the evolution direction is unknown; $+1(-1)$, if the local front evolves into the positive (negative) neighborhood's half plane respectively (see Figure 1).
- u_i, s_i (*Speed model parameters*): The speed U_i of the local front's line segment is considered to be a random variable that follows a Normal distribution $\mathcal{N}(u_i, s_i^2)$. The source of stochasticity and the use of the Normal distribution are discussed in Section 3.

2.4 Sensor Node Information and Tables

Each sensor node S_i maintains locally the following information about itself:

Identity (ID_i): An integer that uniquely identifies S_i in the network.

Local Timer (t_i): It is started when S_i detects the phenomenon.

Location ($L_i = (x_i, y_i)$): The coordinates of the location of sensor node S_i.

Detection Status Flag (DSF_i): A Boolean flag; it is set, $DSF_i \leftarrow 1$, when the hazard is detected by node S_i.

Sensor Status (SS_i): A small integer encoding the current status of S_i. Possible values are {0: Quiescent, 1: Master Candidate, 2: Slave, 3: Master)}.

Prior Model (PM_i): The model $m_i = \{\phi_i, \delta_i, u_i, s_i\}$ which captures node's S_i current belief about the local front's evolution characteristics.

Updated Model (UM_i): The posterior model, after the Master S_i^M has updated the PM_i parameters, i.e. $m_i^* = \{\phi_i^*, \delta_i^*, u_i^*, s_i^*\}$.

A sensor node S_i organizes and stores locally the above information into the following tables:

Sensor Information Table (T_i^S): S_i keeps in this table the following information about itself: $\{ID_i, L_i, DSF_i, SS_i, PM_i\}$.

Neighborhood Table (T_i^N): S_i maintains in a separate row of this table information about each neighbor S_m, i.e. $\{ID_{im}, L_{im}, t_{im}, DSF_{im}\}\ \forall\{S_m \in N_i\}$, where t_{im} is the value of the local timer of S_i when it is notified that its neighbor S_m has also detected the phenomenon. If this notification arrives before the timer of S_i is initiated (i.e. if S_m detects the phenomenon before S_i) then the value of t_{im} is set to $null$. At deployment time, a sensor node S_i retrieves the IDs and location coordinates of its neighbors using the following simple procedure: S_i broadcasts a special message asking its neighbors to provide their IDs and location coordinates. When the neighbors receive this request they return their information which is stored in T_i^N.

The above tables are formed when a sensor node is initialized. In addition, each node creates dynamically and maintains the following tables:

Helpers Table (T_i^H): Created when a sensor node S_i^M becomes a Master, stores the IDs of legitimate pairs of neighbors which may potentially become Helpers. (How these legitimate pairs are selected and how this table is used is explained in Section 4).

Masters Status Table (T_m^M): Slave node S_m^L creates this table and stores in a separate row the following information about each Master (S_i^M) it serves: $\{ID_i, UM_i\}$. (How this table is used is explained in Section 4).

Hereafter, we will use the prefix "S" (e.g. Section S1, Figure S3, etc.) to refer to Sections, Figures and Tables that have been included in the Supplementary Material due to lack of space.

2.5 Sensor Messages

The proposed in-network algorithm assumes that each sensor node can handle the following messages (the attributes carried by each message are provided in parenthesis). (Details of the algorithm will be presented in Section 4).

TABLE OF SYMBOLS	
Symbols	**Definitions**
S_i	Sensor node i
$S_i^Q, S_i^M, S_i^L, S_i^C, S_i^H$	S_i has a *Quiescent*, *Master*, *Slave*, *Master Candidate*, and *Helper* status
$T_i^S, T_i^N, T_i^H, T_i^M$	*Information*, *Neighborhood*, *Helper* and *Master* Table of S_i
$L_i = (x_i, y_i)$	Location coordinates of S_i
t_i	Local timer of S_i
t_{im}	Local timer value of S_i when it is notified that its neighbor S_m has detected the phenomenon
N_i	The set of nodes that belongs to S_i's neighborhood
$N_i^+, (N_i^-)$	The set of neighbors of S_i that belong to the positive (negative) half plane
$N_i^{+0}, (N_i^{-0})$	The set of neighbors of S_i that belong to the positive (negative) half plane and have not detected the phenomenon yet
N_i^H	The set of neighbors of S_i that Help their Master to update its local front model
$m_i, (m_i^*)$	Prior (Posterior) local front model of node S_i
$\phi_i, (\phi_i^*)$	Prior (Posterior) local front's line gradient
$\delta_i, (\delta_i^*)$	Prior (Posterior) local front's line direction
$\{u_i, s_i\}, (\{u_i^*, s_i^*\})$	Prior (Posterior) local front's line speed parameters
R_d	Sensing model radius
μ_d, σ_d, α	Sensing model parameters
p_{ij}	The projection point of S_j location on the local front line of S_i
u_{ij}, s_{ij}	p_{ij} speed parameters
w_{ij}	Mixture weight of U_{ij}

TABLE 1: Table of Symbols

Broadcast type Messages:

Detection Message (`DM(ID_i)`): Broadcasted by sensor node S_i to notify its neighbors that it has detected the phenomenon (detection event).

Master Declaration Message (`MDM`): Broadcasted by a node to notify its neighbors that it satisfies the necessary conditions to become a Master. This message has two slightly different versions: `MDM1(ID_i, PM_i)` and `MDM2(ID_i)`.

Update Prior Message (`UPM(ID_i, UM_i)`): Broadcasted by Master node S_i^M after it has updated its prior model.

Free Slaves Message (`FSM(ID_i)`): Broadcasted by Master node S_i^M in order to release its Slaves.

Pass My Posterior Message (`PMPM(ID_i)`): Broadcasted by a Master node S_i^M when none of its Helpers satisfies the necessary conditions to become the new Master.

Pass Posterior Message (`PPM(UM_i)`): Broadcasted by the Slave nodes $\{S_m^L \in N_i\}$ of Master node S_i^M after they have received a `PMPM(ID_i)` message.

Unicast type Messages:

Master Declaration Message Acknowledgement (`MDMA(ID_j)`): Slave node (S_j^L) sends this message to notify its Master (S_i^M) that it has received its `MDM` message and has become its Slave.

Master Offer Message (`MOM(ID_i)`): Master node S_i^M sends this message to offer to one of its Helpers to become the new Master.

Accept Master Offer Message (`AMOM(ID_j)`): It is sent by a Helper (S_j^H) to a Master to acknowledge that it accepts the offer to become the new Master.

Decline Master Offer Message (`DMOM(ID_j)`): It is sent by a Helper (S_j^H) to a Master to notify it that it rejects the offer to become the new Master.

We summarize the symbols used in the text and their definitions in Table 1 to facilitate the reading of the paper.

3 Modeling Approach

3.1 Modeling Detection Distance Uncertainty

A commonly made assumption is that a sensor node can detect an event inside a disk area of radius R_d. Although this may not always hold in real applications, it is frequently adopted since it simplifies the analysis [17]–[22]. Many disk based sensing models have been proposed in the literature such as the binary, staircase, probabilistic, etc. [19]. Among the most popular is the probabilistic sensing model given below,

$$p(x) = \begin{cases} 1 & x \leq R_s \\ e^{\lambda(x-R_s)^\gamma} & R_s < x < R_d \\ 0 & x \geq R_d \end{cases} \quad (1)$$

where the probability of detection is exponentially decreasing with the distance x in the range $[R_s, R_d]$ and a sensor detects an event with certainty if it occurs within the inner circle of radius R_s (see Figure 2a). The parameters γ and λ in equation (1) control the rate of probability decrease and can be determined considering the physical properties of the sensor, the noise in sensor measurements, the characteristics of the sensed physical quantity etc. [18].

We introduce a variation of the probabilistic sensing model which, in addition to capturing the detection distance uncertainty, it also accounts for the real possibility that a sensor node may malfunction in a harsh environment as the hazard front is approaching. As for the probabilistic model, the sensing range of a node S_i is assumed to be a circular region of radius R_d (see dotted circle in Figure 2b) centered at the sensor's location (L_i). R_d is hazard specific and depends on: (i) The sensor's technical specifications (e.g. its sensitivity), (ii) how the hazard affects the physical quantity measured by the sensor. Using this information we can estimate the expected detection distance [23]. We set this expected value to $\frac{\alpha R_d}{2}$, where $0 \leq \alpha \leq 1$ (see Figure 2b). However, due to the stochastic nature of a hazard's detection process the real distance may deviate from this expected value. To account for this stochasticity we treat the detection distance as a normally distributed random variable, $D_i \sim \mathcal{N}(\mu_d, \sigma_d^2)$, with parameters:

$$\mu_d = \frac{\alpha R_d}{2}, \quad 3\sigma_d = R_d(1 - \frac{\alpha}{2}) \Rightarrow \sigma_d = \frac{R_d}{3}(1 - \frac{\alpha}{2}). \quad (2)$$

In setting the standard deviation as in (2) above we have assumed that the probability for a sensor to detect the approaching diffusive phenomenon at a distance larger than R_d is negligible.

As observed in Figure 2b the probability of detection increases monotonically as the distance of the local front to the sensor decreases in the range $\left[\frac{\alpha R_d}{2}, R_d\right]$. However,

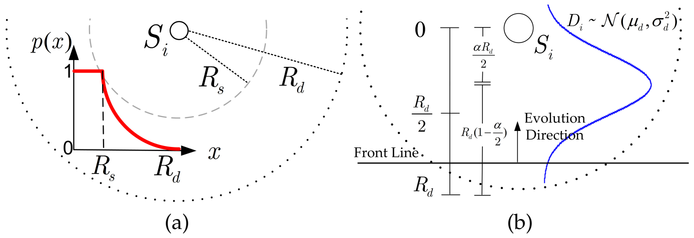

Fig. 2: Sensing Modeling: (a) Probabilistic exponential sensing model. (b) The proposed shifted Gaussian sensing model.

in close range $\left[0, \frac{\alpha R_d}{2}\right]$ the probability of detection decreases. This modeling decision is justified if we consider that the inability of a sensor to detect the approaching front at the expected detection range $\left[\frac{\alpha R_d}{2}, R_d\right]$ is an indication of a potential hazard-induced malfunction reducing the node's ability to detect the hazard as it gets closer. This simple and realistic sensing behavior model in the presence of propagating hazards allows us to capture both the stochasticity associated with the detection distance as well as the real possibility that a sensor node malfunctions as the hazard gets in close range. Importantly, it also does not harm the generality since by setting parameter $\alpha = 0$ in equation (2) (i.e. $\mu_d = 0$) we can always revert back to a monotonic traditional probabilistic sensing model centered at the sensor node's location. Therefore the proposed "shifted" Gaussian model offers flexibility since it can cover both scenarios: diffusive hazards whose presence may, or may not, disrupt the functionality of deployed sensor nodes.

We also remark that the Gaussian distribution has been used by many researchers to describe the dependence of sensor node detection probability to distance ([20]–[22]) since it has all the necessary ingredients to characterize the uncertainty while offering also a simple parameterization.

3.2 Modeling Speed Uncertainty

Let's now consider the cluster of three sensor nodes (triplet) shown in Figure 3. As soon as the Master node S_i^M receives two *Detection Messages* (DM), one message from each one of its two Helpers, $\{S_h^H$, where $h \in \{j,k\}\}$, it has all the information it needs to start updating its prior model.

Using the coordinates of its Helpers ($L_h = (x_h, y_h)$ stored in its table T_i^N) and the coordinates of their projection points $\{p_{ih} = (x_{ih}, y_{ih}),\ h \in \{j,k\}\}$ on its local front line, Master S_i^M can calculate the Euclidean distances $\{d_{ih},\ h \in \{j,k\}\}$ using equation (3) below (see also Figure 3),

$$d_{ih} \equiv dist(L_h, p_{ih}) = \sqrt{(x_h - x_{ih})^2 + (y_h - y_{ih})^2}. \quad (3)$$

Let's now call D_{ih} the distance that the local front at node S_i has to travel before it gets detected by a Helper node.

$$D_{ih} = d_{ih} - D_h, \quad h \in \{j,k\} \quad (4)$$

Since D_h, the detection distance of the progressing front from Helper S_h, $h \in \{j,k\}$, follows a Normal

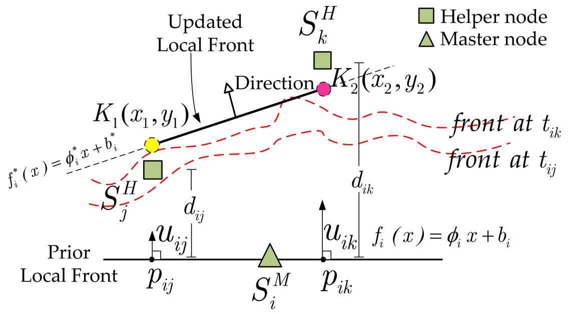

Fig. 3: Updating the local front model.

distribution $\mathcal{N}(\mu_d, \sigma_d^2)$, D_{ih} will also follow a Normal distribution $\mathcal{N}(\mu_{ih}, \sigma_{ih}^2)$ with parameters:

$$\mu_{ih} = d_{ih} - \mu_d = d_{ih} - \frac{\alpha R_d}{2}, \quad \sigma_{ih} = \sigma_d = \frac{R_d}{3}(1 - \frac{\alpha}{2}), \quad (5)$$

where d_{ih} has been computed using equation (3), $h \in \{j,k\}$.

Upon estimating the parameters μ_{ih} and $\sigma_{ih}, h \in \{j,k\}$ using equation (5), Master node S_i^M can calculate the speed at which the two Helper projection points, p_{ij} and p_{ik}, have to move forward in order to cover the distances D_{ij} and D_{ik} in the measured time intervals t_{ij} and t_{ik} respectively (see Figure 3). Since D_{ij} and D_{ik} are random variables that follow a Normal distribution, it can be shown that the corresponding speeds of the two projection points, U_{ij} and U_{ik}, will also follow Normal distributions of the form $U_{ih} \sim \mathcal{N}(u_{ih}, s_{ih}^2)$. Their parameters can be computed easily using equations (6) below, $h \in \{j,k\}$:

$$u_{ih} = \frac{\mu_{ih}}{t_{ih}} = \frac{2d_{ih} - \alpha R_d}{2t_{ih}}, \quad s_{ih} = \frac{\sigma_{ih}}{t_{ih}} = \frac{R_d(1 - \frac{\alpha}{2})}{3t_{ih}}. \quad (6)$$

3.3 Model Parameters Updating

3.3.1 *Speed*

The speed model is updated based on a sequential Bayes procedure which however has been designed to respect the limited processing capabilities and energy constraints of the WSN nodes. As in every Bayesian method, to compute the posterior model we need: (i) an assumption about the random variable's current "behavior" (prior model) and (ii) the likelihood of the observed data.

As discussed in Section 2.3 (see last bullet) the local front's speed is a Normal random variable. To update its parameters (mean and variance), Master node S_i^M uses its prior speed information $U_i \sim \mathcal{N}(u_i, s_i^2)$ (parameters are stored in its prior model m_i) as well as the likelihood computed using information related to the "observed" speeds (U_{ih}) of the two Helper node projection points on the current local front, namely $\{p_{ih}$, where $h \in \{j,k\}\}$.

Since the number of the available "observations" is very small (only two), we introduce below a technique which exploits the availability of information about the uncertainty (s_{ih}) associated with the speed "observations" (U_{ih}, see Figure 4a) to improve the likelihood estimation accuracy.

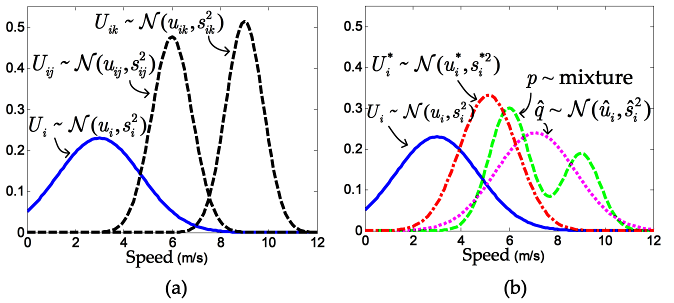

Fig. 4: Speed model updating procedure. (a) The prior model (U_i), and the two "observation" models (U_{ij}, U_{ik}). (b) The mixture model p resulting by combining speed "observation" models; the normal distribution $\hat{q}$ that best approximates p by minimizing the Kullback Lebier divergence ($KL(p||\hat{q})$); the resulting posterior speed model U_i^*.

Having available its local prior model and the parameters of the speed models $U_{ih} \sim \mathcal{N}(u_{ih}, s_{ih}^2)$ (computed using the equations in (6)), the Master node S_i^M computes the weights $\{w_{ih},\ h \in \{j,k\}\}$ of a Gaussian mixture model with two components (see Figure 4b)

$$p(u) = \sum_{h \in \{j,k\}} w_{ih} \mathcal{N}(u | u_{ih}, s_{ih}^2), \tag{7}$$

as follows:

$$w_{ij} = \frac{1}{1+C}, \quad w_{ik} = \frac{C}{1+C}, \quad C = \frac{s_{ij}|u_i - u_{ij}|}{s_{ik}|u_i - u_{ik}|}. \tag{8}$$

Fixing the mixture weights as in (8) is justified based on the following arguments:

- The speed model $U_{ih}, h \in \{j,k\}$ with the smaller standard deviation (smaller uncertainty) should be trusted more by the Master.
- Since in short time periods (e.g. the time interval between two successive local model updates) environmental diffusive phenomena tend to exhibit smooth changes in terms of their evolution characteristics (speed and direction), more trust should be assigned to the speed "observation" with mean value (u_{ih}) closer to that of the prior model (u_i).

Estimating the posterior parameters by using directly the Gaussian mixture likelihood (7) and Bayes rule would be computationally expensive since analytical closed form expressions cannot be derived. Due to the limited processing capabilities and low power constraints of microprocessors used in WSN nodes, in this work we consider prohibitive the use of an iterative, slowly converging, parameters estimation procedure. Therefore, in order to be able to derive closed form algebraic expressions for the posterior distribution parameters we employ variational calculus and approximate the Gaussian mixture by a Normal distribution. To this end, we estimate the parameters of the Normal distribution that minimizes the Kullback-Leibler (KL) divergence (maximizes the similarity) from the Gaussian mixture. The general form of the equations which can be used to compute the parameters of this Normal distribution are [24], [25]:

$$\hat{\mu} = \sum_n w_n \mu_n \tag{9}$$

$$\hat{\Sigma} = \sum_n w_n (\Sigma_n + (\mu_n - \hat{\mu})(\mu_n - \hat{\mu})^T) \tag{10}$$

In our specific case these equations reduce to:

$$\hat{u}_i = w_{ij} u_{ij} + w_{ik} u_{ik} \tag{11}$$

$$\hat{s}_i^2 = w_{ij} s_{ij}^2 + w_{ik} s_{ik}^2 + w_{ij} w_{ik} (u_{ij} - u_{ik})^2 \tag{12}$$

Having computed the mixture weights using (8), Master S_i^M calculates the Normal distribution parameters $\hat{u}_i$ and $\hat{s}_i^2$ using equations (11) and (12). By applying simple manipulations on the Bayes theorem it can be proved [26] that since the prior $\mathcal{N}(u_i, s_i^2)$ and the likelihood $\mathcal{N}(\hat{u}_i, \hat{s}_i^2)$ are both Gaussian, the posterior will also be a Gaussian $\mathcal{N}(u_i^*, s_i^{*2})$ (conjugate distributions, see Figure 4b) with parameters provided by the following easy to compute closed form expressions:

$$u_i^* = \frac{u_i \hat{s}_i^2 + \hat{u}_i s_i^2}{\hat{s}_i^2 + s_i^2}, \quad s_i^{*2} = \frac{\hat{s}_i^2 s_i^2}{\hat{s}_i^2 + s_i^2} \tag{13}$$

3.3.2 Orientation

Let K_1 (K_2) be the point to be reached by p_{ij} (p_{ik}) as it moves in the direction of the local front's evolution with speed u_{ij}, (u_{ik}) respectively for a time interval t_{ik} (see Figure 3). The coordinates of K_1 and K_2, to be called (x_1, y_1) and (x_2, y_2), can be found by solving a system of a linear and a quadratic equation. This problem is formulated and solved analytically in Section S4. Using the calculated coordinates, Master S_i^M can update the orientation parameter of its local front model using equation (14) below,

$$\phi_i^* = \frac{y_2 - y_1}{x_2 - x_1}. \tag{14}$$

3.3.3 Evolution Direction

To update the direction parameter δ_i^*, node S_i^M derives the equation of line $f_i^*(x)$ that is defined by points $K_1(x_1, y_1)$ and $K_2(x_2, y_2)$ (see Figure 3).

$$f_i^*(x) = \phi_i^* x + b_i^* \tag{15}$$

where $b_i^* = y_1 - \phi_i^* x_1$

Subsequently, node S_i^M substitutes its abscissa (x_i) in (15) and checks the $sgn(f_i^*(x_i))$. If $sgn(f_i^*(x_i)) > 0$ ($sgn(f_i^*(x_i) < 0$) then Master node S_i^M infers that the new local front line evolves into the *negative (positive)* half plane and it updates the direction parameter $\delta_i^* = -1(1)$ accordingly.

4 In Network Collaborative Algorithm

4.1 Sensor Network Assumptions

To better explain the proposed in-network algorithm we will use a running example that facilitates the understanding of its operations. Let's assume, without loss of generality (*w.l.o.g.*) that a part of the evolving front has just entered the WSN's deployment region and none of the sensor nodes, (which are currently in the default Quiescent status), has detected the phenomenon yet (DSF = 0). Each node is equipped with sensors that can measure physical parameters affected by the hazard's presence when it enters its sensing range (a circle of radius R_d). All nodes are initialized with the same prior model $m_i = \{\phi_i, \delta_i, u_i, s_i\}$ and $\delta_i = 0$.

4.1.1 Forming a Local Cluster

Let's now assume that the evolving front enters the sensing range of node S_i (see Figure 5a). As soon as S_i detects the front, it initiates the *Detection Procedure* described below and also summarized by the UML sequence diagram of Figure 6.

Detection Procedure: Sensor node S_i starts a local timer, changes the value of its detection status flag DSF_i from 0 to 1 (stored in its information table T_i^S) and checks its status variable SS_i, which may have value 0 (Quiescent) or 2 (Slave).

- If $SS_i = 0$ (S_i is Quiescent, as in the example's case) the node makes a status transition, $S_i^Q \rightarrow S_i^C$ ($SS_i \leftarrow 1$), and initiates the *Master Check Necessary Conditions Procedure* (presented in the next paragraph).
- If $SS_i = 2$ (S_i is a Slave), node S_i^L broadcasts a *Detection Message* $\mathtt{DM(ID_i)}$. Each neighbor $\{S_m \in N_i,$ where $m = \{j, k, l\}$ in Figure 5a$\}$ when it receives this message it updates in its neighborhood table T_m^N (in the row corresponding to S_i) the attributes t_{mi} and DSF_{mi} (see Figure 6). The value assigned to t_{mi} is the time value t_m indicated by the local timer of S_m when message $\mathtt{DM(ID_i)}$ was received. If S_m receives message $\mathtt{DM(ID_i)}$ before its local timer has been started (this can happen if S_m has not sensed the phenomenon yet) it assigns to attribute t_{mi} a *null* value.

Master Check Necessary Conditions Procedure: Sensor node S_i^C finds in its table T_i^N the subset of neighbors which have not detected the phenomenon yet (i.e. $\{S_m \in N_i^0\}$). Based on the cardinality $|N_i^0|$ of this set, S_i^C proceeds as follows: (see UML sequence diagram in Figure 7).

- *If* $|N_i^0| < 2$: Node S_i^C transitions back to the Quiescent state, $S_i^C \rightarrow S_i^Q$, and broadcasts a $\mathtt{DM(ID_i)}$ message. Each receiving neighbor $\{S_m \in N_i\}$ updates its attributes t_{mi} and DSF_{mi} in its table T_m^N, in the way already discussed in the *Detection Procedure* paragraph and shown in Figure 6.
- *If* $|N_i^0| \geq 2$ (example's case): Node S_i^C initiates the *Local Front Line Derivation* activity and then checks the evolution direction parameter δ_i of its initial model m_i.

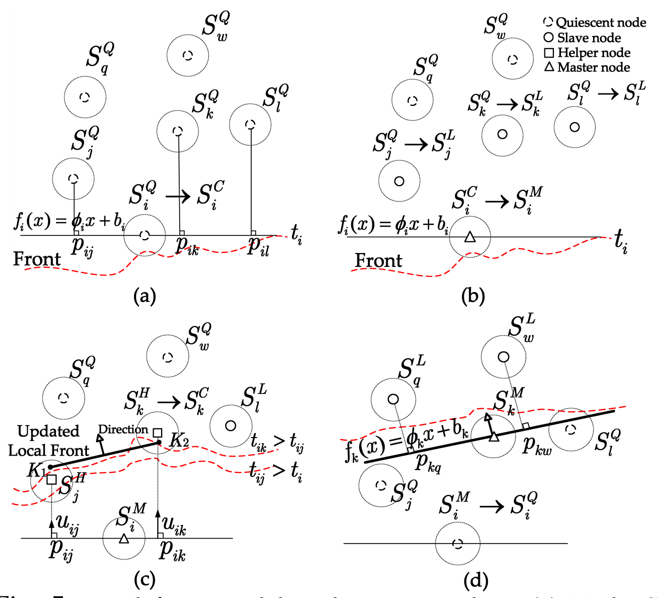

Fig. 5: Local front model updating procedure: (a) Node S_i becomes Master candidate and checks if it satisfies the conditions to become a Master, (b) node S_i becomes a Master and "enslaves" its neighbors S_j, S_k and S_l, (c) Master S_i^M uses the information received from its two Helpers (S_j^H and S_k^H) and updates the local front's line parameters, (d) node S_k becomes the new Master and S_i releases its slaves.

Based on the value of δ_i, S_i^C initiates the appropriate *Create Helpers Table* activity followed by the *Master Declaration* activity (see Figure 7).

Local Front Line Derivation: Node S_i^C uses the *orientation* parameter ϕ_i of its initial model m_i and its location information $L_i = (x_i, y_i)$, to derive the equation of the line where the local front segment belongs, $f_i(x) = \phi_i x + b_i$, where $b_i = y_i - \phi_i x_i$ (see Figure 5a).

Create Helpers Table: Node S_i^C checks the value of the front evolution direction parameter δ_i in m_i.

- If $\delta_i = +1$ (the local front evolves into the positive half plane), S_i^C searches in its neighborhood table T_i^N to find the subset of neighbors that belong to the positive neighborhood half plane and have not detected the phenomenon yet (N_i^{+0}). If $|N_i^{+0}| \geq 2$, S_i^C calculates the coordinates of these neighbors' projections on the local front line (see Section S4 for details). These are the points p_{ij}, p_{ik}, p_{il} in the example (Figure 5a). Then S_i^C calculates the Euclidean distances among all possible projection pairs and identifies those pairs with distances larger than a pre-specified threshold (that is application dependent). These pairs are considered to be the *legitimate Helpers pairs*, in the sense that any one of them could be used by S_i^C to update its prior model. Finally S_i^C stores the IDs of nodes of legitimate Helper pairs in its respective Helpers table T_i^H. How a particular Helpers pair is selected among the legitimate ones will be discussed in Section 4.2.
- If $\delta_i = -1$ (the local front evolves into the negative half plane), S_i^C performs the same aforementioned steps but for the nodes which belong to the negative neighborhood half plane and have not detected the phenomenon

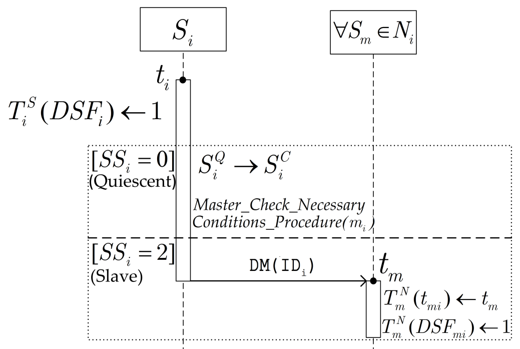
S_i
$\forall S_m \in N_i$
t_i
$T_i^S(DSF_i) \leftarrow 1$
$[SS_i = 0]$ (Quiescent)
$S_i^Q \rightarrow S_i^C$
Master_Check_Necessary Conditions_Procedure(m_i)
$[SS_i = 2]$ (Slave)
DM(ID_i)
t_m
$T_m^N(t_{mi}) \leftarrow t_m$
$T_m^N(DSF_{mi}) \leftarrow 1$

Fig. 6: *Detection Procedure* (UML sequence diagram).

yet (subset N_i^{-0}).

- If $\delta_i = 0$ (the local front's evolution direction is unknown - example's case), S_i^C searches in table T_i^N and finds the sensor nodes that belong to its neighborhood and have not detected the phenomenon yet (N_i^0). Then S_i^C partitions them into two half planes defined according to the local front line $f_i(x)$ (see *Local Front Line Derivation*). For each subset (N_i^{+0} and N_i^{-0}) S_i^C performs the steps described above for the cases $\delta_i = +1$ and $\delta_i = -1$ respectively.

Master Declaration: After the end of the *Create Helpers Table* activity node S_i^C checks its Helpers Table T_i^H:
If $T_i^H = \oslash$: S_i^C does not become a Master, transitions back, $S_i^C \rightarrow S_i^Q$, and broadcasts a $\mathtt{DM(ID_i)}$. The receiving neighbors $\{S_m \in N_i\}$ update their attributes t_{mi} and DSF_{mi} in their tables T_m^N (see Figure 8).
If $T_i^H \neq \oslash$ (i.e. there exist at least one pair of legitimate Helpers for S_i^C): Node S_i^C becomes a Master, $S_i^C \rightarrow S_i^M$, and checks its model's evolution direction parameter (δ_i):

- If $\delta_i \neq 0$, S_i^M broadcast a *Master Declaration Message of type 1*, ($\mathtt{MDM1(ID_i,PM_i)}$). Each neighbor $\{S_m \in N_i\}$ when it receives this message it updates the attributes t_{mi} and DSF_{mi} in the corresponding row of its table T_m^N, as shown by the UML sequence diagram of Figure 8. Moreover, each $\{S_m \in N_i^0\}$ uses the Master's S_i^M initial model (PM_i is carried in MDM1), derives the equation of the local front line $f_i(x)$ and substitutes its abscissa (x_m) in the argument of $f_i(x)$. If $sgn(f_i(x_m)) = sgn(\delta_i)$ (i.e. if S_m belongs to the half plane, with respect to the Master, that the front evolves into), S_m becomes a slave, $S_m \rightarrow S_m^L$, adds a row in its Masters table T_m^M for Master S_i^M with attributes $\{ID \leftarrow ID_i, UM \leftarrow null\}$ and sends a *Master Declaration Message Acknowledgement* ($\mathtt{MDMA(ID_m)}$) back to node S_i^M. Otherwise S_m keeps its status unchanged.
- If $\delta_i = 0$ (example's case), Master S_i^M broadcasts a *Master Declaration Message of type 2* ($\mathtt{MDM2(ID_i)}$). Each neighbor $\{S_m \in N_i\}$ receiving this message updates the attributes t_{mi} and DSF_{mi} in the corresponding row of its table T_m^N (see Figure 8). Moreover, if $\{S_m \in N_i^0\}$ it becomes a slave ($S_m \rightarrow S_m^L$) (see Figure 5b) adds a row in his T_m^M for Master S_i^M with attributes $\{ID \leftarrow ID_i, UM \leftarrow null\}$ and sends a *Master Declaration Message Acknowledgement* ($\mathtt{MDMA(ID_m)}$) to node S_i^M.

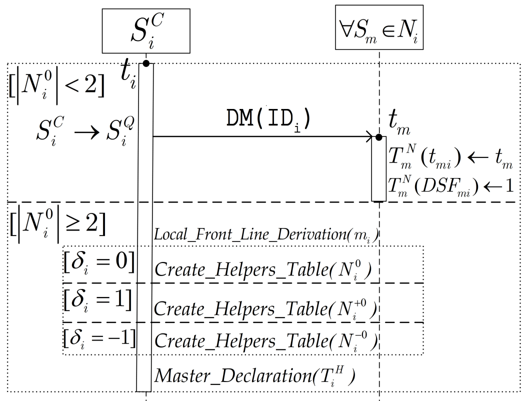
S_i^C
$\forall S_m \in N_i$
$[|N_i^0| < 2]$
t_i
$S_i^C \rightarrow S_i^Q$
DM(ID_i)
t_m
$T_m^N(t_{mi}) \leftarrow t_m$
$T_m^N(DSF_{mi}) \leftarrow 1$
$[|N_i^0| \geq 2]$
Local_Front_Line_Derivation(m_i)
$[\delta_i = 0]$
Create_Helpers_Table(N_i^0)
$[\delta_i = 1]$
Create_Helpers_Table(N_i^{+0})
$[\delta_i = -1]$
Create_Helpers_Table(N_i^{-0})
Master_Declaration(T_i^H)

Fig. 7: *Master Check Necessary Conditions Procedure* (UML sequence diagram).

When node S_i^C becomes a Master it waits until it receives two detection messages (DMs), one message from each one of the two nodes of a legitimate Helpers pair (among those pairs stored in its local Helpers Table T_i^H). However, the potentially adverse conditions created by the propagation of a diffusive hazard may impair the communication between the Master and its Helpers. In the a worst case scenario the Master may never receive the DM messages sent by its Helpers and thus never get the chance to update its local model parameters (formation of a "zombie" cluster). We should emphasize that the possible formation of "zombie" clusters does not affect the global functionality of the algorithm since the model updates within "healthy" clusters will normally occur (see Section 5.3 for details). Nevertheless, to reduce the probability of a "zombie" cluster formation the Master node may implement the following procedure:

Master Neighbourhood redefinition: Master node S_i^M waits (for an application dependent time interval) to receive the MDMAs from its Slaves. After this time, it checks if the received Slave IDs (contained in the received MDMAs) correspond to at least one of its legitimate Helper pairs (stored in its Helpers Table (T_i^H)):

- If they do, Master node S_i^M keeps its status unchanged and waits until it receives the two detection messages (DMs) that will be used to update its model parameters (see Section 4.2).
- If they do not, Master node S_i^M broadcasts a *Free Slaves Message* $\mathtt{FSM(ID_i)}$ and changes its status ($S_i^M \rightarrow S_i^Q$). When the slaves $\{S_m^L \in N_i\}$ receive the FSM message, they remove from their tables T_m^M the information corresponding to Master S_i^M and if they do not serve any other Master(s), they change their status back to Quiescent ($S_m^L \rightarrow S_m^Q$).

4.2 Model Updating

In our example we assume *w.l.o.g.* that the two messages received by S_i^M come from $N_i^H = \{S_j^H, S_k^H\}$ (see Figure 5c). Furthermore it is assumed that the two Helpers have detected the evolving front at time instances t_{ij} and t_{ik} respectively, where *w.l.o.g.* $t_{ij} < t_{ik}$. When the Master S_i^M receives the DMs from the pair of Helpers it updates

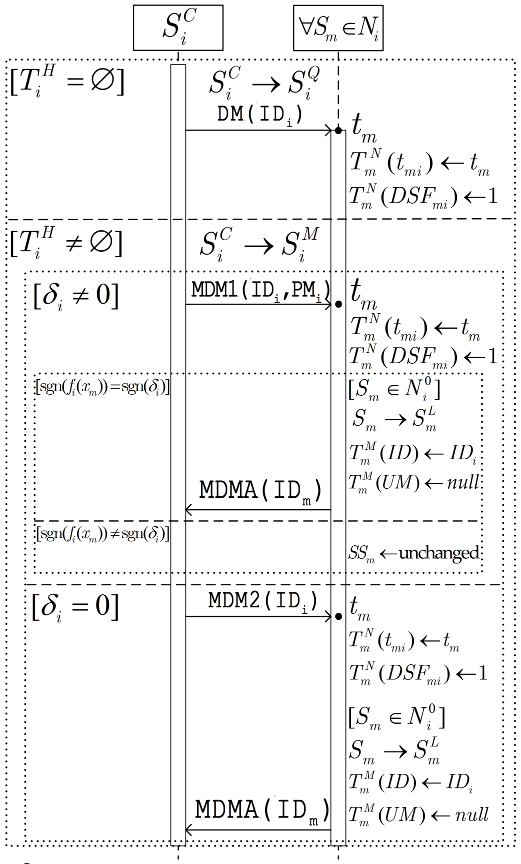
S_i^C
$\forall S_m \in N_i$
$[T_i^H = \varnothing]$
$S_i^C \to S_i^Q$
DM(ID_i)
t_m
$T_m^N(t_{mi}) \leftarrow t_m$
$T_m^N(DSF_{mi}) \leftarrow 1$
$[T_i^H \neq \varnothing]$
$S_i^C \to S_i^M$
$[\delta_i \neq 0]$
MDM1(ID_i,PM_i)
$[\mathrm{sgn}(f_i(x_m)) = \mathrm{sgn}(\delta_i)]$
$[S_m \in N_i^0]$
$S_m \to S_m^L$
$T_m^M(ID) \leftarrow ID_i$
$T_m^M(UM) \leftarrow null$
MDMA(ID_m)
$[\mathrm{sgn}(f_i(x_m)) \neq \mathrm{sgn}(\delta_i)]$
$SS_m \leftarrow$ unchanged
$[\delta_i = 0]$
MDM2(ID_i)

Fig. 8: *Master's Declaration* (UML sequence diagram).

its neighborhood table T_i^N and initiates the procedure described below.

Model Updating Procedure: The updating starts with the calculation of the "new" (posterior) local front speed model parameters ($U_i^* \sim \mathcal{N}(u_i^*, s_i^{*2})$). Master node S_i^M uses the expressions in (6) and calculates the parameters of the Normal speed models of the two Helper projection points p_{ij} and p_{ik} (see Section 3.2). By substituting these parameter values in (8), S_i^M calculates the Gaussian mixture weights w_{ij} and w_{ik} (see Section 3.3.1). Then, by applying the resulting mixture weight values into (11) and (12) the Master calculates the parameters ($\hat{u}_i$ and $\hat{s}_i$) of the Normal distribution that best approximates the Gaussian mixture. Finally, having available these parameters ($\hat{u}_i$ and $\hat{s}_i$), along with the prior model parameters (u_i, and s_i), S_i^M applies them to equation (13) to obtain parameters (u_i^*, s_i^{*2}) of the posterior speed model.

Next, Master S_i^M estimates the local front's orientation, ϕ_i^*. As discussed in Section 3.3.2 to update this parameter the Master finds the coordinates of two points, $K_1 = (x_1, y_1)$ and $K_2 = (x_2, y_2)$ (see Section S4 for details), which are expected to lie on the "new" local front line (see Figure 5c), and applies them directly to equation (14). Finally, S_i^M follows the procedure described in Section 3.3.3 and updates the evolution direction parameter, δ_i^*. All model parameters are updated using closed form expressions that can be realized easily

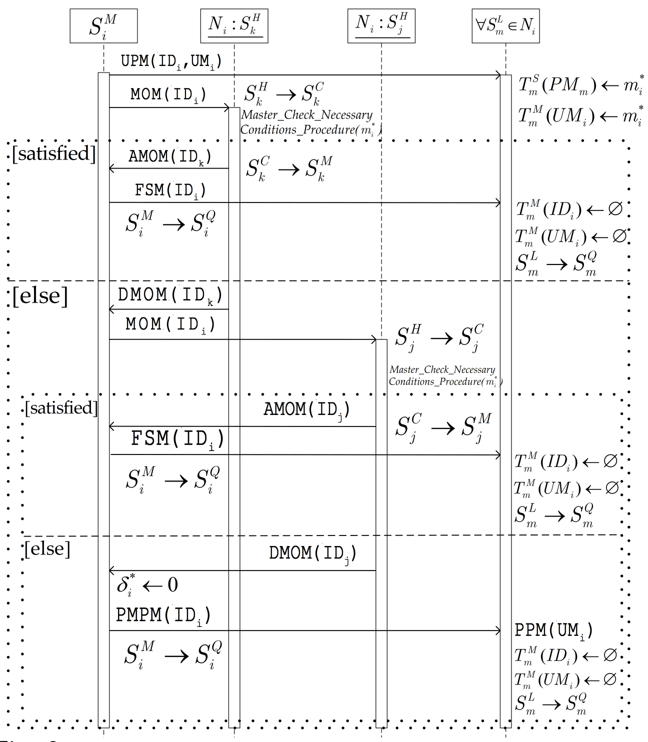

Fig. 9: *Model Propagation Procedure* (UML sequence diagram).

by embedded microprocessors commonly used in WSN node architectures.

4.3 Model Propagation

After updating its model, Master S_i^M initiates the *Model Propagation Procedure* (see the UML sequence diagram of Figure 9).

Master node S_i^M first broadcasts an *Update Prior Message* ($\mathtt{UPM(ID_i, UM_i)}$). The sensors which serve it $\{S_m^L \in N_i\}$ when they receive this message update the prior model information in their tables T_m^S using the received model m_i^*. Moreover, they update attribute $UM_i \leftarrow m_i^*$ in the corresponding row of their Masters' table T_m^M. In addition, S_i^M sends a *Master Offer Message* ($\mathtt{MOM(ID_i)}$) to the Helper that detected most recently the phenomenon (it is S_k^H *w.l.o.g.* in the running example of Figure 5c) and asks it to become the new Master. This sensor node becomes temporarily a Master candidate ($S_k^H \to S_k^C$) and uses the updated model parameters (m_i^*) to initiate the *Master Check Necessary Conditions procedure* (see Section 4.1.1).

If Helper S_k^C meets the conditions to become the new Master (as in the example's case), it accepts the offer ($S_k^C \to S_k^M$) and replies to S_i^M with an *Accept Master Offer Message* ($\mathtt{AMOM(ID_k)}$). When S_i^M receives this message it broadcasts a *Free Slaves Message* $\mathtt{FSM(ID_i)}$ and changes its status back to default ($S_i^M \to S_i^Q$, see Figure 5d). Each Slave S_m^L, when it receives the FSM removes from its table T_m^M the information corresponding to Master S_i^M and if it does not serve any other Master(s), it changes its status back to Quiescent ($S_m^L \to S_m^Q$).

On the other hand, if Helper S_k^C does not satisfy the necessary conditions to become the new Master, it rejects the offer made by S_i^M by replying with a *Decline Master Offer Message* (`DMOM(ID_k)`). This forces S_i^M to try exactly the same negotiation with its second Helper S_j^H. If S_j^H also rejects its offer, S_i^M gives up with its Helpers, resets in its updated model m_i^* the value of the evolution direction ($\delta_i^* \leftarrow 0$), broadcasts a *Pass My Posterior Message* (`PMPM(ID_i)`) and returns to Quiescent status. The neighbors ($\{S_m \in N_i\}$) enslaved to S_i^M, when receiving `PMPM` broadcast a *Pass Posterior Message* (`PPM(UM_i)`) containing the Master's S_i^M updated model m_i^*. The neighbors of the nodes S_m when receiving the `PPM` they replace their prior model in their table T^S with the updated model m_i^*. Finally each neighbor $\{S_m^L \in N_i\}$ deletes from its Masters table T_m^M the information related to S_i^M and if it does not serve another Master it changes its status back to Quiescent ($S_m^L \rightarrow S_m^Q$).

5 Evaluation of the Algorithm

We present next simulation results demonstrating the ability of the proposed collaborative WSN algorithm to estimate accurately the local evolution characteristics (speed and direction) of a continuous object. The phenomenon may include multiple diffusion processes (hazards), possibly expanding at a time varying rate and/or assuming irregular shapes.

5.1 WSN Simulation Workflow

For the evaluation we developed a flexible simulation workflow which allows us to generate and execute realistic WSN simulation scenarios with different sensor node densities, deployment strategies, sensor node failure probabilities, communication (Rx and Tx) failure probabilities, and propagating hazard front properties (shape, speed and acceleration).

The WSN simulation workflow has two main components: i) The flexible WSN simulator COOJA (COntiki Os JAva) [27] for the Contiki sensor node operating system, and ii) a Matlab-based component which prepares the COOJA input file and at the end evaluates the estimation accuracy of the proposed in-network algorithm. As shown in Figure 10, the Matlab component takes as input information about: a) the deployed sensor nodes (location, prior model parameters, etc.), and b) the propagating hazard's front properties, and determines the sequence in which the deployed sensor nodes detect the evolving hazard. After that, it generates a file (*Detection Events Sequence*) which contains for each sensor node the following information: {*ID, location, time of detection, prior model parameters*}. This file is passed as input to COOJA that simulates the proposed distributed algorithm as if it was implemented by a WSN consisting of Atmel's AVR RAVEN nodes [28]. To achieve this, the code every sensor node needs to run to implement the proposed in-network algorithm was programmed in C on the Real Time Operating System (RTOS) Contiki. Using COOJA we simulate the IEEE 802.15.4 MAC protocol's byte stream (preamble,

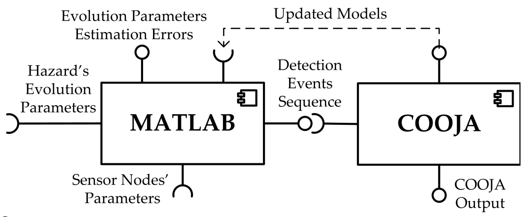

Fig. 10: UML component diagram of Matlab-COOJA based simulation workflow.

start of frame delimiter, data, and checksum) which is also used by the Atmel's AVR Raven nodes. Moreover using COOJA's Unit Disk Graph Medium (UDGM) with a distance loss propagation model [29] (that considers interferences), we can evaluate the proposed algorithm's behavior under different Rx/Tx failure probabilities.

At the end of a simulation, a *COOJA Output* file is produced containing: a) The updated model parameters, b) the number of Rx and Tx messages/Bytes exchanged in the WSN, and c) the energy consumed for communication (Rx and Tx). To evaluate the estimation accuracy of the proposed algorithm, the updated models information is passed back to the Matlab component which compares the estimated orientation and speed with the ground truth values (see Sections S1.2 and S2.2).

5.2 Experimental Setup

A notable advantage of the proposed in-network processing algorithm is that it can estimate accurately the evolution characteristics of a local front using low density WSNs. Specifically, in our experiments we used densities $7.5 \times 10^{-5}, 10^{-4}, 1.25 \times 10^{-4}$ $sensors/m^2$, which correspond to 75, 100 and 125 sensor nodes respectively deployed within an $1km^2$, which are considered low for environmental monitoring applications. For each network density case we used a large number of randomly drawn WSN deployments and investigated how the proposed algorithm performs under different sensor node failure and communication (Rx and Tx) failure probabilities (equal to 0, 0.1, 0.2 and 0.3).

In all experiments, the radius of the nodes' communication range was set to $r = 150m$ to guarantee that we have a connected network (every node has at least one neighbor) for every density scenario. Furthermore, in order to evaluate how the radius of the sensing range R_d affects the accuracy of the algorithm we repeated the experiments with small and large R_d values (equal to 0.1m and 15m). For the sensing models the parameters were calculated using the equations in (2) for $\alpha = 1$. Each sensor node was initialized with the same prior model, $m_i = \{\phi_i = 0, \delta_i = 0, u_i = 5m/min, s_i = 2m/min\}$. The mean speed value of the prior model was intentionally chosen to differ significantly from the simulated hazard front speeds in order to demonstrate the ability of the proposed distributed algorithm to estimate the true model parameter values even when the initial prior belief model of the sensor nodes deviates significantly from the reality. The communication energy consumed

Experiment 1						
Sensing Radius ($R_d = 0.1m$)						
$P(f)$	75 nodes		100 nodes		125 nodes	
	Orient.	Speed%	Orient.	Speed%	Orient.	Speed%
0	4.69/10.78	13.04/13.2	4.22/10.31	12.43/13.9	3.97/9.89	12.15/13.19
0.1	4.91/10.32	13.29/12.91	4.31/10.17	12.91/14.11	4.04 /10.03	13.01/13.41
0.2	5.01/10.2	12.97/13.13	4.63/10.82	13.22/13.67	3.86/10.41	12.88/13.01
0.3	5.23/10.97	13.53/13.66	5.08/10.74	13.09/14.42	4.24/10.19	12.92/13.38
Sensing Radius ($R_d = 15m$)						
0	5.21/11.12	13.92/14.25	4.67/10.3	13.54/13.66	4.59/10.25	13.59/13.97
0.1	4.99/10.83	14.07/14.88	4.54/10.88	14.81/13.31	4.63/10.16	13.82/14.24
0.2	5.07/10.74	13.96/14.91	5.19/10.46	13.27/14.82	5.11/10.72	13.64/14.4
0.3	4.86/10.89	13.71/14.7	5.03/10.97	14.35/14.69	4.97/10.58	14.01/14.82

TABLE 2: Experiment 1 results summary: The Median/Inter Quartile Range of the orientation error (in degrees) and percent speed error under different network density, node failure probability, and sensing radii conditions. For each entry the reported statistics were computed based on 200 simulation runs (50 random WSN deployments x the 4 Rx/Tx failure probability cases considered).

by the simulated AVR Raven nodes was measured using: a) their maximum power (3dBm) for transmission (at this power level the communication range of the AVR RAVEN nodes is approximately 150m) and b) reception sensitivity -101dBm (fixed). Finally, we used the policy that a sensor node retransmits a message only once if it does not receive an acknowledgement from its recipient(s).

5.3 Results and Discussion

In the conducted experiments the diffusive phenomenon (continuous object) was simulated using either a Matlab program or FLogA a wildfires behavior simulator developed in our group [30]. To evaluate the accuracy of the proposed distributed algorithm, we compared the estimated direction and speed of the local fronts to the corresponding ground truth values. A detailed description of the evaluation metrics used is provided in Sections S1.2 and S2.2.

5.3.1 Experiment 1: Multi-source diffusive hazards

In the first experiment a complex diffusive phenomenon is modeled as two circles of fixed centers and radii increasing with equal but time varying rates. The circles represent two distinct diffusive hazards which start entering the WSN deployment area at the beginning of the simulation. As the circles grow, they start to overlap and form a complex front line before they cover half of the deployment area. In order to help the reader visualize the complex phenomenon and get a sense of the model updates as they take place during its propagation, we provide a video animation (created using Matlab) as Supplementary Material (see file Experiment1TwoFronts.mp4 [31]). A discussion of the video is also provided in Section S1.1.

Modeling propagating hazards with circular shapes is justified because: a) Fick's second law indicates that the diffusion of a substance emanating from a single point source covers a circular area whose size is increasing at a rate indicated by the diffusion coefficient [32]. b) Moreover, the circle's properties allow us to evaluate analytically the speed and direction estimation errors (see Section S1.2).

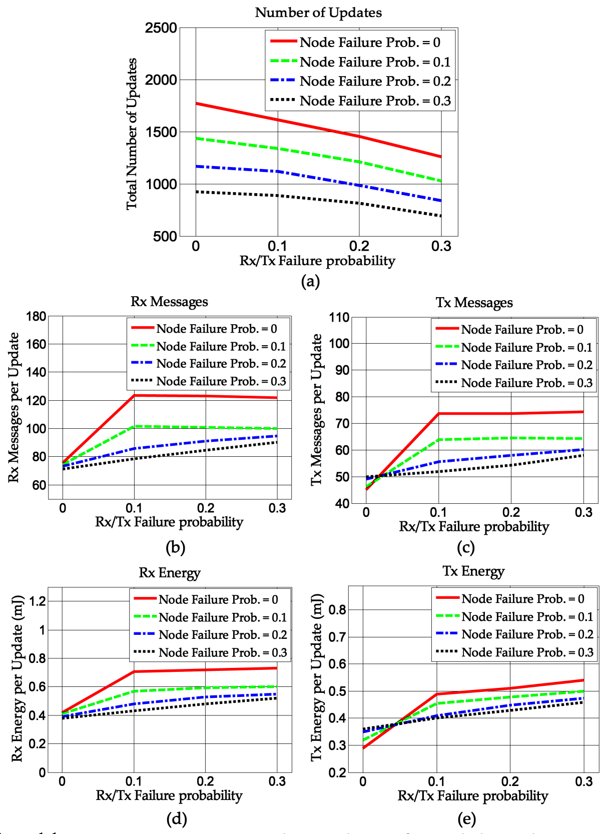

Fig. 11: Experiment 1: Total number of model updates, mean number of messages and Rx/Tx energy consumed per model update, for each node failure and Rx/Tx failure probability considered (density = 100 sensor nodes per km^2).

As observed from Table 2, the parameters estimation accuracy of the proposed algorithm is rather insensitive to changes in sensor nodes density, node failure probability, and Rx/Tx failure probability. This was also confirmed by comparing pairwise the means of error densities using Student's t-test. For all cases the difference of the means was found to be insignificant at the 0.05 significance level. Before applying the t-tests we verified that the usual assumptions (normality, variance homogeneity and independence) hold for all datasets compared. Moreover, the results indicate that the accuracy slightly decreases when the sensing radius R_d increases. This is so because an increase of the sensing radius implies increasing the uncertainty associated with the front line's location at the time of the hazard's detection. This in turn implies increasing the uncertainty regarding the estimated mean speed values u_{ih}, $h \in \{j, k\}$ used to estimate the local front's orientation and speed (see Section 3.2 for details).

In Figure 11(a) we see that the total number of model updates is reduced as the nodes failure and Rx/Tx failure probabilities increase. A higher node failure probability implies a reduction of the operational sensor nodes participating in the distributed algorithm i.e. a reduction of the effective network's density. This in turn implies fewer neighbors within a sensor's communication range, making it more difficult for a Master Candidate to satisfy the necessary conditions to become a Master (see Section

4.1.1 - Forming a Local Cluster). Moreover, increasing the Rx/Tx communication failure probability implies higher difficulty for the sensor nodes to collaborate with their neighbors in order to update the parameters of a local front model.

Figures 11(b) and 11(c) show (for the 100 sensor nodes per $1km^2$ density case) the mean number of Rx and Tx messages exchanged in the network per model update. From the presented line plots when the Rx/Tx failure probability increases in the range [0, 0.1] we observe a significant increase of messages per model update for each node failure probability curve. This can be explained if we consider that Rx/Tx failures trigger message retransmission which increases the number of messages exchanged over the network. Another interesting observation is that this trend becomes less profound as the nodes failure probability increases. This behavior can be explained if we consider that: a) increasing node failures implies an effective network density reduction and therefore a reduction of the mean number of neighbors within a node's communication range and thus a reduction of the mean number of the Rx and Tx messages exchanged in the neighborhoods. b) When the node failure probability is non-zero, retransmission is triggered even when the Rx/Tx failures probability is zero, since the failing nodes are not able to send the required acknowledgments. These triggered retransmissions increase in turn the mean number of the Rx and Tx message exchanged over the network. Thus, the already increased number of messages for the non-zero node failure probability cases explains why we observe a smoother increment of the mean number of messages when the Rx/Tx failure probability increases in the range [0, 0.1].

Figures 11(b) and 11(c) also suggest that for zero Rx/Tx failure probability the mean number of Rx and Tx messages per model update for all the non-zero nodes failure probability cases is almost equal (Rx) to or larger (Tx) than the corresponding Rx and Tx mean number of messages of the zero node failure probability case. At first glance this behavior may seem counter-intuitive since for the non-zero nodes failure probability cases the effective density of the network is reduced and therefore we would expect the mean number of messages exchanged per model update to be smaller. However, this is not the case since the potential retransmission triggered if node failure probability is non-zero increases the number of messages exchanged over the network.

Moreover, in Figures 11(b) and 11(c) we also observe that as the Rx/Tx failure probability increases in the range [0.1, 0.3] the mean number of the Rx and Tx messages per model update remains almost unchanged for node failure probabilities 0 and 0.1 and increases only slightly for larger node failure probabilities with values 0.2 and 0.3. To explain this behavior we have to consider the following four mechanisms which affect the number of messages exchanged per model update: a) The increase of node and Rx/Tx failure probabilities

Experiment 1					
Average Percent Change Compared to 100 Nodes Density Scenario					
Nodes Density	Total Updates	Mean Rx Mess.	Mean Tx Mess.	Mean Rx Energ.	Mean Tx Energ.
75	-37.4/2.9	-15.8/3.7	-8.9/5.1	-15.2/3.9	-5.8/4.5
125	+31.9/5.3	+18.3/4.9	+12.6/4.7	+17.3/5.4	+7.6 /4.1

TABLE 3: The average percent change (increase(+), decrease(-))/stdvs of the a) total number of model updates, b) mean number of Rx and Tx messages exchanged per update, and c) mean Rx and Tx energy consumed per update, for the 75 and 125 nodes (per km^2) density scenarios relatively to the 100 nodes density scenario.

increases the mean number of messages exchanged over the network due to the triggered retransmissions. b) The increase of the node and Rx/Tx failure probabilities increases the probability of "zombie" clusters formation (see Section 4.1.1), i.e. clusters in which the sensor node malfunctions and Rx/Tx failures render the Master node incapable to update its model parameters. The messages exchanged (wasted) within "zombie" clusters combined with the smaller number of model updates *increase* the mean number of messages required per model update. c) On the other hand, the increase of the nodes failure probability *reduces* the effective network's density and therefore the mean number of the Rx and Tx messages exchanged over the network. d) Finally, the increase of the Rx/Tx failure probability reduces the probability for a sensor node to receive or transmit successfully a message, which in turns *reduces* the total number of Rx and Tx messages. The line plots in Figures 11(b) and 11(c) suggest that for node failure probability 0 and 0.1 the increse of the mean number of messages, caused due to mechanisms (a) and (b) is counterbalanced by the message traffic reduction mechanisms (c) and (d) and therefore no significant changes are observed as the Rx/Tx failure probability increases in the range [0.1,0.3]. However, for higher node failure probabilities, i.e. 0.2 and 0.3, the more frequent formation of "zombie" clusters combined with the more frequent triggering of retransmissions results to a small increase of the mean number of messages per model update in the same Rx/Tx probability of failure range.

Figures 11(d) and 11(e) show the mean energy consumed for Rx and Tx communications per model update. As expected, due to the direct relation between the Rx/Tx messages (Bytes) and Rx/Tx communication energy, the energy and messages per model update line plots follow similar trends. However, as the Rx/Tx communication failure probability increases we observe a small increase of the gradient of the energy line plots as compared to the corresponding line plots for the messages. This behavior can be explained if we consider that an increase of the Rx/Tx failure probability makes it more difficult for a Master node to find a qualified new Master and eventually forces it to broadcast a message to its Slaves so that they propagate its updated model to their neighbors (see Section 4.3 Model Propagation). The message broadcasted by the Slave nodes ($PPM(UM_i)$) carries the information of the Master's updated model and therefore requires the transmission of many bytes which increases the mean Rx and Tx energy consumption.

Finally, we have to mention that the corresponding Figures for the 75 and 125 nodes per km^2 density scenarios follow similar trends and are therefore subject to similar interpretations. Due to space limitations the corresponding plots are provided in Section S1.3 (see Figures S3, S4). However, in Table 3 we summarize the differences (average percent change) relatively to the 100 nodes per km^2 density scenarios. The provided statistics were computed by considering as sample points all local front model updates from all node density ($\{75, 100, 125\}$) and failure probability cases ($\{0, 0.1, 0.2, 0.3\}$). We observe that as the network density increases (decreases) the total number of model updates also increases (decreases). This is as expected since a higher (lower) nodes density implies more (fewer) neighbors within a sensor's communication range, making it more easy (difficult) for a Master Candidate node to satisfy the necessary conditions to become a Master (see Section 4.1.1 - Forming a Local Cluster). Finally, the increased (decreased) number of neighbors also explains why the mean number of Rx and Tx messages exchanged and energy consumed per model update increases (decreases) with the increase (decrease) of the network density.

In Section S1.4, we also provide experimental evaluation results showing how the algorithm performs in cases where the "real" probabilistic sensing model differs from the sensing model assumed by the sensor nodes (model mismatch conditions). The results suggest that the accuracy of the local front evolution parameter estimates (orientation and speed) is rather insensitive to sensing model mismatches.

5.3.2 Experiment 2: Diffusive hazards with irregular shapes

The objective of Experiment 2 was to evaluate the ability of the proposed distributed algorithm to estimate accurately the evolution parameters of hazardous phenomena having non-geometric irregular front shapes, large propagation speed variations, etc. To generate hazards with such more realistic characteristics we employed FLogA [30], a web-based interactive software tool (developed in our group) which allows us to draw a forest area anywhere in Europe over Google Earth [33], insert fire ignition points ("hotspots"), define wind direction and speed scenarios, and then simulate and geo-animate the evolving wildfires under different conditions (see Section S2.1).

Using FLogA we defined a square forest area (of $1km^2$) at Hymettus mountain in Attica Greece and generated 5 different wildfire scenarios. Altering the hotspot locations and prevailing wind conditions (speed and direction) gives rise to different wildfire propagation patterns (see Section S2.1 and Figures S6-S10). Similarly to Experiment 1, we evaluated the proposed algorithm's behavior considering different: a) sensor node densities (75, 100, 125 nodes per $1km^2$), b) sensor node deployments (10 per wildfire scenario), c) sensor node failure and Rx/Tx failure probabilities (with values $\{0, 0.1, 0.2, 0.3\}$), and d) small and large sensing range radii (with values $\{0.1m, 15m\}$).

Due to space limitations the results are summarized in Table S2 and Figures S12-S14 which are isomorphic to Table 2 and Figure 11 respectively. By comparing corresponding table entries we observe that the mean speed and direction estimation errors are on average larger in Experiment 2, but only 4.26% and 2.74^o degrees respectively (with standard deviations 0.46% and 0.42^o degrees), despite the irregularities and the more dynamic evolution characteristics of the considered wildfire fronts.

In addition, in order to investigate how wind speed affects the accuracy of the algorithm we compared the estimation errors under strong vs. light wind speed conditions (for details see Section S2.1). For strong wind scenarios the mean speed and direction errors are larger on average by 3.92% (standard deviation 0.69%) and 1.91^o degrees (standard deviation 0.53^o) respectively relatively to light wind conditions. This modest error differential is justified since strong winds produce larger front line variations which are more difficult to track.

In all wildfire scenarios described above the wind speed and direction were considered constant within the whole forest area. In order to investigate how their spatial variation affects the estimation errors we used WindNinja [34], a tool that generates spatially varying wind field parameters by modulating a reference speed and direction value taking into account the terrain's geomorphology, and repeated the same simulations. We observe that the speed and direction estimation errors increased on average by only 1.41% and 0.88^o degrees (with standard deviations 0.61% and 0.38^o) respectively, despite the fact that spatially varying wind speed and direction cause locally more irregular wildfire evolution patterns.

As mentioned in the Introduction, most in-network processing schemes reported in the literature try to delineate dynamically the boundaries of an evolving continuous object using a dense array of deployed sensor nodes [5]–[10]. These schemes do not attempt to estimate the local front line's evolution characteristics or predict their spatiotemporal evolution. One notable exception is the work in [11], [12] where the authors introduced a simple way to estimate, as we do, the speed and direction of the local front. They use them to implement a "wake up" mechanism to decide which "sleeping" nodes to activate selectively for near term front tracking in order to reduce the network's overall energy consumption. However, PRECO (PREdictive Continuous Object tracking scheme) requires global sensor nodes synchronization [5] rendering it impractical even for medium size WSNs. Nevertheless, for completeness purposes we compared

R_d	Exp. 1		Exp. 2		Exp. 2 Var. Wind	
	Orient.	Speed%	Orient.	Speed%	Orient.	Speed%
$0.1m$	4.31/10.68	12.84/13.07	6.92/11.84	16.88/16.13	7.84/12.42	18.51/17.42
$15m$	4.79/10.89	13.52/13.96	7.21/12.03	17.31/16.92	8.17 /12.61	18.76/17.54
$PRECO$	28.15/22.54	39.62/89.15	29.03/23.17	40.84/92.89	29.42/23.26	43.23/94.2

TABLE 4: The Median/Inter Quartile Range of the orientation error (in degrees) and percent speed error of the proposed in network algorithm (under different sensing radii assumptions) and PRECO method for all conducted experiments.

it to our method under the scenarios of Experiment 1, Experiment 2, and Experiment 2 with spatially varying wind parameters.

Table 4 provides for each Experiment, the orientation error (in degrees) and the percent speed error when using the proposed in-network algorithm and PRECO. For each table entry the reported statistics were computed by considering as sample points all local front updates from all node density ($\{75, 100, 125\}$) and failure probability cases ($\{0, 0.1, 0.2, 0.3\}$). Our method is shown to outperform considerably (for all sensing radii scenarios) PRECO, resulting to smaller estimation errors. When using very high node densities (thousand of sensor nodes per km^2) PRECO achieves reasonable accuracy, however it fails to estimate correctly the spatiotemporal characteristics of the continuous object in WSNs with practical sensor densities. This behavior can be explained if we consider the following:

PRECO considers as a local front (boundary line), a line segment that connects two adjacent special Boundary Nodes (BN), called Master Boundary Nodes (MBN). It uses the location coordinates of the corresponding fixed MBNs to calculate the orientation parameter of a local front. In contrast, our method calculates the orientation of a local front based on the coordinates of two points (K_1 and K_2) estimated using two local speed observations of the diffusive hazard's front line (see Section 3.3.2). Our orientation estimation approach, which is independent from the sensor node locations, explains why this parameter's estimation accuracy is almost insensitive to WSN's density variations (see Section 5.3.1 paragraph 3). Moreover, the small number of MBNs present at low and realistic WSN density scenarios leads to a coarser piece-wise linear approximation of the diffusive hazard's boundary, which in turn explains the larger orientation estimation errors when using PRECO.

Finally, to estimate the evolution speed of a boundary line, PRECO uses the locations and time of detection of the MBNs and of their neighbors. As indicated by PRECO's speed equations (see formulas on page 4 in [11]), the speed's estimation accuracy depends on the number of MBN neighbors and their positions relatively to the continuous object front's evolution direction. In general, it is expected that as the number of MBN neighbors increases the accuracy of the local front speed estimates will also increase. A detailed presentation of the comparison results is provided in Section S3.

Finally, to assess the practicality of the proposed algorithm for real world WSN implementations we ported it to the Atmel Raven WSN platform [28]. Using an embedded 8-bit CPU (ATmega1284P) clocked at 8 MHz the average time required by a Master node to update and propagate its local model parameters was 523ms (492ms for computation and 31ms for communication), a fact fully supporting the claim that of our approach is suitable for real world WSN-based environmental monitoring applications.

6 Conclusions

We presented a distributed WSN algorithm for estimating accurately the spatiotemporal evolution parameters (orientation, direction and speed) of the local front of a diffusive hazard. It is based on a Bayesian parameters estimation procedure implemented in a collaborative fashion by dynamically formed clusters (triplets) of sensor nodes. The algorithm updates the local front model parameters and propagates them to sensor nodes situated in the direction of the hazard's propagation in a fully decentralized manner. Extensive simulation results show that the proposed scheme can estimate accurately the time-varying local parameters of different types of irregular fronts, while using WSNs of realistic density. Moreover, its estimation accuracy is robust to changes in WSN density, sensor node failures and communication link failures.

Model parameters are updated based on closed form algebraic expressions making the presented approach practical and appealing for real-world hazard tracking applications. Relatively to other published schemes, our in-network algorithm exhibits the following unique characteristics: It works with *low* and realistic density WSNs, it is robust to sensor node and communication link failures which are certainly expected in harsh environments, and *does not* require any sensor node clocks synchronization, which is very difficult to achieve anyway even in small scale WSNs operating in non-harsh environments.

We are currently developing an algorithm which combines the produced local front model estimates dynamically, as they become available to a fusion center, to construct an estimate of the overall hazard's front line and "project" it (propagate it in space and time) to the future. This will allow us not only to make predictions but also characterize the associated uncertainty in a Bayesian manner. Front line predictions of course will be more accurate in areas populated with more sensors. This will enable the dynamic assimilation of WSN extracted information into integrative decision support systems for large scale environmental monitoring, hazard tracking and evolution prediction applications.

Acknowledgments

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund. The authors would like to thank Mr. Bogdos, University of Athens, for generating the wildfire scenarios of Experiment 2 and providing useful comments.

References

[1] P. Parmar, M. Zaveri, "Multiple Target Tracking and Data Association in Wireless Sensor Network," IEEE Int. Conf. CICN, pp.158,163, 3-5 Nov. 2012.

[2] M.A.Tinati, T.Y.Rezaii, "Multi-target Tracking in Wireless Sensor Networks Using Distributed Joint Probabilistic Data Association and Average Consensus Filter," Int. Conf. on Advanced Computer Control, ICACC '09. pp.51,56, 22-24 Jan. 2009.

[3] J. Liu, M. Chu, J.E.Reich, "Multitarget Tracking in Distributed Sensor Network," Signal Processing Magazine, 24 (3), pp. 36-46, May 2007.

[4] K. Martinez, J.K. Hart, R. Ong, "Environmental sensor networks," Computer , vol.37, no.8, pp.50,56, Aug. 2004

[5] H. Park, S. Oh, E. Lee, S. Park, S.-H. Kim, W. Lee, "Selective wakeup discipline for continuous object tracking in grid-based wireless sensor networks," Int. Conf. WCNC, pp.2179,2184, 2012.

[6] Y. Xu, W. Bao, H. Xu, "An Algorithm for Continuous Object Tracking in WSNs," Int. Conf. on Research Challenges in Computer Science, pp.242,246, 28-29 Dec. 2009.

[7] C.Yang, Q. Li, J. Liu,"A multisink-based Continuous Object Tracking in wireless sensor networks by GIS," Int. Conf. on, Advanced Communication Technology (ICACT), pp.7-11, 2012

[8] J.Kim, K. Kim, S.Chauhdary, W. Yang, M. Park, "DEMOCO: Energy-Efficient Detection and Monitoring for Continuous Objects in WSN", IEICE Trans. on Comm. vol.E91-B, pp.3648-3656, Nov.2008.

[9] W. Chang, H. Lin, Z. Cheng: "CODA: A Continuous Object Detection and Tracking Algorithm for Wireless Ad Hoc Sensor Networks". In Proc. of IEEE Int. Conf. CCNC, pp. 168-174, 2008.

[10] C. Zhong, M. Worboys, "Energy Efficient Continuous Boundary Monitoring in Sensor Networks" Technical Report, 2007. Available: http://ilab1.korea.ac.kr/papers/ref2.pdf.

[11] S.W. Hong, S. K. Noh, E. Lee, S. Park and S. H. Kim,"Energy Efficient Predictive Tracking for Continuous Objects in Wireless Sensor Networks" in Proc. 21st Int. Symp. on Personal, Indoor and Mobile Radio Communications, pp. 1725-1730, 2010.

[12] S.W. Hong, S. K. Noh, E. Lee, S. Park and S. H. Kim,"A Novel Continuous Object Tracking Scheme for Energy Constrained Wireless Sensor Networks" in Proc. of the IEEE. Conf. Vehicular Technology, pp.1-5, 2010.

[13] D. V. Manatakis, E. S. Manolakos, "Collaborative Sensor Network algorithm for predicting the spatiotemporal evolution of hazardous phenomena," Int. Conf. on SMC 2011, pp. 3439-3445

[14] D. V. Manatakis, E. S. Manolakos, "Predictive modeling of the spatiotemporal evolution of an environmental hazard and its sensor network implementation" In Proc. ICASSP 2011, May 2011, Prague-Czech pp. 2056-2059

[15] Y. Wang, R. Tan, G. Xing, J. Wang, X. Tan, "Accuracy-Aware Aquatic Diffusion Process Profiling using Robotic Sensor Networks", In Proc. of the 11th Int. Conf. on IPSN, pp.281-292, 2012.

[16] L. A. Rossi, B. Krishnamachari, C. C. J. Kuo, "Distributed Parameter Estimation for Monitoring Diffusion Phenomena Using Physical Models," Int. Conf. IEEE SECON, pp.460,469, Oct. 2004.

[17] A. Ghosh, S. K. Das, "Coverage and Connectivity Issues in Wireless Sensor Networks", Elsevier journal. Pervasive and Mobile Computing Vol. 4, Issue. 3, pp. 303-334, June 2008.

[18] A. Hossain, P. K. Biswas and S. Chakrabarti, "Sensing Models and its Impact on Network Coverage in Wireless Sensor Network", Int. Conf. Industrial and Information Systems pp.1,5, Dec. 2008.

[19] H. Ahmadi, "Probabilistic Coverage and Connectivity in Wireless Sensor Networks", M.S. thesis, Dept. of Computer Science. Simon Fraser University, Vancouver, Canada, 2007.

[20] N. Ahmed, S.S. Kanhere, and S. Jha, "Probabilistic Coverage in Wireless Sensor Networks", In Proc. of IEEE Conf. on Local Computer Networks (LNC'05), pp. 672-681, Nov. 2005.

[21] Y. Tsai, "Sensing Coverage for Randomly Distributed Wireless Sensor Networks in Shadowed Environments", IEEE Tran. on Vehicular Technology, Vol.57, no. 1, pp. 556-564, January 2008.

[22] J. Zhang, T. Yan, S. H. Son, "Deployment Strategies for Differentiated Detection in Wireless Sensor Networks", In Proc. of IEEE Int. Conf. SECON '06, pp.316,325, 28-28 Sept. 2006.

[23] E.S. Manolakos, D. V. Manatakis, G. Xanthopoulos, "Temperature Field Modeling and Simulation of Wireless Sensor Network Behaviour During a Spreading Wildfire", In Proc. of IEEE Int. Conf. EUSIPCO 2008.

[24] A. Runalls, "A Kullback-Leibler Approach to Gaussian Mixture Reduction",IEEE Trans. On Aerospace And Electronic Systems Vol. 43, Issue. 3, pp. 989-999, 2007.

[25] J.R. Hershey, P.A. Olsen, "Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models",In Proc. of the IEEE Int. Conf. ICASSP 2007, pp.IV-317 - IV-320.

[26] P. A. Bromiley "Products and Convolutions of Gaussian Distributions": http://www.tina-vision.net/docs/memos/2003-003.pdf

[27] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - A lightweight and exible operating system for tiny networked sensors. In Proc. of IEEE Int. Conf. on LCN 04, pp. 455-462, 2004.

[28] Atmel AVR Raven: http://www.atmel.com (10/02/2014).

[29] Martin Stehlik,"Comparison of Simulators for Wireless Sensor Networks," M.S. thesis, Faculty of Informatics., Masaryk University., Brno, Czech Republic 2011.

[30] N. Bogdos, E. S. Manolakos, "A tool for simulation and geo-animation of wildfires with fuel editing and hotspot monitoring capabilities," Elsevier Journal of Environmental Modelling and Software, Vol.46, August 2013, pp. 182-195.

[31] Animation of algorithm's behavior in the presence of multi-source diffusion processes: https://www.dropbox.com/sh/wwe154r4qn0gal6/xIEVNBNKH4

[32] U.S. Tristan, "The Diffusion Equation A Multi-dimensional Tutorial", Available: www.rpgroup.caltech.edu/ natsirt/aph162/diffusion.pdf (07/12/2013).

[33] Google Earth: http://www.google.com/earth/ (20/01/2014).

[34] Firemodels.org-WindNinja: http://www.firemodels.org/index.php/research-systems/windninja (07/12/2013)

Dimitris V. Manatakis is currently a Ph.D. candidate at the National and Kapodistrian University of Athens, Dept. of Informatics and Telecommunications. He obtained the M.Sc. degree with *honors* in Signal Processing from the same Department (2007), and holds a BEng degree in Electronics Engineering from the Dept. of Electronics, Technological Educational Institute of Lamia Greece (2004). In 2004 Dimitris was awarded a prize from the *Greek State Scholarships Foundation* after ranking among the top three students in his class. His doctoral research is supported by an "Hrakleitos II" grant co-financed by national and EU funds. He has participated as researcher in the EC funded research project SCIER (Sensor and Computing Infrastructure for Environmental Risks) and is currently participating in the GSRT funded "Aristeia II" research project "StochSoCs" (Flexible Systems on Chip for Parallel Stochastic Simulation of large biochemical networks in Systems Biology). His research interests are in Statistical and Distributed Signal Processing, Machine Learning, Estimation Theory, Dynamic Systems Modeling, and their application to Environmental Risk Management and Systems Biology.

Elias S. Manolakos (eliasm@di.uoa.gr) is a Visiting Scholar at the Wyss Institute for Biologically Inspired Engineering, Harvard University and Assoc. Professor with the Dept. of Informatics and Telecommunications, University of Athens. Before returning to Greece he was a tenured faculty with the ECE Dept. Northeastern University Boston. His research interests are in parallel and distributed signal processing, machine learning, and their applications to environmental and biomedical applications. Elias enjoys interdisciplinary research and has played a leadership role in more than 20 funded research projects in the EU and the US. He has served in the Editorial Board of several journals, such as the IEEE Trans. on Signal Processing, IEEE Signal Processing Letters, Journal of Signal Processing Systems, Springer etc. He has authored, or co-authored with his students, more than 110 publications in refereed scientific journals and Conference Proceedings. He holds a PhD degree from University of Southern California, an MSc degree from University of Michigan, Ann Arbor, and a Diploma in EE from the National Technical University of Athens. Dr. Manolakos is an IEEE Senior Member, currently serving in the IEEE TC on Machine Learning for Signal Processing (MLSP).