NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

PhD DISSERTATION

Distributed Signal Processing and Data Fusion Methods for Large Scale Wireless Sensor Network Applications

Dimitrios V. Manatakis

European Union
European Social Fund

OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION & RELIGIOUS AFFAIRS
MANAGING AUTHORITY

NSRF 2007-2013
programme for development
EUROPEAN SOCIAL FUND

Co-financed by Greece and the European Union

ATHENS

NOVEMBER 2014

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Μέθοδοι Κατανεμημένης Επεξεργασίας Σήματος και Σύντηξης Δεδομένων για Εφαρμογές Ασυρμάτων Δικτύων Αισθητήρων Ευρείας Κλίμακας

Δημήτριος Β. Μανατάκης

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΑΘΗΝΑ

ΝΟΕΜΒΡΙΟΣ 2014

PhD DISSERTATION

Distributed Signal Processing and Data Fusion Methods for Large Scale Wireless Sensor Network Applications

Dimitrios V. Manatakis

ADVISOR: Elias Manolakos, Associate Professor, NKUA

THREE-MEMBER ADVISING COMMITTEE:
Elias Manolakos, Associate Professor NKUA
Efstathios Hadjiefthymiades, Associate Professor NKUA
Stelios Thomopoulos, Senior Researcher, NCSR “Demokritos

SEVEN-MEMBER EXAMINATION COMMITTEE

Elias Manolakos
Associate Professor NKUA

Efstathios Hadjiefthymiades
Associate Professor NKUA

Stelios Thomopoulos
Senior Researcher, NCSR “Demokritos"

Ioannis Emiris
Professor NKUA

Sergios Theodoridis
Professor NKUA

Evangelos Zervas
Professor T.E.I. of Athens

Gavriil Xanthopoulos
Associate Researcher HAO “DEMETER”

Examination Date: 27/11/2014

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Μέθοδοι Κατανεμημένης Επεξεργασίας Σήματος και Σύντηξης Δεδομένων για Εφαρμογές Ασυρμάτων Δικτύων Αισθητήρων Ευρείας Κλίμακας

Δημήτριος Β. Μανατάκης

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ηλίας Σ. Μανωλάκος Αναπλ. Καθηγητής ΕΚΠΑ

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:
Ηλίας Μανωλάκος Αναπλ. Καθηγητής ΕΚΠΑ
Ευστάθιος Χατζηευθυμιάδης Αναπλ. Καθηγητής ΕΚΠΑ
Στυλιανός Θωμόπουλος Ερευνητής Α, ΕΚΕΦΕ "ΔΗΜΟΚΡΙΤΟΣ"

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Ηλίας Μανωλάκος
Αναπλ. Καθηγητής ΕΚΠΑ

Ευστάθιος Χατζηευθυμιάδης
Αναπλ. Καθηγητής ΕΚΠΑ

Στυλιανός Θωμόπουλος
Ερευνητής Α, ΕΚΕΦΕ "ΔΗΜΟΚΡΙΤΟΣ"

Ιωάννης Εμίρης
Καθηγητής ΕΚΠΑ

Σέργιος Θεοδωρίδης
Καθηγητής ΕΚΠΑ

Ευάγγελος Ζέρβας
Καθηγητής Τ.Ε.Ι. Αθηνών

Γαβριήλ Ξανθόπουλος
Αναπλ. Ερευνητής ΕΛΓΟ
“ΔΗΜΗΤΡΑ”

Ημερομηνία Εξέτασης: 27/11/2014

Abstract

Tracking and predicting, with reasonable accuracy, the spatiotemporal evolution of diffusive hazardous phenomena (e.g. wildfires, oil slicks, chemical leaks, etc) is of paramount importance for civil authorities since it helps them to optimize their response and contain the potential damage. Hazard specific, mechanistic or semi-empirical methods are commonly used for this purpose. However these methods usually fail to make good predictions, due to the large number of time-varying parameters governing these complex phenomena. Recently significant efforts have been invested worldwide towards in the design and development of distributed systems for large scale environmental monitoring. The goal of such efforts is to predict, monitor and manage the consequences of diffusive hazards, often modeled as ``continuous objects'' (i.e. objects that tend to occupy large areas and their shape and size continuously changing with time). Wireless Sensor Network (WSN) is a mature technology which can play a major role in the development of continuous object tracking systems.

In this PhD dissertation we study the problem of continuous object tracking using large scale WSNs. We propose a novel practical WSN-based scheme that is able to track and predict the evolution behavior of a continuous object's boundary under realistic assumptions. The proposed scheme consists of two main components: a) A collaborative in-network WSN algorithm that estimates the local evolution parameters (orientation, direction and speed) of an evolving continuous object, and b) a novel algorithm which combines the produced local estimates, as they become available to a fusion center, to reconstruct the overall continuous object's boundary.

The proposed asynchronous WSN collaborative algorithm is based on the assumption that the boundary of a continuous object can be approximated as a piecewise linear curve. Each line segment (local front model) of this curve can be adequately characterized by a small set of parameters namely the orientation, angle, direction and speed of the segment's propagation. As the continuous objects boundary evolves these parameters are

re-estimated using ad-hoc formed clusters of collaborative sensor nodes. A flexible probabilistic sensing model that can capture the sensor nodes' detection distance uncertainty as well as their disruption probability is introduced which allow us to formulate the model parameters estimation problem in a Bayesian manner. We solve this estimation problem analytically and derive simple algebraic closed-form expressions that can be easily implemented by the energy constrained microprocessors of the sensor nodes. To further support our claim that the proposed collaborative algorithm is suitable for large-scale WSN deployment, we introduce a simulation-driven WSN emulation scheme which allows us to estimate, using small number of ``real'' sensor nodes, the energy, memory and processing requirement as a function of the the WSN's density.

When a small number of local fronts estimates becomes available to a fusion center, the proposed boundary reconstruction algorithm appropriately combines their information and determines a ``new'' set of the local fronts estimates which describe the continuous object's boundary evolution characteristics at a specific time instance. Using this information, the boundary reconstruction algorithm forms a simple polygon that approximates the shape of the boundary. Next, using the formed polygon and uniform B-splines curves, it determines a ``smooth'' approximation of the boundary. Finally, based on the local fronts' parameter estimation uncertainties the algorithm can produce a probability field that indicating for each point of the considered area the probability to be reached by the object's front line.

Extensive computer simulations demonstrate the ability of the proposed collaborative algorithm to estimate accurately the evolution characteristics of complex continuous objects (e.g. with time-varying evolution rates and/or irregular boundary shapes) using reasonably dense WSNs. Moreover, it shown that the algorithm is robust to sensor node failures and communication link failures which are expected in harsh environments. Finally, we show that the proposed boundary reconstruction algorithm is able to track with accuracy the evolution of different types of continuous objects, using a small number of local front estimates that may be distorted with error.

Subject Area: Distributed Signal Processing, Data Fusion, Sensor Networks.

Keywords: Machine Learning, distributed estimation, Bayesian estimation, continuous object tracking, environmental hazards, wireless sensor networks.

ΠΕΡΙΛΗΨΗ

Η ικανότητά παρακολούθησης και πρόβλεψης της χώρο-χρονικής εξέλιξης ενός διάχυτου καταστροφικού φαινομένου (π.χ. δασική πυρκαγιά, πετρελαιοκηλίδα, χημική διαρροή κτλ.) παρέχει καθοριστικής σημασίας πληροφορία στις αρμόδιες αρχές καθώς τις βοηθά να βελτιώσουν τις επιχειρήσεις καταστολής του φαινομένου καθώς και να περιορίσουν τις καταστροφικές του συνέπειες. Τα τελευταία χρόνια έχουν προταθεί πολλά μαθηματικά καθώς και εμπειρικά μοντέλα τα οποία προσπαθούν να προβλέψουν την χώρο-χρονική εξέλιξη διαφόρων καταστροφικών φαινομένων. Παρόλα αυτά, οι προβλέψεις αυτών των μοντέλων συνήθως αποκλίνουν από την πραγματικότητα, λόγω του ότι εξαρτώνται από πολλές δυναμικά μεταβαλλόμενες παραμέτρους οι οποίες είναι πολύ δύσκολο να εκτιμηθούν. Πρόσφατα, έχουν γίνει σημαντικές προσπάθειες για τον σχεδιασμό και την ανάπτυξη κατανεμημένων συστημάτων για παρακολούθηση μεγάλων γεωγραφικών περιοχών. Στόχος αυτών των τεχνικών είναι να προβλέψουν, να παρακολουθήσουν και να διαχειριστούν τις συνέπειες των διάχυτων καταστροφικών φαινομένων τα οποία αναφέρονται στη βιβλιογραφία ως «συνεχή αντικείμενα» (π.χ. αντικείμενα τα οποία καταλαμβάνουν μεγάλες γεωγραφικές περιοχές και το σχήμα τους αλλάζει κατά την εξέλιξή τους). Τα Ασύρματα Δίκτυα Αισθητήρων (ΑΔΑ) είναι μια ώριμη τεχνολογία η οποία μπορεί να διαδραματίσει σημαντικό ρόλο στην ανάπτυξη συστημάτων παρακολούθησης διάχυτων καταστροφικών φαινομένων.

Στην παρούσα διδακτορική διατριβή μελετάμε το πρόβλημα της παρακολούθησης συνεχών αντικειμένων χρησιμοποιώντας ευρείας κλίμακας δίκτυα αισθητήρων. Προτείνουμε μια καινοτόμο μέθοδο βασισμένη στην τεχνολογία των ΑΔΑ η οποία είναι ικανή να παρακολουθεί και να προβλέπει την συμπεριφορά εξέλιξης των χαρακτηριστικών εξέλιξης μετώπου ενός συνεχούς αντικειμένου κάτω από ρεαλιστικές υποθέσεις. Η προτεινόμενη μέθοδος αποτελείται από δύο κύρια συστατικά: α) Έναν συνεργατικό αλγόριθμο ΑΔΑ ο οποίος μπορεί και εκτιμά με ακρίβεια τις τοπικές παραμέτρους εξέλιξης (ταχύτητα και κατεύθυνση) του μετώπου ενός καταστροφικού φαινομένου και β) έναν αλγόριθμο ο οποίος, συνδυάζει τις τοπικές εκτιμήσεις όταν αυτές

γίνονται διαθέσιμες σε κάποιο κεντρικό σταθμό (fusion center) και ανακατασκευάζει το συνολικό μέτωπο του φαινομένου οποιαδήποτε στιγμή της εξέλιξης του επιθυμούμε.

Ο προτεινόμενος ασύγχρονος αλγόριθμος ΑΔΑ είναι βασισμένος στην υπόθεση ότι το μέτωπο ενός συνεχούς αντικειμένου μπορεί να προσεγγισθεί από μια τμηματικά συνεχή γραμμική συνάρτηση. Τα χαρακτηριστικά εξέλιξης κάθε γραμμικού τμήματος (μοντέλο τοπικού μετώπου) αυτής της συνάρτησης μπορούν επαρκώς να περιγραφούν από ένα μικρό σύνολο παραμέτρων. Καθώς το συνεχές αντικείμενο εξελίσσεται οι παράμετροι των τοπικών μοντέλων ενημερώνονται από μικρές δυναμικά σχηματιζόμενος ομάδες ασυρμάτων κόμβων. Η ενημέρωση των παραμέτρων βασίζεται σε ένα πρωτότυπο στοχαστικό μοντέλο ανίχνευσης του μετώπου από τον αισθητήρα, το οποίο μοντελοποιεί την αβεβαιότητα της απόστασης ανίχνευσής καθώς και την πιθανότητα καταστροφής του αισθητήρα από το φαινόμενο. Με τη χρήση του στοχαστικού μοντέλου ανίχνευσής, μπορέσαμε και διατυπώσαμε το πρόβλημα εκτίμησης των παραμέτρων εξέλιξης με ένα Μπαυσιανό τρόπο ο οποίος μας οδήγησε σε αναλυτικές αλγεβρικές λύσεις, οι οποίες μπορούν εύκολα να υλοποιηθούν από τους περιορισμένης επεξεργαστικής ικανότητας ασύρματους κόμβους του δικτύου. Για να ελέγξουμε αν ο αλγόριθμός μας μπορεί να υλοποιηθεί σε πραγματικό δίκτυο αισθητήρων ευρείας κλίμακας, σχεδιάσαμε ένα σύστημα εξομοίωσης ασυρμάτων δικτύων αισθητήρων ο οποίος μας επιτρέπει να εκτιμήσουμε χρησιμοποιώντας ένα μικρό αριθμό από πραγματικούς ασύρματους κόμβους τις απαιτήσεις του αλγορίθμου σε ενέργεια, επεξεργασία αλλά και σε μνήμη και να δούμε πως αυτές μεταβάλλονται συναρτήσει της πυκνότητας του δικτύου.

Όταν ένας μικρός αριθμός από τοπικές εκτιμήσεις (τοπικά μέτωπα) γίνει διαθέσιμος σε έναν κεντρικό κόμβο, ο προτεινόμενος αλγόριθμος ανακατασκευής του μετώπου, συνδυάζει την πληροφορία τους και καθορίζει ένα νέο σύνολο από τοπικές εκτιμήσεις οι οποίες περιγράφουν τα χαρακτηριστικά εξέλιξης του τοπικού μετώπου τη χρονική στιγμή που επιθυμούμε να γίνει η ανακατασκευή. Χρησιμοποιώντας την πληροφορία τους, ο αλγόριθμος ανακατασκευής του μετώπου, καθορίζει ένα πολύγωνο το οποίο προσεγγίζει το σχήμα του συνεχούς αντικειμένου. Στη συνέχεια, χρησιμοποιώντας το πολύγωνο και τις καμπύλες uniform B-splines, καθορίζει μια καμπύλη η οποία

προσεγγίζει «ομαλά» το μέτωπο του φαινομένου. Τέλος, βασιζόμενος στην αβεβαιότητα που έχουμε ως προς την εκτίμηση των παραμέτρων των τοπικών μετώπων, ο αλγόριθμος παράγει ένα χωρικό πεδίο πιθανότητας το οποίο περιγράφει για κάθε σημείο της περιοχής εξέλιξης την πιθανότητα να έχει καλυφθεί από το καταστροφικό φαινόμενο.

Μέσω ενός μεγάλου αριθμού προσομοιώσεων παρουσιάζουμε την ικανότητα του προτεινόμενου συνεργατικού αλγορίθμου να εκτιμά με ακρίβεια τα χαρακτηριστικά εξέλιξης πολύπλοκων συνεχών αντικειμένων (π.χ. με χρονικά μεταβαλλόμενα χαρακτηριστικά εξέλιξης καθώς και σχήματα μετώπου) χρησιμοποιώντας ρεαλιστικής πυκνότητας δίκτυα αισθητήρων. Επιπλέον, δείχνουμε ότι ο αλγόριθμός μας είναι εύρωστος σε αστοχίες που μπορεί να συμβούν στους κόμβους κατά την επικοινωνία τους ή/και λόγω καταστροφής τους από το πέρασμα του καταστροφικού φαινομένου. Τέλος, μέσω πειραματικών αποτελεσμάτων αποδεικνύουμε ότι ο προτεινόμενος αλγόριθμος ανακατασκευής του μετώπου μπορεί να παρακολουθεί με ακρίβεια την εξέλιξη διαφόρων τύπων συνεχών αντικειμένων, χρησιμοποιώντας ένα μικρό αριθμό από τοπικές εκτιμήσεις στις οποίες μπορεί να υπεισέρχονται σφάλματα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Κατανεμημένη Επεξεργασία Σήματος, Σύντηξη Δεδομένων, Δίκτυα αισθητήρων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μηχανική μάθηση, κατανεμημένη εκτίμηση παραμέτρων, Μευσιανή εκτίμηση, παρακολούθηση συνεχών αντικειμένων, κατανεμημένοι αλγόριθμοι, περιβαλλοντικές καταστροφές, ασύρματα δίκτυα αισθητήρων.

Acknowledgments

First and foremost, I would like to express my deep gratitude to my research advisor, Assoc. Prof. Elias S. Manolakos for offering me the opportunity to pursue my PhD degree at the University of Athens. I am really grateful for his continuous guidance and support throughout these years. Apart from the scientific advice and several insightful discussions we had, I am particularly grateful for the opportunity he gave me to conduct research with the required level of freedom. With his deep knowledge on the fields of signal processing, machine learning, distributed systems, and of course his endless support, working with him was a real pleasure. I am really proud of being his student. Also, I would like to express my gratitude to Dr. Gavriil Xanthopoulos (Associate Researcher at Hellenic Agricultural Organization ``DEMETER''), for his guidance and scientific advice. Dr. Xanthopoulos was the link with the environmental hazards community and provided me an insight on how to combine environmental hazards modeling with signal processing and computer science. Moreover, I also want to express my gratitude to Assoc. Prof. Efstathios Hadjiefthymiades and to Prof. Ioannis Emiris for their guidance and fruitful discussions on the fields of Wireless Sensor Networks and Computational Geometry respectively.

In addition, I am grateful for the scientific advice that Prof. Sergios Theodoridis has offered me during my PhD research. I feel very lucky for having the opportunity to attend Prof. Sergios Theodoridis courses since they have inspired me to pursuit a PhD degree on the field of Signal Processing. Due thanks also go to the rest of my dissertation committee: Dr. Stelios Thomopoulos, and Prof. Evangelos Zervas.

Special thanks go to Mr. Nickolaos Bogdos, Mr. Michael Nennes, and Mr. Ioannis Bakas, for helping me to realistically evaluate the WSN-based continuous object tracking scheme which proposed in this dissertation. I would also like to thank the current and former members of the lab: Dr. Panagiotis Tsakanikas, Dr. Symeon Chouvarda, Dr. Yan-

nis Kopsinis, and PhD candidate Evangelos Logaras. I could not forget my good friends Mr. Dimitrios Deves and Mr. Ioannis Manolopoulos for their endless encouragement and support throughout these years.

I would like to dedicate this dissertation to Vasilios and Theodora, Maria and Petty for always being there for me.

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek National funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: ``Heracleitus II''. Investing in knowledge society through the European Social Fund. I am grateful to the supporting agency that provided the resources needed to carry out competitive research at very difficult times for the country and its junior researchers.

List of Publications

Refereed Journal Publications

- **D. V. Manatakis**, E. S. Manolakos, ``Estimating the Spatiotemporal Evolution Characteristics of Diffusive Phenomena using Wireless Sensor Networks." *IEEE Transactions on Parallel and Distributed Systems*, Sept. 2014, Accepted for publication. DOI: 10.1109/TPDS.2014.2357033., in press.

Refereed Conference Publications

- **D. V. Manatakis**, M. G. Nennes, I. G. Bakas, E. S. Manolakos, Simulation-Driven Emulation of Collaborative Algorithms to Assess their Requirements for a Large-Scale WSN Implementation. In *Proc. IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP 2014)*, Florence-Italy, pages 8360 - 8364.

- **D. V. Manatakis**, E. S. Manolakos Collaborative Sensor Network Algorithm for Predicting the Spatiotemporal Evolution of Hazardous Phenomena. In *Proc. IEEE Systems, Man and Cybernetics (SMC 2011)*, (special session: Collaborative Wireless Sensor Networks), Anchorage-Alaska, October 2011, pp. 3439-3445.

- **D. V. Manatakis**, E. S. Manolakos Predictive Modeling of the Spatiotemporal Evolution of an Environmental Hazard and its Sensor Network Implementation. In *Proc. IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP 2011),* Prague-Czech, May 2011, pp. 2056-2059.

Journal Publication in Preparation

- **D. V. Manatakis**, E. S. Manolakos, ``Predictive Tracking of a Continuous Object Boundary using Sparse Local Estimates of its Evolution Characteristics." *IEEE*

Transactions on Systems Man and Cybernetics, in preparation for submission, Dec. 2014.

Συνοπτική Παρουσίαση της Διδακτορικής Διατριβής

Τα ασύρματα δίκτυα αισθητήρων (Wireless Sensor Networks - WSN), είναι μια ραγδαία αναπτυσσόμενη τεχνολογία με μεγάλο εύρος εφαρμογών (π.χ. παρακολούθηση στόχων, περιβαλλοντικών φαινομένων, ασθενών κτλ.). Τα WSN αποτελούνται συνήθως από ένα μεγάλο αριθμό αυτόνομων ηλεκτρονικών συσκευών (αισθητήριων κόμβων) χαμηλού κόστους οι οποίες τοποθετούνται σε μεγάλες γεωγραφικές περιοχές για να καταγράψουν τις τιμές φυσικών ή περιβαλλοντικών παραμέτρων. Εκτός από το να «αισθάνονται» το περιβάλλον, οι αισθητήριοι κόμβοι έχουν την δυνατότητα να επεξεργάζονται δεδομένα και να ανταλλάσσουν (ασύρματα) πληροφορία. Οι πρόσφατες εξελίξεις στους τομείς της μικροηλεκτρονικής και των ασύρματων επικοινωνιών καθιστούν την τεχνολογία των ασυρμάτων δικτύων αισθητήρων ιδανική υποψήφιο για την ανάπτυξη κατανεμημένων εφαρμογών ευρείας κλίμακας με δυνατότητες επεξεργασίας πληροφορίας και λήψης αποφάσεων.

Η παρακολούθηση στόχων (π.χ. ο καθορισμός της θέσης τους συναρτήσει του χρόνου) αποτελεί ένα ενδιαφέρον πρόβλημα το οποίο έχει μελετηθεί εκτενώς μιας και βρίσκει πολλές εφαρμογές (π.χ. στρατιωτικές, περιβαλλοντικές κτλ.). Εκτός από τον καθορισμό της τροχιάς των κινούμενων στόχων, είναι πολύ σημαντικό να μπορούμε να εκτιμούμε και τα χαρακτηριστικά εξέλιξής τους (π.χ. διεύθυνση και ταχύτητα) σε πραγματικό χρόνο, καθώς αυτή η πληροφορία μπορεί να χρησιμοποιηθεί για να προβλέψουμε τις μελλοντικές θέσεις τους και να καταλάβουμε την συμπεριφορά εξέλιξής τους.

Τα ασύρματα δίκτυα αισθητήρων έχουν ευρέως χρησιμοποιηθεί για την παρακολούθηση στόχων (ενός ή πολλαπλών). Λόγω του συνεχώς μειωμένου κόστους τους γίνονται όλο και πιο δημοφιλή σε εφαρμογές περιβαλλοντικής παρακολούθησης. Πρόσφατα, πολλές τεχνικές βασισμένες στην τεχνολογία των ασυρμάτων δικτύων αισθητήρων έχουν προταθεί για την ανίχνευση και παρακολούθηση διάχυτων καταστροφικών φαινομένων (όπως δασικές πυρκαγιές, πετρελαιοκηλίδες, διάχυση βιοχημικών υλών κτλ.), τα οποία μοντελοποιούνται ως "συνεχή αντικείμενα" (δηλαδή

αντικείμενα που αλλάζουν το μέγεθος και το σχήμα τους με την πάροδο του χρόνου). Η ικανότητα να παρακολουθούμε και να προβλέπουμε, με ικανοποιητική ακρίβεια, την θέση των διάχυτων καταστροφικών φαινομένων, είναι μείζονος σημασίας μιας και βοηθά τις αρμόδιες αρχές να οργανώνουν αποδοτικά τις επιχειρήσεις τους (π.χ. για τη καταστολή του φαινομένου, την αποτελεσματική εκκένωση περιοχών κτλ). Ωστόσο, οι κλασικοί αλγόριθμοι παρακολούθησης στόχων δεν μπορούν να χρησιμοποιηθούν για την παρακολούθηση συνεχών αντικειμένων μιας και τα δύο προβλήματα έχουν θεμελιώδεις διαφορές. Πιο συγκεκριμένα, τα συνεχή αντικείμενα καταλαμβάνουν συνήθως μεγάλες γεωγραφικές περιοχές και το μέγεθός και το σχήμα τους αλλάζει δυναμικά με το χρόνο. Αντίθετα οι “απλοί” διακριτοί στόχοι (όπως οχήματα, άνθρωποι, ζώα κτλ.) έχουν μικρό μέγεθος σε σχέση με την έκταση που καλύπτει το δίκτυο και συνήθως ένας μικρός αριθμός από κόμβους επαρκεί για να ανιχνεύσουμε την τροχιά τους.

Η βασική ιδέα των τεχνικών παρακολούθησης του μετώπου συνεχών αντικειμένων, που βασίζονται στην τεχνολογία των ασυρμάτων δικτύων αισθητήρων, είναι η επιλογή των κόμβων του δικτύου που βρίσκονται πλησιέστερα στο μέτωπο του καταστροφικού φαινομένου όσο αυτό εξελίσσεται. Παρόλο που αυτές οι τεχνικές επιτυγχάνουν να ανιχνεύσουν τα όρια του συνεχούς αντικειμένου εμμέσως από τις θέσεις των επιλεγμένων κόμβων, έχουν σημαντικά μειονεκτήματα τα οποία καθιστούν προβληματική τη χρήση τους στην ανάπτυξη πραγματικών συστημάτων παρακολούθησης συνεχών αντικειμένων.

Τα βασικότερα μειονεκτήματα που εμφανίζονται σε σχεδόν όλες τις προτεινόμενες τεχνικές παρακολούθησης συνεχών αντικειμένων στη σύγχρονη βιβλιογραφία είναι:

1. Η απαίτηση για χρήση δικτύων εξωπραγματικής πυκνότητας (χιλιάδες κόμβοι ανά τετραγωνικό χιλιόμετρο). Παρόλο που το κόστος των αισθητήριων κόμβων έχει μειωθεί σημαντικά, η κάλυψη μεγάλων περιοχών με δίκτυα υψηλής πυκνότητας παραμένει ακόμα απαγορευτική.

2. Η απουσία θεώρησης της πιθανότητας αστοχίας των κόμβων, καθώς και της μεταξύ τους επικοινωνίας. Ωστόσο, σε ένα πραγματικό δίκτυο αισθητήρων, οι

αστοχίες θεωρούνται βέβαιες ότι θα συμβούν, ιδιαίτερα όταν αυτό βρίσκεται τοποθετημένο στα αφιλόξενα περιβάλλοντα που δημιουργούνται από τα εξελισσόμενα καταστροφικά φαινόμενα.

3. Η απαίτηση ύπαρξης συγχρονισμού μεταξύ των κόμβων του δικτύου, η οποία είναι δύσκολο να επιτευχθεί ακόμα και σε χαμηλής πυκνότητας ασύρματα δίκτυα αισθητήρων.

4. Η θεώρηση ιδανικού μηχανισμού ανίχνευσης του φαινομένου (π.χ. ανίχνευση του φαινομένου σε συγκεκριμένη απόσταση), καθώς και η απουσία θεώρησης διακοπής της λειτουργίας ανίχνευσής των κόμβων λόγω καταστροφής των αισθητήρων τους.

5. Η αδυναμία παροχής πληροφορίας σχετικά με τα χώρο-χρονικά χαρακτηριστικά εξέλιξης του μετώπου του συνεχούς αντικειμένου. Αυτή η αδυναμία καθιστά τις εν λόγω τεχνικές ακατάλληλες για την ανάπτυξη συστημάτων πρόβλεψης.

6. Η αδυναμία τους να αποτιμήσουν με ακρίβεια τις ενεργειακές, επεξεργαστικές και επικοινωνιακές απαιτήσεις του δικτύου πριν την τελική τοποθέτηση του στην περιοχή ενδιαφέροντος.

7. Η αδυναμία τους να ανακατασκευάσουν με αυτοματοποιημένο τρόπο τα όρια του μετώπου κατά την εξέλιξη του καταστροφικού φαινομένου και να παρέχουν συντελεστές εμπιστοσύνης για τις διαφορετικές περιοχές του μετώπου που ανακατασκευάστηκε.

Στη παρούσα διδακτορική διατριβή, σχεδιάσαμε και αναπτύξαμε ένα σύστημα βασισμένο στην τεχνολογία τον ασύρματων δικτύων αισθητήρων το οποίο ξεπερνά *όλους* τους προαναφερθέντες περιορισμούς.

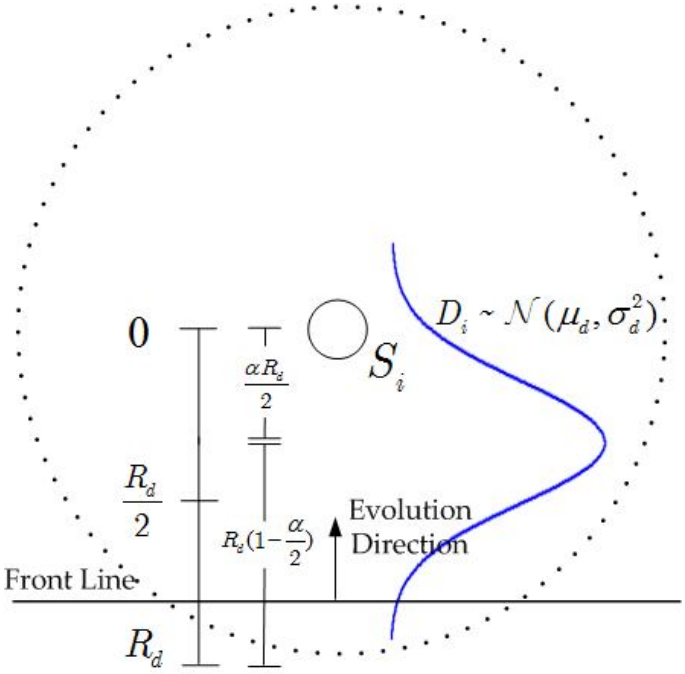
Στοχαστικό Μοντέλο Ανίχνευσης του Μετώπου – Πρόβλημα Εκτίμησης των Παραμέτρων Εξέλιξης Ενός Συνεχούς Αντικειμένου.

Αναπτύξαμε ένα καινοτόμο πιθανοτικό μοντέλο ανίχνευσης του φαινομένου το οποίο μοντελοποιεί α) την αβεβαιότητα που έχει ο αισθητήριος κόμβος ως προς την

απόσταση ανίχνευσης του φαινομένου, καθώς και β) την πιθανότητα αδυναμίας ανίχνευσης του λόγω αστοχίας του αισθητήρα. Όπως παρατηρούμε από το Σχήμα 1 η αβεβαιότητα ως προς την απόσταση ανίχνευσης μοντελοποιήται από μια κανονική κατανομή $D_i \sim \mathcal{N}(\mu_d, \sigma_d^2)$

με παραμέτρους: $\mu_d = \frac{\alpha R_d}{2}, \quad \sigma_d = \frac{R_d}{3}(1-\frac{\alpha}{2}) \quad \text{where } 0 \le \alpha \le 1$

Η οποία δίχνει ότι η πιθανότητα ανίχνευσης αυξάνει μονότονα καθώς η απόσταση του τοπικού μετώπου από τον αισθητήρα μειώνεται στο διάστημα $\left[\frac{\alpha R_d}{2}, R_d\right]$. Ενώ ελλατώνεται όταν το τοπικό μέτωπο κινήται στο διάστημα $\left[0, \frac{\alpha R_d}{2}\right]$ λόγω πιθανής καταστροφής του αισθητήρα από το φαινόμενο.

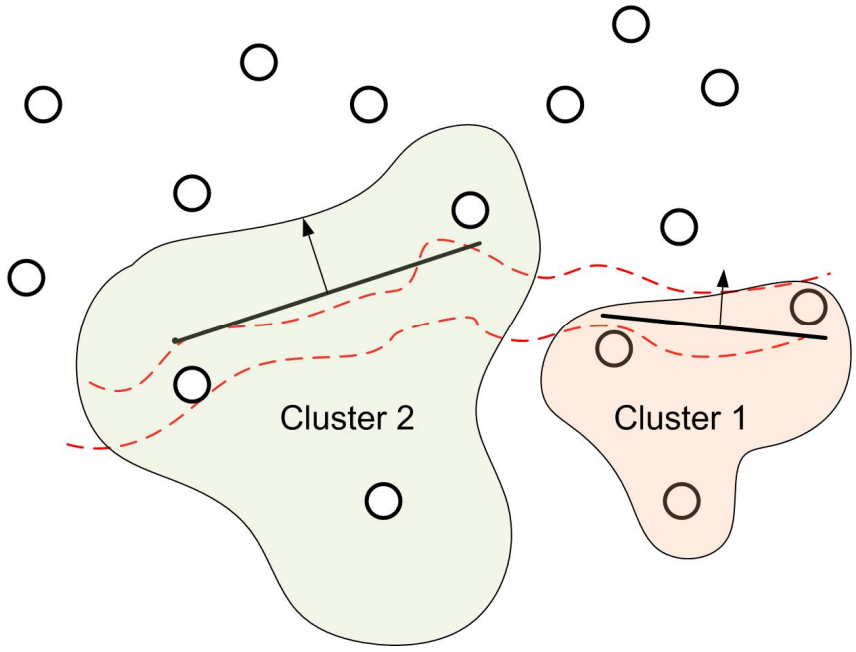


Σχήμα 1: Το προτεινόμενο στοχαστικό μοντέλο ανίχνευσης του μετώπου.

Βασιζόμενοι στο προτεινόμενο πιθανοτικό μοντέλο ανίχνευσης, διατυπώσαμε ένα Μπευσιανό πρόβλημα εκτίμησης των παραμέτρων εξέλιξης του φαινομένου το οποίο και λύσαμε αναλυτικά. Οι κλειστού τύπου αλγεβρικές εκφράσεις που προέκυψαν από τη λύση του προβλήματος, μπορούν με ευκολία να υλοποιηθούν από τους ενσωματωμένους μικροεπεξεργαστές των αισθητήριων κόμβων, καθώς σέβονται τις περιορισμένες επεξεργαστικές δυνατότητες των ασύρματων κόμβων.

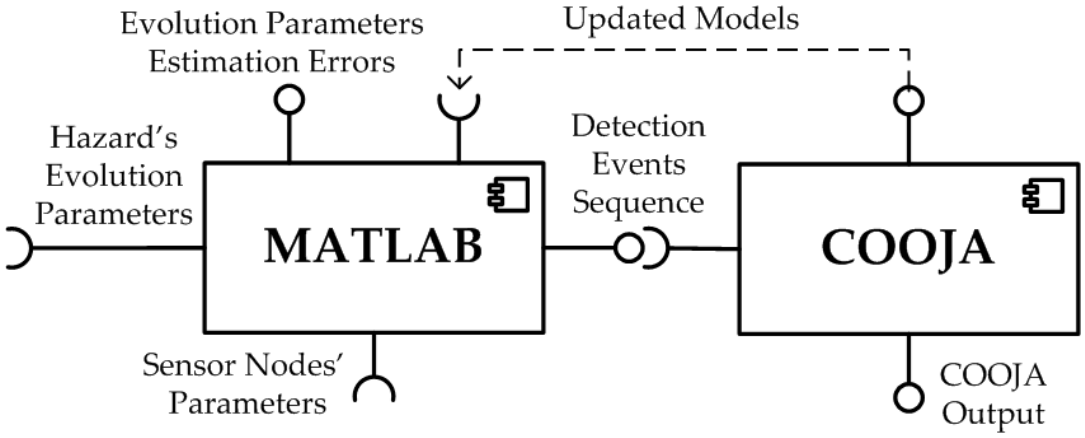
Συνεργατικός Αλγόριθμος Ασυρμάτων Δικτύων Αισθητήρων για την Εκτίμηση των Χαρακτηριστικών της Χώρο-Χρονικής Εξέλιξης Συνεχών Αντικειμένων.

Στη συνέχεια παρουσιάζουμε ασύγχρονο συνεργατικό αλγόριθμο ο οποίος, βασιζόμενος στη μοντελοποίηση του στοχαστικού μοντέλου ανίχνευσης και χρησιμοποιώντας δίκτυα αισθητήρων ρεαλιστικής πυκνότητας, μπορεί και εκτιμά με ακρίβεια τα χώρο-χρονικά χαρακτηριστικά εξέλιξης του μετώπου ενός συνεχούς αντικειμένου. Η εκτίμηση των χαρακτηριστικών εξέλιξης, υλοποιείται από μικρές συνεργαζόμενες ομάδες αισθητήριων κόμβων (clusters) οι οποίες σχηματίζονται δυναμικά (βλέπε Σχήμα 2). Κατά την εξέλιξη του συνεχούς αντικειμένου, ο αλγόριθμος ενημερώνει τις εκτιμώμενες παραμέτρους και με έναν πλήρως κατανεμημένο τρόπο τις προωθεί στους κόμβους οι οποίοι βρίσκονται στην κατεύθυνση εξέλιξης του μετώπου.



Σχήμα 2: Το μέτωπο του φαινομένου (κόκκινες διακεκομμένες καμπύλες) εισέρχεται στην περιοχή τοποθέτησης των κόμβων (μαύροι κύκλοι). Οι ασύρματοι κόμβοι του δικτύου αυτό-οργανώνονται δυναμικά σε μικρές ομάδες (τριάδες) οι οποίες εκτιμούν τις χώρο-χρονικές παραμέτρους εξέλιξης των τοπικών μετώπων (μαύρα ευθύγραμμα τμήματα).

Για την αποτίμηση του προτεινόμενου συνεργατικού αλγορίθμου υλοποιήσαμε ένα σύστημα προσομοίωσης το οποίο αποτελείται από δύο συστατικα: α) Έναν προσομοιωτή που αναπτύχθηκε σε γλώσσα προγραμματισμού Matlab και β) τον δημοφιλή προσομοιωτή ασυρμάτων δικτύων αισθητήρων COOJA (βλέπε Σχήμα 3). Ο προσομοιωτής που αναπτύχθηκε σε Matlab μας δίνει τη δυνατότητα να προσομοιώσουμε διαφορετικά σενάρια εξέλιξης καταστροφικών φαινομένων καθώς και διαφορετικά σενάρια πυκνότητας αλλά και στρατηγικές τοποθέτησης των ασύρματων κόμβων του δικτύου. Η χρήση του προσομοιωτή COOJA, μας επιτρέπει να προσομοιώσουμε ρεαλιστικά την συμπεριφορά του ασυρμάτου δικτύου αισθητήρων μιας και έχει τη δυνατότητα να υλοποιεί πρωτοκολλά επικοινωνίας ασυρμάτων δικτύων αισθητήρων (όπως το 802.15.4) καθώς και διαφορετικά σενάρια πιθανότητας αστοχίας των κόμβων κατά την επικοινωνία τους.

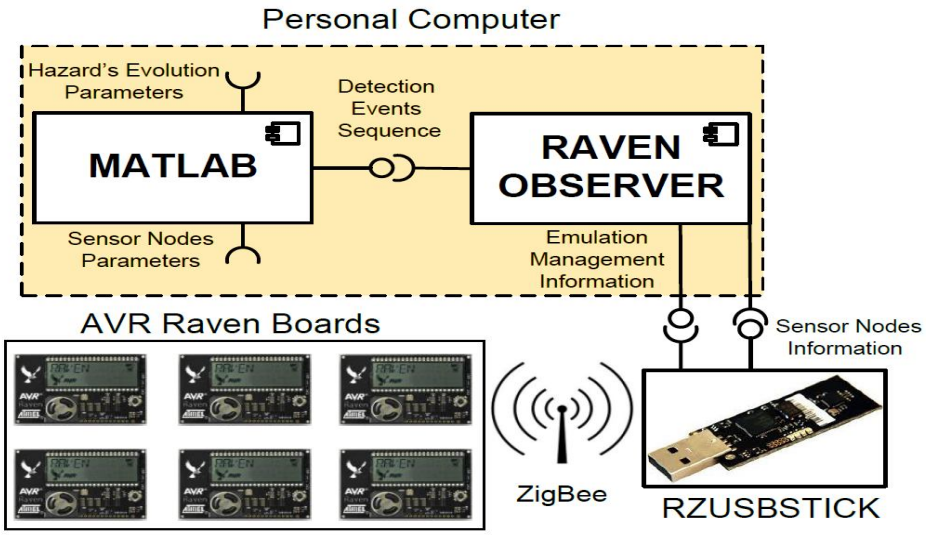


Σχήμα 3: UML διάγραμμα του συστήματος προσομοίωσης για την αποτίμηση της συμπεριφοράς του προτεινόμενου συνεργατικού αλγορίθμου.

Μέσω ενός μεγάλου αριθμού προσομοιώσεων παρουσιάζουμε την ικανότητα του αλγορίθμου να εκτιμά με ακρίβεια τα χαρακτηριστικά εξέλιξης πολύπλοκων συνεχών αντικειμένων κάτω από διαφορετικές συνθήκες σφαλμάτων των κόμβων και της μεταξύ τους επικοινωνίας τα οποία αναμένονται στα αφιλόξενα περιβάλλοντα που δημιουργούνται από εξελισσόμενα καταστροφικά φαινόμενα.

Σύστημα Εξομοίωσης για την Αποτίμηση της Συμπεριφοράς Συνεργατικών Αλγορίθμων Ασυρμάτων Δικτύων Αισθητήρων

Μια ακόμα σημαντική συνεισφορά της παρούσας διδακτορικής διατριβής είναι η ανάπτυξη ενός πρωτότυπου συστήματος εξομοίωσης το οποίο μας επιτρέπει να αποτιμήσουμε τις απαιτήσεις σε μνήμη, ενέργεια και επεξεργαστική ισχύ που έχουν κατανεμημένοι αλγόριθμοι όταν υλοποιούνται από ευρείας κλίμακας ασύρματα δίκτυα αισθητήρων στο πεδίο, χρησιμοποιώντας ένα μικρό αριθμό από πραγματικούς αισθητήρες. Η βασική ιδέα της προτεινόμενης μεθόδου εξομοίωσης, είναι η εικονική επανατοποθέτηση του μικρού αριθμού από διαθέσιμους πραγματικούς αισθητήριους κόμβους (π.χ. AVR Raven nodes), έτσι ώστε να εξομοιώνουν την συμπεριφορά των συνεργαζόμενων κόμβων μίας συστάδας (cluster) σε περιοχές εξέλιξης του φαινομένου.



Σχήμα 4: Το προτεινόμενο σύστημα εξομοίωσης για ρεαλιστική αποτίμηση της συμπεριφοράς συνεργατικών αλγορίθμων ασυρμάτων δικτύων αισθητήρων.

Το προτεινόμενο σύστημα εξομοίωσης αποτελείται από δύο συστατικά: α) έναν προσομοιωτή που υλοποιήσαμε σε γλώσσα προγραμματισμού Matlab και β) ένα Java πρόγραμμα (που ονομάσαμε Raven Observer) το οποίο διαχειρίζεται τον μηχανισμό της εικονικής επανατοποθέτησης των πραγματικών κόμβων (βλέπε Σχήμα 4). Με τη χρήση του Matlab προσομοιωτή, μας δίνεται η δυνατότητα προσομοιώνουμε την συμπεριφορά εξέλιξης διαφορετικών καταστροφικών φαινομένων καθώς και να δοκιμάζουμε

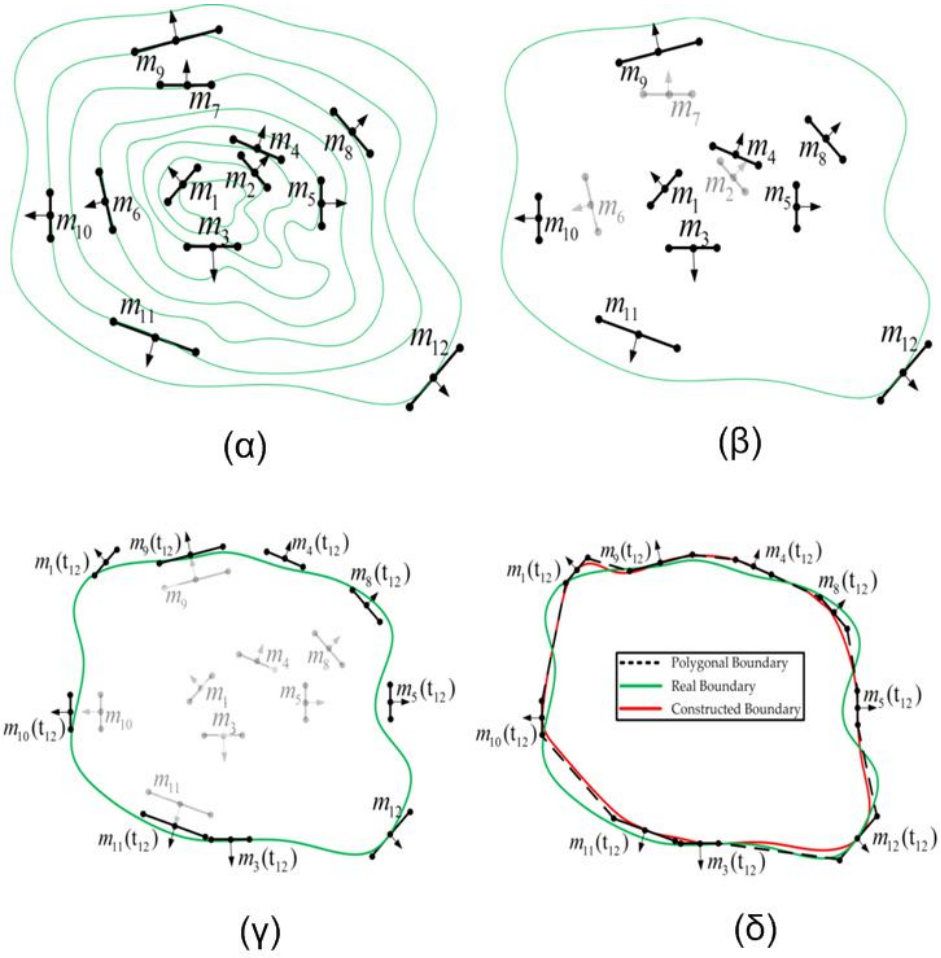
διαφορετικές πυκνότητες και στρατηγικές τοποθέτησης των κόμβων του δικτύου. Ο Matlab προσομοιωτής παράγει σαν έξοδο ένα αρχείο το οποίο περιέχει τις θέσεις των ασύρματων κόμβων του δικτύου καθώς και τους χρόνους που ανίχνευσαν το μέτωπο του καταστροφικού φαινομένου. Χρησιμοποιώντας αυτό το αρχείο ως είσοδο, ο Raven Observer καθορίζει την ακολουθία με την οποία πρέπει να γίνει η εικονική επανατοποθέτηση των πραγματικών κόμβων του δικτύου.

Παρουσιάζουμε την δυνατότητα της πραγματικής υλοποίησης του προτεινόμενου συνεργατικού αλγορίθμου χρησιμοποιώντας την πλατφόρμα AVR RAVEN της ATMEL. Μέσω ενός μεγάλου αριθμού αποτελεσμάτων εξομοίωσης του αλγορίθμου παρέχουμε σημαντικές ενδείξεις ότι ο προτεινόμενος συνεργατικός αλγόριθμος μπορεί να χρησιμοποιηθεί στη πράξη για την ανάπτυξη κατανεμημένου συστήματος εκτίμησης των παραμέτρων εξέλιξης συνεχούς αντικειμένου μιας και σέβεται πλήρως τους περιορισμούς μνήμης, επεξεργασίας και ενέργειας των ασύρματων κόμβων χαμηλού κόστους του εμπορίου που χρησιμοποιούνται στις εφαρμογές ασυρμάτων δικτύων αισθητήρων.

Αλγόριθμος Ανακατασκευής του Μετώπου Συνεχών Αντικειμένων

Χρησιμοποιώντας τις τοπικές εκτιμήσεις των χαρακτηριστικών εξέλιξης ενός συνεχούς αντικειμένου σε διαφορετικά σημεία του μετώπου (βλέπε Σχήμα 5α), εισάγαμε έναν αλγόριθμο ο οποίος συνδυάζει δυναμικά την πληροφορία από ένα μικρό αριθμό τοπικών εκτιμήσεων και ανακατασκευάζει τη συνολική καμπύλη του μετώπου οποιαδήποτε στιγμή της εξέλιξης του επιθυμούμε. Πιο συγκεκριμένα, ο προτεινόμενος αλγόριθμος χρησιμοποιεί τις τοποθεσίες και τις εκτιμώμενες παραμέτρους εξέλιξης τοπικών μετώπων και χρησιμοποιώντας κατάλληλες τεχνικές σύντηξης πληροφορίας, καθορίζει ένα υποσύνολο των διαθέσιμων τοπικών μετώπων (βλέπε Σχήμα 5β) τα οποία χρησιμοποιεί για να προσεγγίσει τμήματα του μετώπου του συνεχούς αντικειμένου μια συγκεκριμένη χρονική στιγμή (βλέπε Σχήμα 5γ). Στη συνέχεια χρησιμοποιώντας τεχνικές βασισμένες στην υπολογιστική γεωμετρία, ανακατασκευάζει ένα πολύγωνο το οποίο στη συνέχεια προσεγγίζεται από μια “ομαλή” κλειστή καμπύλη

(B-spline) η οποία αποτελεί το ανακατασκευασμένο μέτωπο του συνεχούς αντικειμένου (βλέπε Σχημα 5δ). Τέλος, βασιζόμενοι στην αβεβαιότητα των παραμέτρων εξέλιξης των τοπικών εκτιμήσεων, ο προτεινόμενος αλγόριθμος παράγει ένα χωρικό πεδίο πιθανότητας το οποίο μας δίνει για κάθε σημείο του πεδίου την πιθανότητα να έχει επηρεαστεί από το εξελισσόμενο συνεχές αντικείμενο.



Σχήμα 5: α) Οι διαθέσιμες τοπικές εκτιμήσεις (μαύρα ευθύγραμμα τμήματα) που περιγράφουν τα τοπικά χαρακτηριστικά εξέλιξης του μετώπου (πράσινες καμπύλες). β) Οι επιλεγμένες τοπικές εκτιμήσεις που θα χρησιμοποιηθούν στην ανακατασκευή του μετώπου την χρονική στιγμή που επιθυμούμε (t_{12}). γ) Η χώρο-χρονική εξέλιξη των επιλεγμένων τοπικών εκτιμήσεων τη χρονική στιγμή t_{12}. δ) Το πολύγωνο (μαύρες διακεκομμένες γραμμές) και η ανακατασκευασμένη "ομαλή" καμπύλη (κόκκινη καμπύλη) που προσεγγίζουν το μέτωπο του συνεχούς αντικειμένου.

Μέσω ενός μεγάλου αριθμού προσομοιώσεων, παρουσιάζουμε την ικανότητα του προτεινόμενου αλγορίθμου να καθορίζει με ακρίβεια τα όρια συνεχών αντικειμένων με διαφορετικά χαρακτηριστικά εξέλιξης (π.χ. σχήματα, επιτάχυνσης, επιβράδυνσης κτλ.), χρησιμοποιώντας τις παραμέτρους εξέλιξης ενός μικρού αριθμού τοπικών μετώπων στις οποίες μπορεί να υπεισέρχονται σημαντικά σφάλματα.

Συμπεράσματα

Στην παρούσα διδακτορική διατριβή αναπτύξαμε ένα σύστημα παρακολούθησης και πρόβλεψης της χώρο-χρονικής εξέλιξης συνεχών αντικειμένων το οποίο βασίστηκε πάνω στην τεχνολογία των ασύρματων δικτύων αισθητήρων. Το προτεινόμενο σύστημα ξεπερνά όλους τους περιορισμούς που εισάγονται από τις μέχρι τώρα προτεινόμενες τεχνικές. Συγκεκριμένα,

Αναπτύξαμε ένα ρεαλιστικό στοχαστικό μοντέλο ανίχνευσης μετώπου το οποίο μοντελοποιεί: α) την αβεβαιότητα του αισθητήρα ως προς την απόσταση ανίχνευσης του μετώπου καθώς και β) την πιθανότητα αστοχίας ανίχνευσης του μετώπου λόγω καταστροφής του αισθητήρα από το φαινόμενο. Το προτεινόμενο στοχαστικό μοντέλο ανίχνευσης, μας οδήγησε στην διατύπωση ενός Μπαυσιανού προβλήματος εκτίμησης των χώρο-χρονικών παραμέτρων εξέλιξης του μετώπου, από το οποίο προέκυψαν αναλυτικές λύσεις.

Αναπτύξαμε ένα ασύγχρονο συνεργατικό αλγόριθμο ασυρμάτων δικτύων αισθητήρων ο οποίο χρησιμοποιώντας δίκτυα ρεαλιστικής πυκνότητας εκτίμα με ακρίβεια τα χώρο-χρονικά χαρακτηριστικά εξέλιξης του μετώπου, κάτω από διαφορετικά σενάρια: α) εξέλιξης καταστροφικών φαινομένων, β) πυκνότητας δικτύου, γ) στρατηγικής τοποθέτησης κόμβων, γ) πιθανότητας αστοχίας των κόμβων κατά την επικοινωνία τους αλλά και λόγω καταστροφής τους.

Αναπτύξαμε ένα σύστημα προσομοίωσης το οποίο μας επιτρέπει χρησιμοποιώντας ένα μικρό αριθμό από πραγματικούς αισθητήρες να αποτιμήσουμε τη συμπεριφορά του προτεινόμενου συνεργατικού αλγορίθμου και τις απαιτήσεις του ως προς την

κατανάλωση ενέργειας, την επεξεργασία και την μνήμη. Τα αποτελέσματα προσομοίωσης έδειξαν ότι ο προτεινόμενος αλγόριθμος είναι κατάλληλος για την ανάπτυξη ρεαλιστικών εφαρμογών ασυρμάτων δικτύων ευρείας κλίμακας μιας και σέβεται τις περιορισμένες δυνατότητες (ενεργειακές, επεξεργασίας και μνήμης) των σημερινών ασυρμάτων δικτύων αισθητήρων.

Τέλος, αναπτύξαμε έναν αλγόριθμο ο οποίος χρησιμοποιώντας ένα μικρό αριθμό τοπικών εκτιμήσεων της χώρο-χρονικής εξέλιξης ενός καταστροφικού φαινομένου καταφέρνει να ανακατασκευάσει με ακρίβεια τα όρια του εξελισσόμενου μετώπου οποιαδήποτε στιγμή επιθυμούμε. Μέσω ενός μεγάλου αριθμού προσομοιώσεων αποδείξαμε ότι ο προτεινόμενος αλγόριθμος είναι ικανός να ανακατασκευάζει με ακρίβεια τα όρια διαφορετικών τύπων συνεχών αντικειμένων ακόμα και όταν στις παραμέτρους εξέλιξης των τοπικών μετώπων υπεισέρχονται σημαντικά σφάλματα.

Contents

1 Introduction **47**

1.1 Motivation 47

1.2 Contributions 49

2 Background and Related Work **53**

2.1 Continuous object tracking techniques based on WSNs - Literature Review 53

2.2 Comparison and Conclusions 61

2.3 Current Limitations - Objectives of our method 62

3 Local Front Probabilistic Modeling Approach and Parameters Estimation **65**

3.1 Preliminaries 65

3.1.1 Sensor Network Assumptions 66

3.1.2 Sensor Node Status 67

3.1.3 Local Front Models Parameters 68

3.1.4 Sensor Node Information and Tables 69

3.2 Modeling Detection Distance Uncertainty 70

3.3 Modeling Speed Uncertainty 73

3.4 Model Parameters Updating 74

3.4.1 Speed 74

3.4.2 Orientation 77

3.4.3 Evolution Direction 77

4 Collaborative Algorithm for Estimating the Spatiotemporal Evolution of Continuous Objects **79**

4.1 Preliminaries . . . 79
4.1.1 Sensor Messages . . . 79
4.1.2 Sensor Network Assumptions . . . 80
4.2 Collaborative In Network Algorithm . . . 81
4.2.1 Forming a Local Cluster . . . 81
4.2.2 Model Updating . . . 87
4.2.3 Model Propagation . . . 88
4.3 Evaluation Setup . . . 91
4.3.1 WSN Simulation Workflow . . . 91
4.3.2 Experimental Setup . . . 92
4.4 Results and Discussion . . . 93
4.4.1 Experiment 1: Multi-source diffusive hazards . . . 94
4.4.2 Experiment 2: Complex diffusive hazards with irregular shapes . . . 105
4.4.3 Comparison to PRECO sheme . . . 118

5 Assesing Requirements for Large Scale implementation using Simulation-Driven WSN Emulation 123
5.1 Motivation . . . 123
5.2 WSN Implementation . . . 124
5.2.1 WSN Platform Specifications . . . 124
5.2.2 Simulation-Driven Emulation Workflow . . . 126
5.3 Evaluation Results . . . 128

6 Continuous Object Boundary Reconstruction Algorithm 133
6.1 Preliminaries . . . 133
6.1.1 Local Front Parameters . . . 135
6.2 Finding Parts of the Boundary . . . 138
6.2.1 Local Front Information Processing . . . 139
6.3 Boundary Reconstruction . . . 145
6.3.1 First Phase: Polygonal Approximation of the Boundary . . . 145
6.3.2 Second Phase: Smooth Boundary Reconstruction using Uniform Cubic B-Splines . . . 150

6.4 Evaluation Setup 153
6.4.1 Simulation Workflow 154
6.4.2 Evaluation Metrics 154
6.4.3 Experimental Setup 156
6.5 Results and Discussion 156
6.5.1 Experiment 1: diffusive hazard with regular shape 156
6.5.2 Experiment 2: diffusive hazard with irregular shape 162

7 Conclusions and Further Research Directions 167

A Local Front Orientation Updating Procedure 171

B Performance Evaluation 175
B.1 Experiment 1 175
B.2 Experiment 2 177

C Experiment 1: Setup and Animation 181

D Calculating the Coordinates of Local Front End Points 185

E Space-Time Evolution of a Local Front Estimate 187

F Information Fusion of Local Fronts 189

G Initialization of Local Front Evolution Parameters 193
G.1 Experiment 1: Circular Front 193
G.2 Experiment 2: FLogA Simulation 194

List of Tables

2.1 Comparison of Continuous Object Tracking Schemes based on WSN technology. . 61

4.1 Experiment 1 results summary: The Median/Inter Quartile Range of the orientation error (in degrees) and percent speed error under different nodes density, probability of node failure, and sensing radii conditions. For each entry the reported statistics were computed based on 200 simulation runs (50 WSN random deployments x the 4 Rx/Tx failure probability cases considered). 95

4.2 The average percent change (increase(+), decrease(-))/stdvs of the a) total number of model updates, b) mean number of Rx and Tx messages exchanged per update, and c) mean Rx and Tx energy consumed per update, for the 75 and 125 nodes (per km^2) density scenarios relatively to the 100 nodes density scenario. 99

4.3 Summary of Estimation Results under Sensing Model Match and Mismatch Conditions: The Median/Inter Quartile Range of the orientation (in degrees) and percent speed errors of the proposed algorithm for different nodes density, probability of node failure and sensing models assumptions. For each table entry the statistics were computed based on 200 simulation runs (50 WSN random deployments x the 4 Rx/Tx failure probability cases considered). The sensing model match case and two sensing model mismatch cases have been considered. 104

4.4 Experiment 2 results summary: The Median and Inter Quartile Range of the orientation (in degrees) and speed errors under different sensor density, probability of node failure and sensing range radii assumptions. For each entry the statistics were computed based on 200 simulation runs (10 WSN random sensor node deployments for each of the 5 wildfire evolution scenarios x the 4 Rx/Tx failure probability cases considered). 113
4.5 Summary of Experiment 1 Results: The Median/Inter Quartile Range of the orientation (in degrees) and percent speed errors of the proposed and the PRECO method ([37,38]) under different density, probability of node failure conditions and sensing radii assumptions. For each condition the statistics were computed based on 50 simulation runs (50 WSN random deployments). 119
4.6 Summary of Experiment 2 Results: The Median and Inter Quartile Range of the orientation (in degrees) and speed errors of the proposed and the PRECO method ([37,38]) under different sensor density, probability of node failure conditions and sensing radii assumptions. For each condition the statistics are computed based on 50 simulation runs (10 random sensor node deployments for each one of the 5 wildfire evolution scenarios). . . . 119
4.7 Summary of Experiment 2 Results using space varying wind parameters: The Median and Inter Quartile Range of the orientation (in degrees) and speed errors of the proposed and the PRECO method ([37, 38]) under different sensor density, probability of node failure conditions and sensing radii assumptions. For each condition the statistics are computed based on 50 simulation runs (10 random sensor node deployments for each one of the 5 wildfire evolution scenarios). 120

List of Figures

3.1 The neighborhood (grey) and local front model characteristics of sensor S_i^M. 67

3.2 Sensing Modeling: (a) Probabilistic exponential sensing model. (b) The proposed shifted Gaussian sensing model. 71

3.3 Updating the local front model. 73

3.4 Speed model updating procedure. (a) The prior model (U_i), and the two ``observation'' models ($U_{ij}, U_{ik}$). (b) The mixture model $p$ resulting by combining speed ``observation'' models; the normal distribution $\hat{q}$ that best approximates p by minimizing the Kullback Lebier divergence $(KL(p||\hat{q}))$; the resulting posterior speed model U_i^*. 76

4.1 Local front model updating procedure: (a) Node S_i becomes Master candidate and checks if it satisfies the conditions to become a Master, (b) node S_i becomes a Master and ``enslaves'' its neighbors S_j, S_k and S_l, (c) Master S_i^M uses the information received from its two Helpers (S_j^H and S_k^H) and updates the local front's line parameters, (d) node S_k becomes the new Master and S_i releases its slaves. 82

4.2 *Detection Procedure* (UML sequence diagram). 83

4.3 *Master Check Necessary Conditions Procedure* (UML sequence diagram). 84

4.4 *Master's Declaration* (UML sequence diagram). 85

4.5 *Updating Model Parameters* (UML sequence diagram). 87

4.6 *Model Propagation Procedure* (UML sequence diagram). 90

4.7 UML component diagram of Matlab-COOJA based simulation workflow. . . 92

4.8 Experiment 1: Total number of model updates, mean number of messages and Rx/Tx energy consumed per model update, for each node failure and Rx/Tx failure probability considered (density = 100 sensor nodes per km^2). 96

4.9 Experiment 1: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 75 sensor nodes per km^2). 100

4.10 Experiment 1: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 125 sensor nodes per km^2). 101

4.11 Scenario 1: The red curve corresponds to the shifted Gaussian sensing model assumed by the sensor nodes. The black (blue) curve corresponds to the Gaussian (exponential) monotonic sensing model used to draw the detection distance under the model mismatch conditions considered (see text for details). 103

4.12 Wildfire Scenario 1: Snapshot (a) shows the wildfire's ignition point, located at the bottom right corner and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. The yellow arrow indicates prevailing wind direction. Snapshots (b), (c) and (d) show the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively. 108

4.13 Wildfire Scenario 2: Snapshot (a) shows the wildfire's ignition point, located at the bottom right corner outside the $1km^2$ forest area and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. The yellow arrow indicates prevailing wind direction. Snapshots (b), (c) and (d) show the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively. 109

4.14 Wildfire Scenario 3: Snapshot (a) show the wildfire's ignition points, located near to the top left corner and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. The yellow arrow indicates prevailing wind direction. Snapshots (b), (c) and (d) show the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively. 110

4.15 Wildfire Scenario 4: Snapshot (a) shows the wildfire's ignition points, located outside the left side of the square forest area respectively and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. The yellow arrow indicates prevailing wind direction. Snapshots (b), (c) and (d) show the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively. 111

4.16 Wildfire Scenario 5: Snapshot (a) shows the wildfire's multiple ignition points (burn stripe), on the top left corner of the square forest area and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. Snapshots (b), (c) and (d) shows the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively. 112

4.17 Experiment 2: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 75 sensor nodes per km^2). 114

4.18 Experiment 2: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 100 sensor nodes per km^2). 115

4.19 Experiment 2: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 125 sensor nodes per km^2). 116

5.1 AVR Raven evaluation kit: (a) AVRRAVEN board (sensor node), (b) RZUSBSTICK boards (sink node) (adopted from [94]). 125

5.2 AVR Dragon programmer (adopted from [95]). 125
5.3 Simulation driven emulation workflow. 127
5.4 Mean and stdev of received/transmitted Messages/Bytes, at the Cluster and Network level for different WSN densities. 129
5.5 Mean and stdev of energy consumed for Rx and for Tx operations (as a function of the node□s Tx power). Network and Cluster-level analysis for the 50 nodes scenario. 130
5.6 Mean and stdev of energy consumed for Rx and for Tx operations (as a function of the node□s Tx power). Network and Cluster-level analysis for the 75 nodes scenario. 131
5.7 Mean and stdev of energy consumed for Rx and for Tx operations (as a function of the node□s Tx power). Network and Cluster-level analysis for the 100 nodes scenario. 131

6.1 a) The green curves show different instances of an evolving continuous object's boundary; each boundary corresponds to the time instance where a local front estimate (black segments) takes place. The black segments correspond to the selected local front estimates that will be used to determine the continuous object's boundary at time t_{12}. c) The new locations of the selected local front estimates after their space-time evolution at time t_{12}. d) The polygon (black dashed polygon) and the smooth curve (red curve) that approximate the continuous object's boundary. 134
6.2 Orientation angle uncertainty of a local front model. 135
6.3 The local front estimate m_i and its evolution parameters. 136
6.4 The space-time evolution of the local front estimate m_i at time t_j. 137
6.5 Model selection procedure: Local front m_i and m_j (where $t_i \leq t_j$) is assumed to describe the evolution behavior of the same part of an evolving boundary. Only the more recently estimated local front (m_j) is kept in $\mathcal{M}_n$ (see text for details). 139
6.6 The four events that require special handling during the space-time evolution of the local fronts estimates (see text for details). 141

6.7 The local fronts' lengths adjustment procedure. a) The overlap of the local front segments, results a curly B-spline curve during the boundary reconstruction procedure, b) After applying the size adjustment procedure we avoid the curly curve formation. 144

6.8 First step: Finding the connection sequence of the local front segments. . . 146

6.9 a) The middle points of each local front is connected with its end points (formation of black segments); the red dashed lines show the first steps of local front segments connection. b) First polygonal boundary approximation (note that the direction vectors of m_1 and m_{10} points inside the polygon) c) The polygonal approximation that satisfies the boundary's polygon conditions (direction vectors of m_1 and m_{10} points outside the polygon after interchanging the connections of their end points.) d) The smooth boundary construction using a B-spline curve (red curve). 151

6.10 The blue (red) curve correspond to the reconstructed (estimated) boundary of the diffusive 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 metric. 155

6.11 Experiment 1: a) Boxplots summarizing the distribution of the boundary tracking accuracy under different local estimates density scenarios. b) The boundary tracking accuracy as a function of time for different densities of local estimates. c) The mean number of the available local estimates that participate at each time instance to the boundary construction algorithm, for different local estimates density scenarios. d) The boundary construction probability as a function of time for different local estimates density scenarios.158

6.12 Experiment 1: Boundary tracking accuracy under different local estimates densities and (a) orientation errors, (b) speed errors. For all the orientation (speed) error scenarios the speed (orientation) error was set to 0 in order to focus on how the orientation (speed) errors affect the boundary tracking accuracy. 160

6.13 Experiment 1: Boxplots summarizing for the uncertainty pairs of values a) $\{10^0, 10\%\}$ b) $\{20^o, 10\%\}$, c) $\{30^o, 30\%\}$ the distribution of the boundary tracking accuracy under different local estimates density scenarios. d) The mean boundary tracking accuracy under different local estimates density scenarios and uncertainty pairs of values. 161

6.14 Experiment 2: a) Boxplots summarizing the distribution of the boundary tracking accuracy under different local estimates density scenarios. b) The boundary tracking accuracy as a function of time for different local estimates densities. c) The mean number of the available local estimates that participate at each time instance to the boundary construction algorithm, for different local estimates density scenarios. d) The boundary reconstruction probability as a function of time for different local estimates density scenarios. 164

6.15 Experiment 2: Boundary tracking accuracy under different local estimates densities and (a) orientation errors, (b) speed errors. For all the scenarios orientation (speed) errors scenarios the speed (orientation) error was set to 0 in order to focus on how the orientation (speed) errors affect the boundary tracking accuracy. 165

6.16 Experiment 2: Boxplots summarizing for the uncertainty pairs of values a) $\{10^0, 10\%\}$ b) $\{20^o, 10\%\}$, c) $\{30^o, 30\%\}$ the distribution of the boundary tracking accuracy under different local estimates density scenarios. d) The mean boundary tracking accuracy under different local estimates density scenarios and uncertainty pairs of values. 166

A.1 Updating the local front orientation parameter. 172

B.1 The orientation error evaluation method followed when the Helper nodes of a triplet have been affected from (a) the same circle, (b) different circles. . . 176

B.2 The $1km^2$ forest area is seen by FLogA as a grid of square cells. At the end of a wildfire simulation FLogA predicts, for each one of the square cells, information such as the time of fire's arrival, frontline's evolution direction and speed. At the bottom right corner we visualize the direction error evaluation procedure described in the text (see Section S2.2). 178

C.1 Figures C.1a -C.1f, show 6 snapshots of the presented simulation scenario of Experiment 1. In Figure C.1f the red arrows show the Master status inheritance trajectories that occur during the hazard's front line propagation. 184

D.1 The local front estimate m_i; the coordinates of its end points (red dots) are determined by the points of intersection between line ε_i and circle C_i. . . . 186

E.1 Space-time evolution of the local front m_i at time t_j. 188

F.1 Models Information Fusion. 190

F.2 Estimating the speed/orientation (U_f/Φ_f) distributions of model m_k. (a) The speed/orientation distributions of models m_i and m_j. (b) The mixture models P_U/P_Φ resulting by combining speed/orientation normal distributions of $\{m_i, m_j\}$ and the normal distribution U_f/Φ_f that best approximates P_U/P_Φ by minimizing the Kulback-Leibler divergence. 191

G.1 Initalization of the local fronts parameters. 193

Preface

This research has been co-financed by the European Union (European Social Fund-ESF) and Greek national funds through the Operational Program “Education and Lifelong Learning” of the National Strategic Reference Framework (NSRF)- Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

Chapter 1

Introduction

This dissertation addresses the problem of tracking and predicting the boundaries of diffusive hazardous phenomena, often modeled as ``continuous objects'' (i.e. objects that tend to occupy large areas and their shape and size continuously changing with time), using Wireless Sensor Networks (WSNs). In this chapter, we present the motivation, the objectives and the contributions of this research.

1.1 Motivation

Wireless Sensor Networks (WSNs) is a rapidly maturing technology with a wide range of applications (e.g. target tracking, surveillance, environmental monitoring, patient monitoring to name a few). A WSN typically consists of a large number inexpensive autonomous electronic devices (sensor nodes) which are deployed over a geographical region and monitor physical or environmental parameters. Apart from ``sensing'' the environment, WSN nodes are also able to process data and exchange information. Recent advances in microelectronics and wireless communication have made WSN technology an ideal candidate for large-scale decision and information-processing tasks.

Tracking objects (i.e. determining their location over time) has been a well studied problem with numerous civilian and military applications. Apart from finding the trajectory of the objects, it is also important to estimate their motion characteristics (i.e. direction and speed) in real time, since this information can be used to predict their future locations

and understand their evolution behavior.

Wireless sensor network technology has been extensively used for single and multiple target tracking applications [1--12]. Due to the rapidly dropping cost of the sensor nodes, WSNs are also gaining popularity in environmental monitoring applications [13--21]. Recently, sensor network-based methods have been proposed for detecting the boundaries of diffusive hazardous phenomena [22--44], modeled as ``continuous objects'' (such as expanding wildfires, oil spills, diffusing bio-chemical materials etc.). The ability to track and predict, with reasonable accuracy, the location of a diffusive hazard's boundary is of paramount importance since it helps the authorities to organize efficiently their responses (hazard suppression, possible evacuation etc). However, traditional target tracking algorithms cannot be applied for continuous object tracking, since the two problems are fundamentally different. Continuous objects are usually spread in large regions and their size and shape is dynamically changing with time. In contrast, individual targets (such vehicles, animals, humans etc) have very small size compared to the WSN's deployment area and therefore a much smaller number of sensors usually suffices to track their trails.

The key idea behind the reported WSN-based continuous object tracking methods has been an attempt to identify over time the sensor nodes located closest to the evolving object's front line (boundary nodes). Although these methods can estimate implicitly the boundaries of a continuous object (evolving hazard) using the locations of the boundary nodes, they have important limitations that renders them impractical for the development of real world application systems for hazard tracking.

The main limitations that appear in almost all reported WSN-based continuous object tracking schemes are (see also Chapter 2):

L1: They require unrealistic sensor nodes densities (thousands sensors per km^2) to determine with reasonable accuracy the boundary of an evolving continuous object. Although, the cost of the sensor nodes has been significantly reduced, it still remains prohibitive to cover large geographical regions (many km^2) with high density WSNs.

L2: They do not consider node or communication failures. However, these failures are

certainly expected in large scale WSNs applications, and especially in the harsh environments created by the evolving hazardous phenomena (e.g. wildfires).

L3: They require synchronization between the sensor nodes, a capability that is difficult to achieve even in small scale WSNs.

L4: They assume an idealized sensing mechanism (i.e. fixed sensor nodes detection distance, do not consider sensing functionality disruptions etc) that renders them impractical for hazard tracking.

L5: They are incapable to provide information about the spatiotemporal evolution characteristics (e.g. direction and speed) of the continuous object's boundary. This limitation makes them incapable to be used for predictive modeling as part of decision support systems.

L6: They are incapable to assess the processing, memory and energy requirements before a real field deployment.

L7: They propose naive techniques to reconstruct the continuous object's boundary or are incapable to reconstruct it without using the human ability to identify the boundary's shape from the boundary nodes locations.

1.2 Contributions

The main contribution of this dissertation is the conception, design and development of a WSN-based continuous object tracking scheme that addresses all the aforementioned limitations. In particular,

Chapter 2 presents the related work in continuous object tracking; points to severe limitations that render them impractical for real applications and motivates the original work that has been performed in the context of this doctoral dissertation.

Chapter 3 presents a novel probabilistic sensing modeling approach that is able to capture sensor nodes' detection distance uncertainty and possible functionality disruptions as the front of the diffusive hazard approaches closer than a certain distance (addresses L4).

Based on the probabilistic sensing modeling we formulate a Bayesian parameter estimation problem which could be solved analytically. The derived closed formed algebraic expressions of the solution can be easily implemented by the embedded microprocessors of the WSN nodes in order to estimate the local front evolution characteristics (speed, orientation and evolution direction) of a continuous object, since they respect the nodes' processing capabilities and strict energy constraints.

Chapter 4 presents the proposed *asynchronous* collaborative algorithm that based on the modeling of Chapter 3 and using WSNs of *realistic density* can estimate with accuracy the spatiotemporal evolution parameters (orientation, direction and speed) of a continuous object's local boundary (addresses L1, L3, L5). The parameters estimation procedure is implemented in a collaborative fashion by dynamically formed clusters (triplets) of sensor nodes. The algorithm updates the local front model parameters and propagates them to sensor nodes situated in the direction of the hazard's propagation in a fully decentralized fashion. Extensive computer simulations are also presented in this chapter and demonstrate the ability of the algorithm to estimate accurately the evolution characteristics of complex continuous objects under different conditions and sensor node and communication link failures which are expected in harsh environments (addresses L2).

Chapter 5 presents a novel simulation-driven emulation scheme which allows to realistically asses the memory, processing and energy requirements of a cluster based collaborative algorithm before attempting to deploy a large scale WSN in the field (addresses L6). The key idea of the proposed method is to re-allocate (virtual repositioning) a small number of available real sensor nodes so that they implement the cluster nodes located closer to the hazard's boundary. We demonstrate the capabilities of the proposed emulation scheme to implement the proposed collaborative algorithm presented in Chapter 4 using a small number of ATMEL's Raven nodes. Finally we present extensive WSN emulation results that provides convincing evidence that the collaborative algorithm of Chapter 4 is suitable for large-scale WSN deployment, since it respect the memory, processing and energy constraints of commodity sensor nodes used in WSN implementations.

Chapter 6 presents a novel algorithm which combines dynamically the information of a small number of local estimations about the boundary's evolution characteristics, as they become available to hypothesized fusion center, to reconstruct the continuous object's

boundary (addresses L7). More specifically the proposed algorithm uses the locations and evolution parameters of the computed local front estimates and using information fusion techniques it determines a set of local fronts segments that approximate different parts of the continuous object's boundary at a specific time instance. Next, using concepts grounded on computational geometry, it reconstructs a smooth closed curve that approximates the object's boundary. Finally, based on the uncertainty characterization of the local front models evolution parameters (Chapter 3), the proposed algorithm generates a probability field that indicates for each point of the considered area the probability to be reached by the continuous object. Extensive simulations demonstrate that the proposed boundary reconstruction algorithm can be used to determine with accuracy the boundaries of different types of continuous objects (e.g. time-varying evolution rates and/or irregular boundary shapes), using only a small number of local front estimates even in cases where the evolution parameters have been distorted with error.

Finally, Chapter 7 summarises our findings and point to interesting future research directions worth pursuing in the field of continuous object tracking under uncertainties that is fast moving towards an integration with simulation based hazard predictive models.

Most parts of the doctoral dissertation have been published in peer reviewed scientific journals and high quality referred Conferences Proceedings at the time of its writing.

Chapter 2

Background and Related Work

During the last few years significant efforts have been invested worldwide in the design and development of distributed systems for large scale environmental monitoring. The goal of such efforts is to track, predict and manage the consequences of diffusive hazardous phenomena, such as wildfires, oil slicks, chemical leaks etc. Wireless Sensor Networks, is a maturing technology which is increasingly expected to play a major role in the development of such systems. In this chapter we present the state of the art of the developed continuous object tracking schemes based on the WSNs, and we appoint their current limitations that make them impractical for hazard tracking applications.

2.1 Continuous object tracking techniques based on WSNs - Literature Review

Predicting with reasonable accuracy the spatiotemporal evolution of a diffusive hazardous phenomenon (such as a wild fires, oil slick, etc.) is of paramount importance since it helps the authorities to organize efficiently their response (hazard suppression, possible evacuations etc.). Research efforts around the globe are focusing in developing hazard specific predictive models [46--49, 51--62, 89]. However, most of these mechanistic models depend on a large number of space and time varying parameters which are difficult or even impossible to estimate in real time, making predictions deviate significantly from reality. To address this limitation many researchers have proposed system architectures which

attempt to integrate recent sensor measurements and simulation based predictive modeling into closed loop systems. The majority of these works reported so far in literature rely on remote sensing, where measured data (e.g. satellite spectral images) are used to re-calibrate simulation models in real-time in order to minimize model prediction errors. These methods, also known as Dynamic Data Driven Assimilation methods, have recently drawn the attention of the scientific community due to their expected high societal impact [63--67]. Unfortunately, in many cases, satellite images, or image data in general, are not available, or are inappropriate for detecting a certain diffusing hazard. Moreover, in most cases it's almost impossible to exploit directly (in realtime) the sensing data to calibrate the simulation models.

Wireless Sensor Networks (WSNs), is a mature state of the art technology and and has been extensively used for tracking the trajectories of single or multiple target [1--12]. The rapidly dropping cost of WSN technology, makes them a viable alternative for environmental monitoring applications [13--21]. Recently, a number of collaborative WSN-based methods have been proposed for detecting the boundaries of a diffusive hazards [22--40] often modeled as ``continuous objects''. Below we present in chronological order the most important works reported so far in continuous object tracking literature.

In their seminal paper [22], Chintalapudi and Govindan propose three methods (statistical, image processing and classifier) for detecting edges in sensor fields. The key idea of these methods is to determine the sensor nodes that are located in the interior of the event and within a pre-specified distance from the object's boundary. To achieve this, each sensor node collects information from its neighborhood (sensor nodes located within its communication area) and based on a parameter named ``tolerance radius'' decides whether it should be considered as an edge node. Using the locations of these sensors (edge sensors) they implicitly determine the boundary of the continuous object. In the statistical approach, a sensor node collects information from its neighbors and decides (using a boolean decision function) if it is an edge sensor node or not. The image processing approach is based on modified image processing edge detection technique where each sensor node decides if it is an edge sensor based on the detection values of the sensor nodes located within its probing area. Finally, the classifier based approach uses a linear classifier to determine a line (edge) that partitions the probing area of a sensor node in two

areas (interior and exterior). If the line passes through the sensor's tolerance circular area, it is deemed as edge sensor. Simulation results show that the classifier method is the most promising since it performs much better than the other schemes. Moreover in contrast to the statistical method its accuracy is independent from the selection of threshold values which are very difficult to set them correctly.

In [23] the authors Mitra and Nowak propose a novel technique based on hierarchical WSN structure that determines a staircase-like approximation of the continuous object's boundary. Initially the WSN deployment area is virtually partitioned in square cells. In each cell the algorithm determines a Cluster Head node which is responsible to collect the measurements from its neighbors (sensors inside the corresponding cell). Processing the collected data, Cluster head determines whether its cell lies on the continuous object's boundary or not. If the cell lies on the boundary, the algorithm produces a more fine-grained partition of the corresponding cell and for each one of them it repeats the aforementioned procedure. The cell partition procedure stops when the cells becomes small enough such that do not contain sensor nodes. The authors present simulation results that demonstrate the accuracy, the communication cost and the final cell partition size as a function of the sensor nodes density.

Based on the observation that the statistical method (proposed in [22]) is robust to sensor node failures, Liao et al. in [24, 25] propose a novel statistical edge detection technique based on hypothesis testing. Their method uses a set of local and global rules that help the nodes to decide wether they lie on the continuous object's boundary [24, 25]. Although the proposed method had low complexity and it was robust to node failures, it did not solve the main problem of the statistical method which was the appropriate threshold value selection. Thus, two years later, the authors proposed a new version of the statistical method [26] which solved the threshold selection problem based on the Neyman-Pearson optimality criterion. Simulation results indicate that the proposed statistical method outperforms the classifier based method (presented in [22]) in terms of boundary estimation accuracy while being robust to noisy environments and possible node failures.

In [27], Liu et al. present a novel approach based on a dual-space transformation, to determine the frontier of continuous object using binary sensor measurements. The dual-space transformation maps the location of the sensor nodes to a set of lines that help us to

determine the continuous object's frontier. Based on the estimated front line the authors claim that its information can be used to reduce WSNs energy consumption by switching, the nodes located away from it, to power saving modes. Using the results of a laboratory experiment the authors prove that the proposed method is able to determine the front line of a slowly moving continuous object.

In [28], Ji et al. propose a WSN-base methods which is able to detect and track the boundary of an evolving continuous object. The proposed method, named ``Dynamic Clustering Scheme (DCS)'', similar to the aforementioned techniques, it implicitly determines the boundary of the evolving object using the boundary nodes' locations. A sensor node is assumed as boundary node when it has detected the continuous object and has at least one node in its neighborhood (nodes inside its communication range) which has not detected the object yet. In sequence, these boundary nodes are dynamically organized into local clusters. Each cluster, determines a special node (Cluster Head node - CH) which is responsible to collect and fuse the local boundary information and transmit it to the sink. This dynamic clustering technique significantly reduces the communications cost compared to the naive technique where each sensor node sends its information to the sink. When the sink receives the local boundary information from the CHs, it is able to estimate the global boundary of the continuous object. Extensive simulation results demonstrate how the number of messages used for collecting the boundary's information scales with the number of clusters and how the precision of the boundary's detection scales with network's density.

In [29] the authors propose a WSN-based method named ``Continuous Object Detection and tracking Algorithm (CODA), which allows us using a hybrid static/dynamic clustering scheme to detect and track the boundary of an evolving continuous object. The algorithm uses a number of static clusters that is formed during the network's deployment. Each cluster has a special node named Cluster Head (CH). When a sensor detects the presence of a continuous object it transmits a detection message to its CH. Next, the CH uses a ``local boundary estimation function'' and determines which nodes within its cluster can be assumed as boundary nodes (nodes that lie on the continuous object's boundary). In sequence, the CH organizes its boundary nodes into a dynamic cluster, and sends the boundary information of this cluster to the sink. When the sink receives the boundary

information from a sufficient number of CHs, it forms the ``global boundary'' of the continuous object. Extensive simulation results show that CODA outperforms the DCS scheme since it reduces network's communication cost and achieves better boundary estimation precision.

In [30] the authors present a WSN-based algorithm named ``COntinuous BOundary Monitoring (COBOM) for detecting the boundary of a continuous object''. In this algorithm each node maintains a binary array (BN-array) which describes the detection status of each neighbors. When a sensor node changes its detection status it broadcast a message that informs its neighbors about this change. The neighbors when receive this message update the corresponding entry of their BN-arrays. If their BN-arrays contains ``0'' and ``1'' they become Boundary Nodes (BN). Based on the locations of the BNs the proposed scheme can implicitly determine the boundary of the continuous object. To further reduce the communication cost the authors introduce mechanism that selects only a subset of the BNs to report to the sink (Reporting Nodes - RN). Each RN collects the information of its neighbors compress it and send it to the sink. Simulation results show that this technique substantially reduces the number of message exchanged in the network. However, the authors do not provide numerical result to show how their algorithm performs in terms of boundary estimation accuracy.

Based on [30] the authors in [31] present a scheme named ``Energy-Efficient DEtection and MOnitoring for Continuous Objects in Wireless Sensor Networks (DEMOCO)''. In this scheme a sensor node becomes a Boundary Node (BN) when it receives a message, from at least one of its neighbors, that has different detection status. Based on the number of messages received with different detection status in a time period, a BN sets a random back-off time which is inversely proportional to the number of the received messages. The BN with the smaller back-off time becomes a Reporting Node (RN) and sends its information to the sink. Simulation results demonstrate that the DEMOCO outperforms in terms of energy efficiency the algorithm COBOM under different WSN density scenarios (which are extremely high) and object shapes (smooth and unsmooth). Although the authors claim that their scheme is able to track with accuracy the boundary of evolving continuous objects they do not provide sufficient results to support it.

In [32] the authors propose a novel algorithm named ``Two-tier Grid based Continuous

Object Detection and tracking (TG-COD) for tracking the boundary of an evolving continuous object''. The proposed algorithm is based on the formation of a grid structure which determined using the location information of a reference point, a grid cell size value and the sensor nodes' locations. When the sensor nodes deployed in the field, TG-COD creates a square grid of low density (large cell size) that covers the whole deployment area. When the boundary of a continuous object inserts in a grid cell, TG-COD produces within this cell a fine-grained grid which helps us to increase the tracking accuracy of the evolving boundary. TG-COD reduces the network's communication traffic, using the following data report method: ``In a fine-grained grid cell, a header node collects the data from the boundary nodes. In sequence, the header node reports this data to the header node of coarse-grained grid cell that it belongs. When the header node of the coarse-grained grid cell collects the reports of its fine-grained grid cell headers, it transmits them to the sinks''. The authors compare the proposed algorithm to CODA and DEMOCO and provide results that show that it outperforms with respect to energy communication cost and boundary detection precision.

In [33] the authors propose a scheme named ``Dynamic Rectangle Zoned-based Collaborative mechanism (DRZC) to detect and track the boundary of evolving continuous objects''. The key idea of this scheme is as follows: The sensor nodes that detect the continuous object collaborate and determine the sensor node that is located closer to the center of a rectangular area that contains them. Next, the selected node sends its location information to the sensor nodes located within the rectangle and they report him their data. It is worth to mention that the selection of the node located closer to the center of the rectanglular area, minimizes the network's communication cost during the data reporting phase. The size and the position of the rectangular area may dynamically change since they depend on continuous object's location and geometrical characteristics (size and shape). When the continuous object geometrical characteristics change, the rectangular area also changes and the selected node is altered by another node near the center of the ``new'' rectanglular area. Using the QualNet simulator the authors compare their algorithm with DCS with respect to the boundary reconstruction accuracy under different WSN densities and continuous object evolution scenarios. Simulation results shows that the boundary accuracy of the DRZC is similar with this of the DCS. However it is worth to

mention that the authors do not provide any information about the algorithm's communication cost which is expected by design to be high.

In [35] the authors propose a ``Scalable topology control based protocol for COntinuous Object detection and Tracking (SCOOT)'' which is able to reconstruct the boundary of an evolving continuous object using the information of a small subset of the deployed sensor nodes. This SCOOT protocol is separated in two phases: a) The collaborative data processing phase, and b) the object's location reporting phase. The collaborative data processing phase has two steps. At first step, the algorithm finds the nodes that have detected the phenomenon and have at least one neighbor which has not detected it yet. At second step the algorithm uses the information of these nodes and determines a subset of them (reporters) where their location information suffices to determine the continuous objects boundary without compromise the accuracy. Next, in the object's location reporting phase, the reporter nodes periodically transmit their location information to the sink node. The proposed protocol can track single source and multiple source continuous objects. Simulation results show that the proposed protocol significantly reduces the communication cost (requires a small number of reporters), the control message overhead and the data message overhead. Finally, the authors investigate how the effect of destroyed nodes would affect the protocol's performance.

In [36] the authors propose an interesting scheme that determines the area covered by the diffusive phenomenon. This scheme named, ``Ring-based Continuous Object Tracing (RCOT)'' uses the protocol described in [45] and organizes the nodes in connected rings. Using this ring structure the RCOT algorithm applies the following steps to determine the boundary of a continuous object. a) It determines the rings that lies on the continuous object's boundary. b) For each boundary ring, the algorithm determines the nodes that belong inside the object and those that belong outside the object and estimates the coordinates of a point which theoretically lies on the object's boundary. Finally, using the coordinates of these points the algorithm is able to determine the continuous objects boundary. Simulation results show that the RCOT performs significantly better to CODA with respect to communication energy consumption. However, the design of the RCOT algorithm indicate that its accuracy depends on the sensor nodes density and deployment strategy. Finally, although the authors claim that RCOT is able to trace the continuous

object boundary they do not provide numerical results to support it.

In [37, 38] the authors propose a novel WSN based predictive continuous object tracking scheme named ``PRECO (PREdictive Continuous Object tracking scheme)''. PRECO proposes a novel sensor nodes' ``wake-up'' mechanism to reduce the WSN's energy consumption. The proposed scheme predicts the future location of the continuous object's boundary line which provides knowledge for implementing a ``wake-up'' mechanism and decide which sleeping nodes need to be activated for future tracking. To predict the future location of the continuous object's boundary line the authors propose a simple method for the estimation of its spatiotemporal evolution parameters (speed and orientation). PRECO assumes as a boundary line, a line segment that connects two adjacent special Boundary Nodes (BN), named Master Boundary Nodes (MBN) (see Figure 5 in [37]). The sequential connection of all the adjacent boundary lines determines a polygon that approximates the continuous object. Considering the above we can easily conclude that the accuracy of PRECO's polygonal representation, depends on the number of the MBNs. However as the WSN's density decreases, the number of the MBNs is reduced and therefore the boundary's polygonal representation becomes coarser (imagine Figure 4 in [37] without the middle MBN). Simulation results show that using very high density WSNs, PRECO is able to determine with accuracy the boundary of an evolving continuous object (see Chapter 4 for details). Moreover, the proposed wake up mechanism significantly reduces the total WSN energy consumption.

All the works described consider the boundary detection problem in 2 dimensions (2D). In [40] the authors propose a novel flexible and energy efficient scheme that is able to track the boundary of a continuous object either it evolves in plain (2D) or space (3D). The proposed scheme implicitly determines the continuous object's boundary based on the location information of special selected nodes named Event Boundary Nodes (EBN). For the selection of EBNs the algorithm uses the sensor's measurements and fits a Gaussian mixture model where the number of its mixture components is determined using the Bayesian Information Critirion (BIC). Next, the mixture model is compared with a threshold value which help us to decide if the node is an EBN or not. Although this algorithm has larger computational cost than COBOM and DEMOCO, simulation results show that it outperforms in terms of energy efficiency and number of selected BNs.

Comparison Table of Continuous Object Tracking Schemes						
Name	High Density	Fault Tolerance	Prediction Ability	Boundary Accuracy	Bourndary Reconstruction	Reference
Chintalapudi	✓	✓	-	-	-	[22]
Nowak	✓	-	-	-	-	[23]
Liao	✓	✓	-	-	-	[24--26]
Liu	✓	-	-	-	-	[27]
DCS	✓	-	-	✓	-	[28]
CODA	✓	-	-	✓	✓	[29]
COBOM	✓	-	-	-	-	[30]
DEMOCO	✓	-	-	-	-	[31]
TG-COD	✓	-	-	-	-	[32]
DRZC	✓	-	-	-	-	[33]
SCOOT	✓	✓	-	-	-	[35]
RCOT	✓	-	-	-	-	[36]
PRECO	✓	-	✓	-	-	[37, 38]
Chen	✓	-	-	-	-	[40]

Table 2.1: Comparison of Continuous Object Tracking Schemes based on WSN technology.

2.2 Comparison and Conclusions

From the above analysis of the stat of the art we conclude that the key idea behind WSN-based continuous object tracking methods has been to identify over time the sensor nodes located closest to the evolving object's front line (boundary nodes). Although these methods can estimate implicitly the boundaries of an evolving hazard, they use assumptions which renders them impractical for real applications.

Table 2.1 summarizes the main characteristics of the aforementioned reported schemes. The first column contains the name of the algorithms or the names of the corresponding authors. In second column we mark the schemes that require unrealistic density networks (thousand of sensors per km^2). In third column we mark the schemes that consider possible sensor node failures during the continuous object's propagation. In fourth column we mark the schemes that are capable to estimate the boundary's evolution characteristics (speed and direction). In the fifth column we mark the schemes that provide results about the boundary reconstruction accuracy. Finally, in the sixth column we mark the schemes that propose a technique that is able to automatically reconstruct the continuous object's boundary without requiring the human ability to delineate it form the boundary nodes' locations.

2.3 Current Limitations - Objectives of our method

After a thorough review of the related literature we conclude this chapter by presenting the main pitfalls of the reported schemes, which consist the motivation of this dissertation.

- **L1:** The reported continuous object tracking schemes require unrealistic sensor node densities (thousands sensors per km^2) a fact the render them impractical for real applications.
- **L2:** The majority of the proposed schemes do not consider nodes or communication failures which are totally expected in harsh environments created by the hazardous phenomena (e.g. wildfires, chemical leaks etc).
- **L3:**They require synchronization between the nodes. However, synchronizing the clocks of the sensor nodes is very difficult to achieve even in small scale WSNs.
- **L4:** All the reported schemes consider a perfect sensing mechanism which cannot be achieved in real application.
- **L5:** The majority of the reported schemes (with few exceptions [37,38]) are incapable to estimate the spatiotemporal evolution characteristics (e.g. direction and speed) of the diffusing phenomenon and therefore they cannot by exploited directly to make valuable predictions.
- **L6:** They are incapable to assess their processing, memory and energy requirements before their real field deployment.
- **L7:** The use naive techniques to reconstruct the boundary of a continuous object or are incapable to reconstruct it without using the human ability identify its shape from the locations of the boundary nodes.

Based on the above limitations in this dissertation we propose a scheme which:

- Is able to determine the boundary of an evolving continuous object using WSNs of realistic density.

- Is robust to node and communication failures which may be permanent or intermittent.
- Does not require synchronisation between the nodes.
- Assume a realistic sensing model that can capture the sensor nodes' detection distance uncertainty and the possibility of their functionality disruptions due to the hazards passing.
- Is able to estimate with accuracy the continuous object's boundary evolution parameters and to predict its spatiotemporal evolution.
- Allow us to asses with accuracy the WSN functionality as well as its energy, processing and memory requirements.
- It can reconstruct with accuracy continuous objects boundary at any time instance of its evolution.

Chapter 3

Local Front Probabilistic Modeling Approach and Parameters Estimation

One of the main contributions of this research is the relaxation of the sensor nodes' deterministic sensing mechanism assumption. This idealized sensing mechanism implies that the sensor nodes detect the boundary of a continuous object at a fixed distance. However, this assumption is unrealistic for hazard tracking, since the highly stochastic behavior of hazardous phenomena may affect the sensors and their detection distance. In this chapter we introduce a flexible probabilistic sensing modeling approach which allows us not only to capture the sensor nodes detection distance uncertainty but also to account for the possibility of sensor node malfunctions in the harsh environmental conditions potentially created by an approaching hazard. The proposed modeling, allows us to formulate the local front s speed estimation problem in a Bayesian manner. We analytically solve this Bayesian problem and derive closed-form algebraic expressions for updating all local front parameters (orientation, direction and speed) which can be easily implemented by the microprocessors of today's commodity WSN nodes.

3.1 Preliminaries

The key idea of the proposed in-network collaborative algorithm is the following: As soon as the deployed sensor nodes detect the evolving front line of a propagating hazard they

are dynamically organized into ad-hoc local clusters of 3 nodes (*triplets*). Each triplet consists of a Master sensor (S_i^M) who initiates cluster formation and two Helper sensors $\{S_j^H, S_k^H\}$ that the Master selects among the nodes in its neighborhood and uses (without them knowing it!) to update its current (prior) local front evolution belief model. The parameters of the updated (posterior) model are then propagated forward to other sensor nodes residing in the area where the evolving phenomenon is moving into.

In this section we state the basic assumptions made, the notation used, and everything else needed to facilitate the presentation of the modeling approach and collaborative in-network processing algorithm presented in later sections.

3.1.1 Sensor Network Assumptions

We assume that the deployed sensor nodes are stationary with their positions known. A sensor S_i can communicate directly only with the nodes located within its neighborhood N_i that may change dynamically (grey shaded area in Figure 3.1) and is a subset of S_i's ideal communication range (a disk of radius r). We have to note that the proposed scheme has been designed to be tolerant to communication link failures, since in real WSN deployments we expect that the parameters such as physical obstacles, adverse local conditions created by propagating hazardous phenomena, may affect the operation and/or communication capabilities of sensor nodes.

We assume that each sensor node is aware of:

- Its own location.
- The locations of its neighbors.
- A parametric model consisting of its prior belief about the local front line's evolution characteristics (see Section 3.1.3).

We assume that in a valid WSN deployment:

- Each sensor node has at least two neighbors (two nodes inside its communication range).

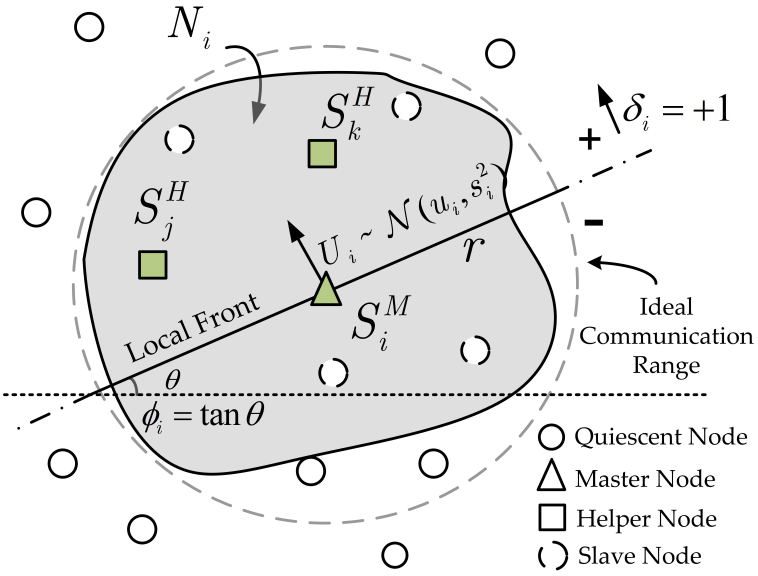


Figure 3.1: The neighborhood (grey) and local front model characteristics of sensor S_i^M.

- The local clocks of sensor nodes *do not* need to be synchronized to a global clock reference.
- Sensor nodes may fail at any time due to the hazard's propagation. Once a node fails it is assumed that it cannot communicate with its neighbors. Sensor node failures may be either permanent or intermittent.

3.1.2 Sensor Node Status

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

Quiescent (S_i^Q): Default initial status.

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

Master Candidate (S_i^C): This transitional state is entered when a sensor node checks if it satisfies the necessary conditions to become a Master (details are provided in Chapter 4).

To denote a status transition we will use the right arrow symbol $(\rightarrow)$. e.g. $S_i^Q \rightarrow S_i^C$ denotes that the Quiescent sensor node S_i^Q becomes a Master Candidate S_i^C.

Slave (S_i^L): A Slave node is responsible for monitoring the phenomenon upon receiving a

request from a Master. A Slave may serve more than one Masters at any given time.

As we can see in Figure 3.1 the neighborhood N_i of Master node S_i^M is partitioned into two half planes (positive and negative) by the local front line (a line segment of length equal to the diameter of S_i^M's ideal communication range). To refer to the neighbors of S_i^M located in the positive (negative) half plane we will use the notation N_i^+ (N_i^-) respectively. It holds that $N_i = N_i^+ \cup N_i^-$. Furthermore, we denote by N_i^0 the subset of neighbors of S_i which have not detected the phenomenon yet. It holds that $N_i^0 = N_i^{+0} \cup N_i^{-0}$, where $N_i^{+0} \subseteq N_i^+$ $(N_i^{-0} \subseteq N_i^-)$ are the subsets of neighbors which have not detected the phenomenon yet and are residing in the positive (negative) half plane respectively. Similarly we will use notation $N_i^H = \{S_j^H, S_k^H\}$ to refer to two special Slave nodes of S_i^M, called *Helpers*, that the Master S_i^M selects and uses to help him update its local front model; S_j^H is assumed to be the first selected and S_k^H the second selected Helper of S_i^M. (Details on how the Helpers are selected by the Master are provided in Chapter 4).

3.1.3 Local Front Models Parameters

Each sensor node S_i uses a parametric model to represent its belief about the local front's evolution characteristics, namely its orientation, direction and speed. This model approximates the local front as an evolving line segment of length equal to the diameter (*2r*) of the sensor's circular communication range (see Figure 3.1).

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

- ϕ_i (*Orientation*): Is the tangent of the angle formed between the local front's line and the x-axis (see Figure 3.1).

- δ_i (*Direction*): It is assumed to be always perpendicular to the local front's line segment. The direction coefficient may take one of the following values: 0, if the evolution direction is unknown; $+1(-1)$, if the local front evolves into the positive (negative) neighborhood's half plane respectively (see Figure 3.1).

- u_i, s_i (*Speed model parameters*): The speed U_i of the local front's line segment is con-

sidered to be a random variable that follows a Normal distribution $\mathcal{N}(u_i, s_i^2)$. The source of stochasticity and the use of the Normal distribution are discussed in Section 3.2.

3.1.4 Sensor Node Information and Tables

Each sensor node maintains locally the following information about itself:

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

Local Timer (t_i): Each sensor S_i starts a local timer t_i when it detects the phenomenon.

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

Detection Status Flag (DSF_i): A Boolean flag, with value 1(0), indicating that S_i has (has not) detected the phenomenon. It is set, $DSF_i \leftarrow 1$, when the hazard is detected by node S_i.

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

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

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

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

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

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

neighbors receive this request they return their information which is stored in T_i^N.

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

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

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

3.2 Modeling Detection Distance Uncertainty

It is usually assumed that a sensor node can detect an event inside a disk area of radius R_d. Although this may not always hold in real applications, it is frequently adopted since it simplifies the analysis [68--77]. Many disk based sensing models have been proposed in the literature e.g. the binary, staircase, probabilistic, etc. [74]. Among the most popular ones is the probabilistic sensing model given below,

$$p(x) = \begin{cases} 1 & x \leq R_s \\ e^{\lambda(x-R_s)^\gamma} & R_s < x < R_d \\ 0 & x \geq R_d \end{cases} \tag{3.1}$$

where the probability for a sensor node to detect an event is exponentially decreasing with distance x in the range $[R_s, R_d]$ and it is assumed that the sensor will detect an event with probability 1 (perfect sensor) if it occurs within the inner circle of radius R_s (see Figure 3.2a). The value of R_s (in the range $[0, R_d]$) is application dependent. The parameters γ and λ in equation (4.2) control the rate of probability decrease and can be determined considering the physical properties of the sensor, the noise in sensor measurements, the characteristics of the sensed physical quantity etc. [72].

We introduce a novel variation of the probabilistic sensing model which, in addition to describing the detection distance uncertainty, it also accounts for the real possibility of

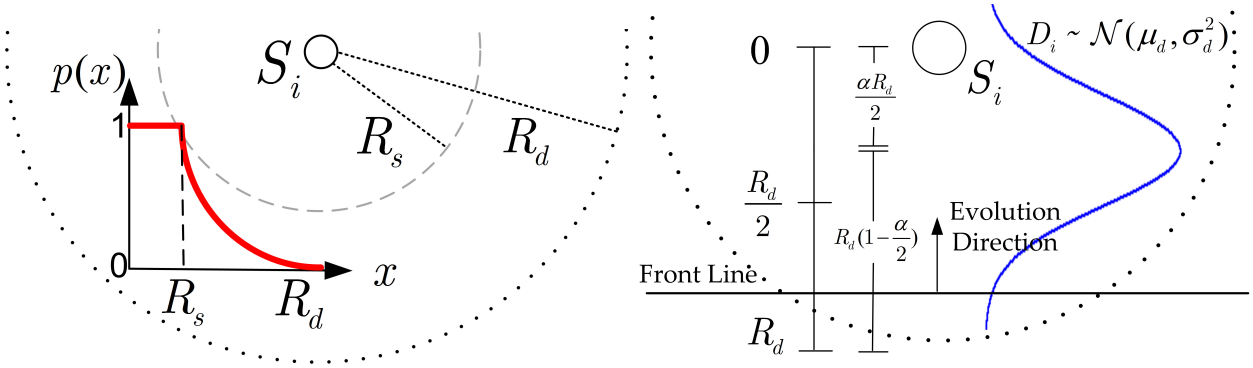


Figure 3.2: Sensing Modeling: (a) Probabilistic exponential sensing model. (b) The proposed shifted Gaussian sensing model.

a sensor node malfunctioning in a harsh environment as the hazard's front gets closer. This sensing model variation was inspired by the analysis of real WSN data collected from two outdoor experimental burns that took place at Gestosa's experimental field site in Portugal [78]. The data analysis has shown that in many cases the sensors were unable to detect the approaching fire front since abrupt increases in temperature (usually due to sudden flame fluctuations) destroyed the sensors before they detected the phenomenon (their measurements overcome a predetermined threshold).

The sensing range of a node S_i is assumed to be a circular region of radius R_d (see dotted circle in Figure 3.2b) centered at the sensor's location (L_i), as for the probabilistic model. The value of R_d is hazard specific and depends on: (i) The sensor's technical specifications (e.g. its sensitivity), (ii) how the monitored phenomenon affects the physical quantity measured by the sensor. Using this information we can estimate the expected distance at which the evolving front is detected by the sensor [49]. We set this distance equal to $\frac{\alpha R_d}{2}$, where $0 \leq \alpha \leq 1$ (see Figure 3.2b). However, due to the stochastic nature of a hazard's detection this distance may actually deviate from its expected value. To account for this stochasticity we treat the detection distance as a normally distributed random variable, $D_i \sim \mathcal{N}(\mu_d, \sigma_d^2)$, with parameters:

$$\mu_d = \frac{\alpha R_d}{2}, \quad 3\sigma_d = R_d(1 - \frac{\alpha}{2}) \Rightarrow \sigma_d = \frac{R_d}{3}(1 - \frac{\alpha}{2}). \tag{3.2}$$

In setting the standard deviation as in (4.1) above we assumed that the probability for a sensor to detect the approaching diffusive phenomenon at a distance larger than R_d is

negligible.

As observed in Figure 3.2b the probability of detection increases monotonically as the distance of the local front from the sensor decreases in the range $\left[\frac{\alpha R_d}{2}, R_d\right]$. However, in close range $\left[0, \frac{\alpha R_d}{2}\right]$ the probability of detection decreases. This modeling decision is justified considering that the inability of a sensor to detect the approaching front at the expected detection range $\left[\frac{\alpha R_d}{2}, R_d\right]$ is an indication of a potential hazard-induced malfunction reducing the probability of detecting the hazard as it gets closer to the sensor node. This simple and realistic, sensing model in the presence of propagating hazards allows us to capture both the inherent stochasticity associated with the detection distance as well as the sensor node's increasing probability to malfunction as the hazard gets in close range. Importantly, it does not harm at all the generality since by setting the parameter $\alpha = 0$ in equation (4.1) (i.e. $\mu_d = 0$) we can relax the assumption that a node may malfunction and revert back to a monotonic probabilistic sensing model centered at the sensor node's location. The proposed "shifted" Gaussian model is therefore very flexible since it can cover both scenarios: diffusive hazards which may, or may not, affect the functionality of deployed sensor nodes. This is in contrast to the classical monotonic probabilistic model which ignores the real possibility of sensing mechanism failures as the hazard propagates in close range. We have tested how the proposed algorithm (to be presented in Chapter 4) performs when the ``real'' sensing distance model deviates from the assumed ``shifted'' Gaussian model (mismatch conditions).

We also remark that the Gaussian distribution has been used by many researchers to describe the dependence of sensor node detection probability to distance ([71, 75--77]) since it has all the necessary ingredients to characterize the uncertainty while offering a simple parameterization. We will show later in this section that an added advantage is that it also leads to simple algebraic expressions for updating the local front model parameters. This is important because such calculations can be easily performed by the embedded microprocessors of WSN nodes which have limited computing power and operate under a strict energy budget.

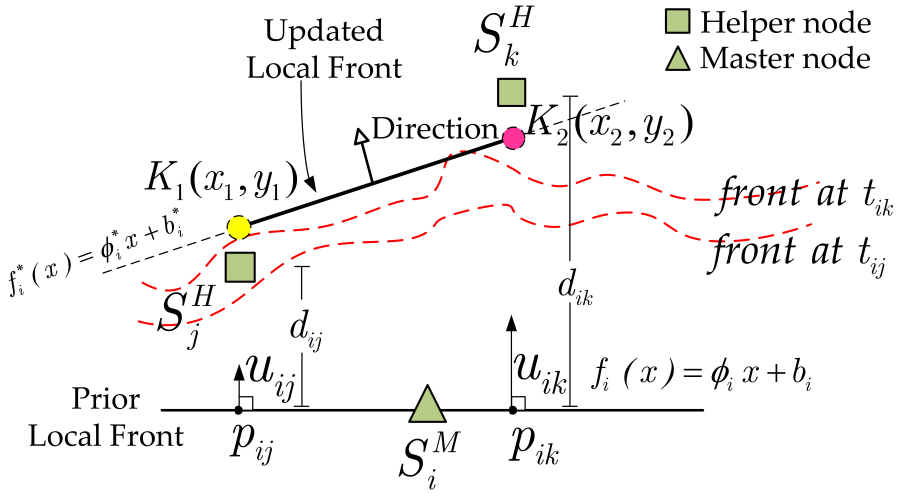


Figure 3.3: Updating the local front model.

3.3 Modeling Speed Uncertainty

Let us now consider the cluster of three sensor nodes (triplet) shown in Figure 3.3. As soon as the Master node S_i^M receives two *Detection Messages* (DM), one message from each one of its two Helpers, $\{S_h^H$, where $h \in \{j,k\}\}$, it has all the information it needs to start updating its prior model.

Using the coordinates of its Helpers ($L_h = (x_h, y_h)$ stored in its table T_i^N) and the coordinates of their projection points $\{p_{ih} = (x_{ih}, y_{ih}),\ h \in \{j,k\}\}$ on its local front line, Master S_i^M can calculate the Euclidean distances $\{d_{ih},\ h \in \{j,k\}\}$ using equation (3.3) below (see also Figure 3.3),

$$d_{ih} \equiv dist(L_h, p_{ih}) = \sqrt{(x_h - x_{ih})^2 + (y_h - y_{ih})^2}. \tag{3.3}$$

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

$$D_{ih} = d_{ih} - D_h, \quad h \in \{j,k\} \tag{3.4}$$

Since D_h, the detection distance of the progressing front from Helper $S_h,\ h \in \{j,k\}$, follows a Normal distribution $\mathcal{N}(\mu_d, \sigma_d^2)$, D_{ih} will also follow a Normal distribution $\mathcal{N}(\mu_{ih}, \sigma_{ih}^2)$

with parameters:

$$\mu_{ih} = d_{ih} - \mu_d = d_{ih} - \frac{\alpha R_d}{2}, \quad \sigma_{ih} = \sigma_d = \frac{R_d}{3}(1 - \frac{\alpha}{2}), \tag{3.5}$$

where d_{ih} has been computed using equation (3.3), $h \in \{j, k\}$.

Upon estimating the parameters μ_{ih} and $\sigma_{ih}, h \in \{j, k\}$ using equation (3.5), Master node S_i^M can calculate the speed at which the two Helper projection points, p_{ij} and p_{ik}, have to move forward in order to cover the distances D_{ij} and D_{ik} in the measured time intervals t_{ij} and t_{ik} respectively (see Figure 3.3). Since D_{ij} and D_{ik} are random variables that follow a Normal distribution, it can be shown that the corresponding speeds of the two projection points, U_{ij} and U_{ik}, will also follow Normal distributions of the form $U_{ih} \sim \mathcal{N}(u_{ih}, s_{ih}^2)$. Their parameters can be computed easily using equations (3.6) below, $h \in \{j, k\}$:

$$u_{ih} = \frac{\mu_{ih}}{t_{ih}} = \frac{2d_{ih} - \alpha R_d}{2t_{ih}}, \quad s_{ih} = \frac{\sigma_{ih}}{t_{ih}} = \frac{R_d(1 - \frac{\alpha}{2})}{3t_{ih}} \tag{3.6}$$

3.4 Model Parameters Updating

3.4.1 Speed

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

As discussed in Section 3.1.3 the local front's speed is a Normal random variable. To update its parameters (mean and variance), Master node S_i^M uses its prior speed information $U_i \sim \mathcal{N}(u_i, s_i^2)$ (parameters are stored in its prior model m_i) as well as the likelihood computed using information related to the ``observed'' speeds (U_{ih}) of the two Helper node projection points on the current local front, namely $\{p_{ih}$, where $h \in \{j, k\}\}$.

Since the number of the available "observations" is very small (only two), we introduce below a technique which exploits the availability of information about the uncertainty

(s_{ih}) associated with the speed ``observations" (U_{ih}) to improve the likelihood estimation accuracy.

Having available its local prior model and the parameters of the speed models $U_{ih} \sim \mathcal{N}(u_{ih}, s_{ih}^2)$ (computed using the equations in (3.6)), the Master node S_i^M computes the weights $\{w_{ih},\, h \in \{j,k\}\}$ of a Gaussian mixture model with two components (see Figure 3.4b)

$$p(u) = \sum_{h\in\{j,k\}} w_{ih}\mathcal{N}(u|u_{ih}, s_{ih}^2), \tag{3.7}$$

as follows:

$$w_{ij} = \frac{1}{1+C},\;\; w_{ik} = \frac{C}{1+C},\;\; C = \frac{s_{ij}|u_i - u_{ij}|}{s_{ik}|u_i - u_{ik}|}. \tag{3.8}$$

Fixing the mixture weights as in (3.8) is justified based on the following arguments:

- The speed model $U_{ih}, h \in \{j,k\}$ with the smaller standard deviation (smaller uncertainty) should be trusted more by the Master.

- Since in short time periods (e.g. the time interval between two successive local model updates) environmental diffusive phenomena tend to exhibit smooth changes in terms of their evolution characteristics (speed and direction), more trust should be assigned to the speed ``observation" with mean value (u_{ih}) closer to that of the prior model (u_i).

Estimating the posterior parameters by using directly the Gaussian mixture likelihood (3.7) and Bayes rule would be computationally expensive since analytical closed form expressions cannot be derived. Due to the limited processing capabilities and low power constraints of microprocessors used in WSN nodes, in this work we consider prohibitive the use of an iterative, slowly converging, parameters estimation procedure. Therefore, in order to be able to derive closed form algebraic expressions for the posterior distribution parameters we employ variational calculus and approximate the Gaussian mixture by a Normal distribution. To this end, we estimate the parameters of the Normal distribution that minimizes the Kullback-Leibler (KL) divergence (maximizes the similarity) from the Gaussian mixture. The general form of the equations which can be used to compute the parameters of this Normal distribution are [79, 80]:

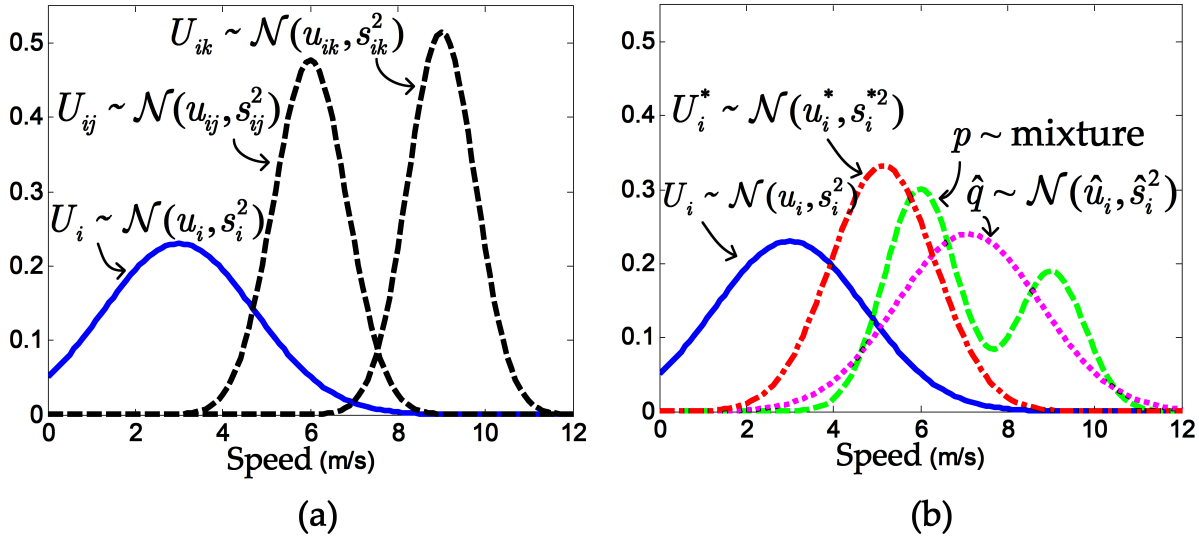


Figure 3.4: Speed model updating procedure. (a) The prior model (U_i), and the two ``observation'' models ($U_{ij}, U_{ik}$). (b) The mixture model $p$ resulting by combining speed ``observation'' models; the normal distribution $\hat{q}$ that best approximates p by minimizing the Kullback Lebier divergence $(KL(p||\hat{q}))$; the resulting posterior speed model U_i^*.

$$\hat{\mu} = \sum_n w_n \mu_n \tag{3.9}$$

$$\hat{\Sigma} = \sum_n w_n (\Sigma_n + (\mu_n - \hat{\mu})(\mu_n - \hat{\mu})^T) \tag{3.10}$$

In our specific case these equations reduce to:

$$\hat{u}_i = w_{ij} u_{ij} + w_{ik} u_{ik} \tag{3.11}$$

$$\hat{s}_i^2 = w_{ij} s_{ij}^2 + w_{ik} s_{ik}^2 + w_{ij} w_{ik} (u_{ij} - u_{ik})^2 \tag{3.12}$$

Having computed the mixture weights using (3.8), Master S_i^M calculates the Normal distribution parameters $\hat{u}_i$ and $\hat{s}_i^2$ using equations (3.11) and (3.12). By applying simple manipulations on the Bayes theorem it can be proved [81--83] that since the prior $\mathcal{N}(u_i, s_i^2)$ and the likelihood $\mathcal{N}(\hat{u}_i, \hat{s}_i^2)$ are both Gaussian, the posterior will also be a Gaussian $\mathcal{N}(u_i^*, s_i^{*2})$ (conjugate distributions, see Figure 3.4b) with parameters provided by the following easy to compute closed form expressions:

$$u_i^* = \frac{u_i \hat{s}_i^2 + \hat{u}_i s_i^2}{\hat{s}_i^2 + s_i^2}, \quad s_i^{*2} = \frac{\hat{s}_i^2 s_i^2}{\hat{s}_i^2 + s_i^2} \tag{3.13}$$

3.4.2 Orientation

Let K_1 (K_2) be the point to be reached by p_{ij} (p_{ik}) as it moves in the direction of the local front's evolution with speed u_{ij}, (u_{ik}) respectively for a time interval t_{ik} (see Figure 3.3). The coordinates of K_1 and K_2, to be called (x_1, y_1) and (x_2, y_2), can be found by solving a system of a linear and a quadratic equation. This problem is formulated and solved analytically in Appendix A. Using the calculated coordinates, Master S_i^M can update the orientation parameter of its local front model using equation (3.14) below,

$$\phi_i^* = \frac{y_2 - y_1}{x_2 - x_1}. \tag{3.14}$$

3.4.3 Evolution Direction

To update the direction parameter δ_i^*, node S_i^M derives the equation of line $f_i^*(x)$ that is defined by points $K_1(x_1, y_1)$ and $K_2(x_2, y_2)$ (see Figure 3.3).

$$f_i^*(x) = \phi_i^* x + b_i^* \tag{3.15}$$

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

Subsequently, node S_i^M substitutes its abscissa (x_i) in (3.15) and checks the $sgn(f_i^*(x_i))$. If $sgn(f_i^*(x_i)) > 0$ ($sgn(f_i^*(x_i) < 0$) then Master node S_i^M infers that the new local front line evolves into the *negative (positive)* half plane and it updates the direction parameter $\delta_i^* = -1(1)$ accordingly.

In this chapter we presented a novel flexible Gaussian sensing model that allow us to capture sensor nodes' detection distance uncertainties and possible disruptions of their functionality. Based on the detection distance uncertainty we formulate a Bayesian parameter estimation problem which solved analytically. As we will present in Chapter 4 the derived algebraic closed formed expressions allow us to estimate with accuracy the local evolution parameters of a continuous objects boundary, while respecting the strict

processing and energy constraints of the commodity sensor nodes (see Chapter 5).

Chapter 4

Collaborative Algorithm for Estimating the Spatiotemporal Evolution of Continuous Objects

In this Chapter we present the proposed collaborative WSN algorithm for estimating and tracking the local evolution characteristics of continuous objects. Using extensive simulation results, we demonstrate its ability to estimate with accuracy the evolution characteristics of complex continuous objects, using realistic WSN densities while allocating also sensor node and communication link failures.

4.1 Preliminaries

4.1.1 Sensor Messages

The proposed in-network algorithm assumes that each sensor node can handle the following messages (the attributes carried by each message are provided in parenthesis):

Broadcast type Messages:

Detection Message $(\mathtt{DM}(\mathtt{ID_i}))$: It is broadcasted by a sensor node S_i to notify its neighbors that it has detected the phenomenon (detection event).

Master Declaration Message $(\mathtt{MDM})$: It is broadcasted by a node to notify its neighbors that

it satisfies the necessary conditions to assume a Master's role (to be presented in Section 4.2.1). This message has two slightly different versions: $\texttt{MDM1}(\texttt{ID}_\texttt{i}, \texttt{PM}_\texttt{i})$ and $\texttt{MDM2}(\texttt{ID}_\texttt{i})$.

Update Prior Message $(\texttt{UPM}(\texttt{ID}_\texttt{i}, \texttt{UM}_\texttt{i}))$: It is broadcasted by a Master node S_i^M after it has updated its prior model.

Free Slaves Message $(\texttt{FSM}(\texttt{ID}_\texttt{i}))$: It is broadcasted by a Master node S_i^M in order to release its Slaves.

Pass My Posterior Message $(\texttt{PMPM}(\texttt{ID}_\texttt{i}))$: It is broadcasted by a Master node S_i^M when none of its Helpers satisfies the necessary conditions to become the new Master.

Pass Posterior Message $(\texttt{PPM}(\texttt{UM}_\texttt{i}))$: It is broadcasted by the sensor nodes $\{S_m^L \in N_i\}$ enslaved to a Master node (S_i^M) after they have received a $\texttt{PMPM}(\texttt{ID}_\texttt{i})$.

Unicast type Messages:

Master Declaration Message Acknowledgement $(\texttt{MDMA}(\texttt{ID}_\texttt{j}))$: it is sent by a Slave node (S_j^L) to notify its Master (S_i^M) that it has received its MDM message and has become its Slave.

Master Offer Message $(\texttt{MOM}(\texttt{ID}_\texttt{i}))$: It is sent by a Master node (S_i^M) to one of its Helpers to make it an offer to become the new Master.

Accept Master Offer Message $(\texttt{AMOM}(\texttt{ID}_\texttt{j}))$: It is sent by a Helper (S_j^H) to a Master to acknowledge that it accepts the offer to become the new Master.

Decline Master Offer Message $(\texttt{DMOM}(\texttt{ID}_\texttt{j}))$: It is sent by a Helper (S_j^H) to a Master to notify it that it does not accepts the offer to become the new Master.

4.1.2 Sensor Network Assumptions

To better explain the proposed in-network algorithm we will use a running example to facilitate the understanding of its operations. Let's assume, without loss of generality (*w.l.o.g.*) that a part of the evolving front has just entered the WSN's deployment region and none of the sensor nodes, (which are currently in the default Quiescent status), has detected the phenomenon yet (DSF = 0). Each node is equipped with sensors that can measure physical parameters affected by the phenomenon's presence when it enters in its sensing range (a circle of radius R_d - see Chapter 3 for details). All sensor nodes are

initialized with the same prior model with parameters $m_i = \{\phi_i, \delta_i, u_i, s_i\}$ and $\delta_i = 0$.

4.2 Collaborative In Network Algorithm

4.2.1 Forming a Local Cluster

Let's now assume that the evolving front enters the sensing range of node S_i (see Figure 4.1a). As soon as S_i detects the front, it initiates the *Detection Procedure* described below and also summarized by the UML sequence diagram of Figure 4.2.

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

- If $SS_i = 0$ (S_i is Quiescent, as in the example's case) the node makes a status transition, $S_i^Q \rightarrow S_i^C$ ($SS_i \leftarrow 1$), and initiates the *Master Check Necessary Conditions Procedure* (presented in the next paragraph).

- If $SS_i = 2$ (S_i is a slave), node S_i^L broadcasts a *Detection Message* $\texttt{DM(ID}_\texttt{i}\texttt{)}$. Each neighbor $\{S_m \in N_i$, where $m = \{j, k, l\}\}$ in Figure 4.1a when it receives this message it updates in its neighborhood table T_m^N (in the row corresponding to S_i) the attributes t_{mi} and DSF_{mi} (see Figure 4.2). The value assigned to t_{mi} is the time value t_m indicated by the local timer of S_m when message $\texttt{DM(ID}_\texttt{i}\texttt{)}$ was received. If S_m receives message $\texttt{DM(ID}_\texttt{i}\texttt{)}$ before its local timer has been started (this can happen if S_m has not sensed the phenomenon yet) it assigns to attribute t_{mi} a $null$ value.

Master Check Necessary Conditions Procedure: Sensor node S_i^C finds in its table T_i^N the subset of neighbors which have not detected the phenomenon yet (i.e. $\{S_m \in N_i^0\}$). Based on the cardinality $|N_i^0|$ of this set, S_i^C proceeds as follows: (see UML sequence diagram in Figure 4.3).

- ***If*** $|N_i^0| < 2$: Node S_i^C transitions back to the Quiescent state, $S_i^C \rightarrow S_i^Q$, and broadcasts a $\texttt{DM(ID}_\texttt{i}\texttt{)}$ message. Each receiving neighbor $\{S_m \in N_i\}$ updates its attributes t_{mi} and DSF_{mi} in its table T_m^N, in the way already discussed in the *Detection Procedure* paragraph and shown in Figure 4.2.

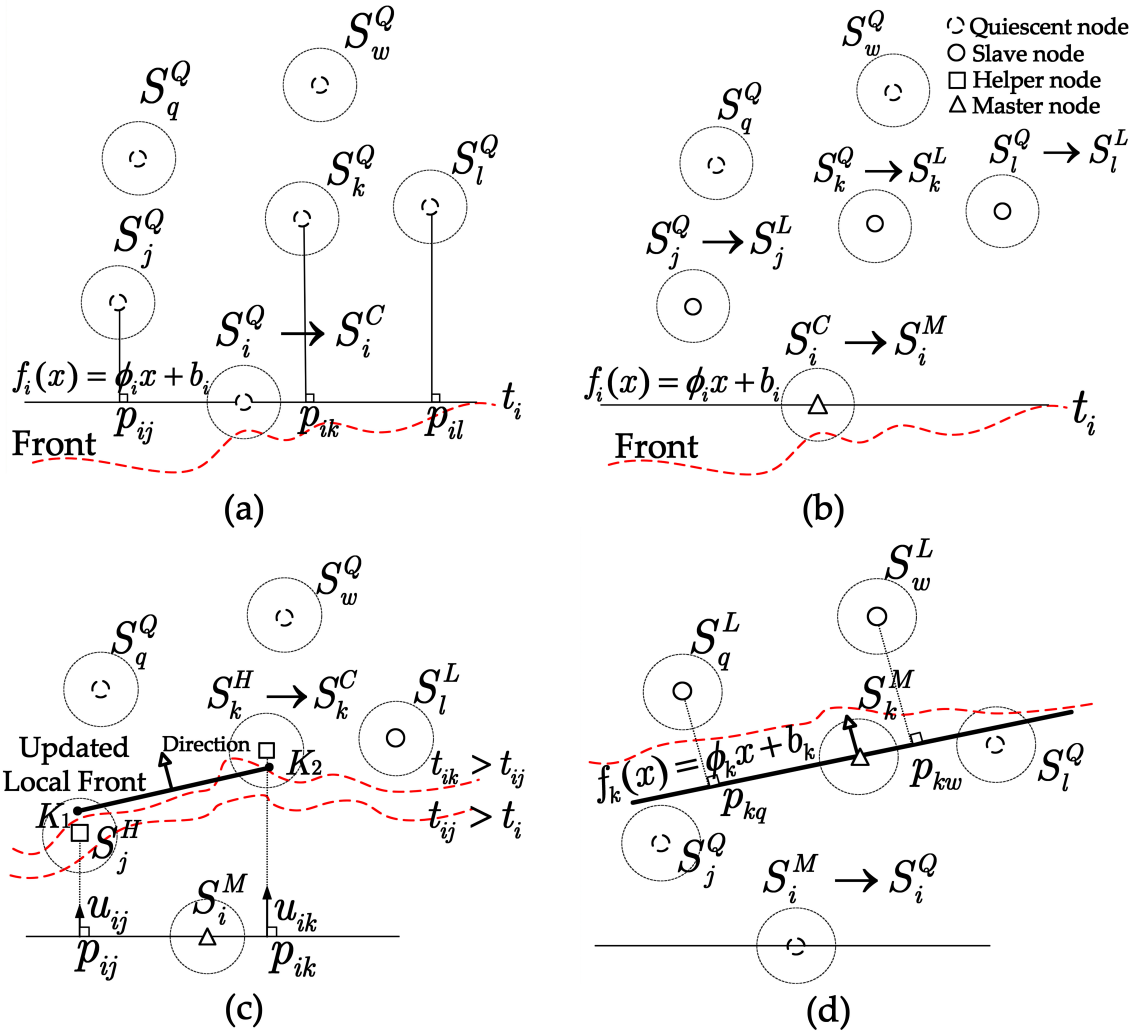


Figure 4.1: Local front model updating procedure: (a) Node S_i becomes Master candidate and checks if it satisfies the conditions to become a Master, (b) node S_i becomes a Master and ``enslaves'' its neighbors S_j, S_k and S_l, (c) Master S_i^M uses the information received from its two Helpers (S_j^H and S_k^H) and updates the local front's line parameters, (d) node S_k becomes the new Master and S_i releases its slaves.

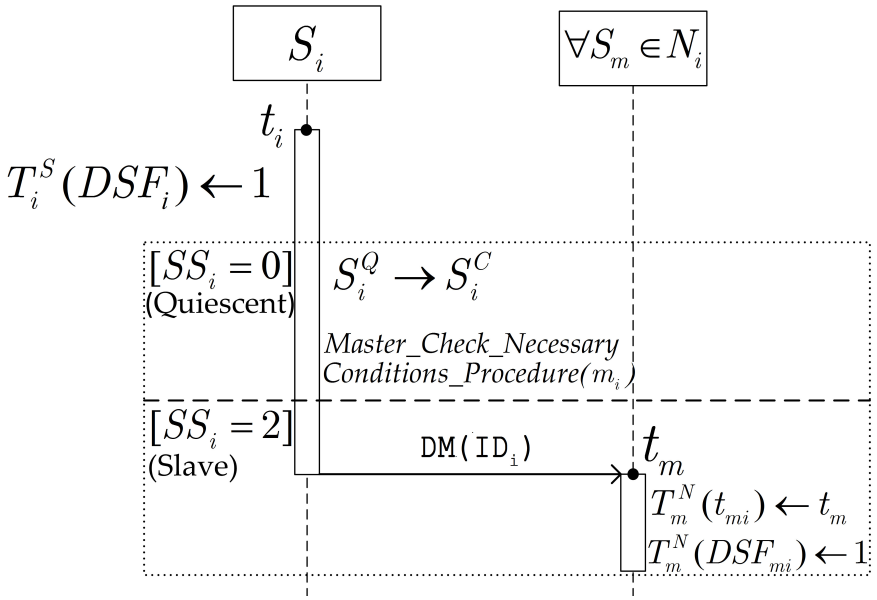


Figure 4.2: *Detection Procedure* (UML sequence diagram).

- *If* $|N_i^0| \geq 2$ (example's case): Node S_i^C initiates the *Local Front Line Derivation* activity and then checks the evolution direction parameter δ_i of its initial model m_i. Based on the value of δ_i, S_i^C initiates the appropriate *Create Helpers Table* activity followed by the *Master Declaration* activity (see Figure 4.3).

Local Front Line Derivation: Node S_i^C uses the *orientation* parameter ϕ_i of its initial model m_i and its location information $L_i = (x_i, y_i)$, to derive the equation of the line where the local front segment belongs, $f_i(x) = \phi_i x + b_i$, where $b_i = y_i - \phi_i x_i$ (see Figure 4.1a).

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

- If $\delta_i = +1$ (the local front evolves into the positive half plane), S_i^C searches in its neighborhood table T_i^N to find the subset of neighbors that belong to the positive neighborhood half plane and have not detected the phenomenon yet (N_i^{+0}). If $|N_i^{+0}| \geq 2$, S_i^C calculates the coordinates of these neighbors' projections on the local front line (see Appendix A for details). These are the points p_{ij}, p_{ik}, p_{il} in the example (Figure 4.1a).

  Then S_i^C calculates the Euclidean distances among all possible projection pairs and identifies those pairs with distances larger than a pre-specified threshold (that is application dependent). These pairs are considered to be the *legitimate Helpers pairs*, in

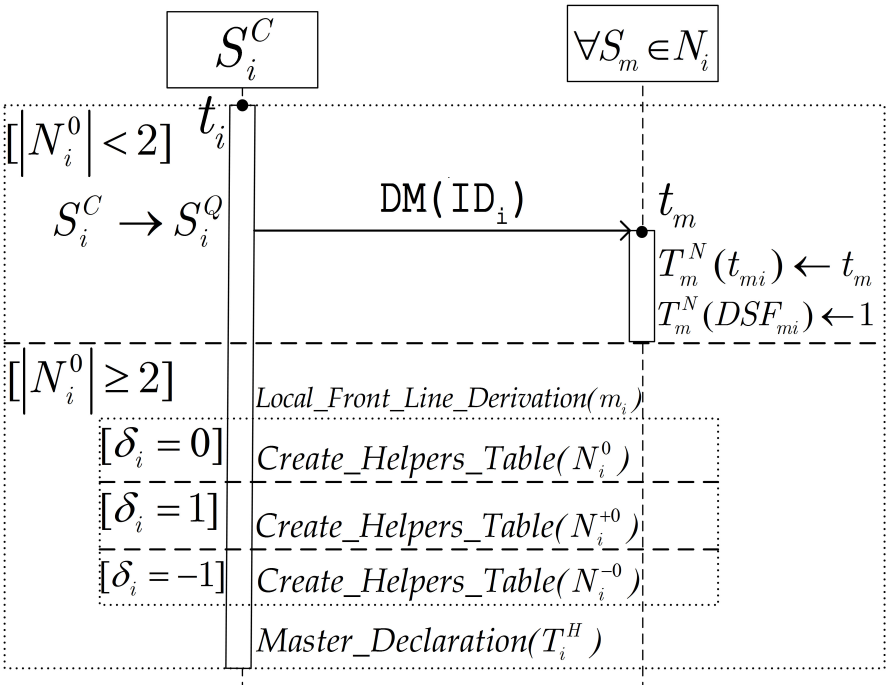


Figure 4.3: *Master Check Necessary Conditions Procedure* (UML sequence diagram).

the sense that any one of them could be used by S_i^C to update its prior model. Finally S_i^C stores the IDs of nodes of legitimate Helper pairs in its respective Helpers table T_i^H. How a particular Helpers pair is selected among the legitimate ones will be discussed in Section 4.2.2.

- If $\delta_i = -1$ (the local front evolves into the negative half plane), S_i^C performs the same aforementioned steps but for the nodes which belong to the negative neighborhood half plane and have not detected the phenomenon yet (subset N_i^{-0}).

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

Master Declaration: After the end of the *Create Helpers Table* activity node S_i^C checks its Helpers Table T_i^H:

If $T_i^H = \oslash$: S_i^C does not become a Master, transitions back, $S_i^C \rightarrow S_i^Q$, and broadcasts a

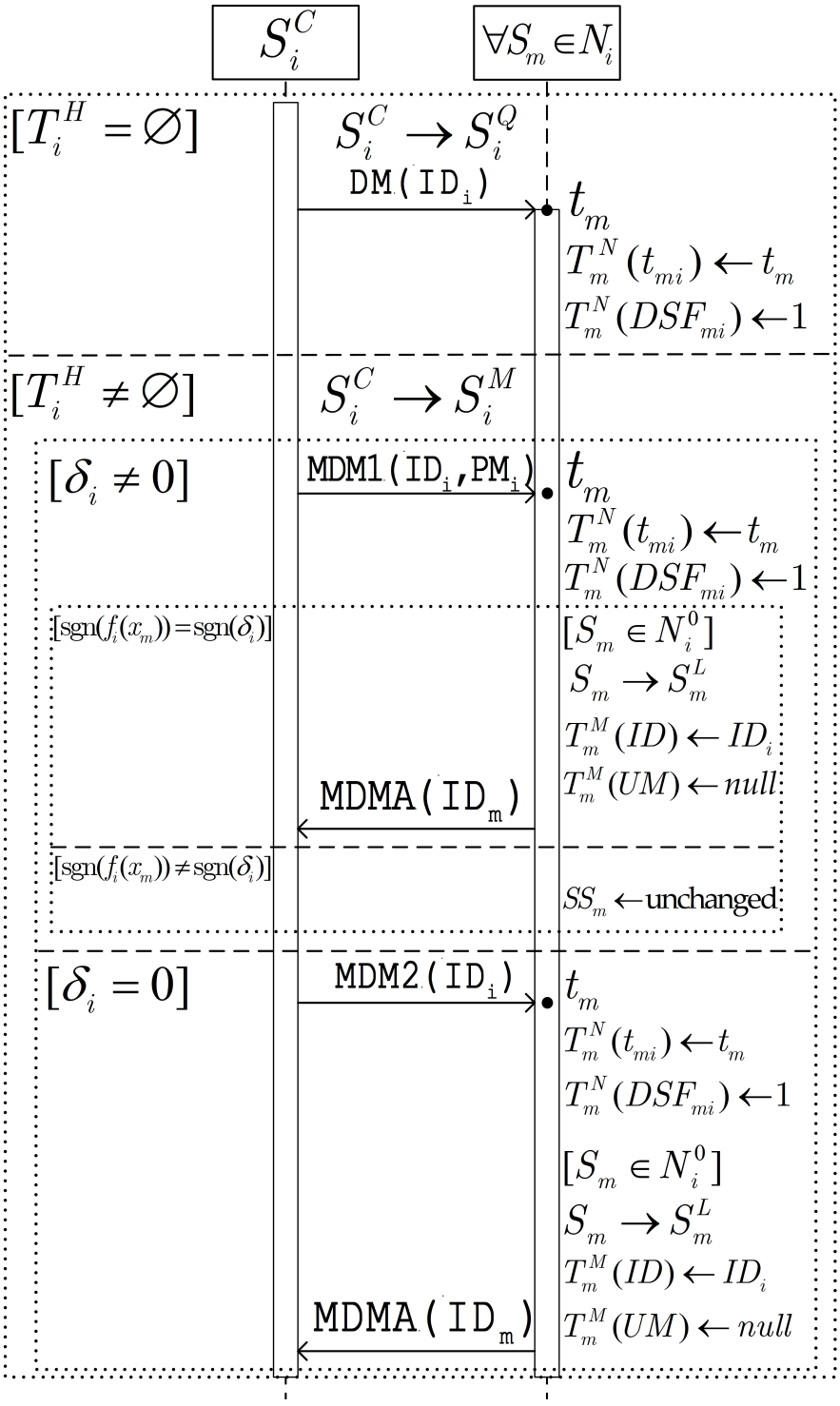


Figure 4.4: *Master's Declaration* (UML sequence diagram).

$\texttt{DM}(\texttt{ID}_\texttt{i})$. The receiving neighbors $\{S_m \in N_i\}$ update their attributes t_{mi} and DSF_{mi} in their tables T_m^N (see Figure 4.4).
If $T_i^H \neq \oslash$ (i.e. there exist at least one pair of legitimate Helpers for S_i^C): Node S_i^C becomes a Master, $S_i^C \rightarrow S_i^M$, and checks its model's evolution direction parameter (δ_i):

- If $\delta_i \neq 0$, S_i^M broadcast a *Master Declaration Message of type 1,* $(\texttt{MDM1}(\texttt{ID}_\texttt{i}, \texttt{PM}_\texttt{i}))$. Each neighbor $\{S_m \in N_i\}$ when it receives this message it updates the attributes t_{mi} and DSF_{mi} in the corresponding row of its table T_m^N, as shown by the UML sequence diagram of Figure 4.4. Moreover, each $\{S_m \in N_i^0\}$ uses the Master's S_i^M initial model (PM_i is carried in $\texttt{MDM1}$), derives the equation of the local front line $f_i(x)$ and substitutes its abscissa (x_m) in the argument of $f_i(x)$. If $sgn(f_i(x_m)) = sgn(\delta_i)$ (i.e. if S_m belongs to the half plane, with respect to the Master, that the front evolves into), S_m becomes a slave, $S_m \rightarrow S_m^L$, adds a row in its Masters table T_m^M for Master S_i^M with attributes $\{ID \leftarrow ID_i, UM \leftarrow null\}$ and sends a *Master Declaration Message Acknowledgement* $(\texttt{MDMA}(\texttt{ID}_\texttt{m}))$ back to node S_i^M. Otherwise S_m keeps its status unchanged.

- If $\delta_i = 0$ (example's case), Master S_i^M broadcasts a *Master Declaration Message of type 2* $(\texttt{MDM2}(\texttt{ID}_\texttt{i}))$. Each neighbor $\{S_m \in N_i\}$ receiving this message updates the attributes t_{mi} and DSF_{mi} in the corresponding row of its table T_m^N (see Figure 4.4). Moreover, if $\{S_m \in N_i^0\}$ it becomes a slave ($S_m \rightarrow S_m^L$) (see Figure 4.1b) adds a row in his T_m^M for Master S_i^M with attributes $\{ID \leftarrow ID_i, UM \leftarrow null\}$ and sends a *Master Declaration Message Acknowledgement* $(\texttt{MDMA}(\texttt{ID}_\texttt{m}))$ to node S_i^M.

When node S_i^C becomes a Master it waits until it receives two detection messages $(\texttt{DMs})$, one message from each one of the two nodes of a legitimate Helpers pair (among those pairs stored in its local Helpers Table T_i^H - see Figure 4.5). However, the potentially adverse conditions created by the propagation of a diffusive hazard may impair the communication between the Master and its Helpers. In the a worst case scenario the Master may never receive the $\texttt{DM}$ messages sent by its Helpers and thus never update its local model parameters (formation of a ``zombie'' cluster). We should emphasize that the possible formation of ``zombie'' clusters does not affect the global functionality of the algorithm since the model updates within "healthy" clusters will normally take place. Nevertheless,

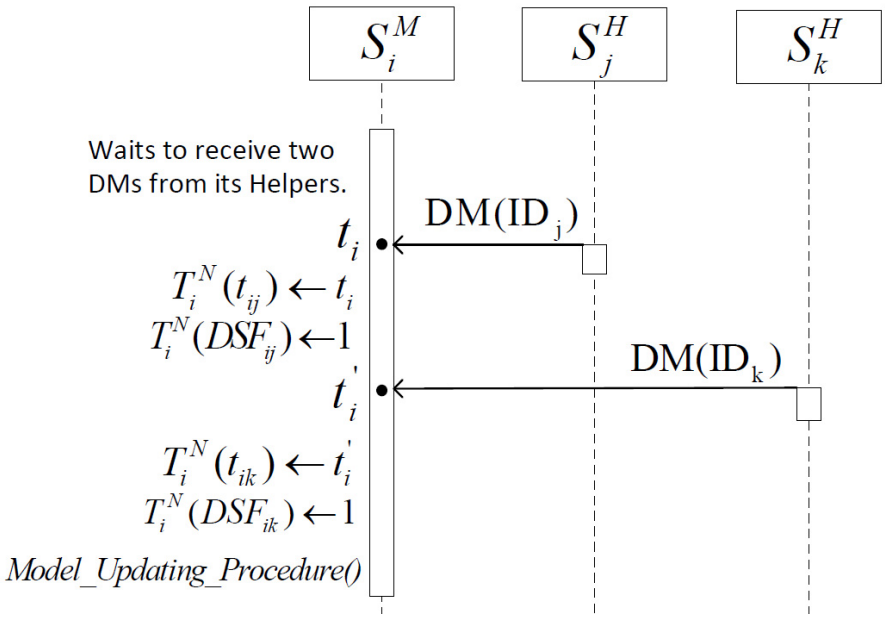


Figure 4.5: *Updating Model Parameters* (UML sequence diagram).

to reduce the probability of a ``zombie'' cluster formation the Master node implements the following procedure:

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

- If they do, Master node S_i^M keeps its status unchanged and waits until it receives the two detection messages $(\texttt{DMs})$ that will be used to update its model parameters (see Section 4.2.2).

- If they do not, Master node S_i^M broadcasts a *Free Slaves Message* $\texttt{FSM}(\texttt{ID}_\texttt{i})$ and changes its status ($S_i^M \rightarrow S_i^Q$). When the slaves $\{S_m^L \in N_i\}$ receive the `FSM` message, they remove from their tables T_m^M the information corresponding to Master S_i^M and if they do not serve any other Master(s), they change their status back to Quiescent ($S_m^L \rightarrow S_m^Q$).

4.2.2 Model Updating

In our example we assume *w.l.o.g.* that the two messages received by S_i^M come from $N_i^H = \{S_j^H, S_k^H\}$ (see Figure 4.1c). Furthermore it is assumed that the two Helpers have

detected the evolving front at time instances t_{ij} and t_{ik} respectively, where *w.l.o.g.* $t_{ij} < t_{ik}$. When the Master S_i^M receives the DMs from the pair of Helpers it updates its neighborhood table T_i^N and initiates the procedure described below.

Model Updating Procedure: The updating starts with the calculation of the ``new'' (posterior) local front speed model parameters ($U_i^* \sim \mathcal{N}(u_i^*, s_i^{*2})$). Master node S_i^M uses the expressions in (3.6) and calculates the parameters of the Normal speed models of the two Helper projection points p_{ij} and p_{ik} (see Section 3.3). By substituting these parameter values in (3.8), S_i^M calculates the Gaussian mixture weights w_{ij} and w_{ik} (see Section 3.4). Then, by applying the resulting mixture weight values into (3.11) and (3.12) the Master calculates the parameters ($\hat{u}_i$ and $\hat{s}_i$) of the Normal distribution that best approximates the Gaussian mixture. Finally, having available these parameters ($\hat{u}_i$ and $\hat{s}_i$), along with the prior model parameters (u_i, and s_i), S_i^M applies them to equation (3.13) to obtain parameters (u_i^*, s_i^{*2}) of the posterior speed model.

Next, Master S_i^M estimates the local front's orientation, ϕ_i^*. As discussed in Section 3.4.2 to update this parameter the Master finds the coordinates of two points, $K_1 = (x_1, y_1)$ and $K_2 = (x_2, y_2)$ (see Appendix A), which are expected to lie on the ``new'' local front line (see Figure 4.1c), and applies them directly to equation (3.14). Finally, S_i^M follows the procedure described in Section 3.4.3 and updates the evolution direction parameter, δ_i^*. All model parameters are updated using closed form expressions that can be realized easily by embedded microprocessors commonly used in WSN node architectures.

4.2.3 Model Propagation

After updating its model, Master S_i^M initiates the *Model Propagation Procedure* (see the UML sequence diagram of Figure 4.6).

Master node S_i^M first broadcasts an *Update Prior Message* ($\texttt{UPM}(\texttt{ID}_\texttt{i}, \texttt{UM}_\texttt{i})$). The sensors which serve it $\{S_m^L \in N_i\}$ when they receive it update the prior model information in their tables T_m^S using the received model m_i^*. Moreover, they update attribute $UM_i \leftarrow m_i^*$ in the corresponding row of their Masters' table T_m^M. In addition, S_i^M sends a *Master Offer Message* ($\texttt{MOM}(\texttt{ID}_\texttt{i})$) to the Helper who detected most recently the phenomenon (it is S_k^H $w.l.o.g.$ in the running example of Figure 4.1c) and asks it to become the new Master. This

sensor node becomes temporarily a Master candidate ($S_k^H \rightarrow S_k^C$) and uses the updated model parameters (m_i^*) to initiate the *Master Check Necessary Conditions procedure* (see Section 4.2.1).

If Helper S_k^C meets the conditions to become the new Master (example's case), it accepts the offer ($S_k^C \rightarrow S_k^M$) and responds to S_i^M with an *Accept Master Offer Message* ($\texttt{AMOM}(\texttt{ID}_\texttt{k})$). When S_i^M receives this message it broadcasts a *Free Slaves Message* $\texttt{FSM}(\texttt{ID}_\texttt{i})$ and changes its status back to default ($S_i^M \rightarrow S_i^Q$, see Figure 4.1d). Each Slave S_m^L, when it receives the FSM removes from its table T_m^M the information corresponding to Master S_i^M and if it does not serve any other Master(s), it changes its status back to Quiescent ($S_m^L \rightarrow S_m^Q$).

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

At this point we want to mention the following interesting scenario that can be handled without any problem: A Helper node may be enslaved to more than one Masters (if it belongs to the intersection of their clusters). If this Helper node receives a MOM from one of its Masters it may accept the offer (if it satisfies the necessary conditions) while it also continues to serve as Helper to another Master. If now this new Master receives (before it updates its model parameters) a MOM from a second Master, it will checks (for the corresponding model) the ``*Master check Necessary Conditions*'' and if they are satisfied it will accepts that offer as well. By the end of this procedure the new Master has formed for each accepted offer a T^H table that contains the legitimate Helper pairs that could be

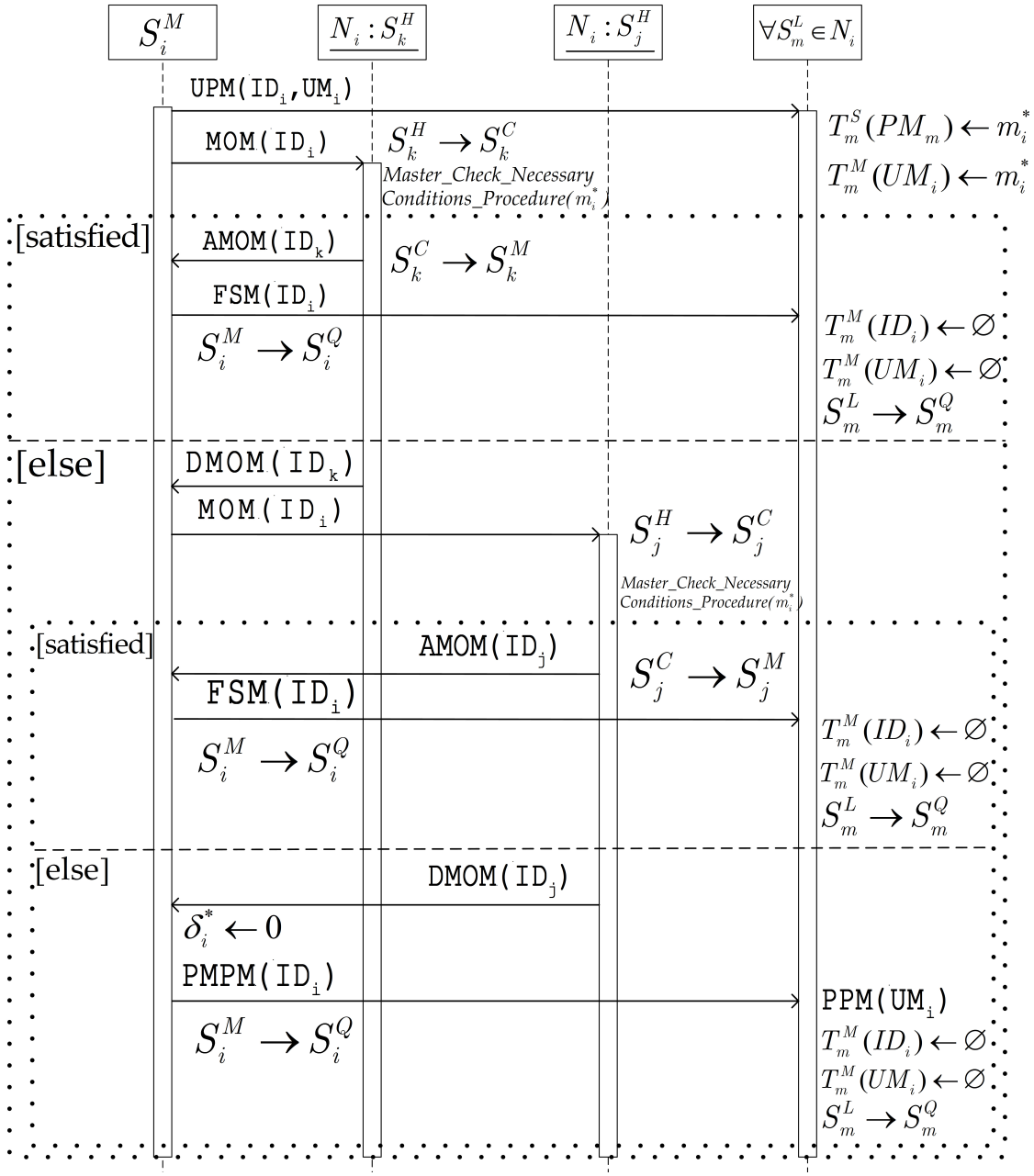


Figure 4.6: *Model Propagation Procedure* (UML sequence diagram).

used for updating each model. The new Master waits to receive the DMs from two of its neighbors that constitute a legitimate Helpers pair in at least one of its T^H tables. If the pair that responds is contained in only one of its T^H tables then the node updates only the corresponding model and discards the rest. Otherwise, if the same Helpers pair is present in more than one T^H tables, the node updates the corresponding models and selects among them to propagate the one with the smallest speed variance (smallest speed uncertainty).

4.3 Evaluation Setup

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

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

4.3.1 WSN Simulation Workflow

The WSN simulation workflow includes two main components: i) The flexible WSN simulator COOJA (COntiki Os JAva) [84] for the Contiki sensor node operating system, and ii) a Matlab-based component which prepares the COOJA input file and evaluates the estimation accuracy of the proposed in-network algorithm.

As shown in the UML component diagram of Figure 4.7, the Matlab component takes as input information about: a) the deployed sensor nodes (location, prior model parameters, etc.), and b) the propagating hazard's front properties, and determines the sequence in which the deployed sensor nodes detect the evolving hazard. After that, it generates a file (*Detection Events Sequence*) which contains for each sensor node the following infor-

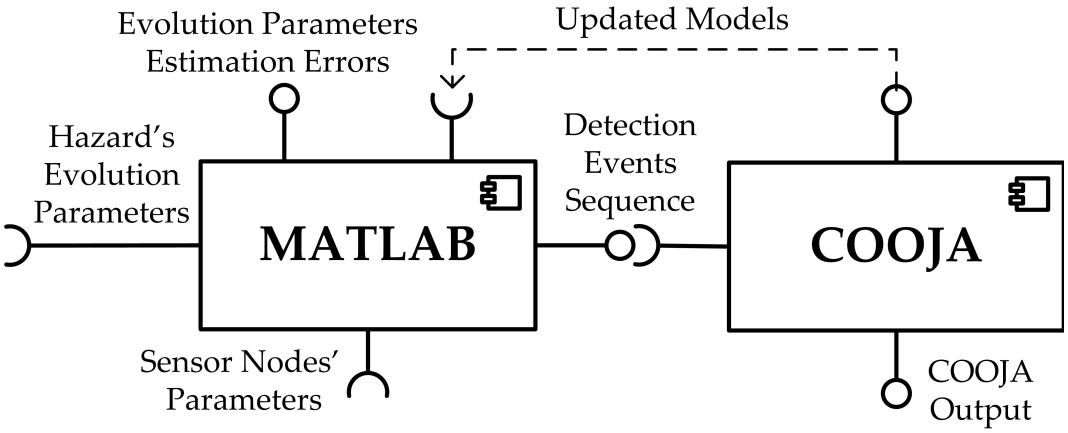


Figure 4.7: UML component diagram of Matlab-COOJA based simulation workflow.

mation: {*ID, location, time of detection, prior model parameters*}. This file is passed as input to COOJA used to simulate the behavior of the proposed distributed algorithm, as if it was implemented by a WSN consisting of Atmel's AVR RAVEN nodes [85]. To achieve this, the code every sensor node needs to run to implement the proposed in-network algorithm was programmed in C on the Real Time Operating System (RTOS) Contiki. Using COOJA we simulate the IEEE 802.15.4 MAC protocol's byte stream (preamble, start of frame delimiter, data, and checksum) which is also used by the Atmel's AVR Raven nodes. Moreover using COOJA's Unit Disk Graph Medium (UDGM) with a distance loss propagation model [86] (that considers interferences), we can evaluate the proposed algorithm's behavior under different Rx/Tx failure probabilities.

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

4.3.2 Experimental Setup

A notable advantage of the proposed in-network processing algorithm is that it can estimate accurately the evolution characteristics of a local front using low density sensor networks. To demonstrate this feature in all the conducted experiments we have used

WSN densities that are considered low for environmental monitoring applications. Specifically the used densities $7.5 \times 10^{-5}, 10^{-4}, 1.25 \times 10^{-4}\ sensors/m^2$, which correspond to 75, 100 and 125 sensor nodes respectively deployed within an $1km^2$ square area. For each WSN density value we use a large number of randomly drawn sensor node deployments and demonstrate how the proposed algorithm performs under different sensor node as well as communication (Rx and Tx) failure probabilities (equal to 0, 0.1, 0.2 and 0.3).

For all experiments the radius of the communication range of a node was set to $r = 150m$, such as to guarantee that we have a connected network (every node has at least one neighbor) for every density scenario. Furthermore, in order to evaluate how the size of the sensing radius R_d affects the accuracy of the proposed algorithm we repeated the experiments with different R_d values equal to 0.1m and 15m. For the sensing models the parameters were calculated using the equations in (4.1) for $\alpha = 1$. Each sensor node is initialized with the same prior model $m_i = \{\phi_i = 0, \delta_i = 0, u_i = 5m/min, s_i = 2m/min\}$. The mean speed value in the prior model was intentionally chosen to differ significantly from the simulated hazard front speeds in order to demonstrate the ability of the proposed distributed algorithm to estimate the true model parameter values even when the initial prior belief model of the sensor nodes deviates significantly from the reality. The communication energy consumed by the simulated AVR Raven nodes, was measured using: a) their maximum power (3dBm) for transmission (at this power level communication range of the AVR RAVEN nodes is approximately 150m) and b) reception sensitivity -101dBm which is fixed for the AVR Raven nodes. Finally, in all the conducted experiments we used a policy where a sensor node retransmits once its message if it does not receive an acknowledgement from the message recipient(s).

4.4 Results and Discussion

In the conducted experiments the diffusive phenomenon (continuous object) was simulated using either a Matlab program or FLogA a wildfires behavior simulator developed in our group [87]. To evaluate the accuracy of the proposed distributed algorithm, we compared the estimated direction and speed of the local fronts to the corresponding ground truth values. A detailed description of the evaluation metrics used is provided in Appendix

B.

4.4.1 Experiment 1: Multi-source diffusive hazards

In the first experiment a complex diffusive phenomenon is modeled as two circles of fixed centers and radii that are increasing with equal but time varying rates. The circles represent two distinct diffusive hazards which have just started entering the WSN deployment area at the beginning of the simulation. The two circles start to overlap as they grow to form a complex front line before covering half of the deployment area. A detailed presentation of the experimental setup is presented in Appendix C. Moreover, in order to help the reader visualize the complex phenomenon and get a sense of the model updates taking place during its propagation, we provide a video animation (see file Experiment1TwoFronts.mp4 [88]) created using Matlab. A discussion of what is shown in the video can also be found in Appendix C.

Modeling propagating hazards with circular shapes is justified because: a) Fick's second law of diffusion (which in two or more dimensions is analogous to the heat equation) 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 [89]. b) The circle's geometrical properties allow us to evaluate analytically the speed and direction estimation errors (see Section Appendix B).

As observed from Table 4.1, for each sensing radius case the parameters estimation accuracy of the proposed algorithm seems to be insensitive to changes in sensor nodes density, node failure probability, and Rx/Tx failure probability. This was also confirmed by comparing pairwise the means of the error densities using Student's t-test. For all cases the difference of the means was found to be insignificant at the 0.05 significance level. Moreover, the results indicate that the accuracy of the proposed algorithm slightly decreases when the sensing radius R_d increases. This can be explained if we consider that an increase of the sensing radius implies increasing the uncertainty associated with the front line's location at the time of the hazard's detection. This in turn implies increasing the uncertainty regarding the estimated mean speed values u_{ih}, $h \in \{j, k\}$ used to estimate the local front's orientation and speed (see Section 3.3 for details).

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

Table 4.1: Experiment 1 results summary: The Median/Inter Quartile Range of the orientation error (in degrees) and percent speed error under different nodes density, probability of node failure, and sensing radii conditions. For each entry the reported statistics were computed based on 200 simulation runs (50 WSN random deployments x the 4 Rx/Tx failure probability cases considered).

In Figure 4.8(a) we see that the total number of model updates is reduced as the nodes failure and Rx/Tx failure probabilities increase. A higher node failure probability implies a reduction of the operational sensor nodes participating in the distributed algorithm (in the fixed deployment area) i.e. a reduction of the effective network's density. This in turn implies fewer neighbors within a sensor's communication range, thus increasing the difficulty for a Master Candidate to satisfy the necessary conditions to become a Master (see Section 4.2.1 - Forming a Local Cluster). Moreover, increasing the Rx/Tx communication failure probability implies higher difficulty for the sensor nodes to collaborate with their neighbors in order to update the parameters of a local front model. Another interesting observation in Figure 4.8(a) is that the number of updates is affected more by increasing the node failure probability rather than the Rx/Tx failure probability. This can be explained if we consider that: a) in contrast to node failures (considered as permanent) the Rx/Tx failures do not necessarily imply a reduction of the effective network's density, since nodes which may fail to receive or transmit some of the messages remain functional. b) The single message retransmission policy used in this experiment increases the probability for successful communications between the nodes, which in turn increases the probability for a local model update to eventually succeed.

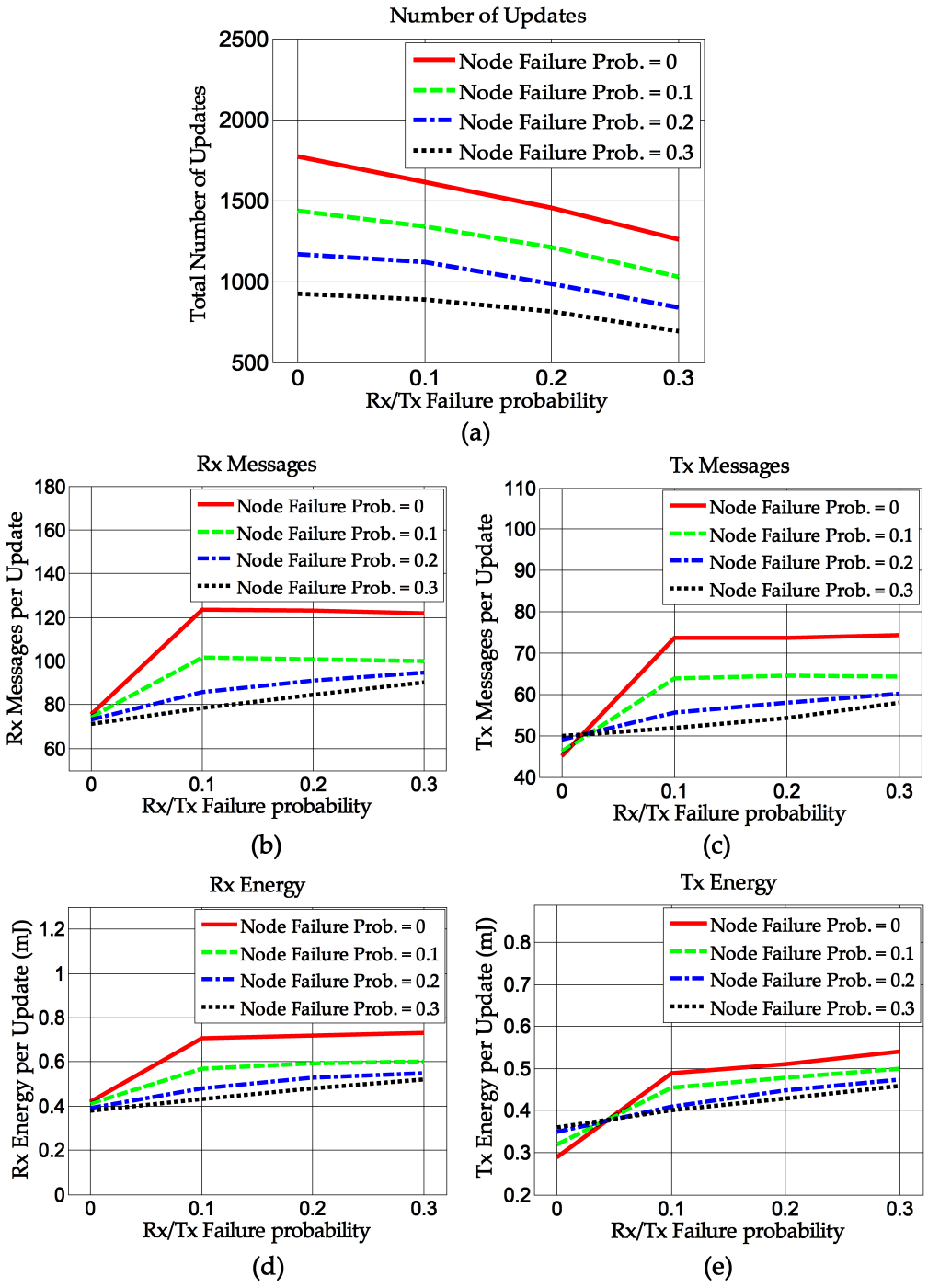


Figure 4.8: Experiment 1: Total number of model updates, mean number of messages and Rx/Tx energy consumed per model update, for each node failure and Rx/Tx failure probability considered (density = 100 sensor nodes per km^2).

Figures 4.8(b) and 4.8(c) show (for the 100 sensor nodes per $1km^2$ density case) the mean number of Rx and Tx messages exchanged in the network per model update. From the presented line plots when the Rx/Tx failure probability increases in the range [0, 0.1] we observe a significant increase of messages per model update for each node failure probability curve. This can be explained if we consider that Rx/Tx failures trigger message retransmission which increases the number of messages exchanged over the network. Another interesting observation is that this trend becomes less profound as the nodes failure probability increases. This behavior can be explained if we consider that: a) increasing node failures imply an effective network density reduction and therefore a reduction of the mean number of neighbors within a node's communication range and thus a reduction of the mean number of the Rx and Tx messages exchanged in the neighborhoods. b) When the node failure probability is non-zero, retransmission is triggered even when the Rx/Tx failures probability is zero, since the failing nodes are not able to send the required acknowledgments. These triggered retransmissions increase in turn the mean number of the Rx and Tx message exchanged over the network. Thus, the already increased number of messages for the non-zero node failure probability cases explains why we observe a smoother increment of the mean number of messages when the Rx/Tx failure probability increases in the range [0, 0.1].

Figures 4.8(b) and 4.8(c) also suggest that for zero Rx/Tx failure probability the mean number of Rx and Tx messages per model update for all the non-zero nodes failure probability cases is almost equal (Rx) to or larger (Tx) than the corresponding Rx and Tx mean number of messages of the zero node failure probability case. At first glance this behavior may seem counter-intuitive since for the non-zero nodes failure probability cases the effective density of the network is reduced and therefore we would expect the mean number of messages exchanged per model update to be smaller. However, this is not the case since the potential retransmission triggered if node failure probability is non-zero increases the number of messages exchanged over the network.

Moreover, in Figures 4.8(b) and 4.8(c) we also observe that as the Rx/Tx failure probability increases in the range [0.1, 0.3] the mean number of the Rx and Tx messages per model update remains almost unchanged for node failure probabilities 0 and 0.1 and increases only slightly for larger node failure probabilities with values 0.2 and 0.3. To explain

this behavior we have to consider the following four mechanisms which affect the number of messages exchanged per model update: a) The increase of node and Rx/Tx failure probabilities *increases* the mean number of messages exchanged over the network due to the triggered retransmissions. b) The increase of the node and Rx/Tx failure probabilities increases the probability of ``zombie'' clusters formation (see Section 4.2.1 - Master Declaration), i.e. clusters in which the sensor node malfunctions and Rx/Tx failures render the Master node incapable to update its model parameters. The messages exchanged (wasted) within ``zombie'' clusters combined with the smaller number of model updates *increase* the mean number of messages required per model update. c) On the other hand, the increase of the nodes failure probability *reduces* the effective network's density and therefore the mean number of the Rx and Tx messages exchanged over the network. d) Finally, the increase of the Rx/Tx failure probability reduces the probability for a sensor node to receive or transmit successfully a message, which in turns *reduces* the total number of Rx and Tx messages. The line plots in Figures 4.8(b) and 4.8(c) suggest that for node failure probability 0 and 0.1 the increse of the mean number of messages, caused due to mechanisms (a) and (b) is counterbalanced by the message traffic reduction mechanisms (c) and (d) and therefore no significant changes are observed as the Rx/Tx failure probability increases in the range [0.1,0.3]. However, for higher node failure probabilities, i.e. 0.2 and 0.3, the more frequent formation of ``zombie'' clusters combined with the more frequent triggering of retransmissions results to a small increase of the mean number of messages per model update in the same Rx/Tx probability of failure range.

Figures 4.8(d) and 4.8(e) show the mean energy consumed for Rx and Tx communications per model update. As expected, due to the direct relation between the Rx/Tx messages (Bytes) and Rx/Tx communication energy, the energy and messages per model update line plots follow similar trends. However, as the Rx/Tx communication failure probability increases we observe a small increase of the gradient of the energy line plots as compared to the corresponding line plots for the messages. This behavior can be explained if we consider that an increase of the Rx/Tx failure probability makes it more difficult for a Master node to find a qualified new Master and eventually forces it to broadcast a message to its Slaves so that they propagate its updated model to their neighbors (see Section 4.2.3 - Model Propagation). The message broadcasted by the Slave nodes

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

Table 4.2: The average percent change (increase(+), decrease(-))/stdvs of the a) total number of model updates, b) mean number of Rx and Tx messages exchanged per update, and c) mean Rx and Tx energy consumed per update, for the 75 and 125 nodes (per km^2) density scenarios relatively to the 100 nodes density scenario.

($PPM(UM_i)$) carries the information of the Master's updated model and therefore requires the transmission of many bytes which increases the mean Rx and Tx energy consumption.

Figures 4.9 and 4.10 show, for the 75 and 125 sensor nodes per km^2 density scenarios respectively, the line plots for the: (i) total number of updates, (ii) mean number of Rx and Tx messages per model update, and (iii) the mean energy consumed for Rx and Tx per model update for all the considered nodes and Rx/Tx failure probability cases. The line plots follow similar trends with these of the 100 sensor nodes density scenario and are therefore subject to similar interpretations.

In Table 4.2 we summarize the differences (average percent change) relatively to the 100 nodes per km^2 density scenarios. The provided statistics were computed by considering as sample points all local front model updates from all node density ($\{75, 100, 125\}$) and failure probability cases ($\{0, 0.1, 0.2, 0.3\}$). We observe that as the network density increases (decreases) the total number of model updates also increases (decreases). This is as expected since a higher (lower) nodes density implies more (fewer) neighbors within a sensor's communication range, making it more easy (difficult) for a Master Candidate node to satisfy the necessary conditions to become a Master (see Section 4.2.1 - Forming a Local Cluster). Finally, the increased (decreased) number of neighbors also explains why the mean number of Rx and Tx messages exchanged and energy consumed per model update increases (decreases) with the increase (decrease) of the network density.

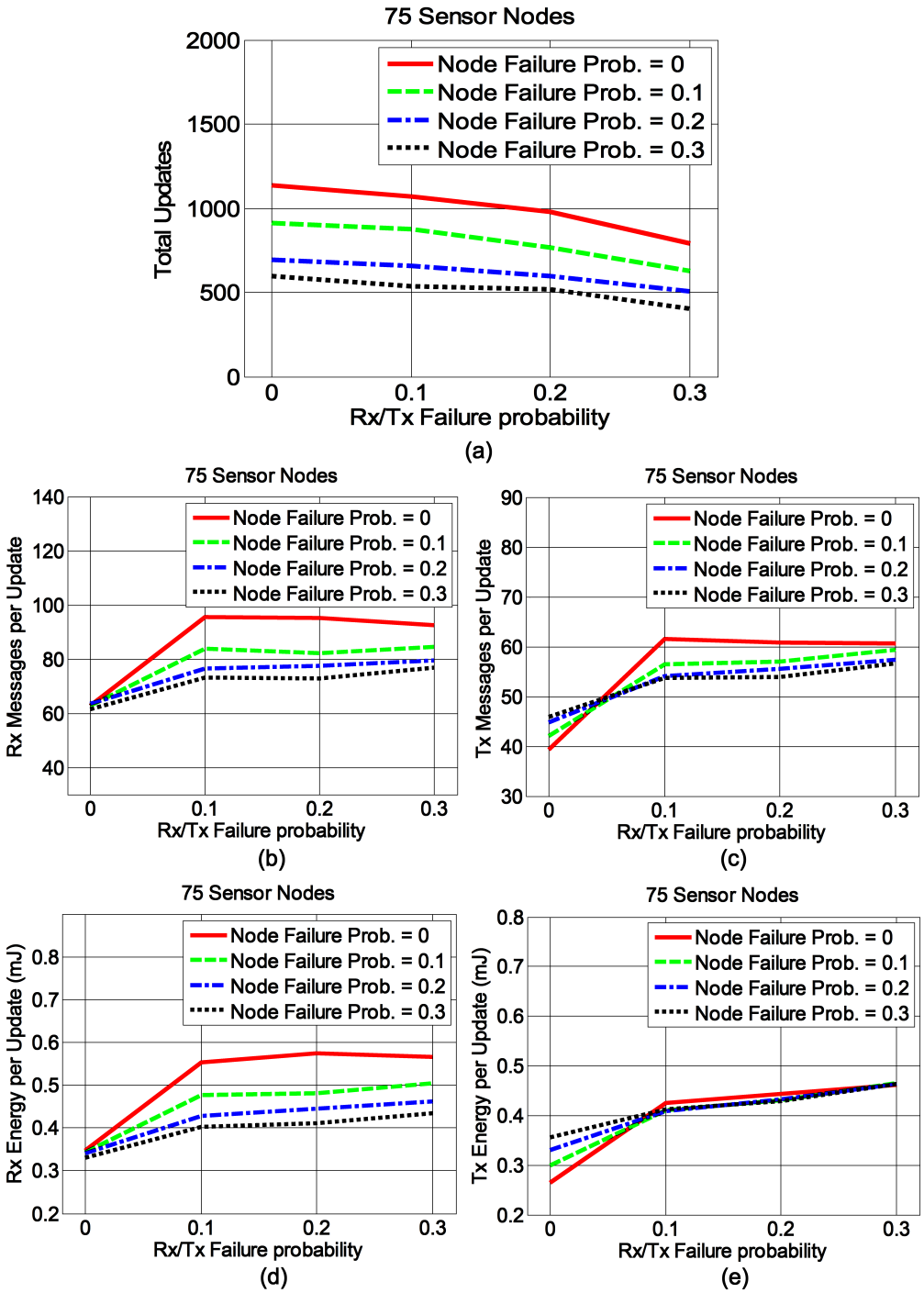


Figure 4.9: Experiment 1: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 75 sensor nodes per km^2).

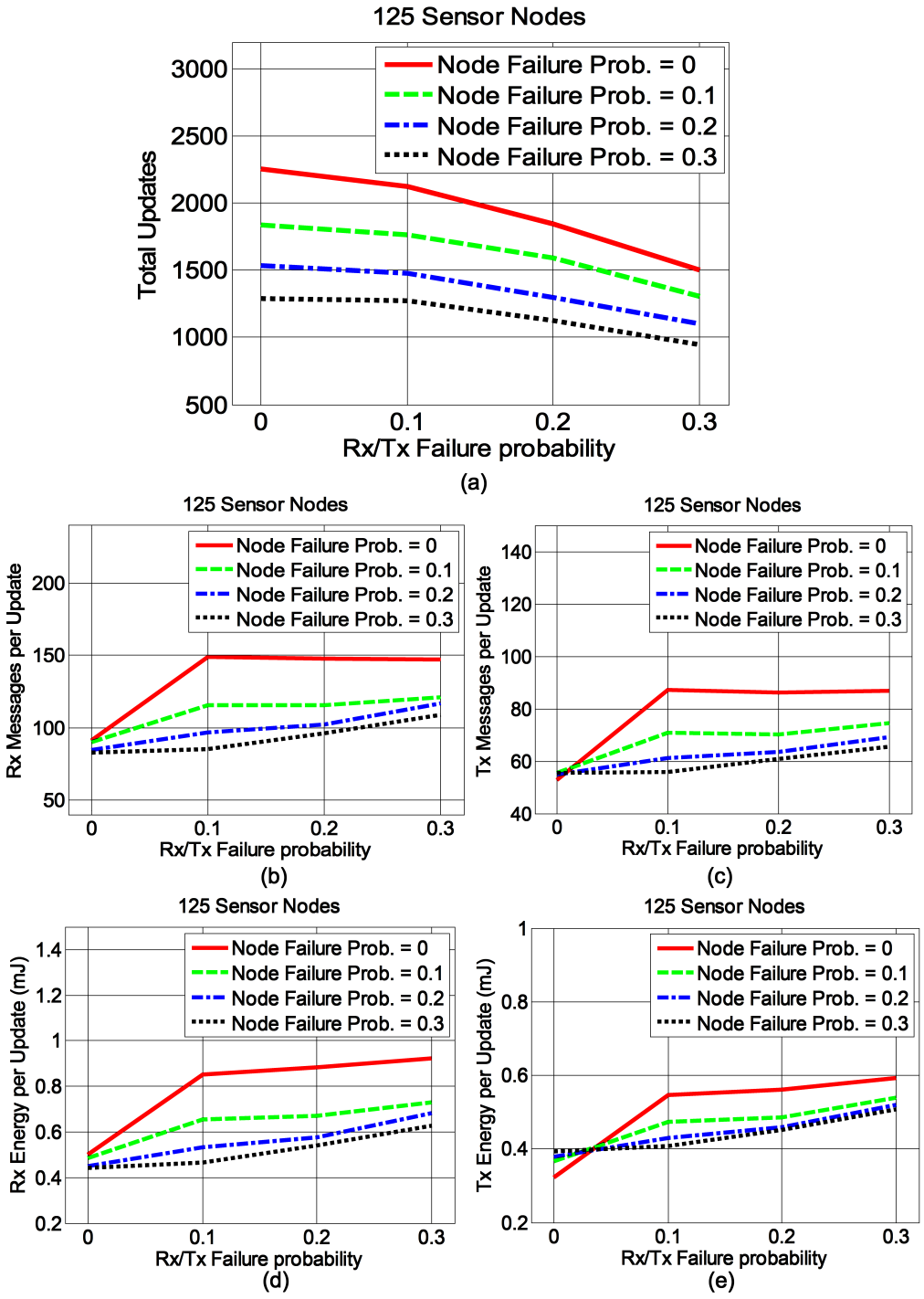


Figure 4.10: Experiment 1: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 125 sensor nodes per km^2).

Evaluation under Sensing Models Mismatch:

Below we provide additional results that demonstrate the efficiency and robustness of the proposed in-network algorithm in estimating with accuracy the evolution parameters of an evolving front line under different sensing model mismatch assumptions. As a sensing model mismatch we consider the situation where the ``actual'' sensing distance is drawn from a different distribution than the one assumed by the sensor nodes.

In all simulation scenarios of Experiment 1 we have assumed that the ``actual'' sensing behavior matches the one assumed by the sensor nodes i.e. a Gaussian model with parameters μ_d and σ_d calculated using the equations in (4.1) below with $R_d = 15m$ and $a = 1$ (see Section 3.2),

$$\mu_d = \frac{\alpha R_d}{2}, \quad \sigma_d = \frac{R_d}{3}(1 - \frac{\alpha}{2}). \tag{4.1}$$

For the evaluation of the proposed algorithm under sensing model mismatch conditions, we repeated the simulation scenarios of Experiment 1, but this time the sensing distance was drawn from either: (i) A Gaussian distribution with parameters μ_d and σ_d derived using the equations in (4.1) when $R_d = 15m$ and $a = 0$. By setting the parameter $a = 0$ we essentially relax the assumption that the node may malfunction as the hazard approaches (see Chapter 3). (ii) An exponential distribution (see equation (4.2)) below with parameters $\{R_d = 15m, R_s = 0m, \lambda = 0.35\}$.

$$p(x) = \begin{cases} \lambda e^{-\lambda(x - R_s)} & x \geq R_s \\ 0 & x < R_s \end{cases} \tag{4.2}$$

The plots of the three different models used to draw the detection distance are shown in Figure 4.11. All models assume that the probability for a sensor node to detect the evolving front line at distance larger than $R_d = 15m$ is negligible. Moreover, a sensor node is assumed to be located at the origin.

Table 4.3 provides for each sensor node density value (75, 100, 125), node failure probability and Rx/Tx failure probability scenario (0, 0.1, 0.2, 0.3) considered, the param-

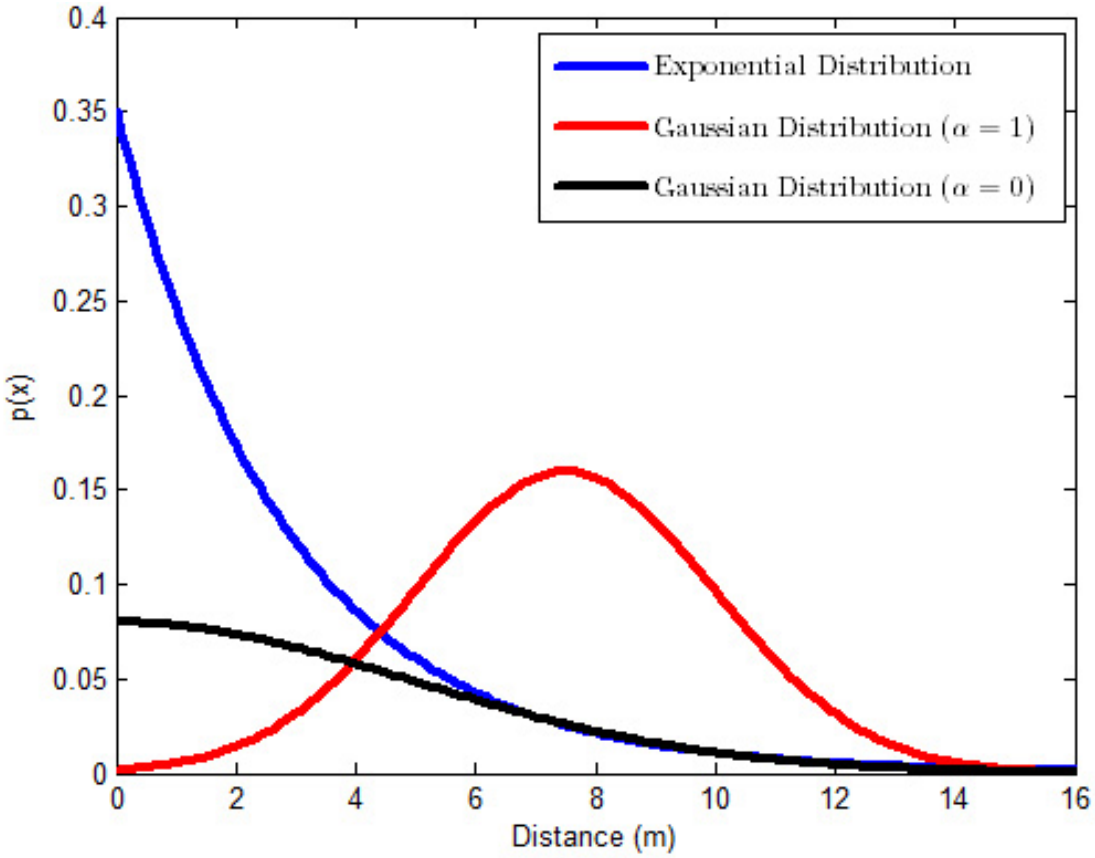


Figure 4.11: Scenario 1: The red curve corresponds to the shifted Gaussian sensing model assumed by the sensor nodes. The black (blue) curve corresponds to the Gaussian (exponential) monotonic sensing model used to draw the detection distance under the model mismatch conditions considered (see text for details).

Experiment 1: Parameter estimates under Sensing Model Match and Mismatch Conditions						
$P(f)$	75 nodes		100 nodes		125 nodes	
	Orient.	Speed%	Orient.	Speed%	Orient.	Speed%
Sensing Model Match						
0	5.21/11.12	13.92/14.25	4.67/10.3	13.54/13.66	4.59/10.25	13.59/13.97
0.1	4.99/10.83	14.07/14.88	4.54/10.88	14.81/13.31	4.63/10.16	13.82/14.24
0.2	5.07/10.74	13.96/14.91	5.19/10.46	13.27/14.82	5.11/10.72	13.64/14.4
0.3	4.86/10.89	13.71/14.7	5.03/10.97	14.35/14.69	4.97/10.58	14.01/14.82
Gaussian Sensing Model ($a = 0$) Mismatch						
0	5.1/10.28	14.12/13.4	4.85/10.07	13.72/12.97	4.69/9.72	13.22/13.29
0.1	5.03/10.33	14.31/13.69	4.68/10.24	14.22/13.19	4.72/9.79	13.94/13.86
0.2	5.21/10.25	14.37/13.92	5.24/10.32	14.39/13.39	4.99/10.06	14.16/13.73
0.3	5.28/10.42	14.26/13.81	5.21/10.36	14.47/13.66	5.08/10.03	14.37/13.95
Exponential Sensing Model Mismatch						
0	5.18/10.01	14.28/13.07	4.96/9.76	14.06/12.82	4.83/9.31	13.97/12.97
0.1	4.97/9.93	14.6/13.52	4.82/10.06	14.34/13.03	4.77/9.68	14.36/13.22
0.2	5.29/10.2	14.44/13.7	5.27/10.09	14.51/13.28	5.05/9.59	14.21/13.19
0.3	5.36/10.14	14.71/13.59	5.22/9.98	14.4/13.15	5.14/9.85	14.53/13.61

Table 4.3: Summary of Estimation Results under Sensing Model Match and Mismatch Conditions: The Median/Inter Quartile Range of the orientation (in degrees) and percent speed errors of the proposed algorithm for different nodes density, probability of node failure and sensing models assumptions. For each table entry the statistics were computed based on 200 simulation runs (50 WSN random deployments x the 4 Rx/Tx failure probability cases considered). The sensing model match case and two sensing model mismatch cases have been considered.

eter estimation error statistics of the proposed algorithm under sensing model match and mismatch conditions. We observe that the median orientation error (in degrees) and the median percent speed error increase only slightly under sensing model mismatch conditions. Specifically, the maximum increase of the median orientation and median percent speed errors are: a) for the Gaussian mismatch case: 0.42^o and 1.12% and, b) for the exponential mismatch case: 0.5^o and 1.24% respectively.

The slightly larger median errors observed for the mismatch cases are justified if we consider that the Gaussian centered at the origin and the exponential models generate with high probability detection events closer to the sensors' locations (the origin in Figure 4.11) when the sensor nodes assume for the calculation of the orientation and speed parameters (see equations (3.5) and (3.6) in Section 3.3) that detection has a higher probability to occur at the distance indicated by the mean of their shifted Gaussian sensing model ($\frac{R_d}{2} = 7.5m$ in our case see red curve in Figure 4.11). However, despite the fact

that the difference between the ``actual'' and the "assumed" expected detection distance is considerable for the sensing models mismatch cases, the increase of the estimation errors is small. Finally, the slightly larger estimation errors observed for the exponential mismatch case, are justified if we consider that the smaller variance of the exponential sensing model (blue curve in Figure 4.11) increases the probability of generating detection events closer to the sensors' locations (and further from the expected detection distance assumed by the sensors). Overall these results indicate that our algorithm is robust to sensing model mismatch conditions, a fact that further supports its applicability in real hazard tracking applications where sensing behavior variations are expected for different types of hazards and sensors.

4.4.2 Experiment 2: Complex diffusive hazards with irregular shapes

This experiment was designed to evaluate the ability of the proposed distributed algorithm to estimate accurately the evolution parameters of more complex and more realistic hazardous phenomena, having irregular shapes, large speed variations, etc. To generate hazards with such realistic behavior, we employed FLogA [87], a web-based interactive software tool (developed in our group) which allows us to draw a forest area anywhere in Europe using Google Earth [90], insert fire ignition points, define wind condition scenarios and then simulate and geo-animate the behavior of the evolving fire line under different conditions.

Setup and Visualization of the Wildfire Evolution Scenarios:

FLogA (Fire Logic Animator) is a web-based software tool which allows us to draw a forest area on Google Earth anywhere in Europe, insert interactively fire ignition points, simulate and animate the behavior of the evolving fire line under different conditions (see reference [87] in the paper). Using FLogA we defined a square forest area (of $1km^2$) at Hymettus mountain in Attica Greece and simulated 5 different wildfire scenarios in the same area. Forest's topographic parameters (slope, aspect, fuel model, fuel moisture) were the same for all the scenarios. However differences in: a) the prevailing wind conditions (speed and direction) and b) the number and the location(s) of the fire ignition point(s) were responsible

for the generation of very different wildfire evolution patterns. Below we describe the set up of the simulated wildfire scenarios.

Similarly to Experiment 1, for each wildfire evolution scenario we tested and evaluated the behavior of the proposed algorithm under different: a) sensor node densities (75, 100, 125 nodes per $1km^2$), b) sensor node arrangements (10 random deployments per wildfire scenario), c) sensor node failure and Rx/Tx failure probabilities with values $\{0, 0.1, 0.2, 0.3\}$, and d) small and large sensing range radii, $\{0.1m, 15m\}$. The detailed setup of the wildfire simulation scenarios is provided below.

Using FLogA we defined a square forest area (of $1km^2$) at Hymettus mountain in Attica Greece and simulated 5 different wildfire scenarios in the same area. Figures 4.12 - 4.16 show for each one of the five wildfire evolution scenarios simulated with FLogA, four snapshots that help us visualize on Google Earth the ignition point(s) as well as the corresponding part(s) of the forest area that has (have) been affected by the fire (area covered with red color) 45, 90 and 135 minutes after the ignition respectively. The cyan ``stars'' correspond to randomly deployed sensor nodes within the square forest area of $1km^2$.

Scenario 1: In this scenario the fire was initiated from a single ignition point source which was placed at the bottom right corner of the considered $1km^2$ squared forest area. The wind speed and direction parameters were fixed within the forest area and their values were set to $2m/s$ (light breeze) and 135^o (with respect to the x axis) respectively (see Figure 4.12).

Scenario 2: Similar to scenario 1 and in this case the fire was initiated from a single ignition point source which was placed outside the bottom right corner of the considered $1km^2$ squared forest area. The wind speed and direction parameters were fixed within the forest area and their values were set to $9m/s$ (fresh breeze) and 135^o (with respect to the x axis) respectively (see Figure 4.13).

Scenario 3: In this scenario the fire was initiated using two ignition points which were placed near to the top left corner of the forest's square area. The wind speed and direction parameters were fixed within the forest area and their values were set to $2m/s$ (light breeze) and -45^o (with respect to the x axis) respectively (see Figure 4.14).

Scenario 4: In this scenario the fire was initiated using two ignition points which were placed outside the left side of the considered $1km^2$ squared forest area. The wind speed and direction parameters were fixed within the forest area and their values were set to $9m/s$ (fresh breeze) and 0^o (with respect to the x axis) respectively (see Figure 4.15).

Scenario 5: In this scenario we uses as ignition source a burn stripe (multiple ignition points that form a line) located at the top left corner of the considered $1km^2$ square forest area. The wind speed and direction parameters were fixed within the forest area and their values were set to $5m/s$ (fresh breeze) and -45^o (with respect to the x axis) respectively (see Figure (see Figure 4.16).

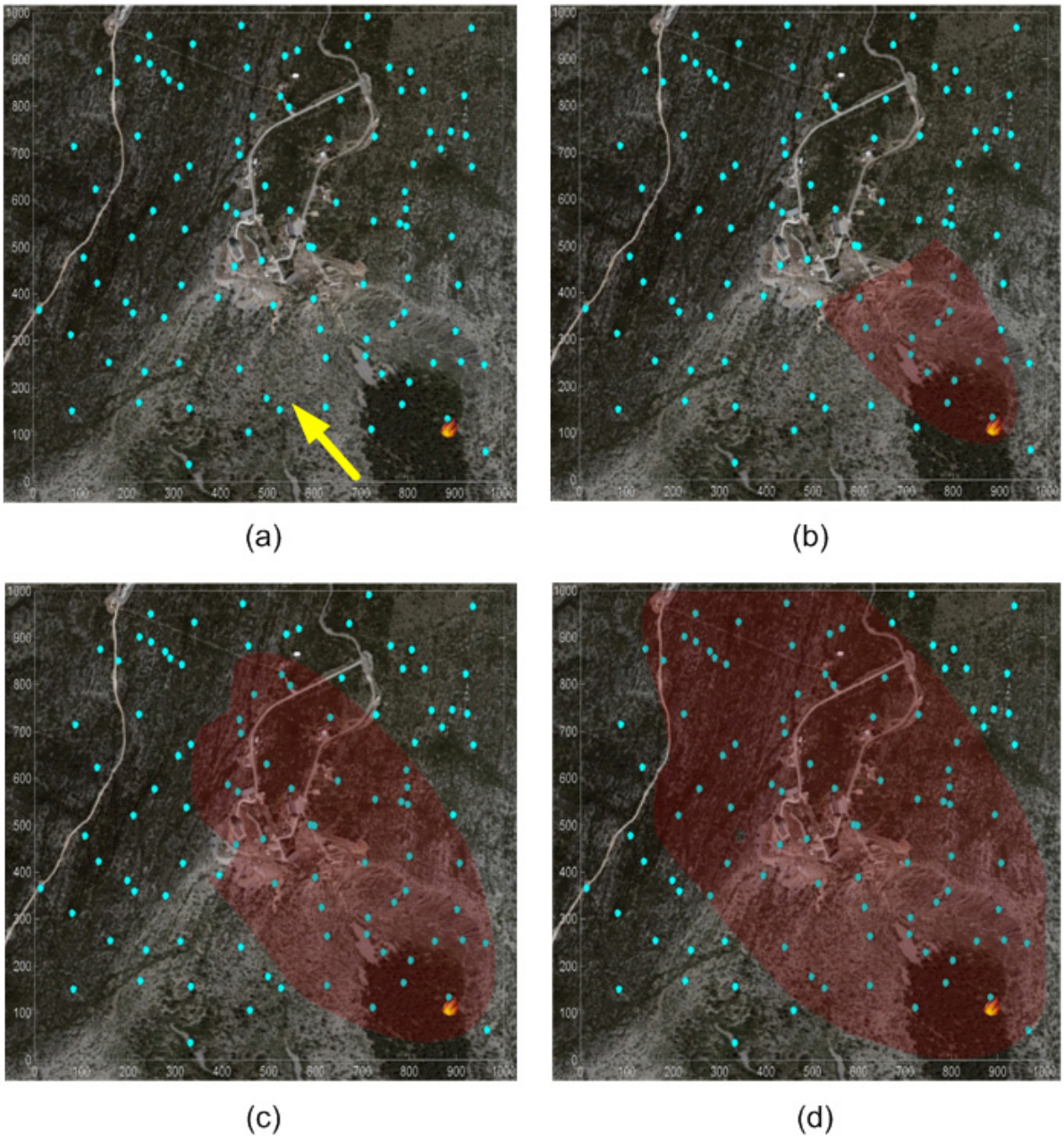


Figure 4.12: Wildfire Scenario 1: Snapshot (a) shows the wildfire's ignition point, located at the bottom right corner and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. The yellow arrow indicates prevailing wind direction. Snapshots (b), (c) and (d) show the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively.

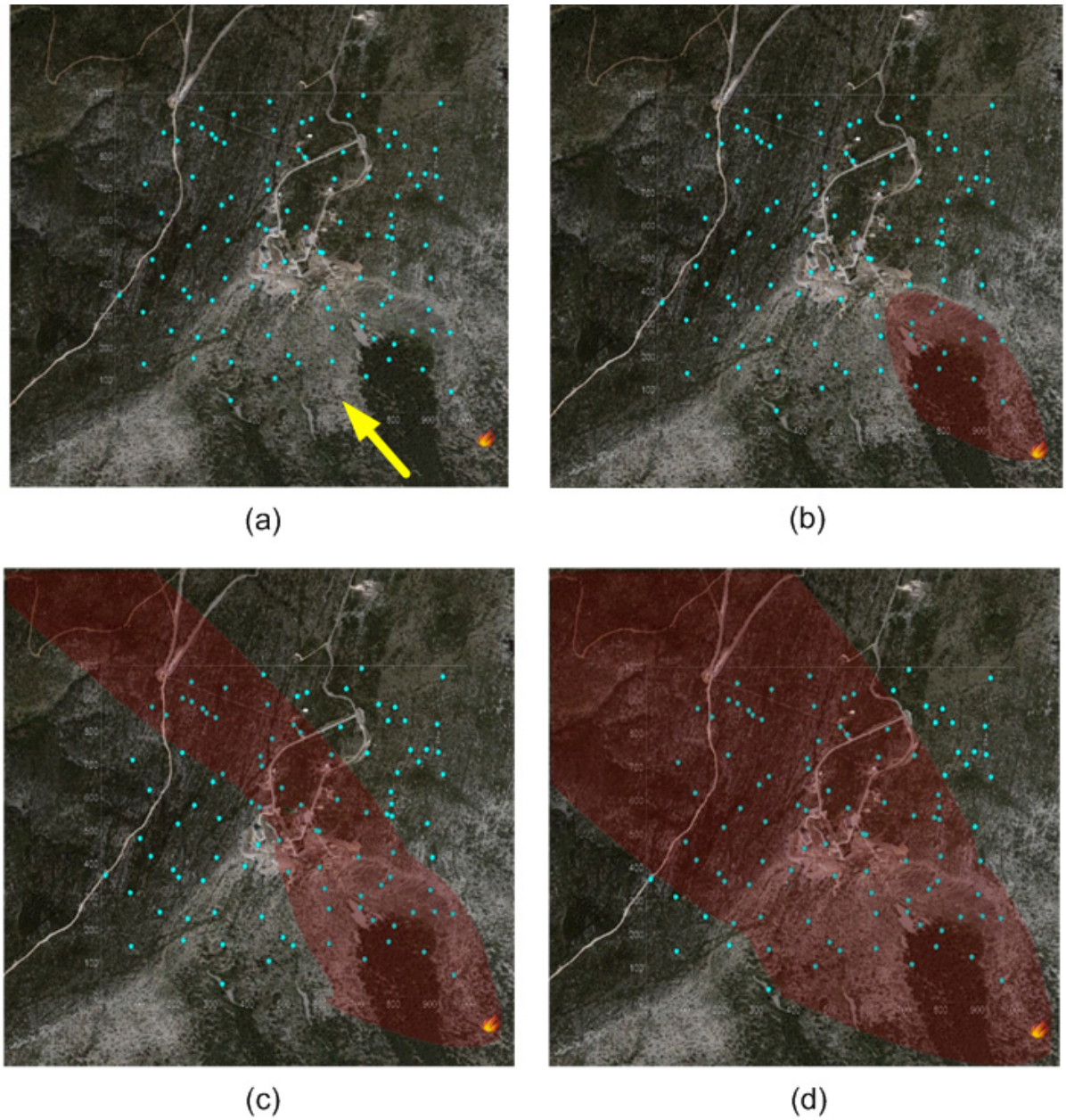


Figure 4.13: Wildfire Scenario 2: Snapshot (a) shows the wildfire's ignition point, located at the bottom right corner outside the $1km^2$ forest area and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. The yellow arrow indicates prevailing wind direction. Snapshots (b), (c) and (d) show the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively.

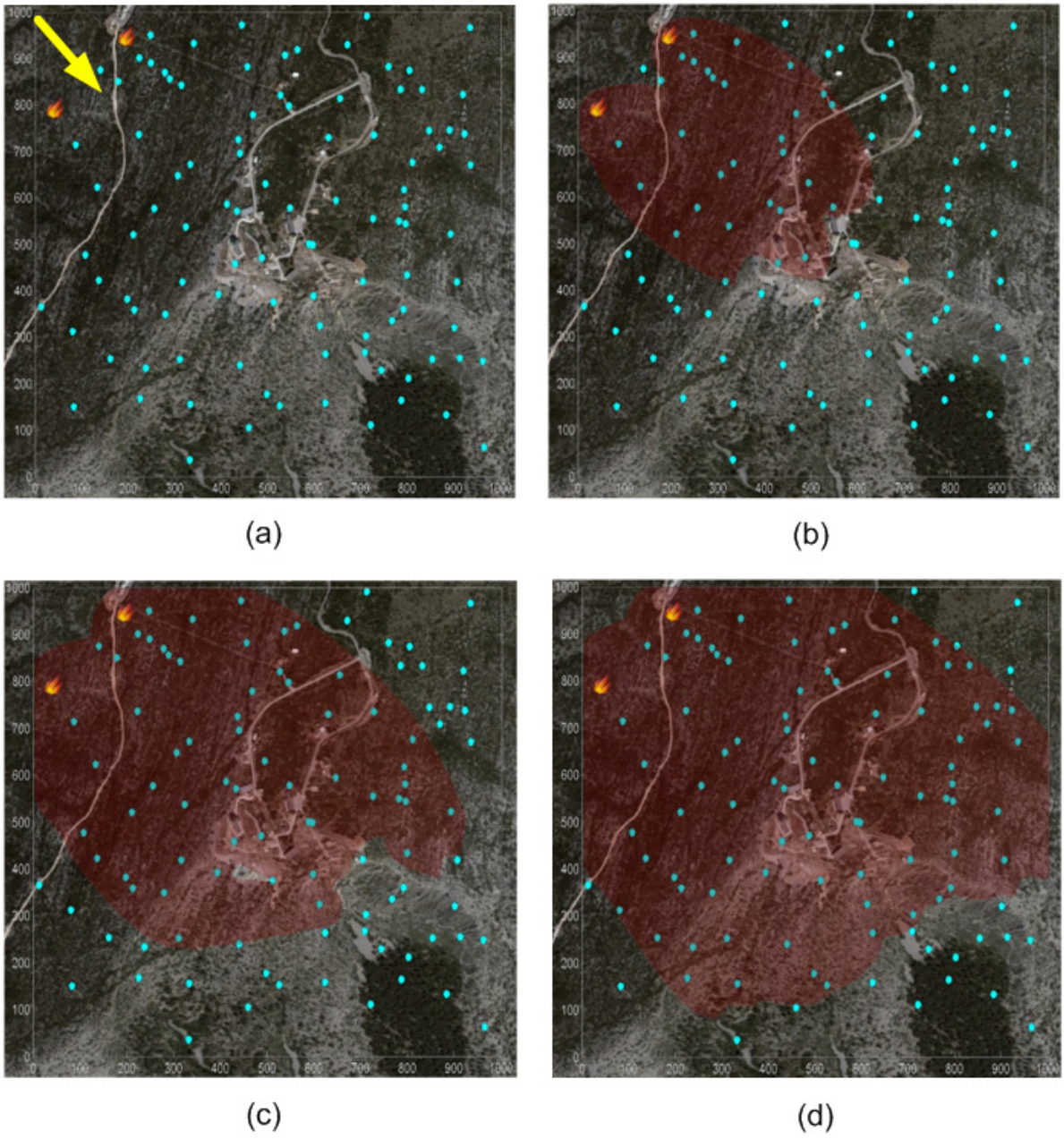


Figure 4.14: Wildfire Scenario 3: Snapshot (a) show the wildfire's ignition points, located near to the top left corner and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. The yellow arrow indicates prevailing wind direction. Snapshots (b), (c) and (d) show the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively.

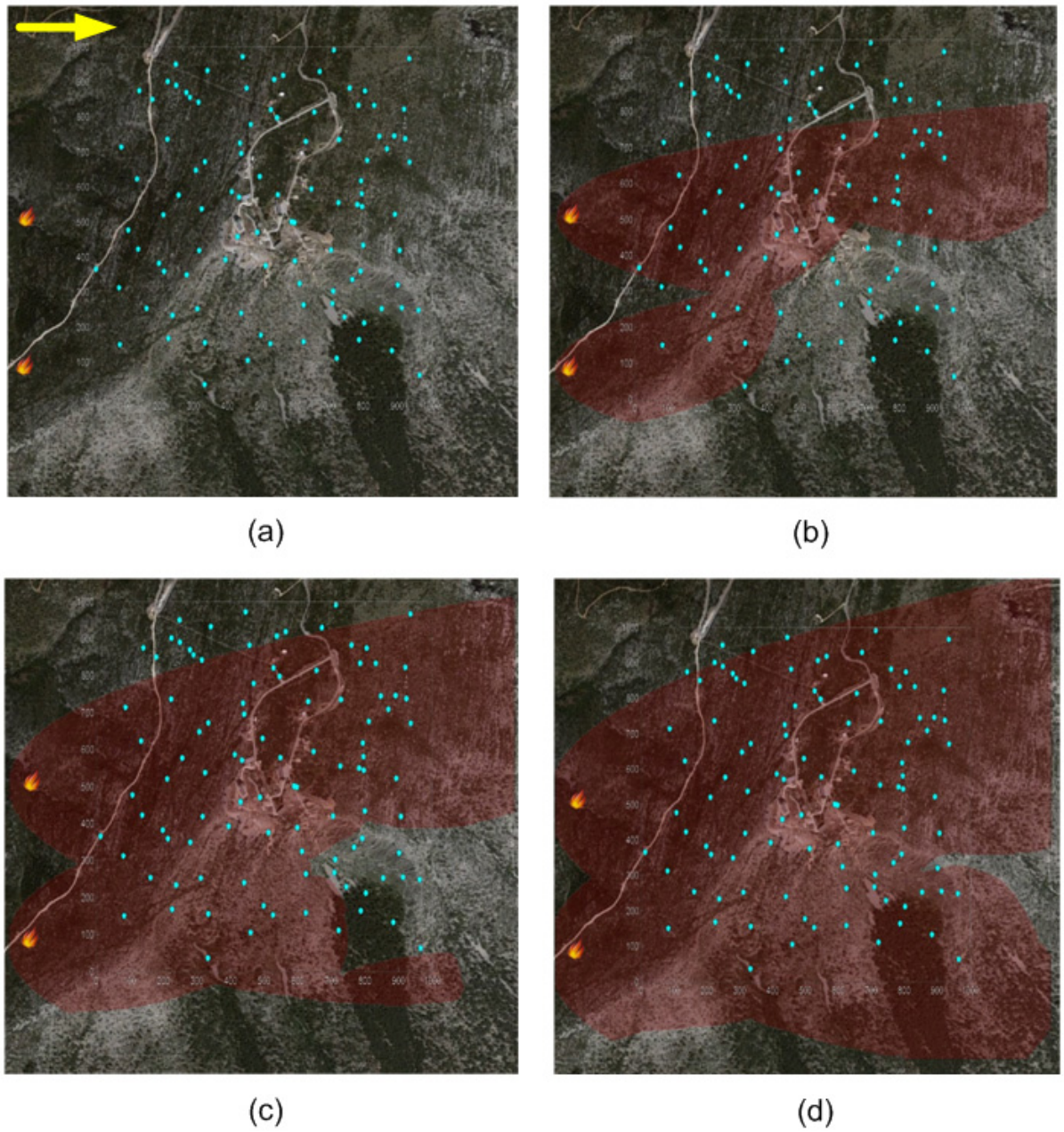


Figure 4.15: Wildfire Scenario 4: Snapshot (a) shows the wildfire's ignition points, located outside the left side of the square forest area respectively and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. The yellow arrow indicates prevailing wind direction. Snapshots (b), (c) and (d) show the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively.

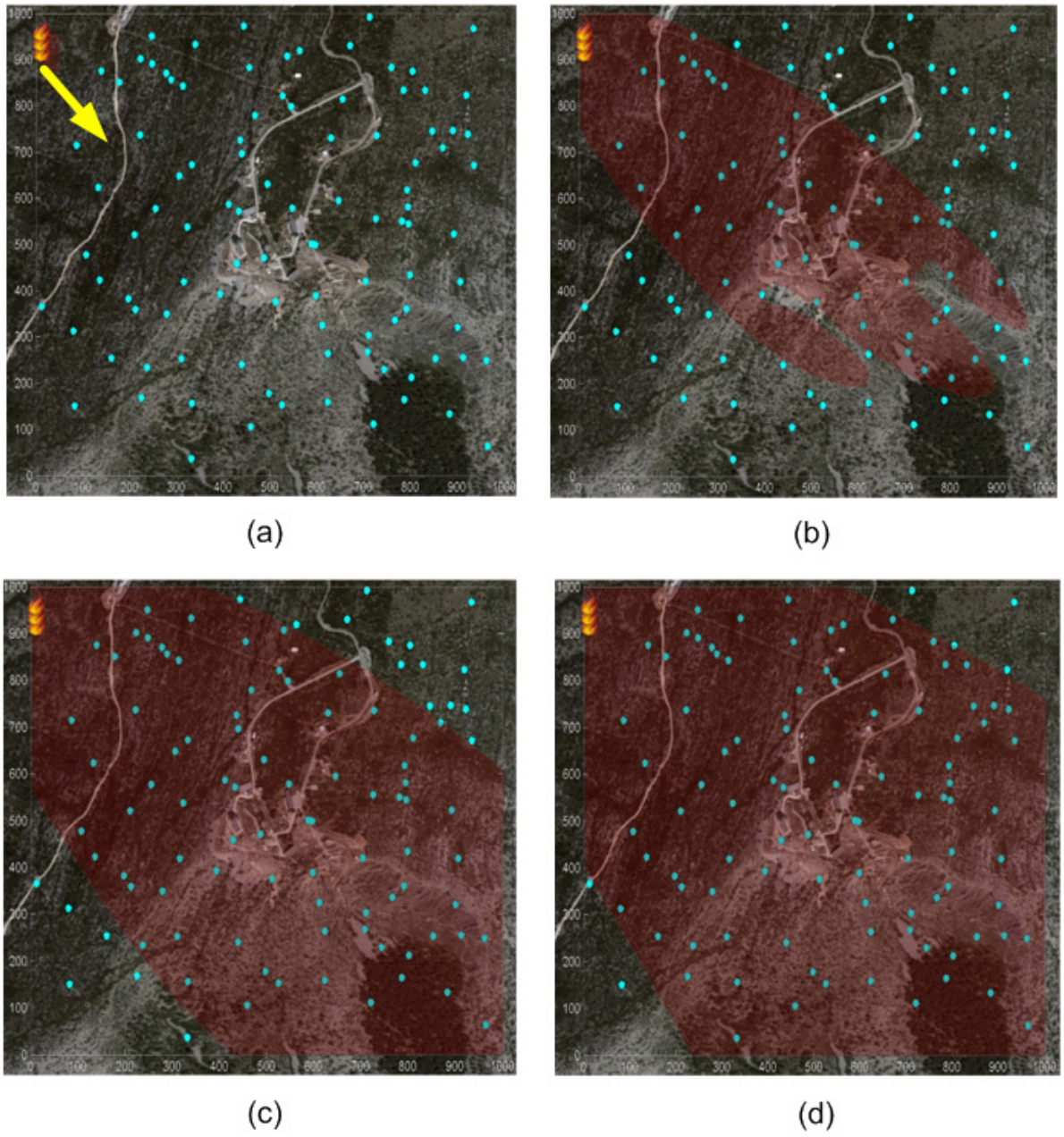


Figure 4.16: Wildfire Scenario 5: Snapshot (a) shows the wildfire's multiple ignition points (burn stripe), on the top left corner of the square forest area and a random deployment of sensor nodes (cyan stars) within the $1km^2$ square forest area. Snapshots (b), (c) and (d) shows the corresponding parts of the forest area that have been affected by the fire (area in red color) 45, 90 and 135 minutes after the ignition respectively.

Experiment 2						
Sensing Radius ($R_d = 0.1m$)						
$P(f)$	75 nodes		100 nodes		125 nodes	
	Orient.	Speed	Orient.	Speed	Orient.	Speed
0	7.37/12.31	17.01/16.1	6.44/12.09	16.92/16.22	6.13/11.92	16.51/15.94
0.1	7.22/12.49	17.22/16.54	6.82/12.31	17.17/16.36	6.21 /12.03	16.23/16.01
0.2	7.81/12.37	17.46/16.8	7.17/11.93	17.4/16.81	7.02/11.87	17.04/16.42
0.3	7.64/12.23	18.01/16.92	7.29/12.2	17.32/16.6	6.75/12.17	17.13/16.77
Sensing Radius ($R_d = 15m$)						
0	7.32/12.14	18.14/16.51	6.73/11.85	17.96/16.45	7.64/11.86	17.45/16.07
0.1	7.14/12.2	17.53/16.27	7.22/12.61	17.82/16.91	7.29/11.43	17.93/15.89
0.2	7.51/12.17	18.41/17.11	6.85/11.91	18.26/17.04	6.92/12.07	18.11/16.48
0.3	8.13/12.48	18.64/17.42	7.76/12.43	17.95/17.21	7.5/11.91	17.76/15.99

Table 4.4: Experiment 2 results summary: The Median and Inter Quartile Range of the orientation (in degrees) and speed errors under different sensor density, probability of node failure and sensing range radii assumptions. For each entry the statistics were computed based on 200 simulation runs (10 WSN random sensor node deployments for each of the 5 wildfire evolution scenarios x the 4 Rx/Tx failure probability cases considered).

Table 4.4, similarly to Table 4.1, provides the median and IQR of the orientation (in degrees) and percent speed estimation errors respectively computed over all simulated scenarios, for sensing radii cases $0.1m$ and $15m$.

As for Experiment 1, the results indicate that the parameters estimation accuracy of the proposed algorithm is robust to changes in sensor nodes density, sensor node failures and Rx/Tx communication failures. This was also confirmed by comparing pairwise the means of the error densities using Student's t-test. For all cases the difference of the means was found to be insignificant at the 0.05 significance level. Finally, the presented results indicate that the accuracy of the proposed algorithm slightly decreases when the sensing radius increases.

Figures 4.17, 4.18 and 4.19 show the line plots for all the sensor node densities (75, 100 ,125 nodes per km^2) of Experiment 2. As observed for all the experiments and for each density scenario the provided line plots follow similar trends with the corresponding line plots of Figures 4.8, 4.9 and 4.10 respectively and therefore the interpretation of the results is similar. The trends and therefore the interpretation of line plots for the total number of updates, the mean number of Rx/Tx messages and the mean energies consumed per model update remain similar with those of Experiment 1 (Figure 4.8).

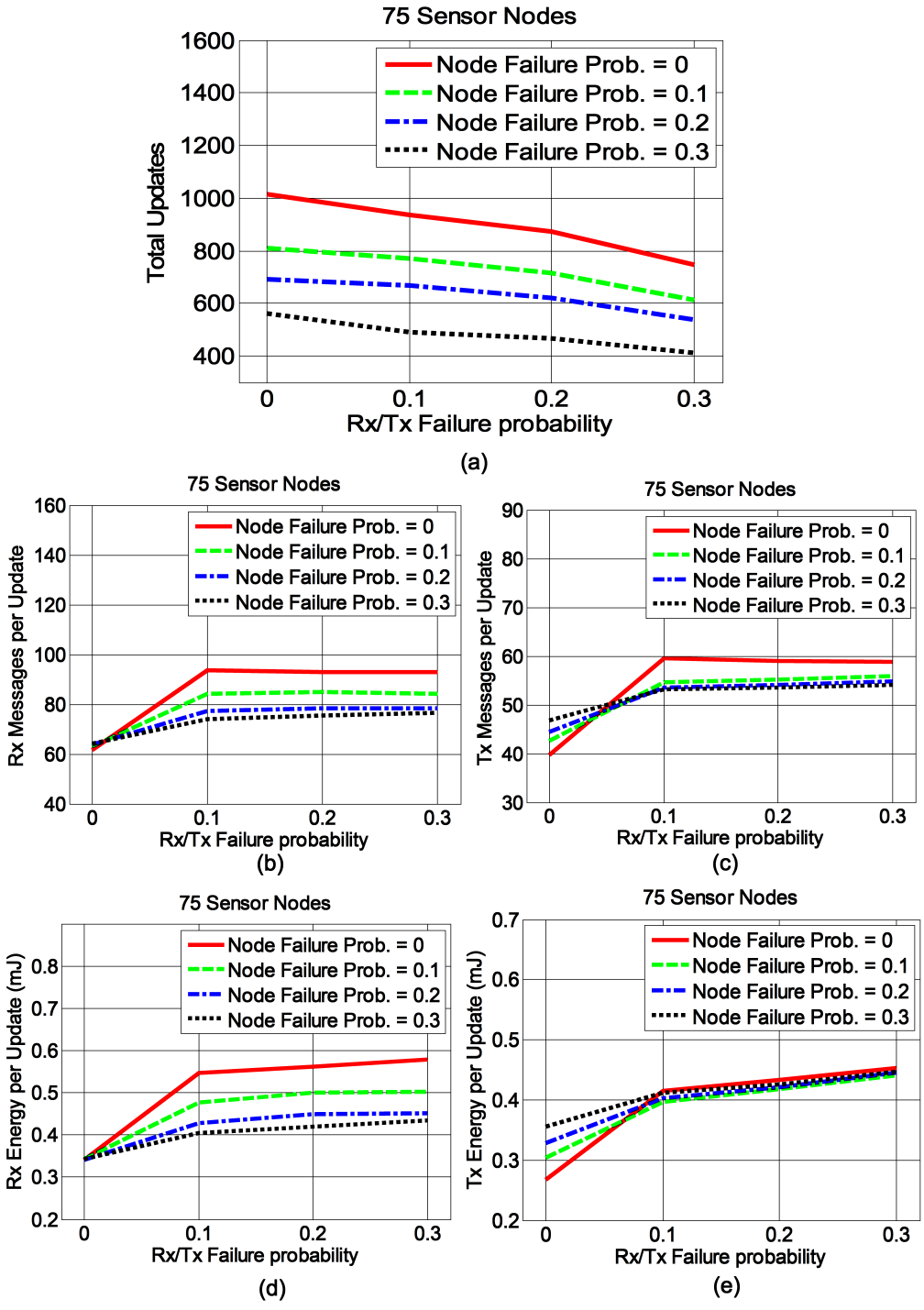


Figure 4.17: Experiment 2: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 75 sensor nodes per km^2).

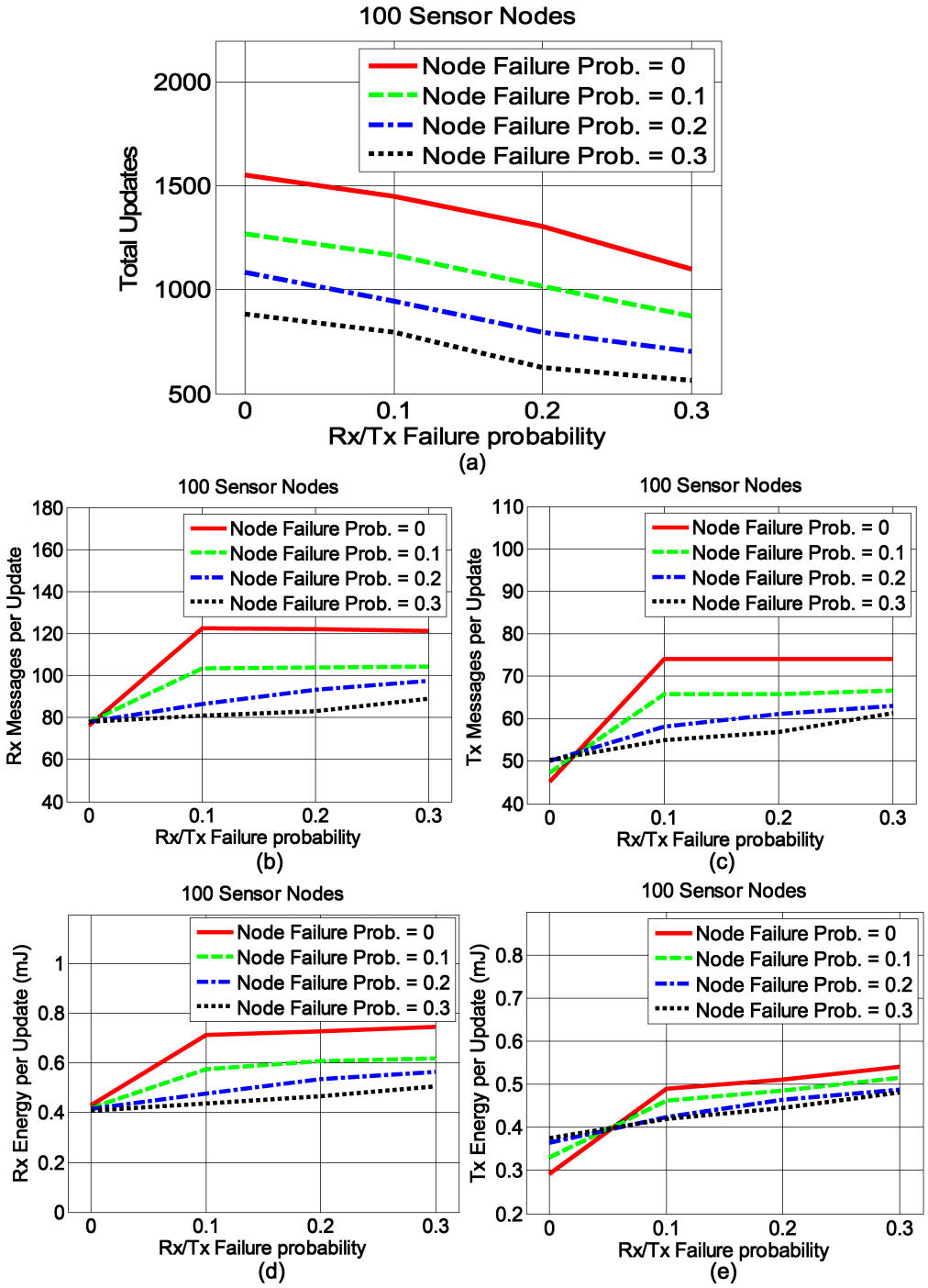


Figure 4.18: Experiment 2: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 100 sensor nodes per km^2).

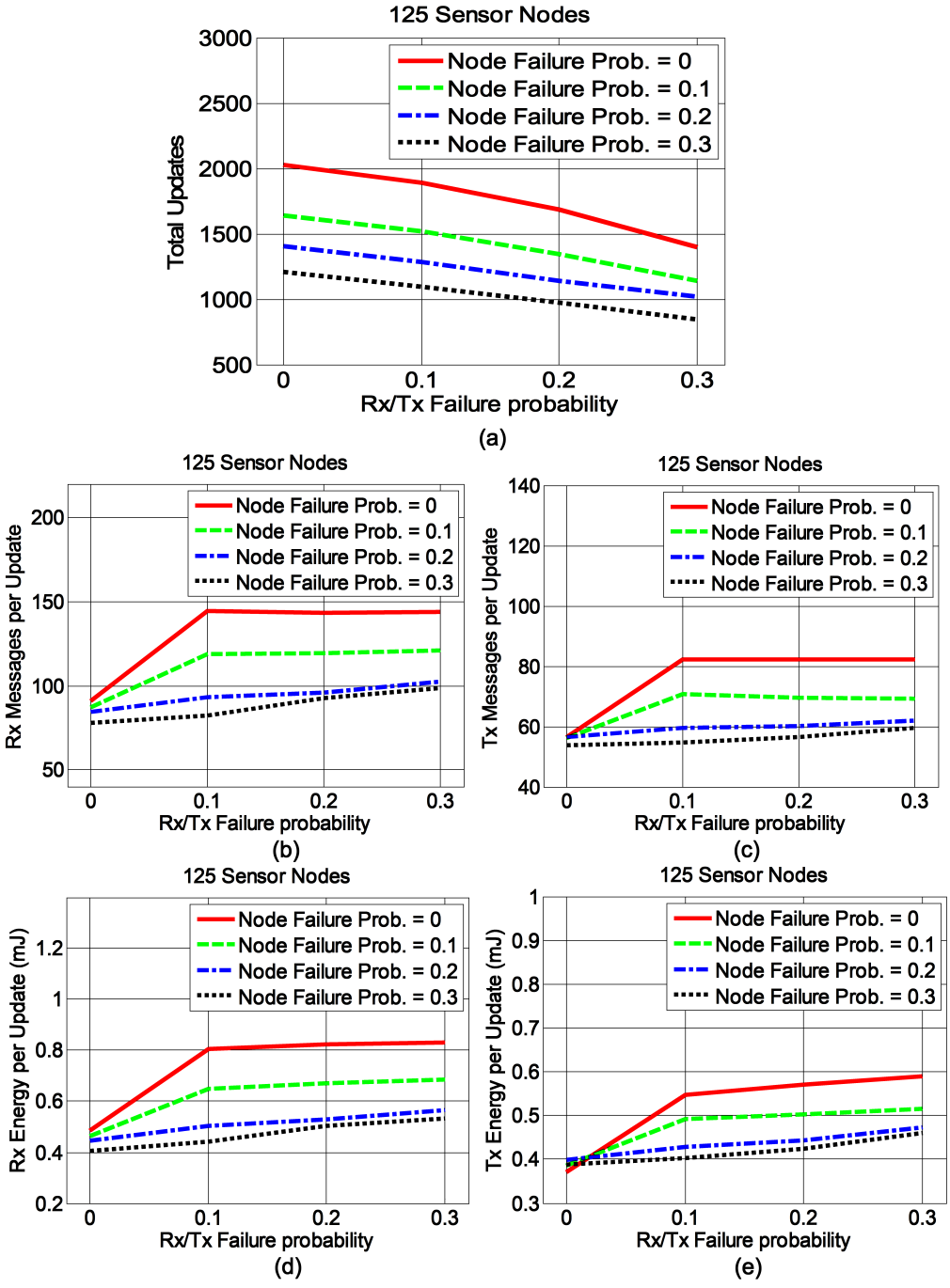


Figure 4.19: Experiment 2: Total number of updates, mean number of messages and Rx/Tx energy consumed per model update, for each node and Rx/Tx failure probability case considered (density = 125 sensor nodes per km^2).

Comparing corresponding results (corresponding table entries) we observe that the mean speed and direction errors are on average larger for Experiment 2 (relatively to Experiment 1) but only by 4.26% and 2.74^o degrees respectively (with standard deviations 0.46% and 0.42^o degrees). The slightly larger estimation errors observed for Experiment 2 are due to the more irregular front shapes and the more dynamic evolution of the hazard front's characteristics.

It is well recognized that the "captain wind" is the most critical factor affecting a wildfire's behavior. As we discuss, two of the five wildfire scenarios were simulated using light winds (speed $2m/s$), two using strong winds ($9m/s$) and one using intermediate wind conditions ($5m/s$). In order to investigate how wind speed affects the parameters estimation accuracy we compared the estimation errors for light and the strong wind cases. For strong wind scenarios the mean speed and direction estimation error increases on average by 3.92% (standard deviation 0.69%) and 1.91^o degrees (standard deviation 0.53^o) respectively compared to the corresponding errors obtained for light wind speed scenarios. This modest error increase is justified since strong winds result to larger front line speed variations (acceleration/decelerations) which are more difficult to track.

In the wildfire scenarios described above the wind speed and direction values were constant within the forest area. In order to investigate how spatial wind speed and direction variations may affect the estimation accuracy of the proposed algorithm we repeated the wildfire scenarios simulation of Experiment 2 using spatially varying wind fields. To generate realistic wind speed and direction perturbations we used WindNinja [91], which generates spatially varying wind parameters by modulating a reference value based on the terrain's morphology. The analysis revealed (results not shown) that when using spatially varying wind parameters the speed and direction estimation errors increased on average by 1.41% and 0.88^o degrees (with standard deviations 0.61% and 0.38^o) respectively. This modest increase is justified if we consider that wind speed and direction variations result to more irregular wildfire evolution patterns and larger front line speed variations which are more difficult to track.

4.4.3 Comparison to PRECO sheme

As mentioned in the Introduction, most in-network processing schemes reported in the literature try to delineate dynamically the boundaries of an evolving continuous object using a dense array of deployed sensor nodes [22--40]. These schemes do not attempt to estimate the local front line's evolution characteristics or predict their spatiotemporal evolution. One notable exception is the work in [37, 38] where the authors introduced a simple way to estimate, as we do, the speed and direction of the local front. They use them to implement a ``wake up'' mechanism to decide which ``sleeping'' nodes to activate selectively for near term front tracking in order to reduce the network's overall energy consumption. However, PRECO (PREdictive Continuous Object tracking scheme) requires global sensor nodes synchronization [32] rendering it impractical even for medium size WSNs. Nevertheless, for completeness purposes we compared it to our method under the scenarios of Experiment 1, Experiment 2, and Experiment 2 with spatially varying wind parameters.

Tables 4.5, 4.6, 4.7 provide for each sensor nodes density case (75, 100, 125 nodes per km^2) and node failure probability case (0, 0.1, 0.2, 0.3) of Experiment 1, Experiment 2 and Experiment 2 with spatially varying wind speed parameters (using WindNinja) respectively, the orientation error (in degrees) and the precent speed error when using the proposed in-network algorithm and the PRECO algorithm. For each scenario the provided statistics were computed by considering as sample points all local front updates for all the 50 simulation runs. A scenario corresponds to a set of simulation runs with a specific number of sensor nodes deployed in the area (e.g. 100) and a fixed nodes failure probability (e.g. 0.1).

Our method is shown to outperform considerably (for all sensing radii scenarios) PRECO, resulting to smaller estimation errors. When using very high node densities (thousand of sensor nodes per km^2) PRECO achieves reasonable accuracy, however it fails to estimate correctly the spatiotemporal characteristics of the continuous object in WSNs with practical sensor densities. This behavior can be explained if we consider the following:

PRECO considers as a local front (boundary line), a line segment that connects two adjacent special Boundary Nodes (BN), called Master Boundary Nodes (MBN). It uses the

Sensing Radius ($R_d = 0.1m$)						
$P(f)$	75 nodes		100 nodes		125 nodes	
	Orient.	Speed	Orient.	Speed	Orient.	Speed
0	4.46/11.01	12.77/13.09	4.07/10.22	12.58/13.86	4.26/10.01	12.07/13.1
0.1	4.84/10.41	13.11/13.13	4.19/10.31	12.74/13.73	4.07 /10.26	12.82/12.97
0.2	4.93/10.27	13.07/12.84	4.72/10.59	12.96/13.41	4.19/10.16	12.7/12.83
0.3	5.03/11.02	13.24/13.21	4.96/10.47	12.89/13.83	4.16/10.04	12.76/13.03
Sensing Radius ($R_d = 15m$)						
0	4.87/10.71	13.36/13.89	4.51/10.42	13.17/13.31	4.44/10.62	13.33/13.75
0.1	4.69/10.39	13.71/14.33	4.32/10.61	13.75/13.76	4.76/10.55	13.92/13.92
0.2	5.03/11.09	13.64/14.03	4.9/11.13	13.43/13.91	4.88/10.47	13.29/14.18
0.3	5.13/10.97	13.82/14.06	5.11/10.77	13.64/13.88	4.93/10.32	13.61/14.07
PRECO Algorithm						
0	28.44/23.31	41.27/90.83	27.09/21.73	37.91/90.1	25.73/19.15	36.13/86.42
0.1	28.86/22.66	40.76/91.28	27.56/23.04	39.18/88.47	26.84/20.64	37.18/84.21
0.2	29.32/23.11	43.17/94.21	28.16/23.72	41.03/89.9	26.67/19.72	38.89/87.5
0.3	31.04/23.89	45.82/97.64	28.81/24.18	42.17/92.16	28.46/21.18	38.46/90.63

Table 4.5: Summary of Experiment 1 Results: The Median/Inter Quartile Range of the orientation (in degrees) and percent speed errors of the proposed and the PRECO method ([37, 38]) under different density, probability of node failure conditions and sensing radii assumptions. For each condition the statistics were computed based on 50 simulation runs (50 WSN random deployments).

Sensing Radius ($R_d = 0.1m$)						
$P(f)$	75 nodes		100 nodes		125 nodes	
	Orient.	Speed	Orient.	Speed	Orient.	Speed
0	6.94/11.97	16.83/15.76	6.21/11.71	16.48/15.82	6.26/11.79	16.16/15.62
0.1	7.19/12.13	16.68/15.91	6.99/12.45	16.73/16.21	6.32 /11.7	16.42/16.44
0.2	7.35/12.09	17.27/16.62	6.61/11.79	16.97/15.99	6.7/11.62	16.71/15.97
0.3	7.33/12.07	17.14/16.48	7.16/12.36	17.13/16.09	7.09/12.13	16.94/16.49
Sensing Radius ($R_d = 15m$)						
0	7.18/12.17	17.25/17.05	6.79/12.26	16.87/16.9	6.68/11.73	17.22/15.78
0.1	7.24/12.36	17.39/16.83	6.91/11.85	17.27/17.16	6.93/11.52	17.51/16.27
0.2	7.43/11.98	17.66/17.4	6.74/12.02	17.81/16.73	6.85/11.9	16.99/16.14
0.3	7.86/12.09	17.86/16.68	7.47/12.06	17.72/17.58	7.27/12.14	17.18/16.03
PRECO Algorithm						
0	28.09/24.62	42.55/93.41	27.36/23.01	39.07/93.25	26.61/21.77	37.24/89.48
0.1	29.17./23.79	42.78/95.06	27.09/22.9	41.21/92.82	26.82/20.83	39.19/91.8
0.2	31.7/24.18	44.08/95.13	28.61/23.11	41.56/94.1	27.27/21.35	38.36/93.72
0.3	32.41/24.62	44.81/96.47	29.82/23.82	42.99/93.74	28.02/21.47	39.9/92.94

Table 4.6: Summary of Experiment 2 Results: The Median and Inter Quartile Range of the orientation (in degrees) and speed errors of the proposed and the PRECO method ([37, 38]) under different sensor density, probability of node failure conditions and sensing radii assumptions. For each condition the statistics are computed based on 50 simulation runs (10 random sensor node deployments for each one of the 5 wildfire evolution scenarios).

Sensing Radius ($R_d = 0.1m$)						
$P(f)$	75 nodes		100 nodes		125 nodes	
	Orient.	Speed	Orient.	Speed	Orient.	Speed
0	8.14/12.79	18.92/17.03	7.63/12.22	17.98/16.93	7.01/12.35	17.32/16.81
0.1	8.23/12.61	19.09/17.95	7.89/12.48	18.25/17.17	7.53 /12.19	17.74/17.65
0.2	8.11/12.32	19.41/17.5	8.03/12.36	18.66/17.35	7.92/12.4	17.9/17.44
0.3	8.32/12.56	19.26/17.86	8.31/12.82	19.09/17.91	7.74/12.66	18.23/17.59
Sensing Radius ($R_d = 15m$)						
0	8.24/12.57	19.55/17.14	7.85/12.35	18.21/17.09	7.24/12.62	17.2/17.02
0.1	8.63/12.95	19.07/17.28	8.01/12.49	18.38/17.86	7.71/12.45	17.69/17.26
0.2	8.4/13.11	19.33/17.61	8.23/12.76	18.89/18.18	7.6/12.79	18.23/17.18
0.3	8.63/13.26	19.74/17.36	8.47/13.01	19.27/18.11	8.22/12.55	18.59/17.83
PRECO Algorithm						
0	29.55/23.31	43.7/94.22	27.02/22.52	41.29/93.31	26.19/20.92	39.41/93.64
0.1	30.71./23.57	44.94/94.57	28.16/21.97	42.18/94.17	26.512/21.2	39.78/93.26
0.2	30.96/24.46	45.61/95.89	28.18/22.86	42.86/94.08	27.13/22.11	40.37/93.81
0.3	31.16/24.27	45.89/95.71	29.21/23.69	43.26/95.45	27.97/21.93	41.41/94.52

Table 4.7: Summary of Experiment 2 Results using space varying wind parameters: The Median and Inter Quartile Range of the orientation (in degrees) and speed errors of the proposed and the PRECO method ([37, 38]) under different sensor density, probability of node failure conditions and sensing radii assumptions. For each condition the statistics are computed based on 50 simulation runs (10 random sensor node deployments for each one of the 5 wildfire evolution scenarios).

location coordinates of the corresponding fixed MBNs to calculate the orientation parameter of a local front. In contrast, our method calculates the orientation of a local front based on the coordinates of two points (K_1 and K_2) estimated using two local speed observations of the diffusive hazard's front line (see Section 3.4.2). Our orientation estimation approach, which is independent from the sensor node locations, explains why this parameter's estimation accuracy is almost insensitive to WSN's density variations (see Section 4.4.1 paragraph 3). Moreover, the small number of MBNs present at low and realistic WSN density scenarios leads to a coarser piece-wise linear approximation of the diffusive hazard's boundary, which in turn explains the larger orientation estimation errors when using PRECO.

Finally, to estimate the evolution speed of a boundary line, PRECO uses the locations and time of detection of the MBNs and of their neighbors. As indicated by PRECO's speed equations (see formulas on page 4 in [37]), the speed's estimation accuracy depends on the number of MBN neighbors and their positions relatively to the continuous object front's evolution direction. In general, it is expected that as the number of MBN neighbors increases the accuracy of the local front speed estimates will also increase. It is important to mention that, in contrast to our algorithm PRECO, requires global synchronisation of the nodes which is difficult to achieve even in small scale WSNs.

We presented a distributed WSN algorithm for estimating accurately the spatiotemporal evolution parameters (orientation, direction and speed) of the local front of a diffusive hazard. The algorithm updates the local front model parameters and propagates them to sensor nodes situated in the direction of the hazard's propagation in a fully decentralized manner. Extensive simulation results show that the proposed scheme can estimate accurately the time-varying local parameters of different types of irregular fronts, while using WSNs of realistic density. Moreover, its estimation accuracy is robust to changes in WSN density, sensor node failures and communication link failures. Model parameters are updated based on closed form algebraic expressions making the presented approach practical and appealing for real-world hazard tracking applications. Relatively to other published schemes, our in-network algorithm exhibits the following unique characteristics: It works with *low* and realistic density WSNs, it is robust to sensor node and communication link failures which are certainly expected in harsh environments, and *does not* require

any sensor node clocks synchronization, which is very difficult to achieve anyway even in small scale WSNs operating in non-harsh environments. In next Chapter we will present emulation results that further support our claim that the proposed algorithm is suitable for large-scale WSN deployments.

Chapter 5

Assesing Requirements for Large Scale implementation using Simulation-Driven WSN Emulation

In this Chapter we present a novel method for emulating the operation of collaborative algorithms in large-scale WSNs by re-using a small number of available real sensor nodes. We demonstrate the potential of the introduced simulation-driven WSN emulation approach by using it to estimate how communication and energy costs scale with the network□s size when implementing our collaborative WSN algorithm (presented in Chapter 4) for tracking the spatiotemporal evolution of a continuous object.

5.1 Motivation

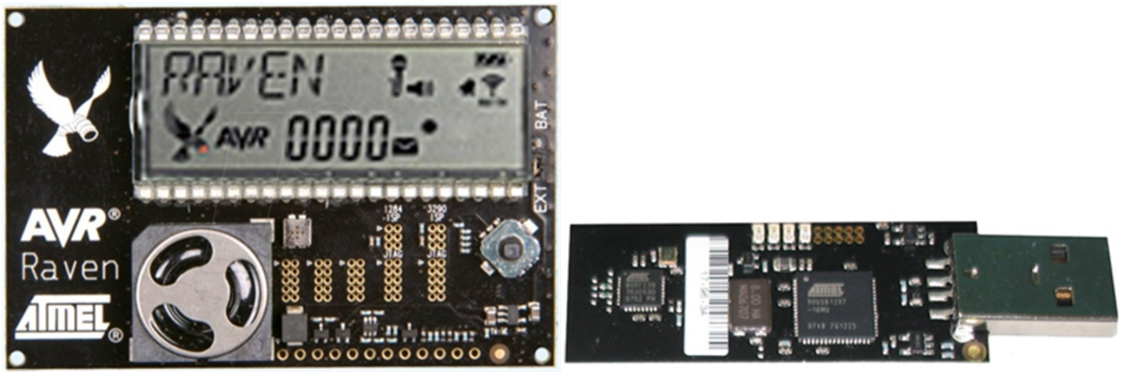
For all WSN schemes, computer simulations can be used to assess the expected WSN behavior as a function of its density. However simulations fail to provide: a) accurate energy consumption estimates and how they scale with the size of the network, and b) information about the processing and memory requirements of the distributed algorithm's implementation. Since having such estimates is very important before attempting to deploy a large-scale WSN for environmental monitoring application the real question becomes, how can we meet this requirement without having to deploy a large-scale WSN?

To address this question we introduce a method which allows us to emulate the operation of a large-scale WSN deployment for environmental applications by reutilizing only a small number of real sensor nodes. The key idea of the proposed method is to re-allocate (virtually reposition) the available sensor nodes so that they implement WSN nodes located close to the hazard□s front line as it evolves. Sensor nodes re-allocation has been used before in [92] to evaluate the performance of geographical routing protocols in large scale WSN. The scheme used in [92] has been specifically designed for evaluating routing protocols before network deployment and cannot be applied for the evaluation of energy aspects of collaborative WSN algorithms in general. To the best of our knowledge our proposed emulation scheme is the first attempt to use a small number of sensor nodes to realistically estimate the energy consumption of a collaborative algorithm in a large-scale WSN implementation. We demonstrate its capabilities using the distributed algorithm we introduced in Chapter 4 for estimating the spatiotemporal evolution parameters of diffusing environmental hazards. WSN emulation provides convincing evidence that the collaborative algorithm is suitable for large-scale WSN deployment since it respects the memory, processing and energy constraints of commodity sensor nodes used in WSN implementations.

5.2 WSN Implementation

5.2.1 WSN Platform Specifications

The WSN implementation of the proposed collaborative algorithm was based on the affordable Atmel Raven evaluation kit [85] consisting of AVRRAVEN boards (sensor nodes - see Figure 5.1a) and RZUSBSTICK boards (sink node - see Figure 5.1b). The AVRRAVEN board has three main modules [93]: The ATmega3290 8-bit MCU which has 32 KB ISP flash memory, 1 KB EEPROM, 2KB SRAM and is responsible for handling the on board sensors. The ATmega1284P MCU, which has 128 KB ISP flash, 4KB EEPROM, 15KB SRAM, and is responsible for handling the 2.4GHz AT86RF230 radio transceiver designed for low-cost IEEE 802.15.4 applications. Its transmission power can be adjusted in the range [-17dBm, 3dBm] and its reception sensitivity was fixed to -101dBm.



(a) (b)

Figure 5.1: AVR Raven evaluation kit: (a) AVRRAVEN board (sensor node), (b) RZUSB-STICK boards (sink node) (adopted from [94]).

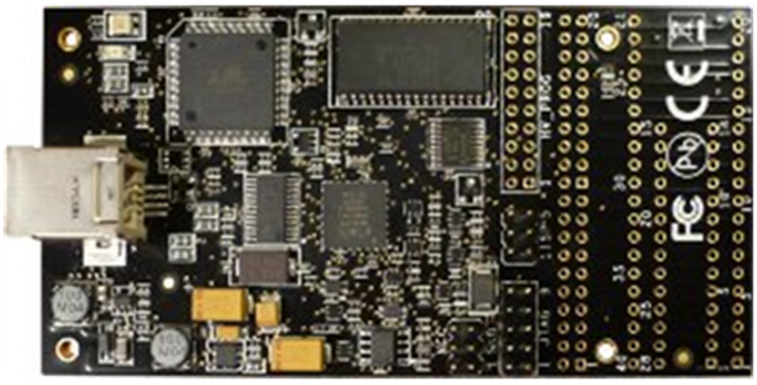


Figure 5.2: AVR Dragon programmer (adopted from [95]).

The distributed algorithm was coded in C on the IPv6 ready RTOS Contiki [84]. Contiki is an open source operating system for networked, memory-constrained systems with a particular focus on low-power wireless Internet of Things devices. The RTOS and the C code were loaded on the AVRRAVEN boards using the AVR Dragon programmer [96] (see Figure 5.2) and occupied 71KB on the ATmega1284P, i.e. 55.4% of its total ISP flash memory. We also designed a Java application, called RavenObserver, running on the host PC, to monitor the WSN and collect data from the sensor nodes during the conducted experiments.

5.2.2 Simulation-Driven Emulation Workflow

Deploying a large-scale WSN to validate an in-network algorithm is unrealistic. To overcome this fundamental limitation we developed a simulation-driven emulation procedure which allows us to mimic the behavior of a large-scale WSN, during the evolution of a diffusive hazard, using only a small number of real sensor nodes. The basic idea of the proposed scheme is to cleverly re-use sensor nodes which are no longer able to participate in the distributed algorithm. Specifically we developed a technique which allows us to virtually re-position these nodes forward, in the direction of the hazard□s front movement. For its virtual repositioning to be possible, a node should satisfy the following conditions: a) it must be at Quiescent state, and b) it must have detected the front of the phenomenon. These conditions guarantee that the sensor node cannot participate to the distributed algorithm any more.

Using the Matlab-based WSN simulator presented in (Chapter 4) we were able to create simulation scenarios with different sensor node densities, deployment strategies, and progressing hazard front evolution characteristics. For our evaluation we modified the simulator so that it can also generate an ASCII file containing the following setup record for each sensor node: $\{$node ID, location coordinates, time of hazard□s detection, IDs of its neighbors$\}$. Using this file as input, the *RavenObserver* coordinates the re-use and virtual repositioning of the available sensor nodes (6 in our case), in order to emulate the behavior of the large-scale WSN as prescribed by the Matlab simulation. Figure 5.3 shows a UML diagram of the proposed simulation driven emulation workflow.

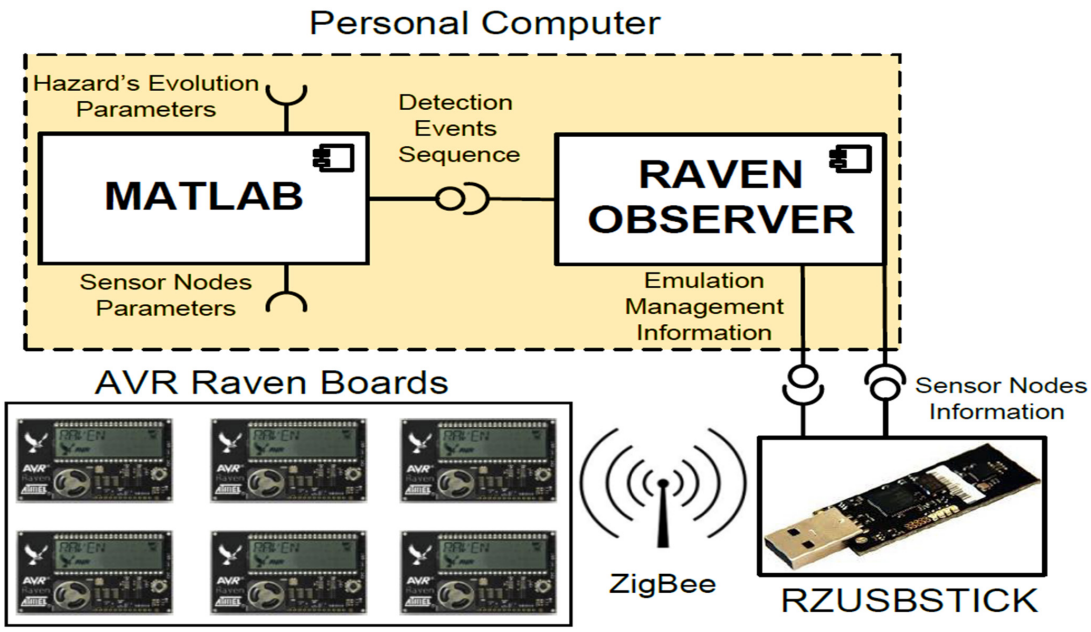


Figure 5.3: Simulation driven emulation workflow.

Assuming that N real sensor nodes are available when a WSN emulation experiment is initiated, *RavenObserver* checks the Matlab generated file and extracts, for the N nodes that have detected the phenomenon first, the aforementioned setup records. Then, the WSN sink node takes over and sends this information to the available N real sensor nodes. Upon reception, the RavenObserver starts an internal timer and checks in the Matlab generated file the hazard's expected detection times for the aforementioned sensor nodes. When the timer reaches the detection time of a sensor node, RavenObserver asks the sink to send a special message to it in order for that node to start emulating the detection of the hazard. Upon reception of this message, the node changes its status and acts as prescribed by the algorithm. Finally, when a node is no longer able to participate in the algorithm (satisfies the necessary aforementioned conditions), it sends a special message to the sink, which in turn informs RavenObserver that this sensor is available for re-allocation. Then RavenObserver finds the record for the node that is expected to detect the phenomenon next (according to the hazard's evolution simulation) and asks the sink to forward this record to the freed sensor node. This emulation method works *w.l.o.g* with any number N of available sensor nodes ($N > 3$), albeit the emulation time depends on that number.

5.3 Evaluation Results

We now present experimental results, obtained using the available ATMEL Raven sensor nodes that helped us assess the processing, communication and energy efficiency of the collaborative algorithm and how it is expected to scale with the WSN's size in practice. Since the main contribution of the specific algorithm is its ability to estimate accurately the hazard's evolution parameters using low density WSNs, in our experiments we used node densities that are considered low for environmental applications. Specifically we used 5×10^{-5}, 7.5×10^{-5}, 10^{-4} $sensors/m^2$, which correspond to 50, 75 and 100 sensor nodes deployed within an area of $1km^2$. In order to establish that we have a connected network we use the transmission (Tx) powers shown in Figure 5.5. A Matlab program was used to generate random sensor node deployments. With N AVR Raven nodes available, we can emulate deployments in which every sensor node has at most $N-1$ neighbors, and N is larger than 3. To simulate realistically the behavior of a diffusive hazard we used a wildfire simulation software called FLogA (Fire Logic Animator) developed in our group [87]. FLogA is a web-based interactive software tool which allows us to draw a forest area on Google Earth [90] anywhere in Europe, insert ignition points, simulate realistically and geo-animate the behavior of the evolving fire line under different prevailing wind conditions. Using FlogA we have generated five different fire scenarios affecting the same square forest area of $1km^2$ in Hymettus mountain, Attica, Greece (see Experiment 2, Section 4.4, Chapter 4). For each scenario, the fire ignition points were placed at different locations, giving rise to very different wildfire front evolution patterns. The duration of each wildfire experiment was set to $180min$ in order to guarantee that most of the forest area would be affected by the wildfire.

During the emulation of the network's operation RavenObserver collects the following information from the available Raven sensor nodes: a) the number of the received/transmitted (Rx/Tx) messages b) the number of the Rx/Tx Bytes c) the energy consumed by Rx/Tx operations and d) the computation time required for each model update. This data is collected by the sink node. At the end of the emulation, it is analyzed at two levels:

- *Network-level:* Considers the data of all sensor nodes participating in the simulation.
- *Cluster-level:* Analyzes the data of the nodes participating in the local front's model

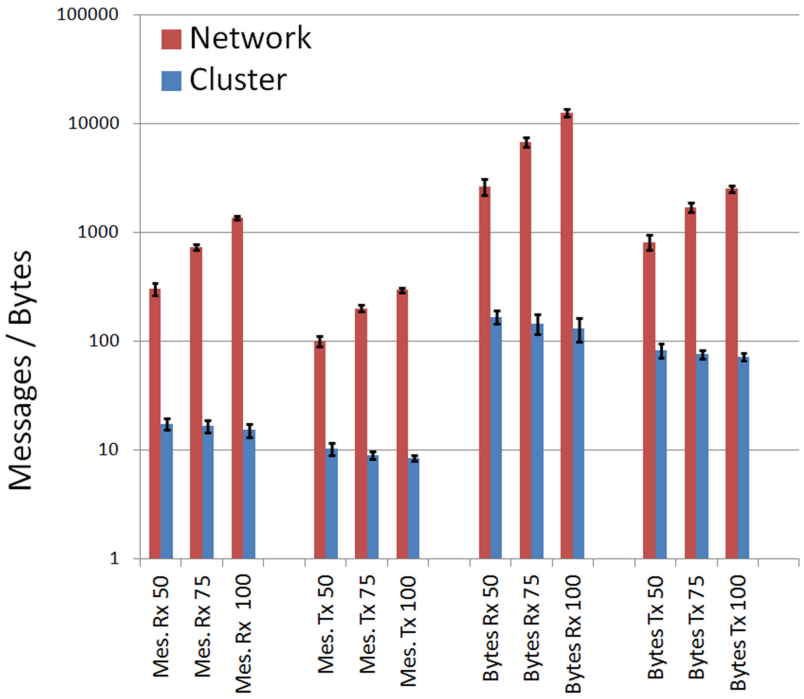


Figure 5.4: Mean and stdev of received/transmitted Messages/Bytes, at the Cluster and Network level for different WSN densities.

updating procedure (Master and its two Helpers).

Figure 5.4 provides, for each analysis level, the mean of the Rx/Tx Messages and Bytes and their stdev, for three density scenarios. The statistics for each scenario were computed considering all the corresponding Rx and Tx Messages/Bytes for the 50 different simulation runs (10 random deployments * 5 wildfires). We observe that the bar plots for the Messages (Bytes) follow similar trends as the number of deployed sensors increases. This is as expected due to the direct relation between Messages and Bytes for a given density. From the cluster-level analysis, we observe that as the number of deployed sensors increases the number of the Rx and Tx Messages and Bytes decreases a little. This behavior is justified if we consider that: a smaller number of sensor nodes in a given area implies fewer neighbors which in turn implies that it is more difficult for a Master node to □inherit□ its Master status to one of its Helpers. As discussed in in Section 4.2.3 (Model propagation) in Chapter 4, when a Master cannot find a new qualified Master, it is forced to broadcast a message to its Helpers, so that they can propagate its updated model to their neighbors. These extra □negotiations□ are responsible for the small increase of Messages (Bytes) as the number of the deployed sensors decreases.

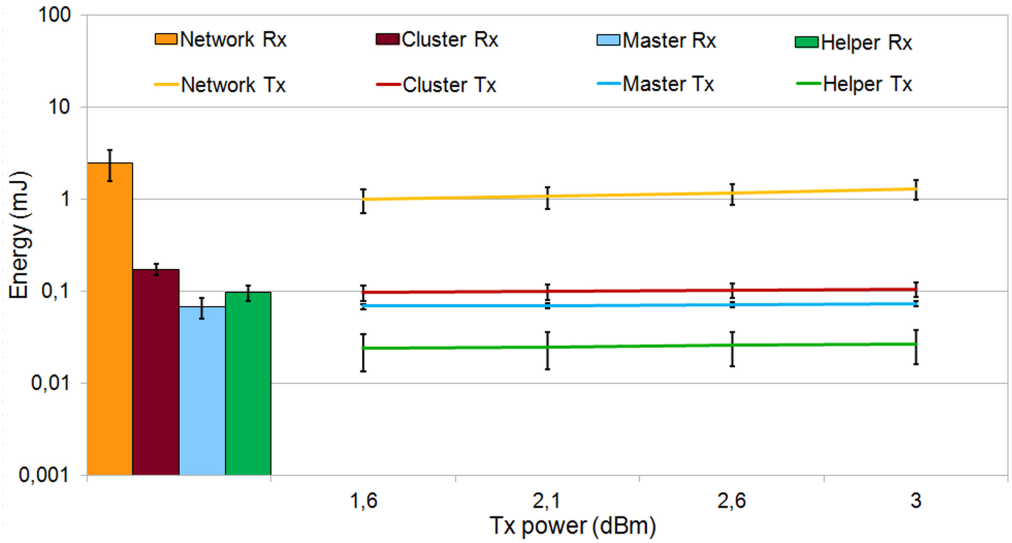


Figure 5.5: Mean and stdev of energy consumed for Rx and for Tx operations (as a function of the node's Tx power). Network and Cluster-level analysis for the 50 nodes scenario.

However, at the network level an increase in the number of deployed sensor nodes leads to an increase in the total number of model updates, and therefore the total number of the Rx and Tx Messages and Bytes increases accordingly.

Figures 5.5, 5.6 and 5.7 provides, for the 50, 75 and 100 sensor nodes density scenario respectively, the mean and stdev of energies consumed for Rx/Tx. For both levels of analysis, the energy plots show that the required Rx energy is on average larger than the Tx energy. At a first glance this may seem counterintuitive, but it can be explained if we consider that most of the messages are of broadcast type, which means that a single Tx corresponds to many Rxs (by nodes in the same neighborhood). Furthermore, a more detailed analysis at the cluster level shows that the Master consumes for Tx (blue line) about 5 times more energy than both of its Helpers combined (green line). This happens because the Master transmits many more messages than its Helpers during the model's forward propagation (see Section 4.2.3 in Chapter 4). The large stdev observed for the Helpers Tx energy is due to the negotiation which takes place between the Master and either one, or both, of its Helpers (in sequence) during the model's propagation phase.

Measuring the energy consumed by the nodes for processing and sensing tasks during emulation was not feasible since the AVRRAVEN nodes do not provide such functions. However, since it is well known [97, 98] that the radio communication is the prominent

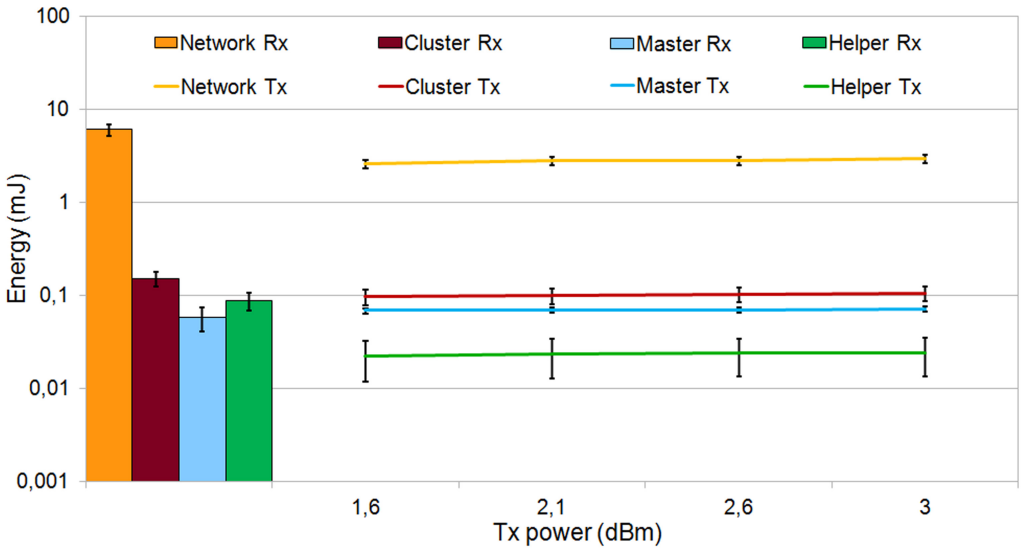


Figure 5.6: Mean and stdev of energy consumed for Rx and for Tx operations (as a function of the node□s Tx power). Network and Cluster-level analysis for the 75 nodes scenario.

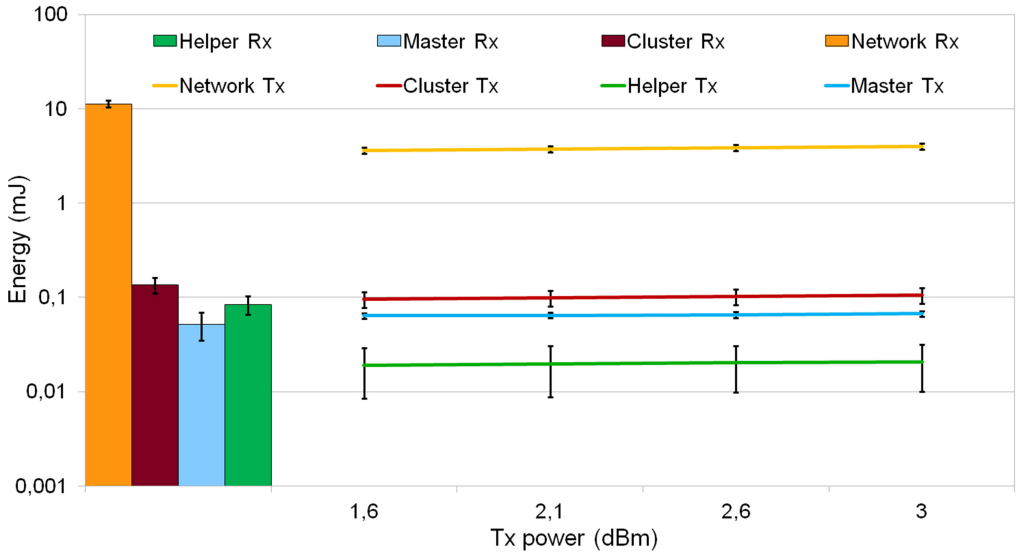


Figure 5.7: Mean and stdev of energy consumed for Rx and for Tx operations (as a function of the node□s Tx power). Network and Cluster-level analysis for the 100 nodes scenario.

energy consumer in a WSN, we can safely conclude that the provided Rx/Tx Network level energy results provide a good estimate of the lower bound of the total WSN energy consumed by the nodes during the 180min of operation captured by the wildfire simulation. This conclusion is further supported by the fact that the sensor nodes activate (exit the Quiescent status) only for the time period where the fire front line is close to their vicinity (emulated period). Furthermore, the mean computation time for a model update, required by the 8-bit ATmega1284P MPU of the Raven sensor node clocked at 8MHz, is about 523 ms (492 ms fro computation and 31ms for communication).

We presented a simulation driven emulation procedure which allow us to realistically evaluate even by using a small number of ``real'' sensor nodes the behavior of the collaborative WSN algorithms. We demonstrate the validity of the approach by evaluating the proposed collaborative algorithm presented in Chapter 4. The results clearly indicate that our algorithm algorithm is suitable for a large-scale WSN deployment, since it respects WSNs' communication, processing, memory and energy constraints. The proposed emulation approach can be followed to assess the practicality of large-scale WSN deployment of other in-network algorithms of similar nature for environmental monitoring applications.

Chapter 6

Continuous Object Boundary Reconstruction Algorithm

In this Chapter we present a novel algorithm which reconstructs with accuracy the boundary of an evolving continuous object using a small number of local front estimates. Each local front estimate describes locally the evolution characteristics (orientation angle, direction and speed) of the continuous object's boundary. When a sufficient number of local front estimates becomes available at a fusion center the algorithm combines their information and determines a ``smooth'' curve that approximates the object's boundary. Simulation results demonstrate its ability to reconstruct with accuracy the boundary of complex evolving objects, even in cases where the local front estimates are distorted with error.

6.1 Preliminaries

The key idea of the proposed boundary reconstruction algorithm is as follows: Let's assume that a monitoring system (e.g. based on WSN technology - see Chapters 4) is able to estimate the evolution characteristics (orientation angle, direction and speed) of a continuous object at different locations and/or time instances (see black segment in Figure 6.1a). As soon as a sufficient number (application dependent) of local front estimates becomes available, the proposed algorithm combines their information and determines the

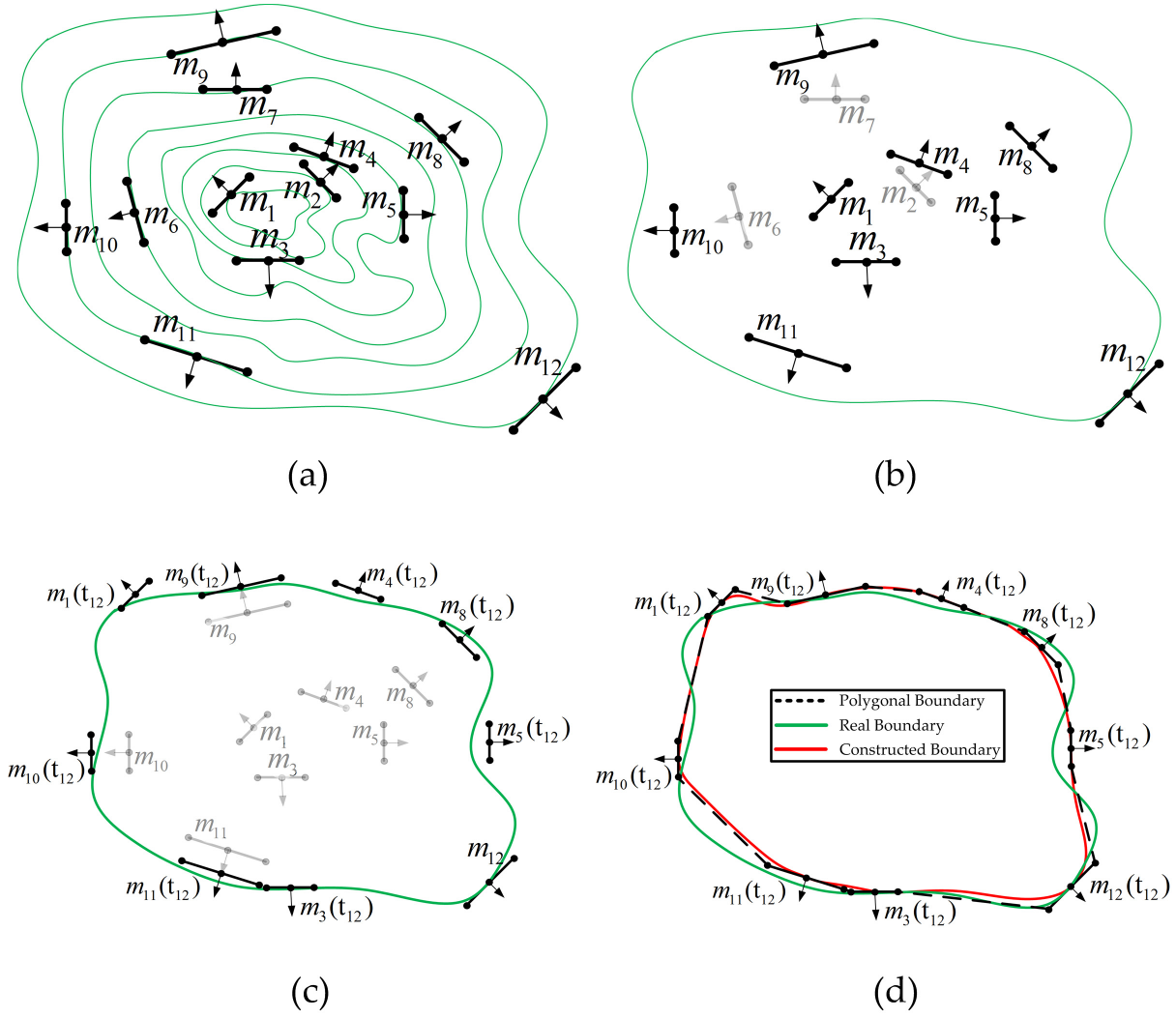


Figure 6.1: a) The green curves show different instances of an evolving continuous object's boundary; each boundary corresponds to the time instance where a local front estimate (black segments) takes place. The black segments correspond to the selected local front estimates that will be used to determine the continuous object's boundary at time t_{12}. c) The new locations of the selected local front estimates after their space-time evolution at time t_{12}. d) The polygon (black dashed polygon) and the smooth curve (red curve) that approximate the continuous object's boundary.

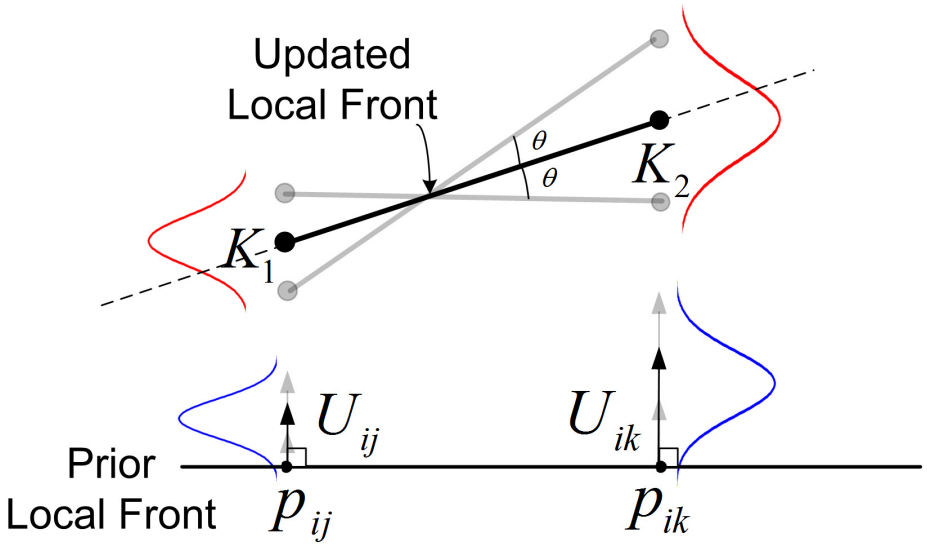


Figure 6.2: Orientation angle uncertainty of a local front model.

set of local front estimates (black segments in Figure 6.1b) that will be used to reconstruct the boundary of the continuous object. In sequence, using their evolution characteristics it determines their locations at the time instance that we wish to reconstruct the continuous object's boundary (time t_12 see Figure 6.1c). Using the ``new'' location coordinates and the evolution direction parameters of the local fronts, the proposed algorithm determines a polygon that approximates the continuous object's boundary (see black dashed polygon in Figure 6.1d). Next, based on the uniform cubic B-splines the algorithm determines a curve (see red curve in Figure 6.1d) that approximates the object's boundary. Finally, based on the estimation uncertainties of local fronts' parameters, the algorithm produces a probability field, that indicates for each point of the considered area, the probability to be affected by the continuous object.

6.1.1 Local Front Parameters

In this section we state the notation used, and everything else needed to facilitate the presentation of the proposed boundary reconstruction algorithm.

As discussed in Section 3.4.2, the orientation parameter of a local front model is calculated using the coordinates two points $K_1 = (x_1, y_1)$ and $K_2 = (x_2, y_2)$. To calculate the coordinates of these points, the algorithm uses the mean speed values $\{u_{ij}, u_{ik}\}$ of the

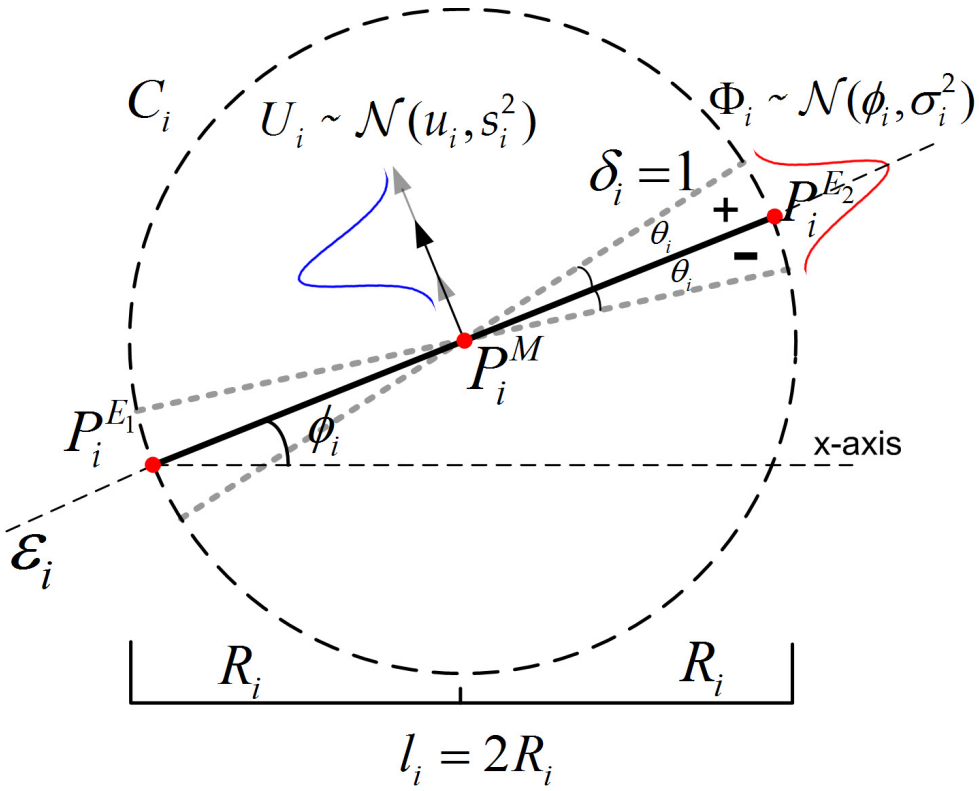


Figure 6.3: The local front estimate m_i and its evolution parameters.

projections points $\{p_{ij}, p_{ik}\}$ respectively (see Figure 6.2). As indicated by equation (3.6), the speeds of the projection points follow normal distributions of the form $U_{ih} = \mathcal{N}(u_{ih}, s_{ih}^2)$ where $h \in \{j, k\}$. However, the speed uncertainties (s_{ih}^2) of projection points, imply in turn uncertainty about the locations of K_1 and K_2 and therefore uncertainty about the estimated orientation parameter (see Figure 6.2). It can be proved that the orientation angle of an updated local front will also follow Normal distribution with mean value ϕ_i and standard deviation $\sigma_i = \frac{\theta_i}{3}$ (see Figure Figure 6.3).

A local front estimate m_i (subscript i is used to uniquely identify a local front estimate) described by a line segment with the following parameters (see also Figure 6.3):

- $P_i^M = (x_i^M, y_i^M)$ (Location): The coordinates of the physical location of the line's segment middle point.
- l_i (Length): 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.
- δ_i (Evolution direction): A vector perpendicular to the local front's line segment. The

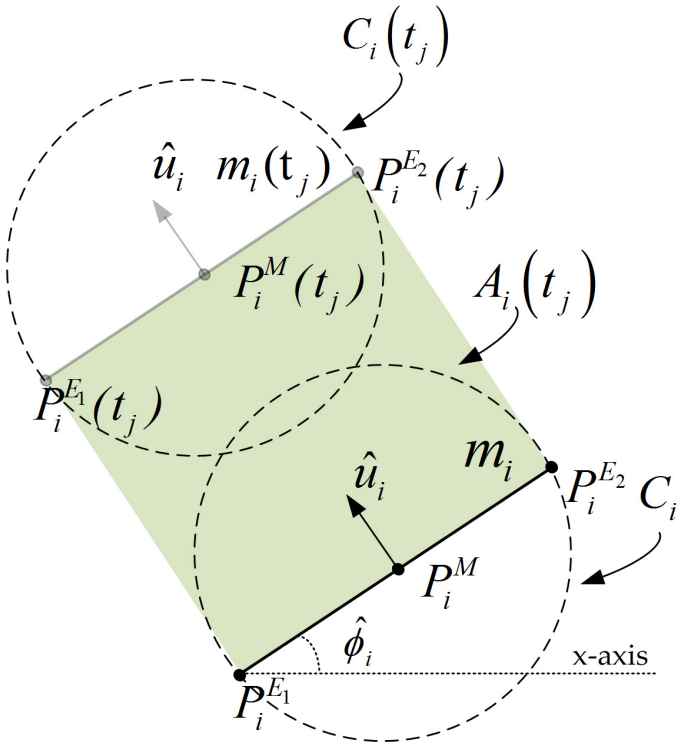


Figure 6.4: The space-time evolution of the local front estimate m_i at time t_j.

direction coefficient may take one of the following values: +1(-1), if the local front evolves into the positive (negative) half plane that determined by the line on which the local front's line segment lies (line ε_i in Figure 6.3).

- t_i (Local front parameters estimation time): The time (subject to a global time reference) where the local front's parameters estimation occurs.

- ϕ_i, σ_i (Orientation parameters): The angle Φ_i that formed between the local front's line segment and the horizontal axis (x-axis) is considered to be a random variable Φ_i that follows a Normal distribution $\Phi_i = \mathcal{N}(\phi_i, \sigma_i^2)$.

- $\hat{\phi}_i$ (Angle realization): The angle $\hat{\phi}_i$ is a realization of Φ_i (random sample) and consists the angle value of m_i that will be used by the boundary reconstruction algorithm.

- u_i, s_i (Speed parameters): The speed U_i of the local front's line segment is considered to be a random variable that follows a Normal distribution $U_i = \mathcal{N}(u_i, s_i^2)$.

- $\hat{u}_i$ (Speed realization): The speed $\hat{u}_i$ is a realization of U_i (random sample) and consists the speed value of m_i that will be used by the boundary reconstruction

algorithm.

Using the middle point's coordinates $P_i^M = (x_i^M, y_i^M)$, the length l_i, and the orientation realization $\hat{\phi}_i$ 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 $z = \{1, 2\}$ (see Figure 6.3 and Appendix D).

We will use the notation $m_i(t_j)$ to denote the space-time evolution of the local front estimate m_i at time instance t_j (where $t_i < t_j$). The middle point, the end points and the circular area of the space-time evolved local front $m_i(t_j)$ will be denoted as $P_i^M(t_j) = \big(x_i^M(t_j), y_i^M(t_j)\big)$, $P_i^{E_z}(t_j) = \big(x_i^{E_z}(t_j), y_i^{E_z}(t_j)\big)$ (where $z = \{1, 2\}$) and $C_i(t_j)$ respectively (see Figure 6.4). The algebraic expressions used to calculate the location coordinates of a space-time evolved local front are presented in Appendix E. Finally, we will use the notation $A_i(t_j)$ to denote the area covered by m_i after its space-time evolution at time t_j (see green shaded area in Figure 6.4).

6.2 Finding Parts of the Boundary

In this section we present the procedure that determines the set of the local front estimates that will be used to reconstruct the continuous object's boundary.

Let's assume that at time instance t_n, we decide to determine the continuous object boundary at time instance t_b (where $t_b \in \mathbb{R}_+$). From the available local front estimates we select those that have estimation times $t_i \leq t_b$, and form a set $\mathcal{M}_n$ that contains their parameters (see Section 6.1.1).

- If $\{|\mathcal{M}_n| < N\}$: The algorithm exits since M_n does not contain the minimum number N (determined by the user) of local fronts estimates to reconstruct the continuous object's boundary.

- If $\{|\mathcal{M}_n| \geq N\}$: The algorithm initiates the *local fronts' information processing* procedure (presented below).

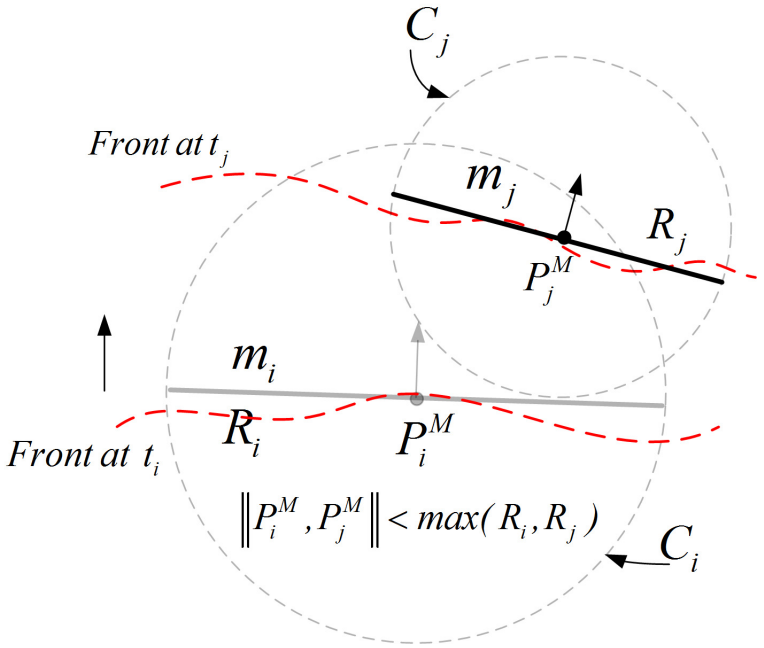


Figure 6.5: Model selection procedure: Local front m_i and m_j (where $t_i \leq t_j$) is assumed to describe the evolution behavior of the same part of an evolving boundary. Only the more recently estimated local front (m_j) is kept in $\mathcal{M}_n$ (see text for details).

6.2.1 Local Front Information Processing

This procedure combines the information of the local front estimates in $\mathcal{M}_n$ and generates a new set of local fronts estimates that describes the location and evolution behavior of the continuous object's boundary at time instance t_b. The procedure has three steps (described below).

First step

We assume that two local front estimates (m_i and m_j) describe the evolution behavior of the same part of the boundary, if the Euclidian distance of their middle points ($||P_i^M, P_j^M||$) is smaller or equal to the larger radius of the corresponding local front circular areas (see Figure 6.5).

$$||P_i^M, P_j^M|| \leq max(R_i, R_j) \tag{6.1}$$

For each pair of local front estimates in $\mathcal{M}_n$ (e.g. $\{m_i, m_j\}$) that satisfies the condition in (6.1), we keep (in $\mathcal{M}_n$) only the more recently updated local front estimate (m_j in Figure

6.5, where we assume *w.l.o.g* $t_i \leq t_j$). The reason we keep the more recently updated local front estimate is based on the assumption that it describes better the current time varying evolution characteristics of the continuous object. At the end of this step, we sort the remained local fronts in $\mathcal{M}_n$ in estimation times ascending order and form a new set $\mathcal{M}_f \subseteq \mathcal{M}_n$ that contains their parameters. In sequence, we check the following conditions:

- If $\{|\mathcal{M}_f| < N\}$: The algorithm exits since $\mathcal{M}_f$ does not contain the minimum number N (determined by the user) of local fronts estimates to reconstruct the continuous object's boundary.

- If $\{|\mathcal{M}_f| \geq N\}$: The procedure continuous to the second step (see below).

Second step

This step determines the set of local front estimates that provides information about the location and the evolution characteristics of the continuous object's boundary at time instance t_b. Finding the space-time evolution of the local front estimates in $\mathcal{M}_f$ at time t_b, without considering possible intersections of their evolution paths, usually results the construction of boundary shapes that deviates from reality. Below we propose a method that identifies and handles the events that are responsible for the boundary's shape deviations.

Event 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 occur:

Event 1: If $\{\{P_i^M(t_j) \in C_j\}$ *or* $\{P_j^M \in C_i(t_j)$ *and* $P_j^M \notin A_i(t_j)\}\}$ holds (see Figure 6.6a), we capture as event's time the time instance t_j.

Event 2: If $\{P_j^M \in A_i(t_j)\}$ holds (see Figure 6.6b), we capture as event's time the time instance t_j.

Event 3: If there is a time instance t_d ($t_i \leq t_j < t_d \leq t_b$) where the condition$\{\{P_i^M(t_d) \in A_j(t_d)\}$ *or* $\{P_j(t_d) \in A_i(t_d)\}\}$ holds (see Figure 6.6c), we capture as event's time the time instance t_d.

Event 4: If there is a time instance t_f ($t_i \leq t_j < t_f \leq t_b$) where $\{\{P_i^M(t_f) \in C_j(t_f)$ *and* $P_i^M(t_f) \notin A_j(t_f)$ *and* $||P_i(t_f), P_j(t_f)|| = max(R_i, R_j)\}$ *or* $\{P_j^M(t_f) \in C_i(t_f)$ *and* $P_j(t_f) \notin$

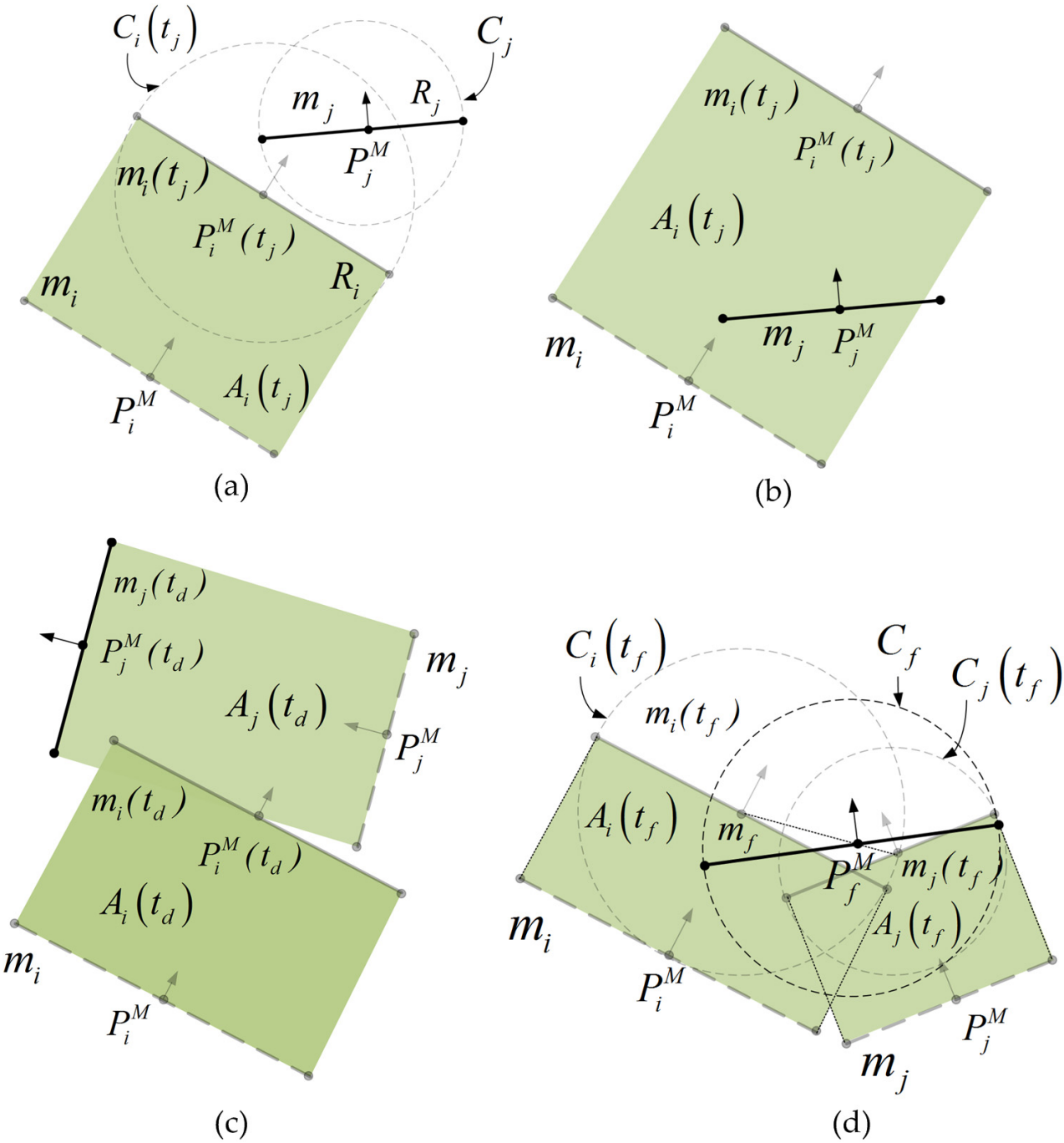


Figure 6.6: The four events that require special handling during the space-time evolution of the local fronts estimates (see text for details).

$A_i(t_f)$ *and* $||P_i(t_f), P_j(t_f)|| = max(R_i, R_j)\}\}$ holds (see faded grey local fronts in Figure 6.6d), we capture as event's time the time instance t_f.

After determine the local fronts' pairs in $\mathcal{M}_f$ that participate to an event, we find the pair with the smallest captured event time. We have to note that if more than two pairs of local fronts have equal event times, we select the pair with the smallest sum of local fronts' estimation times (e.g. $t_i + t_j$).

Event handling procedure: This procedure takes as input the local fronts' pair (e.g. $\{m_i, m_j\}$) that determined by the *events identification procedure*, and acts as follows:
Handling Event 1: In this event the local front estimates $m_i(t_j)$ and m_j describe at time instance t_j the evolution behavior of the same part of the continuous object boundary (see Figure 6.6a). Using the assumption that the more recently estimated local front describes better the current evolution behavior of the continuous object's boundary we keep in $\mathcal{M}_f$ only the most recently estimated local front (m_j).
Handling Event 2: In this event the local estimate m_j appears inside the area which has already been covered by $m_i(t_j)$ (see green shaded area in Figure 6.6b). This event implies that the local front $m_i(t_j)$, fails to describe with accuracy the evolution behavior of the continuous object boundary, since the local front m_j at time t_j, should be located on the continuous object's boundary and not inside the affected area. Based on this assumption we keep in $\mathcal{M}_f$ only the local front estimate m_j.
Handling Event 3: In this event the evolution path of a local front estimate ($m_i(t_d)$) intersects the evolution path of another local front estimate ($m_j(t_d)$) (see Figure 6.6c). To handle this event we erase from $\mathcal{M}_f$ the local front estimate ($m_i(t_d)$) that inserts into the area which has already been covered by the other local front ($m_j(t_d)$).
Handling Event 4: In this event the local front estimates ($m_i(t_f)$ and $m_j(t_f)$ in Figure 6.6d) meet during their space-time evolution. To handle this event we apply a technique that fuses the information of the local front estimates $\{m_i(t_f), m_j(t_f)\}$ and produces ``new'' local front estimate $\{m_f\}$ that describes at time $t_f$ the local evolution behavior of the continuous object's boundary (see black local front $m_k$ Figure 6.6d). To estimate the parameters of the ``new'' local front estimate we apply the following equations:

$$\begin{aligned} \mathcal{Z}_f &= \sum_{h=\{i,j\}} w_h \mathcal{Z}_h \;\; where \;\; \mathcal{Z} = \{\hat{u}_f, \hat{\phi}_f, l_f, x_f, y_f\} \\ u_f &= \sum_{h=\{i,j\}} w_h u_h, \;\; s_f^2 = w_i s_i^2 + w_j s_j^2 + w_i w_j (u_i - u_j)^2 \\ \phi_f &= \sum_{h=\{i,j\}} w_h \phi_h, \;\; \sigma_f^2 = w_i \sigma_i^2 + w_j \sigma_j^2 + w_i w_j (\phi_i - \phi_j)^2 \end{aligned} \tag{6.2}$$

As the local front's parameters estimation time t_k we assume the event's time instance t_f. The derivation of the equations in (6.2) and the calculation of the weights $\{w_i, w_j\}$ are presented in detail in Appendix F.

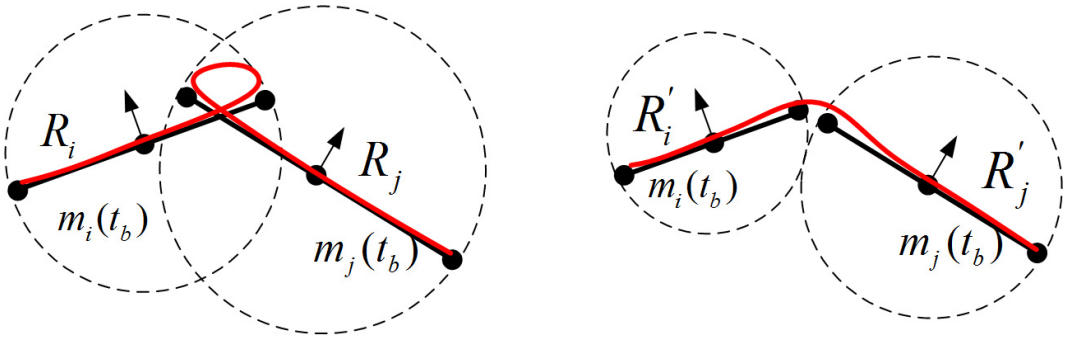
To estimate the evolution direction parameter δ_f we use the local fronts orientations ($\hat{\phi}_h$), the evolution directions (δ_h) and the weights (w_h) and determines two vectors which: a) are perpendicular to the corresponding local fronts segments, b) point to the corresponding local fronts' evolution directions and c) the ratio of their lengths is equal to the corresponding ratio of their weights $\frac{w_i}{w_j}$. In sequence, using these vectors the algorithm 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 and we assign the corresponding value (+1 or -1) to the direction parameter δ_f.

After calculating the parameters of m_f, we add its information in $\mathcal{M}_f$ and erase the information of its ``parents'' (m_i and m_j).

When handling an event, the procedure repeats the second step from the beginning. The second step completes when there are no events between the local fronts estimates in $\mathcal{M}_f$. At the end of the second step the procedure checks:

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 time t_b, forms a new set ($\mathcal{M}_b$) that contains their information and proceeds to the third and final step of the *local fronts' information processing procedure* (described below).

Else, the algorithm exits.



(a) (b)

Figure 6.7: The local fronts' lengths adjustment procedure. a) The overlap of the local front segments, results a curly B-spline curve during the boundary reconstruction procedure, b) After applying the size adjustment procedure we avoid the curly curve formation.

Third step

In this step the procedure adjusts the lengths of the local front estimates in $\mathcal{M}_b$, to avoid possible boundary shape's irregularities. The procedure checks if there is (are) any pair(s) of local front estimates in $\mathcal{M}_b$ where their local circular areas overlap (e.g. see $\{m_i(t_b), m_j(t_b)\}$ Figure 6.7a). If there is (are), we adjust their lengths such as to satisfy the following equation:

$$R_i' + R_j' = ||P_i^M(t_b)P_j^M(t_b)|| \tag{6.3}$$

The equation in (6.3) indicates that the circular areas of the adjusted local front segments should osculate externally (see Figure 6.7b). We propose a technique that appropriately adjusts the lengths of the local fronts radii $\{R_i, R_j\}$, based on the confidence we have (w_i and w_j, see Appendix F) about the local front estimates ($m_i(t_b)$ and $m_j(t_b)$).

Using equations (6.4) and (6.5) we calculate the ``new'' radii R_i' and R_j' of the local fronts' circular areas:

$$\begin{cases} R_i' = R_i - \frac{c}{w_i} \\ R_j' = R_j - \frac{c}{w_j} \end{cases} \tag{6.4}$$

$$c = \frac{w_i w_j (R_i + R_j - ||P_i^M(t_b)P_j^M(t_b)||)}{w_i + w_j} \tag{6.5}$$

The equations in (6.4) indicate that the decrease of the initial radii $\{R_i, R_j\}$, is inversely

proportional of the corresponding local fronts' confidence values (weights $\{w_i, w_j\}$ see Appendix F). Equation (6.5) consists the solution of the system of equations (6.3) and (6.4).

Using the new radii R'_i and R'_j, we calculate the end points coordinates $P_i^{E_z}(t_b) = (x_i^{E_z}(t_b)), y_i^{E_z}(t_b))$ where $z = \{i, j\}$ of the local front estimates (see Appendix D), update the corresponding information in $\mathcal{M}_b$ and repeat the third step form the beginning. The third step completes when there are no overlaps between the circular areas of the local front estimates in $\mathcal{M}_b$. When the *third step* completes, we initiate the *boundary reconstruction algorithm*.

6.3 Boundary Reconstruction

In this section we present the proposed algorithm which using the information of the local front estimates in $\mathcal{M}_b$, it determines the boundary of the continuous object. The algorithm has to two phases:

- At the *first phase* it determines a polygon that approximates the continuous object's boundary.
- At the *second phase* it produces a ``smooth'' representation of the boundary, based on the uniform B-spline curves.

To better explain the two phases of the algorithm, we will use a running example: Let's assume that $\mathcal{M}_b$ contains 14 local front estimates (see Figure 6.8). In Figure 6.8 the black arrows indicate the evolution directions of the local front segments and the black dots their corresponding middle and end points.

6.3.1 First Phase: Polygonal Approximation of the Boundary

This procedure has two steps: The *first step* determines the order in which the local front estimates in $\mathcal{M}_b$ have to be connected, to form the polygon that approximates the continuous object's boundary. The *second step* uses the local fronts connection order and based on their parameters it determines the boundary's polygon.

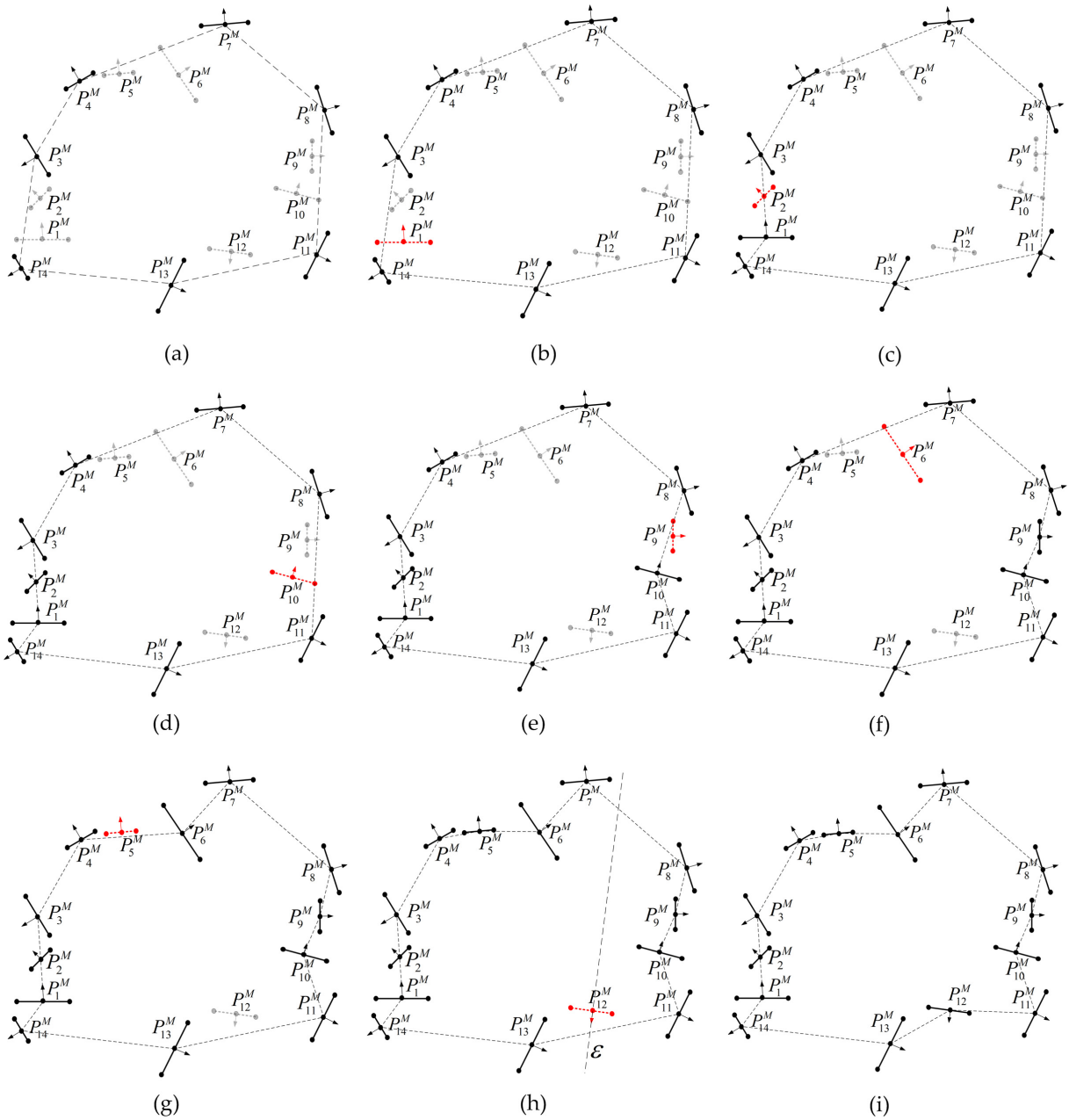


Figure 6.8: First step: Finding the connection sequence of the local front segments.

To produce a realistic approximation of the continuous object's boundary, the formed polygon should satisfy the following conditions:

- It should be simple (no intersections between its edges)
- All the local fronts' evolution direction vectors should point outside the polygon's area (the diffusive phenomena increase their size with time)

Before initiating the first step, we form the following sets:

$\mathcal{C}$: It contains the middle points P_i^M of the local front estimates in $\mathcal{M}_b$.

$\mathcal{V}$: It contains the middle points of the local front estimates that belong on the convex hull of $\mathcal{C}$ (in our example, $\mathcal{V} = \{P_3^M, P_4^M, P_7^M, P_8^M, P_{11}^M, P_{13}^M, P_{14}^M\}$, see Figure 6.8a). It holds that $\mathcal{V} \subseteq \mathcal{C}$.

$\mathcal{Q} = \mathcal{C} \cap \mathcal{V}^c$: It contains the middle points in $\mathcal{C}$ that do not belong to the convex polygon. In Figure 6.8a, $\mathcal{Q} = \{P_1^M, P_2^M, P_5^M, P_6^M, P_9^M, P_{10}^M, P_{12}^M\}$. We have to note that $\mathcal{C} = \mathcal{V} \cup \mathcal{Q}$.

First step

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

- If $\{\mathcal{Q} = \emptyset\}$: We continue to the *second step*, since all the local fronts middle points are contained in $\mathcal{V}$. The order of the middle points in $\mathcal{V}$, determines the order in which the local front segments in $\mathcal{M}_b$ have to be connected to form the polygon that approximates the continuous object's boundary.
- If $\{\mathcal{Q} \neq \emptyset\}$ (example's case): We repeat the following procedure until $\mathcal{Q}$ becomes an empty set:

Using the polygon determined by $\mathcal{V}$ (e.g. see convex polygon in Figure 6.8a), we find the relative positions (on, inside, outside) of the middle points that contained in $\mathcal{Q}$, and form the following sets:

$\mathcal{Q}_{on}$: It contains the middle points that lies *on* the polygon's edges.

$\mathcal{Q}_{out}$: It contains the middle points located *outside* the polygon's area.

$\mathcal{Q}_{in}$: It contains the middle points located *inside* the polygon's area.
We have to note that $\mathcal{Q} = \mathcal{Q}_{on} \cup \mathcal{Q}_{out} \cup \mathcal{Q}_{in}$.

After forming these sets we check:

If $\{\mathcal{Q}_{on} \neq \emptyset\}$: We select a point from $\mathcal{Q}_{on}$ (e.g. P_2^M in Figure 6.8c), determine the polygon edge on which it lies (e.g. $\overline{P_1^M, P_3^M}$ in Figure 6.8c) and apply an edge split. An edge split is the procedure where a point from $\mathcal{Q}$ becomes a vertex of the polygon (moves to $\mathcal{V}$). In our example the point P_2^M (Figure 6.8c) becomes a polygon vertex moves to $\mathcal{V}$, between points P_1^M, P_3^M that determine the edge on which it lies. After an edge split we delete the ``new" polygon vertex (P_2^M) from $\mathcal{Q}$. The aforementioned procedure is repeated until $\mathcal{Q}_{on}$ becomes an empty set.

If $\{\mathcal{Q}_{on} = \emptyset$ and $\mathcal{Q}_{out} \neq \emptyset\}$: We use the points in $\mathcal{Q}_{out}$ and form the following sets:
$\mathcal{Q}_{out}^I$: It contains the middle points of the local front segments that intersect at least one edge of the polygon (e.g. P_9^M in Figure 6.8e).
$\mathcal{Q}_{out}^T$: It contains the middle points of the local front segments that are located totally outside the polygon (e.g. P_5^M in Figure 6.8g).
It holds that $\mathcal{Q}_{out} = \mathcal{Q}_{out}^I \cup \mathcal{Q}_{out}^T$.

After forming these sets we check:

If $\{\mathcal{Q}_{out}^I \neq \emptyset\}$: We find the middle point in $\mathcal{Q}_{out}^I$ which has the smallest distance from the polygon's edge that intersected by the corresponding local front segment (e.g. P_9^M in Figure 6.8e). Using this point we split the intersected edge (see $\overline{P_8^M P_{10}^M}$ in Figure 6.8f) and update the sets $\mathcal{V}$ and $\mathcal{Q}$. After an edge split we repeat the *first step* from the beginning.

If $\{\mathcal{Q}_{out}^T \neq \emptyset\}$: We calculate for each middle point in $\mathcal{Q}_{out}^T$, the corresponding minimum distances from all the polygon's edges. In sequence, we find the middle point which has the smallest distance from a polygon's edge, split the corresponding edge and update the sets $\mathcal{V}$ and $\mathcal{Q}$. After an edge split we repeat the *first step* from the beginning.

If $\{\mathcal{Q}_{on} = \emptyset$ and $\mathcal{Q}_{out} = \emptyset$ and $\mathcal{Q}_{in} \neq \emptyset\}$: We use the points in $\mathcal{Q}_{in}$ and form the following sets:
$\mathcal{Q}_{in}^I$: It contains the middle points of the local front segments that intersect at least one edge of the polygon (e.g. $\{P_1^M, P_6^M, P_{10}^M\}$ in Figure 6.8a).
$\mathcal{Q}_{in}^T$. It contains the middle points of the local front segments that are located totally inside the polygon (e.g. $\{P_2^M, P_{12}^M\}$ in Figure 6.8h).

It holds that $\mathcal{Q}_{in} = \mathcal{Q}_{in}^{I} \cup \mathcal{Q}_{in}^{T}$.

After the formation of these sets we check:

If $\{\mathcal{Q}_{in}^{I} \neq \emptyset\}$: We find the middle point in $\mathcal{Q}_{in}^{I}$ (e.g. $P_1^M(t_b)$, in Figure 6.8a) which has the smallest distance from the polygon's edge that intersected by the corresponding local front segment. Using this point we split the intersected edge (see $\overline{P_2^M, P_{14}^M}$, in Figure 6.8c), and update the sets $\mathcal{V}$ and $\mathcal{Q}$. After an edge split we repeat the *first step* from the beginning.

If $\{\mathcal{Q}_{in}^{T} \neq \emptyset\}$: We derive for each point $\mathcal{Q}_{in}^{T}$ the equation of the line (e.g. line ε in Figure 6.8h) that emanates from it and is perpendicular to the corresponding local front segment. For each of these lines, we determine their intersection points with the polygon (see points A,B in Figure 6.8h). In sequence, for each local front segment we select the point that belongs to its evolution direction (point A in Figure 6.8h). We have to note that if there are more than one intersection points at its evolution direction (e.g. this case may hold only for non-convex polygons), the procedure selects the point which has the smaller Euclidean distance from the local front's middle point. In sequence, we find the middle point with the the smaller distance from its corresponding intersection point, and split the polygon's edge on which it lies (e.g. $\overline{P_{11}^M P_{13}^M}$, in Figure 6.8h). After the edge split we update the sets $\mathcal{V}$, and $\mathcal{Q}$ and repeat the *first step* from the beginning.

At the end of the first step, the ordered set $\mathcal{V}$, uniquely determines a polygon (see Figure 6.8i) that indicates the local front's connection sequence.

Second step

This step uses the local fronts' middle points in $\mathcal{V}$ and the local fronts' end points (contained in $\mathcal{M}_b$) and constructs a more detailed polygonal approximation of the continuous object's boundary.

Based on the order of the local fronts' middle points in $\mathcal{V}$, we sort the local fronts in $\mathcal{M}_b$. Next, using the coordinates of the end points of the first two local front segments in M_b ($\{P_1^{E_1}, P_1^{E_2}, P_2^{E_1}, P_2^{E_2}\}$ in Figure 6.9a) we find the minimum distance pair, excluding the pairs that belong to the same local front segment, and connect them (see red dashed edge $\overline{P_1^{E_1}, P_2^{E_2}}$ in Figure 6.9a). In sequence, using the non-connected end point of the second local front segment ($P_2^{E_1}$) 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 the points (see edge $\overline{P_2^{E_1}, P_3^{E_1}}$ in Figure 6.9a).

Applying the aforementioned method for all the local front segments in $\mathcal{M}_b$, we construct a polygon which has as vertices the middle and the end points of the local front segments (see Figure 6.9b).

The aforementioned construction method does not guarantee that the formed polygon will satisfy boundary's polygon conditions that discussed in *Phase I*. To guarantee that the formed polygon will satisfy these conditions, we do the following: For each local front segment we check: a) if the edges that connects it with its adjacent local fronts segments intersect with other polygon's edge(s) and b) if its evolution direction vector points inside the polygon's area. If at least one of these conditions is satisfied (e.g. in Figure 6.9b the direction vectors of m_1 and m_{10} points inside the polygon), we interchange the corresponding local front segments end points connections (see Figure 6.9c). In case where after a connection interchange, the aforementioned conditions are still not satisfied, we erase the corresponding local front estimate from $\mathcal{M}_b$ and we form a new polygon by connecting the end points of its two adjacent local fronts segments.

The polygon formed at the end of this step is passed as input to the *smooth boundary reconstruction* procedure (described below).

6.3.2 Second Phase: Smooth Boundary Reconstruction using Uniform Cubic B-Splines

This phase uses the uniform B-spline [102] and generates a smooth curve that approximates the continuous object's boundary.

A spline is a smooth polynomial function that is piecewise defined and consist a convenient form for representing complicated, smooth curves. Due to the simplicity of their construction and to their capacity to approximate complex shapes with accuracy, spline curves have been extensively used in the fields of computer graphics, image analysis and CAD (marine, aeronautical, automobile etc.).

Basis splines (B-splines) are the most commonly used spline curves since they trade-off complexity and flexibility [102]. An elegant way to describe a B-spline curve is presented below:

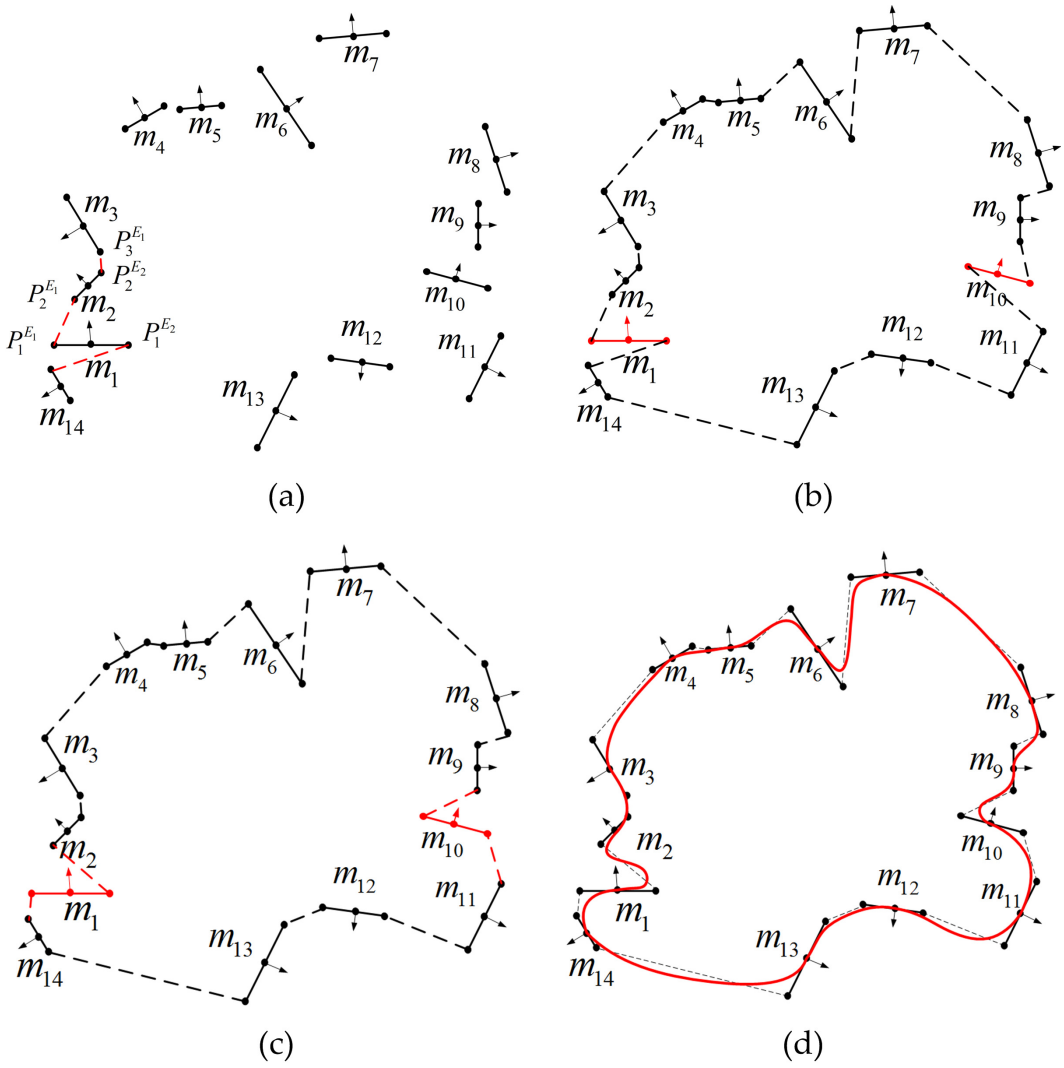


Figure 6.9: a) The middle points of each local front is connected with its end points (formation of black segments); the red dashed lines show the first steps of local front segments connection. b) First polygonal boundary approximation (note that the direction vectors of m_1 and m_{10} points inside the polygon) c) The polygonal approximation that satisfies the boundary's polygon conditions (direction vectors of m_1 and m_{10} points outside the polygon after interchanging the connections of their end points.) d) The smooth boundary construction using a B-spline curve (red curve).

$$X(t) = \sum_{i=0}^{n} P_i N_{ik}(t) \tag{6.6}$$

Where $N_{ik}(t)$ describes the basis functions of the B-spline. The basis functions are computed using the *Cox de Boor* algorithm.

$$N_{ik}(t) = \begin{cases} 1 & if \;\; t_i \leq t \leq t_{i+1} \\ & \qquad\qquad k = 1 \\ 0 & otherwise \end{cases} \tag{6.7}$$

$$N_{ik}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) k > 1 \tag{6.8}$$

From equations (6.6), (6.7), (6.8) it is obvious that to construct a B-spline curve we need the following information:

The B-splines degree k.

A vector $P = \{P_0, ..., P_n\}$ of size $n + 1$ that contains the B-spline's control points (polygon vertices in our case).

A vector $T = \{t_0, t_1 ... t_{n+k+1}\}$ of size $n + k + 2$ that contains the knots of the B-spline curve. The knot points divide a B-spline curve into curve segments each of which is defined on a knot span. These curve segments are all Bezier curves of degree k. If we want to define a B-spline curve of degree k with $n + 1$ control points we have to supply $n + k + 2$ knots. Using the constraint of equally spacing knot points ($t_{i+1} - t_i = constant$) we obtain the uniform B-spline curves.

To produce a smooth representation of the continuous object's boundary we selected the uniform cubic B-spline curves (degree $k = 3$). These curves uses cubic functions (3rd degree polynomial) to represent each curve segment and constraints the points that joint the curve segments to meet the following requirements:

1. Positional Continuity (C^0): i.e. the end point of segment i is the same as the starting point of segment $i + 1$.

2. Tangential Continuity (C^1): i.e. no abrupt change in slope occurs at the transition between segment i and segment $i + 1$.

3. Curvature Continuity (C^2): i.e. no polarity changes in slope at the transition between segment i and segment $i + 1$.

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 Figure 6.9a). This selection guaranties that the formed cubic spline will be tangent to middle points of the local fronts segments. The red curve in Figure 6.9d corresponds to the formed B-spline curve that approximates the polygonal boundary of the continuous object. To draw a closed spline curve curve, we add at the end of the the control points set (vertices set $\mathcal{V}$) the first 3 polygon's vertices (e.g. $\{P_1^{E_1}, P_2^{E_1}, P_2^{M}\}$).

As discussed in Section 6.1.1, the evolution characteristics (orientation and speed) of the local fronts m_i that participate to the boundary reconstruction algorithm are described by scalar values $\{\hat{\phi}_i, \hat{u}_i\}$ which are randomly sampled from the corresponding Normal distributions (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 the same boundary shapes. Based on this observation, we developed a technique which allows us to calculate a spatial probability field that indicates for each point of the continuous object's evolution area the probability to be affected by the continuous object. To calculate the spatial probability field we do the following:

We consider the continuous object's evolution area as a grid of squares cells. A cell is assumed to be affected by the evolving continuous object, when its center is located inside the continuous object's area. To calculate the probability for a cell C to be affected by the continuous object, we run the boundary reconstruction algorithm N times and we count the number N_C of simulations scenarios that cell C contained inside the object's area. The probability for the continuous object to arrive at the cell C can be estimated as:

$$P(C) = \frac{N_C}{N}. \tag{6.9}$$

6.4 Evaluation Setup

We present next simulation results demonstrating the ability of the proposed algorithm to track accurately the boundary of a diffusive phenomenon under different number of local front estimates, local front parameters estimation errors and continuous object evolution

scenarios.

6.4.1 Simulation Workflow

For the evaluation of the proposed boundary tracking algorithm we developed a flexible simulator in Matlab which allows us to generate scenarios with different a) propagating diffusive object's properties (shape, speed and acceleration), b) local fronts estimates densities, deployment strategies and evolution parameters errors. Before initiating a simulation, a Matlab procedure takes as input the coordinates of the local fronts segments middle points, and the diffusive object's propagating properties, and determines: a) the time instances where the diffusive object reaches the local fronts' middle points (local fronts' estimation times), and b) the speed and orientation of the diffusive object's front line at the middle points locations (see Appendix G for details). Using these speed and orientation values, the Matlab procedure determines the evolution characteristics (speed and direction) of the deployed local fronts segments, after distorting them with error (its value is determined by the user). The distorted speed and orientation values consist the mean values of the corresponding local fronts' normal distributions (U_i, Φ_i see Section 6.1.1). The standard deviations of the corresponding normal distributions are also determined by the user. Using this information, the Matlab procedure generates a structure that contains for each local front estimate, the parameters presented in Section 6.1.1. The generated local fronts structure is passed as input along with the propagating properties of the diffusive object to the Matlab simulator which initiates the evaluation of the proposed boundary tracking algorithm accuracy.

6.4.2 Evaluation Metrics

To evaluate the accuracy of the proposed algorithm, we compare the ``similarity'' of the areas occupied by the real and constructed boundaries as described below:

We consider the diffusive hazard's evolution area as a grid of square cells (see Figure 6.10). For all the conducted experiments we set the size of the cells equal to $20m \times 20m$. Based on the real and reconstructed boundary of the continuous object, we classify the square cells to the following categories:

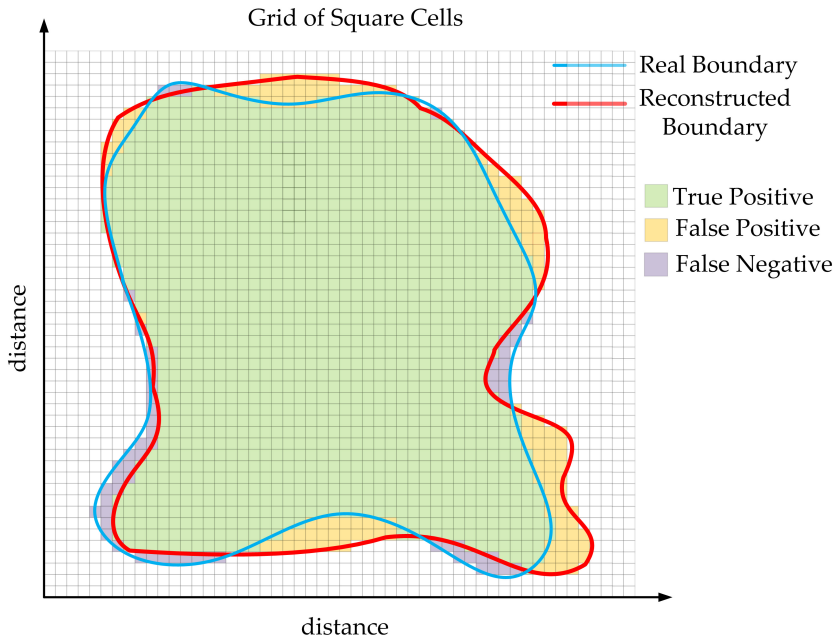


Figure 6.10: The blue (red) curve correspond to the reconstructed (estimated) boundary of the diffusive phenomenon. Classifying the square cells of the gridified area (as shown in the figure), we can evaluate the accuracy of the proposed boundary reconstruction algorithm using the F1-score metric.

- True Positive (TP): Contains the square cells that located inside the actual and reconstructed boundary.

- False Positive (FN): Contains the square cells that located outside the actual and inside the reconstructed boundary.

- False Negative (FN): Contains the square cells that located inside the actual and outside the reconstructed boundary.

To calculate the boundary reconstruction accuracy we used the F1-score which is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{6.10}$$

where,

$$precision = \frac{TP}{TP + FP}, \quad recall = \frac{TP}{TP + FN}. \tag{6.11}$$

6.4.3 Experimental Setup

A notable advantage of the proposed continuous object boundary tracking algorithm is that it can track with accuracy the boundary of a diffusive phenomenon using a small number of local front estimates. To demonstrate this feature we have used local fronts' densities that are considered low for environmental monitoring applications. Specifically we used densities $\{10^{-6}, 2\times10^{-6}, 3\times10^{-6}, 4\times10^{-6}\}\ \frac{local\ fronts}{m^2}$, which correspond to $10, 20, 30$ and 40 local front estimates deployed within a $1km^2$ square area respectively. For each density value we used a large number of randomly drawn local fronts' deployments and demonstrate how the algorithm performs under different densities as well as errors to the local fronts evolution parameters (speed and direction). For all the experiments the length of the local fronts segments was set to $l = 100m$. The evolution parameters (speed, orientation angle and direction) of the local estimates were set equal to the true values that had the diffusive phenomenon when it reached them (perfect estimation, see Appendix G). However, to evaluate how the algorithm performs under local fronts' orientation and speed estimation errors, we repeated 6 times the experiment and tested the accuracy of the boundary construction. Specifically the error values used for 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.

6.5 Results and Discussion

In the conducted experiment the spatiotemporal evolution of the continuous object was simulated using either the developed Matlab program or FLogA a wildfire behavior simulator developed in our group [87].

6.5.1 Experiment 1: diffusive hazard with regular shape

In this experiment the continuous object is modeled as a circle of fixed center located at the center of a $2km \times 2km$ square area. Considering the area's bottom left corner as the origin, the circle is centered at point $(1000m, 1000m)$ and has radius equal to $1m$. The speed at which the radius is increasing is described by a triangular function with

initial value $2.5m/min$. The speed increases with time varying rate $(0.0265m/min^2)$ until it reaches its maximum value $(5m/min)$ when the radius of the circle becomes $250m$. Beyond that point the speed starts decreasing at the same rate until it returns back to its initial value. Therefore when the radius of the circle becomes equal to $500m$ its speed is again back to $(2.5m/sec)$. At the end of a simulation the circle becomes circumscribed to the $1km^2$ local fronts' deployment square area (centred at point $(1000m, 1000m)$). Modeling propagating hazard with circular shape is justified 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 [89]. To help the reader visualize the phenomenon and get sense of the boundary tracking algorithm that takes place during the evolution of the diffusive object, we provide a video animation (see Experiment1RegularFront.mov [103]) created using Matlab.

To evaluate the boundary tracking accuracy of the proposed algorithm we attempt to construct the boundary of the diffusive phenomenon 20 times during its evolution at equally spaced (every $188/20min$) sequential time intervals. The boxplots in Figure 6.11a summarize the distribution of the boundary tracking accuracy considering 1000 random deployments (1000 runs) per local fronts estimates density scenario. For the generation of each boxplot we used as sample points the F1-scores estimated at the time instants where a boundary construction has occurred. As we observe from the provided boxplots the boundary construction accuracy increases with the local fronts estimates density. Another interesting observation is that the standard deviation of the boundary construction accuracy decreases as the density of local estimates increases. This implies that as the number of local estimates increases we become more certain that the proposed algorithm will produce an accurate representation of the diffusive object's boundary.

Figure 6.11b shows the boundary tracking accuracy (F1-score) as a function of time for different local front estimates density scenarios. As shown in Figure 6.11b, the accuracy of the boundary tracking drops for a certain time interval and then recovers again. The time extent and experienced accuracy drop increases as the number 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

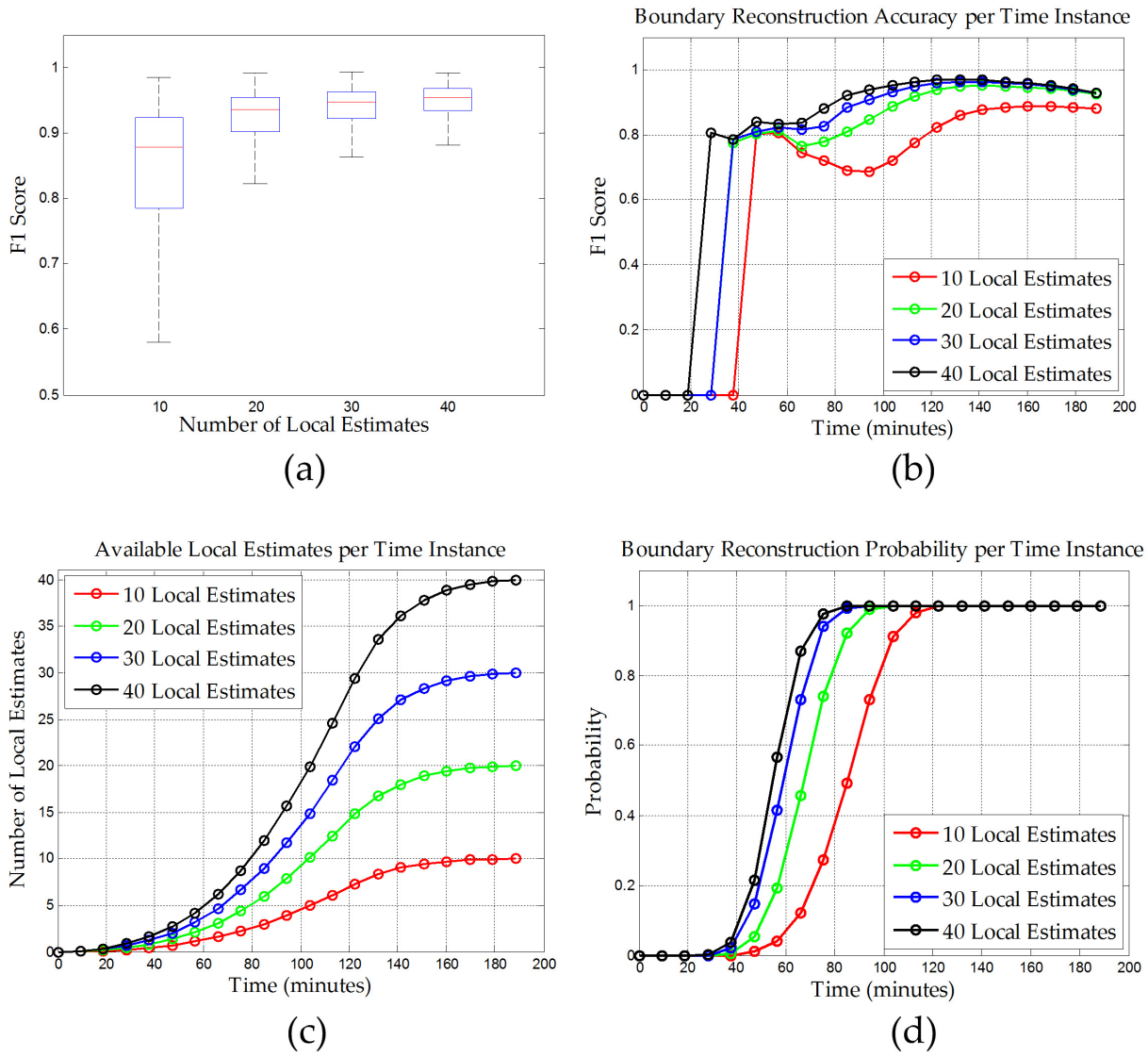


Figure 6.11: Experiment 1: a) Boxplots summarizing the distribution of the boundary tracking accuracy under different local estimates density scenarios. b) The boundary tracking accuracy as a function of time for different densities of local estimates. c) The mean number of the available local estimates that participate at each time instance to the boundary construction algorithm, for different local estimates density scenarios. d) The boundary construction probability as a function of time for different local estimates density scenarios.

the diffusive phenomenon gets bigger with time the accuracy of the boundary construction drops until more local estimates (sufficient number) become available to participate in the boundary construction algorithm. In Figure 6.11c we see that as the local estimates density increases, the number of the available local estimates to participate to the boundary tracking algorithm at each time step of the diffusive phenomenon evolution also increases. This larger number of available local estimates explains why the time period where the accuracy drops are smaller at larger local estimates densities. Finally, the small decreasing trend of the boundary tracking accuracy observed beyond $140min$ is explained if we consider that up to this time instant most local estimates have become available (see Figure 6.11c) and therefore the construction of the boundary beyond that point is based mostly on ``old'' local front estimates.

Figure 6.11d shows the probability to have a boundary construction event at each one of the considered time instances. We observe that as the local estimates density increases, the probability for a boundary construction event also increases for all time instants. This behavior is justified if we consider that the higher local estimates density implies more available local estimates at each time instant which in turn increases the probability for a boundary construction to occur.

Boundary tracking evaluation under orientation and speed estimation errors

In this experiment we evaluate the boundary tracking accuracy of the proposed algorithm under different orientation and percent speed error values. For each scenario (specific density and error values) the evolution parameters (orientation or speed) of each local estimate are distorted by the selected error values (see Section 6.4.3). The distortion occurs by adding or subtracting (randomly selected operation) the selected error value from the corresponding parameter.

Figures 6.12a and 6.12b show for each angle and percent speed error value respectively, the boundary construction accuracy for all the local estimates density scenarios considered. For each case (e.g. 20 local estimates, 10 degrees error) the results are generated using as sample points the F1-score values estimated from all the boundary constructions occurred using 1000 different (randomly selected) local estimates deployments. The line plots in Figures 6.12a and 6.12b indicate that as the angle and speed error

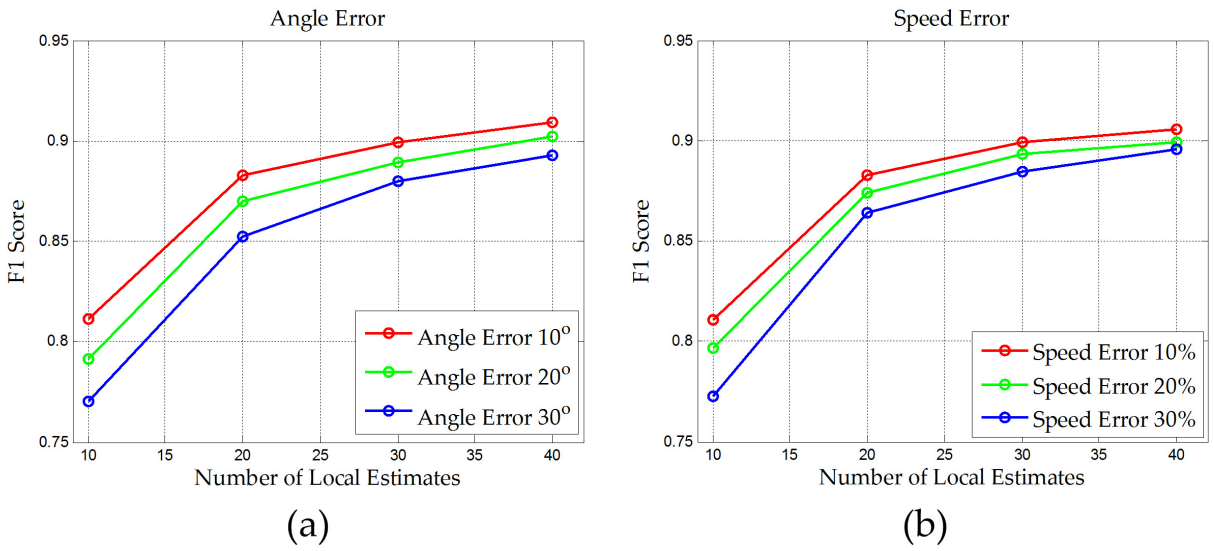


Figure 6.12: Experiment 1: Boundary tracking accuracy under different local estimates densities and (a) orientation errors, (b) speed errors. For all the orientation (speed) error scenarios the speed (orientation) error was set to 0 in order to focus on how the orientation (speed) errors affect the boundary tracking accuracy.

values increase, the boundary construction accuracy decreases. Another interesting observation is that the difference, between the boundary construction accuracies achieved for each speed or percent speed error value, reduces as the number of local estimates increases. This behavior is justified if we consider that for larger densities, the mean distance that the local fronts segments (that participate to the boundary construction) have to travel in order to construct the diffusive objects boundary is smaller (they continuously replaced from more recent estimates). Traveling smaller distances with erroneous evolution parameters implies smaller boundary location errors and therefore better boundary construction accuracy.

Boundary tracking evaluation under orientation and speed estimation uncertainties

In this experiment we evaluate for each local fronts' density scenario, how the boundary's tracking accuracy is affected under different orientation and speed uncertainties (stds). For each local fronts' density scenario we used the deployment that scored the median boundary tracking accuracy from the 1000 considered random deployments. The orientation and speed uncertainty values pairs used were $\{10^o, 10\%\}$, $\{20^o, 20\%\}$ and $\{30^o, 30\%\}$. For each density and uncertainty value pair (e.g. 20 local fronts, $\{10^o, 10\%\}$) we attempt

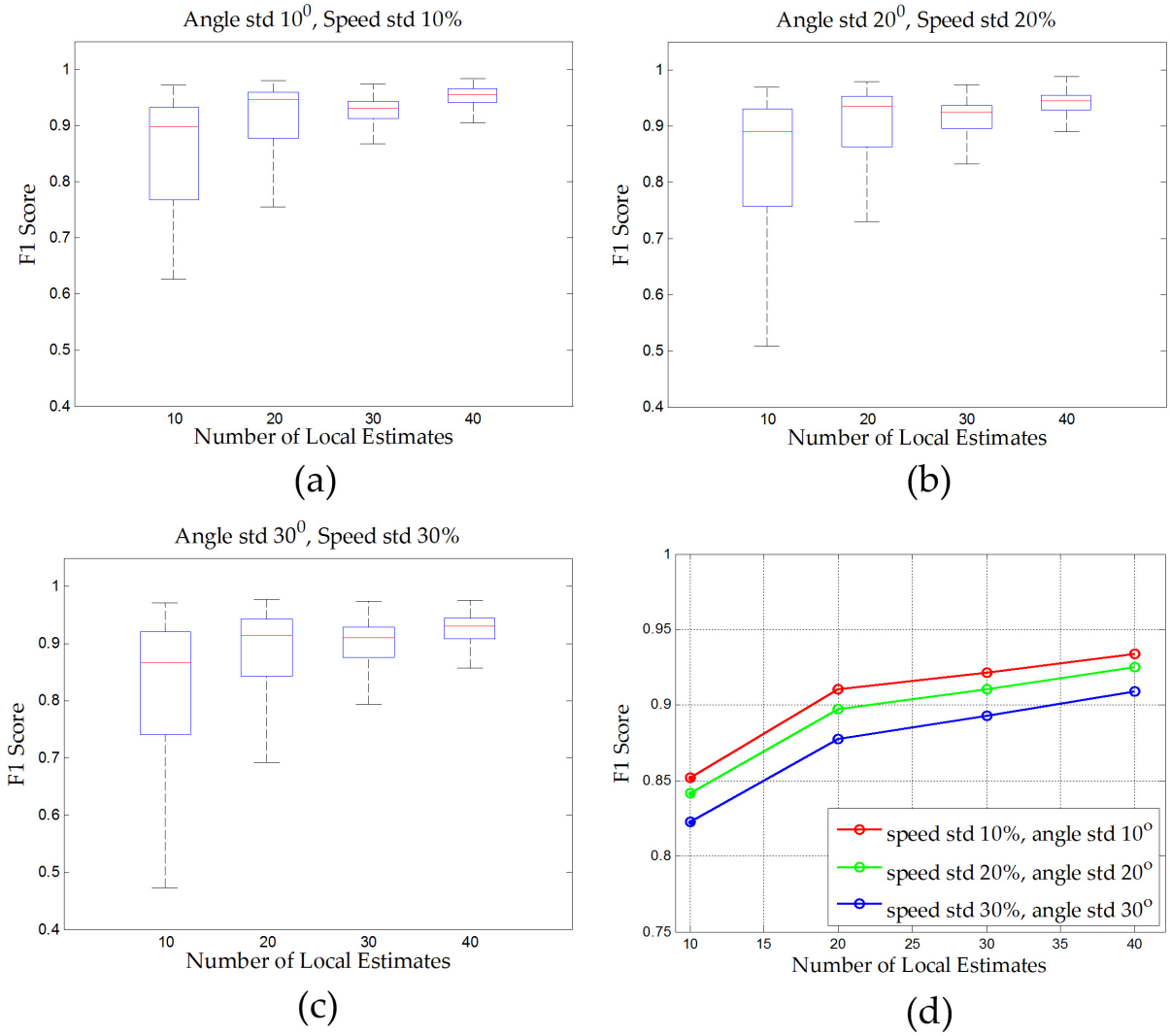


Figure 6.13: Experiment 1: Boxplots summarizing for the uncertainty pairs of values a) $\{10^0, 10\%\}$ b) $\{20^o, 10\%\}$, c) $\{30^o, 30\%\}$ the distribution of the boundary tracking accuracy under different local estimates density scenarios. d) The mean boundary tracking accuracy under different local estimates density scenarios and uncertainty pairs of values.

to track the boundary 1000 times, where at each time, the evolution parameters (orientation $\hat{\phi}_i$ and speed $\hat{u}_i$) of each local front m_i, were randomly drawn from its corresponding normal distributions (U_i, Φ_i).

Figures 6.13a, 6.13b, 6.13c summarize for each uncertainty value pair, the distribution of the boundary's tracking accuracy, for all the considered local fronts densities scenarios. For the generation of each boxplot we used as sample points the F1-scores estimated at the time instants where a boundary construction has occurred. As we observe from the provided boxplots the boundary tracking accuracy increases with the sensors density

for all the evolution parameters uncertainty scenarios. Another interesting observation is that the standard deviation of the boundary tracking accuracy decreases as the density of sensors increases. This implies that as the number of sensors increases we become more certain that the proposed algorithm will produce an accurate representation of the diffusive object's boundary.

Figure 6.13d shows for each pair of evolution uncertainty parameter values, the mean boundary tracking accuracy for all the local fronts densities scenarios considered. For each case (e.g. 20 local fronts, $\{10^o, 10\%\}$) the results are generated using as sample points the F1-score values estimated from all the boundary constructions occurred using 1000 runs. The line plots in Figures 6.13d indicate that as the orientation and speed uncertainty values increase, the boundary tracking accuracy decreases. Another interesting observation is that the difference, between the boundary tracking accuracies achieved for each orientation and speed uncertainty values pairs, reduces as the number of local estimates increases. This behavior is justified if we consider that for larger local estimates densities, the mean distance that the local front segments (that participate to the boundary construction) have to travel (space-time projection) in order to construct the diffusive phenomenon boundary is smaller. Traveling smaller distances with erroneous evolution parameters implies smaller boundary location errors and therefore better boundary construction accuracy.

6.5.2 Experiment 2: diffusive hazard with irregular shape

In this experiment 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 generate hazards realistic characteristics we use FLogA [87], a web-based interactive tool (developed in our group) which allow us to draw a forest area anywhere in Europe over Google Earth [90], insert fire ignition points (``hotspots''), define wind direction and speed scenarios, and then simulate and geo-animate the evolving wildfire.

Using FLogA we define a square forest area $3km \times 3km$ at Hymettus mountain in Attica Greece and generate a wildfire scenario. Considering the area's bottom left corner

as the origin, the fire ignite at point $(1500m, 1500m)$. The wind speed and direction parameters were fixed within the forest area and their values were set to $2m/s$ (light breeze) and 0^o (with respect to the x axis) respectively. Similarly to Experiment 1, we evaluate the proposed boundary reconstruction algorithm under different: a) local fronts estimates densities ($\{10, 20, 30, 40\}$ local fronts estimates per $1km^2$), b) local fronts' deployments (1000 random deployments), c) local fronts orientation and percent speed errors scenarios ($\{10^o, 20^o$ and $30^o\}$ degrees and $\{10\%, 20\%$ and $30\%\}$) and d) orientation and percent speed estimation uncertainties scenarios $\{10^o, 10\%\}$, $\{20^o, 20\%\}$ and $\{30^o, 30\%\}$.

We summarize the simulation results in Figures 6.14 - 6.16. Their interpretations are similar with these of Figures 6.11 - 6.13 of experiment 1. In this experiment the mean boundary reconstruction accuracy (F1-scores) was on average smaller by 3.91 percent (standard deviation 1.15 percent) relatively to Experiment's 1 considering the results from all simulation scenarios. This accuracy decrease is justified if we consider the complex evolution behavior of the wildfire's front line (irregular accelerations/decelerations, see also the video provided in [104]) which makes the boundary tracking more difficult.

We presented an algorithm which is able to reconstruct with accuracy the boundary of evolving continuous object using a small number of estimates that describe locally the object's evolution characteristics. The algorithm exploits the uncertainty about the estimated evolution parameters and generates the probability for each point of the considered area to be affected by the continuous object. Extensive simulation results demonstrate that the proposed algorithm is able to track accurately the boundary of different types of continuous objects (e.g. time-varying evolution rates and/or irregular boundary shapes), while using a small number of local fronts estimates which may be distorted with error.

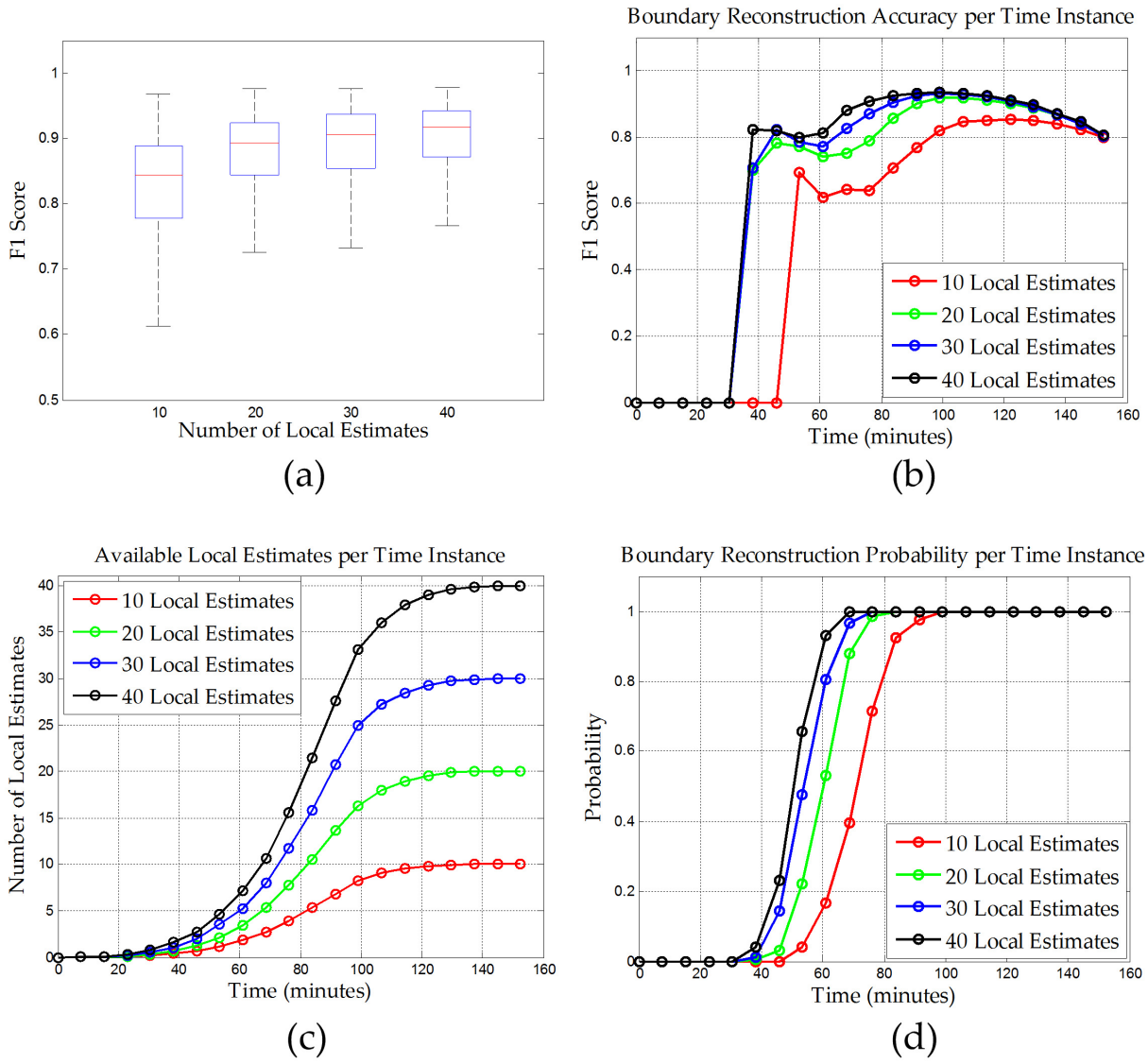


Figure 6.14: Experiment 2: a) Boxplots summarizing the distribution of the boundary tracking accuracy under different local estimates density scenarios. b) The boundary tracking accuracy as a function of time for different local estimates densities. c) The mean number of the available local estimates that participate at each time instance to the boundary construction algorithm, for different local estimates density scenarios. d) The boundary reconstruction probability as a function of time for different local estimates density scenarios.

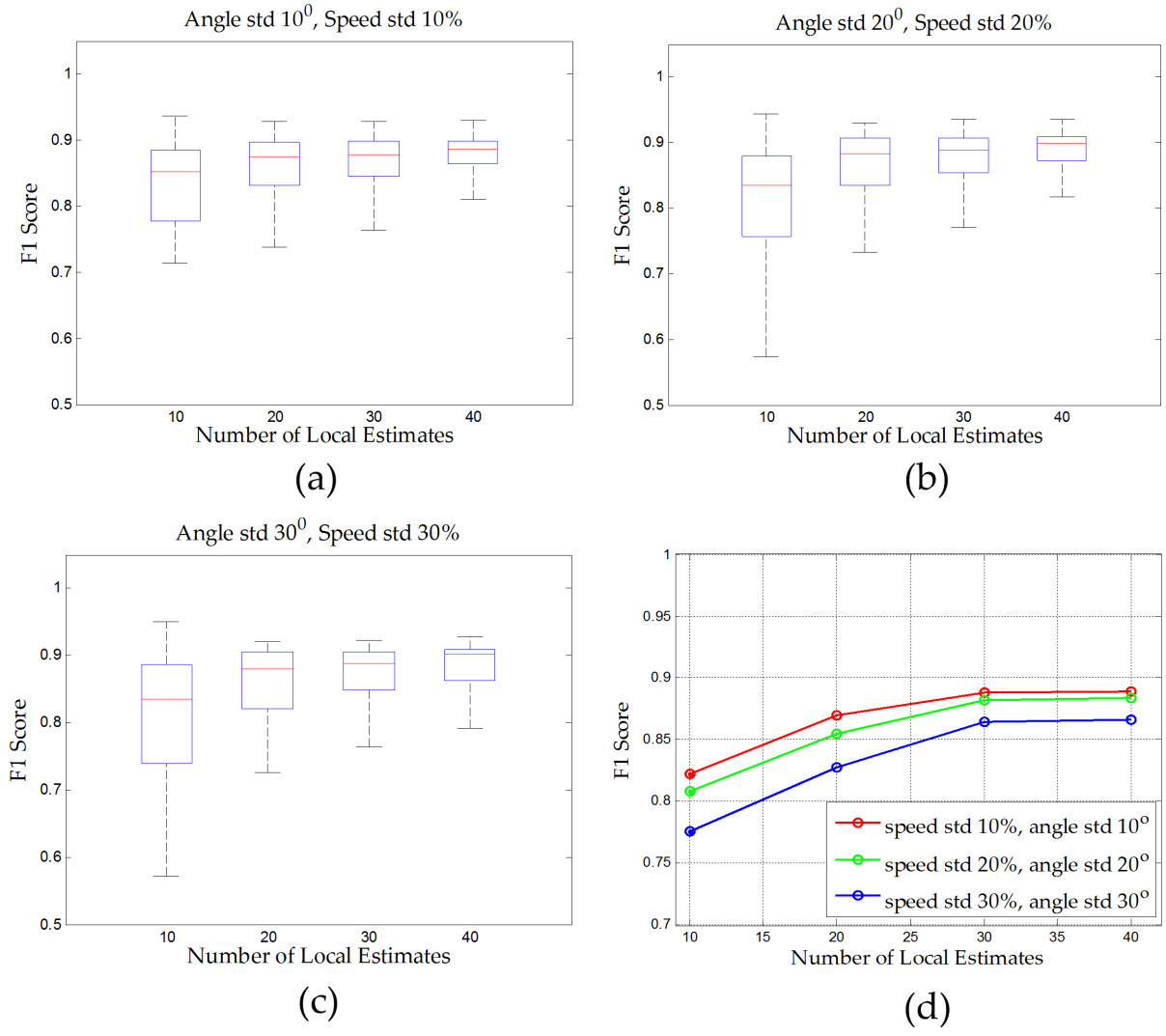


Figure 6.15: Experiment 2: Boundary tracking accuracy under different local estimates densities and (a) orientation errors, (b) speed errors. For all the scenarios orientation (speed) errors scenarios the speed (orientation) error was set to 0 in order to focus on how the orientation (speed) errors affect the boundary tracking accuracy.

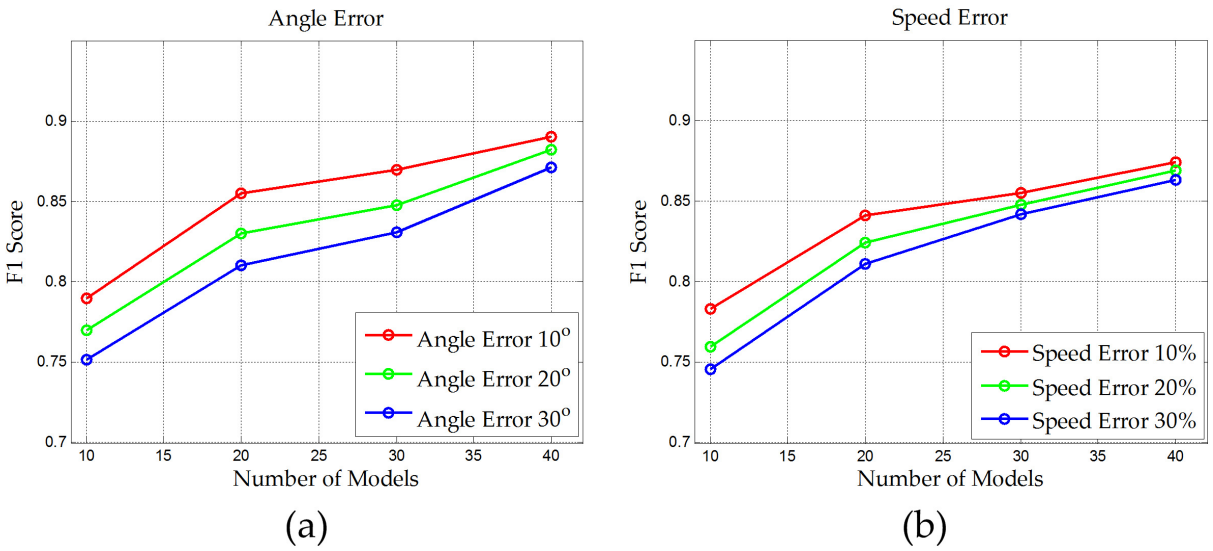


Figure 6.16: Experiment 2: Boxplots summarizing for the uncertainty pairs of values a) $\{10^0, 10\%\}$ b) $\{20^o, 10\%\}$, c) $\{30^o, 30\%\}$ the distribution of the boundary tracking accuracy under different local estimates density scenarios. d) The mean boundary tracking accuracy under different local estimates density scenarios and uncertainty pairs of values.

Chapter 7

Conclusions and Further Research Directions

During the last decade there has been a fast growing interest on developing efficient systems for tracking and predicting the evolution of hazards. Recently, a large number of WSN-based schemes have been proposed for tracking the boundaries of continuous objects [22--40]. However, all the proposed schemes suffer from severe limitations (see Chapter 1) that make them impractical for real world applications. In this doctorate dissertation we have presented a continuous object tracking scheme aimed at addressing these limitations. We summarize below the contributions of this work:

Probabilistic sensing mechanism modeling in presence of hazards (Chapter 3)

We proposed a flexible probabilistic sensing modeling approach which in contrast with the existing works that assume a perfect sensing mechanism (see L4 in Chapter 1), can capture the detection distance uncertainty and the possibility for a sensor node to malfunction in a harsh environment created by an approaching hazard. This simple, yet realistic, sensing model which was inspired by the analysis of real sensing data collected from two outdoor experimental burns that (conducted at Gestosa's experimental field site in Portugal [78]), allow us to formulate a local front models' parameters estimation problem in a Bayesian manner. We analytically solved this Bayesian problem and derived closed-form algebraic expression that can be easily implemented by microprocessors of the commodity sensor nodes (see Chapter 5).

Collaborative WSN algorithm for estimating the spatiotemporal evolution characteristics of local fronts of continuous object (Chapters 4)

To address limitations L1, L3, L4 and L5 (presented in Chapter 1) of the state of the art schemes, we developed an *asynchronous* collaborative algorithm that is able, using WSNs of *realistic* density, to estimate with accuracy the spatiotemporal evolution parameters (orientation, direction and speed) of a continuous object's boundary. The proposed parameters estimation procedure implemented in a collaborative fashion by dynamically formed clusters (triplets) of sensor nodes. The algorithm updates the local front model parameters and propagates them to sensor nodes situated in the direction of the hazard's propagation in a fully decentralized manner. For the evaluation of the proposed collaborative algorithm we developed a flexible simulation workflow which allows us to simulate scenarios with different i) sensor node densities, ii) deployment strategies, iii) sensor node and communication (Rx and Tx) failure probabilities, and iv) propagating hazard properties (shape, size and acceleration). Extensive simulation results show that the proposed scheme can estimate with accuracy the time-varying local parameters of different types of irregular fronts, while using WSNs of realistic density. Moreover, its estimation accuracy is robust to changes in WSN density, sensor node failures and communication link failures a fact that makes the presented approach very appealing for real-world hazard tracking applications.

Simulation-driven emulation deployment in the field to assess the requirements of the collaborative WSN algorithm (Chapters 5)

To realistically asses the requirements and behavior of the proposed algorithm (see L6 in Chapter 1), we developed a simulation-driven WSN emulation workflow which allows us to estimate, before attempting to deploy a large scale WSN, the energy, processing and memory requirements of collaborative algorithms as the WSN's size increases. Extensive emulation results demonstrate that our scheme is suitable for a large-scale WSN deployment, since it respects WSNs' communication, processing, memory and energy constraints.

Continuous object boundary reconstruction algorithm (Chapters 6)

As discussed in Chapter 1, the state of the art are incapable to delineate automatically the boundary of an evolving continuous object (see L7 in Chapter 1). To address this limitation we developed an algorithm which combines dynamically the information of a small number of estimated local front models, as they become available to a fusion center, and determines a smooth curve that approximates the boundary of the continuous object at a specific time instance. By exploiting the estimation uncertainty of the local fronts evolution parameters, the proposed algorithm generates a probability field that indicate for each point of the considered area, the probability to be affected by the continuous object. Extensive simulations results demonstrate that the proposed algorithm is able to reconstruct with accuracy the boundaries of different types of continuous objects (e.g. time-varying evolution rates and/or irregular boundary shapes), using a small number of local estimates where their evolution parameters may be distorted with error.

Further Research directions

Let us conclude this document by identifying some research directions worth pursuing:

An interesting open problem, is the development of efficient WSN-based methods able to determine the number of continuous objects that evolve simultaneously into the area of interest. Another challenging problem is the development of methods able to successfully associate the estimated local front estimates to their originating continuous objects. Solving these problems can help us to develop continuous object tracking systems able to track the boundaries of multiple continuous objects that evolve simultaneously in the considered area. Moreover, by making simple modifications to the proposed boundary reconstruction algorithm we will be able to capture possible merges of the continuous object boundaries which are expected to occur during their evolution.

Another worth pursuing research direction is the development of efficient Dynamic Data Driven Assimilation Systems (DDDAS) based on the proposed WSN-based continuous object tracking scheme. As discussed in Chapter 2, a major limitation of the DDDAS systems is that in most cases the sensing data cannot be exploited to calibrate the parameters of hazard predictive models. However, the proposed continuous object tracking scheme can help us to overcome this difficulty since the produced boundary's information can be directly compared with the information of the boundary extracted by a hazard-

specific model. The direct comparison of the boundaries information can help us to develop appropriate parameter calibration methods able to improve the predictions of the hazards-specific models.

Appendix A

Local Front Orientation Updating Procedure

In this section we present the details of the local front's orientation parameter (ϕ_i^*) updating procedure (see also Section 3.4.2, Chapter 3). Let K_1 and K_2 be the points that determine the updated local front's orientation. For the calculation of the coordinates of points K_1 and K_2, namely (x_1, y_1) and (x_2, y_2), Master node S_i^M implements the following procedure: When S_i^M receives two detection messages (DMs), one message from each one of its Helpers (S_h^H, where $h \in \{j, k\}$, see Figure A.1) it checks in its neighborhood table T_i^N to find which Helper has detected the phenomenon most recently (it will be assumed to be Helper S_k^H w.l.o.g). Using the detection time of this sensor (t_{ik}) and the corresponding estimated speed values u_{ih} (see Section 3.3, Chapter3) S_i^M can calculate the expected distances $\{l_{ih},\ h \in \{j, k\}\}$ (see Figure A.1) that the two Helpers projection points, p_{ih}, would cover if they were to move with the estimated speeds $\{u_{ih},\ h \in \{j, k\}\}$ for a time interval equal to t_{ik} respectively on a line perpendicular to the local front.

$$l_{ih} = u_{ih} t_{ik},\ h \in \{j, k\}. \tag{A.1}$$

The coordinates of the projection points $p_{ih} = (x_{ih}, y_{ih})$ are determined during the *Create Helper Table Procedure* (see Section 4.2.1, Chapter 4), where the Master S_i^M solves the following linear system of equations for each one of the Slaves:

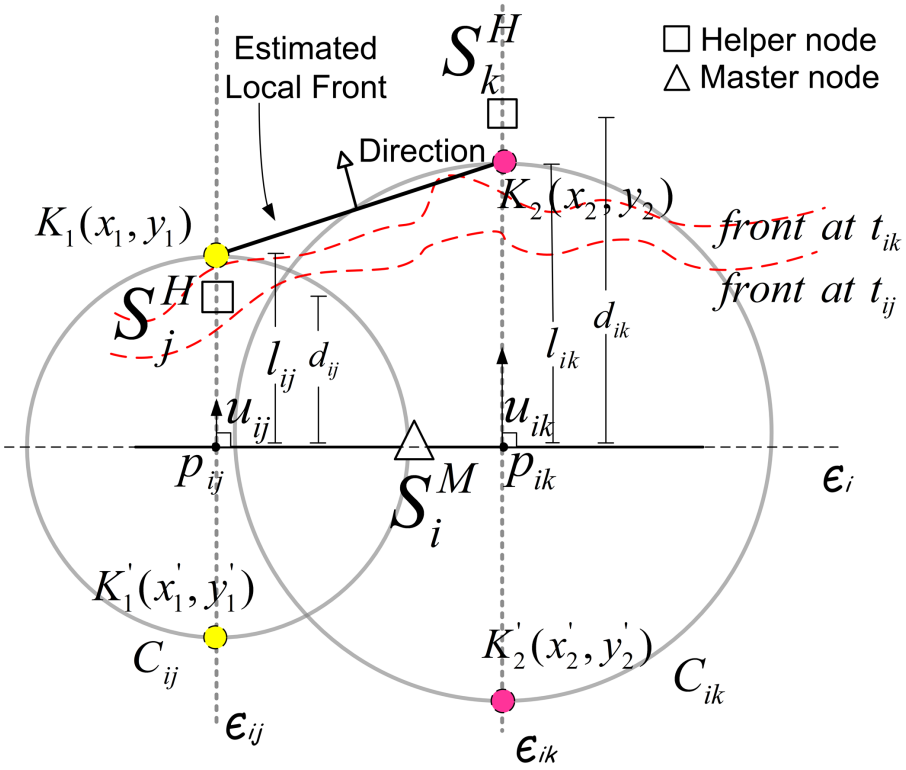


Figure A.1: Updating the local front orientation parameter.

$$\begin{cases} \epsilon_i : y_{ih} - y_i = \phi_i(x_{ih} - x_i) \\ \epsilon_{ih} : y_{ih} - y_h = -\frac{1}{\phi_i}(x_{ih} - x_h) \end{cases} \tag{A.2}$$

where (x_i, y_i) are the coordinates of Master S_i^M. Solving this system leads to the following equations for the coordinates of the projection points:

$$\begin{cases} x_{ih} = \frac{\phi_i(y_h - y_i + \phi_i x_i) + x_h}{\phi_i^2 + 1} \\ y_{ih} = \frac{\phi_i^2 y_h + \phi_i x_h + y_i - \phi_i x_i}{\phi_i^2 + 1} \end{cases} \tag{A.3}$$

To find the coordinates (to be called $(x_z, y_z), z \in \{1, 2\}$) of the two points K_z we can solve the following two systems of equations, where for K_1 ($z \leftarrow 1$,$h \leftarrow j$) and for K_2 ($z \leftarrow 2$,$h \leftarrow k$) we have:

$$\begin{cases} \epsilon_{ih} : y_z - y_{ih} = -\frac{1}{\phi_i}(x_z - x_{ih}) \\ C_{ih} : (x_z - x_{ih})^2 + (y_z - y_{ih})^2 = l_{ih}^2 \end{cases} \tag{A.4}$$

The first equation in (A.4) defines a line which is perpendicular to the local front and emanates from the corresponding Helper's projection point $p_{ih} = (x_{ih}, y_{ih})$. The second equation defines a circle centered at the corresponding projection point p_{ih} that has radius equal to l_{ih}.

Solving this system of equations provides us with two closed form algebraic expressions (A.5) for computing the intersection points of the line and the circle (yellow and magenta points in Figure A.1). From these solutions we accept only the one which lies in the half plane (determined by the local front line) where the Helper nodes reside.

$$\begin{cases} x_z = x_{ih} \pm \frac{l_{ih}\phi_i\sqrt{\phi_i^2+1}}{\phi_i^2+1} \\ \\ y_z = y_{ih} \mp \frac{l_{ih}\sqrt{\phi_i^2+1}}{\phi_i^2+1} \end{cases} \tag{A.5}$$

where x_{ih}, y_{ih} are computed according to (A.3).

Using (A.5), ϕ_i^* can be computed as shown in equation (3.14) in the main text.

$$\phi_i^* = \frac{y_2 - y_1}{x_2 - x_1}. \tag{A.6}$$

Appendix B

Performance Evaluation

B.1 Experiment 1

As discussed in the paper (see Section 4.4.1) in this experiment the diffusive phenomenon (continuous object) was modeled as two growing circles that start overlapping to form a complex front line, before they cover half of the WSN deployment area. The experimental setup indicates that at each time instance the time-varying speeds at which the two circles are expanding are equal. The evaluation metrics used to assess the accuracy of the proposed scheme are described below:

Percent Speed Error: For a model update i at time t it is defined as:

$$SE_i(t)\% = 100 \cdot \frac{|u_i^*(t) - u_R(t)|}{u_R(t)} \tag{B.1}$$

where $u_i^*(t)$ is the estimated local front's mean speed value and $u_R(t)$ is the reference (ground truth) evolution speed of the circular fronts at the time t of a model update.

Orientation Error: For the evaluation of the local front's orientation error we use the following procedure:

Let us consider first the case in which the Helper nodes of a triplet have been affected by the same circular front. In Figure B.1a the sensor node S_i^M has just updated its prior

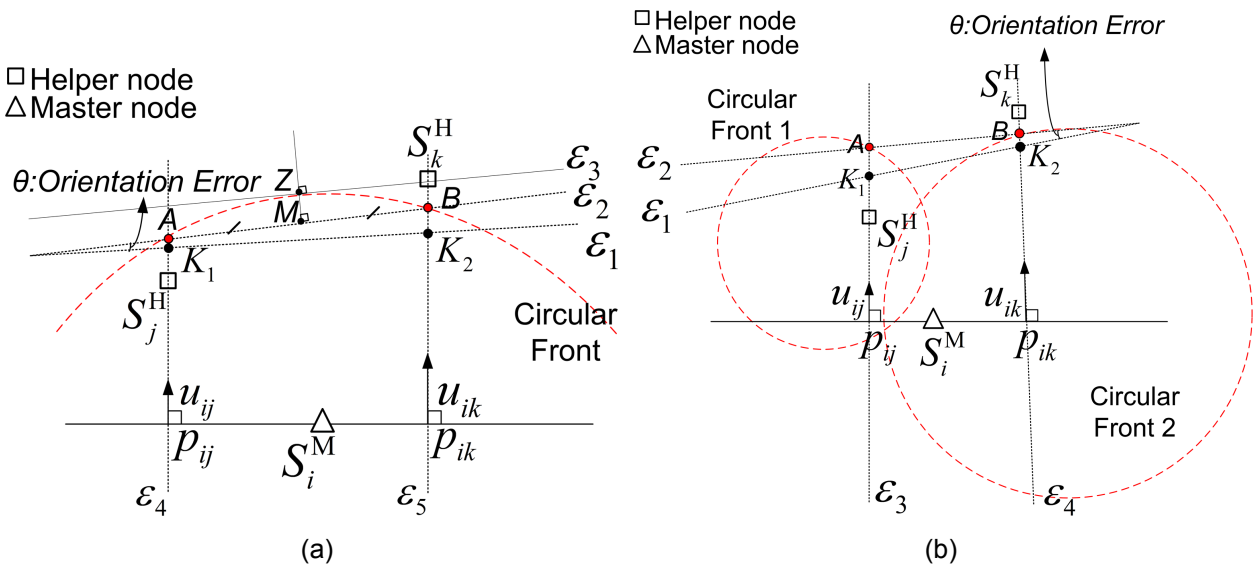


Figure B.1: The orientation error evaluation method followed when the Helper nodes of a triplet have been affected from (a) the same circle, (b) different circles.

model. The real front is represented by the dashed red line curve (an arc of the evolving circular front), and the updated local front line ε_1 is defined by points K_1 and K_2, as discussed in Section 3.3.2. Using the coordinates of the projection points p_{ij} and p_{ik} and the coordinates of points K_1 and K_2 we can derive the equations of the lines ε_4 and ε_5 respectively. By solving the system of equations of the circle (modeling the evolving front) with the aforementioned lines, we can find the coordinates of the corresponding intersection points A and B (see Figure B.1a) which define a chord of the circle. Points A and B correspond to the points where K_1 and K_2 would lie respectively if we had estimated correctly their locations (see Section S4). By assuming that the orientation of the real local front is represented by the tangent line ε_3 that osculates the circle at the middle of its arc $\overset{\frown}{AB}$ (point Z), we can compute the orientation error between the ``real" local front (ε_3) and the estimated local front (ε_1) by calculating the angle θ (in degrees) which is formed between the lines ε_1 and ε_2, taking into consideration that $\varepsilon_2 // \varepsilon_3$ (a fact easy to prove).

Let us now also consider the case where the Helper nodes of a triplet have been affected by different circular fronts (S_j^H affected by circular front 1 and S_k^H by circular front 2, see Figure B.1b). The orientation error θ is calculated as the angle (in degrees) formed by line (ε_1) of the updated local front and line (ε_2) that is determined by points A and B

(see Figure B.1b) which are the points of the corresponding circular fronts where K_1 and K_2 would lie if we had estimated correctly their coordinates. The coordinates of points A and B are calculated by solving the following systems of equations: a) of the circular front 1 and line ε_3 and b) circular front 2 and line ε_4 respectively.

B.2 Experiment 2

FLogA (Fire Logic Animator) is a web-based software tool which allows us to draw a forest area on Google Earth anywhere in Europe, insert interactively fire ignition points, simulate and animate the behavior of the evolving fire line under different conditions (see reference [87]).

FLogA considers the forest area as a grid of square cells, with every cell exhibiting different topography and weather conditions. The dimensions of the cells are determined 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 one of the grid's cells information such as the time of fire's arrival, the fire line's speed and the fire line's evolution direction etc. (see Figure B.2).

For all the wildfire scenarios of Experiment 2 (see Section 4.4.2) the $1km^2$ forest area was seen by FLogA as a grid of 500 x 500 square cells. The size of each cell was 2m x 2m. To evaluate the parameter estimation accuracy of the proposed in-network processing algorithm we used the following procedure: At the end of a COOJA simulation the generated file *Updated Models* is passed to the the Matlab component (see Section 4.3.1). This file contains the parameters of the updated local front models and the location coordinates of their middle points (see point M in Figure B.2). For each middle point M its coordinates (x_m, y_m) are calculated using the corresponding equations provided in (B.2) below where (x_1, y_1) and (x_2, y_2) are the coordinates of points K_1 and K_2 respectively (see Section 3.4.2).

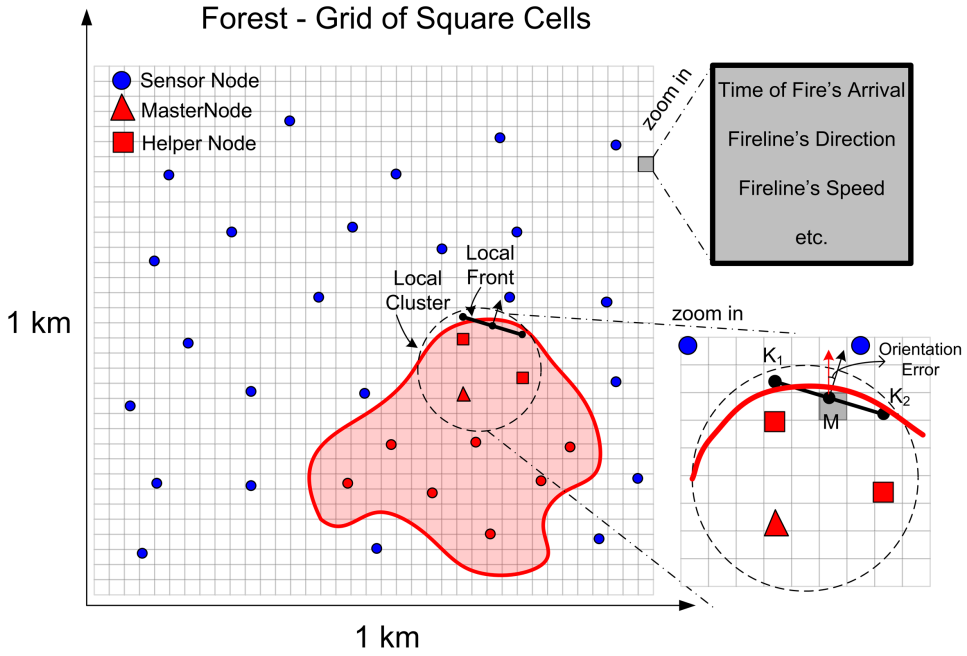


Figure B.2: The $1km^2$ forest area is seen by FLogA as a grid of square cells. At the end of a wildfire simulation FLogA predicts, for each one of the square cells, information such as the time of fire's arrival, frontline's evolution direction and speed. At the bottom right corner we visualize the direction error evaluation procedure described in the text (see Section S2.2).

$$x_m = \frac{x_1 + x_2}{2}, \qquad y_m = \frac{y_1 + y_2}{2} \tag{B.2}$$

Using the location coordinates of the updated local fronts' mid-points, a Matlab procedure determines the grid's cells that contain them. Subsequently, another Matlab procedure compares the evolution parameters (orientation and speed) of the updated local front models with the ground truth values, as provided by FLogA's output files for the corresponding cells.

Orientation Error: is defined as the angle of the estimated local front's segment evolution direction (black arrow in Figure B.2) and the real front line's evolution direction (red arrow in Figure B.2) provided by FLogA for cell c where the corresponding local front's middle point belongs to.

Percent Speed Error: Similar to Experiment 1 the speed error is determined as:

$$SE_i(c)\% = 100 \cdot \frac{|u_i^*(c) - u_R(c)|}{u_R(c)} \tag{B.3}$$

where $u_i^*(c)$ is the estimated $i\ th$ local front's mean speed value and $u_R(c)$ is the corresponding ``real'' (reference, ground truth) front's evolution speed at cell c where the corresponding local front's middle point M lies.

Appendix C

Experiment 1: Setup and Animation

In Experiment 1 a complex diffusive phenomenon is assumed, modeled as two circles of fixed centers located outside the $1km^2$ square deployment area of the WSN. Considering the area's bottom left corner as the origin, the circles (colored black and red in Figure C.1) are centered at points (100,-50) and (950,-300) (not visible in the Figure C.1) and have radii initially equal to 50m and 300m respectively, i.e. the circles represent two distinct diffusive hazards which have just started entering the deployment area at the beginning of the simulation (see Figure C.1a). The two circles start to overlap as they grow, before covering half of the WSN deployment region (an $1km^2$ square area with 100 randomly deployed sensor nodes), to form a complex front line (e.g. see Figures C.1b, C.1c, C.1d). The speed at which their radii are increasing is a triangular function with initial value $0.5\text{m}/\text{min}$ and a rate of increase of $(0.006\text{m}/\text{min}^2)$ until it assumes its maximum value $(2.5\text{m}/\text{min})$, which happens when the two circles reach the middle of the deployment area. Beyond that point it starts decreasing at the same rate until it returns back to its initial value. Therefore, when the front arrives at the other end of the deployment area its speed is again back to $0.5\text{m}/\text{min}$.

For Experiment 1 (presented in Section 4.4.1) we have created (using Matlab) an animation of the model updates (see file `Experiment1TwoFronts.mp4` in [88]) in order to help the reader get a visual impression of the distributed algorithm's results (estimated local fronts) as the hazard's front line is progressing in the area of a deployed WSN. Figure C.1 shows six snapshots of the created animation. When a sensor detects the front its colour is changed from blue to red. When a sensor assumes a Master status it is marked by a black

triangle. The line segments that appear in the animation during the front's evolution represent the computed local front estimates (model updates) approximating the progressing frontline. The four more recent local front updates are coloured black. The snapshot in Figure C.1e clearly demonstrates that the local front estimates (black line segments) may approximate even complex hazard fronts (formed by the two overlapping circles).

Figure C.1f shows, for the presented simulation scenario of Experiment 1, the Master status ``inheritance'' trajectories during the hazard's evolution. The arrowed red lines point to the Helper nodes that inherited the Master status. A sequence may get discontinued when none of the Helpers meet the conditions to become the new Master (see Section 4.2.1). In that case the last arrow in a sequence points to the Helper node that detected most recently the evolving front line. Notice that each Master node (black triangle) propagates the updated local front parameter estimates in the direction of the hazard's propagation. Each updated local front segment (represented by a cyan or black line segment) is centred at the location of the Helper node who detected last the evolving front (regardless of whether it mets the conditions to become the new Master node or not). In this Figure we also observe that two Master status ``inheritance'' trajectories may merged at the same sensor node (the same Helper). This interesting case may happen when a Helper node is enslaved to two Masters (belongs to the intersection of their clusters).

At this point we want to mention the following interesting scenario that can be handled without any problem: A Helper node may be enslaved to more than one Masters (if it belongs to the intersection of their clusters). If this Helper node receives a `MOM` from one of its Masters it may accept the offer (if it satisfies the necessary conditions) while it also continues to serve as Helper to another Master. If now this new Master receives (before it updates its model parameters) a `MOM` from a second Master, it will checks (for the corresponding model) the ``*Master check Necessary Conditions''* (see Section 4.2.1) and if they are satisfied it will accepts that offer as well. By the end of this procedure the new Master has formed for each accepted offer a T^H table that contains the legitimate Helper pairs that could be used for updating each model. The new Master waits to receive the DMs from two of its neighbors that constitute a legitimate Helpers pair in at least one of its T^H tables. If the pair that responds is contained in only one of its T^H tables then the node updates only the corresponding model and discards the rest. Otherwise, if the same

Helpers pair is present in more than one T^H tables, the node updates the corresponding models and selects among them to propagate the one with the smallest speed variance (smallest speed uncertainty).

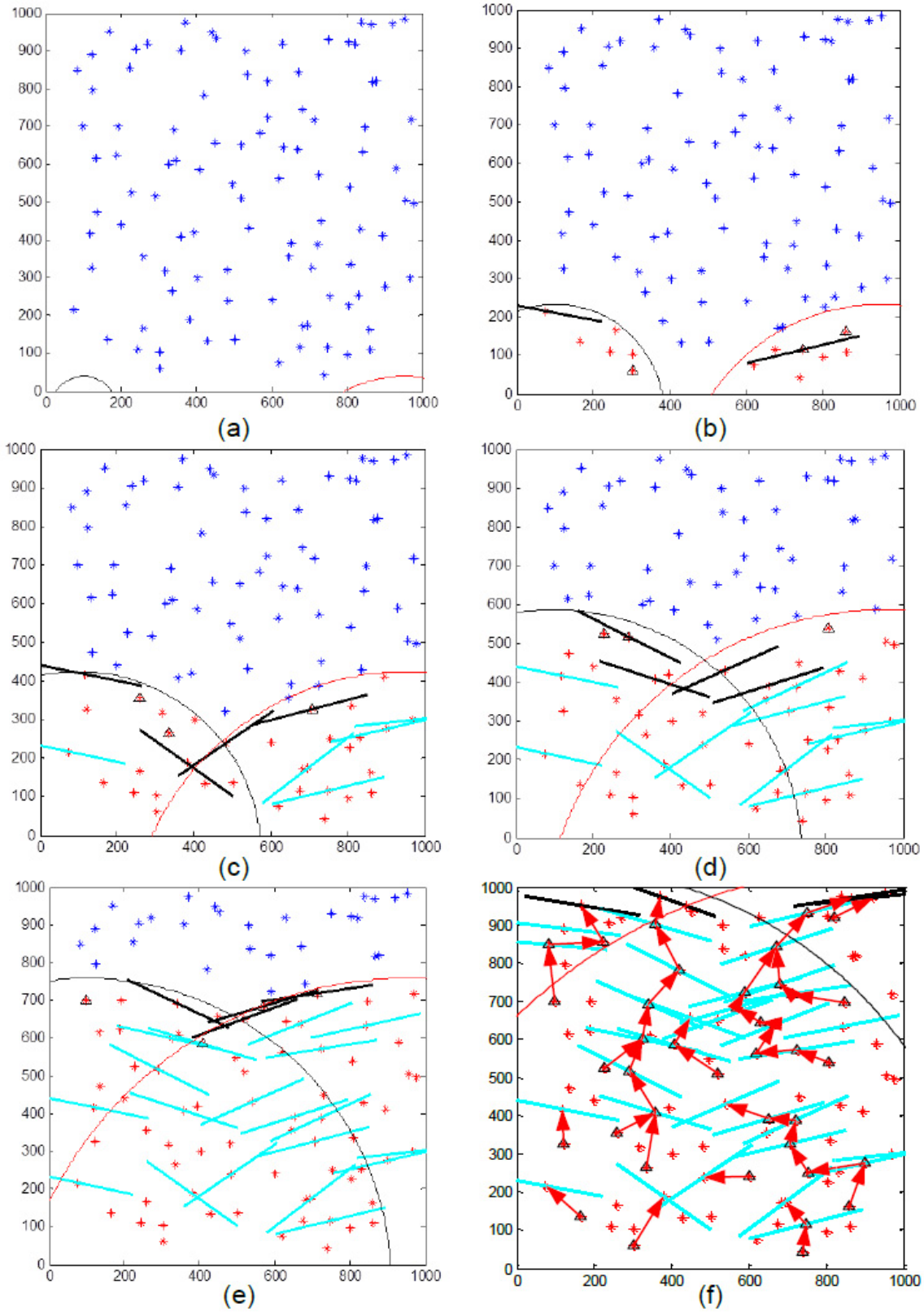


Figure C.1: Figures C.1a -C.1f, show 6 snapshots of the presented simulation scenario of Experiment 1. In Figure C.1f the red arrows show the Master status inheritance trajectories that occur during the hazard's front line propagation.

Appendix D

Calculating the Coordinates of Local Front End Points

Using the middle point's coordinates $P_i^M = (x_i^M, y_i^M)$, the length $l_i = 2R_i$, and the orientation realization $\hat{\phi}_i$ 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:

$$\begin{cases} \varepsilon_i : y - y_i^M = tan(\hat{\phi}_i)(x - x_i^M) \\ C_i : (x - x_i^M)^2 + (y - y_i^M)^2 = R_i^2 \end{cases} \tag{D.1}$$

The first equation in (D.1) defines a line (ε_i in Figure D.1) on which the local front's line segment lies. The second equation defines a circle (C_i) of radius equal to $R_i = \frac{l_i}{2}$, centered at the local front's segment middle point P_i^M (see Figure D.1). The solution of (D.1) provides us the following closed formed algebraic expression (see equation (D.2)) which are used for the computation of the local front's end points coordinates (x_i^{Ez}, y_i^{Ez}) where $z = \{1, 2\}$.

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

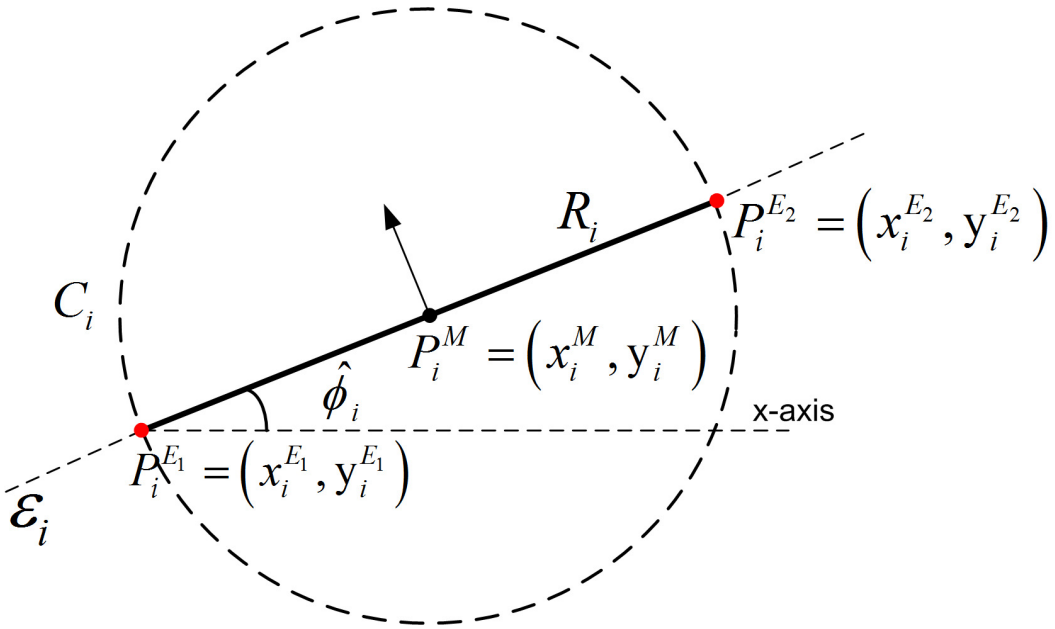


Figure D.1: The local front estimate m_i; the coordinates of its end points (red dots) are determined by the points of intersection between line ε_i and circle C_i.

Appendix E

Space-Time Evolution of a Local Front Estimate

Let's assume that we want to determine the location of a local front estimate m_i (estimation time t_i) at time instance t_j (where $t_i < t_j$).

Using the speed realization $\hat{u}_i$, we calculate (see equation (E.1)) the distance $\hat{d}_{ij}$ that it will be covered by m_i if it moves for time interval equal to $\Delta t_{ij} = |t_i - t_j|$.

$$\hat{d}_{ij} = \hat{u}_i \Delta t_{ij} \tag{E.1}$$

In sequence, solving the system of equations in (E.2), we determine the middle point's coordinates ($P_i^M(t_j) = (x_i^M(t_j), y_i^M(t_j))$) of the space time projected local front estimate ($m_i(t_j)$).

$$\begin{cases} \varepsilon_i : y - y_i^M = \frac{-1}{tan(\hat{\phi}_i)}(x - x_i^M) \\ C_i : (x - x_i^M)^2 + (y - y_i^M)^2 = \hat{d}_{ij}^2 \end{cases} \tag{E.2}$$

The first equation in (E.2) defines a line (ε_i in Figure E.1) which is perpendicular to the local front and emanates form its middle point $P_i^M = (x_i^M, y_i^M)$. The second equation defines a circle of radius $\hat{d}_{ij}$ centered at the local front's segment middle point P_i^M (see Figure E.1).

Solving this system of equations provides us with two closed form algebraic expressions for computing the intersection points of the line ε_i and the circle C_i (see Figure E.1).

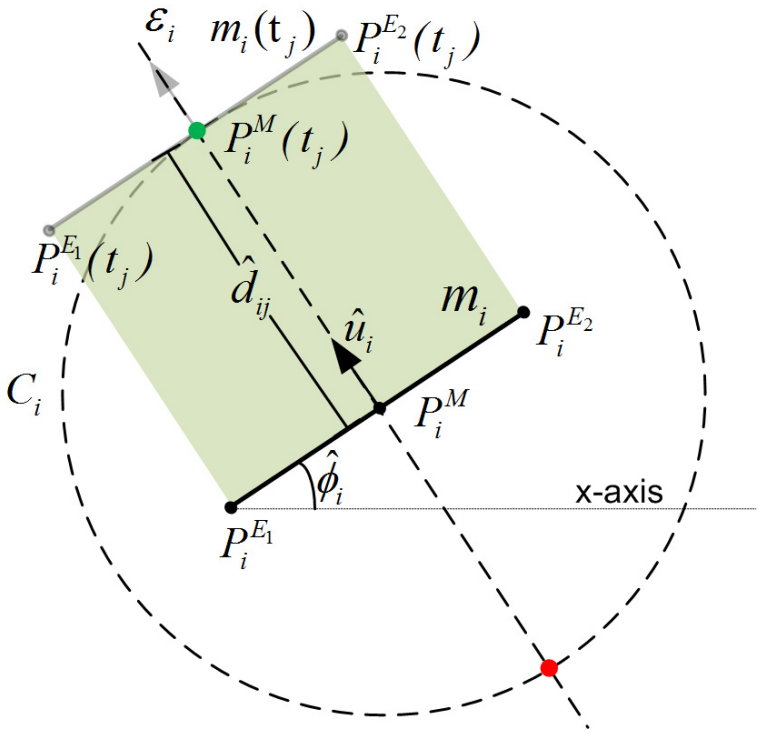


Figure E.1: Space-time evolution of the local front m_i at time t_j.

From these solutions we accept only the point that lies in the half plane which indicated by the direction parameter δ_i (green point in Figure E.1).

$$\begin{cases} x_i^M(t_j) = \pm\frac{\hat{d}_{ij}tan(\hat{\phi}_i)\sqrt{tan^2(\hat{\phi}_i)+1}}{tan^2(\hat{\phi}_i)+1} + x_i^M \\ y_i^M(t_j) = \mp\frac{\hat{d}_{ij}\sqrt{tan^2(\hat{\phi}_i)+1}}{tan^2(\hat{\phi}_i)+1} + y_i^M \end{cases} \tag{E.3}$$

The point $P_i^M(t_j) = (x_i^M(t_j), y_i^M(t_j))$ consists the new location of $m_i(t_j)$ middle point. For the calculation of $m_i(t_j)$ end points coordinates ($P_i^{E_z}(t_j) = (x_i^{E_z}(t_j), y_i^{E_z}(t_j))$ where $z = \{1, 2\}$) we apply the equations provided in (D.2).

Appendix F

Information Fusion of Local Fronts

We present the proposed information fusion technique applied in event's 4 handling procedure (see Section 6.2.1). This technique uses the information of the local fronts estimates ($m_i(t_f)$ and $m_j(t_f)$ in Figure F.1) that satisfy the event's 4 condition, and calculates the parameters of a ``new'' local front (m_f) that describes locally the evolution characteristics of the continuous objects boundary at time t_f.

The information fusion technique calculates for each local front estimate $m_h(t_f)$ (where $h \in \{i, j\}$), the time differences $\Delta t_h = |t_f - t_h|$. Next, using the parameters of the speed distributions U_h, it determines for each local front estimate $m_h(t_f)$ the distance D_h that it will cover if it moves for time interval equal to Δt_h. Since the speed U_h described by a Normal distribution $\mathcal{N}(u_h, s_h^2)$, the distance D_h will also be described by Normal distributions $\mathcal{N}(d_h, \omega_h^2)$ with parameters:

$$\begin{aligned} d_h &= u_h \Delta t_h \\ \omega_h &= s_h \Delta t_h \end{aligned} \tag{F.1}$$

In equation (F.1) it is worth to mention that as the time difference Δt_h increases, the uncertainty (ω_h) about the local front's (m_h) traveled distance will also increases.

In sequence, the procedure calculates for each local front m_h the distance $\hat{d}_h$, that it will cover if it moves with the sampled speed $\hat{u}_h$ for time interval Δt_h.

$$\hat{d}_h = \hat{u}_h \Delta t_h \tag{F.2}$$

Using the orientation Φ_h and distance D_h distributions, and the values $\hat{\phi}_h$ and $\hat{d}_h$, we calculate the weights (see equation (F.3)) that will be used for the calculation of the m_f parameters:

$$w_h = \frac{\Phi_h(\hat{\phi}_h) D_h(\hat{d}_h)}{\sum\limits_{k=i,j} \Phi_k(\hat{\phi}_k) D_k(\hat{d}_k)} \tag{F.3}$$

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

Using the weights w_h, the procedure calculates the parameters of the ``new'' local front estimate m_f as follows:

$$\mathcal{Z}_f = \sum_{h=\{i,j\}} w_h \mathcal{Z}_h \;\; where \;\; \mathcal{Z} = \{\hat{u}_f, \hat{\phi}_f, l_f, x_f^M, y_f^M\} \tag{F.4}$$

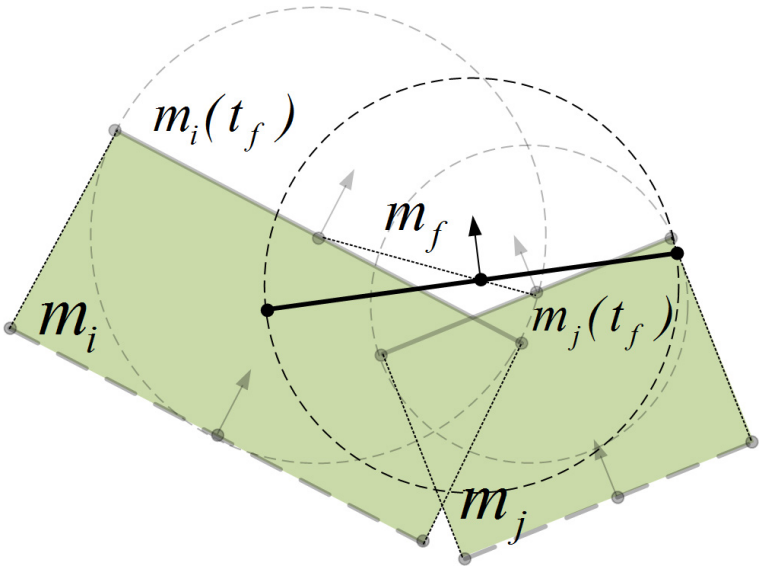
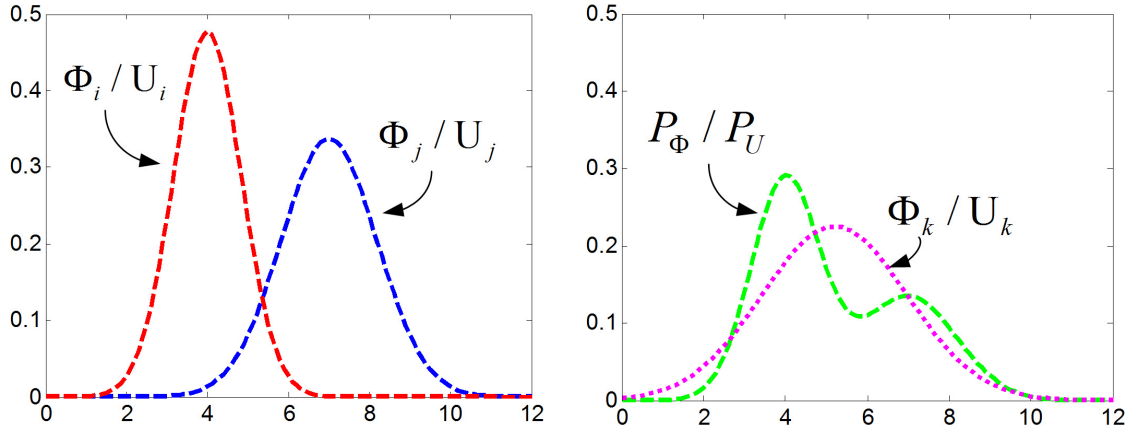


Figure F.1: Models Information Fusion.

To estimate the evolution direction parameter δ_f the procedure uses the local fronts orientations ($\hat{\phi}_h$), the evolution directions (δ_h) and the weights (w_h) and determines two vectors which: a) are perpendicular to the corresponding local fronts segments, b) point to the corresponding local fronts' evolution directions and c) the ratio of their lengths is equal to the corresponding ratio of their weights $\frac{w_i}{w_j}$. In sequence, using these vectors the algorithm 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 and we 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:



(a) (b)

Figure F.2: Estimating the speed/orientation (U_f/Φ_f) distributions of model m_k. (a) The speed/orientation distributions of models m_i and m_j. (b) The mixture models P_U/P_Φ resulting by combining speed/orientation normal distributions of $\{m_i, m_j\}$ and the normal distribution U_f/Φ_f that best approximates P_U/P_Φ by minimizing the Kulback-Leibler divergence.

Using the w_h and the corresponding speed (orientation) distributions of the local fronts m_h, we find their Gaussian mixture (see Figure F.2):

$$p_U(u) = \sum_{h=\{i,j\}} w_h * \mathcal{N}(u|u_h, s_h^2)$$
$$p_\Phi(\phi) = \sum_{h=\{i,j\}} w_h * \mathcal{N}(\phi|\phi_h, \sigma_h^2) \quad \text{(F.5)}$$

In sequence, by applying variational calculus we approximate the Gaussian mixture in F.5 by a Normal distribution which minimizes the Kullback-Leibler (KL) divergence (maximizes the similarity) from the Gaussian mixture [80]. The general form of the equations which can be used to compute the parameters of these normal distributions are:

$$\hat{\mu} = \sum_n w_n \mu_n \quad \text{(F.6)}$$

$$\hat{\Sigma} = \sum_{n} w_n(\Sigma_n + (\mu_n - \hat{\mu})(\mu_n - \hat{\mu})^T) \tag{F.7}$$

In our specific case these equations are reduced to:

Speed parameters

$$u_f = \sum_{h=\{i,j\}} w_h u_h \tag{F.8}$$

$$s_f^2 = w_i s_i^2 + w_j s_j^2 + w_i w_j (u_i - u_j)^2 \tag{F.9}$$

Orientation parameters

$$\phi_f = \sum_{h=\{i,j\}} w_h \phi_h \tag{F.10}$$

$$\sigma_f^2 = w_i \sigma_i^2 + w_j \sigma_j^2 + w_i w_j (\phi_i - \phi_j)^2 \tag{F.11}$$

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

Appendix G

Initialization of Local Front Evolution Parameters

In this Appendix we present how we initialize the local fronts' evolution parameters for the experiments presented in Chapter 6.

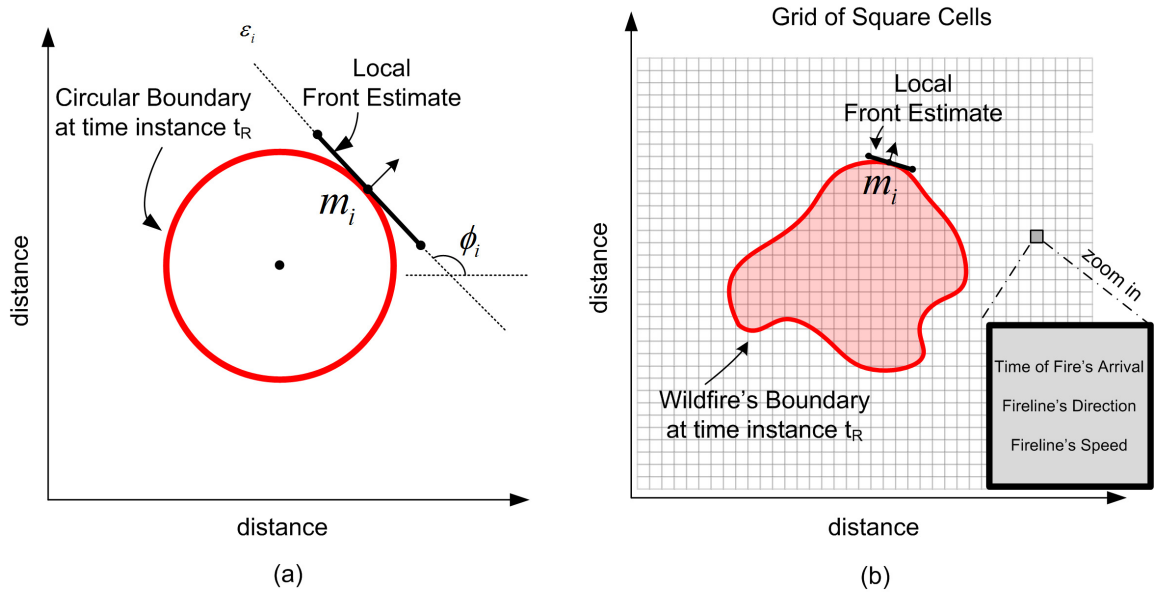


Figure G.1: Initalization of the local fronts parameters.

G.1 Experiment 1: Circular Front

Let's assume that the circular boundary reaches the middle point of a local front segment (m_i) at time instance t_R (see Figure G.1a). We set its speed (u_i) parameter equal to the

speed that has the circle's increasing radius at time t_R and its orientation (ϕ_i) parameter equal to the angle formed between the x-axis and the circle's tangent line (ε_i in Figure G.1a) that osculates from the middle point of m_i.

G.2 Experiment 2: FLogA Simulation

FLogA (Fire Logic Animator) is a web-based software tool which allows us to draw a forest area on Google Earth anywhere in Europe, insert interactively fire ignition points, simulate and animate the behavior of the evolving fire line under different conditions [87]. FLogA considers the forest area as a grid of square cells, with every cell exhibiting different topography and weather conditions. The dimensions of the cells are determined 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 one of the grid's cells information such as the time of fire's arrival, the fire line's speed and the fire line's evolution direction etc (see Figure G.1b). We set the local front's speed (u_i) and orientation (ϕ_i) parameters, equal to the speed and evolution direction values of the cell that contains the middle point of m_i.

Bibliography

[1] P. Parmar, M. Zaveri, ``Multiple Target Tracking and Data Association in Wireless Sensor Network,'' IEEE Int. Conf. on Computational Intelligence and Communication Networks (CICN) , pp.158,163, 3-5 Nov. 2012.

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

[3] J. Lui, J.E. Reich and F. Zhao, ``*Collaborative in-network processing for target tracking*'', EURASIP, Journal of Applied Signal Processing, vol.2003, pp.378-391, Mar. 2003.

[4] J. Lui, M. Chu, J.E. Reich and F. Zhao, ``Distributed state representation for tracking problems in sensor networks'', in Proc. 3rd int. symp. Information Processing Sensor Networks (IPSN), Berkeley, CA, April. 2004, pp. 234-242.

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

[6] M.E. Liggins, C. Y. Chong, I. Kadar, M.G. Alford, V. Vannicola, and S. Thomopoulos, ``Distributed fusion architectures and algorithms for target tracking'', Proceedings of the IEEE , vol.85, no.1, pp.95,107, Jan. 1997.

[7] J. Shin, L. Guibas, and F. Zhao, ``A distributed algorithm for managing multi targets identities in wireless ad-hoc sensor networks'', In proc. 2nd Workshop Information Processing Sensor Networks (IPSN), April 2003.

[8] I. Hwang, K. Roy, H. Balakrishnan, and C. Tomlin, ``A distributed multiple target identity management algorithm in sensor networks,'' in Proc. 43rd IEEE Conf. Decision Control, Paradise Island, Bahamas, Dec. 2004.

[9] J. Liu, M.Chu, E. Reich, ``*Mulittarget Tracking in Distributed Sensor Network*'', IEEE Signal Processing Magazine, vol. 24, no. 3 pp.36-46, May 2007.

[10] T. Fortnamm, Y. Bar-Shalom, and M. Scheffe, ``Multi-target tracking using joint probabilistic data association'', in Proc. 19th IEEE Conf. Decision Control, Albuquerque, NM, Dec. 1980.

[11] S. Oh, S. Russell, and S. Sastry, ``Markov Chain Monte Carlo data association for general multiple target tracking problems,'' in Proc. 43rd IEEE Conf. Decision Control, Paradise Island, Bahamas, Dec. 2004.

[12] S. Oh, I. Hwang, K. Roy, and S. Sastry, ``A fully automated distributed mutiple target tracking and identity management algorithm'' in Proc., AIAA Guideance Navigation, and Control Conf., Aug.2005.

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

[14] O. Sekkas, D.V. Manatakis, E. S. Manolakos and S. Hadjieuthimiades, ``*Sensor and Computing Infrastructure for Environmental Risks - The SCIER System'' chapter 16*, in Advanced ICTs for Disaster Management and Threat Detection: Collaborative and Distributed Frameworks, pp. 262-278 ISBN: 9781615209873 l, 2010.

[15] O. Sekkas, D. V. Manatakis, S. Hadjiefthymiades, E. S. Manolakos, ``Detection and evolution prediction of fires in the Wildland Urban Interface using Sensor Networks and Grid Computing,'' In Proc. VI International Conference on Forest Fire Research, Coimbra-Portugal, November 2010.

[16] N. Voutsinas, D. V. Manatakis, and E. S. Manolakos, ``Using the GRID for Forest Fire Front Evolution Prediction'', In Proc. 4th International Workshop on Grid Computing for Complex Problems (GCCP), Bratislava, Slovakia, Oct. 2008. pp. 20-27.

[17] M. Hefeeda, M. Bagheri, ``Wireless Sensor Networks for Early Detection of Forest Fires'', Pisa, Italy, in the Proceedings of the 1st International Workshop on Mobile Ad hoc and Sensor Systems for Global and Homeland Security (MASS-GHS), Pisa, Italy, pp. 1-6, Oct. 2007.

[18] L. Zhanqing, A. Khananian, R. Fraser, J. Cihlar, ``*Automatic Detection of Fire Smoke using Artificial Neural Networks and Threshold Approaches applied to AVHRR Imagery*'', IEEE Transactions. On Geoscience and Remote Sensing, vol. 39, n9, pp. 1859-1870, 2001.

[19] I. Yoon, D. K. Noh, D. Lee; R. Teguh, T. Honma, H. Shin, ``Reliable Wildfire Monitoring with Sparsely Deployed Wireless Sensor Networks,'' Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on, pp.460,466, 26-29 March 2012.

[20] A. Santoni, T., Santucci, J.F. de Gentili, E. C. Bernadette, ``Using Wireless Sensor Network for Wildfire detection. A discrete event approach of environmental monitoring tool,'' Environment Identities and Mediterranean Area, (ISEIMA '06), First international Symposium on, pp.115,120, 9-12 July 2006.

[21] Y. Li, Z. Wang, Y. Song, ``Wireless Sensor Network Design for Wildfire Monitoring,'' The Sixth World Congress on Intelligent Control and Automation, 2006 (WCICA 2006), vol.1, pp.109,113.

[22] Chintalapudi, and R. Govindan, ``Localized edge detection in sensor fields,'' in IEEE International Workshop on Sensor Network Protocols and Applications, pp. 59 - 70, May 2003.

[23] R. Nowak and U. Mitra, ``Boundary estimation in sensor networks: theory and methods,'' in 2nd Inter. Workshop on Information Processing in Sensor Networks, pp. 22-36. Apr. 2003.

[24] P. K. Liao, M. K. Chang, and C. J. Kuo, ``Distributed edge detection with composite hypothesis test in wireless sensor network',' in Proc. IEEE Inter. Global Communication Conference (GLOBECOM), pp. 129-133, 2004.

[25] P. K. Liao, M. K. Chang, and C. J. Kuo, ``Distributed Edge Sensor Detection with one and two level decisions'', in Proc. IEEE Inter. Conf. on Acoustics, Speech and Signal Processing (ICASSP), pp. 297-300, 2004.

[26] P. K. Liao, M. K. Chang, and C. J. Kuo, ``Statistical Edge Detection with Distributed Sensors under the Neyman-Pearson (NP) Optimality,'' in Proc. IEEE Inter. Conf. Vehicular Technology, pp. 1038 □ 1042, 2006.

[27] J. Liu, P. Cheung, L. Guibas, and F. Zhao. ``Apply geometric duality to energy efficient non-local phenomenon awareness using sensor networks'', IEEE Wireless Communication Magazine, 11:62--68, 2004.

[28] X. Ji, H. Zha, J.J. Metzner, and G. Kesidis, ``Dynamic cluster structure for object detection and tracking in wireless ad-hoc sensor networks,'' Proc. IEEE International Conference on Communications (ICC), pp. 3807 - 3811, 2004.

[29] 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 Consumer Communications and Networking Conference 2008, Las Vegas, USA, pp. 168-174, 2008.

[30] C. Zhong and M. Worboys, ``Energy-efficient continuous boundary monitoring in sensor networks,'' Technical Report, 2007. Available: http://ilab1.korea.ac.kr/papers/ref2.pdf

[31] J.Kim, K. Kim, S.Chauhdary, W. Yang, M. Park, ``textitDEMOCO: Energy-Efficient Detection and Monitoring for Continuous Objects in WSN'', IEICE Trans.on Communications, vol.E91-B, no. 11, pp.3648-3656, Nov. 2008.

[32] B. Park, S. Park, E. Lee, and S.-H. Kim, ``Detection and Tracking of Continuous Objects for Flexibility and Reliability in Sensor Networks,'' in Proc. IEEE Inter. Conf. International Conference on Communications, Cape Town pp.1-6, 2010.

[33] E. Lee, S. Park; F.Yu; M.-S. Jin; H. Park, S.-H. Kim, ``Dynamic Rectangle Zone-Based Collaboration Mechanism for Tracking Continuous Objects in Wireless Sensor Networks'', in Proc. of the 4th IEEE Conf. Wireless Communications, Networking and Mobile Computing, pp. 1-5, 2008.

[34] M.-S. Jin, F. Yu, S. Park, E. Lee, S.-H. Kim, ``Localized Mechanism for Continuous Object Tracking and Monitoring in Wireless Sensor Networks," in Proc. of the IEEE Conf. Autonomous Decentralized Systems, pp. 1-8 2009.

[35] S.-C. Tu, G.-Y. Chang, J.-P. Sheu, W. Li, and K.-Y. Hsieh, ``*Scalable Continuous Object Detection and Tracking in Sensor Networks,*" Journal of Parallel and Distributed Computing (Elsevier), vol. 70, Issue 3, pp.212-224, March 2010.

[36] H. Luan, Y. Zhang, D. Gao, Y. Zhen, and X. Ma, ``Continuous Object Tracing in Wireless Sensor Networks," in Proc. of the IEEE International Conference on Electrical and Control Engineering, pp.3410-3413, 2011.

[37] 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 Inter. Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1725-1730, 2010.

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

[39] H. Hong, S. Oh, J. Lee, S. Kim, ``A Chaining Selective Wakeup Strategy for a Robust Continuous Object Tracking in Practical Wireless Sensor Networks," Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on , vol., no., pp.333,339, 25-28 March 2013

[40] J. Chen, M. Salim and M. Matsumoto, ``A Gaussian Mixture Model-based Event-Driven Continuous Boundary Detection in 3D Wireless Sensor Networks," Publisher: InTech, ISBN: 978-953-307-321-7 , 2010.

[41] G. Jin and S. Nittel, ``Tracking deformable 2D objects in wireless sensor networks," In Proc. of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 491-494, 2008.

[42] Y. Xu, W. Bao, and H. Xu, ``An algorithm for continuous object tracking in WSNs,'' in Proc.of the IEEE Conf. Research Challenges in Computer Science, pp.242-246, 2009.

[43] S. Duttagupta, K.Ramamrithma, ``Distributed Boundary Estimation using Sensor Networks'', in Proc. of the international IEEE conference Mobile Ad-hoc and Sensor Systems (MASS), Vancouver (BC), pp. 316-325, Oct. 2006.

[44] S. Duttagupta, K.Ramamrithma, ``*Tracking Dynamic Boundaries using Sensor Network*'', IEEE Transactions on Parallel and Distributed Systems, vol. 22, num.10, pp. 1766-1774, Oct. 2011.

[45] Q. Huang; C. L. Roman,``Reliable mobicast via face-aware routing,'' Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), vol.3, no., pp.2108,2118 vol.3, 7-11 March 2004

[46] H. Anderson,``Predicting wind-driven wildland fire size and shape''. USDA Forest Service Research Paper. INT-305, Feb 1983.

[47] M. Alexander, ``Estimating the length to breadth ratio of elliptical forest fire patterns''. Proc. 8th Conf. on Fire and Forest Meteorology, pp. 287-304. 1985.

[48] M. Marghany, ``*RADARSAT for oil spill trajectory model*'', Journal of Environmental Modeling and Software, Vol. 19, May 2004, pp. 473-483.

[49] E. S. Manolakos, D. V. Manatakis, G. Xanthopoulos, ``Temperature Field Modeling and Simulation of Wireless Sensor Network Behavior During a Spreading Wildfire'',In Proc. 16th European Signal Processing Conference (EUSIPCO), August, 2008.

[50] U.S. Tristan, ``The Diffusion Equation A Multi-dimensional Tutorial'', Available: www.rpgroup.caltech.edu/ natsirt/aph162/diffusion.pdf (10/09/2011).

[51] J. Glasa , L. Halada,``On elliptical model for forest fire spread modeling and simulation'', International Journal of Mathematics and Computers in Simulation, vol.78 issue.1, pp.76-88, June, 2008.

[52] M. A. Finney, ``FARSITE: fire area simulator - model, development and evaluation'', USDA Forest Service, Res. paper RMRS-RP-4, 1998.

[53] J. Glasa, L. Halada, ``Application of envelope theory for 2D fire front evolution□, Proceedings of the International Conference on Forest Fire Research, Figueira da Foz, 2006.

[54] J. Glasa, E. Pajorova, L. Halada, P. Weisenpacher,``Animation of forest fire simulation'', Proceedings of the International Conference on Environmental Applications and Distributed Computing, Bratislava, 2006, 20-29.

[55] Glasa, P. Weisenpacher, ``Computer reconstruction of a fire in Slovak Paradise National Park in October 2000'', Research report No. APVT-2007-03, Institute of Informatics, Slovak Academy of Sciences, June 2007, 1-26.

[56] L. Halada, P. Weisenpacher, ``Principles of forest fire spread models and their simulation'', Journal of the Applied Mathematics, Statistics and Informatics, Vol.1, Iss.1, 2005, 3-13.

[57] R. R. Linn, J. Reisner, J. C. Colman, J. Winterkamp,``Studying wildfire behaviour using FIRETEC'', International Journal of Wildland Fires, Vol.11, 2002, 233-246.

[58] A. M. G. Lopes, A. C. M. Sousa, D. X. Viegas, ``Numerical simulation of turbulent flow and fire propagation in complex terrain'', Numerical Heat Transfer, Vol. 27, Iss. 2, 1995, 229-353.

[59] D. M. Molina-Terren, E. R. Martinez-Lopez, D. Garcia-Marco, ``Farsite simulations for cost-effective wildland fire planning: case studies in Spain'', Forest Ecology and Management, Vol. 234S, 2006, 127.

[60] D. Morvan, J. L. Dupuy, ``Modeling the propagation of a wildfire through a Mediterranean shrub using a multiphase formulation'', Combustion and Flame, Vol.138, 2004, 199-210.

[61] E. Pajorova, L. Hluchy, L. Halada, P. Slizik, ``3D visualization tool for virtual models of natural disasters'', Proceedings of the Virtual Reality International Conference, Laval, France, 2007, 37-43.

[62] J.Glasa, ``*Computer simulation and predicting dangerous forest fire behavior*'', International Journal of Mathematics and Computer in Simulation, Issue 2,Vol.3, 2009.

[63] J. Mandel, J. Beezley, J. Coen, M. Kim, ``*Data Assimilation for Wildland Fires: Ensemble Kalman filter in coupled atmosphere-surface models*'', IEEE Control Systems Magazine, vol.29, issue 3. pp. 47-65, 2009.

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

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

[66] X. Yan; F. Gu; X. Hu; S. Guo, ``A dynamic data driven application system for wildfire spread simulation,'' Simulation Conference (WSC), In Proc of the Winter, pp.3121,3128, Dec. 2009

[67] T. Artes, A. Cencerrado, A. Cortes, T. Margalef, D. R. Aseretto, T. Petrolagkis, J. S. M. Ayanz, ``Towards a Dynamic Data Driven Wildfire Behavior Prediction System at European Level'', In Proc. of the 14th Int. Conf. Computer Science (ICCS 2014), 29 (2014), pp. 1216 -1226.

[68] A. Ghosh, S. K. Das, ``*Coverage and Connectivity Issues in Wireless Sensor Networks*'', Elsevier Journal. Pervasive and Mobile Computing, Vol. 4, Issue. 3, pp. 303-334, June 2008.

[69] Z. Zhou and J. Hou, ``*Sensor Deployment and Target Localization in distributed Sensor Networks*'', ACM Transactions on Embedded Computing Systems, 3(1):61-91, February 2004.

[70] Y. Zou and K. Chakrabarty, ``*A Distributed Coverage and Connectivity Centric Technique for Selecting Active Nodes in Wireless Sensor Networks*'', IEEE Trans. on Computers, 54(8):978-991, 2005.

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

[72] A. Hossain, P. K. Biswas and S. Chakrabarti, ``Sensing Models and its Impact on Network Coverage in Wireless Sensor Network'', Industrial and Information Systems, (ICIIS 2008). IEEE Region 10 and the Third int. Conf. on , vol., no., pp.1,5, 8-10 Dec. 2008.

[73] P. Soreanu, Z. Volkovich, ``*New Sensing Model for Wireless Sensor Networks*'', Int. Jour. on Advances in Networks and Services, vol 2, no. 4, pp. 261-272, 2009.

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

[75] A. Elfes, ``*Occupancy grids: a stochastic spatial representation for active robot perception*'', In Autonomous Mobile Robots: Perception, Mapping and Navigation, Vol. 1, IEEE Computer Society Press, pp. 60-70, 1991.

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

[77] J. Zhang, T. Yan, S. H. Son, ``Deployment Strategies for Differentiated Detection in Wireless Sensor Networks'', Sensor and Ad Hoc Communications and Networks, (SECON '06). 3rd An.IEEE Comm. Soc. on , vol.1, no., pp.316,325, 28-28 Sept. 2006.

[78] D. V. Manatakis, E. S. Manolakos, A. Roussos, G. Xanthopoulos, D. X. Viegas,``In-silico estimation of the temperature field induced by moving fire. Predictive modelling and validation using prescribed burn data''. Coimbra, Portugal. In Proc. of VI International Conference on Forest Fire Research, November 2010.

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

[80] J.R. Hershey, P.A. Olsen, ``Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models",In Proc. of the IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP 2007), pp.IV-317 - IV-320.

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

[82] C. M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, ISBN 978387310732

[83] S. Theodoridis, K. Koutroubas, *Pattern Recognition*. Academic Press, 4th Edition 2009, ISBN: 9781597492720.

[84] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - A lightweight and flexible operating system for tiny networked sensors. In Proc. of the 29th Annual IEEE Int. Conf. on Local Computer Networks, (LCN 04), pp. 455-462, Washington, DC, USA, 2004.

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

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

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

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

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

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

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

[92] B. Pavkovic , J. Radak, N. Mitton, F. Rousseau, I. Stojmenovic, ``From real neighbors to imaginary destination: emulation of large scale wireless sensor networks'', In Proc. of the 11th int. conf. on Ad-hoc, Mobile, and Wireless Networks (ADHOC-NOW'12), pp. 459-471, isbn: 978-3-642-31637-1.

[93] AVR2016: RZRAVEN Hardware User's Guide: http://www.jm.pl/karty/ATAVRRZRAVEN.pdf Last accessed (20/10/2013).

[94] Contiki 2.6: SICSLoWMAC Implementation: http://contiki.sourceforge.net/docs/2.6/a01788.html Last accessed (10/10/2014)

[95] AVR Dragon Getting Started: http://www.uchobby.com/index.php/2009/03/02/avr-dragon-getting-started/ Last accessed (20/10/2013)

[96] AVR Dragon - Atmel Corporation http://www.atmel.com/tools/avrdragon.aspx. Last accessed (20/10/2013).

[97] E Shih, S Cho, N Ickes, R Min, A Sinha, A Wang and A Chandrakasan ``Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks'', In Proc. of ACM Mobile Computing and Networking (MobiCom 01), pp. 272-287 Rome, Italy.

[98] Raghunathan, V.; Schurgers, C.; Sung Park; Srivastava, M.B., ``*Energy-aware wireless microsensor networks*,'' Signal Processing Magazine, IEEE , vol.19, no.2, pp.40,50, Mar 2002.

[99] D. V. Manatakis, E. S. Manolakos, ``Collaborative Sensor Network algorithm for predicting the spatiotemporal evolution of hazardous phenomena,'' Int. Conf. on Systems Man and Cybernetics (SMC 2011), October 2011, Anchorage-Alaska, pp. 3439-3445.

[100] D. V. Manatakis, E. S. Manolakos, ``Predictive modeling of the spatiotemporal evolution of an environmental hazard and its sensor network implementation'' In Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2011), May 2011, Prague-Czech pp. 2056-2059

[101] D. V. Manatakis, E. S. Manolakos, "Estimating the Spatiotemporal Evolution Characteristics of Diffusive Hazards using Wireless Sensor Networks," Accepted for publication, Parallel and Distributed Systems, IEEE Transactions on, doi: 10.1109/TPDS.2014.2357033, 2014.

[102] L. Piegl, W. Tiller, *The NURBS Book (Monographs in Visual Communication)*, second ed., Springer-Verlag, New York, 1997.

[103] Animation of algorithm's behavior in the presence of diffusive hazard with regular shape:
https://www.dropbox.com/s/nc0wfqfkv3al8k6/20ModelsCircularBoundary.wmv?dl=0

[104] Animation of algorithm's behavior in the presence of diffusive hazard with irregular shape:
https://www.dropbox.com/s/edjb8i5zps9fu4m/20ModelsFLogABoundary.wmv?dl=0