Received June 30, 2020, accepted August 5, 2020, date of publication August 20, 2020, date of current version August 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017074

Predictive Tracking of Continuous Object Boundaries Using Sparse Local Estimates

DIMITRIS V. MANATAKIS[1,2], (Senior Member, IEEE), AND ELIAS S. MANOLAKOS[1], (Senior Member, IEEE)
[1]Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, 16122 Zografou, Greece
[2]Emulate Inc., Boston, MA 02210, USA

Corresponding author: Elias S. Manolakos (eliasm@di.uoa.gr)

This research was 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.

ABSTRACT Environmental hazards (wildfires, floods, oil spills) are often modeled as ''continuous objects'' which evolve in space and time taking irregular shapes. Tracking their boundaries and accurately predicting their spatiotemporal spreading patterns is of paramount importance to combat their often catastrophic consequences. Wireless Sensor Networks (WSN) have been very instrumental for this purpose. However, current WSN-based methods require a prohibitively large density of deployed sensors to achieve a reasonable boundary reconstruction accuracy because they are based on the explicit identification of nodes close to the boundary. We present a novel approach that can track and predict the global boundary using only a relatively small number of distributed local front estimates. Our approach first filters and fuses the available sparse set of local front estimates (e.g. vectors of local orientation, direction and speed) and then uses the resulting information to reconstruct a smooth curve prediction of the evolving object's boundary at a future time. Moreover, since the uncertainty of the local front parameters is modeled, it can provide a heatmap representation of the evolving object, indicating the probability for each point in the area of interest to be reached at a future time by the spreading hazard. These predictive modeling capabilities when combined enable effective decision support for crisis management. We demonstrate that different types of continuous objects can be tracked with accuracy, even when only a relatively small number of noisy local front estimates is available. Our approach is practical since it can be applied in many situations where global boundary prediction updates are important to obtain by new sparsely distributed noisy local front estimates as soon as they reach a control center while the hazard is still progressing.

INDEX TERMS Continuous object tracking, predictive modeling, hazards tracking, boundary reconstruction algorithm, environmental monitoring.

I. INTRODUCTION

During the last decade, there is a growing interest in exploiting Wireless Sensor Networks (WSNs) and remote sensing technologies to track effectively diffusive hazardous phenomena (e.g. wildfires, oil spills, chemical leaks etc.) [1]–[17]. Predicting accurately the evolution of such ''continuous objects'' spreading over a large geographical area is of paramount importance, since it allows authorities to optimize their response (hazard suppression, evacuation plans etc.) to save lives and limit the scope of a hazard's catastrophic effects.

Several WSN-based methods have been proposed for tracking the boundaries of a continuous object [2]–[15]. Their key strategy is to identify over time the sensor nodes closest to the object's evolving front-line (called boundary nodes). More specifically, they either use techniques to reconstruct the continuous object's boundary (e.g. by determining the convex-hull polygon of the boundary nodes locations [15]) or rely on human operators to identify the boundary's shape using the boundary nodes locations. As a results, existing methods demand unrealistic sensor densities (i.e. thousands of sensors per km^2), a requirement that renders them

The associate editor coordinating the review of this manuscript and approving it for publication was Gongbo Zhou.

impractical for real-world environmental monitoring applications. In addition, they do not track the local evolution characteristics of different parts of the continuous object and thus cannot provide decision support using predictive modeling. Dynamic Data Driven Application Systems (DDDAS) [16]–[19] try to address some of these severe limitations. Their main objective is to continuously improve the quality of predictions produced by hazard-specific models (mathematical, semi-empirical, etc.) by recalibrating them periodically using available sensing data streams from the field. However, the development of such systems is a very difficult task since in most cases it is almost impossible to exploit raw sensor data to recalibrate the parameters of hazard-specific simulation models.

Computing the convex-hull, is a very well studied problem [20], [21] and it has been extensively used in many application domains (e.g. image analysis, epidemiology, etc.) to determine the boundary of a given set of points (e.g. pixels, patient locations, etc.). However, there are many cases where the convex-hull cannot accurately represent the shape of the area occupied by the set of points (e.g. when the points form a non-convex shape). To overcome this difficulty, many convex-hull generalization algorithms have been proposed [22]–[25]. Although, they can determine a "tighter" non-convex area occupied by the given set of points, their boundary reconstruction accuracy greatly depends on the selection of specific parameters which are very difficult or even impossible to estimate [25], [26]. Moreover, all these algorithms assume a dense set of points uniformly distributed along the object's boundary area, a fact that renders them impractical for scenarios where only a small number of irregularly distributed points is available. Finally, these algorithms cannot take into account boundary reconstruction constraints (like C1 and C2 as defined in Section IV-A) which are necessary to capture correctly the evolution dynamics of a continuous object.

The boundary (global front) of a continuous object can be approximated by a set of line segments (local fronts) and each segment's evolution can be represented by a model with a small number of parameters (e.g. orientation, direction, and speed), as shown in [27]–[29]. In our prior work we have introduced a probabilistic framework and associated distributed algorithms that can estimate the *local* evolution characteristics of a propagating hazard's front using WSNs of realistic density. As the hazard progresses, we have shown how the deployed sensor nodes can be dynamically re-organized into small-size ad-hoc clusters (node triplets) and compute, within their local area, estimates of the hazard's characteristics. Moreover, we have shown how the sensor nodes can update the local front model parameters using closed-form expressions (analytical solutions of a Bayesian parameter estimation problem) that are easy to implement using their commodity microprocessors [30]. Importantly, a unique feature of our WSN-based environmental monitoring methodology is that each sensor node can also estimate the uncertainty associated with the local front estimates it produces as the hazard progresses [29].

In this work (that complements [29]) we address the interesting problem of how a stream of sparse (in space and time) and inherently noisy local front estimates can be used (as they are becoming available to the decision center) to reconstruct accurately the *global* boundary of a continuous object. Our approach, which combines machine learning and computational geometry methods, offers the following novelties and unique advantages that are also very important from a practical standpoint for environmental monitoring applications: (i) It can reconstruct the boundary of an evolving continuous object using only a small number of sparse local front estimates; (ii) it is robust to local front parameter estimation errors; (iii) in contrast to other methods [2], [6], [8]–[14] it determines the global boundary explicitly and does not require a human in the loop to recognise it implicitly based on the locations of the identified boundary nodes; (iv) it supports a probabilistic notion of a continuous object that is now viewed as a field, indicating for each point of the considered area the probability to be reached by the hazard at any given time. The methods presented here are quite general as different sensor modalities (WSN's, smart cameras etc.) can be abstractly viewed as nodes of a heterogeneous monitoring system generating inherently noisy estimates of local scope that a data center has to reconcile and fuse continuously in order to predict with accuracy an evolving hazard's irregular global boundary, even in areas with a limited number of available sensors.

The rest of the paper is organized as follows: In Section II we introduce the notation and preliminaries needed to understand the paper and motivate the key elements of the proposed global boundary tracking and prediction approach. In Section III we explain how the available local front estimates can be filtered and fused in order to generate consistent information for feeding the boundary reconstruction algorithm, presented in Section IV. In Section V we present and discuss simulation results for different types of hazards. Finally, we summarize our findings and outline work in progress in Section VI. Illustrative video animations [31], [32] (provided also as supplementary material) are used to demonstrate the online reconstruction algorithm "in action". We urge the readers to view these videos as they study Section V.

II. PRELIMINARIES

A. LOCAL FRONT MODELS

The boundary of a continuous object can be approximated by a piecewise linear curve. We represent each segment of this curve (to be called a *local front*) by a simple model that captures locally (i.e. within a circular area C_i of radius R_i which depends on the sensor node type, see Fig. 1) its local evolution characteristics (e.g. direction, orientation and speed). So in our abstract model, a local front estimate m_i is represented by the following parameters:

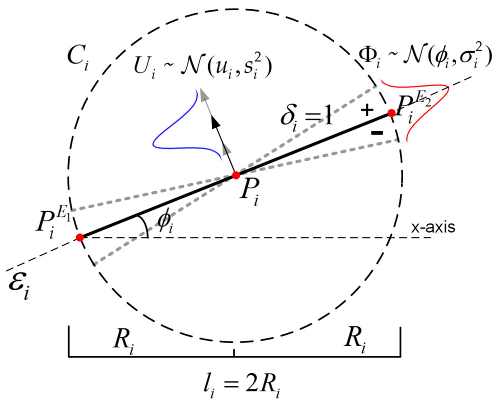


FIGURE 1. The model m_i of a local front segment and its parametric representation.

- Location ($P_i = (x_i, y_i)$): The physical coordinates of the line segment's middle point. Local front segments are assumed to be at close proximity to the surface of the earth (the z offset is ignored).
- Length (l_i): The length of the line segment that approximates locally (within a circle of radius $R_i = \frac{l_i}{2}$) the continuous object's boundary. The larger the radius R_i the larger the scope of the local model.
- Evolution direction (δ_i): A vector perpendicular to the line segment. It may take one of the two values $+1(-1)$, if the local front evolves into the positive (negative) half plane with respect to line ε_i.
- Orientation parameters (ϕ_i, σ_i): The angle Φ_i formed by the local segment and the horizontal axis (x-axis) is considered to be a random variable following a Normal distribution $\Phi_i \sim \mathcal{N}(\phi_i, \sigma_i^2)$. We will denote with $\hat{\phi}_i$ a realization (random sample) of Φ_i.
- Speed parameters (u_i, s_i): The speed U_i of the local front's line segment is considered to be a random variable that follows a Normal distribution $U_i \sim \mathcal{N}(u_i, s_i^2)$. We will denote with $\hat{u}_i$ a realization (random sample) of U_i.
- Local front model parameters estimation time (t_i), measured with respect to a global time reference.

Using the center coordinates $P_i = (x_i, y_i)$, the length l_i, and the orientation $\hat{\phi}_i$ of a local front estimate m_i, we can determine the coordinates of its end points $P_i^{Ez} = (x_i^{Ez}, y_i^{Ez})$ where $z = \{1, 2\}$ (for the proof see Appendix A).

We will use the notation $m_i(t_j)$ to denote the evolution of the local front model estimate m_i generated at time t_i to a future time instance t_j ($t_j > t_i$). In the text we may refer to $m_i(t_j)$ as the future "projection" of estimate $m_i(t_i)$. The midpoint, the end-points and the circular area of the space-time evolved local front $m_i(t_j)$ will be denoted as $P_i(t_j) = \big(x_i(t_j), y_i(t_j)\big)$, $P_i^{E_z}(t_j) = \big(x_i^{E_z}(t_j), y_i^{E_z}(t_j)\big)$ (where $z = \{1, 2\}$) and $C_i(t_j)$ respectively (see Fig. 2). In Appendix B we provide the derivation of the closed form algebraic expressions used to calculate the location coordinates of a space-time evolved local front. Finally, we will use $A_i(t_j)$ to denote the "area covered" or "swept" by m_i during its space-time evolution from time t_i to time t_j (see green shaded area in Fig. 2).

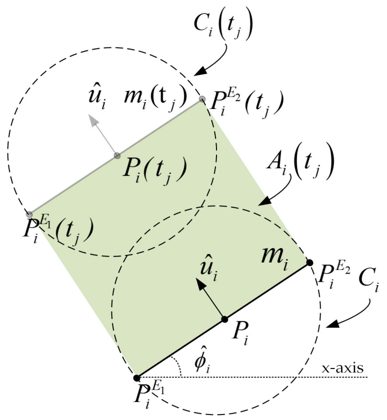


FIGURE 2. The space-time evolution (prediction) of local front m_i generated at time t_i to some future time $t_j > t_i$.

B. KEY ELEMENTS OF THE PROPOSED APPROACH

We assume that a distributed monitoring system (e.g. based on remote sensing, WSNs etc. [29]) provides to a decision center a stream of sparse in space and time local front estimates sampling the evolution behavior of a continuous object (hazard) at different locations and/or time instances, denoted by the black line segments in Fig. 3a. As soon as a sufficient number of local front estimates has become available, the proposed algorithm filters and fuses their information to determine the subset of them (see black segments in Fig. 3b) that can best (most consistently) represent the continuous object's evolution and will thus be used to reconstruct its boundary at a desirable time point (e.g. time instance t_{12} in Fig. 3). Next, using the local fronts' evolution parameters, the algorithm predicts their projected locations at the desired reconstruction time (see Fig. 3c) and determines a polygon approximating the continuous object's boundary (see black dashed polygon in Fig. 3d). Subsequently, using uniform cubic B-splines the algorithm determines a "smooth" curve which is the reconstruction of the hazard's boundary curve (see red curve in Fig. 3d). Finally, based on the uncertainties associated with the local front estimates, the algorithm provides a probabilistic view of the continuous object, computing for each location and time instance the probability to be covered by the object (see Section V-D).

III. FILTERING AND FUSING LOCAL ESTIMATES

In this section we elaborate on the procedure used to determine the best subset of local front estimates providing a consistent view of the phenomenon and can be used to reconstruct a faithful representation of the continuous object's boundary.

Let's assume that we want to predict where the hazard's global boundary will be located at time t_b. From the available local front estimates we select those with estimation times $t_i \leq t_b$, and form a set $\mathcal{M}$ containing their parameters (see Section II-A).

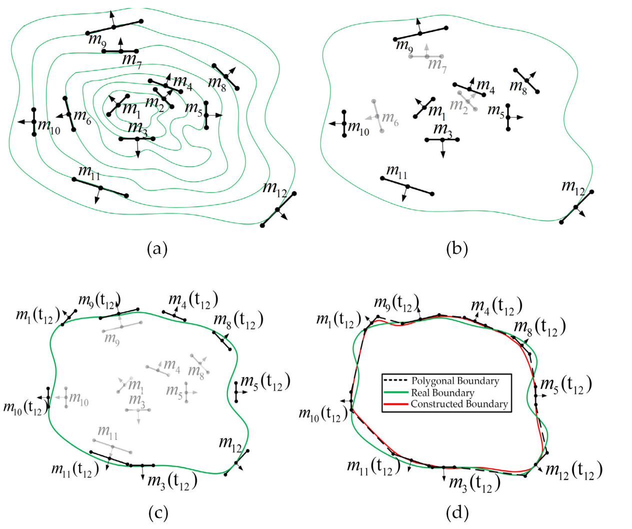


FIGURE 3. a) Each green curve depicts an expanding hazard's boundary at a different time (ground truth). Available local front estimates (black segments) are sparse in space and/or time, generated at different locations of assumed available sensors. b) Dark segments depict the subset of estimates selected to participate to the boundary reconstruction attempted for time $t = t_{12}$. c) The future-evolved (predicted) locations of the selected local front segments at reconstruction time $t = t_{12}$. d) The reconstructed polygon (black dashed line) and the smooth curve (red curve) approximating the hazard's true boundary (green curve).

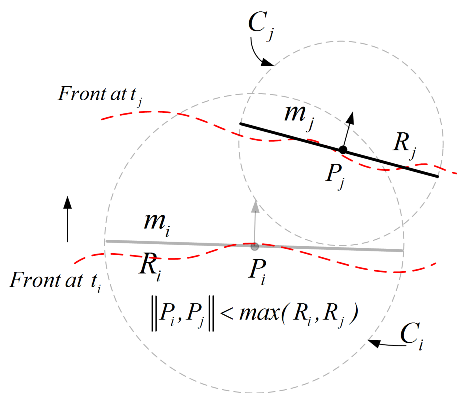


FIGURE 4. Model elimination procedure: Local front estimates $m_i(t_i)$ and $m_j(t_j)$ (where $t_i < t_j$) are considered to describe the evolution behavior of the same part of the evolving continuous object's boundary. Only the more recent estimate ($m_j(t_j)$) is kept in $\mathcal{M}$ (see text for details).

Our objective is to combine (i.e. filter and fuse) the spatio-temporal information of available local front estimates in $\mathcal{M}$ in order to determine the best subset of local front estimates that provide us with a consistent view of the hazard's front at the boundary reconstruction time t_b. If $\mathcal{M}$ contains at least N (specified by the user) of local front estimates (i.e. ($|\mathcal{M}| \geq N$)), we initiate the following procedure. At this point we have to mention that in order to reconstruct a smooth boundary (see Section IV-B) we need at least 2 local front estimates ($N \geq 2$). However, for higher values of N, we expect to have delayed (the algorithm waits until N local fronts become available) but more accurate reconstructions of the evolving boundary.

Step 1: We assume that two local front estimates m_i and m_j, generated at t_i and t_j respectively where $t_i \leq t_j$, describe the same reality (same part of the evolving boundary), if the Euclidean distance of their mid-points ($||P_i, P_j||$) is smaller or equal to the larger radius of their circular areas of scope (see Fig. 4).

$$||P_i, P_j|| \leq max(R_i, R_j) \tag{1}$$

For each pair of local front estimates in $\mathcal{M}$ (e.g. (m_i, m_j)) that satisfies condition (1), we keep only the estimate generated more recently (m_j in Fig. 4, since we assume *w.l.o.g* that $t_i < t_j$). This decision is justified based on the presumption that the more recent local front estimate captures better the current time varying evolution characteristics of the continuous object. The time complexity of *Step 1* is bounded by $O(|\mathcal{M}|^2)$. At the end of this filtering step, we use the local fronts remaining in $\mathcal{M}$ to form a new set $\mathcal{M}_f \subseteq \mathcal{M}$ containing their parameters information. Next, we check if $\mathcal{M}_f$ contains at least N local front estimates, and if so proceed to the next step.

Step 2: Reconstructing the boundary of a continuous object at time t_b, based on the space-time evolved local front estimates in $\mathcal{M}_f$ without considering possible intersections of their evolution paths usually leads to boundary shapes that deviate considerably from reality. Below we propose a method that first identifies and then handles "events" that may lead to such distortions.

Events Identification Procedure: For each pair of local front estimates in $\mathcal{M}_f$ (e.g. (m_i, m_j), where *w.l.o.g.* $t_i \leq t_j$), we check if any of the following *mutually exclusive* events may occur:

Type-1 event - Evolved center P_i approaches center P_j: After projecting the local front estimate m_i to the time t_j of the local front estimate m_j we check if $\{\{P_i(t_j) \in C_j$ *and* $P_j \notin A_i(t_j)\}$ *or* $\{P_j \in C_i(t_j)$ *and* $P_j \notin A_i(t_j)\}\}$. If the above condition holds (see Fig. 5a), an event of type-1 is registered for time t_j.

Type-2 event - Evolved m_i area covers P_j: After projecting the local front estimate m_i to time t_j we check if $\{P_j \in A_i(t_j)\}$. If the above condition holds (see Fig. 5b), an event of type-2 is registered for time t_j.

Type-3 event - Two evolving local front models block each other: If during the joint space-time evolution of two local fronts (m_i, m_j) beyond time t_j there exist a time instance t_d ($t_i \leq t_j < t_d \leq t_b$) such that the evolution path of one event blocks the evolution path of the other i.e. $\{P_i(t_d) \in A_j(t_d)\}$, or $(\{P_j(t_d) \in A_i(t_d)\})$, an event of type-3 is for time t_d (see Fig. 5c).

Type-4 event - Two evolving local front models can be fused: If during the space-time evolution of the two local fronts (m_i, m_j), there is a time instance t_f such that $t_i \leq t_j < t_f \leq t_b$) where the Euclidean distance of their mid-points becomes equal to the larger radius of their circular areas and none of them falls within the already covered area of the other, an event of type-4 is registered for time t_f (see Fig. 5d).

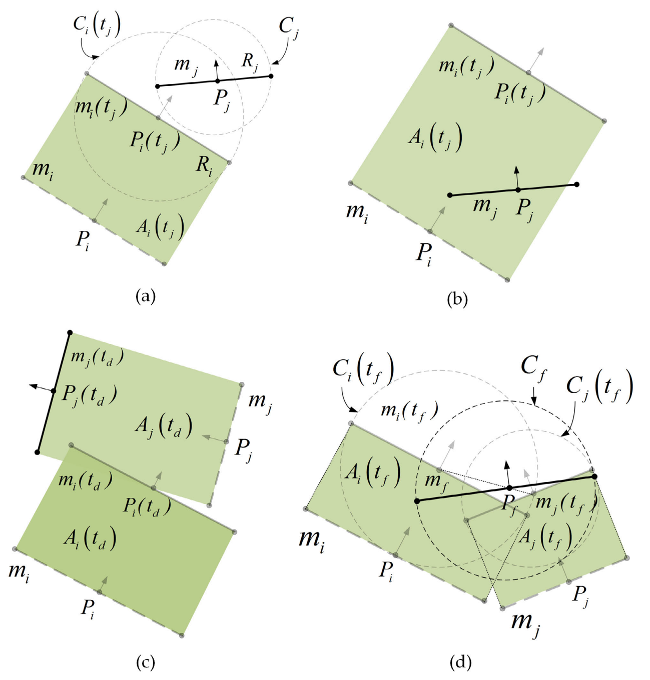


FIGURE 5. **(a)-(d) The four event types (1)-(4) identified and handled by the proposed algorithm during the space-time evolution of the local front estimates (see text for details).**

At this point we have to mention that all the registered event time instances (e.g. t_j, t_d and t_f) correspond at the starting time of the corresponding events. Moreover, the events identification procedure can be parallelized to increase the computational efficiency. At the end of the events identification procedure the algorithm finds the registered event with the earliest time and passes the involved pair of model estimates (e.g. $\{m_i, m_j\}$) to the *events handler* (see below) that implements the following procedure for each event type.

Events Handling Procedure:

Handling type-1 events: In this case the two local front estimates $m_i(t_j)$ and m_j describe at time t_j the evolution behavior of the same part of the continuous object boundary (see Fig. 5a). Using the presumption that the more recently estimated local front (m_j) describes better the current evolution behavior of the continuous object, we keep in $\mathcal{M}_f$ only the most recently updated local front (m_j).

Handling type-2 events: In this case the local estimate m_j appears inside the area which has been already covered by $m_i(t_j)$ (see green shaded area in Fig. 5b). This implies that the evolved local front $m_i(t_j)$, failed to describe the evolution behavior of the continuous object boundary, since the more recent local front m_j at time t_j, should be located on the continuous object's boundary and not inside the area which has already been covered by the continuous object. Based on this reasoning we keep in $\mathcal{M}_f$ only the local front estimate m_j.

Handling type-3 events: In this case the evolution path of a local front estimate $m_j(t_d)$, "blocks" the evolution path of $m_i(t_d)$ or vice versa (see Fig. 5c). To handle this event type we keep in $\mathcal{M}_f$ only the most recent local front estimate ($m_j(t_d)$).

Handling type-4 events: In this case two local front estimates ($m_i(t_f)$ and $m_j(t_f)$ in Fig. 5d) "meet" at some point during their space-time evolution without actually blocking each other. We consider this as an indication that two parts of the global boundary are "merging" at t_f. To handle this event type properly we apply a technique that fuses the information of the two local front estimates $\{m_i(t_f), m_j(t_f)\}$ to produce a new local front estimate $\{m_f(t_f)\}$ that better describes at time t_f the local evolution behavior of the continuous object's boundary (see darkest local front m_f in Fig. 5d). To estimate the parameters of the new local front model we apply the following equations:

$$\begin{aligned}
\mathcal{Z}_f &= \sum_{k=\{i,j\}} w_k \mathcal{Z}_k \quad where\ \mathcal{Z} = \{\hat{u}_f, \hat{\phi_f}, l_f, x_f, y_f\} \\
u_f &= \sum_{k=\{i,j\}} w_k u_k, \quad s_f^2 = w_i s_i^2 + w_j s_j^2 + w_i w_j (u_i - u_j)^2 \\
\phi_f &= \sum_{k=\{i,j\}} w_k \phi_k, \quad \sigma_f^2 = w_i \sigma_i^2 + w_j \sigma_j^2 + w_i w_j (\phi_i - \phi_j)^2
\end{aligned} \tag{2}$$

The equations in (2) above, indicate that the parameters of the new local front m_f (after the fusion), are calculated as linear combinations of the corresponding parameters of the fused local fronts $m_i(t_f)$ and $m_j(t_f)$. The derivation of equations (2) and the calculations of the weights $\{w_i, w_j\}$ are presented in Appendix C.

Events are handled in increasing event registration time. After handling *any event*, we repeat *Step 2* until there are no unhandled events involving local front estimate pairs in $\mathcal{M}_f$. The time complexity of this step is bounded by $O(|\mathcal{M}_f|^2)$.

At the end of *Step 2* we check if $|\mathcal{M}_f| \geq N$: If the condition holds, the procedure finds the space-time evolutions of the local fronts in $\mathcal{M}_f$ at the desired boundary reconstruction time t_b, forms a new set $\mathcal{M}_b$ that contains their information, and proceeds to the third and final step. The derivation of the equations used to calculate the location of projected local fronts is provided in Appendix B.

Step 3: This final correction step adjusts the length parameter of some projected local front estimates in $\mathcal{M}_b$ in order to avoid boundary deformations during the final curve reconstruction (see Fig. 6a).

The procedure checks if there is (are) any pair(s) of local front estimates in $\mathcal{M}_b$ for which their local circular areas overlap (e.g. see $\{m_i(t_b), m_j(t_b)\}$ in Fig. 6a). Next, it finds the local front with the earliest estimation time and reduces their lengths by an amount that is inversely proportional to the confidence ($\{w_i, w_j\}$) we have about their corresponding estimated evolution parameters, such that their circular areas do not overlap but osculate externally (see equation (3) and Fig. 6b).

$$R'_i + R'_j = ||P_i(t_b), P_j(t_b)|| \tag{3}$$

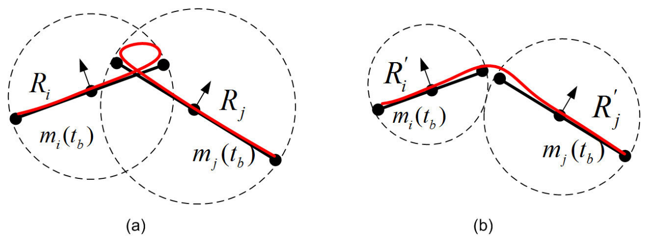


FIGURE 6. Local front lengths correction. a) The overlap of two local front segments results to a loopy B-spline curve during boundary reconstruction, b) The length reduction procedure prevents loopy curve formation.

where $\{R_i', R_j'\}$ are the new radii of the local fronts' circular areas calculated using equations (4) and (5) below.

$$\begin{cases} R_i' = R_i - \dfrac{c}{w_i} \\ R_j' = R_j - \dfrac{c}{w_j} \end{cases} \tag{4}$$

$$c = \frac{w_i w_j (R_i + R_j - ||P_i(t_b)P_j(t_b)||)}{w_i + w_j} \tag{5}$$

Equations (4) reduce the radii $\{R_i, R_j\}$ by an amount that is inversely proportional to the corresponding local front weights $\{w_i, w_j\}$. Equation (5) is the solution for c of the system of equations (3) and (4).

Using the new radii R_k' where $k = \{i, j\}$, we calculate the end-point coordinates $P_k^{E_z}(t_b) = (x_k^{E_z}(t_b)), y_k^{E_z}(t_b))$ where $z = \{1, 2\}$ of the affected local front estimates (see Appendix A), update the corresponding information in $\mathcal{M}_b$ and repeat *Step 3*. This step completes when there are no more overlaps between circular areas of local fronts in $\mathcal{M}_b$. The time complexity of *Step 3* is bounded by $O(|\mathcal{M}_b|^2)$.

IV. GLOBAL BOUNDARY RECONSTRUCTION

We present now the algorithm which reconstructs the continuous object's boundary using the information of the selected local front estimates in $\mathcal{M}_b$. The algorithm has to two main phases:

Phase 1: It forms a polygonal approximation of the boundary.

Phase 2: It constructs a curve that smoothly approximates the polygonal boundary.

To better explain the two phases and without loss of generality, we will use a running example: Let's assume that $\mathcal{M}_b$ contains after filtering and fusing 14 local front estimates (see Fig. 7). In Fig. 7 the black arrows indicate the evolution directions of the local front segments and the black dots their mid- and end-points.

A. PHASE 1: POLYGONAL APPROXIMATION

This phase has two Steps: *Step 1* determines the order in which the local front segments in $\mathcal{M}_b$ have to be "stitched" in order to form the polygonal approximation of the continuous object's boundary. Using the local fronts' connection order determined in Step 1 and their evolution direction parameters, *Step 2* constructs the polygonal approximation.

For the approximation to be realistic, the formed polygon should satisfy the following two conditions:

- C1: Be simple (no intersections between its edges).
- C2: Local front evolution direction vectors should point towards outside the polygon's area.

Before initiating *Step 1*, we form the following sets:

$\mathcal{C}$: Contains the mid-points of all local front estimates in $\mathcal{M}_b$.

$\mathcal{V}$: Contains the vertices of the convex hull of points in set $\mathcal{C}$ (in our example, $\mathcal{V} = \{P_3, P_4, P_7, P_8, P_{11}, P_{13}, P_{14}\}$, see Fig. 7a) in a connection order that forms the convex hull polygon. To find the convex hull polygon we use the Graham scan algorithm. The time complexity of this algorithm is $O(|\mathcal{C}| \cdot log(|\mathcal{C}|))$. It holds that $\mathcal{V} \subseteq \mathcal{C}$.

$\mathcal{Q} = \mathcal{C} \cap \mathcal{V}^c$: Contains the points in $\mathcal{C}$ that do not belong to the convex hull polygon. In Fig. 7a, $\mathcal{Q} = \{P_1, P_2, P_5, P_6, P_9, P_{10}, P_{12}\}$. Note that $\mathcal{V}$ and $\mathcal{Q}$ are partition subsets of $\mathcal{C}$.

The time complexity for the formation of the sets $\mathcal{C}$ and $\mathcal{Q}$ is bounded by $O(|\mathcal{M}_b|)$.

STEP 1

At this step the algorithm appropriately places the points of $\mathcal{Q}$ in $\mathcal{V}$. At the end of this step, the ordering of points in $\mathcal{V}$, indicates the order in which the local fronts in $\mathcal{M}_b$ have to be connected to form the polygon that approximates the continuous object's boundary.

We check the set $\mathcal{Q}$:

- If $\{\mathcal{Q} = \emptyset\}$: We continue to *Step 2*, since all local front mid-points are already included in $\mathcal{V}$.
- If $\{\mathcal{Q} \neq \emptyset\}$ (the general case): We repeat the following procedure until $\mathcal{Q}$ becomes empty:

Using the polygon determined by $\mathcal{V}$ (e.g. see convex polygon in Fig. 7a) we partition $\mathcal{Q}$ into the following subsets:

$\mathcal{Q}_{on}$: Contains the points of $\mathcal{Q}$ located *on* the polygon's edges.

$\mathcal{Q}_{out}$: Contains the points of $\mathcal{Q}$ located *outside* the polygon's area.

$\mathcal{Q}_{in}$: Contains the points of $\mathcal{Q}$ located *inside* the polygon's area.

Then we check:

If $\{\mathcal{Q}_{on} \neq \emptyset\}$: We select a point from $\mathcal{Q}_{on}$ (e.g. P_2 in Fig. 7c), determine the polygon's edge on which it lies (e.g. $\overline{P_1P_3}$ in Fig. 7c) and apply an "edge split" operation (i.e. the point becomes a vertex of the polygon). An edge split occurs when a point from $\mathcal{Q}$ (P_2 in Fig. 7c) becomes a polygon vertex (and is inserted into $\mathcal{V}$, between points P_1, P_3 which define the edge on which it lies). After an edge split we remove the "new" polygon vertex (P_2) from $\mathcal{Q}$. The aforementioned procedure is repeated until $\mathcal{Q}_{on}$ becomes empty.

If $\{\mathcal{Q}_{on} = \emptyset$ and $\mathcal{Q}_{out} \neq \emptyset\}$: We use the points in $\mathcal{Q}_{out}$ and form the following sets: $\mathcal{Q}_{out}^I$: Contains the points of $\mathcal{Q}_{out}$ that are centers of local front segments intersecting at

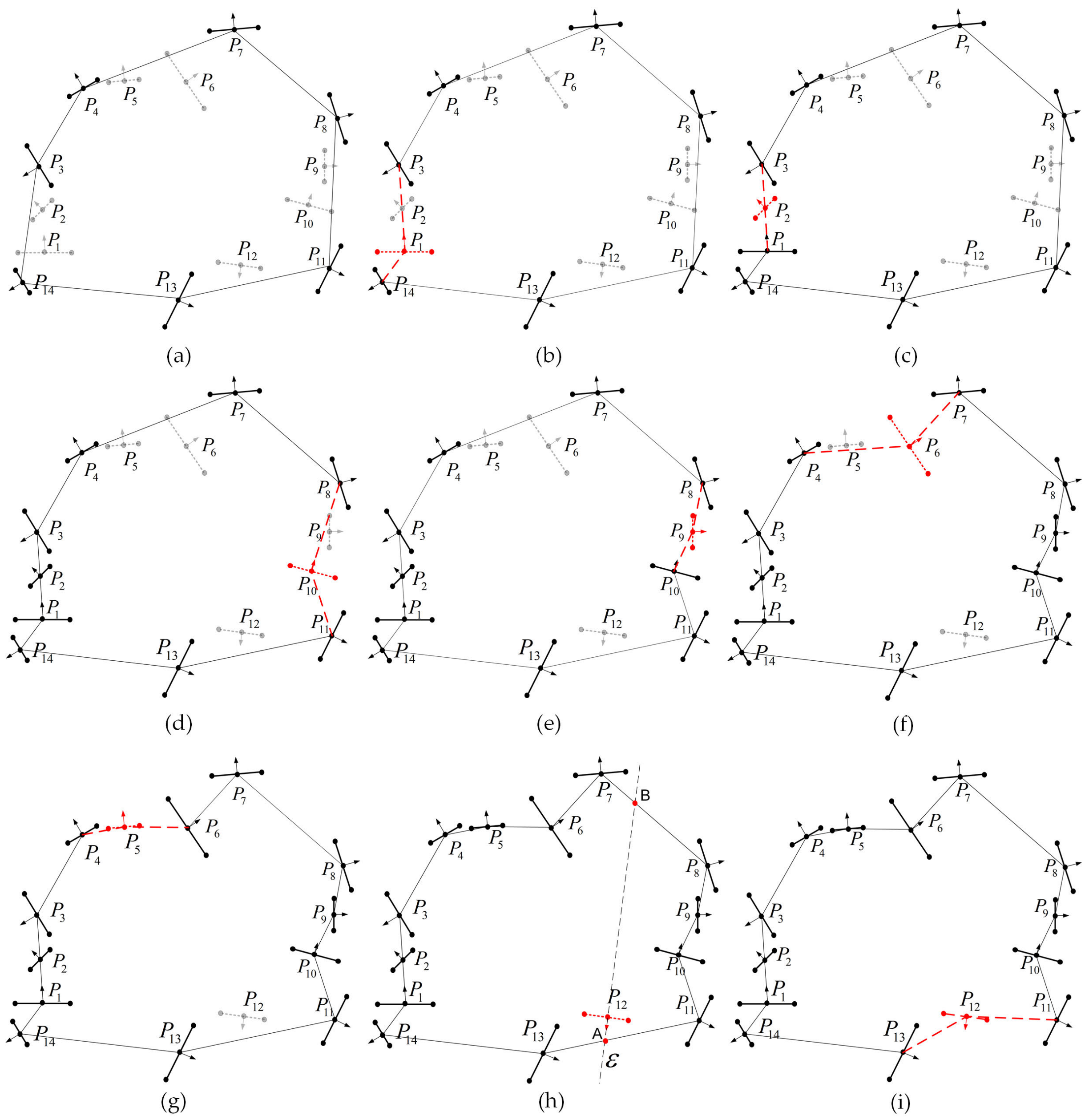


FIGURE 7. Polygonal approximation construction. Example showing how local fronts are processed (see text for details).

least one edge of the polygon (e.g. P_9 in Fig. 7d). $\mathcal{Q}^T_{out}$: Contains the points of $\mathcal{Q}_{out}$ that are centers of local front segments located totally outside (do not intersect) the polygon (e.g. P_5 in Fig. 7f). $\mathcal{Q}^I_{out}$ and $\mathcal{Q}^T_{out}$ are partition subsets of $\mathcal{Q}_{out}$.

After forming these subsets we check:

If $\{\mathcal{Q}^I_{out} \neq \emptyset\}$: We find the point in $\mathcal{Q}^I_{out}$ with the smallest distance to the polygon's edge that it intersects (e.g. P_9 in Fig. 7d). Using this point, we split the intersected edge (edge $\overline{P_8P_{10}}$ is split to $\overline{P_8P_9}$ and $\overline{P_9P_{10}}$, see Fig. 7e) and update the sets $\mathcal{V}$ and $\mathcal{Q}$. After an edge split we repeat *Step 1* from the beginning.

If $\{\mathcal{Q}^T_{out} \neq \emptyset\}$: We find the point in $\mathcal{Q}^T_{out}$ (e.g. P_5 in Fig. 7f) with the smallest distance to a polygon's edge and apply an edge split (e.g. edge $\overline{P_4P_6}$ is split into $\overline{P_4P_5}$ and $\overline{P_5P_6}$, in Fig. 7g). After an edge split, we update the sets $\mathcal{V}$ and $\mathcal{Q}$ and repeat *Step 1* from the beginning.

If $\{\mathcal{Q}_{on} = \emptyset$ and $\mathcal{Q}_{out} = \emptyset$ and $\mathcal{Q}_{in} \neq \emptyset\}$: We partition $\mathcal{Q}_{in}$ into two subsets: $\mathcal{Q}^I_{in}$: Contains the points of $\mathcal{Q}_{in}$ that are centers of local front segments intersecting at least one edge of the polygon (e.g. $\{P_1, P_6, P_{10}\}$ in Fig. 7a). $\mathcal{Q}^T_{in}$: Contains the points of $\mathcal{Q}_{in}$ that are centers to local front segments located totally inside the polygon (e.g. $\{P_2, P_5, P_9, P_{12}\}$ in Fig. 7a). $\mathcal{Q}^I_{in}$ and $\mathcal{Q}^T_{in}$ are partition subsets of $\mathcal{Q}_{in}$.

Then we check:

If $\{\mathcal{Q}_{in}^{I} \neq \emptyset\}$: We find the point in $\mathcal{Q}_{in}^{I}$ (e.g. P_1, in Fig. 7a) with the smallest distance from the polygon's edge that it intersects. Using this point we split the intersected edge ($\overline{P_{14}, P_3}$, in Fig. 7a), and update the sets $\mathcal{V}$ and $\mathcal{Q}$. After an edge split we repeat *Step 1* from the beginning.

If $\{\mathcal{Q}_{in}^{T} \neq \emptyset\}$: We derive for each point in $\mathcal{Q}_{in}^{T}$ the equation of the line that emanates from it and is perpendicular to the local front segment (e.g. line ε in Fig. 7h for point P_{12}). For each such line, we determine its intersection points with the polygon. We select the intersection point that is consistent with the local front's evolution direction (point A in Fig. 7h). Next, we find the point in $\mathcal{Q}_{in}^{T}$ with the smallest distance from the corresponding intersection point and use it to split the corresponding polygon's edge (e.g. $\overline{P_{11}P_{13}}$ is split into $\overline{P_{11}P_{12}}$ and $\overline{P_{12}P_{13}}$, in Fig. 7h). After the edge split we update the sets $\mathcal{V}$, and $\mathcal{Q}$ and repeat *Step 1* from the beginning. The time complexity for the formation of the sets $\mathcal{Q}_{on}$, $\mathcal{Q}_{out}$, $\mathcal{Q}_{in}$, $\mathcal{Q}_{out}^{I}$, $\mathcal{Q}_{out}^{T}$, $\mathcal{Q}_{in}^{I}$ and $\mathcal{Q}_{in}^{T}$ is $O(|\mathcal{Q}|\cdot|\mathcal{E}|)$ where $|\mathcal{E}|$ is the number of the polygon's edges. *Step 1* is repeated $|\mathcal{Q}|$ times. Thus the complexity of *Step 1* is bounded by $O(|\mathcal{Q}|^2 \cdot |\mathcal{E}|)$.

At the end of *Step 1*, the ordered set $\mathcal{V}$ uniquely determines a polygon (see Fig. 7i) which indicates the connection order of the local fronts.

STEP 2

This step uses the connection order and evolution direction parameters of the local fronts and first produces a *refined* polygonal approximation of the continuous object's boundary.

First, we sort the information in $\mathcal{M}_b$ according to the ordering of the mid-points of local fronts in $\mathcal{V}$. To sort the information in $\mathcal{M}_b$ we used the *Heapsort* algorithm that has time complexity $O(|\mathcal{M}_b| \cdot log(|\mathcal{M}_b|))$. Next, using the coordinates of the end-points of the first two local front segments in $\mathcal{M}_b$ (e.g. $\{P_1^{E_1}, P_1^{E_2}, P_2^{E_1}, P_2^{E_2}\}$ in Fig. 8a) we find the minimum distance pair of end-points (excluding pairs that belong to the same segment), and connect them (see red dashed edge $\overline{P_1^{E_1} P_2^{E_1}}$ in Fig. 8a). Then, using the non-connected end-point of the second local front segment ($P_2^{E_2}$) and the end-points of the third local front segment ($P_3^{E_1}, P_3^{E_2}$), we find the minimum distance pair and connect these points (see edge $\overline{P_2^{E_2} P_3^{E_1}}$ in Fig. 8a). Applying the aforementioned sequential procedure successively to all local front segments in $\mathcal{M}_b$, results to a refined polygon which has as vertices the mid- *and* the end-points of the local front segments (see Fig. 8b).

However, the aforementioned construction procedure does not guarantee that the refined polygon will satisfy conditions C1 and C2 discussed in *Phase 1* for polygonal approximations of continuous object boundaries. To ensure that these conditions hold we do the following: For each local front segment we check: a) if the edges that connect it with its adjacent local front segments intersect with other edge(s) (non-simple polygon), and b) if its evolution direction vector

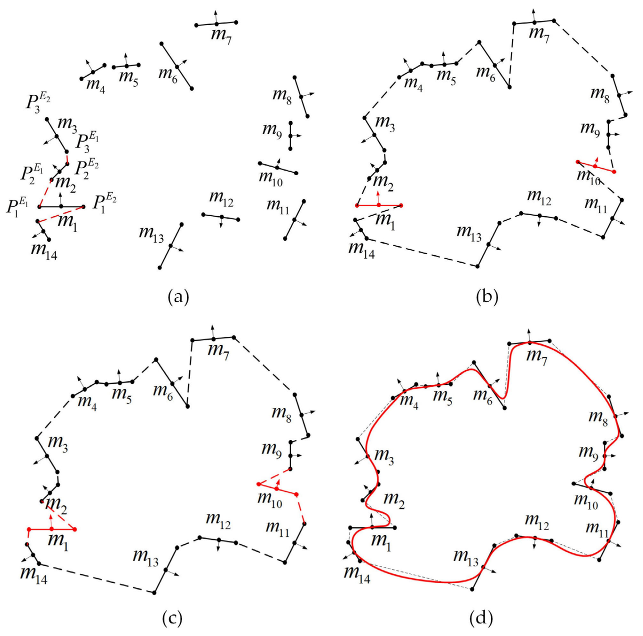


FIGURE 8. **Successive refinements to obtain a faithful representation of the boundary curve: a) Connecting end-points of subsequent local front segments. The red dashed lines show the first steps of local front segment "stitching" (see text for details). b) First polygonal approximation of the boundary, note that the direction vectors of m_1 and m_{10} point towards inside the polygon. c) A refined polygonal boundary describing better the evolution behavior of a diffusive object's boundary (direction vectors of m_1 and m_{10} point towards outside the polygon after interchanging the connections of their end-points). d) Smooth boundary reconstruction using a B-spline (red curve).**

points towards inside the polygon's area. If at least one of these conditions hold (e.g. in Fig. 8b the direction vectors of m_1 and m_{10} point towards inside the polygon), we interchange the corresponding local front's end-point connections to its neighbors (see Fig. 8c). If after a connections interchange, either one of the aforementioned conditions still holds true, we drop the "problematic" local front estimate from $\mathcal{M}_b$ and form a new polygon by connecting the end-points of its adjacent remaining local front segments. The time complexity for the formation of the refined polygon is bounded by $O(|\mathcal{M}_b|^2)$.

B. PHASE 2: SMOOTH BOUNDARY RECONSTRUCTION

In this phase we use the vertices of the refined polygon in conjunction with uniform cubic B-splines to generate a curve that smoothly approximates the continuous object's boundary. In nature, the boundaries of phenomena that can be modeled as continuous objects (e.g. wildfires, oil spills, etc.) tend to form smooth shapes and not polygonal. Inspired by this observation we used the popular cubic B-splines [33]–[36] to produce a smooth approximation of the continuous object polygonal boundary. Cubic B-splines use 3rd degree polynomials to represent each curve segment, and constraint the points that join the curve segments to meet C^2 continuity (no polarity changes in slope at the transition between segment i and segment $i + 1$) which guaranties a smooth curve boundary representation [37].

In our case the cubic spline curve uses as first control point the end-point of a local front segment ($P_1^{E_1}$, see Fig. 8a). This selection guaranties that the formed cubic spline will be tangent to mid-points of the local front segments. The red curve in Fig. 8d corresponds to the formed B-spline curve approximating the refined polygon approximation of the continuous object. For the smooth boundary reconstruction we applied the Carl R. de Boor's algorithm which requires $O(\xi^2)+O(\xi)$ operations where ξ is the degree of the polynomial functions (in our case $\xi = 3$). The time complexity for reconstructing a smooth boundary using cubic B-splines is bounded by $O(|\mathcal{M}_b| \cdot \xi^2)$ [37].

V. EVALUATION RESULTS

We present next simulation results demonstrating the ability of the proposed algorithm to track and predict with accuracy the boundary of continuous objects under different conditions.

A. THE SIMULATION ENVIRONMENT

For the evaluation of the boundary tracking and prediction algorithm we developed a flexible simulator in MATLAB which allows us to generate scenarios with different: a) propagating continuous object properties (shape, speed, acceleration), b) placement strategies (number and locations) of distributed sensors in the continuous object's propagation area, and c) induced estimation errors of local fronts' evolution parameters.

Before initiating a simulation, a MATLAB procedure takes as input the continuous object's propagation properties and the location coordinates of the assumed available sensors that when reached generate the local front estimates. Using this information, the MATLAB procedure computes the local front evolution characteristics (speed, orientation and direction) when the hazard reached a sensor, thus simulating the local sensing process (see Appendix D for details), and store them in a structure containing for each local front the values of its parameters, as defined in Section II. The generated structure is then passed as input, along with the continuous object's propagation properties, to another MATLAB simulator which implements the proposed boundary reconstruction algorithm and evaluates its accuracy.

B. EVALUATION METRICS

To evaluate the accuracy of the proposed algorithm, we consider the continuous object's area as a grid of square cells, and based on the real and reconstructed boundaries we classify the cells as True Positive, False Positives or False Negatives. To assess the boundaries similarity of the true and the reconstructed continuous object we use the *F1-Score* which is the harmonic mean of *precision* and *recall* (see Appendix E for details).

C. EXPERIMENTAL SETUP

A notable advantage of our proposed boundary reconstruction algorithm is that it can track with accuracy continuous objects using a small number of local front estimates (sparsity). To demonstrate this novel feature we have used local front estimate densities that are considered low for environmental monitoring applications, specifically $\{10^{-5}, 2\times10^{-5}, 3\times10^{-5}, 4\times10^{-5}\}\ \frac{estimates}{m^2}$, corresponding to 10, 20, 30 and 40 sensors available within a $1km^2$ square area respectively. For each density value we used a large number of randomly drawn sensor placements and demonstrate how the algorithm performs under different densities as well as errors of local front evolution parameters (speed and direction). For all experiments the local front segments length was set to $l = 100m$ (i.e. assuming sensors of sensing area with radius $R = 50m$). The evolution parameters (speed, orientation angle and direction) of the local fronts were set equal to the true values that the spreading hazard had when it reached the corresponding sensor that provided the measurement (perfect estimation, see assignment method details in Appendix D). However, to evaluate how the algorithm performs under local front orientation and speed estimation errors, we have repeated 6 times each experiment and tested the accuracy of the boundary construction. Specifically the data values used to model the orientation and percent speed errors were $\{10^o, 20^o$ and $30^o\}$ degrees and {10%, 20% and 30%} deviation from the ground truth boundary's speed respectively. For all experiments the minimum number of local front estimates required for the initiation of the boundary reconstruction algorithm was set to $N = 3$.

D. DIFFUSIVE HAZARD WITH REGULAR SHAPE

In Experiment 1 the diffusive hazard is modeled as a circular continuous object of fixed center but increasing radius, with the center located at the middle point of a $2km \times 2km$ square area. Considering the area's bottom left corner as being at the origin, the circle is centered at point $(1000m, 1000m)$ and has initial radius equal to $1m$. The speed at which the radius is increasing is described by a triangular function (see Fig. 9) with initial value $2.5m/min$. The speed increases with constant rate ($0.0265m/min^2$) until it reaches its maximum value ($5m/min$). Beyond that time point the speed starts decreasing at the same rate until it returns back to its initial value. At the end of the simulation the circle encloses the $1km^2$ local fronts'

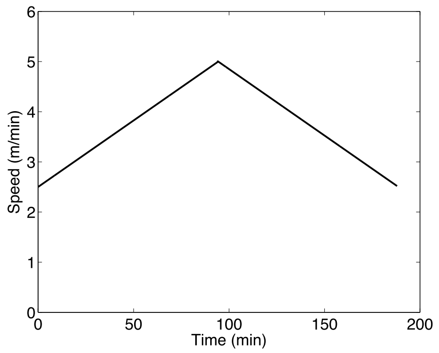


FIGURE 9. The time varying speed profile of the circular front (see text for details).

placement square area centered at point $(1000m, 1000m)$ (see red circle in Fig. 10f). Modeling propagating hazards with circular shape is justified as an approximation because 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 [39].

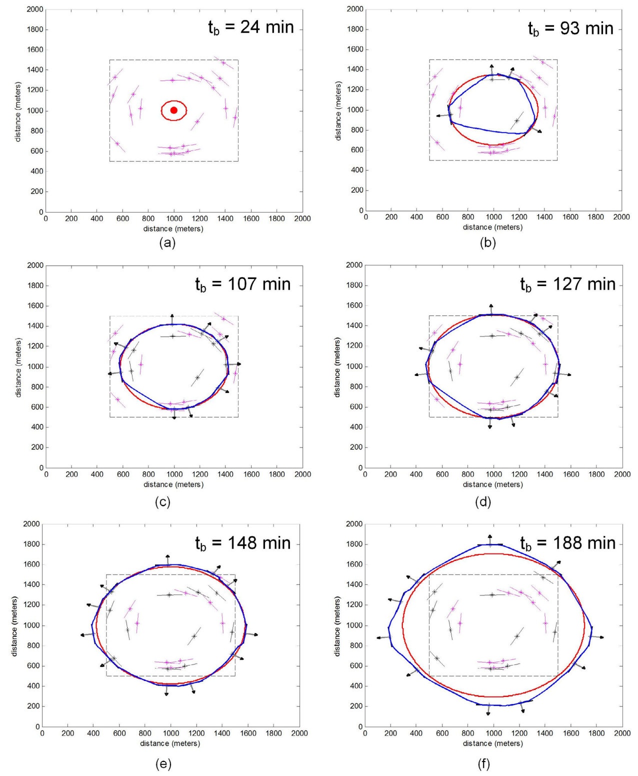


FIGURE 10. Video frame snapshots: Panels (a) through (f) show 6 snapshots of the simulation of Experiment 1. Line segments represent the available sparse local front estimates. With black color we mark the local front estimates selected to participate to the boundary reconstruction algorithm at each t_b. The blue curve represents the reconstructed boundary. The full video is available at [31].

We have created a MATLAB animation [31] (provided online and as supplementary material) that demonstrates the ability of the proposed algorithm to track the boundary of the evolving continuous object. Fig. 10a shows the circular front (red circle), its center (red dot), the 20 randomly placed local front estimates (magenta line segments) and their placement area (dashed square area of $1km^2$). We assume that a local front estimate starts participating to the boundary reconstruction algorithm when the front reaches the mid-point of the corresponding line segment (assumed measurement time at the sensor located at the mid-point). With black color, we indicate the subset of the local front segments (see Fig. 10b - 10f) that are selected to contribute to the approximation of the boundary curve, based on the selection method described in Section III. Using the characteristics of the corresponding local fronts, we determine their space-time evolutions (see Appendix B) and based on their "new" (predicted) locations (black segments on blue curve) we derive the "smooth" B-spline curve (blue curve) that approximates the circular continuous object, based on the algorithm described in Section IV.

To evaluate the boundary tracking accuracy of the proposed algorithm we attempt to reconstruct the boundary of the diffusive phenomenon 20 times during its evolution at equally spaced (every $188/20$ min) time points. We have to note that at each boundary reconstruction time point (t_b), we use only the local front estimates at locations reached by the diffusive phenomenon up to this time. The boxplots in Fig. 11a summarize the distribution of the boundary tracking accuracy considering 1000 random estimate placements (1000 runs) per local front estimates' density value. For the generation of each boxplot we used as sample points the F1-scores computed at all the time points a boundary reconstruction was attempted. As we observe from the boxplots the boundary reconstruction accuracy increases with the local front estimates density. Moreover the variability of the boundary reconstruction accuracy decreases as the density of local estimates increases. This implies that as the number of local front estimates increases we become more certain that the proposed algorithm will reconstruct with accuracy the diffusive object's boundary.

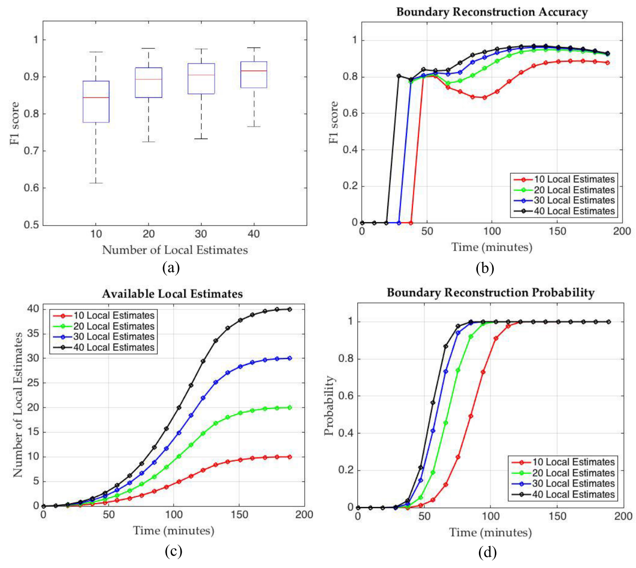


FIGURE 11. For different local front estimate densities: a) Boxplots summarizing the distribution of the boundary tracking algorithm's accuracy. b) The average boundary reconstruction accuracy as a function of time. c) The mean number of local front estimates available to participate at each time instance to the boundary reconstruction algorithm. d) The probability of successive boundary reconstruction as a function of time.

Fig. 11b shows the average boundary reconstruction accuracy (F1-score) as a function of time for different local front estimate densities. We observe that the accuracy of the boundary tracking drops slightly for a certain time interval and then recovers again. The time extent and experienced accuracy dip increases as the density of available local estimates decreases. To explain this behavior we have to consider the following:

At the first time steps the size of the continuous object is small and therefore a small number of local estimates suffices to produce an accurate representation of its boundary. As the size of the diffusive phenomenon gets bigger with time the accuracy of the boundary reconstruction drops until a sufficient number of local estimates becomes available (see also Fig. 11c) to participate in the boundary reconstruction algorithm.

In Fig. 11c we see that as the local front estimates' density increases, the mean number of the available local estimates that participate to the boundary reconstruction algorithm at each time step of the diffusive phenomenon evolution also increases. This larger number of available local estimates explains why the time interval where the accuracy drops is smaller at larger densities (see Fig. 11b). The slightly decreasing trend of the accuracy observed beyond 150 *min* is explained if we consider that up to that time point most local estimates have become available (see Fig. 11c) and therefore the reconstruction of the boundary beyond that point is based mostly on ''outdated'' local front estimate projections.

Another interesting observation is that beyond $t = 150\,min$ the boundary reconstruction accuracy reaches almost the same high level for all considered local front estimate densities (see Fig. 11b). This behavior can be explained if we consider that the proposed algorithm selects only a small number (the most suitable subset) of the available local front estimates (see Section III) to reconstruct the object's boundary, which was about the same for all the local front densities. This shows that the prediction ability of our algorithm is almost insensitive to the local front estimates' density, a fact that further supports our claim that it can be used to accurately track and predict the evolving boundary of a continuous object using sparse local front estimates.

Fig. 11d shows the probability to have a successful boundary reconstruction event at each one of the considered time instances. We observe that as the local estimates density increases, the probability for a successful boundary reconstruction event also increases for all reconstruction time instances (dots on the curves of the Fig. 11d). This behavior is justified if we consider that the higher local estimates density implies more available local estimates at each reconstruction time instance which in turn increases the probability to have a sufficient number of local front estimates in $\mathcal{M}_b$ ($|\mathcal{M}_b| \geq N$ see Section III) which is necessary condition for a boundary reconstruction to be attempted.

Boundary tracking evaluation under orientation and speed estimation errors

In this experiment we evaluate the boundary tracking accuracy of the proposed algorithm under different local front orientation and percent speed error values. For each scenario (specific density and error values) the measured evolution parameters (orientation or speed) of each local front estimate are distorted by applying the selected error values (discussed in Section V-C). The distortion is implemented by adding to, or subtracting from, each estimate (randomly selected action) the chosen error value for the corresponding parameter.

Fig. 12a and 12b show for each angle and percent speed error value used, the average boundary tracking accuracy for all the local estimates density scenarios considered. For the 20^o angle error and the 20% speed error we also provide for each local estimates density value the corresponding standard deviations (error bar). The standard deviations for the other angle ($10^o, 30^o$) and percent speed errors ($10\%, 30\%$), are similar to these shown for the 20^o angle and 20% speed error respectively. For each case (e.g. 20 local estimates, 10 degrees angle error) the results are generated using as sample points the F1-score values estimated from all the boundary reconstructions that occurred using 1000 different (randomly selected) local estimates placements. The line plots in Fig. 12a and 12b indicate that as the angle and speed error increase, the boundary reconstruction accuracy decreases. Another observation is that the difference, between the boundary reconstruction accuracies achieved for each angle or percent speed error value, gets smaller as the number of local estimates increases. This behavior is justified if we consider that for higher densities, the mean distance that the local front segments (which participate in the boundary construction) have to ''travel'' as we project them to the future in order to predict the diffusive objects boundary is smaller (they continuously get replaced by more recent estimates). Traveling smaller distances with erroneous evolution parameters implies smaller boundary location errors and therefore higher boundary reconstruction accuracy. This also explains why the standard deviation drops as the local fronts' density increases for all the considered angle and speed error cases.

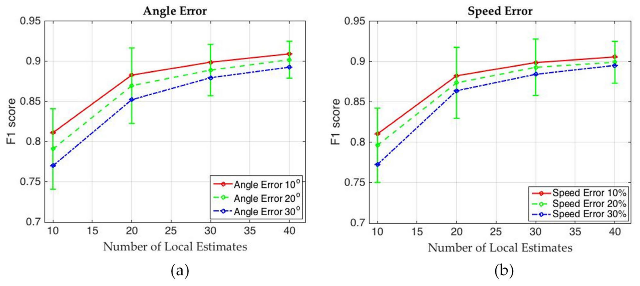


FIGURE 12. Boundary tracking accuracy (mean and standard deviation under a) angle errors, b) speed percent errors for different local front estimate densities.

Probabilistic boundary tracking of a continuous object

As discussed in Section II, the evolution characteristics (orientation and speed) of the local fronts $\{m_i\}$ that participate in the boundary's reconstruction algorithm are described by scalar values $\{\hat{\phi}_i, \hat{u}_i\}$ randomly sampled from the corresponding Normal distributions (for Φ_i and U_i). This random sampling implies that if we run the boundary reconstruction algorithm several times, we will get similar but not identical boundary shapes. Based on this observation we developed a technique which allows us to consider the area occupied by a continuous object as a field of probabilities where every point has a probability to be affected by the continuous object. To calculate the probability field we do the following:

We consider the continuous object's evolution area as a grid of square cells. A cell is assumed to be affected by the evolving continuous object when its center is located inside the object's area. To calculate the probability for a cell C to belong to the continuous object at a specific time instance t_b, we run the boundary reconstruction algorithm n times and count the number n_C of simulation scenarios that cell C is located inside the object's area. The probability for the continuous object to arrive at cell C can then be estimated as:

$$P(C) = \frac{n_C}{n}. \tag{6}$$

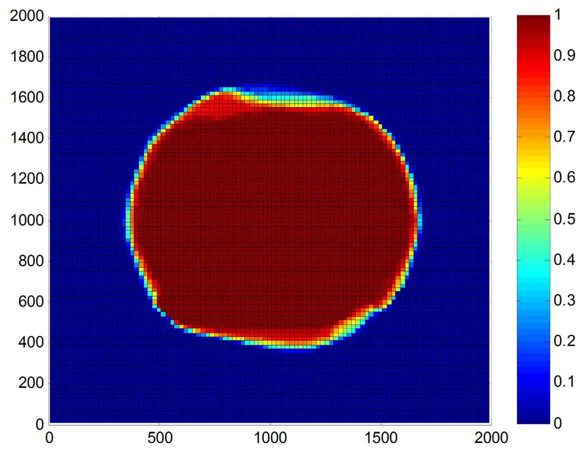


FIGURE 13. **Probability of circular continuous object (experiment 1) affecting a cell of the considered area at time $t_b = 188min$. Each color correspond to different coverage probability as indicated by the colorbar on the right.**

Fig. 13 shows for experiment 1 the probability field at $t_b = 188min$ that produced after running 1000 times the boundary reconstruction algorithm of a specific local fronts' placement scenario with 20 local fronts per km^2. As observed, the proposed boundary reconstruction algorithm exploits the uncertainties of evolution parameters of local fronts to characterize the uncertainty about the continuous object's boundary location. In this example the used angle and speed uncertainties of the local front segments in Φ_i and U_i were set to $\sigma_i = 10^o$ degrees and $s_i = 10\%$ respectively.

E. DIFFUSIVE HAZARDS WITH IRREGULAR SHAPE

In Experiment 2 we evaluate the proposed boundary reconstruction algorithm using diffusive hazards with irregular evolution patterns (e.g. non-geometric front shapes, large propagation speed variations). To simulate such hazards with realistic characteristics we exploited FLogA [38], a web-based interactive tool which allows to draw a forest area anywhere in Europe over Google Earth [40] insert fire ignition points ("hotspots"), define wind direction and speed scenarios, and then simulate and geo-animate the evolving wildfire.

Using FLogA [38] we defined a square forest area of $3km \times 3km$ at Hymettus mountain in Attica Greece and simulated a wildfire scenario. The fire was initiated from a single ignition point which was placed at the middle of the considered $9km^2$ forest area. The wind orientation and speed parameters were fixed within the forest area and their values were set to 0^o (with respect to the x-axis) and $2m/s$ respectively. Fig. 14 show four snapshots that help us visualize on Google Earth the ignition point as well as the corresponding part of the forest area that has been affected by the fire (continuous object is shown with red color) 50, 100 and 150 minutes after the ignition respectively.

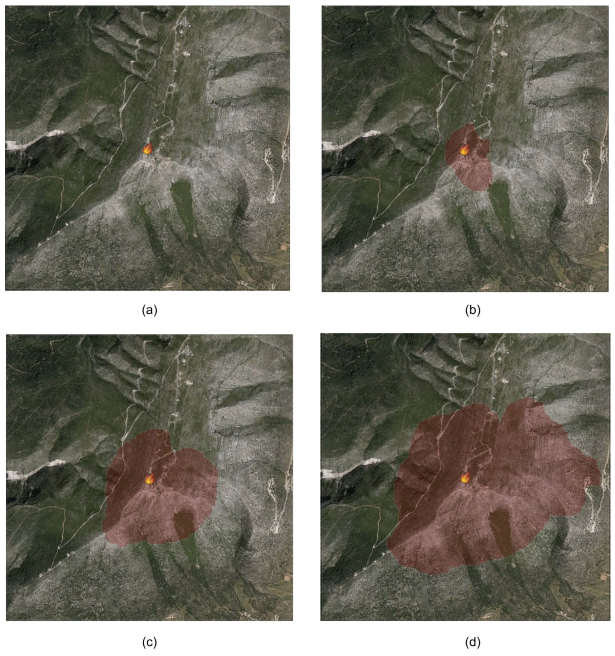


FIGURE 14. **a) The wildfire's ignition point, located at the center of the considered $9km^2$ square forest area. Snapshots (b), (c), (d) show the parts of the forest area that have been affected by the fire (red area) 50, 100, 150 minutes after the ignition.**

In Fig. 15 and 16 we present a summary of the results for experiment 2 following the same format as for Fig. 11 and 12 respectively.

Similarly to Experiment 1, we evaluated the proposed boundary reconstruction algorithm under different local front: a) densities ($\{10, 20, 30, 40\}$ estimates per $1km^2$), b) placements (1000 random placements), and c) orientation and percent speed error scenarios ($\{10^o, 20^o$ and $30^o\}$ and $\{10\%, 20\%$ and $30\%\}$ respectively.

In summary, in this experiment the mean boundary reconstruction accuracy (F1-scores) was on average smaller by 3.91% (standard deviation 1.15%) relatively to Experiment 1 considering the results from all simulation scenarios. This accuracy drop is justified considering the complex evolution behavior of the wildfire's front line (irregular accelerations/decelerations) which makes the boundary tracking a lot more difficult.

Similarly to Experiment 1 we have created a MATLAB animation [32] that is provided online and as supplementary material that shows how the proposed

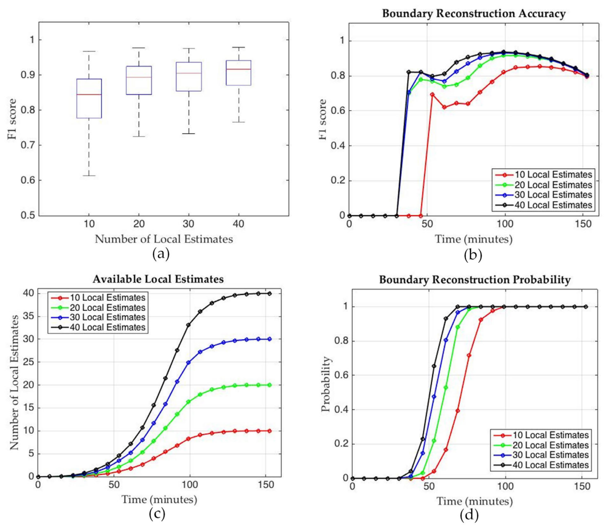


FIGURE 15. **Wildfire simulation (Experiment 2): For different local front estimate densities: a) Boxplots summarizing the distribution of the boundary tracking algorithm's accuracy. b) The average boundary reconstruction accuracy as a function of time. c) The mean number of the local front estimates available to participate at each time instance to the boundary reconstruction algorithm. d) The probability of successive boundary reconstruction as a function of time.**

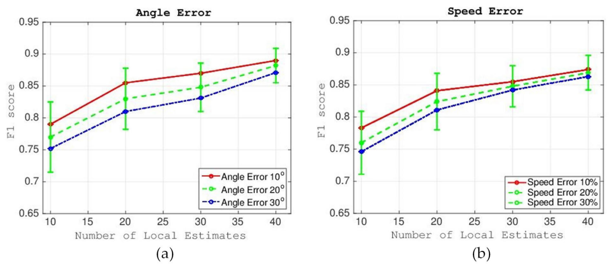


FIGURE 16. **Wildfire simulation (Experiment 2): The average boundary tracking accuracy and the standard deviation (error bars) under different local front estimates densities and (a) angle errors, (b) speed percent errors.**

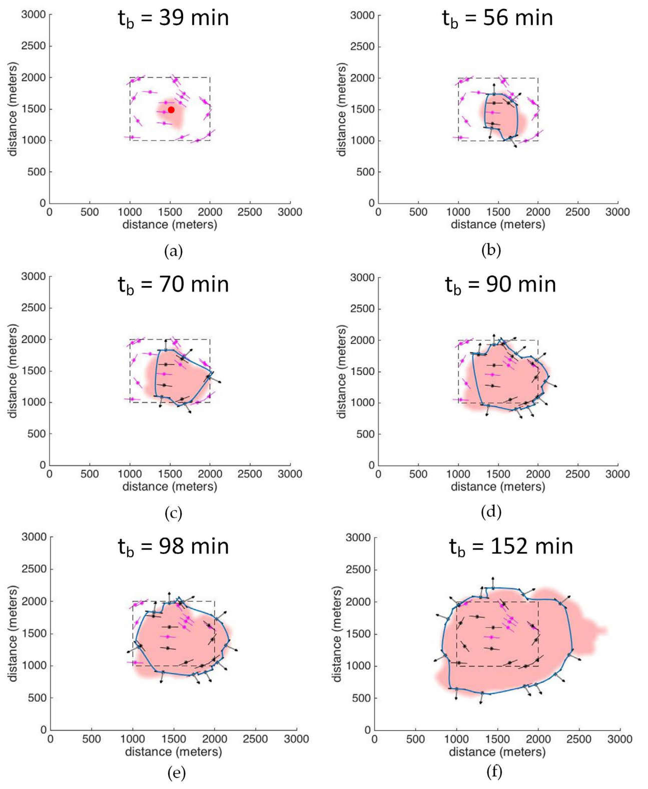


FIGURE 17. **Video frame snapshots: Panels (a) through (f) show 6 boundary reconstruction snapshots for the wildfire simulation of Experiment 2. Line segments represent the available local front estimates. With black color we indicate the local front estimates used in the boundary reconstruction. The blue curve represents the reconstructed boundary. The full video is available at [32].**

algorithm can track accurately evolving continuous objects with complex boundary shapes and evolution behavior (Fig. 17). In Fig. 17a we can see the area affected by an evolving wildfire (pink area), the wildfire's ignition point (red dot), the 20 randomly placed local front estimates (magenta line segments) and their placement area (dashed square area of $1km^2$). The simulation terminates when the wildfire covers the local fronts' placement area (152min after the ignition).

A local front model estimate gets activated and participates in the boundary reconstruction when its mid-point is reached by the wildfire's front line. With black color (see Fig. 17b - 17f) we depict the set of the local front segments that participate (after the selection/fusion procedure, see Section III) to the boundary reconstruction algorithm. Based on local fronts evolution characteristics, we determine their space-time propagation (black segments on the blue curve, see also Appendix B) and using the proposed boundary reconstruction algorithm (see Section IV) we reconstruct the "smooth" B-spline curve (blue curve) that approximates the boundary of the wildfire. Fig. 17b - 17f, demonstrate the ability of the proposed algorithm to track continuously with accuracy the irregular shapes of the boundaries which result from wildfire's complex evolution behavior, despite the very small number of the local fronts estimates used (only 20 in $9km^2$). Notice that the boundary tracking algorithm underestimates the wildfire's boundary only in regions where no local front estimates are available (see Fig. 17b - 17e). Furthermore, in Fig. 17f the underestimation of the wildfire's boundary is expected if we observe that the wildfire's boundary is located outside the sensors placement area and therefore its tracking is based mostly on "outdated" and far away local front estimate projections.

In real hazard tracking applications the topography at the hazard's site can skew the placement of the local front estimates in the field. To evaluate the accuracy of our boundary reconstruction algorithm under this assumption, we repeated Experiment 2 using highly skewed placements of the local front estimates in the 1 km^2 square area (dashed square in Fig. 17) generated as follows:

We partition the 1 km^2 square deployment area into 4 equal square areas (0.25 km^2 each) by connecting the middle points of its opposite sides. Next we randomly select 2 of the 4 formed square areas and we randomly deploy there 80%

of the local front estimates used in the specific local front estimate density scenario. The rest 20% of the local front estimates are randomly deployed in the other 2 non-selected squares. This procedure allows us to generate highly skewed placements of the local front estimates in the original deployment area (1 km^2 square area).

Using the placements we evaluated the proposed boundary reconstruction algorithm under different local front densities ($\{10, 20, 30, 40\}$ estimates per $1km^2$). As expected, the boundary reconstruction accuracy was higher in the square areas where the 80% of the local fronts were placed. However, an interesting finding is that the average boundary reconstruction accuracies (F1-scores) did not significantly drop compared to the nominal Experiment 2. More specifically, for each local front estimate densities scenario (i.e. $\{10, 20, 30, 40\}$) the percentage reduction of the average boundary reconstruction accuracy was $\{5.98\%, 4.66\%, 2.8\%, 2.62\%\}$ respectively. The results clearly show that our boundary reconstruction algorithm is almost insensitive to the skewed placements of the local front estimates, and its sensitivity is further reduced as the density of the local front estimates increases.

F. COMPARISON TO CODA

As mentioned in the introduction Section, most of the reported WSN-based schemes use the locations of the boundary sensor nodes to implicitly determine the boundary of a continuous object. Therefore these schemes rely on a human expert in order to reconstruct the boundary curve of a continuous object. One notable exception is the work in [15] where the authors propose CODA - a Continuous Object Detection and Tracking Algorithm which approximates a continuous object's boundary using the convex hull polygon of the boundary node locations. For completeness we compared CODA to our method assuming the local front estimates density scenarios (10, 20, 30 and 40 sensor nodes deployed within $1km^2$ and $9km^2$ square areas for Experiment 1 and Experiment 2 respectively). For all density scenarios, CODA failed to track with accuracy the continuous object's boundary. The accuracy (F1-score) of CODA for Experiment 1 and Experiment 2 were on average smaller by 50.71% and 59.36% respectively, relatively to the proposed boundary reconstruction scheme. We remark that these results were extracted using for each density a large number (1000) of randomly drawn sensor node deployments.

We also repeated the aforementioned experiments with very high density sensor networks with 5×10^{-4}, 10^{-3}, 2×10^{-3} and 3×10^{-3} $\frac{sensors}{km^2}$ corresponding to 500, 1000, 2000 and 3000 sensor nodes respectively deployed within a square area of $1km^2$. Even for these very high densities, the accuracy (F1-score) achieved by CODA for Experiment 1 and Experiment 2 were on average smaller by 1.55% and 7.79% percent, relatively to the proposed boundary reconstruction scheme. The smaller boundary reconstruction accuracy of CODA in this case can be explained if we consider its inability to predict the location of the boundary when the object moves beyond the sensor nodes deployment area (see animations [31], [32] and Fig. 10f, 17f). Moreover, the larger decrease of the boundary reconstruction accuracy for Experiment 2, can be justified if we consider that CODA can form only convex boundary shapes. Therefore, the more realistic non-convex shapes formed for example as a wildfire evolves (see Fig. 17f and animation [32]) cannot be tracked accurately using CODA.

VI. CONCLUSION

We have presented a novel approach which allows us to track accurately the boundary of an evolving continuous object using only sparse in space and time estimates of its local front characteristics provided by distributed sensors. Our continuous object tracking scheme has the following unique characteristics relatively to prior schemes: a) It first filters and fuses the information of the available local front estimates, and b) then uses the surviving subset of estimates to reconstruct a smooth curve approximating the evolving object's boundary; c) it requires a much smaller number of sensors to provide local front estimates than other methods; d) it supports predictive modeling of the boundary even in areas with limited number of sensors; e) it provides higher accuracy and remains robust to local front estimate errors. Moreover, by exploiting the inherent uncertainties associated with the local front estimates, it can provide a field representation of the continuous object, indicating for every location the probability to be reached by the continuous object at any given time. Realistic simulations demonstrate that the proposed method can track accurately the evolving boundaries of different types of continuous objects (e.g. with time-varying evolution characteristics and/or irregular shapes), using a realistic number of possibly noisy local estimates. Moreover, global boundary reconstruction remains robust to local front estimation errors.

We are currently investigating a method that will allow us to determine the number of continuous objects that may evolve simultaneously in an area of interest. Furthermore, we are developing methods to probabilistically associate local front estimates to their originating continuous object. Solving these challenging problems can help us extend the current framework allowing us to track the boundaries of multiple co-evolving hazards. Finally, we are working on the development of a novel method able to estimate for *every* boundary point (using the evolution characteristics of the sparse local front estimates) the following parameters i.e. latitude and longitude coordinates, speed, direction and acceleration. This information can be directly exploited by hazard-specific predictive models in order to re-calibrate their parameters and to improve their prediction accuracy.

APPENDIX A
COORDINATES OF LOCAL FRONT END POINTS

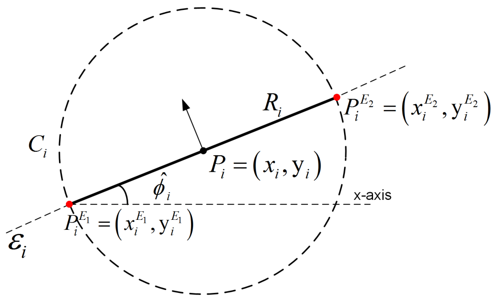
Using the mid-point coordinates $P_i = (x_i, y_i)$, the length $l_i = 2R_i$, and the orientation $\hat{\phi}_i$ of a local front estimate m_i, we can calculate the coordinates of its end-points,

$P_i^{Ez} = (x_i^{Ez}, y_i^{Ez})$ where $z = \{1, 2\}$, by solving the following system of equations:

$$\begin{cases} \varepsilon_i : y_i^{Ez} - y_i = tan(\hat{\phi}_i)(x_i^{Ez} - x_i) \\ C_i : (x_i^{Ez} - x_i)^2 + (y_i^{Ez} - y_i)^2 = R_i^2 \end{cases} \tag{7}$$

The first equation in (7) defines a line (ε_i in Fig. 18) on which the local front's line segment lies. The second equation defines a circle (C_i) of radius $R_i = \frac{l_i}{2}$, centered at the local front's segment mid-point P_i the assumed sensor location. The solution of (7) provides the following closed form algebraic expressions which are used for the computation of the local front's end-point coordinates (x_i^{Ez}, y_i^{Ez}), $z = \{1, 2\}$.

$$\begin{cases} x_i^{Ez} = x_i \pm \dfrac{R_i\sqrt{tan^2(\hat{\phi}_i) + 1}}{tan^2(\hat{\phi}_i) + 1} \\ y_i^{Ez} = y_i \pm \dfrac{R_i tan(\hat{\phi}_i)\sqrt{tan^2(\hat{\phi}_i) + 1}}{tan^2(\hat{\phi}_i) + 1} \end{cases} \tag{8}$$



C_i
R_i
$P_i^{E_2} = (x_i^{E_2}, y_i^{E_2})$
$P_i = (x_i, y_i)$
$\hat{\phi}_i$
x-axis
$P_i^{E_1} = (x_i^{E_1}, y_i^{E_1})$
ε_i

FIGURE 18. The local front estimate m_i centered at P_i; the coordinates of its end-points (red dots) are determined by the points line ε_i intersects circle C_i.

APPENDIX B
SPACE-TIME EVOLUTION OF A LOCAL FRONT ESTIMATE

Let's assume that we want to determine the location of a local front $m_i(t_i)$ (with parameters estimated at time t_i) at a future time instance t_j, where $t_j > t_i$.

Using the speed realization $\hat{u}_i$, we calculate the distance $\hat{d}_{ij}$ "traveled" by m_i if we assume that it "moves" for time interval equal to $\Delta t_{ij} = t_j - t_i$.

$$\hat{d}_{ij} = \hat{u}_i \Delta t_{ij} \tag{9}$$

Next, by solving the system of equations in (10) below, we can determine the mid-point's coordinates $P_i(t_j) = (x_i(t_j), y_i(t_j))$ of the space time projected local front estimate $m_i(t_j)$.

$$\begin{cases} \varepsilon_i : y_i(t_j) - y_i = \dfrac{-1}{tan(\hat{\phi}_i)}(x_i(t_j) - x_i) \\ C_i : (x_i(t_j) - x_i)^2 + (y_i(t_j) - y_i)^2 = \hat{d}_{ij}^2 \end{cases} \tag{10}$$

The first equation in (10) defines a line (ε_i in Fig. 19) which is perpendicular to the local front and emanates form its middle point $P_i = (x_i, y_i)$. The second equation defines a circle of radius $\hat{d}_{ij}$ centered at the local front's segment middle point P_i (see Fig. 19).

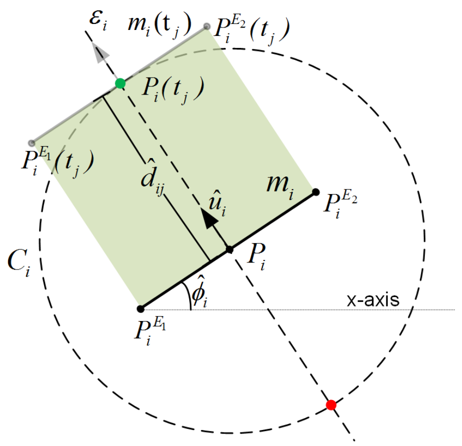


FIGURE 19. Space-time evolution of the local front $m_i(t_i)$ from time t_i of its generation to some future time t_j ($t_j > t_i$).

Solving this system of equations provides two closed form expressions for computing the intersection points of line ε_i and circle C_i (see Fig. 19). From these solutions we accept only the point that lies in the half plane indicated by the direction parameter δ_i (green point in Fig. 19)

$$\begin{cases} x_i(t_j) = x_i \pm \dfrac{\hat{d}_{ij} tan(\hat{\phi}_i)\sqrt{tan^2(\hat{\phi}_i) + 1}}{tan^2(\hat{\phi}_i) + 1} \\ y_i(t_j) = y_i \mp \dfrac{\hat{d}_{ij}\sqrt{tan^2(\hat{\phi}_i) + 1}}{tan^2(\hat{\phi}_i) + 1}. \end{cases} \tag{11}$$

$P_i(t_j) = (x_i(t_j), y_i(t_j))$, is the new location of the mid-point of $m_i(t_j)$. For the calculation of $m_i(t_j)$ end-point coordinates, $(P_i^{Ez}(t_j) = (x_i^{Ez}(t_j), y_i^{Ez}(t_j))$ where $z = \{1, 2\}$), we apply the equations (8) of Appendix A.

APPENDIX C
LOCAL FRONTS INFORMATION FUSION

We present the proposed information fusion technique applied to handle events of type 4 (see Section III, Step 2). This technique uses the information of two close by local front estimates $m_i(t_f)$ and $m_j(t_f)$ in Fig. 20 that satisfy the condition of event 4 to calculate the parameters of a new local front m_f capturing locally the evolution characteristics of the continuous object's boundary at time t_f.

The information fusion technique calculates for each local front estimate $m_k(t_f)$ (where $k \in \{i, j\}$), the time differences $\Delta t_k = t_f - t_k$. Next, using the parameters of the speed distributions U_k, it determines for each local front estimate m_k the distance D_k that it will cover if it moves for time interval equal to Δt_k. Since the speed U_k is described by a Normal distribution $\mathcal{N}(u_k, s_k^2)$, the distance D_k will also be described by Normal distribution $\mathcal{N}(d_k, \omega_k^2)$ with parameters

$$\begin{aligned} d_k &= u_k \Delta t_k \\ \omega_k &= s_k \Delta t_k. \end{aligned} \tag{12}$$

It is worth mentioning that as the time difference Δt_k increases, the uncertainty (ω_k) about the local front's (m_k)

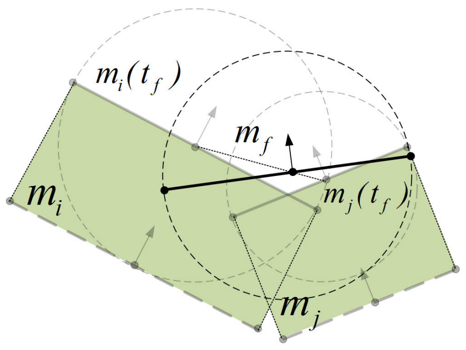


FIGURE 20. Information fusion of local fronts.

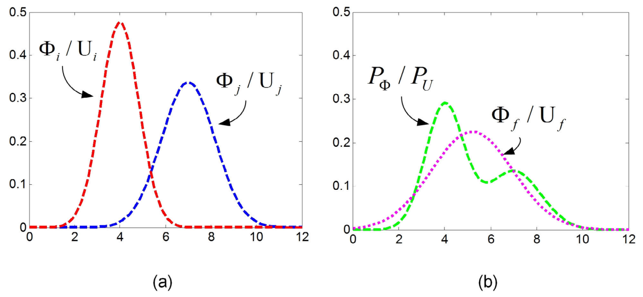


FIGURE 21. Estimating the orientation/speed (Φ_f/U_f) distributions of fusion model m_f. (a) The orientation/speed distributions of models m_i and m_j. (b) The mixture models P_Φ/P_U resulting by combining orientation/speed normal distributions of $\{m_i, m_j\}$ and the normal distribution Φ_f/U_f that best approximates P_Φ/P_U by minimizing the Kulback-Leibler divergence.

traveled distance will also increase. Next, we calculate for each local front m_k the distance $\hat{d}_k$, that it will cover if it moves with speed $\hat{u}_k$ for time interval Δt_k

$$\hat{d}_k = \hat{u}_k \Delta t_k. \tag{13}$$

Using the distribution for the orientation Φ_k and distance D_k, and the realization values $\hat{\phi}_k$ and $\hat{d}_k$, we calculate the weights (see equation (14)) that will be used for the calculation of the fused model m_f parameters

$$w_k = \frac{\Phi_k(\hat{\phi}_k) D_k(\hat{d}_k)}{\sum\limits_{l=\{i,j\}} \Phi_l(\hat{\phi}_l) D_l(\hat{d}_l)}. \tag{14}$$

For the calculation of the weights we consider that the local front estimate with the larger likelihood values $(\Phi_k(\hat{\phi}_k), D_k(\hat{d}_k))$ should be trusted more.

Using the weights w_k $k \in \{i, j\}$, we calculate the parameters of the new local front estimate m_f as follows

$$\mathcal{Z}_f = \sum_{k=\{i,j\}} w_k \mathcal{Z}_k \ \ where \ \ \mathcal{Z} \in \{\hat{u}_f, \hat{\phi}_f, l_f, x_f, y_f\}. \tag{15}$$

To estimate the evolution direction parameter δ_f we use the local fronts' orientations ($\hat{\phi}_k$), evolution directions (δ_k) and the weights (w_k) and determine two vectors which: a) are perpendicular to the corresponding local front segments, b) point towards the corresponding local fronts' evolution directions and c) the ratio of their lengths is equal to the corresponding ratio of their weights $\frac{w_i}{w_j}$. Next, using these vectors we calculate their resultant which determines the evolution direction of the new local front estimate m_f. Using the equation of the line where the new local front estimate m_f lies, we determine in which half plane (positive or negative) the vector of the resultant points to and assign the corresponding value (+1 or −1) to the direction parameter δ_f.

To calculate the speed (orientation) distribution of m_f we apply the following procedure:

Using the weights w_k $k \in \{i, j\}$ and the corresponding speed (orientation) distributions U_k, (Φ_k) of the local front estimates, we find their Gaussian mixture (see Fig. 21):

$$\begin{aligned} p_U(u) &= w_i \cdot \mathcal{N}(u|u_i, s_i^2) + w_j \cdot \mathcal{N}(u|u_j, s_j^2) \\ p_\Phi(\phi) &= w_i \cdot \mathcal{N}(\phi|\phi_i, \sigma_i^2) + w_j \cdot \mathcal{N}(\phi|\phi_j, \sigma_j^2) \end{aligned} \tag{16}$$

Next, by applying variational calculus we approximate the Gaussian mixture in (16) by the Normal distribution which minimizes the Kullback-Leibler (KL) divergence (maximizes the similarity) from the Gaussian mixture (see [41], [42]). The equations which can be used to compute the parameters of these Normal distributions are:

Speed parameters

$$u_f = w_i u_i + w_j u_j \tag{17}$$
$$s_f^2 = w_i s_i^2 + w_j s_j^2 + w_i w_j (u_i - u_j)^2 \tag{18}$$

Orientation parameters

$$\phi_f = w_i \phi_i + w_j \phi_j \tag{19}$$
$$\sigma_f^2 = w_i \sigma_i^2 + w_j \sigma_j^2 + w_i w_j (\phi_i - \phi_j)^2 \tag{20}$$

The parameters calculated from the above equations govern m_f's speed ($U_f \sim \mathcal{N}(u_f, s_f^2)$) and orientation angle ($\Phi_f \sim \mathcal{N}(\phi_f, \sigma_f^2)$) normal distributions after the fusion.

APPENDIX D INITIALIZATION OF LOCAL FRONT PARAMETERS

We explain here how we initialize the local front evolution parameters for the experiments presented in Section V-C, to simulate the measurements process of distributed sensors producing the local front estimates in the absence of errors (perfect sensing).

A. EXPERIMENT 1: CIRCULAR FRONT

Let's assume that the circular boundary of a continuous object reaches the mid-point (sensor location) of a local front segment (m_i) at time instance t_R (see Fig. 22a). We set the speed (u_i) parameter of m_i to the speed of the circle's radius at time t_R, and its orientation (ϕ_i) parameter to the angle formed between the x-axis and the circle's tangent line (ε_i in Fig. 22a) that osculates from the middle point of m_i.

B. EXPERIMENT 2: IRREGULAR FRONT

FLogA (Fire Logic Animator) is a web-based software tool which allows the user to define (draw) a forest area on Google Earth anywhere in Europe, insert interactively fire ignition points, simulate and geo-animate the behavior of the evolving

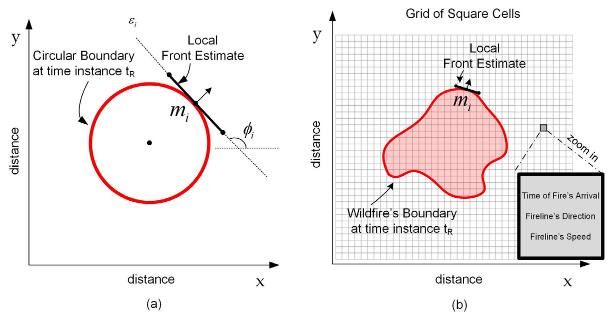


FIGURE 22. **Initalization of the local front parameters when a continuous object reaches a sensor location, (a) for a circular object and (b) for a more realistic non-convex object as a spreading wildfire.**

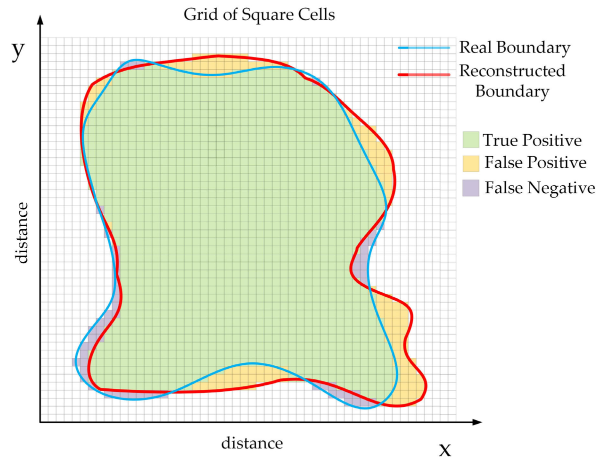


FIGURE 23. **The blue (red) curve corresponds to the real (reconstructed) boundary of the diffusing phenomenon. Classifying the square cells of the gridified area as shown in the figure, we can evaluate the accuracy of the proposed boundary reconstruction algorithm using the F1-Score.**

fire line under different weather conditions [38]. FLogA considers the forest area as a grid of square cells, with every cell experiencing different topography and weather conditions. The dimensions of the cells are defined by the user. To simulate the evolution behavior of a wildfire, FLogA accepts as input a set of raster ASCII files that contain information about the forest's topographic layers (slope, aspect, fuel model, fuel moisture), the prevailing weather conditions (wind speed and wind direction) in the forest's area, as well as the number and locations of the fire ignition points (''hotspots''). FLogA uses cellular automata like algorithms to predict for each cell of the grid information such as the expected time of fire's arrival, the fire line's speed, the fire line's evolution direction etc. (see Fig. 22b). We set the speed (u_i) and orientation (ϕ_i) parameters of a local front estimate m_i at a sensor location P_i, to be equal to the speed and evolution direction values of the hazard for the forest cell containing the mid-point of the line segment of m_i (sensor location).

APPENDIX E
ASSESSING SIMILARITY OF CONTINUOUS OBJECTS

To evaluate the accuracy of the proposed boundary reconstruction algorithm, we compare the similarity of the areas occupied by the real and the reconstructed continuous object as described below:

We consider the continuous object's evolution area as a grid of square cells (see Fig. 23). For all conducted experiments we set the size of the square cells to be $20m \times 20m$. Based on the real and reconstructed boundary of the continuous object, we classify the square cells to the following categories:

- True Positives (TP): Cells located inside the real and the reconstructed object.
- False Positives (FP): Cells located outside the real but inside the reconstructed object.
- False Negatives (FN): Cells located inside the real but outside the reconstructed object.

To estimate the boundary reconstruction accuracy we used the F1-Score which is the harmonic mean of *precision* and *recall*

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad where$$

$$precision = \frac{TP}{TP + FP}, \quad recall = \frac{TP}{TP + FN}.$$

APPENDIX F
TABLE OF SYMBOLS

TABLE OF SYMBOLS	
Symbols	**Definitions**
m_i	The *i-th* local front model.
$P_i = (x_i, y_i)$	The physical coordinates of the *i-th* line segment's mid-point.
$P_i^{Ez} = (x_i^{Ez}, y_i^{Ez}),\ z = \{1, 2\}$	The physical coordinates of the *i-th* line segment's end-points.
l_i	The length of the *i-th* line segment.
δ_i	The evolution direction of the *i-th* line segment.
$\Phi_i \sim \mathcal{N}(\phi_i, \sigma_i^2)$	The model for the orientation of the *i-th* line segment.
$\hat{\phi}_i$	A realization (random sample) of Φ_i.
$U_i \sim \mathcal{N}(u_i, s_i^2)$	The model for the speed of the *i-th* line segment.
$\hat{u}_i$	A realization (random sample) of U_i.
$C_i(t_j)$	The circular local area of model m_i at time t_j.
R_i	The radius of the circular area C_i.
R_i'	The adjusted radius of m_i circular area.
$A_i(t_j)$	The area covered by m_i during its space time evolution from time t_i to time t_j.
t_i	The time instance of m_i parameters estimation by a sensor located at P_i.
t_d	The earliest time instance when the evolution path of m_i is blocked by the evolution path of m_j.
t_f	The time instance where m_i and m_j fuse their information.
t_b	The time instance of a boundary reconstruction.
$D_k \sim \mathcal{N}(d_k, \omega_k^2),\ k = \{i, j\}$	The model for distance that $m_k(t_f)$ will cover if it moves for time interval equal to Δt_k.
$\hat{d_k}$	The distance that will be covered by m_k if it moves with the sample speed $\hat{u_k}$ for time interval Δt_k.
Δt_k	The time difference between t_f and t_k.
w_k	The weight used for the information fusion of the local front models m_i and m_j.
$\mathcal{Z}_f$	The set of m_f parameters which result after fusing the information of m_i and m_j.
$\mathcal{M}$	A set that contains the parameters of all the local front models.
$\mathcal{M}_b$	A set that contains the parameters of all the local front models selected to be used to reconstruct the boundary at time t_b.
$\mathcal{C}$	A set that contains the mid-points of all local front estimates in $\mathcal{M}_b$.
$\mathcal{V}$	An ordered set that contains the vertices of the convex hull polygon of $\mathcal{C}$.

$\mathcal{Q}$	A set that contains the mid-points in C that do not belong to the convex hull polygon.
$\mathcal{Q}_{on}$	A set that contains the points of $\mathcal{Q}$ located *on* the polygon's edges.
$\mathcal{Q}_{out}$	A set that contains the points of $\mathcal{Q}$ located *outside* the polygon's area.
$\mathcal{Q}_{out}^{I}$	A set that contains the points of $\mathcal{Q}_{out}$ that are centers of local front segments intersecting at least one edge of the polygon.
$\mathcal{Q}_{out}^{T}$	A set that contains the points of $\mathcal{Q}_{out}$ that are centers of local front segments located totally outside (do not intersect) the polygon.
$\mathcal{Q}_{in}$	A set that contains the points of $\mathcal{Q}$ located *inside* the polygon's area.
$\mathcal{Q}_{in}^{I}$	A set that contains the points of $\mathcal{Q}_{in}$ that are centers of local front segments intersecting at least one edge of the polygon.
$\mathcal{Q}_{in}^{T}$	A set that contains the points of $\mathcal{Q}_{in}$ that are centers of local front segments located totally inside (do not intersect) the polygon.
$P(C)$	The probability for the continuous object to arrive at cell C.
n_C	The number of simulation scenarios that cell C is located inside the object's area.
n	Total number of simulation scenarios.
N	The minimum number of local front estimates required for launching a boundary reconstruction.

ACKNOWLEDGMENT

The authors would like to thank Mr. Nikos Bogdos, University of Athens, for generating the wildfire scenario of Experiment 2 and for providing useful comments.

REFERENCES

[1] S. Duttagupta, K. Ramamritham, and P. Kulkarni, "Tracking dynamic boundaries using sensor network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 10, pp. 1766–1774, Oct. 2011.

[2] H. Hong, S. Oh, J. Lee, and S.-H. Kim, "A continuous object tracking protocol suitable for practical wireless sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 2351–2356.

[3] S. Imran and Y.-B. Ko, "A continuous object boundary detection and tracking scheme for failure-prone sensor networks," *Sensors*, vol. 17, no. 2, p. 361, Feb. 2017.

[4] H. Ping, Z. Zhou, T. Rahman, and Y. Duan, "Localization and tracking of continuous objects boundary area leveraging planarization algorithms in duty-cycled wireless sensor networks," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2017, pp. 8476–8481.

[5] Y. Zhang, Z. Wang, L. Meng, and Z. Zhou, "Boundary region detection for continuous objects in wireless sensor networks," *Wireless Commun. Mobile Comput.*, vol. 2018, May 2018, Art. no. 5176569.

[6] S. Park, S.-W. Hong, E. Lee, S.-H. Kim, and N. Crespi, "Large-scale mobile phenomena monitoring with energy-efficiency in wireless sensor networks," *Comput. Netw.*, vol. 81, pp. 116–135, Apr. 2015.

[7] S. Tarek, K. Shehryar, S. Elhadi, and M. Menshawi, "Continuous objects detection and tracking in wireless sensor networks," *J. Ambient Intell. Hum. Comput.*, vol. 7, pp. 489–508, May 2016, 10.1007/s12652-016-0380-5.

[8] H. Park, S. Oh, E. Lee, S. Park, S.-H. Kim, and W. Lee, "Selective wakeup discipline for continuous object tracking in grid-based wireless sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2012, pp. 2179–2184.

[9] Y. Xu, W. Bao, and H. Xu, "An algorithm for continuous object tracking in WSNs," in *Proc. Int. Conf. Res. Challenges Comput. Sci.*, Dec. 2009, pp. 242–246.

[10] C. Yang, Q. Li, and J. Liu, "A multisink-based Continuous Object Tracking in wireless sensor networks by GIS," in *Proc. 14th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2012, pp. 7–11.

[11] J. Kim, K. Kim, S. Chauhdary, W. Yang, and M. Park, "DEMOCO: Energy-efficient detection and monitoring for continuous objects in wireless sensor networks," *IEICE Trans. Commun.*, vol. 91, no. 11, pp. 3648–3656, Nov. 2008.

[12] G. Han, J. Shen, L. Liu, A. Qian, and L. Shu, "TGM-COT: Energy-efficient continuous object tracking scheme with two-layer grid model in wireless sensor networks," *Pers. Ubiquitous Comput.*, vol. 20, no. 3, pp. 349–359, Jun. 2016.

[13] 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 Annu. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, Sep. 2010, pp. 1725–1730.

[14] 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. IEEE 72nd Veh. Technol. Conf. (Fall)*, Sep. 2010, pp. 1–5.

[15] W.-R. Chang, H.-T. Lin, and Z.-Z. Cheng, "CODA: A continuous object detection and tracking algorithm for wireless ad hoc sensor networks," in *Proc. 5th IEEE Consum. Commun. Netw. Conf.*, Jan. 2008, pp. 168–174.

[16] Y. Wang, R. Tan, G. Xing, J. Wang, and X. Tan, "Accuracy-aware aquatic diffusion process profiling using robotic sensor networks," in *Proc. ACM/IEEE 11th Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2012, pp. 281–292.

[17] L. A. Rossi, B. Krishnamachari, and C.-C.-J. Kuo, "Distributed parameter estimation for monitoring diffusion phenomena using physical models," in *Proc. 1st Annu. IEEE Commun. Soc. Conf. Sensor Ad Hoc Commun. Netw. (SECON)*, Oct. 2004, p. 460.

[18] X. Yan, F. Gu, X. Hu, and S. Guo, "A dynamic data driven application system for wildfire spread simulation," in *Proc. Winter Simul. Conf. (WSC)*, Dec. 2009, pp. 3121–3128.

[19] T. Artes, A. Cencerrado, A. Cortes, T. Margalef, D. R. Aseretto, T. Petrolagkis, and J. S. M. Ayanz, "Towards a dynamic data driven wildfire behavior prediction system at European level," in *Proc. 14th ICCS*, vol. 29, 2014, pp. 1216–1226.

[20] R. L. Graham, "An efficient algorith for determining the convex hull of a finite planar set," *Inf. Process. Lett.*, vol. 1, no. 4, pp. 132–133, Jun. 1972.

[21] R. A. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Inf. Process. Lett.*, vol. 2, no. 1, pp. 18–21, Mar. 1973.

[22] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Trans. Inf. Theory*, vol. 29, no. 4, pp. 551–559, Jul. 1983.

[23] A. J. C. Moreira and Y. M. Santos, "Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points," in *Proc. Int. Conf. Comput. Graph. Theory Appl.*, 2007, pp. 61–68.

[24] J. S. Park and S. J. Oh, "A new concave hull algorithm and concaveness measure for n-dimensional datasets," *J. Inf. Sci. Eng.*, vol. 29, no. 2, pp. 379–392, 2013.

[25] S. Asaeedi, F. Didehvar, and A. Mohades, "Alpha-concave hull, a generalization of convex hull," *Theor. Comput. Sci.*, vol. 702, pp. 48–59, Mar. 2017.

[26] T. J. Cholewo and S. Love, "Gamut boundary determination using alpha-shapes," in *Proc. 7th Color Imag. Conf. (IS&T/SID)*, 1999, pp. 200–204.

[27] D. V. Manatakis and E. S. Manolakos, "Collaborative sensor network algorithm for predicting the spatiotemporal evolution of hazardous phenomena," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2011, pp. 3439–3445.

[28] D. V. Manatakis and E. S. Manolakos, "Predictive modeling of the spatiotemporal evolution of an environmental hazard and its sensor network implementation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 2056–2059.

[29] D. V. Manatakis and E. S. Manolakos, "Estimating the spatiotemporal evolution characteristics of diffusive hazards using wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 9, pp. 2444–2458, Sep. 2015.

[30] D. V. Manatakis, M. G. Nennes, I. G. Bakas, and E. S. Manolakos, "Simulation-driven emulation of collaborative algorithms to assess their requirements for a large-scale WSN implementation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 8360–8364.

[31] *Animation of Algorithm's Behavior in the Presence of Diffusive Hazard With Regular Shape (See Supplementary Material)*, IEEE, 2020.

[32] *Animation of Algorithm's Behavior in the Presence of Diffusive Hazard With Irregular Shape (See Supplementary Material)*, IEEE, 2020.

[33] N. Laiche and S. Larabi, "Retrieval of 2D objects and shape matching using the B-splines representation," in *Proc. IEEE Int. Conf. Signal Image Process. Appl. (ICSIPA)*, Kuala Lumpur, Nov. 2011, pp. 495–500.

[34] L. Stanberry and J. Besag, "Boundary reconstruction in binary images using splines," *Pattern Recognit.*, vol. 47, no. 2, pp. 634–642, Feb. 2014.

[35] H. Hang, X. Yao, Q. Li, and M. Artiles, "Cubic B-Spline curves with shape parameter and their applications," *Math. Problems Eng.*, vol. 2017, Dec. 2017, Art. no. 3962617.

[36] M. Sarfraz, M. Ishaq, and M. Z. Hussain, "Shape designing of engineering images using rational spline interpolation," *Adv. Mater. Sci. Eng.*, vol. 2015, Mar. 2015, Art. no. 260587.

[37] L. Piegl, W. Tiller, *The NURBS Book (Monographs in Visual Communication)*, 2nd ed. New York, NY, USA: Springer-Verlag, 1997.

[38] N. Bogdos and E. S. Manolakos, "A tool for simulation and geo-animation of wildfires with fuel editing and hotspot monitoring capabilities," *Environ. Model. Softw.*, vol. 46, pp. 182–195, Aug. 2013.

[39] U. S. Tristan. (Apr. 10, 2018). *The Diffusion Equation A Multi-dimensional Tutorial*. [Online]. Available: www.rpgroup.caltech.edu/~natsirt/aph162/diffusion.pdf

[40] (Apr. 10, 2018). *Google Earth*. [Online]. Available: http://www.earth.google.com

[41] A. Runalls, "Kullback-Leibler approach to Gaussian mixture reduction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 3, pp. 989–999, Jul. 2007.

[42] J. R. Hershey and P. A. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2007, p. IV-317.

DIMITRIS V. MANATAKIS (Senior Member, IEEE) received the M.Sc. degree (Hons.) in signal processing for communications and multimedia and the Ph.D. degree in informatics from the Department of Informatics and Telecommunications, University of Athens, Greece. He was a Postdoctoral Research Associate with the Department of Computational and Systems Biology, School of Medicine, University of Pittsburgh, PA, USA. He is currently an Associate Director in biomedical data science, Emulate Inc., Boston, MA, USA. He has played critical roles in many NIH and EU funded research projects. His Ph.D. research proposal was selected after the National and International Evaluation for Support from the Heracleitus II Grant, co-financed by the national and EU funds. His research interests include machine learning, causal inference, artificial intelligence, deep learning, bioinformatics, data mining, data fusion, statistical analysis, wireless sensor networks, computational geometry, signal processing, predictive modeling, dynamic systems modeling, and optimization and estimation theory. He received the Prize from the Greek State Scholarships Foundation after ranking among the top students in his class.

ELIAS S. MANOLAKOS (Senior Member, IEEE) received the Diploma degree in EE from the National Technical University of Athens, the M.Sc. degree from the University of Michigan, Ann Arbor, and the Ph.D. degree from the University of Southern California. He was a Visiting Scholar with the Wyss Institute of Biologically Inspired Engineering, Harvard University. He was also a tenured Faculty Member with the ECE Department, Northeastern University, Boston. He is currently a Professor with the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens. He is also a Visiting Professor with Northeastern University. He enjoys interdisciplinary research. He has played a leadership role in more than 20 funded research projects in EU and USA. He has authored or coauthored with his students and more than 130 publications in refereed journals and conference proceedings. His research interests include machine learning, signal processing, parallel and distributed computing, and their applications in biomedicine and environment. He was elected and served for two terms with the IEEE SPS Technical Committees on Machine Learning for Signal Processing and the Design and Implementation of Signal Processing Systems. He has served on the editorial board for several journals, such as the IEEE Transactions on Signal Processing, the IEEE Signal Processing Letters, the *Journal of Signal Processing Systems* (Springer), and so on.