SPRINGER BRIEFS IN ELECTRICAL AND COMPUTER ENGINEERING · COOPERATING OBJECTS

José Ramiro Martínez-de Dios
Alberto de San Bernabé-Clemente
Arturo Torres-González
Anibal Ollero

Cluster-based Localization and Tracking in Ubiquitous Computing Systems

Springer

SpringerBriefs in Electrical and Computer Engineering

SpringerBriefs in Cooperating Objects

Series editor

Pedro José Marron, Fraunhofer IAIS, Duisburg, Germany

More information about this series at http://www.springer.com/series/10208

José Ramiro Martínez-de Dios
Alberto de San Bernabé-Clemente
Arturo Torres-González · Anibal Ollero

Cluster-based Localization and Tracking in Ubiquitous Computing Systems

Springer

José Ramiro Martínez-de Dios
Departamento de Ingeniería de Sistemas y Automática
Universidad de Sevilla
Sevilla
Spain

Arturo Torres-González
Departamento de Ingeniería de Sistemas y Automática
Universidad de Sevilla
Sevilla
Spain

Alberto de San Bernabé-Clemente
Departamento de Ingeniería de Sistemas y Automática
Universidad de Sevilla
Sevilla
Spain

Anibal Ollero
Departamento de Ingeniería de Sistemas y Automática
Universidad de Sevilla
Sevilla
Spain

ISSN 2191-8112 ISSN 2191-8120 (electronic)
SpringerBriefs in Electrical and Computer Engineering
SpringerBriefs in Cooperating Objects
ISBN 978-3-662-54759-5 ISBN 978-3-662-54761-8 (eBook)
DOI 10.1007/978-3-662-54761-8

Library of Congress Control Number: 2017937479

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer-Verlag GmbH Germany
The registered company address is: Heidelberger Platz 3, 14197 Berlin, Germany

To Victoria and Fernando,
to Elisa and Fernando

Preface

Localization and tracking are key functionalities in ubiquitous computing systems and techniques. Most applications in pervasive computing inherently rely on location information, e.g. to make the gathered measurements geographically meaningful. Also, location information can be used to support fundamental network layer services such as clustering, topology control and routing, among others.

In the past few years, the integration of miniaturized GPS receivers in ubiquitous computing devices has greatly improved localization and tracking in outdoor environments providing sufficient accuracy for many applications. Despite these advances, reducing GPS shadows and improving accuracy still require intense R&D effort. However, today the most relevant research topics in localization and tracking focus on indoor and GPS-denied environments. A very high variety of localization approaches, sensors and techniques have been developed. However, none of the proposed schemes is ideal—all have pros and cons—and the selection of the method for a given problem highly depends on the specific requirements and constraints of the application and scenario.

This book briefly summarizes the current state of the art in localization and tracking in ubiquitous computing systems. It is focused on cluster-based schemes, which is probably the most widely adopted architecture. This book analyzes the existing techniques for measurement integration, node inclusion/exclusion in/from the cluster and cluster head selection.

Although significant advances have been performed in the past, many issues still remain open for current and future research. We hope that this book may be helpful to contribute to provide a broader perspective of the localization and tracking problems and may spark new innovative ideas.

Sevilla, Spain
January 2017

José Ramiro Martínez-de Dios
Alberto de San Bernabé-Clemente
Arturo Torres-González
Anibal Ollero

Acknowledgements

The authors thank all members of the Robotics, Vision and Control Group at the University of Seville for their contribution and support in the preparation of this book. The authors are grateful to Ph.D. Adrian Jiménez and Mr. José Manuel Sánchez-Matamoros for their support in the experimentation of the techniques and schemes described in the UBILOC testbed.

Contents

Acronyms

AoA	Angle of Arrival
CH	Cluster Head
CMOS	Complementary Metal Oxide Semiconductor
COTS	Commercial Off-The-Shelf
EIF	Extended Information Filter
EKF	Extended Kalman Filter
GPS	Global Positioning System
IF	Information Filter
IMU	Inertial Measurement Unit
KF	Kalman Filter
LIDAR	Light Detection and Ranging
LQI	Link Quality Indicator
LS	Least Squares
ML	Maximum Likelihood
MSE	Mean Square Error
PDA	Personal Digital Assistant
PF	Particle Filter
PRR	Packet Reception Rate
RBF	Recursive Bayesian Filter
RFID	Radio-Frequency Identification
RGB	Red, Green, Blue
RSSI	Received Signal Strength Indicator
SNR	Signal to Noise Ratio
ToF	Time of Flight
TDoA	Time Difference of Arrival
USB	Universal Serial Bus
UWB	Ultra-Wideband
WCN	Wireless Camera Network

WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1
Introduction

1.1 Localization and Tracking of Cooperating Objects

In the last years a variety of technological fields have emerged in the context of ubiquitous computing systems. Technologies such as smartphones, personal mobile computing devices, camera networks, wearable computers, radio frequency identification (RFID) and Wireless Sensor Networks (WSN) are revolutionizing our world into a ubiquitous environment in which all devices are fully networked.

Ubiquitous computing systems are intended to gather information about the characteristics of the physical world. To make the gathered measurements geographically meaningful it is necessary to associate measurements to locations. Most applications in ubiquitous computing technologies explicitly or implicitly rely on location information. Moreover, location information can be also used to support fundamental network layer services such as clustering, topology control and routing, among many others.

The wide adoption of the Global Positioning System (GPS) has greatly improved localization in outdoors. Miniaturized low-cost GPS receivers integrated in ubiquitous computing devices provide accuracies around 3–5 m, sufficient for many applications. Despite these advances, reducing GPS shadows and improving accuracy in low-cost receivers still require intense R&D effort. However, the most relevant research topics in localization and tracking take place in indoors and GPS-denied environments. A very high variety of sensors, technologies and techniques have been developed for localization in GPS-denied environments in the recent years. However, none of the proposed sensors or techniques is ideal—all have pros and cons—and the selection of sensors and methods for a particular problem highly depends on the specific requirements and constraints of the application and scenario.

This book briefly summarizes the current state of the art in cluster-based localization and tracking in ubiquitous computing systems. This book deals with architectures, schemes and techniques in which a target is localized and tracked by ubiquitous computing systems deployed in the environment. Dead reckoning, odometry-based

J.R. Martínez-de Dios et al., *Cluster-based Localization and Tracking in Ubiquitous Computing Systems*, SpringerBriefs in Cooperating Objects,
DOI 10.1007/978-3-662-54761-8_1

techniques or methods based on sensors carried by the target such as Inertial Measurement Units (IMUs), cameras or LIDARs are out of the scope of this book.

Cluster-based schemes are the most widely employed and researched approaches in target localization and tracking in ubiquitous computing systems. Clustering naturally improves scalability and simplifies communications. Besides, it is efficient in the resources consumed since only the nodes that participate in target localization and tracking are kept active while the rest are left inactive not consuming resources.

Cluster-based tracking requires methods to integrate the measurements gathered by the cluster nodes. Most existing techniques rely exclusively on one type of sensors. Link quality and radio signal strength measurements are by far the most widely used since they are measured naturally by most radio modules in Commercial Off-the-Shelf (COTS) ubiquitous computing devices without incurring in extra hardware or energy consumption costs. However, in indoors these measurements are affected by significant perturbations originated by reflections and other interactions of the radio signals with the environment.

The cluster should track the target as it moves. Cluster-based schemes should deal with the task of deciding which nodes should be included or excluded from the cluster. Nodes are activated when they are invited to participate in the cluster and, they are deactivated when they are excluded. Node inclusion/exclusion methods aim at keeping within the cluster only the nodes that provide useful information trying to reduce the active sensors to the minimum.

The performance of cluster-based target tracking highly depends on which node acts as cluster head. Cluster-based tracking schemes should also deal with the dynamic selection of the most suitable cluster head at each time. The objective is to select which among the cluster nodes is the most suitable one to perform the cluster head role.

This book summarizes the main existing techniques in cluster-based target tracking that deal with the aforementioned three issues: measurement integration, node inclusion/exclusion and cluster head selection.

1.2 Assumptions and Requirements

The techniques, schemes and architectures described in this book are suitable for ubiquitous computing systems comprised of embedded nodes with sensing, processing and communication capabilities that organize autonomously into clusters in order to accomplish target localization and tracking missions. Nodes are designed to be low cost, i.e. engineered to have low energy consumption, endowed with low sensing/actuating resources and low computational capability. It is assumed that nodes and transmissions can fail. Measurements from sensors are assumed also subject to noise.

Each node is assumed endowed with low-energy consumption modes. Each node can enter and get out of its low-energy mode by its own. Besides, each node can put another node in the low-energy node and can wake it up. Most COTS nodes

are endowed with these capabilities and these assumptions do not involve practical constraints.

Nodes are assumed equipped with sensors suitable for tracking such as small cameras. Nodes radio circuitry allows measuring the strength of the radio signal of the received messages (RSSI) and other link quality estimators. Sensor nodes are intrinsically static, although mobile nodes may exist if carried by people, vehicles, robots or even animals. Besides, each anchor node is assumed to know its own location and the parameters of its sensors, such as the orientation of its camera and its RSSI-range model.

In this context technologies such as personal computing devices, smartphones and PDAs or camera networks are also considered as ubiquitous computing systems. The popularization of smartphones and the improvement in their performance and connectivity has boosted research in many applications. Networks of distributed static cameras are widely employed for ubiquitous perception in ambient intelligence systems.

Below the main requirements in our problem are briefly discussed:

- **Efficiency**. Low energy consumption, low computational capability and low communication ranges are inherent to ubiquitous computing nodes. The architecture and techniques involved should consume as low resources as possible including energy, computational resources and communication bandwidth, among others.
- **Accuracy**. The target location and tracking estimation should be as accurate as possible. Improving accuracy often requires higher resource consumption.
- **Modularity**. The architecture and the techniques developed should have a clear modular approach that enables scalability and expandability.
- **Robustness**. Nodes are designed to be low cost. However, in contrast to the individual fragility of each node, the strength of the architecture should originate from the cooperation between nodes. The architecture and methods should be robust to failures of individual nodes and transmission errors.
- **Scalability**. Ubiquitous computing systems exploit the complementarity between nodes in order to improve the global performance and operation robustness. The complexity of the architecture and the techniques should scale with the number of nodes and with the size of the scenario. Also, the system should be able to locate and track simultaneously a high number of targets.
- **Heterogeneity**. (Widely understood) heterogeneity is a crucial point in cooperating nodes. Heterogeneity may affect all levels from physical characteristics to sensing, computing and communication capabilities.

1.3 Book Structure

This book describes architectures, schemes and techniques for cluster-based localization and tracking using measurements of the target taken by heterogeneous sensors

mounted on anchored nodes deployed in the environment, which location and orientation is assumed known.

This book addresses localization and tracking in ubiquitous computing systems from different perspectives. Besides the introductory and concluding chapters, this book is structured in the following five chapters.

Chapter 2 presents the main existing architectures for target localization and tracking in ubiquitous computing systems. This chapter presents a general cluster-based architecture that is used as the main conductor of this book. This architecture comprises modules that deal with the aforementioned issues in cluster-based tracking, namely: measurement integration, inclusion/exclusion of nodes in/from the cluster and selection of the cluster head. This chapter also presents UBILOC (Ubiquitous Localization Testbed), a testbed designed for target localization and tracking in ubiquitous computing systems, where the schemes and techniques described in this book were experimented.

Chapter 3 deals with measurement integration for cluster-based target localization and tracking. First, existing techniques are classified according to the type of physical measurement used. The main existing localization and tracking techniques based on RSSI are briefly described. Cameras are also widely employed, they are more accurate than RSSI-based methods but they involve significantly higher consumption of energy and computational resources. This chapter finally presents efficient probabilistic techniques for the integration of multi-sensor measurements for target localization and tracking based on Extended Information Filters (EIFs).

Chapter 4 deals with node inclusion/exclusion in cluster-based tracking. The objective of these techniques is to reduce the number of active sensors to the minimum by keeping in the cluster only the nodes that provide useful information. This chapter presents the sensor selection problem and summarizes the main sensor selection techniques reported for target localization and tracking. A sensor selection technique based on cost-reward optimization suitable for the general architecture described in Chap. 2 is also presented.

Chapter 5 deals with techniques for cluster head selection. This chapter reviews the main existing techniques. Most of them rely on criteria such as proximity or homogeneity in energy consumption. This chapter also presents a cluster selection technique based on tracking uncertainty minimization that suitable for the architecture described in Chap. 2.

Chapter 2
Architectures for Target Localization and Tracking

2.1 Introduction

This chapter summarizes the main architectures used for target localization and tracking in ubiquitous computing systems. Existing approaches can be coarsely classified into peer-to-peer and hierarchical schemes. In peer-to-peer architectures each node exchanges sensing information with its neighbors in order to reach a consensus on the target status. In hierarchical architectures nodes organize following some hierarchy. Three main hierarchical schemes have been adopted: tree-based, cluster-based and hybrid schemes. Cluster-based schemes are the most widely employed and researched approaches. Clustering naturally solves scalability and efficiency since the measurement flow and the packet interchange are kept within the cluster, simplifying computations and transmissions.

This chapter also presents a general architecture for cluster-based tracking that is used as the main conductor of this book. This architecture comprises three main modules that deal with typical problems in cluster-based tracking. The first one is responsible for measurement integration and target estimation. The cluster evolves as the target moves and it is necessary to dynamically decide the nodes that are included in or excluded from the cluster. Nodes included in the cluster are activated, while nodes excluded from the cluster are deactivated, saving energy. The second module is responsible for deciding which nodes should be included or excluded from the cluster. Clustering-based tracking schemes rely on a cluster head (CH) that acts as the main scheduler of the cluster. The third module of the architecture is responsible for dynamically deciding which node acts as CH.

The experimentation of the schemes and modules shown in this book was performed in UBILOC (Ubiquitous Localization Testbed), a testbed specifically designed for multi-sensor target localization and tracking in ubiquitous computing systems. This chapter briefly presents its main characteristics and components.

J.R. Martínez-de Dios et al., *Cluster-based Localization and Tracking in Ubiquitous Computing Systems*, SpringerBriefs in Cooperating Objects,
DOI 10.1007/978-3-662-54761-8_2

This chapter is structured as follows. The main hierarchical architectures developed for target tracking are presented in Sect. 2.2. The main peer-to-peer architectures are described in Sect. 2.3. The general cluster-based architecture is summarized in Sect. 2.4. UBILOC is briefly described in Sect. 2.5.

2.2 Hierarchical Networks

In the simplest tracking scheme each node in the network is ready for tracking all the time. When a node gathers a measurement of the target, it transmits the measurement to the sink node or to base station, which estimates the target location using the received measurements. This centralized scheme has clear drawbacks. First, it is not efficient in terms of energy because all the nodes in the network are required to be always in a mode ready to track the target. Also, it involves heavy communicational burden at the surroundings of the sink node. Furthermore, robustness is compromised in case of channel congestion or failure of the sink node.

Different schemes have been developed to mitigate the drawbacks of this baseline method. This section briefly summarizes the main hierarchy-based architectures: tree-based, cluster-based and hybrid schemes.

2.2.1 Tree-Based Architectures

The nodes of the network are organized in a hierarchical tree that can be represented as a graph. The nodes that sense the target communicate with each other in order to select a specific node that acts as the root of the tree and collects the information. The root can dynamically change to adapt to the target motion. Although more robust than the aforementioned baseline approach, tree-based schemes still result in high energy consumptions and heavy communications.

One example is STUN (Scalable Tracking Using Networked Sensors) [1]. In STUN each link between two anchor nodes in the tree is assigned with a cost that is proportional to the Euclidean distance between the two nodes. The leaf nodes track the target and send the collected data to the root through intermediate nodes of the tree. The intermediate nodes register the detected targets and send updated information to the root when there is a change.

In DCTC (Dynamic Convoy Tree-Based Collaboration) [2] the tree is rooted at the anchor node that is the closest to the target. The target position is estimated by the location of the root node. In DAT (Deviation Avoidance Tree) [3] tracking is performed in two steps: update and query. Updates are initiated when the target moves to a new location. Update cost is reduced by the deviation avoidance tree algorithm and query cost is reduced by the query cost reduction algorithm in a second step.

DOT (Dynamic Object Tracking) [4] is a scheme that reports the tracking information of moving targets to a moving source. The first stage is to identify the neighbors.

In the second stage, called target discovery, the source sends requests to sensor nodes and the nodes that are close to the target reply. Finally, in the target tracking step, the source sends a query to the beacon node (the node keeping track of the information), which replies with the target next location. Then, the source moves towards the next beacon node.

2.2.2 Cluster-Based Schemes

In these schemes anchor nodes sensing the same target organize into clusters. Each cluster has one node acting as cluster head (CH). Clustering techniques are energetically efficient: nodes not sensing the target are kept in sleep mode, saving energy, during most of the time. Clustering allows several targets to be tracked simultaneously, with a cluster for each target. Cluster-based tracking is particularly interesting in applications that require scalability. There are two main approaches depending on how clusters are created: static and dynamic clustering.

2.2.2.1 Static Clustering

Clusters are formed during the deployment of the network. Nodes are assumed static and for each cluster their members and coverage areas never change [5]. Static clustering is very simple and convenient in many cases but has several problems. It is not robust to failures in cluster head (CH) nodes. Also, different clusters cannot share information or collaborate in measurement integration and processing.

2.2.2.2 Dynamic Clustering

Dynamic clustering offers many interesting features for target localization and tracking. The formation of clusters can be triggered by external events, e.g. the detection of a target. If a node with sufficient battery and computational resources detects the target, it volunteers as cluster head. Other nodes near the CH are invited to become members of the cluster and to report their measurements to the CH.

One example is the so-called information-driven sensor querying technique IDSQ [6] in which the most suitable node to perform the sensing task is determined. This approach assumes that each node can locally estimate the cost of sensing, processing and communicating data with other nodes. It is energy efficient because only a few nodes are active simultaneously at any time.

DELTA [7] is an algorithm for tracking a person moving at constant speed that dynamically organizes a cluster and selects the cluster head, which will be responsible for monitoring the target and for collaborating with the rest of the sensor nodes.

RARE [8] is an energy efficient tracking protocol based on two algorithms, RARE-Area (Reduced Area Reporting) and RARE-Node (Reduction of Active Node

Redundancy). RARE-Area inhibits nodes that are far from the target reducing the number of nodes participating in tracking. RARE-Node identifies overlapping sensors in order to reduce the redundant information. The cluster is created dynamically using predictions computed during target tracking.

In DTSC [9] clusters are created by interchanging packets between the nodes that have detected the target. If a node has received packets from at least 4 nodes it declares itself as CH. DSTC can fail if the density is too low or if the cluster size is too large.

2.2.3 Hybrid Architectures

Hybrid schemes have characteristics from the two aforementioned approaches.

One example is DPT (Distributed Predictive Tracking) [10], which adopts a clustering approach and a prediction-based tracking technique. DPT is robust against prediction failures and target losses.

DCAT (Dynamic Clustering for Acoustic Tracking) [11] forms clusters using Voronoi diagrams. The CH asks its neighbors to join the cluster by broadcasting invitation packets. Nodes decide if they should reply to the CH after analyzing the distance to the target, which is estimated probabilistically. The CH determines the location using their replies and transmits the results to the sink.

In HPS (Hierarchical Prediction Strategy) [12] clusters are also created using Voronoi diagrams and the next location of the target is predicted using a technique based on Least Squares.

2.3 Peer-to-Peer Networks

In hierarchical networks many nodes are involved in sensing at the same time but only one acts as scheduler. The scheduler takes the greater part of the communication and computational burden resulting in inhomogeneous consumption of resources. In contrast, in peer-to-peer architectures all nodes have the same role and target tracking estimation is performed using consensus techniques in which each node interchanges its measurements with its neighbors [13].

Tracking in peer-to-peer networks is often based on average consensus algorithms. These algorithms perform successive refinements of local estimates maintained by individual nodes. The main objective of consensus filters is to estimate the global information contribution using only local and neighboring information [14]. Each iteration in the consensus filter has two main steps. In the first one, called communication step, each node interchanges information with its neighbors. In the second, called update step, each node uses the information gathered in the communication step in order to refine its previous estimate. Consensus filters are completely distributed and can be applied in large-scale sensor networks.

A scalable distributed target location and routing architecture for wide-area peer-to-peer applications is presented in [15]. Work [16] proposes a target tracking system based on the auto regressive moving average model in a distributed peer-to-peer signal processing framework. Sensor nodes act as peers that perform target detection, feature extraction, classification and tracking. It also includes a distributed peer-to-peer signal processing framework that considers the trade-off between tracking accuracy and energy consumption.

In general Recursive Bayesian Filters (RBFs), which are widely-applied in hierarchical-based tracking, are not applied to peer-to-peer networks due to their implementation complexity. However, a number of schemes that combine Kalman Filters and consensus filters have been developed, see e.g. [17, 18].

2.4 A General Cluster-Based Scheme for Localization and Tracking

This chapter presents a general architecture for cluster-based tracking that is used as the main conductor of this book.

Cluster-based schemes is by far the most widely employed and researched approach in target localization and tracking in ubiquitous computing systems. All the nodes that sense the same target organize into a cluster with a cluster head (CH) that schedules its operation. The tracking measurement flow and packet interchange are kept within the cluster, simplifying packet transmission. Each target being tracked has its own cluster and a number of targets can be tracked simultaneously. Of course, only the nodes that participate in target tracking are active: the rest are inactive—in low-energy mode—not consuming resources.

Cluster-based tracking requires methods to integrate the measurements gathered by the cluster nodes. Using different types of sensors for target tracking originates synergies of interest in many applications. The estimation and measurement fusion techniques adopted should allow flexible integration of measurements from sensors of different types. Measurement fusion is usually resource demanding, hence the method adopted should be efficient and prevent inhomogeneous consumption of computational, energy or bandwidth resources between the cluster nodes. Scalability, robustness to sensor failures and packet loss are other properties of interest in measurement integration methods.

Of course, the cluster should track the target as it moves. Then, methods should deal also with the task of deciding the inclusion/exclusion of nodes from the cluster. The objective is to keep within the cluster only the nodes that provide useful information for target tracking trying to reduce the active sensors to the minimum. Nodes are activated when they are invited to participate in the cluster and, they are deactivated when they are excluded from the cluster.

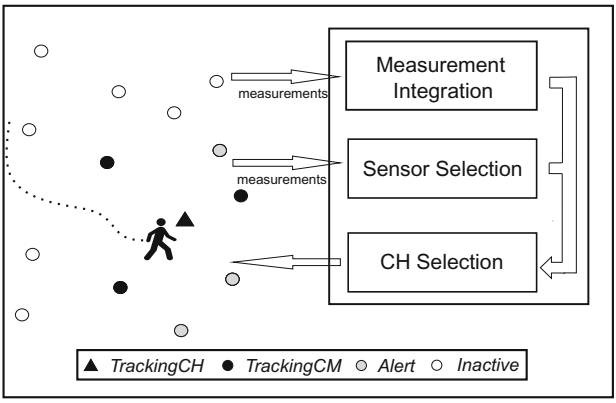


Fig. 2.1 General cluster-based architecture for target tracking

The performance of cluster-based target tracking highly depends on which node acts as cluster head. Cluster-based tracking schemes should also deal with the dynamic selection of the most suitable cluster head.

2.4.1 Architecture

A general cluster-based architecture for target localization and tracking is shown in Fig. 2.1. The nodes can be at different modes depending on the level of involvement in tracking. At time t any node is at one of the following tracking modes:

- *TrackingCH*, the node acts as cluster head—represented by a triangle in Fig. 2.1;
- *TrackingCM*, represented as a black circle, the node participates in the cluster gathering measurement but not as head;
- *Alert*, represented as a gray circle, the node is not currently participating in the cluster but it is at single-hop distance from the head and could be included in the cluster at that time if necessary;
- *Inactive*, it is not involved in tracking and cannot be included in the cluster at that time.

Nodes in modes *trackingCH* and *trackingCM* participate in the cluster and altogether perform the target location and tracking estimation. Nodes in *alert* and *inactive* are not currently participating in tracking but those in *alert* could be invited to enter the cluster at that time t.

This architecture is composed of the following main modules:

- *Module1*: multi-sensor measurement integration. It integrates the measurements of the target gathered by the different cluster nodes. It should allow distributed

implementation. The main existing measurement integration techniques for target localization and tracking are presented in Chap. 3.
- *Module2*: sensor node activation/deactivation. Its objective is to keep active only the nodes that provide useful information for tracking trying to reduce to the minimum the number of active sensors. The main existing techniques are presented in Chap. 4.
- *Module3*: cluster head selection. The objective is to dynamically select which of the nodes within the cluster is the most suitable to perform the cluster head role. The main existing techniques are presented in Chap. 5.

Below, the main requirements of the architecture and its modules are summarized:

- **Computational distribution**. Each node within the cluster should actively participate in tracking. All nodes should gather measurements and perform computations that are used for measurement integration, node inclusion/exclusion and cluster head selection.
- **Efficiency**. The nodes should employ efficient techniques and exploit computation reuse resulting in low computational burden and energy consumption.
- **Scalability**. The consumption of resources from each cluster node should be constant or almost constant regardless of the cluster size.
- **Flexibility**. Each module of the architecture is responsible for addressing a main issue in the cluster-based scheme. Each module can be implemented with different techniques without affecting the rest of the modules. The architecture is flexible also in the sensors used for tracking.
- **Extensibility**. The architecture can be extended with other modules such as sensor calibration methods, which benefit from the synergies between measurements in order to supervise if sensors are calibrated and recalibrate them if necessary.

2.4.2 Schemes

The general architecture depicted in Fig. 2.1 can be adapted to different tracking schemes that use different types of sensors:

- *Scheme1*: Efficient tracking using only camera measurements,
- *Scheme2*: Efficient tracking using only RSSI measurements,
- *Scheme3*: Efficient tracking using camera and RSSI measurements.

The main difference between them is the sensors employed. In *Scheme1* each node is assumed equipped with a camera and tracking uses only camera measurements. Each camera node is assumed with sufficient computational capacity to execute simple image processing methods and measure the coordinates of the target center in its image plane.

Scheme2 performs tracking employing only RSSI measurements. It assumes that the target is tagged with a mobile node and that each anchor node is capable of

Fig. 2.2 Picture of the UBILOC testbed during an experiment

collecting RSSI measurements from the target node. *Module1* integrates RSSI measurements and *Module2* decides which nodes should gather and integrate RSSI measurement for localization and tracking.

Scheme3 is an extension of *Scheme1* that also integrates RSSI measurements. *Scheme3* naturally employs the RSSI measurements that can be measured from the packets actually interchanged in *Scheme1*, without requiring any more transmissions. *Scheme3* assumes that the target is tagged. Nodes sensing the same target—either using the camera or with RSSI– organize autonomously into clusters. In *Scheme3*, *Module1* integrates camera and RSSI measurements and *Module2* decides which camera nodes are included/excluded in/from the cluster and with which measurements these camera nodes contribute to target tracking.

2.5 UBILOC: Ubiquitous Localization Testbed

The experimentation of the aforementioned schemes and modules was performed in UBILOC (Ubiquitous Localization Testbed), a testbed specifically designed for localization and tracking in ubiquitous computing systems, see Fig. 2.2. UBILOC was developed on top of the *CONET Integrated Testbed* [19],[1] a testbed designed for remote experiments with Cooperating Objects [20], and inherits from it its main characteristics.

UBILOC emulates a typical office smart environment scenario and includes sensors widely used for localization in smart environments: a network of cameras for camera-based localization, a Wireless Sensor Network for radio-based localization and a network of Time-of-Flight (ToF) sensors. These heterogeneous devices are integrated using a modular and flexible architecture. It also includes a set of mobile

[1] http://conet.us.es.

Fig. 2.3 Nodes of the WCN: each node is composed of a CMUcam3 module and a WSN node

robots that are used as targets to be localized and tracked in the experiments. UBILOC is set in a room of more than 500 m^2 (22 m × 24 m) in the basement of the School of Engineering of Seville.

Among others, UBILOC includes a Wireless Camera Network (WCN). Each node of the WCN is composed by a *CMUcam3* camera module [21] connected to a *TelosB* WSN node, see Fig. 2.3. *CMUcam3* camera modules are endowed with embedded programmable image processing capabilities. Each *CMUcam3* module captures RGB images of 352 × 288 pixels and applies simple image processing methods. The results of the local image processing methods executed at each camera are transmitted to the *TelosB* node using a simple bidirectional protocol.

In UBILOC robots are used as targets to be localized and tracked. The advantages of using robots as targets instead of humans are: (a) they allow fully unattended experiments; (b) higher repeatability of experiments; and (c) robots include tools to determine their ground truth location, which is necessary to measure the errors provided by the localization techniques tested. A total of 5 Pioneer 3-AT robots are used. Each robot is equipped with one 2D laser range finder and one Microsoft Kinect and an IEEE 802.11 Wireless bridge. Each robot is capable of accurately computing its own location and orientation using the Adaptative Monte Carlo Localization algorithm (AMCL) [22]. These estimates are taken as the ground truth for indoor experiments. In outdoors the ground truth pose is obtained using GPS receivers and Inertial Measurement Unit (IMU).

UBILOC inherits its architecture based on Player [23] from the *CONET Integrated Testbed*. Player makes available user-transparent inter-module communication using standard interfaces. It is based on a modular client/server architecture. The Player Server interacts with the hardware elements and uses abstract interfaces to communicate with the Player Client, which provides access to all the system elements through device-independent APIs.

UBILOC provides infrastructure for experimentation support including high-level abstract interfaces to hardware and also the communication between processors. Users only have to program the modules they want to test. This architecture allows

a high range of localization experiments following centralized and decentralized schemes.

The experiments of *Scheme1* and *Scheme3* were performed with the WCN. The experiments of *Scheme2* were performed using the WSN.

2.5.1 Main Characteristics of the Hardware Components

CMUcam3 uses an ARM7TDMI based fully programmable embedded computer vision sensor. The main processor is a Philips LPC2106 connected to an Omnivision CMOS camera sensor module. Custom C codes can be developed using GNU toolchain along with a set of open source libraries. Executables can be flashed onto the board using the serial port with no external hardware required. Their main energy consumption characteristics are summarized in Table 2.1.

TelosB is a well-known ultra low-power WSN module. *TelosB* uses industry standards like USB and IEEE 802.15.4 to interoperate with other devices. It integrates humidity, temperature and light sensors; provides flexible interconnection with peripherals; and enables a wide range of applications. *TelosB* Revision B is a replacement for *Moteivs Revision A* design. The main operating conditions and energy consumptions of *TelosB* are summarized in Table 2.2.

Table 2.1 Energy consumption of several components of the CMUcam3 module

Power state	Active current (mA)	Idle current	Voltage (V)
All active	130	25	5
CPU (60 MHz)	30	10 μA	1.8
CMOS camera	25	10 μA	5

Table 2.2 Typical operating conditions and energy consumptions of *TelosB* nodes

	Min	Nominal	Max	Unit
Supply voltage	2.1		3.6	V
Current consumption: MCU on, Radio RX		21.8	23	mA
Current consumption: MCU on, Radio TX		19.5	21	mA
Current consumption: MCU on, Radio off		1800	2400	μA
Current consumption: MCU idle, Radio off		54.5	1200	μA
Current consumption: MCU standby		5.1	21.0	μA

2.6 Conclusions

This chapter presented the main architectures for target localization and tracking in ubiquitous computing systems. They can be coarsely classified into hierarchical-based and peer-to-peer based. Clustering-based architecture is the most widely used scheme for localization and tracking.

This chapter also presented a general cluster-based architecture that will be used as the main conductor of the book. This architecture comprises three basic modules that deal with three issues of cluster-based localization and tracking: (1) measurement integration and estimation, (2) dynamic inclusion/exclusion of nodes from the cluster and (3) dynamic selection of the cluster head. These three modules are presented respectively in Chaps. 3, 4 and 5.

The presented architecture is instantiated in three different schemes for target localization and tracking using: only camera measurements (*Scheme1*), only RSSI measurements (*Scheme2*) and integrating camera and RSSI measurements (*Scheme3*).

This section also briefly presented UBILOC, a testbed designed for localization in ubiquitous computing systems, where the experiments described in this book were performed.

References

1. Kung HT, Vlah D (2003) Efficient location tracking using sensor networks. In: Proceedings of the IEEE wireless communications and networking (WCNC'2003), vol 3, pp 1954–1961
2. Zhang W, Cao G (2004) Dctc: dynamic convoy tree-based collaboration for target tracking in sensor networks. IEEE Trans Wireless Commun 3(5):1689–1701
3. Lin CY, Peng WC, Tseng YC (2006) Efficient in-network moving object tracking in wireless sensor networks. IEEE Trans Mob Comput 5(8):1044–1056
4. Tsai HW, Chu CP, Chen TS (2007) Mobile object tracking in wireless sensor networks. Comput Commun 30(8):1811–1825
5. Fayyaz M (2011) Classification of object tracking techniques in wireless sensor networks. Wirel Sens Netw 3:121
6. Zhao F, Shin J, Reich J (2002) Information-driven dynamic sensor collaboration. IEEE Sig Process Mag 19:61–72
7. Wälchli M, Skoczylas P, Meer M, Braun T (2007) Distributed event localization and tracking with wireless sensors. In: Proceedings of the wired/wireless internet communications. Springer, pp 247–258
8. Olule E, Wang G, Guo M, Dong M (2007) Rare: an energy-efficient target tracking protocol for wireless sensor networks. In: Proceedings of the IEEE international conference on parallel processing workshops (ICPPW 2007), p 76
9. Phoha S, Jacobson N, Friedlander D, Brooks R (2003) Sensor network based localization and target tracking through hybridization in the operational domains of beamforming and dynamic space-time clustering. In: Proceedings of the IEEE global telecommunications conference (GLOBECOM'03), vol 5, pp 2952–2956
10. Yang H, Sikdar B (2003) A protocol for tracking mobile targets using sensor networks. In: Proceedings of the IEEE international workshop on sensor network protocols and applications, pp 71–81

11. Chen WP, Hou JC, Sha L (2004) Dynamic clustering for acoustic target tracking in wireless sensor networks. IEEE Trans Mob Comput 3(3):258–271
12. Wang Z, Li H, Shen X, Sun X, Wang Z (2008) Tracking and predicting moving targets in hierarchical sensor networks. In: IEEE proceedings of the international conference on networking, sensing and control (ICNSC'2008), pp 1169–1173
13. Scherber DS, Papadopoulos HC (2005) Distributed computation of averages over ad hoc networks. IEEE J Sel Areas Commun 23(4):776–787
14. Xiao L, Boyd S (2004) Fast linear iterations for distributed averaging. Syst Control Lett 53(1):65–78
15. Rowstron A, Druschel P (2001) Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Lecture Notes in Computer Science, vol 2218. Springer, Heidelberg, pp 329–350
16. Wang X, Wang S, Bi DW, Ma JJ (2007) Distributed peer-to-peer target tracking in wireless sensor networks. Sensors 7(6):1001–1027
17. Zhou Y, Li J (2009) Distributed sigma-point kalman filtering for sensor networks: dynamic consensus approach. In: Proceedings of the IEEE international conference on systems, man and cybernetics (SMC'2009), pp 5178–5183
18. Olfati-Saber R (2005) Distributed kalman filter with embedded consensus filters. In: Proceedings of the IEEE conference on decision and control and european control conference (CDC-ECC'05), pp 8179–8184
19. Jiménez-González A, Martínez-de Dios JR, Ollero A (2011a) An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. Sensors 11(12):11,516–11,543
20. Marrón PJ, Karnouskos S, Minder D, Ollero A (2011) The emerging domain of Cooperating Objects. Springer
21. Rowe A (2012) Cmucam3 datasheet. http://www.cmucam.org/
22. Fox D, Burgard W, Dellaert F, Thrun S (1999) Monte carlo localization: efficient position estimation for mobile robots. AAAI/IAAI 1999:343–349
23. Gerkey B, Vaughan RT, Howard A (2003) The player/stage project: tools for multi-robot and distributed sensor systems. In: Proceedings of the international conference on advanced robotics, vol 1, pp 317–323

Chapter 3
Measurement Integration for Localization and Tracking

3.1 Introduction

This chapter describes existing measurement integration techniques for cluster-based target localization and tracking in ubiquitous computing systems.

The chapter is divided into four parts. In Sect. 3.2 existing techniques are classified according to the type of the physical measurement used. There are schemes based on distance, angle, area or geometric relationships, hop count and neighborhood relationships. Most of them are considered active because they require direct collaboration with the target being tracked, e.g. interchanging packets. In contrast, passive techniques are those that do not require any collaboration from the target.

Most localization and tracking schemes in ubiquitous computing systems integrate measurements of the signal strength of incoming packets, the Received Signal Strength Indicator (RSSI). A wide variety of RSSI-based methods have been developed adopting different measurement integration techniques. The main methods are briefly surveyed in Sect. 3.3.

Cameras are also widely employed for target tracking in ubiquitous computing systems. Camera-based methods are usually more accurate than RSSI-based methods but they involve significantly higher computational burden and energy consumption. Besides, some few schemes integrate RSSI and camera measurements mainly adopting approaches that switch between the two types of measurements depending on their availability in order to reduce energy consumption. Section 3.4 is devoted to camera-based tracking methods.

Finally, efficient probabilistic techniques for the distributed integration of multisensor measurements for target localization and tracking are described in Sect. 3.5.

J.R. Martínez-de Dios et al., *Cluster-based Localization and Tracking in Ubiquitous Computing Systems*, SpringerBriefs in Cooperating Objects,
DOI 10.1007/978-3-662-54761-8_3

3.2 Classification Attending to the Type of Measurements

Most existing techniques are active in the sense that the targets are assumed tagged and there is some kind of cooperation between the target and the static—anchor—nodes. The location of the target is inferred from the measurements of the target gathered by the anchor nodes, which location is assumed known. According to the type of physical measurement, tracking techniques can be classified into: distance, angle, area, hop count and neighborhood relations.

3.2.1 Distance Measurements

A tagged target can be localized and tracked using measurements of the distance it has to a number of anchor nodes which location is assumed known. Below the main sensors used in ubiquitous computing systems are summarized.

3.2.1.1 Radio Signal and Link Quality

These techniques rely on the fact that radio signals attenuate as they propagate. Different radio signal metrics can be used to estimate range. Some of them can be directly obtained from the hardware of radio modules, requiring negligible computation overhead, delay or energy consumption. Received Signal Strength Indicator (RSSI), Link Quality Indicator (LQI) and Signal-to-Noise Ratio (SNR) are some examples. A survey of link quality estimators can be found in [1, 2].

The radio signal measurement most widely-used in localization and tracking in ubiquitous systems is RSSI. RSSI relies on the fact that the strength of the radio signals attenuates with distance. A node receiving a packet can measure its signal strength (RSSI) and use it to estimate the distance to the emitter. Equation (3.1) shows a widely used model that relates the received power strength $P(d)$ in dBm based on the distance to the transmitter [3]:

$$P_r(d) = P_0(d_0) - 10n_p \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma, \tag{3.1}$$

where $P_0(d_0)$ is the strength of the transmitter, n_p is the path loss exponent, that estimates the rate at which the radio strength decreases with distance. X_σ is a Gaussian random variable with zero mean and standard deviation σ, that represents the random effect originated by fading. Both n_p and σ depend on the environment surrounding the emitter and the receiver. Interactions of the radio signal with the environment such as reflections, multi-path, shadowing and path-loss effects affect RSSI measurements. Hence, the accuracy of this model depends of the particular conditions of the environment.

It is also possible to obtain the RSSI-range model experimentally by fitting using regression the RSSI and range measurements. In these cases the above model is simplified, as in [4], which adopts the following expression:

$$rssi(d) = A \log d + B, \tag{3.2}$$

where A and B are the parameters of the model obtained by regression.

RSSI-based localization and tracking has been extensively researched over the years. These techniques are energetically and economically inexpensive. However, interactions of the radio signal with the environment particularly in indoor settings originate inaccuracies in the model, disturbing the accuracy of RSSI-based localization techniques.

Some of the main RSSI-based localization techniques are analyzed in Sect. 3.3. Two main groups of techniques are discussed: range-based methods, which rely on RSSI-range models to obtain distance measurements that are integrated using different approaches; and range-free techniques, which use RSSI measurements to establish geometric or connectivity relationships.

3.2.1.2 Time of Flight

Time-of-Flight (ToF) sensors compute the range between the target and anchor nodes using the propagation delay between a transmitter and a receiver, assuming that the propagation speed is known. They can be further classified between one-way ranging and two-ways ranging methods. The former require synchronization between the transmitter and the receiver, while the latter measure the delay between the transmission of a signal and the reception of the response and do not require synchronization [5]. Besides radio signals, lasers and ultrasounds are very commonly used in ToF-based localization techniques.

3.2.1.3 Time Difference of Arrival

Time Difference of Arrival (TDoA) techniques take advantage of the combination of ultrasound/acoustic and radio signals in order to estimate distance by measuring the time difference between the arrival of these signals. Each node must be equipped with a speaker and a microphone but synchronization is not required.

The idea of TDoA ranging is simple, see Fig. 3.1. An emitter node transmits a radio signal, waits a fixed time t_{delay} and sends an acoustic or ultrasound signal pattern with its speaker. If a node receives the radio signal, then it registers the current time, t_{radio} and turns on its microphone. When the receiver detects the acoustic signal, it registers the current time, t_{sound}. The distance between the transmitter and the receiver can be computed as:

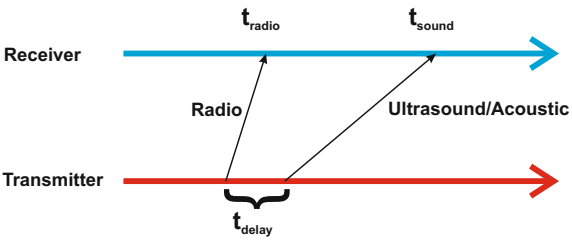


Fig. 3.1 TDoA ranging

$$d = \frac{\nu_{radio}\nu_{sound}}{\nu_{radio} - \nu_{sound}}(t_{sound} - t_{radio} - t_{delay}), \tag{3.3}$$

where ν_{radio} and ν_{sound} are respectively the speed of propagation of radio and sound signals.

Radio propagation is much faster than sound propagation. Thus, Eq. (3.3) is often simplified as $d = \nu_{sound}(t_{sound} - t_{radio} - t_{delay})$.

Techniques based on TDoA can achieve high accuracies. For example, works such as [6] and the cricket system [7] claim accuracies of few centimeters over ranges of several meters. TDoA techniques require line-of-sight communication between the transmitter and receiver, which is not always possible in many environments. Also, each node should be equipped with a speaker and a microphone.

3.2.2 Angle Measurements

Target localization can be inferred from angle measurements using geometry and trigonometry considerations. Below the main approaches researched in ubiquitous computing systems are summarized.

3.2.2.1 Angle of Arrival

In Angle of Arrival (AoA) techniques each node gathers the measurement using radio or microphone arrays, which allows a receiver to determine the direction of the emitter. The target usually transmits signals that are received by the microphones of anchor nodes. The angle of arrival is obtained by each node analyzing the phase between the signal's arrival in each microphone. AoA techniques, such as [8, 9], can obtain accuracies of few degrees.

Each node must be equipped with an array of several microphones. Thus, AoA-based techniques are more difficult to implement in small devices than RSSI-based techniques. AoA techniques are usually more accurate than RSSI techniques but have similar accuracies to TDoA techniques.

3.2.2.2 Camera-Based Techniques

Each node is equipped with a camera. By applying image processing algorithms each node can measure the angle at which it sees the target in the image.

Sensor network nodes have significant constraints in bandwidth and computational capabilities. Thus, the cameras connected to the nodes are usually capable of executing simple image processing techniques so that they can obtain the coordinates of the target in their image plane. From the node perspective, the camera performs just as another sensor that provides the angle where the target is observed.

Camera measurements provide passive tracking with accuracies of few centimeters. Each node should be equipped with a camera. Having a camera active and processing their images requires significant energy consumption and computational burden. Efficiency is critical in these techniques. The main existing localization and tracking schemes with camera measurements are analyzed in Sect. 3.4.

3.2.3 Area Measurements

These techniques frequently use geometric shapes to represent the links or bounds between the anchor nodes and the target and, estimate the target location by constraining the area where the target can be. The intersection of all overlapping areas is computed and, its centroid is chosen as the estimate of the target location, see Fig. 3.2. Depending on how the area is determined, these techniques can be divided into single-reference area estimation and multi-reference area estimation.

3.2.3.1 Single-Reference Area Estimation

They estimate the target location by overlapping regions computed using the RSSI measurements interchanged between the target and anchor nodes. These regions can

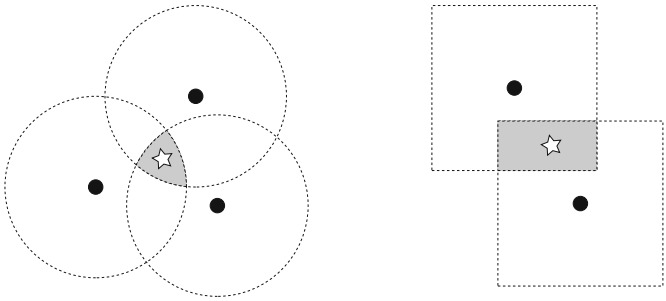


Fig. 3.2 Localization using intersection of *circular* (*left*) and *square* (*right*) areas

have different shapes depending on the type of measurement. In range measurements these regions are usually circles of radius R_{max} or annuli with minimum and maximum radii R_{min} and R_{max}, respectively. In angle—bearing—measurements, they usually employ cones within an angular sector $(\theta_{min}, \theta_{max})$ and range R_{max}. The location of the target is computed as the centroid of the overlapping regions.

One example of a simple implementation of this approach is the bounding box algorithm [6]. The bounding box of an anchor node located at (x_i, y_i) is a square of edge size d_i that is centered at (x_i, y_i). The intersection of bounding boxes can be easily computed, being suitable for nodes with low computational capabilities. This algorithm can be implemented in a distributed way but it is highly sensitive to measurement noise.

3.2.3.2 Multi-reference Area Estimation

One example of this approach is APIT [10], which defines each region by the triangle formed by three anchor nodes. APIT consists of two processes: the triangle intersection and the PIT test. The target originates a number of triangles created by three arbitrary anchor nodes. The target decides if it is inside every triangle by the PIT test. Then, the centroid of the intersection of the triangles that contain the target is selected as the target location, see Fig. 3.3.

The PIT test is based on a geometric property of triangles: if a triangle is formed by points A, B and C, one point D is outside triangle ABC if there is a point adjacent to D that is simultaneously further or closer to points A, B and C. Otherwise, it is inside the triangle.

3.2.4 Hop Count Measurements

These techniques are based on the fact that if two nodes can communicate, their distance is lower than R, the maximum radio range. The connectivity information

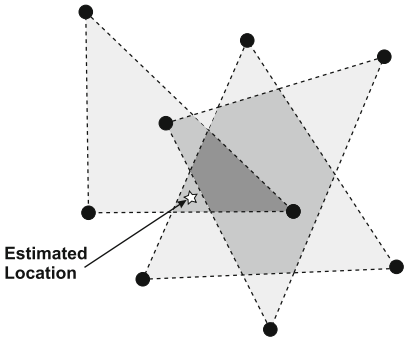


Fig. 3.3 Localization using APIT

defines a graph, in which the vertices are anchor nodes and the edges represent links between nodes. The length of the shortest path between anchor nodes i and j is the hop count $h_{i,j}$. The distance $d_{i,j}$ between i and j must be lower than $R\,h_{i,j}$.

A better estimate of $d_{i,j}$ can be obtained if the expected number of neighbors per node n_{local} is known. Then, instead of using R, the distance of one radio hop can be estimated as:

$$d_{hop} = R\left(1 + e^{-n_{local}} - \int_{-1}^{1} e^{-(n_{local}/\pi)\arccos t - t\sqrt{1-t^2}}\,dt\right) \tag{3.4}$$

Thus, the distance between anchor nodes i and j can be estimated as $d_{i,j} = h_{i,j}\,d_{hop}$. As demonstrated in [11], Eq. (3.4) is accurate if n_{local} is higher than 5 but loses accuracy if n_{local} is higher than 15.

Hop count can be an effective alternative if there are hardware and energy limitations in the nodes but it requires high node density to be accurate. Furthermore, techniques based on hop count would fail if nodes are not deployed uniformly or in networks with coverage holes.

3.2.5 Neighborhood Size Measurements

These schemes use simple connectivity measurements to obtain the target localization estimates. The following three methods are summarized: single neighbor proximity, k-neighbor proximity and ID-CODE.

3.2.5.1 Single Neighbor Proximity

In this basic approach anchor nodes are deployed to avoid or reduce overlapping of their radio coverage areas. Anchor nodes periodically transmit beacon packets that contain their identifier and their locations. Then, the target estimates its location using the locations of the anchor nodes from which it received beacons. Alternatively, if the target emits beacons, the anchor nodes that receive the beacon locate the target using its own location.

3.2.5.2 K-Neighbor Proximity

In this case the coverage areas of different anchor nodes overlap. Several anchor nodes can receive the beacon transmitted by the target. Thus, the target can self-localize with more accuracy than in the single neighbor approach. If there are k anchor nodes detected within the range of the target, the target location (x_0, y_0) can be computed as:

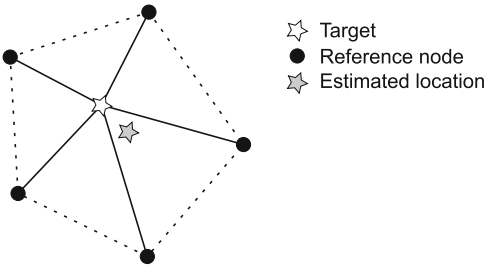


Fig. 3.4 Localization based on k-neighbor proximity. The *red dot* represents the estimated position of the target, which is computed as the centroid of the polygon formed by the anchor nodes

$$x_0 = \frac{1}{k}\sum_{i=1}^{k} x_i, \qquad y_0 = \frac{1}{k}\sum_{i=1}^{k} y_i, \tag{3.5}$$

where (x_i, y_i) is the location of anchor node i.

The estimated location of the target is the centroid of the polygon formed by the anchor nodes, as shown in Fig. 3.4.

This approach provides higher accuracy than the simple neighbor approach and it is also computational efficient. However, taking the centroid of the polygon as the target estimate can lead to large errors in sparse networks.

3.2.5.3 ID-CODE

ID-CODE [12] also uses the overlapping regions to locate the target but in this case the deployment of anchor nodes is planned to ensure that each location is covered by a unique set of anchor nodes. However, as ubiquitous computing systems are usually composed by a large number of nodes, the planned deployment is not usually possible, and the computational burden can be very high.

3.2.6 Discussion

The different types of physical measurements used in localization and tracking in ubiquitous computing technologies are compared in Table 3.1. Some approaches achieve high accuracy but also involve high cost in terms of hardware, energy consumption or computational burden. Also, some techniques are not very accurate but they do not require any additional hardware and can be efficiently implemented. Each approach can be suitable for different cases and applications.

There is not a perfect type of measurement for target tracking. Inexpensive measurements do not provide sufficient accuracy and measurements that provide accuracy require expensive additional hardware devices. The combination of different measurements seems necessary to exploit their synergies.

Table 3.1 Performance comparison of the approaches using different types of physical measurements

Measurement	Hardware cost	Comput. burden	Accuracy
RSSI	Low	Low	Medium
ToF	High	Low	High
TDoA	High	Low	High
AoA	High	Low	High
Camera-based	High	High	High
Single reference area	Medium	Medium	Medium
Multi reference area	Medium	High	Medium
Hop count	Low	Medium	Medium
Single neighbor	Low	Low	Low
Multi-neighbor	Low	Low	Low
ID-CODE	Low	High	Low

3.3 Localization and Tracking Using RSSI

RSSI is by far the most widely researched and applied measurement in localization and tracking in sensor networks. RSSI-based techniques have been traditionally classified into range-based and range-free methods. In range-based techniques RSSI measurements are converted to distance and can be integrated to obtain a location estimate using multilateration, least squares, maximum likelihood or using Recursive Bayesian Filters (RBFs), among others. Range-free methods use RSSI measurements to establish geometry relations. In contrast to range-based techniques, they do not need RSSI-range models and hence are not affected by errors or uncertainties in these models being in general more robust to the influence of the environment. However, range-free methods have lower intrinsic resolution and in general do not perform as well as range-based methods.

Most RSSI-based techniques require communication between the target and the anchor nodes. There are few passive techniques, which in general rely on the disturbances in RSSI originated by the presence of the target. For example, [13] presents a technique to detect the presence of humans using the disturbances in the RSSI measurements between nodes.

Most RSSI localization measurements can be obtained using different wireless communication technologies like Bluetooth [14], RFID (Radio-frequency Identification) [15], ZigBee [16], UWB (Ultra-Wideband) [17], Wi-Fi [18] or WiMAX (Worldwide Interoperability for Microwave Access) [19]. In general, Bluetooth, UWB and RFID are intended for WPAN (Wireless Personal Area Network) communications and for localization in ranges of about 10 m, while Wi-Fi and WiMAX are oriented to WLAN (Wireless Local Area Network) involving localization ranges of about 100 m [20].

Below the main RSSI-based localization and tracking techniques are summarized.

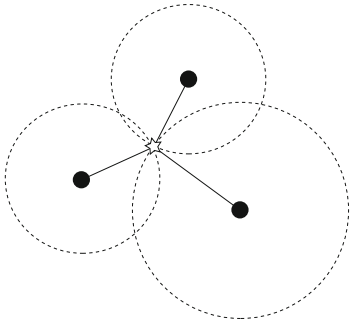


Fig. 3.5 Multilateration in one example

3.3.1 Multilateration

Multilateration [21, 22] is one of the simplest techniques to estimate the target location assuming that range measurements between the target and several anchor nodes are available. The target measures the RSSI of the packets emitted by anchor nodes. The target computes its location by the intersection of the circles centered at the anchors with radii equal to the estimated distances, as shown in Fig. 3.5.

The result of multilateration is a unique position as long as the distance measurements are perfect (noiseless). There are implementations for non perfect RSSI measurements that analyze target location solutions obtained using pairs of anchor nodes. Each pair provides two intersections: one correct and one incorrect. All the pairs will create a set of intersections. The correct solutions distribute in a cluster that can be easily identified. The final estimation is the mean of the coordinates of the intersections in the identified cluster.

3.3.2 Least Squares

The Least Squares (LS) method is widely employed to find an approximate solution in overdetermined systems of equations. LS can be applied to RSSI-based localization as follows. Let $X = [x\, y]^T$ be the target 2D position to be determined and $X_l = [x_l\, y_l]^T$ the known coordinates of the l-th anchor node $l = 1, 2, \ldots, L$, where L is the number of anchor nodes gathering RSSI measurements from the target. The distance between the target and anchor node l, denoted by d_l, is:

$$d_l = \sqrt{(x - x_l)^2 + (y - y_l)^2} \tag{3.6}$$

Using a new variable, range R, $R = x^2 + y^2$ and squaring both sides of Eq. (3.6), it results:

$$-2x_l x - 2y_l y + R = d_l^2 - x_l^2 - y_l^2 \tag{3.7}$$

Equation (3.7) can be represented as $A\theta = b$, where:

$$A = \begin{bmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_L & -2y_L & 1 \end{bmatrix}, \quad \theta = \begin{bmatrix} x\ y\ R \end{bmatrix}^T, \quad b = \begin{bmatrix} d_1^2 - x_1^2 - y_1^2 \\ d_2^2 - x_2^2 - y_2^2 \\ \vdots \\ d_L^2 - x_L^2 - y_L^2 \end{bmatrix} \tag{3.8}$$

Thus, the Least Squares method computes $\hat{\theta}$ as follows:

$$\hat{\theta} = (A^T A)^{-1} A^T b \tag{3.9}$$

The estimated location of the target $\hat{X} = [\hat{x}\ \hat{y}]^T$ can be easily obtained from $\hat{\theta}$. Least Squares is optimum in case of measurements with Gaussian noise. However, RSSI noise is not Gaussian [23] and other techniques obtain higher accuracies. Besides, this technique involves significant burden and can be hardly distributed, constraining scalability.

3.3.3 MinMax

MinMax [24] is a popular localization algorithm due to its simple implementation. Anchor nodes measure the RSSI of the packets emitted by the target and estimate their distance to the target. A pair of horizontal lines and a pair of vertical lines are drawn around each anchor node at the estimated distance to the target. The estimated localization of the target is the center of the overlapping area of all the squares, see Fig. 3.6.

Intuitively, the accuracy is better if the area of the intersection is smaller. Thus, a certain error is unavoidable even if ranging is perfectly noiseless.

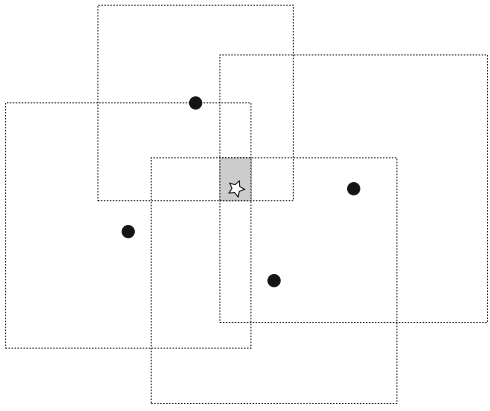


Fig. 3.6 Example of the operation of the MinMax algorithm

3.3.4 ROCRSSI

ROCRSSI [25] is a range-free localization method that relies on the assumption that RSSI decreases with the distance between transmitter and receiver. Consider a set of anchor nodes with known locations. Each anchor node receives packets from the target and from the other anchor nodes. Then, the measurements coming from the anchor nodes are divided into two sets: the first one, with RSSI values lower than the RSSI received from the target; and the second, with RSSI values greater than the RSSI of the target.

The maximum RSSI value in the first set and the minimum RSSI value in the second one are assumed to be close to the target, defining an inner ring of radius $R1$ and an outer ring of radius $R2$. Other ring pairs are created when other anchor nodes are considered. The estimated location of the target is the centroid of these overlapping rings.

An example is shown in Fig. 3.7. Anchor node A reads RSSI values from B, C and from the target T, as $RSSI_{AB}$, $RSSI_{AC}$ and $RSSI_{AT}$. They are ordered as $RSSI_{AB} < RSSI_{AT} < RSSI_{AC}$. B is in a first set of anchor nodes and C is in the second set. The maximum value in the first set is $RSSI_{AB}$ and the distance between A and B is $R1$. Similarly, the minimum value in the second set is $RSSI_{AC}$ and the distance between A and C is $R2$. The target is estimated to lie between the two circles defined by $R1$ and $R2$. Repeating this process for anchor nodes B and C, the estimated target location lies in the centroid of the intersection of the rings.

This algorithm is range free. It does not need to estimate the distances among nodes, but just to compare between different RSSI measurements.

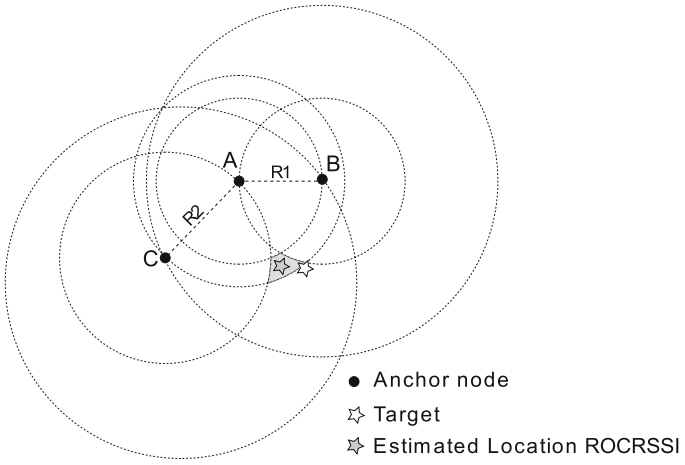


Fig. 3.7 Example of the operation of the ROCRSSI algorithm

3.3.5 Weighted Centroid Localization

Weighted Centroid Localization (WCL) [26] exhibits high robustness against noise in RSSI measurements. Although other techniques such as Least Squares (LS) are optimal in case of measurements with Gaussian noise, the performance of WCL is better with realistic RSSI measurements.

The target measures the RSSI value of the packets it receives from anchor nodes j located at known positions L_j. In this method the location of the target node i is computed using the expression:

$$L_i = \frac{\sum_{j=1}^{n}(\omega_{ij} L_j)}{\sum_{j=1}^{n} \omega_{ij}}, \tag{3.10}$$

where n is the size of the set of RSSI measurements received by the target and ω_{ij} are weighting factors that depend on the distance between the target i and each anchor j as follows:

$$\omega_{ij} = \frac{1}{(D_{ij})^p}, \tag{3.11}$$

where p is an exponent that modifies the influence of distance in the weights. Higher p gives more relevance to measurements from nearby static anchor nodes. RSSI-range models become flat—i.e. insensitive—as range increases. The measurements from distant anchor nodes provide less useful information and are more affected by noise.

In [27] experimental tests to determine the best values of p were performed. The number of anchor nodes, m, used in the localization technique was also analyzed. Figure 3.8 shows the mean localization errors obtained. A minimum of three anchor nodes were necessary to obtain reasonable localization errors. The results also revealed that taking into account distant anchor nodes frequently deteriorates accuracy. Measurements from distant anchor nodes are often less informative (flatter RSSI-range curve) and have higher noise level.

3.3.6 Maximum Likelihood

Target localization and tracking are probabilistic problems. The position and velocities of the target can not be measured directly. The objective of the Maximum Likelihood method (ML) [28] is to infer localization adopting a probabilistic approach. Given the vector of RSSI measurements $r = [r_1\, r_2 \ldots r_n]$ that the target received from n anchor nodes with coordinates $X = [x_1\, x_2 \ldots x_n]$ and $Y = [y_1\, y_2 \ldots y_n]$, the ML algorithm computes the a priori probability of receiving r_i for each potential position $[x_i\, y_i]$ of the target. ML estimates the target location as the position that maximizes that probability.

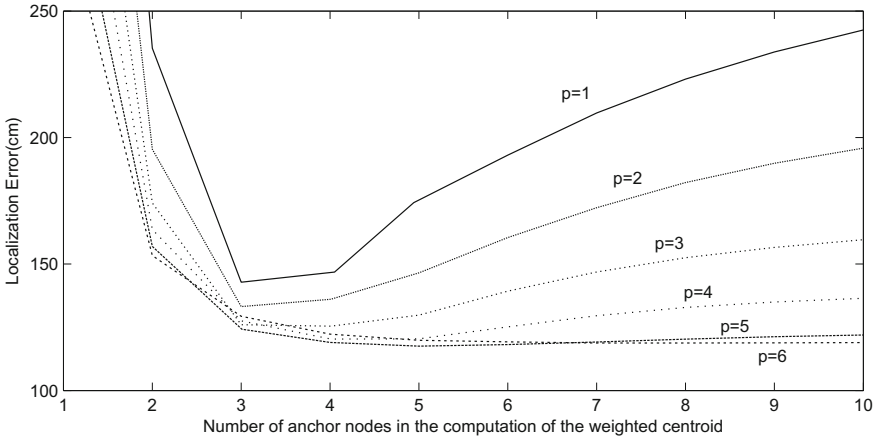


Fig. 3.8 Evaluation of the effect of p and the number of anchor nodes in WCL

Implementing ML is more complex than other techniques such as WCL or MinMax, but it minimizes the variance of the estimation error as the number of measurements grows. However, in most realistic cases the numbers of anchor nodes are limited and its performance can be unsatisfactory.

3.3.7 Recursive Bayesian Filtering

These techniques require defining a state vector with the target characteristics to be estimated, typically the target current location and velocity. The objective of Recursive Bayesian Filtering (RBF) techniques is to obtain beliefs of the state that reflect the knowledge about the state of the target. Probabilistic techniques represent beliefs through conditional probability distributions. A belief distribution assigns a probability (or density value) to each possible hypothesis of the state. Belief distributions are posterior probabilities over state variables conditioned on the available data. We will denote belief over a state variable x_t by $bel(x_t)$, which is an abbreviation for:

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}) \tag{3.12}$$

Thus, $bel(x_t)$ is the probability distribution over the state x_t at time t, conditioned on all past measurements $z_{1:t}$ and all the past control actions $u_{1:t}$. It is also useful to calculate a posterior incorporating z_t:

$$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t}) \tag{3.13}$$

This probability distribution is often referred to as prediction in the context of probabilistic filtering. $\overline{bel}(x_t)$ predicts the state at time t based on the previous state before integrating z_t, the measurement at time t.

The most general algorithm for calculating beliefs is given by the *Recursive Bayesian Filter* (RBF), see Algorithm 1. The RBF is recursive: $bel(x_t)$ at time t is calculated from $bel(x_{t-1})$ at time $t-1$. Its inputs are $bel(x_{t-1})$ and the most recent measurement z_t. Its output is $bel(x_t)$ at time t. Algorithm 1 depicts the step at time t of the RBF algorithm.

Algorithm 1 The general RBF algorithm.

1: **Recursive Bayesian Filter**$(bel(x_{t-1}), u_t, z_t)$
2: **for all** x_t **do**
3: $\quad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) \delta x$
4: $\quad bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t)$
5: **end for**
6: **return** $bel(x_t)$

The RBF algorithm has two main stages. In Line 3, $\overline{bel}(x_t)$ is predicted based on the prior belief over state x_{t-1} and the control action u_t. This step is called the prediction stage. In localization and tracking the target is not controlled and the update stage is based only on the previous state x_{t-1}. The second stage is called the measurement update. In Line 4, the RBF algorithm multiplies $\overline{bel}(x_t)$ by the probability that the measurement z_t may have been observed. The result is normalized by η, the normalization constant. The updated belief $bel(x_t)$ is returned in Line 6 of the algorithm.

There are two basic families of tractable approximations of the RBF: Gaussian techniques, including the Kalman Filters and its derivatives, which assume a Gaussian probabilistic distribution of the belief; and non-parametric filters, including Particle Filters, which approximate the belief by a finite number of samples.

3.3.7.1 Kalman Filter

Since its development in the early sixties, Kalman Filters (KFs) have been extensively researched and applied in different problems. Their success is originated by their simplicity and robustness. KFs have become one of the most common techniques used for localization and tracking in ubiquitous computing systems.

KFs are parametric RBFs that implement an optimal estimator that minimizes the covariance of the estimated error [29]. They represent the Gaussian distribution of $bel(x_t)$ at time t by its mean μ_t and covariance Σ_t. These distributions are Gaussian if the three following properties, in addition to the Markov hypothesis, are satisfied:

1. The probability of the next state $p(x_t|u_t, x_{t-1})$ is a linear function with additive Gaussian noise. This is represented as:

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t, \tag{3.14}$$

   where A_t and B_t are matrices that represent the linear model of the system evolution. ϵ is a Gaussian noise with zero mean and covariance Q_t.
2. The probability of the measurement $p(z_t|x_t)$ is linear with additive Gaussian noise:

$$z_t = H_t x_t + \delta_t, \tag{3.15}$$

   where H_t is the matrix that represents the observation model of the sensor. δ is a Gaussian noise with zero mean and covariance R_t.
3. The initial distribution should be a normal distribution.

In a system modeled by Eqs. (3.14) and (3.15) the operation of KFs is based on two stages: the prediction stage and the update stage. In the prediction stage an estimate of the system state for the next instant is obtained. The update stage integrates the new measurements in order to improve the estimation of the state vector. The Kalman Filter estimates not only the state but also the covariance matrix of the estimation error, i.e. it provides an estimate of how good the estimation is.

However, the assumptions of linear prediction and measurement models and additive Gaussian noise are not met in many cases. For example, RSSI-range model is non-linear. The Extended Kalman Filter (EKF) overcomes the assumption of linearity and can be applied in cases where the prediction or the measurement models are governed by nonlinear functions f and h:

$$x_t = f(u_t, x_{t-1}) + \epsilon_t, \tag{3.16}$$

$$z_t = h(x_t) + \delta_t \tag{3.17}$$

Function f replaces matrices A_t and B_t in Eq. (3.14) and h replaces matrix H_t in Eq. (3.15). However, as f and h are non linear, the estimation is not Gaussian. The EKF computes an approximation to the estimation assuming that it is Gaussian. The EKF can behave similarly to the Kalman Filter except that the estimation distribution is not exact but approximate.

The key idea in the EKF is linearization. Given the non-linear functions f and h, the linearization of the functions is obtained by Taylor expansion, and the Jacobian matrices of f and h take the role of A and C:

$$A = \frac{\partial f}{\partial x}, \qquad C = \frac{\partial h}{\partial x} \tag{3.18}$$

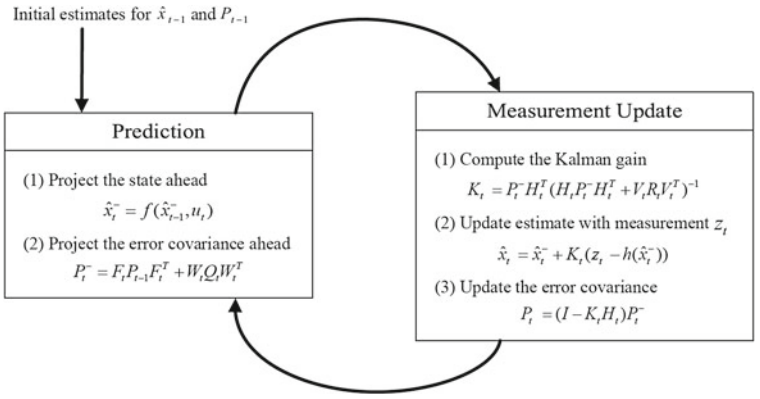


Fig. 3.9 Operation of the EKF algorithm

Figure 3.9 shows the operation of the EKF algorithm. Q is the covariance of the Gaussian noise of the prediction model and R is the covariance of the measurement noise. P is the covariance of the error of the state. K is the so-called Kalman gain.

KFs are suitable only for single-hypothesis estimation and Gaussian noise. For non Gaussian noise or multi-hypothesis problems it is recommended to use non-parametric RBFs, such as Particle Filters.

3.3.7.2 Non Parametric Bayesian Filters

Particle Filters (PFs) are non-parametric implementations of the RBFs: they represent $bel(x_t)$ by a set of random state samples. Instead of representing the distribution by a parametric form, PFs represent the distribution by a set of samples drawn from this distribution. That representation is approximate, but can be suitable for virtually any probability distribution.

The samples of a posterior distribution are called *particles*: $X_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$. Each particle $x_t^{[m]}$ is an instantiation of the state at time t, i.e. a hypothesis of the state at time t. M is the number of particles in the particle set X_t. Ideally, the likelihood for a state hypothesis x_t to be included in the particle set X_t will be proportional to $bel(x_t)$. Thus, the denser a region of the state space is populated by particles, the more likely it is that the true state falls in this region. This property holds in case of having an infinite number of particles. With a finite number of particles, the particles are drawn from a slightly different distribution. In practice, this difference is negligible if the number of particles is not too small.

The PF algorithm also maintains $bel(x_t)$ recursively from $bel(x_{t-1})$, i.e. PFs construct the particle set X_t recursively from the set X_{t-1}. PFs are computationally demanding and their execution can be hardly distributed.

3.3.8 RSSI Map-Based Algorithms

RSSI map-based algorithms compute the location of the target by comparing the RSSI measurements they gather with a previously obtained RSSI map. The RSSI map is built off-line with the RSSI measurements received at certain locations. The characteristics of the signal propagation in the environment are captured in the RSSI map. These techniques avoid creating complex signal propagation models. However, the building of RSSI maps is rather laborious and is not robust to changes in the environment. There are two main approaches: active and passive.

3.3.8.1 Active

Active RSSI map-based localization and tracking is often called fingerprinting [30]. In fingerprinting a target receives signals from anchor nodes or vice versa, the RSSI of the signals is measured and compared to a RSSI map in order to estimate its location.

The building of the RSSI map begins by dividing the area of interest into cells. The ith element in the RSSI map is:

$$M_i = \left(B_i, \left\{\mathbf{a}_{ij} \mid j \in N_i\right\}\right), i = 1, \ldots, M, \tag{3.19}$$

where B_i is the ith cell, which center is p_i. Vector $\mathbf{a}_{ij}$ holds the RSSI values measured from anchor node j and N_i is the set of anchor nodes which signals can be received in cell i.

The RSSI map can be modified or preprocessed before using it in the location estimation phase. The objective can be the reduction of the memory requirements of the RSSI map or the reduction of the computational burden.

Given the RSSI map, the objective of the location estimation phase is to infer the location of the target from the RSSI received from the target received by several anchors. In some cases several RSSI samples from the same anchor are collected and the mean value is used to reduce sensitivity to noise.

In [31] the estimation of the state is performed in a deterministic way, through the weighted K-nearest neighbor technique. If state x is assumed to be a random variable, it can also be estimated using a probabilistic technique such as KFs or PFs.

3.3.8.2 Passive

Passive RSSI map-based localization and tracking [32] relies on the fact that radio signals are affected by changes in the environment. By continuously recording and analyzing the received RSSI this technique can detect the changes in the environment and correlate them in order to infer the presence of people and their locations. It has been used to detect the presence of people in a room and to locate them. The radio frequency used by the nodes is 2.4 GHz. The human body contains more than 70%

Table 3.2 Comparison of the main RSSI-based localization and tracking techniques

	Range based/free	Active/passive	Burden	Accuracy
Multilateration	Based	Active	Low	Low
MinMax	Based	Active	Low	Low
ROCRSSI	Free	Active	Medium	Medium
WCL	Free	Active	Low	Low
Maximum Likelihood	Based	Active	Medium	Medium
Kalman Filter	Based	Active	High	Medium
Particle Filter	Based	Active	Very high	High
Fingerprinting	Free	Active	High	High
Passive RSSI	Free	Passive	High	Medium

of water and it is known that the resonance frequency of water is 2.4 GHz. Thus, the human body acts as an absorber attenuating the radio signals.

The basic setting of these methods consists of several pairs of nodes, acting as transmitters and receivers, deployed in the environment. To build the radio map a person stands at different points and the signal strength characteristics are recorded. For every point the signal strength histogram of each pair of nodes is obtained. Based on the built radio map, a Bayesian inversion-based inference algorithm is used to compare the radio map with the RSSI measurement vector.

3.3.9 Discussion

In this section some of the main RSSI-based localization and tracking techniques have been described. Their main characteristics are shown in Table 3.2.

The Kalman Filter is one of the most complete techniques because it considers separately measurement noise and model uncertainties. In fact, it is one of the most widely used and researched approach. However, Kalman Filters (particularly the update stage) can not be distributed and all the computational burden has to be performed by one node.

Even with an efficient implementation the precision of the most accurate RSSI-based localization technique is not high due to path loss, reflections and other interactions of the radio signals with the environment.

3.4 Localization and Tracking Using Camera Measurements

Camera networks have been widely used for localization and tracking since decades. In the last years the integration of low energy CMOS vision chips with programmable image processing capabilities in ubiquitous computing systems

originated the so-called Wireless Camera Networks (WCNs). WCNs have the sensing power of traditional camera networks with the flexibility, reconfigurability and ease of deployment of Wireless Sensor Networks.

Much effort has been devoted to tracking with camera networks that use only camera measurements [33–38] and combining camera measurements with other measurements typical in ubiquitous sensor networks such as RSSI [39–42] or TDoA measurements [43]. Below the main camera measurement integration techniques are summarized.

3.4.1 Integrating Only Camera Measurements

Different schemes for WCNs have been developed depending on how the camera measurements are integrated. The most widely-used ones are based on Maximum Likelihood, Kalman Filters and Particle Filters.

3.4.1.1 Maximum Likelihood

Maximum Likelihood (ML) techniques estimate the state S maximizing a statistical likelihood function, by computing the state that best matches with the observations:

$$\hat{S} = \arg\max_{S} p(m|S), \tag{3.20}$$

where m represents the location of the target measured by the cameras in a common reference frame.

Let $p_i = [x_{n,i}\, y_{n,i}]$ be the location of a point viewed from camera node i. Assuming a pin-hole model, from p_i it is easy to compute m_i, the location of the target on the scenario measured by camera i expressed in the reference frame local of camera i. Using the transformation matrix of camera i, m_i can be transformed to m_i^c, which represents the location of the target measured by camera i in a common reference frame.

Assume that m_i^c contains errors that can be modeled as Gaussians with zero mean noise and covariance matrix Cov_i. Assume that the measurements from different cameras can be considered statistically independent. In this case the ML technique estimates the state S fusing the measures m_i^c using the expression [44]:

$$S = \sum_{i=1}^{N} Cov_i^{-1} \sum_{i=1}^{N} (m_i^c Cov_i^{-1}) \tag{3.21}$$

Cov_i can be decomposed in an eigenvector matrix and an eigenvalue matrix, $Cov_i = P\Lambda P^{-1}$. Matrix Λ can be constructed taking into account that the eigenvalues, at the diagonal of Λ, are the values of variance of each axis of the

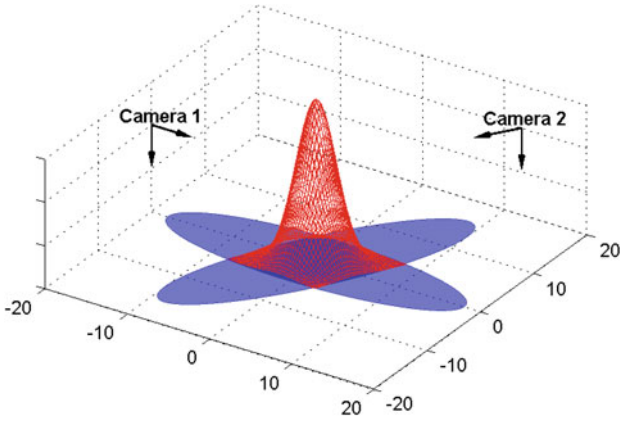


Fig. 3.10 Simple example of the ML technique

camera local reference frame. The eigenvectors form the columns of the eigenvalue matrix P. The eigenvectors are orthonormal vectors that represent the axes of the camera local reference. P and Λ, and thus Cov_i, can be easily built with the orientation of camera i and the measurement noise level.

Figure 3.10 shows an illustration of ML integrating measurements from two cameras. The probability density distributions of the target location computed individually by *Camera1* and *Camera2* are shown in blue color. The resulting probability of the fused estimation, in red, shows the uncertainty improvement.

In ML lack of camera measurements hampers the intersection in Eq. (3.21), potentially involving high localization errors. These errors can be mitigated by RBFs such as Kalman Filters.

3.4.1.2 Kalman Filters

Kalman Filters are one of the most commonly-adopted approaches for the integration of camera node measurements.

In [33] the authors used a Extended Kalman Filter (EKF) that integrates in a centralized way all the measurements it receives from the camera nodes. In [45] a centralized EIF for WCN was reported.

A Kalman-Consensus filter for target localization in peer-to-peer networks is presented in [37]. Its objective is to find a consensus on the state of multiple targets in a dynamic camera network with possibly overlapping fields of view.

The above schemes are centralized and the whole computational burden is assumed by the central node (root or CH) preventing scalability and robustness.

3.4.1.3 Particle Filters

Sequential Monte Carlo techniques, such as the Particle Filters presented in Sect. 3.3.7, are non parametric filters and can deal with non Gaussian noise and multi-hypothesis estimation problems. This flexibility is at the cost of significant computational complexity.

Most tracking algorithms using Particle Filters in sensor networks adopt a centralized approach, which results in nodes with inhomogeneous resource consumption. Distributed algorithms, such as the Distributed Particle Filter (DPF) algorithm described in [46], address the aforementioned problems by decentralizing the computation and communication so that a single fusion center is not required. DPF maintains a local Particle Filter at selected nodes throughout the network. Each local filter is used both to perform estimation and to implement extensive compression of local measurements for transmission to other nodes. DPF mitigates some of the inherent problems of centralization but requires higher computational and communication complexity than centralized PFs.

In [35] the authors describe a sensor network scheme for tracking a target in the presence of static and moving occluders using a network of cameras. The locations of the static occluders are assumed known, but only prior statistics on the positions of the moving occluders are available. An auxiliary Particle Filter that incorporates the information from the occluder is used to track the target.

3.4.2 Integrating Cameras and Other Sensors

Some works combine cameras with other sensors widely used in WSN. RSSI or TDoA provide range measurements. They naturally complement bearing camera measurements and have been widely used in WSN localization. Below, some works integrating both types of measurements are summarized.

In [43] the authors present a system for indoor surveillance that uses WSN nodes deployed on the ceiling and a pan and tilt camera. Each target carries a WSN node. The anchor nodes have an in-built ultrasound transceiver set. The anchors communicate with the listeners using radio and ultrasound signals. The distance between each anchor and the target is estimated using TDoA. The localization is performed combining three different steps: Least Squares minimization, Kalman filtering and outlier rejection. The coordinates of the obtained location are transformed into the camera coordinate system in order to orientate the pan and tilt unit and track the target.

In [41] an architecture for cooperation of networked robots in urban areas is described. The architecture allows the tracking of persons using WSN nodes and static cameras. Each target is tagged with a mobile node. A network of static nodes measure the RSSI of the packets they receive from the mobile nodes. The camera network also tracks the targets. The system uses a decentralized sensor fusion architecture in which each node employs local information (RSSI measurements and camera

measurements) to obtain an estimate of the location of the person. Thus, these nodes share their local estimates to obtain a consistent global estimation.

In [39] a framework based on PFs for tracking employing visual and radio measurements is presented. This approach aims at exploiting the synergies resulting from the use of both types of measurements. In fact, visual tracking commonly outperforms radio localization in terms of accuracy but inefficacy arises in case of occlusions or when the scenario is too large. On the other hand, radio measurements are unambiguously associated to each target using their identification number. Besides, the sensing range is larger and the measurements are available in larger areas. This system enables target localization even where/when either video or radio observations are missed or not satisfactory. The PF operates using visual measurements or RSSI depending on their availability. When the target is not visible, an observation model for RSSI of the target is used to update the prediction of the target position.

In [40] a pedestrian tracking system is presented. The system uses RSSI location estimation and conventional visual surveillance. The target tracking algorithm consists of two parts: a video-based Particle Filtering and a RSSI-based location estimation. PFs using video can accurately detect and track the targets with the same or higher accuracy than GPS, but cannot cover the areas outside the scene. On the other hand, RSSI location estimation has lower coverage constraints, but the location estimation is not always stable and the accuracy is affected by the environment. This system combines these characteristics by switching between the two algorithms. In the area covered by cameras, the system gives priority to tracking using cameras in order to improve accuracy. In the area not covered by the cameras, RSSI tracking is given higher priority. Switching occurs at the boundary between both areas.

3.4.3 Discussion

In this section some of the main WCN localization and tracking schemes have been described. The Table 3.3 summarizes their main characteristics.

Table 3.3 Comparison of the main camera-based tracking schemes in ubiquitous computing systems

	Centralization	Burden	Accuracy
MLE [44]	Centralized	Medium	Low
EKF [33]	Centralized	High	Medium
EIF [45]	Centralized	High	Medium
Consensus KF [47]	Distributed	Medium	Medium
PF [35]	Centralized	Very high	High
DPF [46]	Distributed	High	High

Table 3.4 Comparison of the different schemes for localization and tracking that consider camera and RSSI measurements.

Work	Sensors	Technique	Sensor integration
[43]	Cameras/TDoA	KF	TDoA calibrates cameras
[41]	Cameras/RSSI	EIF	Fuses all data
[39]	Cameras/RSSI	PF	Switching scheme
[40]	Cameras/RSSI	PF	Switching scheme

The schemes based on Kalman Filters seem to be the most complete. KF-based schemes require low computational burden and are more accurate than those based on ML. They also consider separately measurement noise and model uncertainties. However, KFs are often applied in a centralized way, where all the computation burden is performed by a single node. Furthermore, with the exception of [45], none of these schemes employ techniques to improve energy efficiency.

Table 3.4 shows the characteristics of the described schemes that consider cameras and other sensors for localization and tracking in ubiquitous networks.

Work [43] does not integrate both types of measurements. It uses RSSI measurements to help the cameras. Other two, [39, 40], adopt switching schemes that use one type of measurement or the other depending on their availability but do not take advantage of the benefits of the joint use of both types of measurements. Only work [41] integrates both types of measurements.

3.5 Decentralized Bayesian Multi-sensor Measurement Integration

Recursive Bayesian Filters (RBFs) provide a well-founded statistical framework that naturally assumes that measurements and models are subject to uncertainty. They can be used to integrate indistinguishably RSSI and/or camera measurements. They only need a suitable observation model and the sensor uncertainty for every type of measurement, see Fig. 3.11.

RBFs include Kalman Filters (KFs) and also Information Filters (IFs), which present a number of advantages over KFs in localization and tracking in ubiquitous computing systems.

3.5.1 Extended Information Filter

Information Filters (IFs) implement a parametric RBF that employs the so-called canonical representation, which is composed of the information vector $\xi = \Sigma^{-1}\mu$ and the information matrix $\Omega = \Sigma^{-1}$, where μ and Σ are respectively the mean and variance of the state. IFs are duals to Kalman Filters (KFs). The prediction stage of

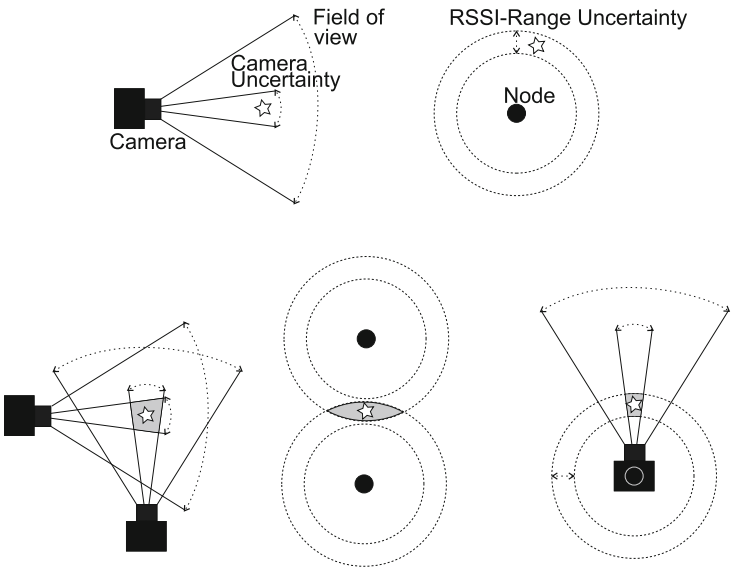


Fig. 3.11 *Top* Uncertainty in camera (*left*) and RSSI (*right*) measurements. *Bottom* Overlapping of uncertainty regions between the measurements from two different cameras (*left*), between two RSSI measurements (*center*) and between RSSI and camera measurements from the same node (*right*)

KFs is more efficient than that of IFs but the update stage of IFs are more efficient than that of KFs [29]. Thus, IFs are significantly more efficient than KFs in case of using a simple prediction model and high number of measurements. That is the case in ubiquitous computing systems, where a high number of nodes can be used in target tracking. Besides, IFs are numerically more stable and are more suitable for representing lack of information, $\Omega = 0$.

The typical state vector in target tracking considers the current target location and its local velocities. $\mathbf{x}_t$ is the system state vector at time t:

$$\mathbf{x}_t = [x_t, y_t, z_t, vx_t, vy_t, vz_t]^T, \tag{3.22}$$

where (x_t, y_t, z_t) is the target 3D location at time t and (vx_t, vy_t, vz_t) is the target 3D velocity.

$\mathbf{z}_{i,t}$ is the measurement vector containing the measurements taken by anchor sensor i at time t. If that sensor is a camera, then $\mathbf{z}^c_{i,t}$ is the measurement vector containing the camera measurements:

$$\mathbf{z}^c_{i,t} = [x^c_{i,t}, y^c_{i,t}, x^c_{i,t-1}, y^c_{i,t-1}]^T, \tag{3.23}$$

where $x^c_{i,t}, y^c_{i,t}$ are the coordinates of the target in the image plane of camera i at time t.

On the other hand, $\mathbf{z}^r_{i,t}$ is the measurement vector containing the RSSI measurements taken by anchor node i at time t:

$$\mathbf{z}^r_{i,t} = [rssi_{i,t}, rssi_{i,t-1}]^T, \tag{3.24}$$

where $rssi_{i,t}$ is the RSSI measured by anchor node i of the packets it receives from the target (assumed tagged with a node) at time t. As can be noticed in Eqs. (3.23) and (3.24), $\mathbf{z}^c_{i,t}$ and $\mathbf{z}^r_{i,t}$ contain also the past measurements at time t-1 in order to improve the observation of the target velocity.

Like all RBFs, IFs require a prediction model and a observation model. Both are assumed to be under White Gaussian Noise parameterized by their means and covariances R_t and Q_t. For the prediction model we employed a simple linear motion model to represent local target motions. More complex models require a priori knowledge of the target motion, which are often unavailable in localization ans tracking applications:

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + \epsilon_t, \qquad A = \begin{bmatrix} \mathbf{I} & T \cdot \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{3.25}$$

where ϵ_t is a Gaussian noise with zero mean and covariance R_t, I is the 3×3 identity matrix and T is the time between two consecutive tracking intervals.

One observation model is required for each type of measurement. The camera observation model is derived from the pin-hole model, see Fig. 3.12. Assume that P_t is the location of the target in the global reference frame G at time t. Assume that $p_{i,t}$ is the projection of the target on the image plane of camera node i expressed in the local reference frame of camera, F_i, related to G by transformation matrix T_i.

The following observation model for camera i holds:

$$p_{i,t} = h^c_{i,t}(P_t) = \begin{bmatrix} t_{i,1}\begin{bmatrix} P_t & 1 \end{bmatrix}^T / t_{i,3}\begin{bmatrix} P_t & 1 \end{bmatrix}^T \\ t_{i,2}\begin{bmatrix} P_t & 1 \end{bmatrix}^T / t_{i,3}\begin{bmatrix} P_t & 1 \end{bmatrix}^T \end{bmatrix} + \delta^c_t, \tag{3.26}$$

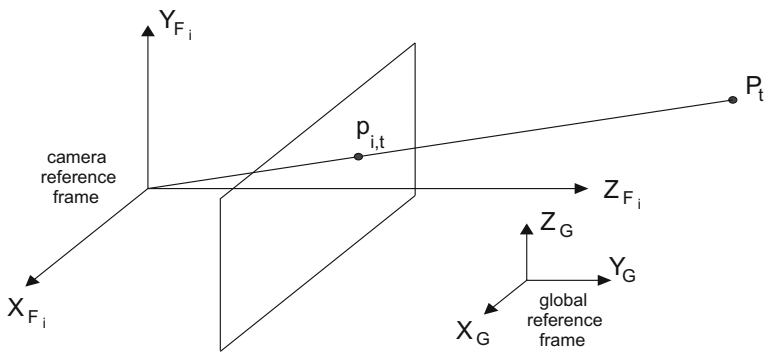


Fig. 3.12 Reference frames used for camera measurements integration

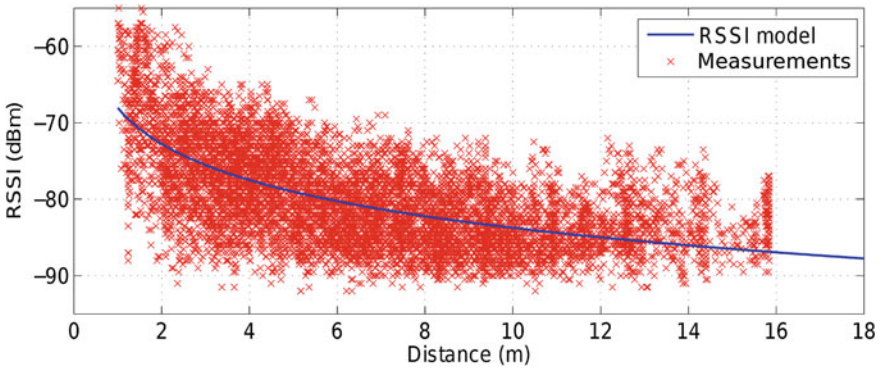


Fig. 3.13 RSSI-range model obtained using measurements between each pair of anchor WSN nodes in the UBILOC testbed

where $t_{i,j}$ is the j-th row of T_i. δ_t^c is a Gaussian noise with zero mean and covariance Q_t^c. This observation model is nonlinear. The need for its linearization leads to the use of an Extended Information Filter (EIF), which uses its Jacobian $H_{i,t}^c$ computed as follows:

$$H_{i,t}^c = \frac{\partial h_{i,t}^c}{\partial \mathbf{x}_t} \tag{3.27}$$

To integrate RSSI measurements the widely accepted RSSI-range model [4] is adopted:

$$rssi_{i,t} = h_{i,t}^r(P_t) = a \log d_{i,t} + b + \delta_t^r, \tag{3.28}$$

where $d_{i,t}$ is the distance between the target and node i at time t and a and b are model parameters. δ_t^r is a Gaussian noise with zero mean and covariance Q_t^r.

The default RSSI-range model is assumed known, taken from works such as [4] or measured experimentally. Figure 3.13 shows the RSSI-range model obtained using RSSI measurements between each pair of anchor nodes in the UBILOC testbed.

RSSI-range models are also nonlinear and the EIF uses its Jacobian $H_{i,t}^r$ computed as follows:

$$H_{i,t}^r = \frac{\partial h_{i,t}^r}{\partial d_{i,t}} \frac{\partial d_{i,t}}{\partial \mathbf{x}_t} \tag{3.29}$$

3.5.2 Distributed Implementation

Another advantage of IFs over KFs is their natural fit for integrating measurements collected in a decentralized way. Measurement integration in RBFs is performed through the Bayes rule. When represented in logarithmic form, the Bayes rule becomes an addition. This is the case of IFs, which canonical parameters

represent probability in logarithmic form. Thus, the measurement update stage in IFs is achieved by summing up the contributions from different sensors. As a result, IFs can integrate measurements in arbitrary order and in a fully distributed manner.

This method exploits this property and adopts a distributed implementation. The cluster head broadcasts the predicted state in the cluster. When a cluster member receives the packet it takes a measurement, integrates it by computing its contribution to the EIF update and transmits the contribution to the cluster head, which only has to add all the contributions it received in order to recover the updated state.

Algorithms 2 and 3 summarize the distributed EIF at time t assuming that node i is the cluster head. The EIF prediction for time $t+1$ is the last step performed at time t. It performs first *EIFUpdate*, and second *EIFPrediction*.

Algorithm 2 Operations of cluster head i in the distributed EIF.

Require: $\overline{\Omega}_t$, $\overline{\xi}_t$ and $\overline{\mu}_t$
1: Create *UpdateReq* with $\bar{\mu}_t$
2: Broadcast *UpdateReq* within the cluster
3: Receive *UpdateResp*
4: Extract $\bar{\Omega}_{j,t}$ from packets
5: Compute Ω_t, ξ_t and μ_t as in (3.32)-(3.34)
6: Compute $\overline{\Omega}_{t+1}$, $\overline{\xi}_{t+1}$ and $\overline{\mu}_{t+1}$ as in (3.35)-(3.37)

Algorithm 3 Operations of cluster member j in the distributed EIF.

1: Receive *UpdateReq* from the cluster head
2: Extract $\bar{\mu}_t$ from packet
3: Take measurement $\mathbf{z}_{j,t}$
4: Compute EIF update contribution as in (3.30) and (3.31)
5: Create *UpdateResp* with $\bar{\Omega}_{j,t}$ and $\bar{\xi}_{j,t}$
6: Transmit *UpdateResp*

The operation of the distributed EIF is as follows. The predicted state for time t—$\overline{\Omega}_t$, $\overline{\xi}_t$ and $\overline{\mu}_t$—is assumed available. It was computed in the last step of the iteration at time $t-1$. At time t the cluster head first broadcasts an *UpdateReq* packet containing $\overline{\mu}_t$. Each cluster member j receiving the packet takes a measurement $\mathbf{z}_{j,t}$ from its sensor and integrates it computing its local information matrix $\Omega_{j,t}$ and information vector $\xi_{j,t}$—its contribution to the EIF update stage:

$$\Omega_{j,t} = H_{j,t}^T Q_{j,t}^{-1} H_{j,t}, \tag{3.30}$$

$$\xi_{j,t} = H_{j,t}^T Q_{j,t}^{-1} [\mathbf{z}_{j,t} - h_j(\overline{\mu}_t) + H_{j,t}\overline{\mu}_t] \tag{3.31}$$

The cluster head—node i—can also sense the target with its sensors. It also takes a measurement and computes $\Omega_{i,t}$ and $\xi_{i,t}$. Next, each node in the cluster mode transmits an *UpdateResp* packet to the cluster head with the resulting $\Omega_{j,t}$ and $\xi_{j,t}$.

The cluster head receives *UpdateResp* packets. The update stage of EIF is additive. When a timeout expires the cluster head reconstructs the updated state (ξ_t and Ω_t) by adding the predicted $\overline{\xi}_t$ and $\overline{\Omega}_t$ to the contributions it received from all cluster members—and also $\xi_{i,t}$ and $\Omega_{i,t}$ computed by the cluster head:

$$\Omega_t = \overline{\Omega}_t + \Omega_{i,t} + \sum_j \Omega_{j,t}, \tag{3.32}$$

$$\xi_t = \overline{\xi}_t + \xi_{i,t} + \sum_j \xi_{j,t} \tag{3.33}$$

With them the cluster head computes μ_t:

$$\mu_t = \Omega_t^{-1} \xi_t \tag{3.34}$$

Finally, the cluster head computes the predicted state for time t+1 as follows:

$$\overline{\Omega}_{t+1} = (A\Omega_t^{-1}A^T + R_{t+1})^{-1}, \tag{3.35}$$

$$\overline{\mu}_{t+1} = A\mu_t, \tag{3.36}$$

$$\overline{\xi}_{t+1} = \overline{\Omega}_{t+1}\overline{\mu}_{t+1} \tag{3.37}$$

This method distributes the EIF computational burden among all the cluster members. Apart from gathering and integrating its own measurements, the cluster head only has to add the contributions from all nodes—(3.32) and (3.33)—and to compute the EIF prediction stage, which requires low burden due to the simple prediction model assumed. As a result, the EIF can be executed almost in constant time regardless of the number of cluster nodes, overtaking traditional schemes based on KFs both in absolute computational burden and in scalability. Besides, each node only requires information of itself.

3.5.3 Evaluation and Comparison

The general architecture devised in Chap. 2 requires a tool to integrate multi-sensor measurements. The main existing techniques for integration of measurements of different type in cluster-based schemes are EIFs, EKFs and Particle Filters. The comparison between the two latter was performed in Sects. 3.3 and 3.4. This section compares the performance of EKF and of the EIF presented in Sect. 3.5.1.

The distributed EIF benefits from the additive nature of measurement integration in EIFs. The EIF prediction stage, which is independent of the cluster size, is computed by the cluster head. In the EIF update stage each node computes the integration of its own measurements: burden is shared among all active nodes. The cluster head is the node with the highest burden: it computes the EIF prediction and its contribution to the EIF update. Both are independent of the cluster size and are executed in constant time.

EKFs cannot be easily implemented in a distributed manner. In centralized schemes the cluster head receives all the measurements of the cluster nodes and computes both the EKF prediction and EKF update stages. As pointed out above, centralized EIFs are more efficient than EKFs in case of using simple prediction models and high number of measurements. Assuming that the sizes of the state vector and of the measurement vector are known, it is simple to estimate the number of operations (float multiplications) needed to complete a full cycle of the filter. Equations (3.38)–(3.40) show the burden expressed in number of operations (NO) of the cluster head to implement EKF, centralized EIF and the distributed EIF in clusters with n nodes:

$$NO_{EKF} = 468 + 808n \tag{3.38}$$

$$NO_{CEIF} = 892 + 392n \tag{3.39}$$

$$NO_{DEIF} = 892 + 392 \tag{3.40}$$

The prediction stage requires 468 operations in the EKF and 892 operations either in the centralized and decentralized EIF. The update stage requires 808 operations per measurement in the EKF while, in the centralized EIF it requires 392 operations per measurement. In the decentralized EIF it only requires 392 operations regardless of the number of measurements. The computational efficiency of the decentralized EIF and its scalability with the number of cluster nodes is illustrated in Fig. 3.14.

In each iteration in EKF and the decentralized EIF each anchor node gathers one measurement and transmits it to the cluster head: its computational burden is negligible. In the decentralized EIF every anchor node has to compute its contribution to the update EIF stage, which requires 392 operations in each cycle. This computational burden is constant regardless of the number of cluster nodes.

In the decentralized EIF, the cluster head broadcasts the predicted state $\overline{\mu}$ after performing the EIF prediction. The predicted state $\overline{\mu}$ is used by each node to integrate its measurement in the distributed EIF update stage. In centralized schemes cluster members do not need anything from the cluster head. Thus, in every iteration the decentralized EIF requires the transmission of only one packet more than the centralized EIF.

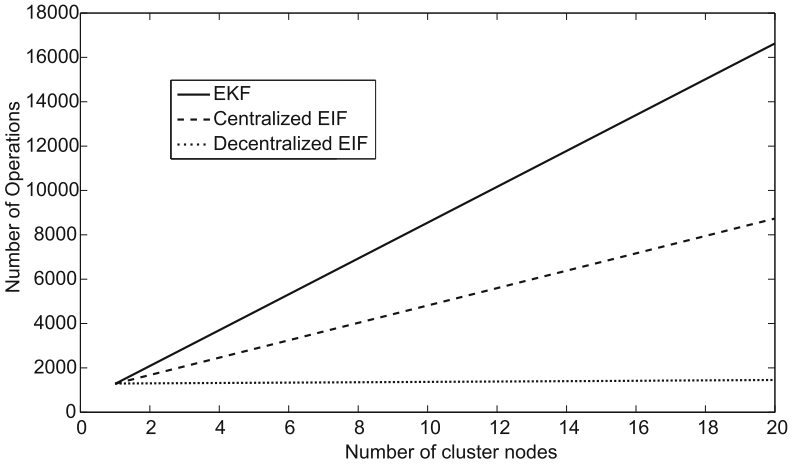


Fig. 3.14 Number of operations performed in the cluster head versus number of cluster nodes

3.6 Conclusions

This chapter summarized the main existing measurement integration techniques for cluster-based target localization and tracking in ubiquitous computing systems. The techniques have been classified according to the type of the physical measurement used.

The main techniques based only on RSSI and only on cameras have been described. Also, the main techniques that combine RSSI and camera measurements have been summarized. Most of them rely on using one or the other type of measurement depending on their availability, and in general take only partial advantage of the synergies between both types of measurements.

This chapter also described the employment of EIFs for flexibly integrating camera and RSSI measurements. EIFs provide efficient implementations of RBFs. Dual to KF, EIFs are computationally more efficient than KFs when dealing with cases that integrate a high number of measurements and use simple target motion models, as in localization and tracking problems. Besides EIFs can be easily distributed in schemes where each node computes its contribution to the EIF update stage. Distributed EIFs can be executed in almost constant time regardless of the cluster size.

This chapter described the algorithm of the distributed EIF and analyzed its computational burden comparing with KFs and centralized EIFs.

References

1. Baccour N, Koubâa A, Ben Jamaa M, Youssef H, Zuniga M, Alves M (2009) A comparative simulation study of link quality estimators in wireless sensor networks. In: Proceedings of the IEEE international symposium on modeling, analysis & simulation of computer and telecommunication systems (MASCOTS'09)
2. Baccour N, Koubaa A, Mottola L, Zuniga MA, Youssef H, Boano CA, Alves M (2012) Radio link quality estimation in wireless sensor networks: a survey. ACM Trans Sens Netw 8(4):34
3. Seidel SY, Rappaport TS (1992) 914 MHz path loss prediction models for indoor wireless communications in multifloored buildings. IEEE Trans Antennas Propag 40(2):207–217
4. Kumar P, Reddy L, Varma S (2009) Distance measurement and error estimation scheme for rssi based localization in wireless sensor networks. In: Proceedings of the conference on wireless communication and sensor networks (WCSN'09)
5. Guvenc I, Chong CC (2009) A survey on toa based wireless localization and nlos mitigation techniques. IEEE Commun Surv Tutorials 11(3):107–124
6. Savvides A, Han CC, Strivastava MB (2001) Dynamic fine-grained localization in ad-hoc networks of sensors. In: Proceedings of the international conference on mobile computing and networking, pp 166–179
7. Priyantha NB, Chakraborty A, Balakrishnan H (2000) The cricket location-support system. In: Proceedings of the ACM international conference on mobile computing and networking, pp 32–43
8. Priyantha NB, Miu AK, Balakrishnan H, Teller S (2001) The cricket compass for context-aware mobile applications. In: Proceedings of the ACM international conference on mobile computing and networking
9. Niculescu D, Nath B (2003) Ad hoc positioning system (APS) using AoA. In: Proceedings of the annual joint conference on IEEE computer and communications (INFOCOM'2003), vol 3, pp 1734–1743
10. He T, Huang C, Blum BM, Stankovic JA, Abdelzaher T (2003) Range-free localization schemes for large scale sensor networks. In: Proceedings of the international conference on mobile computing and networking, pp 81–95
11. Nagpal R, Shrobe H, Bachrach J (2003) Organizing a global coordinate system from local information on an ad hoc sensor network. In: Information processing in sensor networks, Springer, pp 333–348
12. Ray S, Ungrangsi R, Pellegrini D, Trachtenberg A, Starobinski D (2003) Robust location detection in emergency sensor networks. In: Proceedings of the annual joint conference of the IEEE computer and communications (INFOCOM'2003), vol 2, pp 1044–1053
13. Moussa M, Youssef M (2009) Smart devices for smart environments: device-free passive detection in real environments. In: Proceedings of the IEEE international conference on pervasive computing and communications (PERCOM'2009)
14. Hossain AM, Soh WS (2007) A comprehensive study of bluetooth signal parameters for localization. In: Proceedings of the IEEE international symposium on personal, indoor and mobile radio communications (PIMRC'2007)
15. Bouet M, Dos Santos AL (2008) Rfid tags: positioning principles and localization techniques. In: Proceedings of the wireless days 2008 (WD'08)
16. Grossmann R, Blumenthal J, Golatowski F, Timmermann D (2007) Localization in zigbee-based sensor networks. In: Proceedings of the European zigbee developer's conference (EuZDC'2007)
17. Gezici S, Tian Z, Giannakis GB, Kobayashi H, Molisch AF, Poor HV, Sahinoglu Z (2005) Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. IEEE Signal Process Mag 22(4):70–84
18. Olivera VM, Plaza JMC, Serrano OS (2006) Wifi localization methods for autonomous robots. Robotica 24(04):455–461

19. Bshara M, Orguner U, Gustafsson F, Van Biesen L (2010) Fingerprinting localization in wireless networks based on received-signal-strength measurements: a case study on wimax networks. IEEE Trans Veh Technol 59(1):283–294
20. Lee JS, Su YW, Shen CC (2007b) A comparative study of wireless protocols: bluetooth, uwb, zigbee, and wi-fi. In: Proceedings of the IEEE annual conference of the IEEE industrial electronics society (IECON'2007), pp 46–51
21. Wang X, Bischoff O, Laur R, Paul S (2009) Localization in wireless ad-hoc sensor networks using multilateration with rssi for logistic applications. Procedia Chem 1(1):461–464
22. Wessels A, Wang X, Laur R, Lang W (2010) Dynamic indoor localization using multilateration with RSSI in wireless sensor networks for transport logistics. Procedia Eng 5:220–223
23. Lee H, Cerpa A, Levis P (2007a) Improving wireless simulation through noise modeling. In: Proceedings of the IEEE international symposium on information processing in sensor networks (IPSN'2007), pp 21–30
24. Langendoen K, Reijers N (2003) Distributed localization in wireless sensor networks: a quantitative comparison. Comput Netw 43(4):499–518
25. Liu C, Wu K, He T (2004) Sensor localization with ring overlapping based on comparison of received signal strength indicator. In: Proceedings of the IEEE international conference on mobile ad-hoc and sensor systems, pp 516–518
26. Blumenthal J, Grossmann R, Golatowski F, Timmermann D (2007) Weighted centroid localization in zigbee-based sensor networks. In: Proceedings of the IEEE international symposium on intelligent signal processing (WISP'2007)
27. De San-Bernabé A, Martínez-de Dios JR, Ollero A (2012) A wsn-based tool for urban and industrial fire-fighting. Sensors 12(11):15,009–15,035
28. Patwari N, O'Dea RJ, Wang Y (2001) Relative location in wireless networks. In: Proceedings of the IEEE vehicular technology conference (VTC'2001), vol 2, pp 1149–1153
29. Thrun S, Burgard W, Fox D et al (2005) Probabilistic robotics, vol 1. MIT Press, Cambridge
30. Honkavirta V, Perala T, Ali-Loytty S, Piché R (2009) A comparative survey of wlan location fingerprinting methods. In: Proceedings of the workshop on positioning, navigation and communication, pp 243–251
31. Bahl P, Padmanabhan VN (2000) Radar: An in-building rf-based user location and tracking system. In: Proceedings of the annual joint conference of the IEEE computer and communications societies (INFOCOM'2000), vol 2, pp 775–784
32. Youssef M, Mah M, Agrawala A (2007) Challenges: device-free passive localization for wireless environments. In: Proceedings of the ACM international conference on mobile computing and networking, pp 222–229
33. Medeiros H, Park J, Kak A (2008) Distributed object tracking using a cluster-based Kalman Filter in wireless camera networks. IEEE J Sel Top Sign Process 2(4):448–463
34. Goshorn R, Goshorn J, Goshorn D, Aghajan H (2007) Architecture for cluster-based automated surveillance network for detecting and tracking multiple persons. In: Proceedings of the ACM/IEEE international conference on distributed smart cameras (ICDSC'07), pp 219–226
35. Ercan AO, El Gamal A, Guibas LJ (2007) Object tracking in the presence of occlusions via a camera network. In: Proceedings of the ACM international conference on information processing in sensor networks, pp 509–518
36. Song B, Roy-Chowdhury AK (2007) Stochastic adaptive tracking in a camera network. In: Proceedings of the IEEE international conference on computer vision (ICCV'2007)
37. Soto C, Song B, Roy-Chowdhury AK (2009) Distributed multi-target tracking in a self-configuring camera network. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'2009), pp 1486–1493
38. Fernández-Berni J, Carmona-Galán R, Martínez-Carmona J, Rodríguez-Vázquez A (2012) Early forest fire detection by vision-enabled wireless sensor networks. Int J Wildland Fire 21(8):938–949
39. Dore A, Cattoni AF, Regazzoni CS (2007) A particle filter based fusion framework for video-radio tracking in smart spaces. In: Proceedings of the IEEE conference on advanced video and signal based surveillance (AVSS'2007), pp 99–104

40. Miyaki T, Yamasaki T, Aizawa K (2007) Multi-sensor fusion tracking using visual information and wi-fi location estimation. In: Proceedings of the international conference on distributed smart cameras (ICDSC'07), pp 275–282
41. Merino L, Gilbert A, Capitán J, Bowden R, Illingworth J, Ollero A (2012) Data fusion in ubiquitous networked robot systems for urban services. Ann Telecommun 67(7–8):355–375
42. Maza I, Caballero F, Capitan J, Martinez-de Dios J, Ollero A (2011) A distributed architecture for a robotic platform with aerial sensor transportation and self-deployment capabilities. J Field Robot 28(3):303–328
43. Desai P, Rattan KS (2009) Indoor localization and surveillance using wireless sensor network and pan/tilt camera. In: Proceedings of the IEEE national aerospace & electronics conference (NAECON'09)
44. Sánchez-Matamoros JM, Martinez-deDios JR, Ollero A (2009) Cooperative localization and tracking with a camera-based WSN. In: Proceedings of the IEEE international conference on mechatronics (ICM'2009)
45. Jiménez-González A, Martınez-de Dios JR, de San Bernabé A, Ollero A (2011b) WSN-based visual object tracking using extended information filters. In: Proceedings of the international workshop on networks of cooperating objects
46. Coates M (2004) Distributed particle filters for sensor networks. In: Proceedings of the ACM international symposium on information processing in sensor networks, pp 99–107
47. Olfati-Saber R (2005) Distributed Kalman Filter with embedded consensus filters. In: Proceedings of the IEEE conference on decision and control and European control conference (CDC-ECC'05), pp 8179–8184

Chapter 4
Node Inclusion/Exclusion in Cluster-Based Tracking

4.1 Introduction

Cluster-based target tracking schemes require methods for dynamically including/excluding nodes in/from the cluster. As the target moves, new nodes are invited to participate in the cluster so that the target keeps being tracked with the sufficient accuracy. Also, the measurements from other nodes become useless for tracking and these nodes should be excluded from the cluster and turned off in order to reduce energy consumption. In most schemes nodes are active only while they are within the cluster. Hence, node inclusion/exclusion has critical impact on energy consumption and also on computational burden and bandwidth.

A significant variety of node selection methods have been reported. Most of them employ policies that estimate the usefulness of the measurements gathered by the nodes with the intention of keeping them active only when their measurements are necessary for tracking. The objective is to reduce the number of active nodes to the minimum by keeping in the cluster only the nodes that provide useful information for localization and tracking. This chapter describes the main existing methods and presents a sensor activation/deactivation technique devised for the general architecture presented in Chap. 2.

This chapter is structured as follows. Section 4.2 presents the sensor selection problem and Sect. 4.3 summarizes the main node and sensor selection methods reported for target tracking. Section 4.4 describes the sensor activation/deactivation technique devised for the architecture presented in Chap. 2. Section 4.5 evaluates and compares the described techniques. Conclusions is the final section.

4.2 The Sensor Selection Problem

The problem of node inclusion/exclusion from clusters is very similar to the well-known sensor selection problem in case that each node is equipped with only

J.R. Martínez-de Dios et al., *Cluster-based Localization and Tracking in Ubiquitous Computing Systems*, SpringerBriefs in Cooperating Objects,
DOI 10.1007/978-3-662-54761-8_4

one sensor. In this section we adopt this assumption. In the following sections this hypothesis will be relaxed and multi-sensor nodes will be considered.

Given a set of n sensors $S = \{S_1, ..., S_n\}$, the objective of the sensor selection problem is to determine the best subset S' of k sensors that satisfies the requirements of one or multiple goals. The best subset is that which achieves the required sensing accuracy while meeting the resource consumption constraints. Thus, in the sensor selection problem there is a trade-off between two objectives: (1) to collect as much information as possible in order to improve sensing and reduce uncertainty and (2) to reduce the consumption of resources. In sensor nodes most sensor selection techniques have considered energy as one of the main resources.

The notions of reward and cost are commonly used to model this trade-off. The reward is the increment in accuracy originated by the measurements gathered by the selected sensors. The cost is usually taken as the energy expended during the operation of the sensor nodes. Hence, the cost is often proportional to the number of active sensors. The objective of a sensor selection technique is to choose a subset S' of k sensors such that the total utility is maximized while the overall cost is lower than a certain budget.

Many different criteria have been considered in sensor selection schemes. The most widely employed are: the residual energy, the required coverage and the type of information required. Sensors can be selected to perform one specific goal or multiple goals. These goals can be related to the operation of the network, such as monitoring by ensuring complete coverage, or can also be more specific and application-oriented, such as target tracking. In [1] the main existing sensor selection schemes are classified.

4.2.1 Single and Multiple Mission Schemes

In single-mission schemes a sensor network performs a specific mission repeatedly over time. The goal is to select the set of sensors such that the mission is accomplished in the most efficient manner.

In [2] the objectives are defined using utility functions and cost models based on energy consumption. The utility function of a set of sensors depends on the number of sensors and their location. In [3] a generic framework in which the utility values of the sensors can be specified is proposed. The goal is to select the sequence of sensors in order to maximize the total utility function without exceeding the energy consumption constraint.

In multiple-mission schemes sensor network is considered as a multi-objective optimization problem. In [4] the goal is to cover the maximum number of targets with the minimum number of active directional sensors. Covering each target can be viewed as a different mission. The objective is to dynamically change the orientation of the sensors in order to cover as many targets as possible, while activating as few sensors as possible. It is shown that this problem can be formulated as an Integer Programming problem.

In [5] the advantages of applying manager-consumer concepts to sensor management is analyzed. The mission manager allocates budgets to the application. The consumer places bids to the sensor manager. The sensor manager allocates sensors to the missions based on these bids. The complexity of the model hampers its distributed implementation in ubiquitous computing systems.

4.2.2 Coverage Schemes

The goal is to select the sensors in order to ensure complete coverage of the environment. Complete coverage is achieved if every point in the scenario is within the sensing range of one or more sensors. If there is coverage redundancy, full coverage can be achieved using only a subset of active sensors while the rest can enter their sleep mode. This enlarges the network lifetime saving energy. Selection schemes are used to decide which sensors are to be turned on and for how long. The selection scheme typically operates in a dynamic manner, for example, if during the operation a node fails, the selection can be re-executed to restore full coverage.

In [6] the sensor nodes are divided into sets, such that each set is capable of providing complete coverage of the environment. The objective is to optimally select which set is active such that the lifetime of the application is maximized and certain levels of bandwidth and coverage constraints are met. In [6] it was proven that an optimal solution to this problem can be found using linear programming. If sensors are mobile (e.g. mounted on robots) a sensor node can move to a new location in order to fill a coverage hole. Even if a random deployment results in incomplete coverage, the nodes can relocate in order to ensure full coverage. Similarly, if a node fails, the network can be dynamically reconfigured to cover the new hole. However, its main drawback is the difficulty to implement this scheme in a distributed manner.

In [7] a bidding protocol is used for deciding which sensors should move to cover the holes. Static sensors discover coverage holes in the scenario using Voronoi diagrams. Each mobile sensor has a cost for serving one hole. The cost is related to the size of the new hole generated by its movement. The static sensors estimate the size of the coverage holes and place bids for the mobile sensors accordingly. Mobile sensors choose the highest bids and move to heal the largest coverage holes.

4.3 Sensor Selection for Target Localization and Tracking

In this case sensors are selected specifically to deal with target localization and tracking problems. The objective is to activate/deactivate nodes in order to improve sensing while reducing resource consumption. These schemes can be further classified according to the approach used in the sensor selection algorithm: (1) schemes based on the Mean Square Error (MSE), where the aim is to minimize the MSE of the estimation; (2) dynamic information-driven schemes, where the objective is to

maximize the information gained after the integration of the new measurements; and (3) entropy-based schemes, where the selection aims at minimizing the entropy of the estimation. Below the three categories are briefly presented.

4.3.1 Schemes Based on the Mean Square Error

Their goal is to minimize the Mean Square Error (MSE) of the localization and tracking estimations.

In [8] a node is kept active while its distance to the target is below a certain threshold. However, proximity does not imply that the sensor is capable of acquiring information about the target. In Wireless Camera Networks some schemes activate a camera when the target is estimated to enter its field of view and deactivate it when it is expected to come out of it [9]. This criterion keeps active many cameras unnecessarily unless the deployment has been optimized [10].

In [11, 12] several schemes that consider the acoustic properties of the target in order to estimate the direction of arrival are discussed. They use a global node selection scheme in order to determine which set of sensors should be active in order to locate the target. Initially, two sensors not collinear with the target are selected as the active set. After selecting the initial active set, other sensors are added adopting a step-by-step approach that minimizes the MSE of the estimation of the target location.

In [11] the active set is improved by continuously checking if replacing an active node with an inactive one will result in an improvement in localization accuracy. This solution requires global knowledge of the sensors location and thus cannot be easily distributed. The applicability of the scheme is limited to small networks because it incurs in high computational burden, especially when performing the exhaustive search to determine the initial set of active sensors. It also requires the interchange of a large number of packets to obtain a solution, which again constrains its scalability.

4.3.2 Information-Driven Schemes

Information-driven schemes select the nodes that provide the greatest improvement in the estimate of the target location at the lowest cost. In [13] this is solved considering an optimization problem defined in terms of the information gain and cost. The goal is to improve: tracking quality, scalability and resource consumption efficiency.

The scheme selects a node (cluster leader), which becomes active. The initial selection is performed by considering the predicted location of the target. After collecting the required measurements about the target, the leader node selects the next node that it believes it will be the most informative and transmits its measurements to it. The new node becomes the leader and this process continues as long as needed to track the target.

In order to decide which node is going to be the next leader, the current leader analyzes the information utility value of different candidate sensors. This value must be based only on available information such as a sensor's location, its modality and the current belief state. The authors adopt a definition of information utility based on distance. One of the drawbacks of this scheme takes place if the first leader is not close to the target location, due to an error in prediction. In this case the overall tracking quality will degrade and the whole process might fail.

A similar scheme is proposed in [14] for target tracking using video-based sensor networks. The main difference from the method described in [13] is that this work considers the initialization cost that nodes encounter when performing motion segmentation for target tracking. A set of nodes is selected in each time step such that their total information utility is maximized while the average energy is constrained by a certain bound. However, this scheme imposes a high computational burden on the nodes.

4.3.3 Entropy-Based Schemes

Entropy is a widely-used measurement of the uncertainty in a probability distribution. Several sensor selection techniques use entropy. In [15] the mutual information about the future state and the current node measurement is used to determine the information gain of activating different sensors. A greedy approach is used to solve the sensor selection problem. At each time an unused sensor with an observation that is expected to yield the maximum entropy reduction of the target location distribution is selected. This added observation is then used to determine the target location distribution. Sensor selection is performed until the entropy of the target location distribution becomes lower than a predefined value that reflects the required target location uncertainty.

In [16] activation/deactivation is formulated as a decision problem. Several sensor activation/deactivation actions are possible. The method selects the action that optimizes a greedy cost-reward utility function. The main problem of this scheme is that it requires a way to effectively evaluate the expected entropy reduction for the different candidate sensors. This is difficult to determine without actually retrieving the measurements from the different sensors. Also, this scheme is centralized in the sense that sensor selection decisions are made by a single node. This is not scalable and incurs in a high communication overhead which makes it unsuitable in many cases.

Work [17] deals with sensor selection using a heuristic-based technique that efficiently provides a sub-optimal solution instead of finding the optimal solution using mutual information, which is computationally intensive. Given a prior probability distribution of the target location and the locations and sensing models of a set of sensors, this technique selects an informative sensor such that the integration of the observation of the selected sensors with the prior target location distribution results in the greatest reduction in uncertainty. This heuristic adds one sensor at a time in order to reduce the entropy of the target location distribution. Although this solution is more efficient than [15], it is still centralized which constrains its scalability.

Table 4.1 Main existing cluster inclusion/exclusion techniques for localization and tracking in ubiquitous computing systems

Node inclusion/exclusion	Criterion	Burden	Energetic efficiency
[8]	Distance	Low	Low
[9]	Field of view	Low	Low
[11]	Estimation MSE	High	Medium
[13, 14]	Information-driven	High	Medium
[15, 17]	Entropy	High	High

4.3.4 Discussion

The main existing sensor selection schemes are shown in Table 4.1. Only activation/deactivation schemes based on entropy achieve high energetic efficiency reducing the number of nodes needed in the cluster to track the target. However, these techniques involve high computational burden and have bad scalability.

4.4 Sensor Selection Using Uncertainty-Based Decision Making

As concluded in Table 4.1 the sensor selection methods that adopt uncertainty-based decision making approaches achieve the best performance. These methods assume that in general not all sensors contribute equally to sensing and, in heterogeneous networks, not all types of sensors consume the same resources. These techniques activate/deactivate a sensor analyzing the usefulness of its measurements and the resources it consumes adopting a decision making approach that maximizes the trade-off between sensing gain and the cost in terms of resource consumption.

This section is divided in four parts. The first one presents a general reward-cost approach commonly adopted in these techniques. The following two describe cost and reward models commonly adopted. The final one describes the sensor selection techniques for different tracking problems: (1) using only camera measurements, (2) using only RSSI and (3) using both camera and RSSI measurements.

4.4.1 Reward-Cost Analysis for Sensor Activation

Let A_t be the universe of possible sensor activation/deactivation actions that can be performed by all the nodes within the cluster at time t. A_t includes actions in which one node is commanded to activate/deactivate its sensor. Each action a_t impacts on how well the target is sensed at cluster level. This sensing gain is interpreted as a

reward $r(a_t)$, which can be positive if the action reduces the uncertainty of the target state. The reward can also be negative. For instance deactivating a sensor can increase the uncertainty about the target state. Also, each action a_t is associated to a cost $c(a_t)$ that reflects the increase in resource consumption originated by the action. $c(a_t)$ can be positive if the action requires higher energy consumption (e.g. activation of one sensor) or negative, for instance when deactivating a sensor.

These methods select the action $a_t \in A_t$ that maximizes an utility function expressed as the difference between the reward and the cost:

$$J(a_t) = r(a_t) - \alpha c(a_t), \tag{4.1}$$

where α is a weighting factor.

The objective of the sensor selection technique is to choose the action that maximizes $J(a_t)$. Long-term optimization allows high flexibility but, on the other hand its implementation in ubiquitous computing systems involves significant computational burden. Also, long-term optimization scales badly with the problem size. Instead, greedy approaches that decide at time t the action that optimizes $J(a_t)$ in a

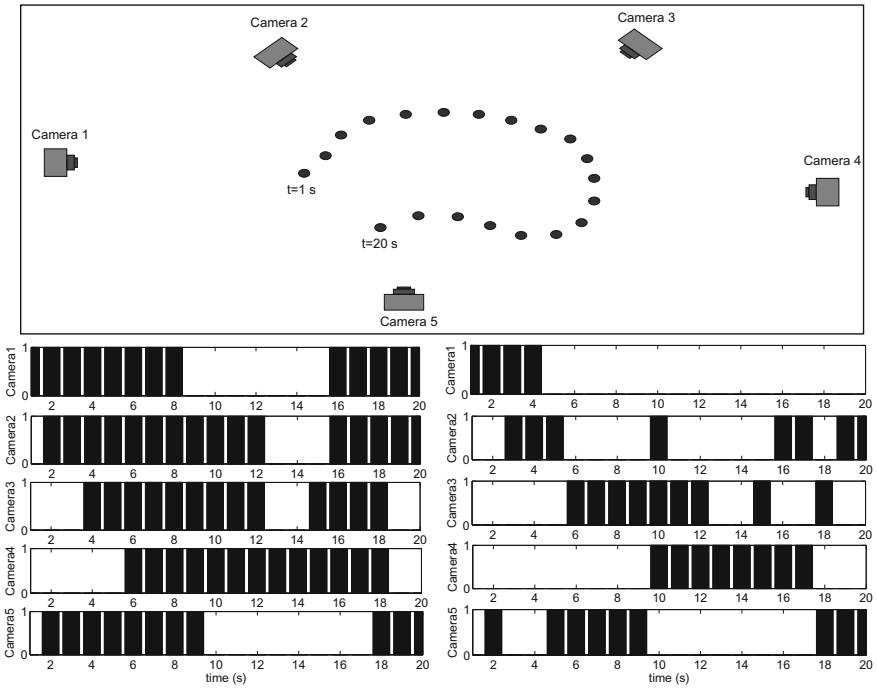


Fig. 4.1 *Top* Setting of 5 camera nodes tracking a target in a simulation. *Down-left* ON/OFF status of the cameras in case of using a low value of α. *Down-right* ON/OFF status of the cameras in case of using a high value of α

step-by-step basis are easily tractable. They require much lower burden and can be applied in COTS nodes.

In these techniques each node selects at each time t the action $\hat{a}_t$ that maximizes $J(a_t)$:

$$\hat{a}_t = \arg\max_{a \in A_t} (J(a_t)) \tag{4.2}$$

$\hat{a}_t$ is performed if $J(\hat{a}_t) > H$, where H is an hysteresis devised to avoid ping-pong effects in sensor activation/deactivation. In cases with $H = 0$ the action is performed simply if its reward overtakes its cost. The value of α establishes the trade-off between tracking accuracy and energy consumption.

As an example, Fig. 4.1-top depicts an scenario where a Wireless Camera Network with 5 camera nodes is tracking a target. Figure 4.1-down-left shows the resulting (ON/OFF) status of all the cameras at every time during the simulation when a low value of α is employed. Figure 4.1-down-right shows the results when a high value of α is used. As expected, low values of the weighting factor α tend to activate many cameras while high values tend to keep a low number of cameras active. For instance, with low values of α, *Camera2* is active 60% of the experiment but with high values, it is kept active only 20% of the time. Also, it is easy to notice that some cameras are more informative than others. For instance, *Camera4* is kept active for longer with both values of α.

4.4.2 Cost Model

$c(a_t)$ reflects the resource consumption required to perform action a_t. Assume that action a_t involves only one node, e.g. $a_t =$ *"Activate node j"*. The cost for activating a node should consider: (1) the resources consumed by the node that performed the action and (2) the currently existing resources in the node that performed the action. Thus, $c(a_t)$ can be expressed as follows:

$$c(a_t) = c_{cr}(a_t) c_{ar}(a_t) \tag{4.3}$$

$c_{cr}(a_t)$ is the cost of the resources consumed by node j when performing a_t. Energy consumption has been considered one of the main resources in ubiquitous computing technologies. $c_{cr}(a_t)$ is often taken as the energy consumed by node j.

$c_{ar}(a_t)$ is the cost of the resources currently available at node j. The goal of using $c_{ar}(a_t)$ is to distribute resource consumption among the nodes with the highest remaining resources. The following model has been adopted for $c_{ar}(a_t)$:

$$c_{ar}(a_t) = 1 + \frac{1}{E_{j,t} - E_{min}}, \tag{4.4}$$

where $E_{j,t}$ is the currently available energy at node j and E_{min} is the minimum energy for node operation.

It is easy to notice that $c_{ar}(a_t) \simeq 1$ if the node has high available energy but $c_{ar}(a_t)$ increases asymptotically as $E_{j,t}$ approaches E_{min}. Thus, the nodes with higher available energy are more likely to be activated.

The cost in case of deactivation actions, e.g. $a_t =$*"Deactivate node j"*, reflects the saving in resource consumption: $c(a_t) = c_{cr}(a_t)c_{ar}(a_t)$. In this case $c_{cr}(a_t)$ and $c_{ar}(a_t)$ play the same role. The goal of $c_{ar}(a_t)$ is also to distribute energy consumption: the nodes with lower available energy are more likely to be deactivated.

4.4.3 Reward Model

$r(a_t)$ reflects the change in sensing after changing the measurements that are integrated for tracking. In general this change will be positive (sensing improvement) for actions that activate nodes. The reward can also be negative. For instance deactivating a sensor can increase the uncertainty about the target state.

Sensing improvement can be expressed in terms of the uncertainty reduction about the target state. In the following the reward of performing action $a_t =$*"Activate sensor m"* is evaluated. $r(a_t)$ is the expected uncertainty reduction about the target state after performing a_t.

Shannon entropy has been considered one of the best and more widely applied metrics to measure uncertainty [18]. The entropy of an Gaussian statistical distribution x_t can be expressed as:

$$H(x_t) = \frac{1}{2}\log|\Sigma_t|, \tag{4.5}$$

where Σ_t the covariance matrix of the distribution x_t at time t.

Using entropy, the reward of action a_t can be expressed as the reduction in uncertainty resulting from the new measurements, which can be expressed as follows:

$$r(a_t) = H(x_{t+1}^{na}) - H(x_{t+1}^{a_t}), \tag{4.6}$$

where $H(x_{t+1}^{a_t})$ is the expected entropy at time $t+1$ if action a_t has been performed and $H(x_{t+1}^{na})$ is the expected entropy at time $t+1$ if no action has been performed. Using Eq. (4.5) and applying logarithm properties $r(a_t)$ can be computed as follows:

$$r(a_t) = \frac{1}{2}\log\frac{|\Sigma_{t+1}^{na}|}{|\Sigma_{t+1}^{a_t}|} \tag{4.7}$$

The information matrix is the inverse of the covariance matrix. Thus, Eq. (4.7) can be rewritten as:

$$r(a_t) = \frac{1}{2}\log\frac{|\Omega_{t+1}^{a_t}|}{|\Omega_{t+1}^{na}|}, \tag{4.8}$$

where Ω_{t+1}^{na} is the expected information matrix at time $t+1$ if no action has been performed and $\Omega_{t+1}^{a_t}$ is the expected information matrix at time $t+1$ if action a_t has been performed.

$\Omega_{t+1}^{a_t}$ and Ω_{t+1}^{na} are computed as follows. Assume that at time t node i is the cluster head and that TS_t is the set of non-cluster head nodes that are currently participating in the cluster. Ω_t is the updated information matrix. $\overline{\Omega}_{t+1}$ is the predicted information matrix computed at time t. We want to estimate at time t the expected reduction in uncertainty that a_t will cause at $t+1$.

If no action is performed, the updated information matrix for $t+1$ can be estimated at time t by adding to $\overline{\Omega}_{t+1}$, the contributions of the cluster head $\Omega_{i,t+1}$ and the contributions of the active non-cluster nodes:

$$\Omega_{t+1}^{na} = \overline{\Omega}_{t+1} + \Omega_{i,t+1} + \sum_{j \in TS_t} \Omega_{j,t+1}, \tag{4.9}$$

where $\Omega_{j,t+1}$ is the predicted contribution of node j to the EIF update computed as follows:

$$\Omega_{j,t+1} = H_{j,t+1}^T Q_{j,t+1}^{-1} H_{j,t+1}, \tag{4.10}$$

where $H_{j,t+1}$ is the predicted Jacobian for node j computed using μ_{t+1}.

Similarly, if action $a_t =$*"Activate sensor m"* is performed, the updated information matrix for $t+1$ can be estimated as:

$$\Omega_{t+1}^{a_t} = \overline{\Omega}_{t+1} + \Omega_{i,t+1} + \Omega_{m,t+1} + \sum_{j \in TS_t} \Omega_{j,t+1}, \tag{4.11}$$

which can be expressed as:

$$\Omega_{t+1}^{a_t} = \Omega_{t+1}^{na} + \Omega_{m,t+1} \tag{4.12}$$

On the other hand, if a_t is the deactivation of a currently active node (e.g. $a_t =$*"Deactivate sensor m"* the updated information matrix for $t+1$ can be estimated as:

$$\Omega_{t+1}^{a_t} = \Omega_{t+1}^{na} - \Omega_{m,t+1} \tag{4.13}$$

Then, the reward $r(a_t)$ can be computed as follows:

$$\begin{cases} r(a_t) = \frac{1}{2} \log \left(\frac{|\Omega_{t+1}^{na} + \Omega_{m,t+1}|}{|\Omega_{t+1}^{na}|} \right) & \text{if } a_t = \textit{"Activate sensor m"} \\ r(a_t) = \frac{1}{2} \log \left(\frac{|\Omega_{t+1}^{na} - \Omega_{m,t+1}|}{|\Omega_{t+1}^{na}|} \right) & \text{if } a_t = \textit{"Deactivate sensor m"} \end{cases} \tag{4.14}$$

For computing $r(a_t)$ the cluster head has to sum all the contributions and compute both determinants and the logarithm. All that should be performed for every possible action in A_t. An implementation based on entropy could be cumbersome and have bad

scalability and involve unsuitable burden for ubiquitous computing systems endowed with limited computational resources.

A wide variety of metrics have been used to measure the uncertainty in a statistical distribution. Some are based on the covariance matrix and others, on the information matrix. In [19] a number of metrics based on the determinant, maximum eigenvalues and trace of the covariance matrix were compared concluding that they all perform similarly. In the following we compute $r(a_t)$ using the trace of the information matrix, which provides interesting advantages for cluster-based schemes.

The reward $r(a_t)$ from performing action a_t is the reduction in uncertainty resulting from the new measurements. Using the trace of the information matrix as uncertainty metric, Eq. (4.6) can be rewritten as:

$$r(a_t) = tr(\Omega_{t+1}^{a_t}) - tr(\Omega_{t+1}^{na}), \tag{4.15}$$

where $\Omega_{t+1}^{a_t}$ and Ω_{t+1}^{na} are computed as Eqs. (4.11) and (4.9), respectively. Since $tr(A+B) = tr(A) + tr(B)$, it is simple to notice that the following expressions hold:

$$\begin{cases} r(a_t) = tr(\Omega_{m,t+1}) & \text{if } a_t = \text{“Activate sensor } m\text{”} \\ r(a_t) = -tr(\Omega_{m,t+1}) & \text{if } a_t = \text{“Deactivate sensor } m\text{”} \end{cases} \tag{4.16}$$

This method is very interesting in cluster-based schemes. First, it is computational efficient and suitable for distributed implementation. Each node can efficiently compute the reward of the actions in which it is involved using Eq. (4.16). Each node transmits to the cluster head the reward and the cost of the actions in which it is involved. The cluster head only has to select the action $a_t \in A_t$ that maximizes $J(a_t)$. Besides, the head does not require having any knowledge from any of the cluster nodes.

Below is an example that illustrates the validity of the trace of the information matrix as uncertainty metric. Figure 4.2 shows three different distributions and their respective values for both metrics. The first row in Fig. 4.2 shows 3σ plots of the uncertainty in three different Gaussian distributions. The second row, their respective information matrices. The bottom row shows the value of both metrics.

Assume that action a_{12} transforms *Distribution1* (shown in Fig. 4.2-left) into *Distribution2* (shown in Fig. 4.2-center). The uncertainty improvement in case of using entropy as uncertainty metric is $r(a_{12}) = \frac{1}{2}(log|\Omega_2| - log|\Omega_1|) = 2.29$. With the trace of the information matrix, the uncertainty improvement is $r(a_{12}) = tr(\Omega_2) - tr(\Omega_1) = 1.9439$. Assume that action a_{13} transforms *Distribution1* into *Distribution3* (shown in Fig. 4.2-left). The uncertainty improvement in both cases are $r(a_{13}) = \frac{1}{2}(log|\Omega_3| - log|\Omega_1|) = 1.4446$ using entropy and $r(a_{13}) = tr(\Omega_3) - tr(\Omega_1) = 1.0436$ using the trace of the information matrix. As expected, the information gain in both cases for both metrics is positive. Also, with both metrics $r(a_{12}) > r(a_{13})$. Besides, the values of the trace of the information matrix and entropy are consistent.

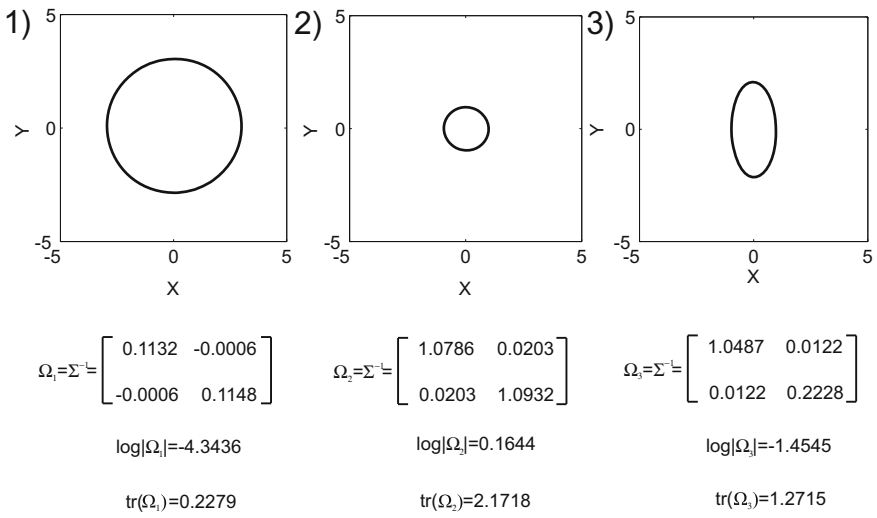


Fig. 4.2 Comparison between entropy and the trace of the information matrix as uncertainty metric in three Gaussian distributions

The reward resulting from action a_{23}, which transforms from *Distribution2* to *Distribution3* are $r(a_{23}) = \frac{1}{2}(log|\Omega_3| - log|\Omega_2|) = -0.8095$ and $r(a_{23}]) = tr(\Omega_3) - tr(\Omega_2) = -0.9003$. As expected the information gain for both metrics is negative.

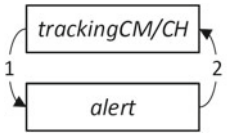
4.4.4 Node/Sensor Activation for Target Tracking

Below four different node/sensor activation/deactivation techniques for cluster-based target tracking are presented. We assume that at time t any node is at one of the following tracking modes:

- *trackingCH*, the node acts as cluster head;
- *trackingCM*, the node participates in the cluster but not as head;
- *alert*, the node is not currently participating in the cluster but it is at single-hop distance from the head and could be included in the cluster at that time if necessary;
- *inactive*, it is not involved in tracking and cannot be included in the cluster at that time. Nodes in *inactive* mode are in a low-power listening state.

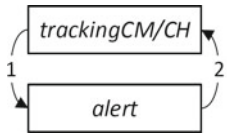
All nodes in the cluster that are in modes *trackingCH*, *trackingCM* and *alert* participate in the computation of costs and rewards of actions in the activation/deactivation method and transmit their contributions to the cluster head which selects the action that obtains the highest $J(a_t)$.

Any node in the *inactive* mode changes to *alert* when it receives a packet transmitted by the cluster head. The transitions between *alert* and *trackingCM* are



Action	Cost
1	-c2
2	c2

Fig. 4.3 Transition model and action costs in camera-only activation



Action	Cost
1	-c1
2	c1

Fig. 4.4 Transition model and action costs in RSSI-only activation

determined by the activation/deactivation method. The transitions between *trackingCM* and *trackingCH* are determined by the cluster head selection method that is presented in Chap. 5.

Camera-only activation

Target tracking is performed using only camera measurements. Each node in modes *trackingCH* or *trackingCM* produces camera measurements that are integrated for target tracking. Nodes in modes *alert* or *inactive* do not produce any measurement. Only one action is possible for each node, see Fig. 4.3-left: to deactivate the camera if the node is in modes *trackingCM* and *trackingCH*; and to activate the camera if the node is in mode *alert*.

The reward when activating a node is the information gain obtained by the new camera measurement. The cost of the resources consumed is $c_{cr}(a_t) = c2$, which is proportional to the energy consumed by the camera module when it is active.

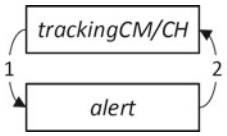
RSSI-only activation

Target tracking is performed using only RSSI measurements. Only the RSSI measurements from the nodes in modes *trackingCH* or *trackingCM* are integrated. Nodes in modes *alert* or *inactive* do not produce any RSSI measurement. Only one action is possible for each node, see Fig. 4.4: (1) nodes currently in modes *trackingCM* or *trackingCH* can be put in the *alert* mode; and nodes currently in *alert* can be put in modes *trackingCM* or *trackingCH*.

The cost of the resources consumed is $c_{cr}(a_t) = c1$, which is proportional to the energy consumed by the radio module when it is gathering a RSSI measurement. $c1$ is often very low and can be considered negligible when compared to $c2$.

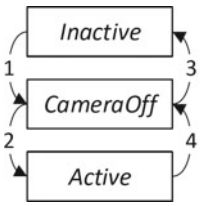
Joint camera-RSSI activation

Target tracking is performed integrating camera and RSSI measurements. Each camera node in modes *trackingCH* or *trackingCM* gathers both camera and RSSI measurements. Nodes in modes *alert* or *inactive* do not gather any measurements. Only one action is possible for each node, see Fig. 4.5. Nodes in the *alert* mode can be activated and nodes in modes *trackingCM* and *trackingCH* can be deactivated.



Action	Cost
1	-(c1+c2)
2	c1+c2

Fig. 4.5 Transition model and action costs in joint camera-RSSI activation



Action	Cost
1	c1
2	c2
3	-c1
4	-c2

Fig. 4.6 Transition model and action costs in gradual camera-RSSI activation

The reward when changing from *alert* to *trackingCM* is the information gain obtained by the new camera and RSSI measurements. The cost of the energy consumed is the energy consumed by the newly activated camera and radio module: $c_{cr}(a_t) = c1 + c2$.

Gradual camera-RSSI activation

Target tracking is performed integrating camera and RSSI measurements. This scheme is motivated by the high differences in energy consumption of cameras and nodes. This scheme allows higher flexibility than joint camera-RSSI activation. Nodes in the *TrackingCM* mode can be in modes: *Active*, in which the node gathers camera and RSSI measurements; or *CameraOff*, in which the camera is off and then-ode takes only RSSI measurements. Besides, nodes can be also in *alert* and *inactive* modes. In the *alert* mode, the node does not take any measurement but it is active. In the *inactive* mode the node is in low-energy mode.

Figure 4.6-left depicts the mode transition model adopted in this scheme. A node at mode *CameraOff* can be commanded to activate its camera ($Action2$) or can be fully deactivated ($Action3$). Each action involves different rewards and costs, see Fig. 4.6-right. $c2$ is the energy consumption of the camera module. The energy consumption increment when changing from *alert* to *CameraOff* is $c1$. Negative costs mean energy savings when stopping gathering camera and RSSI measurements.

4.5 Evaluation and Comparison

This section evaluates and compares different node inclusion/exclusion methods. The objective is to evaluate the node activation/deactivation module of the architecture in Chap. 2.

The impact of the node activation/deactivation module can be better analyzed when the module is integrated in the full schemes presented in Chap. 2. All the experiments shown in this section were performed adopting the distributed EIF

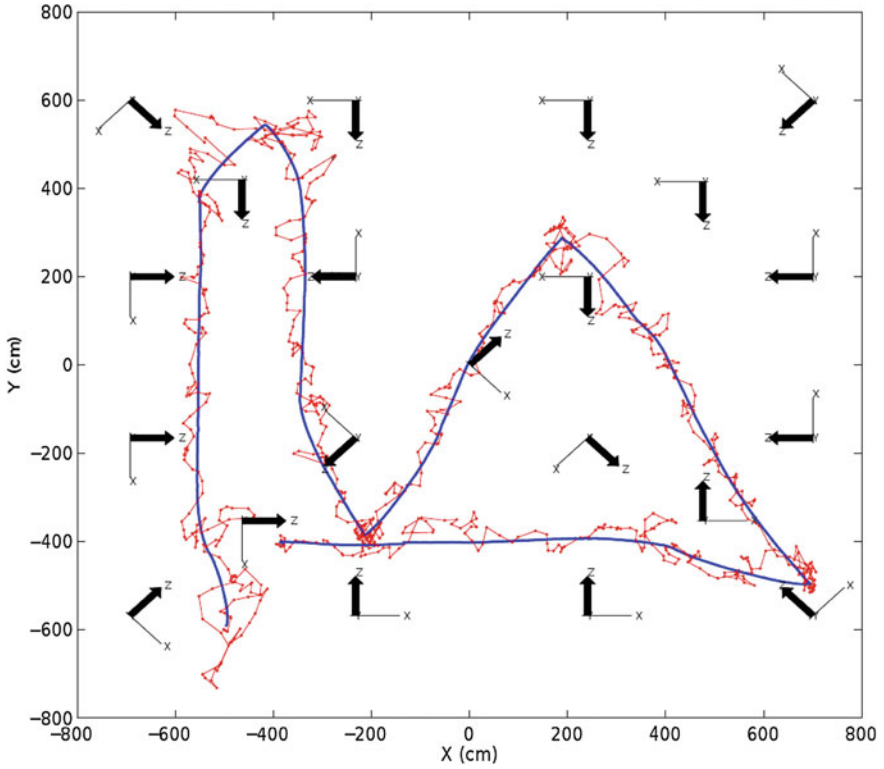


Fig. 4.7 Setting and result of the camera-only tracking in an experiment: ground truth is the *solid blue line* and the resulting estimated location is the *red line*. The optical axis Z of every camera is represented in the XY plane

presented in Sect. 3.5 as the measurement integration (*Module1*). In all these schemes the cluster head was selected as the closest node to the target estimated location.

The experiments were performed in the Ubiquitous Localization Testbed (UBILOC) presented in Chap. 2. A total of 21 camera-nodes with different orientations have been deployed on the ground (35 cm height), see Fig. 4.7.

Each camera node is composed of a TelosB WSN node and a CMUcam3 module. TelosB and CMUcam3 can be configured in different modes. Table 4.2 shows the energy consumptions for each mode reported by datasheets [20, 21]. Nodes in *trackingCH* always have the camera and TelosB on. Those in *trackingCM* have the camera on but the TelosB radio module is off when not necessary. Nodes in *alert* have the camera off. Nodes in *inactive* are in standby but they randomly wake up, take an image and process it to detect targets, which requires having the camera and TelosB active 4% of the time.

Table 4.2 Energy consumption of TelosB and CMUcam3 in different modes

Devices operation	Current (mA)	Voltage (V)
CMUcam3 ON (all modules)	130	5
CMUcam3 OFF	0	5
TelosB MCU ON/radio RX	23	3
TelosB MCU ON/radio OFF	2.4	3
TelosB MCU standby/radioOFF	0.021	3

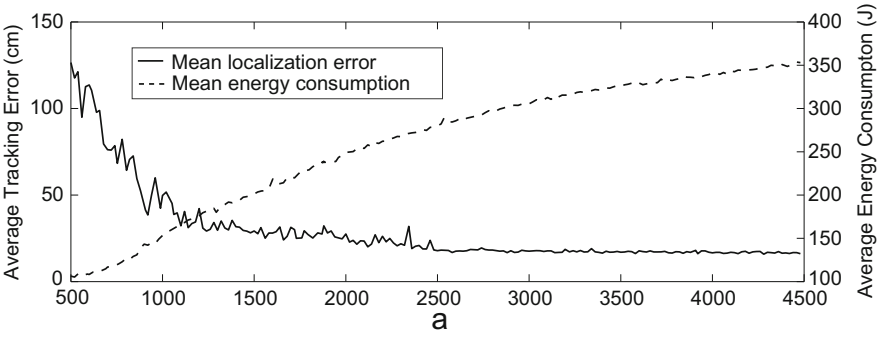


Fig. 4.8 Average tracking error and energy consumption for $\alpha = [500 - 4500]$. For each value of α the figure represents average values of 100 different tracking scenarios

4.5.1 Evaluation of Camera-Only Activation

The camera-only activation technique selects at each time t the action that achieves the highest value of the gain-cost utility function. α is used to weigh gain and cost in $J(a_t)$. Figure 4.8 analyses the influence of α on tracking performance. For each value of α the figure shows the average tracking error and the energy consumption in 100 different simulations with randomly deployed cameras. As expected, α balances between the tracking error and the energy consumption. $\alpha = 1500$ showed a good balance and it was used in all simulations and experiments.

The objective of this section is to compare the described activation method against other methods. Hence, all of them use the same measurement integration method (distributed EIF) and the same cluster selection method, which assigns as cluster head the node that is located closest to the target estimated location.

The following compares two versions of the described camera activation mechanism. *V1* considers actions that affect only one node. *V2* includes also compound actions that involve two nodes at the same time. Figure 4.9 compares the average tracking error and energy consumption in 300 random simulations, only 100 are shown for clarity.

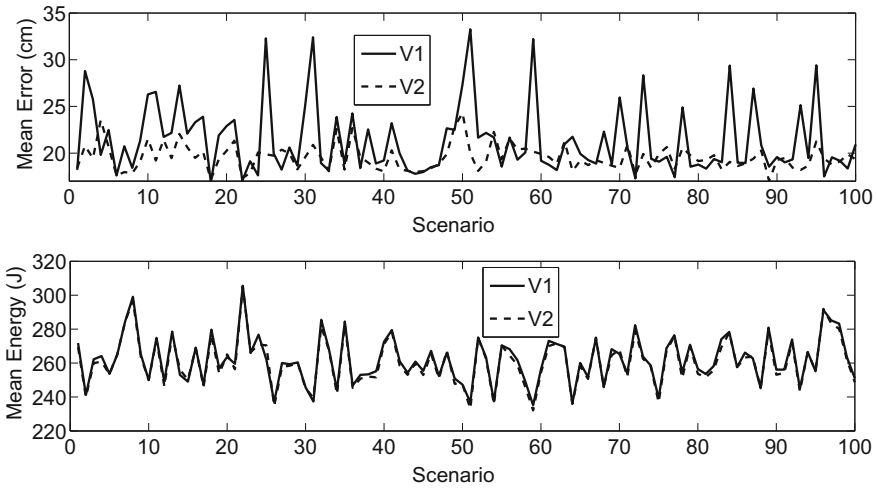


Fig. 4.9 Comparison between *V1* and *V2*: *top* average error and *bottom* average energy consumption

Method *V2* achieved an average tracking error 8% lower than *V1* and both consumed almost the same energy. Figure 4.9 is zoomed for better visualization. The universe of actions in method *V2* includes that in *V1*. Thus, the actions selected in *V2* will be at least as good as those selected in *V1*. However, *V1* and *V2* have high burden differences. In a cluster with N nodes in modes *trackingCH*, *trackingCM* and *alert*, the size of the action universe for *A1* is $N-1$, whereas the action universe for *V2* is $N(N-1)/2$. Their burden is proportional to the action universe size. Thus, *V2* requires $N/2$ times more burden than *V1*. Hence, *V1* is preferred due to its scalability.

Next, the described camera-only activation technique (*V1*) is compared to other reported camera activation techniques. In *V3*, each camera node is kept active while it senses the target—as e.g. in [9]. In *V4* a camera is kept active if it senses the target and if its distance to the target is below a value. Figure 4.10 compares the tracking error and the energy consumed in *V1*, *V3* and *V4* in 100 scenarios.

V3 and *V4* integrate a higher number of measurements. They achieve the lowest mean error but the highest mean energy consumption. The average error for *V1* was 21.53 cm, higher than *V3*, but still suitable for most applications, and its energy consumption was 252.12 J, 65.47% lower than *V3* and 42.41% lower than *V4*. *V1* activates only the nodes that contribute more to sensing, leaving the rest inactive. The balance between tracking error and energy consumption can be set modifying α as shown in Fig. 4.8.

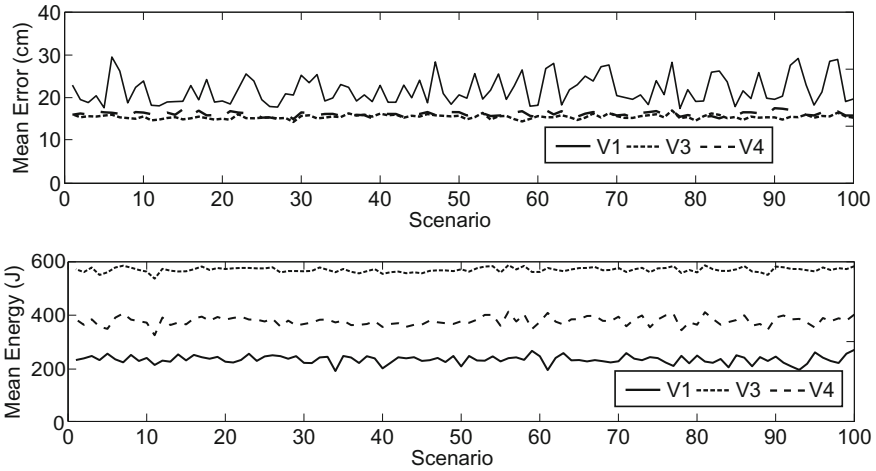


Fig. 4.10 Comparison between techniques *V1*, *V3* and *V4* in 100 scenarios: *top* average tracking error and *down* average energy consumption

4.5.2 Evaluation of RSSI-Only Activation

In this scheme only the RSSI measurements from the nodes in modes *trackingCH* or *trackingCM* are integrated in the distributed EIF. Nodes in *alert* mode also interchange packets within the cluster but their RSSI measurements are not integrated in the EIF. Only one action is possible for each node. Nodes in mode *trackingCM* can be put in mode *alert* and nodes in mode *alert* can be put in mode *trackingCM*. In this case deactivating a node does not turn off its radio module, it only implies that its RSSI measurements are not integrated in the EIF. Due to the high noise levels of RSSI measurements, the lowest localization uncertainty is not always obtained integrating as many measurements as possible. Using measurements from the most informative nodes usually obtains significantly lower uncertainty, as it will be shown in the following.

Next the described RSSI-only activation technique (called *V5*) is compared to *V6*, a technique in which the measurements of all the nodes that sense the target are integrated in the EIF. The experiment is similar to that in the previous subsection. It uses the same settings and includes simulations with 100 random settings. Also, both use as cluster head the node that is located closest to the target estimated location.

Figure 4.11 compares the performance of *V5* and *V6*. The energy consumption is very similar in both, as expected. However, they differ in the average number of active nodes and in the average tracking error. *V6* activates almost all the nodes in this setting, its average number of active nodes is 18.56. On the other hand, *V5* only needs to keep active an average of 9.72 nodes, 50% less than *V6*. The mean tracking error in *V6* is 211.7 cm while in *V5* is 181.8 cm, 14.1% lower, but in some cases the use of the activation/deactivation method improves tracking accuracy in 25%.

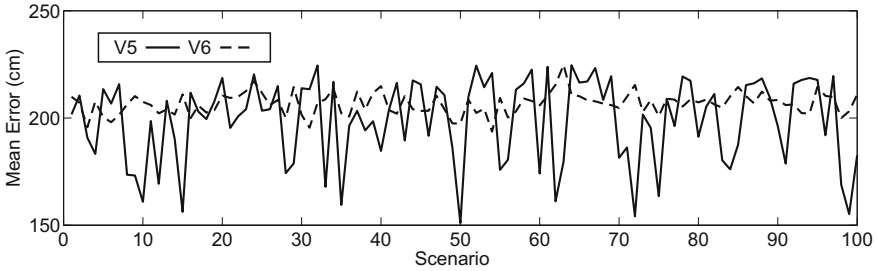


Fig. 4.11 Mean tracking error in *V5* and *V6*. Energy consumption is the same in both cases and it is not shown

Table 4.3 Evaluation and comparison of technique *V1*, *V7* and *V8*

	V1	V7	V8
Mean error (cm)	21.53	21.82	22.22
Average number of active cameras	4.59	3.11	2.56
Average number of RSSI measurements	–	3.11	3.38
Average energy (J)	252.5	172.9	147.9

V5 integrates measurements from fewer nodes but achieves lower tracking error than *V6*. RSSI measurements have high noise levels and this noise is not Gaussian. The integration of measurements gathered by distant nodes can degrade tracking accuracy. *V5* overcomes this problem because distant nodes do not usually achieve enough information gain to be activated.

4.5.3 Evaluation of Joint Camera-RSSI Activation

The adopted camera-RSSI activation technique is evaluated in the context of *Scheme3*, presented in Chap. 2. These experiments assume that the measurement integration adopted (*Module1*) is the EIF and that the cluster head selection method used (*Module3*) chooses as cluster head the closest node to the target estimated location.

Only one action is possible for each camera node: activate the full node, which involves gathering camera and RSSI measurements, or deactivate the node. The method with joint camera-RSSI activation is called *V7*. It is also compared against the gradual camera-RSSI activation, called *V8*, which is described in the next section. Table 4.3 and Fig. 4.12 compare the performance of *V7* against *V1*, which integrates camera-only measurements. A total of 100 experiments in random settings were performed.

The mean tracking error is similar in both (difference is lower than 1%), but they differ in the average number of active nodes and in the average energy consumed.

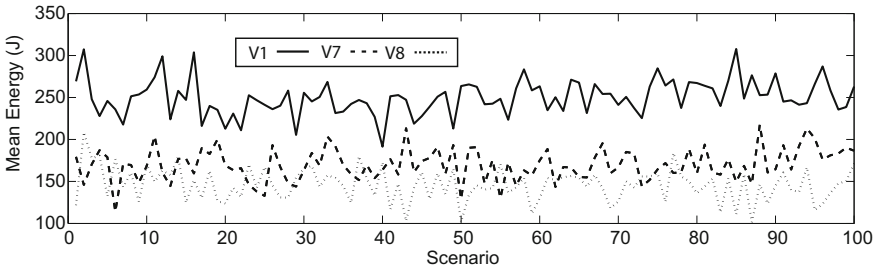


Fig. 4.12 Mean energy consumption in techniques *V1*, *V7* and *V8*. Tracking error is very similar (difference lower than 1%) and is not shown

The joint use of cameras and RSSI measurements in *V7* achieves similar tracking error to *V1* but consumes 32% less energy and requires activating 33% less nodes.

4.5.4 Evaluation of Gradual Camera-RSSI Activation

The scheme with gradual camera-RSSI activation technique will be called *V8*.

Table 4.3 and Fig. 4.12 compare the performance of *V8* against *V1*, the camera-only activation technique, and *V7*, the joint RSSI-camera activation technique. The mean localization error is similar in both techniques (difference is lower than 1%) but they differ in the average number of active nodes and in the average energy consumed. *V8* achieves similar tracking error to *V7* and *V1* activating less cameras. *V8* uses the same number of RSSI measurements but 18% less camera measurements. Since the consumption of nodes in *CameraOff* mode is significantly lower than the consumption of nodes in *Active*, *V8* consumes 15% less energy than *V7* and 42% less than *V1*.

In *V8* it is possible to integrate independently RSSI measurements and/or camera measurements from the same node. This enables flexible behaviors in which the camera of some nodes are turned off since camera measurements are energetically expensive while their RSSI measurements are integrated in the distributed EIF. This flexibility enables significant energy consumption savings.

4.6 Conclusions

This chapter deals with node inclusion/exclusion techniques that determine actuations in order to optimize the trade-off between performance and energy consumption in cluster-based tracking. Nodes are kept turned on only while they are within the cluster. Hence, these techniques have significant impact on energy consumption

enlarging network lifetime by keeping active only the sensors with positive sensing-consumption balance.

This section describes the main approaches used to address the sensor selection problem. The techniques used by existing works to activate/deactivate nodes in cluster-based tracking have been briefly analyzed. A number of techniques for these schemes have been developed. However, they were designed for traditional camera networks and for WSNs but not for WCNs characteristics, capabilities and constraints.

This chapter also presented a sensor activation/deactivation method that addresses the sensor selection problem by choosing the action that optimizes a greedy criterion based on the sensing-consumption trade-off. Four different techniques of the activation/deactivation method have been presented depending on the type of measurements gathered by the sensors and integrated for target tracking: (1) camera-only activation, (2) RSSI-only activation tracking, (3) joint camera-RSSI activation and (4) gradual camera-RSSI activation.

The presented methods adopt an optimal probabilistic approach. They use the trace of the information matrix as uncertainty metric, enabling efficient distributed implementation and integration with the EIF adopted for measurement fusion, see Chap. 3. In the experiments and simulations performed these methods significantly reduced computational burden, energy consumption and improved scalability.

References

1. Rowaihy H, Eswaran S, Johnson M, Verma D, Bar-Noy A, Brown T, La Porta T (2007) A survey of sensor selection schemes in wireless sensor networks. In: Proceedings of SPIE defense and security symposium, pp 65,621A–65,621A
2. Byers J, Nasser G (2000) Utility-based decision-making in wireless sensor networks. In: Proceedings of IEEE workshop on mobile and ad hoc networking and computing
3. Bian F, Kempe D, Govindan R (2006) Utility-based sensor selection. In: Proceedings of IEEE conference on information processing in sensor network (IPSN'2006)
4. Ai J, Abouzeid A (2006) Coverage by directional sensors in randomly deployed wireless sensor networks. J Comb Optim 11:21–41
5. Mullen T, Avarasala V, Hall D (2006) Customer-driven sensor management. IEEE Intell Syst 21:41–49
6. Perillo M, Heinzelman W (2003) Optimal sensor management under energy and reliability constraints. In: Proceedings of IEEE conference on wireless communications and networking
7. Wang G, Cao G, LaPorta T (2003) A bidding protocol for deploying mobile sensors. In: Proceedings of IEEE conference on network protocols (ICNP'2003)
8. Zhang W, Cao G (2004) DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks. IEEE Trans Wirel Commun 3(5):1689–1701
9. Medeiros H, Park J, Kak A (2008) Distributed object tracking using a cluster-based Kalman filter in wireless camera networks. IEEE J Sel Topics Sig Process 2(4):448–463
10. Ercan A, Yang D, El Gamal A, Guibas L (2006) Optimal placement and selection of camera network nodes for target localization. In: Distributed computing in sensor systems, pp 389–404
11. Zhao F, Shin J, Reich J (2006) Global node selection for localization in a distributed sensor network. IEEE Trans Aerosp Electron Syst 42:113–135

12. Zhao F, Shin J, Reich J (2006) Local node selection for localization in a distributed sensor network. IEEE Trans Aerosp Electron Syst 42:136–146
13. Zhao F, Shin J, Reich J (2002) Information-driven dynamic sensor collaboration. IEEE Sig Process Mag 19(2):61–72
14. Pahalawatta P, Pappas P, Katsaggelos A (2004) Optimal sensor selection for video-based target tracking in a wireless sensor network. In: Proceedings of international conference on image processing (ICIP'2004)
15. Ertin E, Fisher J, Potte L (2003) Maximum mutual information principle for dynamic sensor query problems. In: Proceedings of IEEE international workshop on information processing in sensor networks (IPSN'2003)
16. Jiménez-González A, Martınez-de Dios JR, de San Bernabé A, Ollero A (2011) WSN-based visual object tracking using extended information filters. In: Proceedings of international workshop on networks of cooperating objects
17. Wang H, Yao K, Pottie G, Estrin D (2004) Entropy-based sensor selection heuristic for target localization. In: Proceedings of 3rd international conference on information processing in sensor networks (IPSN'04)
18. Gray RM (2011) Entropy and information theory, vol 1. Springer, New York
19. Wenhardt S, Deutsch B, Angelopoulou E, Niemann H (2007) Active visual object reconstruction using d-, e-, and t-optimal next best views. In: Proceedings of IEEE conference on computer vision and pattern recognition
20. Moteiv (2004) Telos (rev b) datasheet. http://www.moteiv.com/
21. Rowe A (2012) CMUcam3 datasheet. http://www.cmucam.org/

Chapter 5
Cluster Head Selection for Target Tracking

5.1 Introduction

The performance of cluster-based target tracking highly depends on which node acts as cluster head. This chapter deals with cluster head selection. The objective is to improve tracking performance selecting which of the nodes in the cluster is the most suitable to perform the cluster head role.

Various cluster head selection techniques have been developed. Some of them employ criteria based on the proximity to the target estimated location. However, the node that collects more information about the target is not necessarily the closest to the target. Others use information-driven criteria trying to optimize the improvement in the estimation of a target location at the lowest cost. Others intend to homogenize energy consumption and rely on criteria based on energy since cluster head nodes consume significantly more resources than non-head nodes. However, these criteria do not reduce the overall energy consumption of the network.

This chapter summarizes the main existing cluster head selection techniques and presents a cluster selection technique for the architecture described in Chap. 2.

The outline of this chapter is as follows. The cluster head selection problem is described in Sect. 5.2. A method based on information gain is presented in Sect. 5.3. The described method is briefly evaluated in Sect. 5.4.

5.2 The Cluster Head Selection Problem

Tracking performance strongly depends on which node acts as cluster head. Figure 5.1 shows a simple example. Nodes A, B and C actively participate in a cluster, tracking a target. Node D is in *alert* mode. The rest are inactive. Assume that links $A - B$ and $A - C$ have high Packet Reception Rate (PRR). If node A is selected as the cluster head, it will receive—and integrate—with high probability

J.R. Martínez-de Dios et al., *Cluster-based Localization and Tracking in Ubiquitous Computing Systems*, SpringerBriefs in Cooperating Objects,
DOI 10.1007/978-3-662-54761-8_5

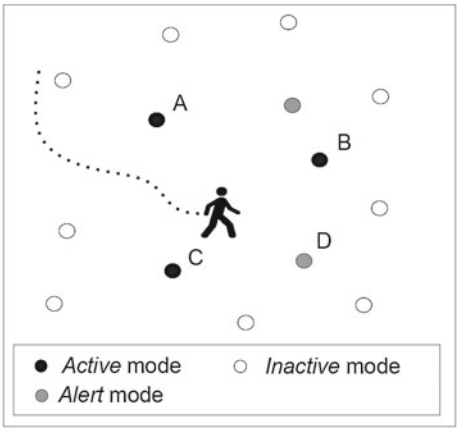


Fig. 5.1 Cluster head selection example

the measurements from B and C, or EIF update contributions in case of using the distributed EIF presented in Sect. 3.5. Assume that link $B - C$ has low PRR. If node B is the cluster head, it will receive with high probability the contributions from A but not those from C. Thus, if node A is the cluster head, it will integrate more measurements than if B is head, resulting in general in a more accurate estimation. Also, when combined with a node activation/deactivation mechanism, if node B is head it will naturally tend to reduce uncertainty by activating other nodes, such as D. Bad cluster head selection involves higher estimation uncertainty and higher resource consumption.

Various cluster head selection techniques have been developed. Below the main existing methods are summarized.

Cluster-head selection can be interpreted similarly to the node activation/deactivation problem. In the latter the problem is to select the modes of the nodes between active and inactive. Now, the problem is to decide on the role of the node—cluster head or not.

Some techniques address cluster head rotation using simple criteria that can be easily obtained by the network. For instance, a number of techniques assign the cluster head role to the node that is the closest to the target estimated location, see [1]. These methods assume that the closest nodes will gather more information about the target than distant nodes. However, the node that collects more information about the target is not necessarily the closest to the target.

Other approaches employ dynamic information-driven methods, where the objective is to maximize the information gain. In [2] an information-driven cluster head selection technique for target tracking is proposed. They consider the problem of selecting the node that provides the greatest improvement in the estimate of the target location at the lowest cost. This is solved as an optimization problem defined in terms of information gain and cost. The goal is to improve: detection and tracking quality, scalability and resource usage. The initial selection is performed by considering the predicted location of the target. After collecting the measurements of the

target, the cluster selects the next node that it believes is the most informative and transmits its measurements to it. One of the drawbacks of this technique is that if the first cluster head is not close to the target location, due to an error in prediction, the overall tracking quality will degrade and the whole process might fail.

Some techniques intend to homogenize energy consumption and rely on criteria based on energy [3] since cluster head nodes consume significantly more energy than non-head nodes. However, this criterion is not useful to improve sensing or to reduce the overall energy consumption.

5.3 Cluster Head Selection Based on Information Gain

This section describes an efficient method that selects as cluster head the node that can obtain the lowest uncertainty integrating the measurements of the nodes already participating in the cluster. It can be implemented in distributed schemes in constant time since the burden of the method is shared among all cluster nodes. It is suitable to be integrated in the architecture presented in Chap. 2.

This method estimates how well each node within the cluster will perform if it is selected as cluster head. The goodness of node i as cluster head can be measured using EU_i: the estimated effective uncertainty about the target at time $t+1$ if node i is selected as cluster head. Assume that at time t, S_t is the set of active nodes in the cluster. If node i is selected as the new head, at time $t+1$ it will receive measurements—or EIF update contributions—from nodes $j \in S_t$ and the updated information matrix will be as follows:

$$\Omega_{t+1} = \overline{\Omega}_{t+1} + \sum_{j \in S_t} \Omega_{j,t+1} \tag{5.1}$$

This expression assumes that the contributions of each node reaches the cluster head. Transmission errors in ubiquitous computing technologies cannot be disregarded. Recalling the distributed measurement integration implementation in Chap. 3, the contribution from node j reaches the cluster head (i.e. node i) if node j receives the *UpdateReq* packet and if the response from node j in a *UpdateResp* packet reaches the cluster head. Assuming both events are statistically independent and assuming symmetric Packet Reception Rate (PRR) the probability that the contribution from node j reaches the cluster head is $p_{j,i}^2$, where $p_{j,i}$ is the PRR between node i and j.

For cluster head selection we also adopt the trace of the information matrix as uncertainty metric, as we did in Chap. 4. In this case EU_i is computed as follows:

$$EU_i = tr(\overline{\Omega}_{t+1}) + \sum_{j \in S_t} p_{j,i}^2 tr(\Omega_{j,t+1}), \tag{5.2}$$

where $p_{j,i}$ is the PRR between node j and node i.

Recalling from Chap. 3, $\overline{\Omega}_{t+1}$ is known by the cluster head. Hence, it is not transmitted and not affected by $p_{j,i}^2$. The cluster head also provides its contribution $\Omega_{i,t+1}$ but of course $p_{i,i} = 1$. The best head is the node that maximizes EU_i:

$$CL = \arg\max_{i \in S_t}(EU_i) \tag{5.3}$$

$\overline{\Omega}_{t+1}$ is independent of i and hence Eq. (5.3) is rewritten as:

$$CL = \arg\max_{i \in S_t}\left(\sum_{j \in S_t} p_{j,i}^2 tr(\Omega_{j,t+1})\right) \tag{5.4}$$

If $EU_{CL} > 0$ and CL is not the current cluster head, it is advantageous to change the head. In this case the old head transmits to the new one Ω_t and μ_t, all it needs to keep with the cluster head role.

The algorithm of the method is shown in Algorithm 4. This method prioritizes nodes that are well communicated with the cluster members. Badly communicated nodes are not likely to be selected as clusters. If all the nodes have similar PRR, it favors the nodes that contribute more to the target estimation. This method is efficient and suitable for distributed implementation. Each node j computes individually $p_{j,i}$ and $tr(\Omega_{j,t+1})$ and transmits them to the cluster head, which efficiently selects CL using Algorithm 4. The cluster head does not require to have any knowledge from any node: nodes transmit to the cluster head everything it needs to determine the new cluster head.

Algorithm 4 Cluster head selection

Require: $p_{j,i}, tr(\Omega_{j,t+1})\ \forall j, i \in S_t$
1: **for** $i = 1 \rightarrow S_t$ **do**
2: $\quad ug_i = \sum_{j \in S_t} p_{j,i}^2 tr(\Omega_{j,t+1})$
3: **end for**
4: $CL = \arg\max_{i \in S_t}(ug_i)$
5: **if** $(EU_{CL} > 0)$ **then**
6: $\quad$ Assign CL as the new cluster head
7: **end if**

5.4 Evaluation

This section evaluates and compares the presented cluster head selection method. The impact of the method can be better analyzed when integrated in the full cluster-based target tracking schemes presented in Chap. 2. These experiments were performed

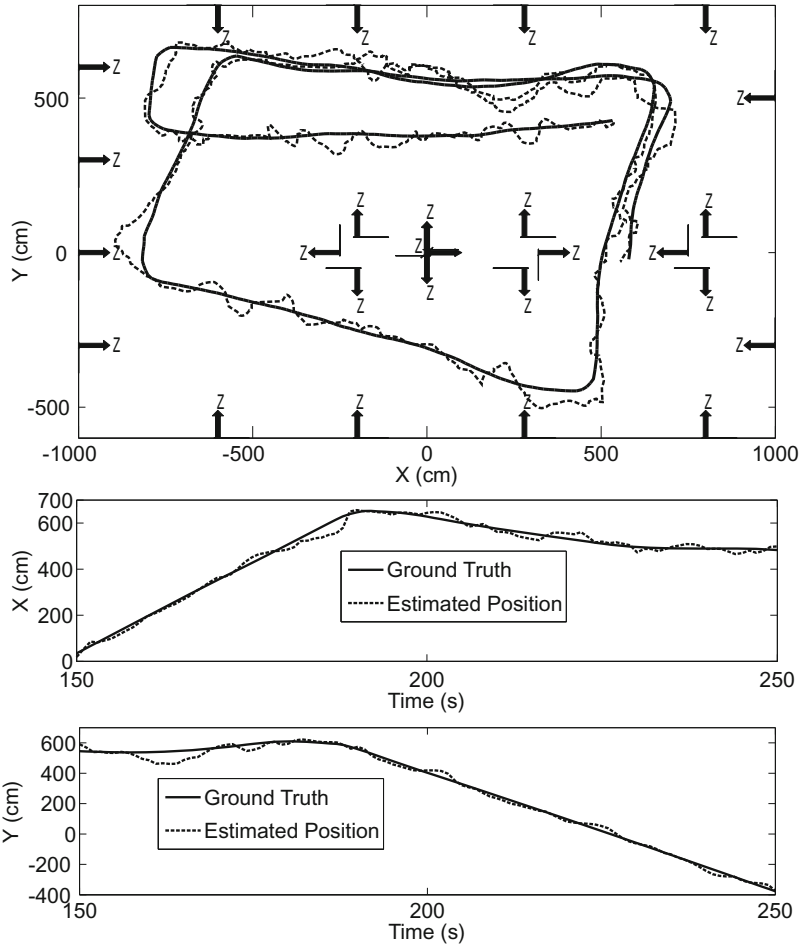


Fig. 5.2 Result of *Scheme1* in a real experiment in the $X - Y$ plane (*top*) and in axes X and Y (*bottom*)

taking the distributed EIF presented in Sect. 3.5 as the measurement integration in *Module1* and the sensor activation/deactivation method presented in Chap. 4 as *Module2*.

The experiments were performed in UBILOC, the localization testbed presented in Chap. 2. Figure 5.2-top shows the result of *Scheme1* in one experiment. The ground truth is in solid line and the estimated locations, in dashed. The average error was 21.91 cm. The average tracking error in axes X and Y was 15.85 cm and 19.42 cm, respectively. Figure 5.2-bottom shows the results obtained for axes X and Y between $t = 150$ s to $t = 250$ s.

This section analyzes and compares the adopted cluster head selection method. *C1* is the presented method. *C2* is a method that selects as cluster head the closest

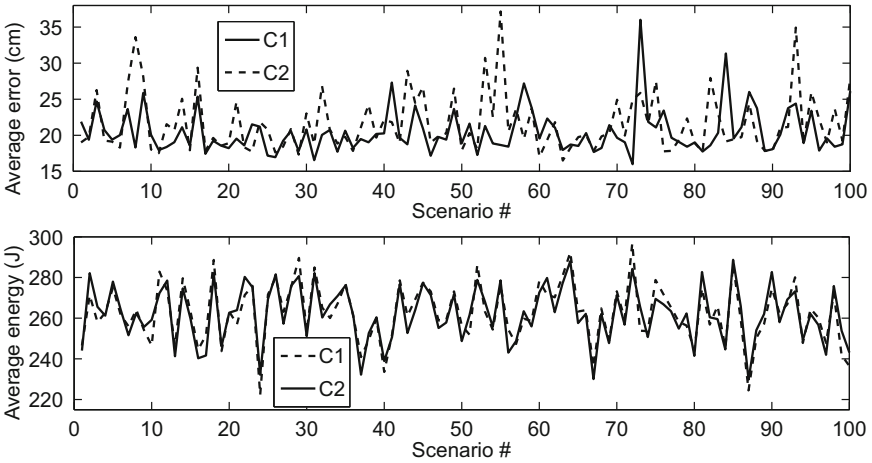


Fig. 5.3 Performance of *C1* and *C2* in 100 experiments in random scenarios: *top* average tracking error and *bottom* average energy consumption

active node to the target estimated location. The performance of *C1* and *C2* in 100 experiments in random scenarios and many different target trajectories is shown in Fig. 5.3.

C1 achieved a tracking error 8.06% lower than *C2*: 20.18 cm against 21.95 cm. The average energy consumptions differed in less than 0.2%. Also, the fluctuations of the performance of *C1* with the scenario are lower with *C1* than with *C2*. Although the advantages of *C1* over *C2* are not high, performing *C1* requires very low effort. There is not need to compute $p_{j,i}^2$ and $tr(\Omega_{j,t+1})$, since they were computed for the sensor activation/deactivation module.

5.5 Conclusions

This chapter deals with cluster head selection techniques. The tracking performance highly depends on which node performs the cluster head role. The objective of these techniques is to select which of the nodes in the cluster is the most suitable to take the cluster head role.

Existing techniques perform cluster head selection using criteria such as proximity based on the idea that the cluster head should be well connected to all the cluster members. Others use criteria based on energy consumption homogenization since cluster head nodes consume significantly more resources than non-head nodes.

This chapter also presented a method that selects as cluster head the active node that can expectedly obtain the lowest tracking uncertainty integrating the measurements from the currently active nodes. This method takes into account the Packet Reception Rates between the nodes and use uncertainty metrics based on the trace

of the information matrix to analyze all the active nodes and select the most suitable as cluster head.

References

1. Medeiros H, Park J, Kak A (2008) Distributed object tracking using a cluster-based kalman filter in wireless camera networks. IEEE J Sel Top Sig Process 2(4):448–463
2. Zhao F, Shin J, Reich J (2002) Information-driven dynamic sensor collaboration. IEEE Sig Process Mag 19(2):61–72
3. Kumarawadu P, Dechene DJ, Luccini M, Sauer A (2008) Algorithms for node clustering in wireless sensor networks: a survey. In: Proceedings of the IEEE international conference on information and automation for sustainability (ICIAFS'2008), pp 295–300

Chapter 6
Conclusions

Target localization and tracking are critical in ubiquitous computing systems and technologies. Although localization and tracking in outdoors have been significantly improved in the last years due to the wide adoption of GPS, in indoors they still attract significant R&D interest. A very high variety of localization and tracking approaches, techniques and technologies have been developed in the recent years. However, none of the proposed sensors, techniques or systems is ideal for all the problems and the selection of the method highly depends on the specific requirements and constraints of the application and of the environment.

Clustering is the most widely employed and researched approach in target localization and tracking in ubiquitous computing systems. Clustering naturally solves scalability and simplifies computation and communications. Besides, it is resource-efficient since only the nodes that participate in target localization and tracking are kept active and the rest are left inactive in low energy mode. Cluster-based localization and tracking systems should deal with three main tasks: measurement integration, node inclusion/exclusion in the cluster and selection of the cluster head.

This book summarized the current state of the art in cluster-based localization and tracking in ubiquitous computing systems. It presented the main architectures, techniques and technologies. This book also summarized the main existing techniques that deal with these three issues: measurement integration, node inclusion/exclusion and cluster head selection.

This book also presented a general architecture for cluster-based tracking. The architecture comprises modules that deal specifically with the three aforementioned issues. The book also briefly presented different distributed information-driven techniques suitable for each of these issues. All these techniques are fully distributed and require the active cooperation of all the nodes within the cluster. They use distribution-friendly tools and metrics, are efficient in the consumption of energy and computational resources and can be executed in almost constant time regardless of the cluster size.

J.R. Martínez-de Dios et al., *Cluster-based Localization and Tracking in Ubiquitous Computing Systems*, SpringerBriefs in Cooperating Objects,
DOI 10.1007/978-3-662-54761-8_6

The techniques and schemes described in this book should be interpreted as a step in a longer-term research effort. The developed techniques and schemes and their operation in real applications open new questions and new research lines that still have not been sufficiently covered.

Cluster-based schemes naturally fit multi-target tracking since each target has its own cluster. However, the presence of several targets requires methods and schemes that specifically address target cross-tracking, cluster collisions and cluster merging, among others. Although some techniques have been proposed, they are too specifically suited to specific problems. The development of flexible, general and efficient techniques to solve these issues opens wide fields for research.

Multi-target multi-sensor localization and tracking requires techniques to solve the association of measurements. Association of RSSI and camera measurements can be addressed for instance using face recognition techniques to identify people or tagging the targets with visual markers that can be detected using computer vision. Also, some measurement association techniques based on voting and establishing local associations using different distance metrics have been developed. However, these solutions are usually very suited to the particular problem and the general data association problem is still far to be solved.

A real-world application has to be reliable and secure. Most of the developed techniques do not consider techniques or protocols to ensure packet delivery or to encrypt the data exchanged. For instance, the loss of packets that notify the decisions taken by the node inclusion/exclusion or cluster head selection methods can involve severe degradation in tracking performance. Lightweight protocols that ensure and confirm the reception of these packets should be implemented. On the other hand, the deployment of sensor nodes in an unattended environment makes the networks vulnerable to a high variety of potential threats. It is of high relevance to protect the network and ensure data integrity and confidentiality. Of course, the integration of such techniques can affect the performance of the system, increasing the computational burden and the number of interchanged packets. Therefore, a suitable trade-off between security and performance must be sought.