FEATURE-LEVEL INFORMATION FUSION METHODS FOR URBAN SURVEILLANCE USING HETEROGENEOUS SENSOR NETWORKS

By

Manish Kushwaha

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

May, 2010

Nashville, Tennessee

Approved:

Professor Xenofon Koutsoukos

Professor Gautam Biswas

Professor Akos Ledeczi

Professor Yuan Xue

Professor Mitch Wilkes

To,

my mother, *Savitri Devi*

my father, *Saligram Kushwaha*

my sister *Sonal*, my brother *Dhruv*

and

to my lovely fiancée, *Aruna*.

PREFACE

Wireless Sensor Networks (WSNs), with low-power wireless devices integrated with sensors and actuators, are emerging as a new computing paradigm that promise to seamlessly integrate cyber- and physical-worlds. WSNs have great potential for many existing and novel application areas such as environmental monitoring, industrial and manufacturing automation, health-care, and military.

Heterogeneous Sensor Networks (HSNs), with heterogeneity in terms of their computation resources, wireless link properties, power capacities or sensing modalities, are the natural step in the evolution of WSNs driven by several factors, such as multiple application support, incorporation of legacy hardware, hierarchical deployment/architecture, and monitoring of multimodal phenomena. HSNs have been increasingly used in surveillance applications such as monitoring and tracking. Such applications require an information fusion framework that incorporates the data from multiple sensors. Classical target tracking approaches, such as probabilistic data association filtering and multiple hypothesis tracking, perform decision-level information fusion, wherein local decisions are made on the sensors which are then fused at a centralized location for global decision and tracking. Such approaches suffer from poor discrimination and exponential complexity, especially for multiple targets. Target tracking approaches based on signal-level information fusion, wherein the entire raw data from sensors are utilized for tracking, are not feasible in WSNs due to limited communication bandwidth.

The goal of this dissertation is to develop feature-level information fusion methods for target tracking in HSNs. Feature-level information fusion, wherein several features that are extracted from the raw data, should be used for tracking due to their lower communication bandwidth requirement, while maintaining good target discrimination capability. We design and implement a multimodal multisensor information fusion system for target tracking in an urban environment using an HSN of audio and video sensors. We demonstrate the system operating online in real-time. Further, we extend the audio sensing component of the multimodal system by including multiple acoustic features. We develop and implement a feature-based approach to collaborative source localization of multiple acoustic sources in WSNs. We also extend the video sensing component of the system to include multiple video features. We develop and implement an approach for collaborative target tracking in 3D space using a wireless network of smart cameras.

ACKNOWLEDGMENTS

In the completion of my research and this dissertation, I owe thanks to my advisor, Professor Xenofon Koutsoukos, whose comments and critiques helped improve the quality and presentation of the work. I would also like to thank the remaining members of the committee, Professors Gautam Biswas, Akos Ledeczi, Yuan Xue, and Mitch Wilkes, for their comments on the research presented in this dissertation. Also, I would like to thank members of our research group, Janos Sallai, Brano Kusy, Peter Volgyesi, Gyorgy Balogh, Miklos Maroti, Andras Nadas, and Karoly Molnar, who my interactions with have also improved the quality of this work. I would like to especially thank Isaac Amundson, for his enourmous help throughout this work. Finally, I would like to thank my parents, who always believed in me and encouraged me; thanks to my sister, Sonal, for the faith she has in me; thanks to my brother, Dhruv, who thinks of me as the smartest person; and thanks to my lovely fiancée, Aruna, who keeps me grounded.

Partial support for this work has been provided by the Army Research Office (ARO) Multi-disciplinary Research Initiative (MURI) program under the title "Heterogeneous Sensor Webs for Automated Target Recognition and Tracking in Urban Terrain" (W911NF-06-1-0076).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Motivation

Recent technological advances in sensing, communication and computing have enabled the emergence of Wireless Sensor Networks (WSNs) as a new computing paradigm. A WSN is a wireless network consisting of spatially distributed autonomous low-power computing devices equipped with various sensors, a processor, memory, a power supply, and a radio. WSNs have great potential for many existing and novel application areas such as environmental monitoring, industrial and manufacturing automation, health-care, and military [1–3].

WSNs with device heterogeneity in terms of their computation resources, wireless link properties, power capacities or sensing modalities are called Heterogeneous Sensor Networks (HSNs). HSNs are the natural step in the evolution of WSNs driven by several factors, such as multiple application support, incorporation of legacy hardware, hierarchical deployment/architecture, and monitoring of multimodal phenomena. Heterogeneity in WSNs is both a design feature, e.g., multimodal tracking, as well as a necessity, e.g., legacy hardware support [4].

HSNs can be used for applications such as monitoring and tracking [5, 6]. Tracking applications have a long history. In WWII, radar and sonar measurements were used to track aircraft and submarines, respectively. Recently video-based surveillance systems have proliferated due to miniaturization of video sensors and lowering costs [7]. Classical target tracking approaches, such as probabilistic data association filtering and multiple hypothesis tracking, are *decision-level fusion* approaches, wherein local decisions are made on the sensors. These local decisions from all sensors are communicated to the fusion node that combines them for global decision and target tracking. Such approaches suffer from poor discrimination and exponential complexity, especially for multiple targets. An alternate approach is *signal-level fusion*, wherein the entire raw data from sensors are utilized for tracking. Such approaches are not feasible in WSNs due to limited communication bandwidth [8]. An alternative approach is *feature-level fusion*, wherein several features are extracted from the raw data that are used for tracking. Feature-level fusion approaches for target tracking can be formulated using the powerful framework of graphical models. **The goal of dissertation is to develop *feature-level information fusion* methods for target tracking in HSNs.**

Research Challenges

Target tracking refers to estimating the trajectory and other desired properties, such as shape, color, acoustic frequency of a target as it moves around in a sensing field. This is achieved by deploying various sensors, such as audio, video, etc., gathering data at the sensors and analyzing the collected data. The major challenges to tracking using HSNs are elaborated below.

1. **Realistic Sensing Models.** Depending on the modality, sensors have capabilities and handicaps. In general, sensor models are based on assumptions for the sensing environment, e.g. propagation speed of sound, energy decay factor, etc. In addition, sensors assume a measurement noise distribution, e.g. zero-mean additive white Gaussian noise in acoustic signal intensity. In reality, such assumptions fail in less than ideal situations. Also, the measurement noise distribution is affected by a number of factors, e.g. temperature, humidity, which are not very well understood.

2. **Realistic Target Models.** Like sensor models, a tracking system is designed with certain target assumptions, e.g. omnidirectional source, ellipsoid shape, monochromatic object, etc. In reality, such assumptions are either limiting or invalid. Other challenges for realistic target models are complex target motion, target interaction, and a variable number of targets entering and leaving the sensing environment. Complex models are available to handle such challenges but model complexity is a trade off with real-time performance.

3. **Real-time Processing Algorithms.** Target tracking is useful and effective only if the system is able to track the targets at a desirable rate. The desirable rate depends on the tracking resolution and target speed. There is a clear trade off between sensor and target model complexity, and real-time performance. Balancing such trade offs is a research challenge.

4. **Communication Bandwidth.** Along with computation constraints that tighten the real-time processing requirements, limited communication bandwidth in HSNs poses a major challenge. As sensor nodes get even smaller, and sensor networks grow larger in size, limited bandwidth will cause packet delay, packet loss, and network congestion.

5. **Urban Environment.** There exist a number of additional challenges in sensor networks deployed in to urban environments. These challenges are often specific to the sensing modality. For example, the challenges for video sensing include gradual and sudden illumination changes, vacillating backgrounds, shadows, visual clutter, reflection from windows, and occlusion. For audio sensing, they include strong background noise, wind gusts, multipath effects, and reverberation.

Contributions

The contributions of this dissertation include approaches for using the capabilities of HSNs to minimize or mitigate the handicap of sensor and target models, due to violation of assumptions. This is achieved by redundant, complementary and cooperative information fusion. A HSN can have a number of sensors of different modalities spatially distributed over the sensing environment. The advantage of HSNs is that when one sensor, or a group of sensors are handicapped due to violation of some sensor or target assumption, other sensors at different locations and of modalities, might be able to carry out tracking.

Along with handicap mitigation, we attempt to address the real-time processing requirements and limited communication bandwidth. We use simple target models and simple features for fast processing. We also focus on concise features for efficient communication. In addition, this dissertation catalogs extensive related work in the area of general target tracking approaches, and specific approaches for target localization and tracking using audio and video sensors. The specific contributions made in this dissertation are listed below:

1. We design and implement a multimodal multisensor information fusion system for target tracking in an urban environment using an HSN [9–13]. The HSN consists of mote class devices equipped with microphone arrays as audio sensors and embedded PCs equipped with web cameras as video sensors. The system operates online in real-time at 4Hz, thus addressing the real-time processing requirement. The audio and video sensors compute local features and communicate them with the fusion node, thus addressing the limited communication bandwidth. We present tracking results gathered in an uncontrolled urban environment and provide a thorough evaluation including a comparison of different fusion and tracking approaches.

2. We develop and implement a feature-based approach to collaborative source localization of multiple acoustic sources in WSNs [14–16]. Acoustic beamform and power spectral density (PSD) extracted from sensor nodes equipped with microphone array are used as acoustic features. We use a graphical modeling framework to formulate the problem, and employ Maximum Likelihood (ML) and Bayesian estimation for multiple source localization and discrimination. We present simulation results for multiple source localization in a grid sensor network for three different simulation scenarios to study the effect of (1) source density, (3) source SNR, and (3) distance between sources. We also present evaluation of the localization accuracy when the assumptions for the acoustic sources are relaxed. Finally, we implement the proposed feature

extraction algorithms on an FPGA chip onboard micaz sensor nodes, and conduct outdoor experiments with real acoustic sources. Outdoor experimental results reinforce the simulation results.

3. We develop an approach for collaborative target tracking in 3D space using a wireless network of smart cameras. We model the targets in 3D space thus circumventing the problems inherent in the tracker based on 2D target models. We use color and texture features to model the target. We propose a number of probabilistic 3D trackers and implement them using sequential Monte Carlo algorithms. The variations include optimizations such as the use of mixture models, in-network aggregation, and the use of image-plane based filtering where it is appropriate. We present qualitative comparison of the trackers according to their supported Quality-of-Service (QoS) and Quality-of-Information (QoI). Finally, we evaluate the trackers using synthetic targets in simulated camera networks, as well as using real targets (objects and people) in real-world camera network deployments. We also compare the proposed trackers with an implementation of previous state-of-the-art approach for 3D tracking. The simulation results show robustness against target scale variation and rotation, while working within the bandwidth constraints.

CHAPTER II

RELATED WORK

Target localization and tracking, in WSNs, usually consists of four parts, (1) modeling, (2) sensor model and feature extraction, (3) sensor data fusion, and (4) tracking. The problem is more challenging for WSNs due to computational constraints, limited communication bandwidth, and spatially distributed sensors. However, WSNs have the advantage that if one sensor or a group of sensors are handicapped, other sensors can be used to provide necessary measurements.

In this chapter, we start by reviewing WSNs and HSNs, their applications, challenges and sources of heterogeneity. In Section II, we review different target tracking approaches. In Section II, we give an overview of information fusion, its classification, methods and architectures. In Section II and II, we review different types of target localization and tracking approaches using audio and video modalities, respectively. We also compare the reviewed approaches in WSN context. Finally in Section II, we review approaches for multimodal target tracking and compare them to the work in this thesis.

Wireless Sensor Networks

WSNs are emerging as an important computing class based on a new platform and networking structure. A WSN is a wireless network consisting of spatially distributed autonomous sensor nodes, which are low-power computing devices equipped with various sensors, a processor, memory, a power supply, and a radio [1–4]. WSNs may consist of many different types of sensors such as seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic, radar, biological, and chemical. These sensors are able to cooperatively monitor a physical phenomenon or environmental conditions, such as temperature, sound, vibration, pressure, humidity, vehicular movement, lightning condition, soil makeup, pollutants, the presence or absence of certain kinds of objects, mechanical stress levels on attached objects, and the current characteristics such as speed, direction, and size of an object. The sensor nodes in a typical WSN are small, with limited processing and computing resources, and they are inexpensive compared to traditional sensors [7, 8].

Applications

WSNs have great potential for many applications in scenarios such as military target tracking and surveillance [4, 17], natural disaster relief [18], biomedical health monitoring [19, 20], and hazardous environment exploration and seismic sensing [21]. In military target tracking and surveillance, a WSN can assist in intrusion detection and identification. With natural disasters, sensor nodes can sense and detect the environment to forecast disasters before they occur. In biomedical applications, surgical implants of sensors can help monitor a patients health. For seismic sensing, ad hoc deployment of sensors along the volcanic area can detect the development of earthquakes and eruptions. WSNs can be used for applications such as monitoring and tracking. Monitoring applications include indoor/outdoor environmental monitoring, health and wellness monitoring, power monitoring, inventory location monitoring, factory and process automation, and seismic and structural monitoring. Tracking applications include tracking objects, animals, humans, and vehicles. Table 1 lists a number of potential WSN applications.

Table 1: Wireless sensor network applications

Military applications	Monitoring friendly forces, equipment and ammunition; battlefield surveillance; reconnaissance of opposing forces and terrain; targeting; battle damage assessment; and nuclear, biological and chemical (NBC) attack detection and reconnaissance.
Environmental applications	Tracking the movements of birds, small animals, and insects; monitoring environmental conditions that affect crops and livestock; irrigation; chemical/biological detection; precision agriculture; forest fire detection; meteorological or geophysical research; flood detection; bio-complexity mapping of the environment; and pollution study.
Health applications	Providing interfaces for the disabled; integrated patient monitoring; diagnostics; drug administration in hospitals; telemonitoring of human physiological data; and tracking and monitoring doctors and patients inside a hospital.
Home applications	Home automation, smart environments
Commercial applications	Monitoring material fatigue; managing inventory; monitoring product quality; constructing smart office spaces; environmental control in office buildings; robot control and guidance in automatic manufacturing environments; monitoring disaster area; smart structures with sensor nodes embedded inside; vehicle tracking and detection.

Wireless Sensor Network Challenges

The suitability of WSNs for monitoring and tracking applications is due to the unique characteristics and fundamentally different capabilities of WSNs as compared to the traditional distributed sensing systems. Consequently, the design of WSN systems has become more challenging than that of the

traditional distributed systems. The development of sensor networks requires technologies from three different research areas: sensing, communication, and computing (including hardware, software, and algorithms). In general, WSNs pose considerable technical problems in data processing, communication, and sensor management. Because of potentially harsh, uncertain, and dynamic environments, along with energy and bandwidth constraints, wireless ad hoc networks pose additional technical challenges in network discovery, network control and routing, collaborative information processing, querying, and tasking. In this section, we review the most pressing challenges to WSNs. Table 2 lists the WSN design assumptions and resulting constraints.

Table 2: Assumptions and constraints in WSNs

Sensor node characteristics	**Constraints/Challenges**
Low-cost components	Limited computation Limited communication bandwidth
Small/lightweight	Small battery Small lifespan Limited range

Table 3 lists the WSN design characteristics and resulting challenges. A short description of each of these challenges is presented below.

Table 3: Characteristics and challenges in WSNs

WSN characteristics	**WSN Challenges**
Ad hoc deployment	Localization, Network discovery, Coverage and Connectivity
Large number of sensors	Scalability Compression and aggregation
Recording of physical event on distributed sensors	Time-synchronization Collaborative signal and information processing
Wide-area deployment w/ limited range	Multihop routing
Harsh/hostile environment	Fault-tolerance Security
Wireless medium	Communication protocols
Low-cost, low-power components	Power management, Coverage and Connectivity, Programming abstractions

Localization and Network Discovery. In WSNs, sensor nodes that are deployed into the environment in an ad hoc manner do not have prior knowledge of their location. The problem of determining the node location is referred to as localization. Existing localization methods include global positioning system (GPS), beacon (or anchor) nodes, and proximity-based localization [22,23].

In addition to self-localization, knowledge of the network is essential for a sensor in the network to operate properly. Each node needs to know the identity and location of its neighbors to support processing and collaboration. In planned networks, the topology of the network is usually known a priori. For ad hoc networks, the network topology has to be discovered in real time, and updated periodically as sensors fail or new sensors are deployed. In the case of a mobile network, since the topology is always evolving, mechanisms should be provided for the different fixed and mobile sensors to discover each other.

Time-Synchronization. In WSN applications, the reasoning about the ordering of the events, the causal relationships and correlations between the events, the rate of change of observations over time, and the ability to coordinate future actions, are critical and important requirements [24, 25]. The sensed data is of limited usage if it is not accompanied by the coordinates of the sensor - position and time stamp. This is perhaps the primary reason for time synchronization in wireless sensor networks. In addition, various time division multiple access (TDMA) schemes proposed in literature for ad hoc networks assume time synchronization of the nodes. The software component responsible for maintaining a common notion of time over possibly multi-hop links is called the time synchronization service.

Routing and Communication Protocol. The development of a reliable and energy-efficient protocol stack is important for supporting various WSN applications [26]. Depending on the application, a network may consist of hundreds to thousands of nodes. Each sensor node uses the protocol stack to communicate with one another and to the sink. Hence, the protocol stack must be energy efficient in terms of communication and be able to work efficiently across multiple sensor nodes.

Fault Tolerance. WSNs introduce new challenges for fault-tolerance, since they are inherently fault-prone due to the shared wireless communication medium: message losses and corruptions (due to fading, collision, and hidden-node effect) are the norm rather than the exception. Moreover, node failures (due to crash and energy exhaustion) are commonplace. Since on-site maintenance is not feasible, sensor network applications should be self-healing. Another challenge for fault-tolerance is the energy-constraint of the sensor nodes. Applications that impose an excessive communication burden on nodes are not acceptable since they drain the battery power quickly. Thus, self-healing of sensor network applications should be local and communication-efficient [27].

Collaborative Signal and Information Processing. The nodes in an ad hoc sensor network collaborate to collect and process data to generate useful information. Collaborative signal and information processing over a network is a new area of research and is related to distributed information fusion [28,29]. Important technical issues include the degree of information sharing between nodes and how nodes fuse the information from other nodes. Processing data from more sensors generally results in better performance but also requires more communication resources (and, thus, energy). Similarly, less information is lost when communicating information at a lower level (e.g., raw signals), but requires more bandwidth. Therefore, one need to consider the multiple trade offs between performance and resource utilization in collaborative signal and information processing using microsensors.

Security and Privacy. Since the WSNs are deployed in a potentially hostile environment, it is vulnerable to threats and risks. An adversary can compromise a sensor node, alter the integrity of the data, eavesdrop on messages, inject fake messages, and waste network resource. Unlike wired networks, wireless nodes broadcast their messages to the medium. Hence, the issue of security must be addressed in WSNs [30].

Power Management. WSNs have been proposed to be deployed in inaccessible or hazardous regions meaning frequent maintenance such as battery replacement is undesirable and in some cases impossible. Intelligent power management for these devices is critical in maximizing the networks life span and success of the WSN application. This longevity must, however, be achieved while maintaining the integrity of the sensory data harvested by the network. Various power-aware WSN protocols and algorithms have been proposed for power management. A new subclass of WSN called Energy Harvesting (EH) WSN have been proposed that harvest environmental energy to extend system lifetime [31].

Programming Abstractions. A key to the growth of WSN is raising the level of abstraction for programmers. Currently, programmers deal with too many low levels details regarding sensing and node to node communication [32,33]. For example, they typically deal with sensing data, fusing data and moving data. They deal with particular node to node communication and details. If we raise the level of abstraction to consider aggregate behavior, application functionality and direct support for scaling issues then productivity increases.

Coverage and Connectivity. A challenge related to power management is of coverage and connectivity (CC). The fundamental problem in CC is to minimize the number of nodes that remain active, while still achieving acceptable quality of service for applications. In particular, maintaining sufficient *sensing coverage* and *network connectivity* with the active nodes are critical requirements in sensor networks. Also, the network must be able to configure itself to any feasible degrees of coverage and connectivity in order to support different applications and environments with diverse requirements [34].

Compression and Aggregation. Both data compression and aggregation reduce communication cost and increase reliability of data transfer. Data compression and aggregation are necessary for WSN applications which have large amount of data to send across the network [35]. In data compression, it is important that no information is lost and individual data readings are retained. For data aggregation, data is collected from multiple sensors and combined together to transmit to the base station. In this case, aggregated data is more important than individual readings. This method is often used in a cluster-based approach.

Heterogeneous Sensor Networks

Heterogeneous sensor networks (HSNs) are the natural step in the evolution of WSNs driven by several factors, such as multiple application support, incorporation of legacy hardware, hierarchical deployment/architecture, and monitoring of multimodal phenomenon [4, 12]. With its increasing popularity and ubiquity/pervasiveness, WSNs will be required to support multiple, although not necessarily concurrent, applications. Different applications may require different resources, and may make use of nodes with different capabilities. As the technology matures, new types of nodes will become available and existing deployments will be refreshed. Diverse nodes will need to coexist and support old and new applications. In multihop WSNs, it may be advantageous to organize the sensors into clusters of low-power sensors with a high-power clusterhead sensor. In such hierarchical architecture, the low-power sensors communicate with the clusterheads, which communicate with the data processing center. In addition, the clusterheads can also act as intermediate data processing and aggregation centers. Furthermore, as WSNs are deployed for applications that observe more complex phenomena, multiple sensing modalities become necessary. Different sensors usually have different resource requirements in terms of processing power, memory capacity, or communication bandwidth. Instead of using a network of homogeneous devices supporting resource intensive sensors, an HSN can have different nodes for different sensing tasks [36–38]. For example, at one end of the

spectrum low data-rate sensors measuring slowly changing physical parameters such as temperature, humidity or light, require minimal resources; while on the other end even low resolution video sensors require orders of magnitude more resources.

The heterogeneity in HSNs can be categorized in following four categories, (1) computational heterogeneity, (2) network heterogeneity, (3) energy heterogeneity, and (4) sensing heterogeneity. Table 4 lists the type of heterogeneity in HSNs.

Table 4: Type of heterogeneity in HSNs

Types of heterogeneity	**Examples**
Computational heterogeneity	*High-power:* Intel's stargate *Low-power:* Mica2, Micaz, etc.
Network heterogeneity	*Long-range reliable links:* 802.11, WiMAX, etc. *Short-range links:* 802.15, Bluetooth, etc.
Energy heterogeneity	*High/unlimited energy:* plugged to wall outlet *Rechargeable:* energy harvesting sensors *Limited energy:* AAA batteries
Sensing heterogeneity	Audio, Video, Seismic, etc.

Target Localization and Tracking

Target tracking is an important representative application in HSNs. The classical application of target tracking is to manned surveillance systems such as air traffic control, maritime surveillance and airborne early warning system. With the emergence and potential ubiquity of WSNs, automated surveillance systems are gaining popularity. In this section, we review some approaches to target localization and tracking. Figure 1 shows the classification of various target tracking approaches.

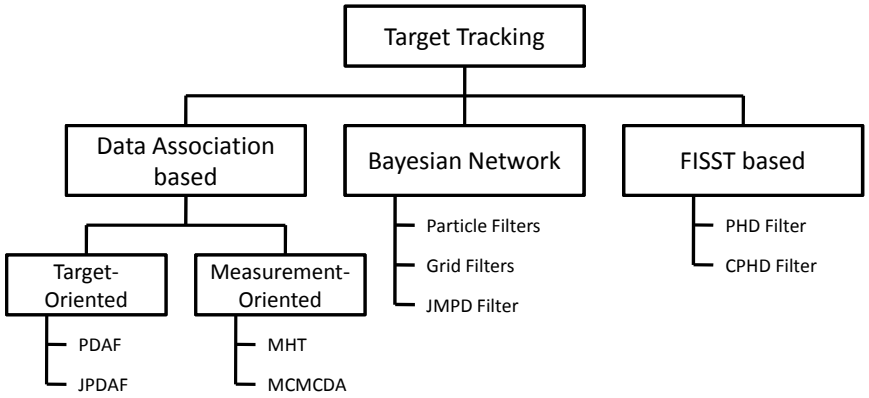


Figure 1: Classification of target tracking approaches.

Data Association-Based Approaches

The classical approaches to multiple target tracking include data association-based approaches such as Multiple Hypothesis Tracking (MHT) [39] and data association filters [5, 6]. These approaches propose a set of exclusive and exhaustive hypotheses either associating measurements with the targets and clutter, called *target-oriented methods*, or associating targets with measurements, called *measurement-oriented methods*. Probabilities are computed for each hypothesis and the most probable (or a set of) hypotheses are used to compute target estimates. The number of hypotheses is combinatorial in number of targets and observations, as well as in time.

In these approaches, the *measurements* are noise-corrupted observations related to the state of a target, such as direct estimate of position, range and/or azimuth from a sensor, time-difference-of-arrival, signal strength, etc. The measurements of interest in multitarget applications are usually not raw data points, but rather the outputs of signal processing and detection subsystems [6]. A *track* is a state trajectory estimated from a set of measurements that have been associated with the same target. The crux of the multitarget tracking problem is to carry out this *data association* for measurements whose origin is uncertain due to, (1) false alarm in detection process, (2) clutter, (3) interfering targets, and (4) decoys or other countermeasures. The *data association model* in these approaches either deterministically or probabilistically associate measurements to targets. The data association results are then used in a standard state estimation algorithm, e.g. Kalman filter.

The problem of multiple target tracking using data association based approaches can be stated as follows. Let $T \in \mathbb{Z}^+$ be the duration of surveillance. Let K be the unknown number of objects moving around the surveillance region. Let $F^k : \mathbb{R}^d \to \mathbb{R}^d$ be the the discrete-time dynamics of the object k, where d is the dimension of the state variable, and let $x_t^k \in \mathbb{R}^d$ be the state of the object k at time t for $k = 1, 2, \cdots, K$. The object k moves according to

$$x_t^k = F^k(x_{t-1}^k) + w_t^k$$

where $w_t^k \in \mathbb{R}^d$ are white noise processes. The noisy observation of the state of the object is measured with a detection probability less than unity. There are also false alarms with a nonzero false alarm rate. Let $y_t^j \in \mathbb{R}^m$ be the jth observation at time t for $j = 1, \cdots, n_t$ where m is the dimensionality of each observation vector and n_t is the number of observations at time t. Each object generates a unique observation at each sampling time if it is detected. Let $H^j : \mathbb{R}^d \to \mathbb{R}^m$ be the observation

model. Then the observations are generated as follows:

$$y_t^j = \begin{cases} H^j(x_t^k) + v_t^j & \text{if } j\text{th observation is originated from } x_t^k \\ u_t & \text{otherwise} \end{cases}$$

where $v_t^j \in \mathbb{R}^m$ are white noise processes and $u_t \sim \text{Unif}(\mathcal{R})$ is a random process for false alarms. Notice that, with a nonzero probability, the object is not detected, which is called a missing observation. The goal is to, either deterministically or probabilistically, associate measurements to targets, and used the associations in a standard state estimation algorithm, e.g. Kalman filter, to estimate target trajectories.

Probabilistic Data Association Filter

Kalman filter [40] is an efficient method for tracking single targets when the distribution on measurements is Gaussian, and the dynamic model and measurement model are linear. Variants of Kalman filter such as Extended Kalman filter (EKF) and Unscented Kalman filter (UKF) exist for cases where the linear model or Gaussian noise assumption does not hold. The basic KF and its variants, however, are not able to track single target in clutter, or multiple targets.

The Probabilistic Data Association Filter (PDAF) [6] is an extension of the Kalman filter to track single target in presence of multiple measurements due to clutter. A key notion in the PDAF is that of an association event Θ or conjunction of association events θ_i (denoting association of ith measurement and the target). At each time step, PDAF generate association events, associating measurements with the target. The probability of a particular Θ depends on the distances between target's predicted measurement and the actual measurement it is associated with in Θ.

Like the computation of the innovation in Kalman filter, PDAF introduces a notion of the combined innovation, computed over the n_t measurements detected at a given time step as the weighted sum of the individual innovations: $\nu = \sum_{i=1}^{n} \beta_i \nu_i$. Each β_i is the probability of the *association event* θ_i that the ith measurement is target-originated. Also computed is β_0, the probability of the event that none of the measurements is target originated (i.e., the target is associated with the null measurement). These events encompass all possible interpretations of the data, so $\sum_{i=1}^{n} \beta_i = 1$.

Joint Probabilistic Data Association Filter

The Joint Probabilistic Data Association Filter (JPDAF) [6] is an extension of PDAF to track a fixed known number of targets in presence of multiple measurements. JPDAF computes the proba-

bilities of measurement-to-target associations jointly across all targets. JPDAF enforces an exclusion principle that prevents two or more measurements to be associated with the same target. In other words, joint data association in JPDAF maintains a one-to-one association between measurements and targets.

Similar to PDAF, the key notion in the JPDAF is that of a joint association event Θ or conjunction of association events θ_{j,t_j} (denoting association of jth measurement and t_jth target). The probability of a particular Θ depends on the distances between each target's predicted measurement and the actual measurement it is associated with in Θ. However, an additional influence on the probability of Θ stems from the interaction of the various association events in the joint association event.

Multiple Hypothesis Tracking

The multiple hypothesis tracking algorithm was originally developed in [39]. Unlike JPDAF, which can only track a fixed known number of targets, MHT provides track initiation and termination capabilities. This enables MHT to track a variable number of targets, which is the case when targets enter and leave the scene. An iteration begins with the set of current hypotheses from iteration $(k-1)$. Each hypothesis (leaf) contains a set of active tracks, and becomes a parent hypothesis node in the current iteration. Each hypothesis provides an interpretation of all past measurements consisting of a collection of disjoint tracks. Predictions are made as to the expected location of measurements and these predictions are matched to actual measurements using the Mahalanobis distance. Each measurement may either (1) belong to a previously known target or (2) be the start of a new target (track initiation) or (3) be a false alarm. In addition, for targets that are not assigned measurements, there is the possibility of (4) deletion of the target (track termination). The resulting enumeration of associations produces a set of children (events) for each parent node, extending the depth of the tree by another level. Associated with each new leaf is a probability. In the final step of the iteration, the tree is pruned to remove unlikely correspondences.

Markov Chain Monte Carlo Data Association

The JPDAF is not able to handle a varying number of targets, which is the case when new targets can come and old targets can leave the the scene at any time. A *single-scan* approach, which updates the posterior based only on the current scan of measurements, can be used to track an unknown number of targets with the help of trans-dimensional MCMC [41, 42] or a detection algorithm [43]. But a

single-scan approach cannot maintain tracks over long periods because it cannot revisit previous, possibly incorrect, association decisions in the light of new evidence. This issue can be addressed by using a *multi-scan* approach, which updates the posterior based on both current and past scans of measurements. The MHT is a multi-scan tracker, however, it is not widely used due to its high computational complexity.

A newly developed algorithm, called Markov chain Monte Carlo data association (MCMCDA), provides a computationally desirable alternative to MHT [44]. The MCMCDA is a data-oriented, combinatorial optimization approach to the data association problem but avoid the enumeration of tracks by applying a sampling method called Markov chain Monte Carlo (MCMC). MCMCDA shows remarkable performance compared to the greedy algorithm and MHT under extreme conditions such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates. The simulation study in [44] showed that MCMCDA was computationally efficient compared to MHT with heuristics (i.e., pruning, gating, clustering, N-scan-back logic and k-best hypotheses). We use the online version of MCMCDA algorithm in our audio-video fusion based target tracking system in Chapter III.

Probabilistic Graphical Model (Bayesian Network) Approaches

In data association based approaches, the sensor model assumes that each measurement (or detection) corresponds to a single target (i.e. each measurement originated either due to a single target or due to clutter). Also, each target gives rise to a single measurement. The measurement due to one target is *not* affected by other targets. In other words, measurement for target A would be same regardless the presence of other targets in the scene. Due to these assumptions, the approach is not able to model target interaction such as partial or full target occlusion. In the worst case, if target A completely occludes target B, the *lack of measurement due to target B* is considered a missed detection. In reality, the measurement due to target B is *contained* in the measurement due to target A.

Due to this model, wherein a measurement is not the raw data but rather the output of signal processing and detection subsystem, data association based tracking can be categorized as high-level (or decision-level) fusion. In high-level fusion, *discriminating* information is lost, especially for multiple target tracking case, where raw data is a mixture of the raw signals originating from multiple targets. Alternate approaches are low-level (or signal-level) fusion and medium-level (or feature-level) fusion. Low-level fusion is not good for WSNs due to low communication bandwidth.

Medium-level fusion includes signal processing and feature extraction algorithms that *compress* raw data into features that are communicated to the sensor fusion node. In this case, we need explicit models for target interaction and process/observation models that capture the generation of observation (and features) on multiple target state. In other words, we need models for how the raw signals originating from multiple targets are mixed together as raw data.

Recent approaches to multiple target tracking include Bayesian estimation where the quantity of interest is a Markov process, the multitarget state, which is the concatenation of several individual target states [45]. The observations in this model are described in terms of the joint multitarget state, hence taking into account the target interactions. The Bayesian approach has the advantages of providing a recursive solution with arbitrary target dynamic models.

The problem of multiple target tracking using Bayesian estimation can be stated as follows. To define the problem of tracking, consider the evolution of the state sequence $\{\mathbf{x}_t, t = 1, \cdots, T\}$ of a target given by

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{v}_t) \tag{1}$$

where $f_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$ is a possibly nonlinear function of the state $\mathbf{x}_{t-1}$, $\mathbf{v}_t$ is process noise, and n_x, n_v are dimensions of the state and process noise vectors, respectively. The objective of tracking is to recursively estimate $\mathbf{x}_t$ from measurements

$$\mathbf{z}_t = h_t(\mathbf{x}_t, \mathbf{n}_t) \tag{2}$$

where $h_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_n} \to \mathbb{R}^{n_z}$ is a possibly nonlinear function, $\mathbf{n}_t$ is measurement noise, and n_z, n_n are dimensions of the measurement and measurement noise vectors, respectively. In particular, we seek filtered estimates of $\mathbf{x}_t$ based on the set of all available measurements $\mathbf{z}_{1:t} = \{\mathbf{z}_i, i = 1, \cdots, t\}$, up to time t.

From a Bayesian perspective, the tracking problem is to recursively calculate some degree of belief in the state $\mathbf{x}_t$ at time t, taking different values, given the data $\mathbf{z}_{1:t}$ up to time t. Thus, it is required to construct the posterior distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$. It is assumed that the initial probability distribution $p(\mathbf{x}_0|\mathbf{z}_0) = p(\mathbf{x}_0)$ of the state vector, which is also known as the prior, is available ($\mathbf{z}_0$ being the set of no measurements). Then, in principle, the posterior $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ may be obtained, recursively, in two stages: prediction and update.

Suppose that the posterior distribution $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$ at time $t-1$ is available. The prediction

stage involves using the system model (Equation (1)) to obtain the predictive prior of the state at time t via the Chapman-Kolmogorov equation

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})\ p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})\ d\mathbf{x}_{t-1} \tag{3}$$

Note that in Equation (3), use has been made of the fact that $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$, as Equation (1) describes a Markov process of order one. The probabilistic model of the state evolution $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is defined by the system equation (1) and the known statistics of $\mathbf{v}_{t-1}$. At time step t, a measurement $\mathbf{z}_t$ becomes available, and this may be used to update the prior (update stage) via Bayes' rule

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{z}_t)\ p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \tag{4}$$

where the normalizing constant

$$p(\mathbf{z}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{z}_t|\mathbf{x}_t)\ p(\mathbf{x}_t|\mathbf{z}_{1:t-1})\ d\mathbf{x}_t \tag{5}$$

depends on the likelihood function $p(\mathbf{z}_t|\mathbf{x}_t)$ defined by the measurement model (Equation (2)) and the known statistics of $\mathbf{n}_t$. In the update stage Equation (5), the measurement $\mathbf{z}_t$ is used to modify the prior density to obtain the required posterior density of the current state. The recurrence relations Equation (4) and (5) form the basis for the optimal Bayesian solution. This recursive propagation of the posterior density is only a conceptual solution in that in general, it cannot be determined analytically. Solutions do exist in a restrictive set of cases, including the Kalman filter and grid-based filters. When the analytic solution is intractable, extended Kalman filters, approximate grid-based filters, and particle filters approximate the optimal Bayesian solution.

A comprehensive review of optimal and suboptimal Bayesian algorithms for nonlinear/non-Gaussian tracking problems, with a focus on particle filters is presented in [46]. Particle filters are sequential Monte Carlo methods based on point mass (or *particle*) representations of probability densities, which can be applied to any state-space model and which generalize the traditional Kalman filtering methods. Several variants of the particle filter such as SIR, ASIR, and RPF are introduced within a generic framework of the sequential importance sampling (SIS) algorithm. Sequential Monte Carlo (SMC) implementation of the Bayesian estimation and tracking are presented in [46, 47]. Several approaches based on Bayesian estimation [48, 49] and graphical models [50, 51] have been proposed.

An approach for audiovisual object tracking based on graphical models that combine the audio and video data is proposed in [50]. The paper presents graphical models and generative models for the audio-video data, and Bayesian inference algorithm for tracking. An EM algorithm also presented for learning model parameters.

A graphical model formulation for self-localization of sensor networks is presented in [51]. They proposed a technique called nonparametric belief propagation that is a generalization of particle filtering. The NBP approach has the advantage of being amenable to distributed implementation, can include a wide variety of statistical models, and can represent multimodal uncertainty.

A Bayesian approach for tracking the DOA of multiple targets using a passive sensor array is proposed in [48]. The paper includes a constant velocity target dynamics model and a data model for uniform linear sensor array. They construct the data likelihood density and marginalizing the nuisance parameters. Two tracking algorithms are proposed based on particle filtering.

A Bayesian approach for multiple target detection and tracking, and particle filter-based algorithms are proposed in [49]. The particle filter is designed in such a way that the importance density is measurement-directed. The paper constructs and computes the joint multitarget probability density (JMPD) for multitarget state estimation. The joint estimation and joint proposal of multitarget state enables the filter to better handle the situations in which several targets in close proximity. For unknown number of target, the multitarget state is extended to include a random variable corresponding to the number of target.

Finite-Set Statistics Based Tracking

An alternative to Bayesian statistics is Finite Set Statistics (FISST). The Bayesian framework treats the states and observations as realizations of random variables. In FISST framework the multitarget state and multiple observations are treated as finite sets. The first systematic treatment of multisensor multitarget tracking using random set theory is conceived in [52, 53], which later developed into FISST [54]. The FISST Bayes multitarget recursion is generally intractable. An approach to approximate the multitarget Bayes recursion by propagating the Probability Hypothesis Density (PHD) of the posterior multitarget state is proposed in [55]. This strategy is similar to the constant gain Kalman filter that propagates the mean of the posterior single-object state. The PHD recursion still involves multiple integrals with no closed forms in general. Generalization of PDH filters, Cardinalized probability hypothesis density (CPHD) filter provides more accurate estimates

of target number than the PHD filter, and hence, also of the states of targets [56]. Several SMC implementations of the PHD filter are proposed in [57, 58].

Like data association based tracking approaches, the measurements in FISST based tracking are not the raw data, but the outputs of signal processing and detection subsystems. This measurement model makes this approach also a decision-level fusion method.

Summary of Target Tracking Approaches

Table 5 summarizes different target tracking approaches.

Table 5: Summary of target tracking approaches

	KF/EKF	PDAF	JPDAF	MHT	MCMCDA	PHD filter	PF	JMPD
Multiple targets								
fixed, known			✓	✓	✓	✓	✓	✓
variable, unknown				✓	✓	✓		✓
Data association based	✓	✓	✓	✓	✓			
Joint multitarget estimation						✓	✓	✓
Decision-level	✓	✓	✓	✓	✓	✓		
Feature-level							✓	✓

Information Fusion in Wireless Sensor Networks

HSNs utilize multiple physically distributed sensors of different modalities to provide a large amount of data. According to the application objectives, the data might need to be processed, communicated and assessed. A fundamental issue in HSNs is the way the collected data is processed. In this context, information fusion is a discipline that is concerned with how data gathered by sensors can be processed to increase the relevance of such a mass of data. In a nutshell, information fusion can be defined as the combination of data from multiple sensors to obtain improved information (i.e. cheaper, greater quality, or greater relevance).

Terminology

There are many terms related to information fusion that refer to similar concepts. Some of these terms are data fusion, multisensor integration, and data aggregation. Joint Directors of Laboratories (JDL) define *data fusion* as a multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from multiple sources. According to the definition in [59], data fusion is the combination of data from multiple sensors, and related information provided by associated databases, to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone.

Another term used in this context, *multisensor integration* is defined as the synergistic use of information provided by multiple sensory devices to assist in the accomplishment of a task by a system; and multisensor fusion deals with the combination of different sources of sensory information into one representational format during any stage in the integration process in [60]. Multisensor integration is intended to be a broader term than multisensor fusion. It makes explicit how the fused data is used by the whole system to interact with the environment.

Information fusion, according to [28], encompasses the theory, techniques and tools created and applied to exploit the synergy in the information acquired from multiple sources (sensor, databases, information gathered by humans, etc.) in such a way that the resulting decision or action is in some sense better (qualitatively or quantitatively, in terms of accuracy, robustness, etc.) than would be possible if any of these sources were used individually without such synergy exploitation.

The term *Data Aggregation* is widely used in WSN community as a synonym for information fusion. According to [35], data aggregation comprises the collection of raw data from pervasive data sources, the flexible, programmable composition of the raw data into less voluminous refined data, and the timely delivery of the refined data to data consumers.

Challenges

HSNs are intended to be deployed in environments where sensors can be exposed to conditions that might interfere with their measurements. Therefore, sensor measurements may be imprecise and noisy in such scenarios. Additionally, due to harsh deployment environments, sensor failures are not an exception. Due to limited sensing coverage and various power saving services on sensors, both spatial and temporal coverage also pose limitations to information fusion in HSNs. In such situations information garnered from a single sensor or a single modality is unreliable, incomplete or intermittent.

To overcome sensor failures, technological limitations, spatial, and temporal coverage problems, HSNs are deployed with three basic characteristics, namely, cooperation, redundancy, and complementarity [61]. These characteristics result in HSNs composed of a large number of multimodal sensor nodes posing scalability and other classical HSN challenges. From information fusion perspective, these characteristics lead to correct and continued operation of HSNs. In other applications, the use of multiple multimodal sensors increases the robustness and reliability.

Classification

Information fusion can be categorized based on several aspects, such as the relationship between the sensors, level of data abstraction, and data input and output format. Figure 2 shows the categorization of information fusion approaches based on different criteria.

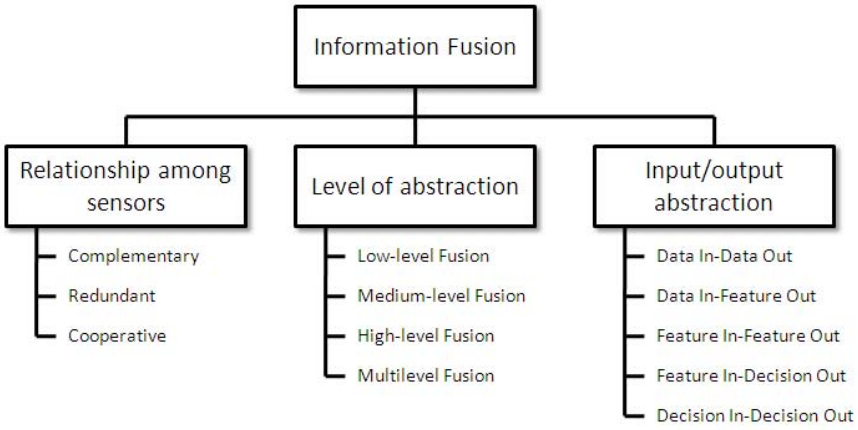


Figure 2: Classification of information fusion.

According to the relationship among the sources, information fusion can be classified as, *complementary*, when the information provided by the sensors is complementary; *redundant*, when the information is redundant; or *cooperative*, when the information from the sensors can be fused into new information that is more complex than the original. Complementary fusion is intended for completeness by compounding information from multiple sensors. Redundant fusion is intended to increase the reliability, accuracy and confidence of the information, and cooperative fusion is intended for extraction/inference of complex information that cannot be gathered directly through sensors.

Information fusion deals with three levels of data abstraction: measurement, feature, and decision. Thus, according to the abstraction level of the data, information fusion can be classified into four categories. *Low-level fusion*, also called *signal-level fusion*, uses raw data (or signals) provided by the sensors as inputs and combines it into new data. The raw data is closest to the physical event, hence low-level fusion incurs smallest amount of information loss. *Medium-level fusion*, also called *feature-level fusion*, uses attributes (or features) extracted from the raw data as inputs and fuses them into new features, or feature map. *High-level fusion*, also called *decision-level fusion*, takes the decisions from each sensor as inputs and fuses them to obtain a global decision. Finally, *multilevel fusion* involves data from different levels of abstraction, e.g. a measurement is fused with a feature to provide a decision. The loss of information increases as data is passed from the lower

level of abstraction to the higher levels. Hence, lower level fusion methods are more accurate than higher level fusion, however, the amount of information required for lower level of abstraction is greater than that for the higher level, which makes the lower level methods expensive in term of computational and communication requirements.

Another classification that considers the abstraction level is provided in [62], in which information fusion methods are categorized based on the abstraction level of the input and output information. The five categories are, Data In - Data Out (DAI-DAO), Data In - Feature Out (DAI-FEO), Feature In - Feature Out (FEI-FEO), Feature In - Decision Out (FEI-DEO), and Decision In - Decision Out (DEI-DEO).

Algorithms

According to the application requirements, information fusion can be performed using different mechanisms such as inference, estimation, classification, feature maps, abstract sensors, aggregation, and compression. Some of these mechanisms are described below.

Inference

Inference, in statistics, is the process of inferring the properties of a random variable, or a random process, using the data generated by the random process. Inference methods are often applied in decision fusion or hypothesis testing. Classical inference methods are based on Bayesian inference and Dempster-Shafer Belief Accumulation theory. Bayesian inference is based on the Bayes' rule, which states that

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

where the posterior probability $P(X|Y)$ represents the belief of hypothesis X given the information Y, $P(X)$ is the prior probability of the hypothesis X, $P(Y|X)$ is the probability of information Y, given that X is true (also called the likelihood of information Y conditioned on hypothesis X), and $P(Y)$ is the information evidence that is independent of hypothesis X and can be treated as a normalizing constant.

Other methods for inference include following.

- Dempster-Shafer Inference. It is based on the Dempster-Shafer Belief Accumulation theory (also referred to as Theory of Evidence or Dempster-Shafer Evidential Reasoning), which is a mathematical theory introduced by Dempster and Shafer [63] that generalizes the Bayesian theory. It deals with beliefs or mass functions just as Bayes' rule does with probabilities.

- Fuzzy Logic. Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with approximate reasoning to draw (possibly imprecise) conclusions from imprecise premises [64]. Each quantitative input is fuzzyfied by a membership function. The fuzzy rules of an inference system produce fuzzy outputs which, in turn, are defuzzyfied by a set of output rules.

- Neural Networks. Artificial neural networks are a set of interconnected programming constructs that mimic the properties of biological neurons. Neural networks are most commonly used to implement supervised/unsupervised learning mechanisms that starting from examples, are able to generalize [65, 66]. Neural Networks represent an alternative to Bayesian and Dempster-Shafer theories, being used by classification and recognition tasks in the information fusion domain.

Estimation

Estimation, in statistics, is the process of estimating the values of parameters and/or unknown data based on measured data. The parameters and the unknown data describe an underlying physical process that determines the distribution of the measured data. Most common estimation methods include maximum likelihood, maximum a posteriori, least squares, moving average filter, Kalman filter, and particle filter.

Maximum Likelihood (ML). Estimation methods based on maximization of likelihood are suitable when the parameter and/or data (called state $\mathbf{x}$) being estimated is not the outcome of a random variable. In the context of information fusion, given $\mathbf{z}_{1:n} = \{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$, a sequence of n observations of $\mathbf{x}$, the ML estimate $\hat{\mathbf{x}}^{\mathrm{ML}}$ is the values of $\mathbf{x}$ that maximizes the likelihood function, given by

$$\hat{\mathbf{x}}^{\mathrm{ML}}(k) = \arg\max_{\mathbf{x}} \mathcal{L}(\mathbf{x}) = \arg\max_{\mathbf{x}} p(\mathbf{z}|\mathbf{x})$$

Maximum A Posteriori (MAP). In Bayesian statistics, a MAP estimate is a mode of the posterior distribution. Unlike ML estimation, it is used when the state $\mathbf{x}$ to be estimated is the outcome of a random variable with known prior density $p(\mathbf{x})$. In the context of information fusion, given $\mathbf{z}_{1:n} = \{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$, a sequence of n observations of $\mathbf{x}$, the MAP estimate $\hat{\mathbf{x}}^{\mathrm{MAP}}$ is the value of $\mathbf{x}$ that maximizes the posterior distribution

$$\hat{\mathbf{x}}^{\mathrm{MAP}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) = \arg\max_{\mathbf{x}} p(\mathbf{z}|\mathbf{x})p(\mathbf{x})$$

Both methods, ML and MAP, try to find the most likely value for the state **x**. However, the former method assumes that state is a fixed, though unknown, point of the state space, while the latter considers the state as the outcome of a random variable with known prior density.

Architectures

The presence of multiple sensors and fusion nodes provides many choices in the architecture, i.e., how the sensors report to fusion node and the connectivity among the nodes. Figure 3 shows four possible architectures: centralized, hierarchical without feedback, hierarchical with feedback, and fully distributed [67].

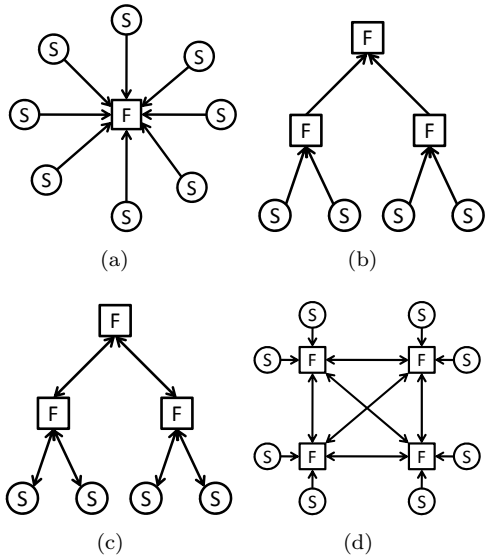


Figure 3: Fusion architectures. **S** and **F** indicate sensor and fusion node, repectively. (a) Centralized, (b) Hierarchical without feedback, (c) Hierarchical with feedback, and (d) Distributed.

The traditional architectures for fusion are centralized. Data from multiple sensors are sent to a single location where the data are fused and the results are distributed to various users. Although, the centralized architecture is theoretically optimal and conceptually simpler, it requires high communication bandwidth to send the sensor data from all the sensors to the fusion node, and more computer resources to process the data.

On the other hand, distributed (including hierarchical) fusion architectures have the following advantages: lighter processing load at each fusion node due to the distribution over multiple nodes; no need to maintain a large centralized database since each node has its own local database; lower communication load since data does not have to be sent to/from a central processing site; faster user access to fusion results since there is less communication delay; and higher survivability since there

is no single point of failure associated with a central fusion node. In a fully distributed architecture, there is no fixed superior/subordinate relationship among nodes. Each node can communicate with other nodes subject to connectivity constraints. Communication can be adaptive and dependent on the information content and needs of the individual nodes.

Biological and Cognitive Foundations

Various psychophysical and neurophysical studies have been conducted to understand the biological and cognitive foundations of intelligent sensor fusion. Such studies have tried to investigate the mechanism of sensor fusion and cognitive model of associated activities, including optimization of sensing configurations, improvement of sensing quality, and filtering of noise [68].

According to the studies, intelligent sensor fusion is defined as a process which can autonomously gather observations from multiple sensors and combine them into a single, coherent percept (execution of the sensor fusion mechanism), allows the sensor fusion mechanism to adapt itself to major environmental changes and sensor malfunctions (perform exception handling), and can determine its own sensing strategies for observing the percept in order to maintain efficient use of shared sensing resources (configuration)

Biological studies and the neurological model describe several aspects of intelligent sensor fusion relevant to robotics. First, sensor fusion couples perception with action. Second, sensor fusion incorporates contextual information. Also, sensors can be combined in different ways for different percepts. Another important aspect is the observation that multisensor neurons can respond more to multiple sources of weak stimuli suggests that sensor evidence accrues rather than is averaged. This is thought to be beneficial to an agent because it can ascertain danger from multiple weak clues (e.g., a camouflaged predator). Cognitive studies reveal following aspect of intelligent sensor fusion for robotics [69], (1) Sensor integration under certain circumstances is a form of closed-loop control, (2) Not all sensors contribute equally, and (3) Control of sensory integration is separate from planning for perception.

Target Localization and Tracking using Audio

Localization of acoustic sources using sensor arrays has been one of the central problems in sonar, navigation, geophysics, and acoustic tracking. Traditional acoustic source localization methods were developed for wired sensor networks. In WSNs, collaborative source localization is needed where the objective is to estimate the positions of multiple sources by fusion of data from multiple sensors.

There are two broad classes of methods for collaborative source localization. The first class of approaches, where the estimation is done by fusion of the sampled signals, is called the signal-level fusion methods. The second class of approaches, where signal features are extracted at each sensor and estimation is done by fusion of the extracted features, is called the feature-level fusion methods. The signal-level fusion methods are not suited for WSNs because they require transmission of the raw signal, which is costly due to limited bandwidth and power on sensors. On the other hand, the feature-level fusion methods are appropriate for WSNs due to its lower bandwidth and power requirements.

The localization approaches depend on the type of the acoustic source. For coherent and narrowband signals, several approaches utilizing the phase differences measured at the receiving sensor have been proposed to estimate the direction-of-arrival (DOA) of the sources in the far-field [70]. A review and comparison of the various narrowband approaches are presented in [71]. For wideband signals, e.g., acoustic or seismic signals, many approaches utilizing the narrowband techniques with suitable transformations have been proposed. A class of methods called coherent signal subspace method (CSM) involves a preprocessing step to transform the wideband signal subspaces into narrowband subspaces and then apply narrowband algorithms [72, 73]. However, the drawback of these methods is the preprocessing that must be performed beforehand.

Another class of wideband source localization algorithms is based on time delays. These techniques typically include two steps, namely, time delay estimation (TDE) between signals received on spatially separated sensors and source localization based on the time delay estimates [74–79]. Other methods for wideband acoustic source localization include beamforming [71, 80], hemisphere sampling [81], and probabilistic accumulation [82].

Among feature-based methods, energy-based localization (EBL) methods utilizing signal energy as acoustic features have been proposed [83–85]. In free space, acoustic energy decays at a rate that is inversely proportional to the distance from the source. Several least-squares formulations for EBL are presented in [83, 85]. A ML formulation with capability for multiple source localization is presented in [84]. Figure 4 shows the categorization of various acoustic source localization approaches.

Signal Propagation Models

Three signal propagation models have been used in the acoustic source localization literature. They are, the ideal single-path propagation model, the multipath model, and the reverberation model.

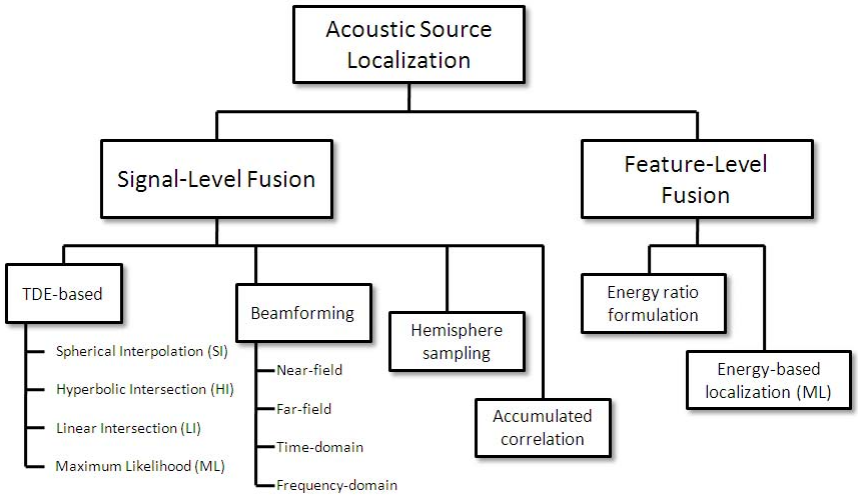


Figure 4: Acoustic source localization approaches.

Ideal Propagation Model

The ideal propagation model assumes that the signal acquired by each sensor is a delayed and attenuated version of the original source signal plus some additive noise. The ideal single-path model can again be categorized into far-field and near-field cases.

The Far-Field Case. When the source is in the far-field of the sensor array, the wave front is assumed to be planar and only the DOA information can be estimated. In this case, one of the sensors is used as the reference point and the signal model is defined based on the relative time delays from this position. The received signal at kth sensor is expressed as

$$z_k[n] = \alpha_k s[n - \tau_{kq}] + w_k[n] \tag{6}$$

where α_k, is the attenuation factor due to propagation effects, $s[n]$ is the unknown source signal, $w_k[n]$ is an additive noise signal at the kth sensor, and τ_{kq} is the relative time delay between sensors k and the reference sensor indexed q. In far-field case, the relative time delay is given by, $\tau_{kq} = t_k - t_q = d_{kq}\cos(\phi - \beta_{kq})/C$, where d_{kq} and β_{kq} are the distance and angle between two sensors, respectively, ϕ is the source DOA, and C is the speed of sound.

The Near-Field Case. In this case, the wave from is assumed to be spherical and the range information can also be estimated in addition to the DOA. The source location is itself considered as the reference point. The received signal at kth sensor is expressed as

$$z_k[n] = \alpha_k s[n - \tau_k] + w_k[n]$$

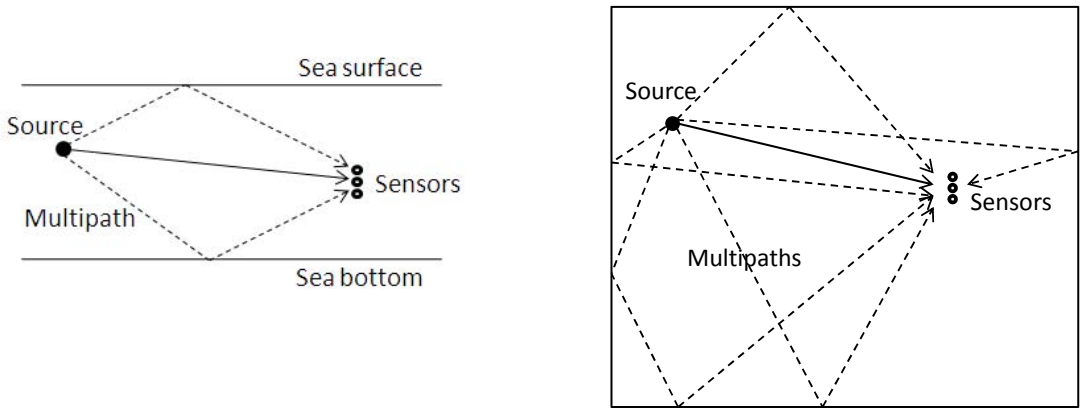
where α_k, is the attenuation factor due to propagation effects, $s[n]$ is the unknown source signal, $w_k[n]$ is an additive noise signal at the kth sensor, and τ_k is the absolute time delay between sensor k and the source. In near-field case, the absolute time delay is given by, $\tau_k = t_k = \| \mathbf{x}_k - \mathbf{x} \| / C$, where $\mathbf{x}_k$ and $\mathbf{x}$ are locations of the kth sensor and the source, respectively, and C is the speed of sound.

Multipath Model

In this model, each sensor receives multiple delayed and attenuated replicas of the source signal due to reflections of the wavefront from boundaries and objects in addition to the direct-path signal (See Figure 5(a)). This multipath effect has been intensively studied in the literature [86, 87]. In this case, the received signals are often described mathematically as

$$z_k[n] = \sum_{m=1}^{M} \alpha_{km} s[n - t - \tau_{km}] + w_k[n] \tag{7}$$

where α_{km} is the attenuation factor from the unknown source to the kth sensor via the mth path, t is the propagation time from the source to a reference sensor via direct path, τ_{km} is the relative delay between sensor k and the reference sensor for path m, M is the number of different paths.



(a) (b)

Figure 5: Illustration of the signal model in a (a) multipath environment, and (b) reverberant environment.

Reverberation Model

The multipath model is not valid for all environments. In addition, if there are many different paths, i.e., M is large, it is difficult to estimate all τ_{km} in Equation (7). Recently, a more realistic

reverberation model has been used to describe the TDE problem in a room environment where each sensor often receives a large number of echoes due to reflections of the wavefront from objects and room boundaries such as walls, ceiling, and floor [88]. In addition, it accounts for multiple reflections, as shown in Figure 5(b). In this model, the received signals are expressed as

$$z_k[n] = h_k * s[n] + w_k[n] \tag{8}$$

where $*$ denotes convolution, h_k is the channel impulse response between the source and the kth sensor. It is assumed that $s[n]$ is reasonably broadband and $w_k[n]$ is uncorrelated with $s[n]$ and the noise signals at other sensors.

Time Delay Estimate Based Localization

These techniques typically include two steps, namely, TDE between signals received on spatially separated sensors and source localization based on the time delay estimates [74–79].

Time Delay Estimation

A number of algorithms have been proposed for TDE under various circumstances. Depending on the application, TDE can be categorized as either the time-of-arrival (TOA) estimation [89] or time-difference-of-arrival (TDOA) estimation [90]. TOA estimation methods measure the time delay between transmission and reception of a known signal. TOA estimation methods are part of active systems such as radar and active sonar, where time delay is measured between transmission of a known signal and reception of its echo. On the other hand, TDOA estimation methods are part of passive systems such as passive sonars and microphone array systems, where the time delay is measured as the difference of travel time of an unknown signal between two spatially separated receiving sensors.

The cross-correlation (CC) method is the most straightforward and the earliest developed TDE algorithm, which is formulated based on the ideal signal propagation model (Equation (6)) for single source and two receivers. The time delay estimate with the CC method is obtained as the lag time that maximizes the cross-correlation function (CCF) between two received signals

$$\hat{\tau}_{CC} = \arg\max_m \Psi_{CC}[m] \tag{9}$$

where $\Psi_{CC}[m] = E[z_i[n]z_j[n+m]]$ is the CCF between $z_i[n]$ and $z_j[n]$ and $E[\cdot]$ is the expectation. Other methods for TDE include, generalized cross-correlation (GCC) method and multichannel cross-correlation (MCC) algorithm.

Source Localization

The second step of TDE-based localization includes estimating the location of the sound source using the time delay estimates. For this step a number of approaches have been proposed. These include the maximum likelihood estimation [76] and several closed-form solutions such as spherical interpolation (SI) [74, 75], hyperbolic intersection (HI) [77], and linear intersection (LI) [78]. These closed-form solutions exploit geometric relationship between sensors to approximate the solution. In this section, we review a typical source localization problem formulation based on the time delay estimates.

The localization problem can be stated as follows. Given K sensors, their 3D locations, and TDOA estimates τ_{ij} for each sensor pair ij for a signal source located at an unknown location $\mathbf{x}$, find an estimate for location of the source. For K sensors there are $N = K(K-1)/2$ sensor pairs. The true TDOA associated with the source and a sensor pair ij is given by

$$T(\mathbf{x}, \mathbf{x}_i, \mathbf{x}_j) = \frac{\| \mathbf{x} - \mathbf{x}_i \| - \| \mathbf{x} - \mathbf{x}_j \|}{C}$$

where C is the speed of propagation in the medium. In practice, τ_{ij} is corrupted by noise with variance σ_{ij}^2, and is an unbiased estimate of the true TDOA and possesses a unimodal probability distribution. If the TDOA estimates are assumed to be independently corrupted by additive zero-mean, uncorrelated Gaussian noise, the ML estimate $\hat{\mathbf{x}}^{\mathrm{ML}}$ can be given by

$$\hat{\mathbf{x}}^{\mathrm{ML}} = \arg\min_{\mathbf{x}} \ell(\mathbf{x}) \tag{10}$$

where

$$\ell(\mathbf{x}) = \sum_{i,j} \frac{1}{\sigma_{ij}^2} \left[\tau_{ij} - T(\mathbf{x}, \mathbf{x}_i, \mathbf{x}_j)\right]^2$$

is the negative log-likelihood function. Since $T(\mathbf{x}, \mathbf{x}_i, \mathbf{x}_j)$ is a nonlinear function of $\mathbf{x}$, Equation (10) does not possess a closed-form solution.

Beamforming

Another class of wideband acoustic source localization algorithms is called beamforming [71, 80]. Two common types of beamforming are time-domain and frequency-domain beamforming.

Time-Domain Beamforming

Traditional time-domain delay-and-sum beamforming computes the likelihood that a sound source is at a location by measuring the energy of the reconstructed signal at that location

$$\mathcal{B}_{beam}(\mathbf{x}) = \int_{t_0-W/2}^{t_0+W/2} \left[\sum_{k=1}^{K} z_k(t + \tau_{k,\mathbf{x}}) \right]^2 dt$$

where K is the number of microphones, z_k is the signal received by the kth microphone, $\tau_{k,\mathbf{x}}$ is the travel time for sound to reach microphone k from location $\mathbf{x}$, and t_0 and W are the center and width of the integration window, respectively. Source localization is then performed by maximizing the likelihood

$$\hat{\mathbf{x}}^{\mathrm{ML}} = \arg\max_{\mathbf{x}} \mathcal{B}_{beam}(\mathbf{x})$$

In far-field case, a sensor array cannot estimate the source location. Instead, beamforming is used to estimate the source direction. The ML estimate for source direction in this case can be given by

$$\hat{\theta}^{\mathrm{ML}} = \arg\max_{\theta} \mathcal{B}_{beam}(\theta) = \int_{t_0-W/2}^{t_0+W/2} \left[\sum_{k=1}^{K} z_k(t + \tau_{k,\theta}) \right]^2 dt$$

where $\tau_{k,\theta}$ is the travel time for sound to reach microphone k from a reference location at angle θ. Beamforming generally produces better results because it takes the raw signals into account during source localization, as opposed to the two-step TDE approaches. However, TDE techniques are more computationally efficient because they rely upon the fast operation of cross-correlation.

Frequency-Domain Beamforming

In frequency-domain beamforming, both the array signal model is formulated in frequency-domain. Frequency-domain beamforming formulation and an *optimal* parametric ML solution to locate wideband sources in the near-field and far-field are proposed in [91–93]. In the frequency domain, the

array signal model for multiple sources is given by

$$\mathbf{Z}(q) = \mathbf{D}(q)\boldsymbol{S}_0(q) + \boldsymbol{\eta}(q) \tag{11}$$

for $q = 0, \cdots, Q-1$, where the array data spectrum is given by $\mathbf{Z}(q) = [Z_1(q), \cdots, Z_K(q)]^{\mathrm{T}}$, the steering matrix is given by $\mathbf{D}(q) = [\mathbf{d}^1(q), \cdots, \mathbf{d}^M(q)]$, the steering vector is given by $\mathbf{d}^m(q) = [d_1^m(q), \cdots, d_K^m(q)]^{\mathrm{T}}$, $d_k^m(q) = \mathbf{e}^{-j2\pi q t_k^m / N}$, for $m = 1, \cdots, M$, the source spectrum is given by $\boldsymbol{S}_0(q) = \left[S_0^1(q), \cdots, S_0^M(q)\right]^{\mathrm{T}}$, and K, M are number of sensors and sources, respecitvely. The noise spectrum vector $\boldsymbol{\eta}(q)$ is zero-mean complex white Gaussian, distributed with variance $L\sigma^2$.

ML Source Localization and DOA Estimation The unknown parameter space is $\boldsymbol{\Theta} = \left[\mathbf{X}^{\mathrm{T}}, \boldsymbol{S}_0^{1\mathrm{T}}, \cdots, \boldsymbol{S}_0^{M\mathrm{T}}\right]^{\mathrm{T}}$, where the source locations are denoted by $\mathbf{X} = [\mathbf{x}_1^{\mathrm{T}}, \cdots, \mathbf{x}_M^{\mathrm{T}}]^{\mathrm{T}}$ and the source signal spectrum is denoted by $\boldsymbol{S}_0^m = [S_0^m(1), \cdots, S_0^m(Q/2)]^{\mathrm{T}}$, for $m = 1, \cdots, M$. By stacking up the $Q/2$ positive frequency bins of the signal model in Equation (11) into a single column, the sensor data can be rewritten into an $QK/2 \times 1$ space-temporal frequency vector as

$$\mathbf{Z} = \mathbf{G}(\boldsymbol{\Theta}) + \boldsymbol{\xi} \tag{12}$$

where $\mathbf{G}(\boldsymbol{\Theta}) = [\boldsymbol{S}(1)^{\mathrm{T}}, \cdots, \boldsymbol{S}(Q/2)^{\mathrm{T}}]^{\mathrm{T}}$, $\boldsymbol{S}(q) = \boldsymbol{D}(q)\boldsymbol{S}_0(q)$, and $R_{\boldsymbol{\xi}} = E\left[\boldsymbol{\xi}\boldsymbol{\xi}^H\right] = L\sigma^2\mathbf{I}_{QK/2}$. The negative log-likelihood function of the data is given by $\mathcal{L}(\boldsymbol{\Theta}) = \| \mathbf{Z} - \mathbf{G}(\boldsymbol{\Theta}) \|^2$. The ML estimation of the source locations and source signals is given by

$$\min_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta}) = \min_{\boldsymbol{\Theta}} \sum_{q=1}^{Q/2} \| \mathbf{Z}(q) - \boldsymbol{D}(q)\boldsymbol{S}_0(q) \|^2 \tag{13}$$

which is equivalent to finding $\min_{\mathbf{X}, \boldsymbol{S}_0(q)} \ell(q)$ for all q bins, where

$$\ell(q) = \| \mathbf{Z}(q) - \boldsymbol{D}(q)\boldsymbol{S}_0(q) \|^2 \tag{14}$$

The minima of $\ell(q)$, with respect to the source signal vector $\boldsymbol{S}_0(q)$, must satisfy $\partial\ell(q)/\partial\boldsymbol{S}_0^H(q) = 0$, hence the estimate of the source signal vector which yields the minimum residual at any source location is given by

$$\hat{\boldsymbol{S}}_0(q) = \boldsymbol{D}^{\dagger}(q)\mathbf{Z}(q) \tag{15}$$

where $\boldsymbol{D}^{\dagger}(q) = \left(\boldsymbol{D}(q)^H \boldsymbol{D}(q)\right)^{-1} \boldsymbol{D}(q)^H$ is the pseudo inverse of the steering matrix $\boldsymbol{D}(q)$. Define the orthogonal projection $\boldsymbol{P}(q, \mathbf{X}) = \boldsymbol{D}(q)\boldsymbol{D}^{\dagger}(q)$ and the complement orthogonal projection $\boldsymbol{P}^{\perp}(q, \mathbf{X}) = \mathbf{I} - \boldsymbol{P}(q, \mathbf{X})$. By substituting Equation (15) into (14), the minimization function becomes

$$\ell(q) = \| \boldsymbol{P}^{\perp}(q, \mathbf{X})\mathbf{Z}(q) \|^2 \tag{16}$$

After substituting the estimate of $\boldsymbol{S}_0(q)$, the ML source locations estimate can be obtained by solving the following maximization problem

$$\max_{\mathbf{X}} \mathcal{J}(\mathbf{X}) = \max_{\mathbf{X}} \sum_{q=1}^{Q/2} \| \boldsymbol{P}(q, \mathbf{X})\mathbf{Z}(q) \|^2 \tag{17}$$

Once the ML estimate of $\mathbf{X}$ is obtained, ML estimate of the source signals can be given by Equation (17). Similarly, in the far-field case, the unknown parameter vector contains only the DOAs, that is, $\boldsymbol{\phi}_s = \left[\phi_s^1, \cdots, \phi_s^M\right]^T$. Thus, the ML DOA estimation can be obtained by $\max_{\boldsymbol{\phi}_s} \sum_{q=1}^{Q/2} \| \boldsymbol{P}(q, \boldsymbol{\phi}_s)\mathbf{Z}(q) \|^2$.

Other Signal-Level Localization Methods

Hemisphere Sampling Like TDE, the method involves correlating pairs of microphone signals, but instead of taking the peak of each correlation vector, all the correlation values from all the vectors are accumulated in the common coordinate system, namely a unit hemisphere centered on the microphone array. The maximum cell in the hemisphere then indicates the azimuthal and elevation angles to the source [81].

Accumulated Correlation This method combines the advantages of both beamforming and TDE approaches. Like beamforming, it is accurate because it takes all the information into account before making a decision. Like TDE, it is computationally efficient because it uses cross-correlation as its basic operation [82].

Energy Based Localization

Among feature-based methods, EBL methods use signal energy as acoustic features [83–85]. In free space, acoustic energy decays at a rate that is inversely proportional to the distance from the source. Given simultaneous measurements of acoustic energy of an omnidirectional point source at known sensor locations, the source location can be estimated based on these readings.

Acoustic Energy Attenuation Model

Let K be the number of acoustic sensors and M be the number of omnidirectional acoustic point sources in the sensor field. The acoustic signal received at the kth sensor $k = 1, 2, \cdots, K$ during time interval n then can be expressed as

$$z_k(n) = \gamma_k \sum_{m=1}^{M} \frac{s_m(n - t_{mk})}{\parallel \mathbf{x}_m(n - t_{mk}) - \mathbf{x}_k \parallel^{\alpha/2}} + \nu_k(n)$$

where $\nu_k(n)$ is the background noise modeled as a zero-mean additive white Gaussian (AWGN) noise random variable with variance ς_k^2, $s_m(n - t_{mk})$ is the intensity of the mth acoustic source measured 1 m away from that source, t_{mk} is the propagation delay of the acoustic signal from the mth source to the kth sensor, and $\alpha(\approx 2)$ is an energy decay factor. Algebraic manipulations lead to following acoustic energy decay model

$$y_k(t) = y_{sk}(t) + \epsilon_k(t) = g_k \sum_{m=1}^{M} \frac{S_m(t)}{d_{mk}^{\alpha}}(t) + \epsilon_k(t) \tag{18}$$

where $d_{mk} = \parallel \mathbf{x}_m(t) - \mathbf{x}_k \parallel$ is the distance between the kth sensor and the mth source. The square of the background noise $\nu_k^2(n)$ will have a χ^2 distribution with mean equal to $E[\nu_k^2(n)] = \varsigma_k^2$ and variance equal to $2\varsigma_k^4/M$. For sufficiently large number of samples ($L >> 30$), ϵ_k can be approximated well with a normal distribution, namely, $\epsilon_k \sim \mathcal{N}(\varsigma_k^2, 2\varsigma_k^4/M) \equiv \mathcal{N}(\mu_k, \sigma_k^2)$.

Energy Ratio Formulation

For single target, a least-squares solution based on *energy ratio formulation* is presented in [83]. The basic idea behind energy ratio formulation is that the ratio of the signal energy received at two separate sensors constraints the locus of acoustic source on a hypersphere (i.e. a circle in 2D and a sphere in 3D). If all the sensors that receive the signal from the same target are used, the corresponding target location hyperspheres must intersect at a point that corresponds to the source location.

The acoustic energy decay model for single target (putting $M = 1$ in Equation (18)) is

$$y_k(n) = g_k \frac{S(n)}{|\mathbf{x}(n) - \mathbf{x}_k|^{\alpha}} + \epsilon_k(n) \tag{19}$$

Approximating the additive noise term $\epsilon_k(n)$ in Equation (19) by its mean value μ_k, the energy ratio κ_{ij} of the ith and the jth sensors can be computed as

$$\kappa_{ij} = \left(\frac{(y_i(t) - \mu_i)/(y_j(t) - \mu_j)}{g_i(t)/g_j(t)} \right)^{-1/\alpha} = \frac{|\mathbf{x}(t) - \mathbf{x}_i|}{|\mathbf{x}(t) - \mathbf{x}_j|} \tag{20}$$

Note that for $0 < \kappa_{ij} \neq 1$, all the possible source coordinates $\mathbf{x}(t)$ that satisfy Equation (20) must reside on a hypersphere described by the equation

$$|\mathbf{x}(t) - c_{ij}|^2 = \rho_{ij}^2$$

where the center c_{ij} and the radius ρ_{ij} of this hypersphere, called *target location hypersphere* associated with sensor i and j are given by

$$c_{ij} = \frac{\mathbf{x}_i - \kappa_{ij}^2 \mathbf{x}_j}{1 - \kappa_{ij}^2} \quad \text{and} \quad \rho_{ij} = \frac{\kappa_{ij}|\mathbf{x}_i - \mathbf{x}_j|}{1 - \kappa_{ij}^2}$$

In the limiting case when $\kappa_{ij} \to 1$, the solution of Equation (20) form a *hyperplane* between $\mathbf{x}_i$ and $\mathbf{x}_j$

$$\mathbf{x}(t) \cdot (\mathbf{x}_i - \mathbf{x}_j) = \frac{|\mathbf{x}_i|^2 - |\mathbf{x}_j|^2}{2} \quad \text{or} \quad \mathbf{x}(t) \cdot \gamma_{ij} = \xi_{ij}$$

where $\gamma_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and $\xi_{ij} = \big(|\mathbf{x}_i|^2 - |\mathbf{x}_j|^2\big)/2$. Then, target localization can be modeled as a nonlinear least square optimization problem where the cost function is defined as

$$J(\mathbf{x}) = \sum_{m=1}^{M_1} \left(\parallel \mathbf{x} - c_m \parallel - \rho_m \right)^2 + \sum_{n=1}^{M_2} \left(\gamma_n^{\mathrm{T}} \mathbf{x} - \xi_m \right)^2$$

where m and n are indices of the energy ratios computed between different pairs of sensor energy readings, M_1 is the number of hyperspheres, and M_2 is the number of hyperplanes.

Maximum Likelihood Formulation

A ML formulation for EBL with the capability for multiple source localization is presented in [84]. The method maximizes the joint likelihood of both the source locations and corresponding acoustic energy readings. By stacking the variables and using matrix notation, the acoustic energy decay

model (Equation (18)) can be represented as

$$\mathbf{Z} = \mathbf{GDS} + \boldsymbol{\xi} = \mathbf{HS} + \boldsymbol{\xi}$$

where $\mathbf{Z} = [(y_1 - \mu_1)/\sigma_1 \cdots (y_N - \mu_N)/\sigma_N]^{\mathrm{T}}$, $\mathbf{G} = \mathrm{diag}\,[g_1/\sigma_1 \cdots g_N/\sigma_N]$, $\boldsymbol{S} = [S_1 \cdots S_K]^{\mathrm{T}}$, $\mathbf{H} = \mathbf{GD}$, $\boldsymbol{\xi} = [\xi_1 \cdots \xi_N]^{\mathrm{T}}$, and

$$\mathbf{D} = \begin{bmatrix} 1/d_{11}^2 & \cdots & 1/d_{1K}^2 \\ 1/d_{21}^2 & \cdots & 1/d_{2K}^2 \\ \vdots & \ddots & \vdots \\ 1/d_{N1}^2 & \cdots & 1/d_{NK}^2 \end{bmatrix}$$

where $\xi_i = (\epsilon_i - \mu_i)/\sigma_i \sim \mathcal{N}(0,1)$ are independent Gaussian random variables. The joint probability density function of $\mathbf{Z}$ then is expressed as

$$\mathcal{L}(\boldsymbol{Z}|\boldsymbol{\theta}) = (2\pi)^{-N/2}\exp\left(-\frac{1}{2}(\mathbf{Z} - \mathbf{HS})^{\mathrm{T}}(\mathbf{Z} - \mathbf{HS})\right)$$

where $\boldsymbol{\theta} = [\mathbf{x}_1^{\mathrm{T}} \cdots \mathbf{x}_M^{\mathrm{T}}\ S_1 \cdots S_M]^{\mathrm{T}}$, is the vector of unknown parameters, $\mathbf{x}_m$ and S_m are the location and energy for mth source. The negative log-likelihood function is proportional to a quadratic form

$$\ell(\boldsymbol{\theta}) = \| \mathbf{Z} - \mathbf{GDS} \|^2$$

Thus, the maximum likelihood parameter estimation of $\boldsymbol{\theta}$ can be obtained by minimizing $\ell(\boldsymbol{\theta})$. Two complementary methods are proposed to solve this nonlinear optimization problem. The first method is based on a projection formulation and uses multiresolution search to expedite computation. The second method is based on an Expectation–Maximization (EM) like iterative algorithm.

Summary of Acoustic Source Localization

Table 6 summarizes different acoustic source localization approaches.

Most of the acoustic source localization methods are signal-level fusion methods. TDE based localization is combination of signal processing algorithm for time delay estimation and ML estimation (or closed-form solution) for localization. Beamforming, hemisphere sampling and probabilistic

Table 6: Summary of acoustic source localization approaches

	TDE (SI/HI/LI)	TDE (ML)	BF (TD/FD)	HemiSamp	ProbCorr	ER	EBL (ML)
Fusion type	SIGNAL	SIGNAL	SIGNAL	SIGNAL	SIGNAL	FEATURE	FEATURE
Number of targets	SINGLE	SINGLE	SINGLE/ MULTIPLE	SINGLE	SINGLE	SINGLE	SINGLE/ MULTIPLE
Method	NLS	ML	ML	ML	ML	NLS	ML

correlation are all signal-level fusion methods, hence cannot be directly applied to WSNs. Energy based methods, which use feature-level fusion, are well-suited for WSNs.

One way beamform can be applied to WSNs is by using multiple spatially separated microphone arrays. Each array computes the beamform and the fusion center uses the entire beamform, not just the DOA estimate, for sensor fusion and localization. This would be an example of feature-level fusion methods, where beamform has been used as an acoustic feature.

In general, feature-level fusion methods are not very common in acoustic source localization and tracking. As WSN applications incorporate more acoustic sensors, it is important to discover and define relevant acoustic features and feature-level fusion algorithms.

Target Localization and Tracking using Video

Target tracking, also called object tracking, is an important task in the field of computer vision and image processing. It is key part of a number of applications such as automated surveillance, traffic monitoring, vehicle navigation, motion-based recognition, human-computer interaction, etc. In its simplest form, object tracking can be defined as the problem of estimating the trajectory and other desired properties, such as orientation, area, or shape of an object as it moves around in a scene. Most common challenges to object tracking are, (1) loss of information caused by projection of the 3D world on a 2D image, (2) noise in images, (3) complex object motion, (4) nonrigid or articulated nature of objects, (5) partial and full object occlusions, (6) complex object shapes, (7) scene illumination changes, and (8) real-time processing requirements.

The key aspects of any object tracking approach are, (1) object representation, e.g. using points, simple geometric shapes, contours or appearance models, (2) selection and extraction of image features from incoming frames, such as color, motion, edges, etc., (3) object detection based on the features, and (4) a tracking algorithm that maintains and updates an estimate of object trajectory and other desired properties. In this section, we review these aspects and various object tracking approaches.

Object Representation

Objects can be represented by their shapes and/or appearances. The most commonly employed shape representations are following [94].

1. Points or a set of points that represent a fixed marker on the object, e.g. centroid. In general, the point representation is suitable for tracking objects that occupy small regions in an image.

2. Primitive geometric shapes, e.g. rectangle, ellipse, etc. Such representations are suitable for tracking objects with rigid shapes.

3. Object silhouette and contour representation that defines the boundary of an object. Such representations are suitable for tracking complex nonrigid shapes.

4. Articulated shape models are used to represent objects that are composed of rigid parts held together with joints. For example, the human body is an articulated object with torso, legs, hands, head, and feet connected by joints.

5. Skeletal models are used to represent object shape outline and connectivity neglecting shape details. This model is commonly used as a shape representation for recognizing objects.

The most commonly employed object appearance representations are following.

1. Probability densities of object appearance, including parametric densities, e.g. Gaussian, mixture of Gaussian, etc., or nonparametric, e.g. histograms [95].

2. Templates that are formed using one or more simple geometric shapes or silhouettes. An advantage of a template is that it carries both spatial and appearance information.

3. Active appearance models that are generated by simultaneous modeling of object shape and appearance. In general, the object shape is defined by a set of landmarks, where for each landmark, an appearance vector is stored which is in the form of color, texture, or gradient.

4. Multiview appearance models that encode different views of an object. One approach to represent the different object views is to generate a subspace from the given views. Subspace approaches, for example, Principal Component Analysis (PCA) and Independent Component Analysis (ICA), have been used for both shape and appearance representation [96].

Feature Selection for Tracking

The most desirable property of a visual feature, also called visual cue, is its uniqueness and discernibility, so that the objects can be easily distinguished in the feature space. Feature selection is closely related to object representation. For example, color is used as a feature for histogram-based appearance representations, while for contour-based representation, object edges are usually used as features. In general, many tracking algorithms use a combination of these features. The details of common visual features are as follows.

Motion

In object tracking, motion feature refers to moving foreground in the video. The motion feature is computed by background subtraction via frame-differencing. Correct extraction of motion feature requires robust and adaptive background model. There exist a number of challenges for the estimation of robust background model [97], including gradual and sudden illumination changes, vacillating backgrounds, shadows, visual clutter, occlusion, etc. In practice, most of the simple motion detection algorithms have poor performance when faced with these challenges.

Color

The apparent color of an object is influenced primarily by two physical factors, (1) the characteristics of the light source and (2) the surface reflectance properties of the object. In image processing, the RGB (red, green, blue) color space is usually used to represent color. However, the RGB space is not a perceptually uniform color space and is highly sensitive to illumination changes. HSV (hue, saturation, value) space, on the other hand, is approximately uniform in perception. The hue parameter in HSV space represents color information, which is illumination invariant as long as the following two conditions hold, (1) the light source color can be expected to be almost white, and (2) the saturation value of object color is sufficiently large [98].

The color feature is computed using one of the two methods. In the simple model, the color of each pixel in the current image is compared to a prototype color $[h_0(t), s_0(t), v_0(t)]$ describing the color of the tracked object, e.g. face color in [99]. The pixels with acceptable deviation from the prototype color constitute the color cue. In a more complex model, the prototype color is modeled with adaptive mixture model. An adaptive Gaussian mixture model in hue-saturation space is used in [100].

Texture

Visual textures are the patterns in the intensity variations of a surface. The patterns can be the result of physical surface properties such as roughness, or they could be the result of reflectance differences such as the color on a surface. Image texture is defined as a function of the spatial variation in pixel intensities. Texture is the most important visual feature in identifying different surface types, which is called texture classification. Image contrast, defined as the standard deviation of the grayscale values within a small image region, is considered a simple texture feature [100].

Shape

An object can be represented as a primitive geometric shape, e.g. rectangle, ellipse, etc. Such object shapes can be called templates, and they carry both spatial and appearance information. The shape feature is computed by *template matching*. Template matching is a brute force method of searching the input image for a region similar to the shape template [99]. The similarity of the region with the template is defined as a similarity measure, e.g. cross correlation.

Prediction

In object tracking applications, the position of the object can be predicted using the previous position and the motion model. When the object is represented by a geometric shape, the motion model is usually in the form of a parametric transformation of the shape, such as translation, rotation, and affine [99, 100].

Edge Map

Edge detection algorithms, such as Canny Edge detector [101], are used to identify points in an image at which the image intensity changes sharply or has discontinuities. Such discontinuities in image intensity are generated by object boundaries. In general, edges are are less sensitive to illumination changes compared to color features. Algorithms that track the boundary of the objects usually use edges as the representative feature.

Interest Points

Interest points in images are the points in the image that have an expressive texture in their respective localities. A desirable quality of an interest point is its invariance to changes in illumination and

camera viewpoint. Commonly used interest point detectors include Moravec's interest operator [102], Harris interest point detector [103], KLT detector, and SIFT detector [104].

Optical Flow

Optical flow is a dense field of displacement vectors that defines the translation of each pixel in a region. It is computed using the brightness constraint, which assumes brightness constancy of corresponding pixels in consecutive frames [105]. Optical flow is commonly used as a feature in motion-based segmentation and tracking applications.

Video Feature Fusion

In many situations information garnered from a single feature is unreliable, incomplete or intermittent. For example, most colorspaces are sensitive to illumination variation. Such situations require the use to multiple visual features for correct and continued operation. In other applications, the use of multiple cues will increase the robustness and reliability of estimation. Fusion of multiple video features is also called cue integration.

Probabilistic approaches have been widely used in fusion of multiple cues [100, 106, 107]. These approaches require a model that fits to the data and prior distributions of possible results, hence often called the *model-based approaches* for feature fusion/cue integration. An alternative to the use of strong models is *model-free approach* to cue integration [99, 100, 108]. These approaches exploit the incidental agreement of multiple cues where methods like voting and fuzzy logic can be used for fusion. These approaches have been widely used in cases where a precise mathematical model of the controlled process is not available. Various cue integration techniques are following.

Bayesian Fusion

In a Bayesian approach, the posterior of the target model given the current image is computed using the Bayes rule. Let $C = \{C_1, \cdots, C_K\}$ represent a set of K visual cues extracted from an image I of an object (e.g., edge maps, surface normals, color), and X represent a particular target model. Using Bayes theorem

$$p(X|I) \quad \propto \quad \underbrace{p(I|X)}_{\text{likelihood}} \quad \underbrace{p(X)}_{\text{prior}} \quad \propto \quad \underbrace{p(C|X)}_{\text{appearance}} \qquad \underbrace{p(X)}_{\text{spatial configuration}} \tag{21}$$

In Equation (21), I is substituted by C to indicate that the image information is represented by a set of visual cues. The target model is estimated using the MAP estimate as

$$\hat{X} = \arg\max_X p(X|I) = \arg\max_X p(X|C)$$

Voting-Based Fusion

The key feature of the voting methods is that each cue makes a decision independent of all other cues before the results are combined. Voting enables increased reliability of an integrated system consisting of a number of modules/cues where the reliability of each individual module varies significantly over time. In principle, each cue estimator may be a simple classifier that votes for a particular attribute or against it (binary voting), and the votes from all cues are fused in a simple way to produce global decision.

Voting can also be loosely put in a probabilistic framework by allowing each cue to compute posterior independent of all other cues and then fuse the posterior together using weighted summation [100]

$$\text{score}(X) = \sum_k w_k p_k(X|Z_k) = \sum_k w_k \frac{p_k(Z_k|X)p(X)}{p(Z_k)}$$

and the target is estimated as

$$\hat{X} = \arg\max_X \text{score}(X)$$

In a similar technique, called *Democratic Integration*, different cues agree on a result, and each cue adapts toward the result [99]. In particular, discordant cues are quickly suppressed and recalibrated, while cues that have been consistent with the result in the recent past are given a higher weight in the future.

Object Detection

Object detection algorithms detect and/or estimate objects of interest in incoming frames based on the selected/extracted image features. An object detection algorithm can either use information in a single frame or use the temporal information computed from a sequence of frames. Various types of object detection algorithms are summarized below.

Point Detectors

Point detectors are used to find interest points in images which have an expressive texture in their respective localities. A desirable quality of an interest point is its invariance to changes in illumination and camera viewpoint. Commonly used interest point detectors include Moravec's interest operator [102], Harris interest point detector [103], KLT detector, and SIFT detector [104].

Background Subtraction

Object detection, especially for moving objects, can be achieved by building a representation of the scene called the background model and then finding deviations from the model for each incoming frame via frame differencing. The pixels constituting the regions undergoing change are marked as foreground for further processing. This process is referred to as the background subtraction. There exist a number of challenges for the estimation of robust background models [97], including gradual and sudden illumination changes, vacillating backgrounds, shadows, visual clutter, occlusion, etc.

In order to learn gradual changes over time, modeling the color of each pixel of a stationary background with a single 3D Gaussian in YUV color space has been proposed [109]. The model parameters are learned from the color observations over several consecutive frames. However, a single Gaussian is not a good model for complex scenes, such as outdoor and urban environments [110], since multiple colors can be observed at a certain location due to repetitive object motion, shadows, or reflectance.

Further improvements in background modeling are achieved by using multimodal statistical models to describe per-pixel background color. A Kalman filter to track the changes in background illumination for every pixel is used in [111]. A per-pixel adaptive parametric mixture model of three Gaussian distributions is proposed in [112]. A kernel estimator for each pixel is proposed in [113] with kernel exemplars from a moving window. Other techniques using high-level processing to assist the background modeling have been proposed; for instance, the Wallflower tracker [97] which circumvents some of these problems using high-level processing rather than tackling the inadequacies of the background model. An adaptive nonparametric Gaussian mixture model to address background modeling challenges is done in [114]. In this method, a pixel in the current frame is checked against the background model by comparing it with every Gaussian in the model until a matching Gaussian is found. If a match is found, the mean and variance of the matched Gaussian is updated, otherwise a new Gaussian with the mean equal to the current pixel color and some initial variance is introduced into the mixture model. The algorithm in [115] proposes an adaptive background modeling method

based on the framework in [114]. The main differences lie in the update equations, initialization method and the introduction of a shadow detection algorithm. The motion detection algorithm that we use in Chapter III is based on the adaptive background modeling method described in [115].

Image Segmentation

Image segmentation algorithms partition the image into perceptually similar regions. Every segmentation algorithm addresses two problems, the criteria for a good partition and the method for achieving efficient partitioning. Image segmentation algorithms include approaches such as (1) mean-shift clustering to find clusters in the joint spatial-color ($[l, u, v, x, y]$) space [116], (2) graph-cuts, where image segmentation is modeled as graph partitioning problem, and (3) active contours, where segmentation is achieved by evolving a closed contour to the object's boundary.

Supervised Learning

Object detection can be performed by learning to classify different object views automatically from a set of examples using supervised learning mechanisms. Supervised learning methods usually require a large collection of samples from each object class. Supervised learning algorithms commonly used in object tracking include (1) adaptive boosting, where an accurate classifier is constructed by combining a number of simple base classifiers, each of which may only be moderately accurate [117, 118], and (2) support vector machines (SVM) to classify objects by finding the maximum marginal hyperplane that separates one object class from another.

Object Tracking

The aim of an object tracker is to maintain and update the trajectory of an object and other desired properties over time. The tasks of object detection and object association across time can either be performed separately or jointly. In the first case, possible objects in every frame are obtained by means of an object detection algorithm, and then the tracker associates the objects across frames. In the latter case, called joint data association and tracking, the objects and associations are jointly estimated by iteratively updating object state. Tracking algorithm strongly depends on the object representation. Various object tracking approaches are described below.

Point Tracking

When objects are represented as points, the association of the points is based on the previous object state which can include object position and motion. The target tracking algorithm reviewed in Section II are, in fact, point tracking approaches. For single object tracking, Kalman filter and particle filters have been used extensively [46]. For multiple object tracking and data association, Kalman filter and particle filters are used in conjunction with a data association algorithm that associates the most likely measurement for an object to its state. The two widely used techniques for data association are JPDAF [6] and MHT [39].

Kernel Tracking

Kernel refers to the object shape and appearance. Objects represented using shape and appearance models are tracked by computing the motion of the kernel in consecutive frames. This motion is usually in the form of a parametric transformation of the kernel, such as translation, rotation, and affine. Kernel tracking can also be called model-based tracking, since the kernel is actually the object model. Kernel tracking approaches are based on the templates and density-based appearance models used for object representation.

Templates and density-based appearance models have been widely used because of their relative simplicity and low computational cost. In kernel tracking, objects can be tracked individually or jointly. For single targets, the most common approach is *template matching*. Usually image intensity or color features are used to form the templates. Since image intensity is very sensitive to illumination changes, image gradients [119] can also be used as features. A limitation of template matching is its high computation cost due to the brute force search. Another approach, called *mean-shift tracker* maximizes the appearance similarity iteratively by comparing the histograms of the object and the window around the hypothesized object location. Histogram similarity is defined in terms of the Bhattacharya coefficient [95, 116].

In joint tracking of multiple objects, the interaction between objects is explicitly modeled allowing the algorithm to handle partial of full occlusion of the objects. An object tracking method based on modeling the whole image as a set of layers is proposed in [120]. This representation includes a single background layer and one layer for each object. Each layer consists of shape priors (ellipse), motion model (translation and rotation), and layer appearance, (intensity modeled using a single Gaussian). Joint modeling of the background and foreground regions for tracking multiple objects is proposed in Bramble [121]. The appearance of background and all foreground objects are modeled

by mixture of Gaussian. The shapes of objects are modeled as cylinders. They assume the ground plane is known, thus the 3D object positions can be computed. Tracking is achieved by using particle filters where the state vector includes the 3D position, shape and the velocity of all objects in the scene.

A color-based probabilistic tracking approach is presented in [122]. The tracking algorithm uses global color reference models and endogenous initialization. They implemented the tracker in a sequential Monte Carlo framework. They defined a color likelihood function based on color histogram distances, the coupling of the color model with a dynamical state space model, and the sequential approximation of the resulting posterior distribution with a particle filter. The use of a sample-based filtering technique permits in particular the momentary tracking of multiple posterior modes. This is the key to escape from background distraction and to recover after partial or complete occlusions.

Silhouette Tracking

When objects are represented as silhouettes or contours, tracking is performed by either shape matching or contour evolution [123]. Both of these methods can essentially be considered as object segmentation applied in the temporal domain using the priors generated from the previous frames. Shape matching approaches search for the object silhouette in the current frame. Contour tracking approaches, on the other hand, evolve an initial contour to its new position in the current frame by either using the state space models or direct minimization of some energy functional.

Tracking with Camera Network

According to the overview paper [124], smart cameras are evolving on three different evolutionary paths. First, *single smart cameras* focus on integrating sensing with embedded on-camera processing power to perform various vision tasks on-board and deliver abstracted data from the observed scene. Second, *distributed smart cameras* (DSC) introduces distribution and collaboration of smart cameras resulting in a network of cameras with distributed sensing and processing. The main motivations for DSC are to (1) resolve occlusion, (2) mitigate single camera handicap, and (3) extend sensing coverage. Finally, *pervasive smart cameras* (PSC) integrate adaptivity and autonomy to DSC. According to the authors, the ultimate vision of PSC is to provide a service-oriented network which is easy to deploy and operate, adapts to changes in the environment and provides various customized services to users.

All of the single camera tracking algorithms are applied in the (2D) image-plane. These (2D) image-plane trackers often run into problems such as target scale selection, target rotation, occlusion, view-dependence, and correspondence across views [125]. There are few 3D tracking approaches [125,126] that fuse results from independent 2D trackers to obtain 3D trajectories. These approaches employ decision-level fusion, wherein local decisions made by the node (i.e. 2D tracks) are fused to achieve global decision (i.e. 3D tracks), while discarding the local information (i.e. images captured at nodes). Because of the decision-level fusion, these approaches also suffer from all the problems associated with 2D tracking.

Target Handoff

Another problem in 2D image-plane based trackers is the (re)initialization of a target when it (re)enters a camera field-of-view. In current state-of-the-art approaches, (re)initialization is performed by handing over target state to adjacent camera node which maintains the target state until it hands over the state to some other camera node.

An autonomous multicamera tracking approach based on a fully decentralized handover mechanism between adjacent cameras in presented in [127]. The system automatically initiates a single tracking instance for each target of interest. The instantiated tracker for each target follows the target over the camera network, migrating the target state to the camera which observed the object. This approach, however, utilizes data from only single camera node at any given time. The authors do admit that the effectiveness of their handover mechanism introduces some requirements for the tracker. First, the tracker must have short initialization time to build a target model. Second, the tracker on the new camera node must be able to initialize itself from a previously saves state, or handed-over state. Finally, the tracker must be robust with respect to the position and orientation of a target such that it must be able to identify the same target on the next camera node. These requirements need sophisticated and fine-tuned algorithms for 2D image-plane based tracker. On the other hand, in 3D trackers, the target state and the target model are maintained in 3D space. The target state and the target model are not tied to the camera network parameters, such as the number of cameras, position and orientation of the cameras, etc. Once initialized, the target model does not need to be re-initialize as it moves in the sensing region and enters, or re-enters a camera field-of-view.

3D Tracking using Ray Intersection

The classical and most naive approach for 3D collaborative target tracking is to combine the 2D tracking results from individual camera nodes for 3D tracking. This can be done by projecting rays from the camera center to the image affine coordinate in the world coordinate system and finding the intersection of multiple such rays from multiple cameras. This approach requires each camera node to maintain a 2D target model and a feature histogram. Hence the problems of scale variation, rotation and occlusion are not alleviated. As soon as the target moves along the camera principal axis, or rotates around its axis, the target model including the size and feature-histogram would become invalid. A target model learning approach while target tracking can help mitigate the problem but any sudden change which is faster than the model learning would cause the tracker to loose the target.

Summary of Video Tracking

There has been a good deal of research on target modeling, making it as close to represent reality as possible, sometimes at the expense of real-time performance. On the sensing and feature extraction side, there has also been a lot of research, however, some features such as color, require a template or prior information. This assumes feedback behavior, where fusion center informs the sensors the template to expect. In HSNs, this might not be a reasonable expectation. It might be appropriate to make the features and feature extraction model-free, i.e. no template or prior information is required. Secondly, all feature-fusion (or cue integration) approaches take the grayscale images as featuremap inputs. This is not suitable for HSNs because sensors cannot send grayscale images to fusion center. Another way is needed to compress the features in a concise format so that they can be communicated over wireless.

Multimodal Target Localization and Tracking

Multimodal sensor fusion and tracking refers to the case when more than a single sensing modality is used to track targets. These modalities can complement each other in cases when one or some of the modalities are compromised. For example in audio-video tracking, the modalities can complement each other in the presence of high background noise that impairs the audio, or in the presence of visual clutter that handicap the video. Additionally, tracking based on multimodal fusion can improve the performance of tracking based on single modality alone. In some cases, some modalities can provide cues for the other modality for actuation. For example, visual steering information

from a video sensor may be used to steer an audio sensor array toward a moving target. Similarly, information from the audio sensors can be used to steer a pan-tilt-zoom camera toward a speaker.

Multimodal sensor fusion and tracking have been applied to biometric speaker identification [128], speech recognition [129], smart videoconferencing [130, 131], indoor tracking applications [50, 132, 133], and multimodal surveillance [12, 134–136].

A speaker identification approach that combines face recognition and audio-visual speech-based biometric identification is described in [128]. The temporal sequence of audio and visual observations obtained from the acoustic speech and the shape of the mouth are modeled using a set of coupled hidden Markov models (CHMM). An audio-visual speech recognition approach using geometric features of lip movement and speech signals is described in [129]. For recognition, speech signals are modeled by hidden Markov models (HMMs) which incorporate both audio and video speech feature (i.e. lip geometric features).

A smart videoconferencing system based on a multimodal tracker using multiple cameras and microphone arrays is proposed in [130]. The video data consist of pairs of image coordinates of features on each tracked object, and the audio data consist of TDOA for each microphone pair in the array. A particle filter is implemented for multimodal tracking.

An approach for audiovisual object tracking based on graphical models that combine the audio and video data is proposed in [50]. The graphical model is designed for single target tracking using a single camera and a microphone pair. The audio data consist of the signals received at the microphones, and the video data consist of grayscale video frames. Generative models are described for the audio-video data that explain the data in terms of the target state. An expectation-maximization (EM) algorithm is described for parameter learning and tracking, which also enables automatic calibration by learning the audio-video calibration parameters during each EM step. A probabilistic framework for indoor multitarget tracking using audio and video data is described in [132]. The video data consist of plan-view images of the foreground. Plan-view images are projections of the images captured from different points-of-view to a 2D plane, usually the ground plane where the targets are moving [137]. Acoustic beamforms for a fixed length signal are taken as the audio data. A particle filter is implemented for tracking. An approach using Time-Delay Neural Network (TDNN) to fuse audio and video data at the feature level for detecting a walking person is proposed in [133]. The TDNN learns the relation between visual motion and step sounds of the walking person for tracking.

A smart surveillance system named CASSANDRA, aimed at detecting aggressive human behavior in public environments is presented in [134]. CASSANDRA exploits the complimentary nature of audio

and video sensing to disambiguate scene activity in real-life, noisy and dynamic environments. The system is validated on a set of scenarios performed by professional actors at an actual train station to ensure a realistic audio and video noise setting. A technology, called audio-assisted cameras is proposed for multimodal surveillance in [135]. The audio-assisted cameras are PTZ cameras augmented with an array of microphones. These cameras, rather than indiscriminate recording of all activity, record the activity of interest by taking cue from microphone array. The ARL Multi-Modal Sensor (MMS) is a research program at the U.S. Army Research Laboratory (ARL) for the development of low cost sensing techniques for the urban environment [136]. The program objectives were to develop a networked personnel detection sensor with the following major criteria: low cost in volume production, support MOUT (Military Operations in Urban Terrain) operations, and employ non-imaging sensor diversity techniques. The physical parameters sensed are seismic, acoustic, and thermal using an accelerometer, microphone, and passive infrared (PIR) transducers, respectively.

A recent book on multimodal surveillance address a number of aspects of multimodal surveillance, from front-end sensors to systems and environmental issues [138]. The book examines thermal, vibration, video, and audio sensors in a broad context of civilian and military applications.

CHAPTER III

AUDIO-VIDEO FUSION BASED TARGET TRACKING

This chapter presents a multimodal multisensor information fusion system for target tracking in an urban environment using HSN consisting of audio and video sensors. We demonstrate the approach utilizing an HSN of mote class devices equipped with microphone arrays as audio sensors and embedded PCs equipped with web cameras as video sensors. We take two different approaches for multisensor information fusion based on Sequential Bayesian Estimation (SBE) and Monte Carlo Markov Chain Data Association (MCMCDA) algorithm. The targets to be tracked are moving vehicles emitting engine noise. The system has many components including audio processing, video processing, WSN middleware services, multimodal sensor fusion, and target tracking based on SBE and MCMCDA.

Introduction

Heterogeneous sensor networks (HSNs) are gaining popularity in diverse fields, such as military surveillance, equipment monitoring, and target tracking [4]. They are natural steps in the evolution of wireless sensor networks (WSNs) driven by several factors. Increasingly, WSNs will need to support multiple, although not necessarily concurrent, applications. Different applications may require different resources. Some applications can make use of nodes with different capabilities. As the technology matures, new types of nodes will become available and existing deployments will be refreshed. Diverse nodes will need to coexist and support old and new applications.

Furthermore, as WSNs are deployed for applications that observe more complex phenomena, multiple sensing modalities become necessary. Different sensors usually have different resource requirements in terms of processing power, memory capacity, or communication bandwidth. Instead of using a network of homogeneous devices supporting resource intensive sensors, an HSN can have different nodes for different sensing tasks [36–38]. For example, at one end of the spectrum low data-rate sensors measuring slowly changing physical parameters such as temperature, humidity or light, require minimal resources; while on the other end even low resolution video sensors require orders of magnitude more resources.

Tracking is one such application that can benefit from multiple sensing modalities [139]. If the moving target emits sound signal then both audio and video sensors can be utilized. These

modalities can complement each other in the presence of high background noise that impairs the audio, or in the presence of visual clutter that handicap the video. Additionally, tracking based on the fusion of audio-video data can improve the performance of audio-only or video-only approaches. Audio-video tracking can also provide cues for the other modality for actuation. For example, visual steering information from a video sensor may be used to steer the audio sensor (microphone array) toward a moving target. Similarly, information from the audio sensors can be used to steer a pan-tilt-zoom camera toward a speaker. Although, audio-visual tracking has been applied to smart videoconferencing applications [131], it does not use a wide-area distributed platform .

Multimodal sensor fusion and tracking pose a number of challenges. In multimodal multisensor fusion, the data may be fused at a variety of levels including the raw data level, where the raw signals are fused, a feature-level, where representative characteristics of the data are extracted and fused, and a decision-level fusion wherein target estimates from each sensor are fused [59]. At successive levels more information may be lost, but in collaborative applications, such as WSN applications, the communication requirement of transmitting large amounts of data is reduced. Another significant challenge is sensor conflict, when different sensors report conflicting data. When sensor conflict is very high, fusion algorithms produce false or meaningless results [140]. Reasons for sensor conflict may be sensor locality, different sensor modalities, or sensor faults. If a sensor node is far from a target of interest then the data from that sensor will not be useful, and will have higher variance. Different sensor modalities observe different physical phenomena. For example, audio and video sensors observe sound sources and moving objects, respectively. If a sound source is stationary or a moving target is silent, the two modalities might produce conflicting data. Also, different modalities are affected by different types of background noise. Finally, poor calibration and sudden change in local conditions can also cause conflicting sensor data.

Another classical problem in multitarget tracking is to find a track of each target from the noisy data. If the association of sequence of data-points with each target is known, multitarget tracking reduces to a set of state estimation problems. The *data association* problem is to find which data-points are generated by which targets, or in other words, associate each data-point with either a target or noise. Our system employs a Markov Chain Monte Carlo Data Association (MCMCDA) algorithm for tracking. The MCMCDA algorithm is a data-oriented, combinatorial optimization approach to the data association problem that avoids the enumeration of tracks. The MCMCDA algorithm enables us to track an unknown number of targets in noisy urban environment

In this chapter, we describe our approach to multimodal target tracking in urban environments utilizing an HSN of mote class devices equipped with microphone arrays as audio sensors and embed-

ded PCs equipped with web cameras as video sensors. The targets to be tracked are moving vehicles emitting engine noise. Our system has many components including audio processing, video processing, WSN middleware services, multimodal sensor fusion, and target tracking based on sequential Bayesian estimation and MCMCDA. While none of these is necessarily novel, their composition and implementation on an actual HSN requires addressing a number of significant challenges.

Specifically, we have implemented audio beamforming on audio sensors utilizing an FPGA-based sensor board and evaluated its performance as well its energy consumption. While we are using a standard motion detection algorithm on video sensors, we have implemented post-processing filters that represent the video data in a similar format as the audio data, which enables seamless audio-video data fusion. Furthermore, we have extended time synchronization techniques for HSNs consisting of mote and PC networks. Finally, the main challenge we address is system integration including making the system work on an actual platform in a realistic deployment. The chapter provides results gathered in an uncontrolled urban environment and presents a thorough evaluation including a comparison of different fusion and tracking approaches.

Comparison with Related Work

Classical target tracking approaches are based on decision-level fusion, and assume one-to-one association between target and measurements [5,6,39,44]. Our approach is feature-level fusion and allows for target interaction, occlusion and mixed observations [9, 12, 14]. This is especially important in cases where multiple targets are in close proximity of each other.

Most of the research in video tracking focus on multiple feature fusion on single camera system [99, 100, 107]. Little work has been done in surveillance using camera network [127]. Even in such camera networks, the individual cameras perform tracking without collaboration with other cameras. Collaborative target tracking in camera network requires sharing of visual features with multiple sensors. Our multimodal tracking system [10,12] uses a network of cameras that collaborate with each other for feature-level fusion.

Multimodal sensor fusion has been used in applications such as intelligent/interactive terminals [129], biometric identification [128], videoconferencing [130, 131], smart environment and indoor surveillance [50, 132, 133]. These applications are designed for small sensing region, e.g. face, a desk, or a room. Like multimodal surveillance approaches in [134–136], our approach is also targeted for multimodal surveillance in urban environments [11, 12], e.g. section of a road.

Target Application	Intelligent/ interactive term. [123], biometric ident. [122]	Videoconf. [124, 125], smart/ interactive envr. [50, 126, 127]	Surveil. and monitor. [128, 129, 130]	Our approach for tracking [11, 12]
Technique Used	face tr. , speech recog.	speaker ident./ tr., indoor tr., face tr.	target det., loc. and tr.	target det., loc. and tr.
Sensing Region	small/face	medium/room	road, busy public places	section of road
Features Used	face-color, audio spectrum, lip geometry and motion	face-color, motion, audio signals	motion, acoustic, seismic, and thermal signals	motion, acoustic beamform, PSD
Fusion Level	signal-level	signal-level	signal-level (local), decision-level (global)	feature-level
Hardware/ Platform	high-cost, centralized, wired	high-cost, centralized, wired	low-cost, distributed, wireless	low-cost, distributed, wireless

Figure 6: Comparison to Related Work.

The comparison to related work is shown in Figure 6, with the proposed work [11, 12] shown in the last column. Most of these systems are either single sensors systems with multiple feature fusion [100, 107, 128], or multiple sensors connected to the same hardware platform, e.g. a PTZ camera connected with a microphone array [134, 135]. Ours is a distributed system, with multiple audio and video sensors deployed over a large area and connected through wireless [11,12]. Tracking algorithms on such single or multiple sensor systems employ signal-level fusion. The distributed system in [136] uses decision-level fusion. Our system uses feature-level fusion, hence loses the least amount of information [12, 16].

Architecture

Figure 7 shows the system architecture. The audio sensors, consisting of MICAz motes with acoustic sensor boards equipped with a microphone array, form an IEEE 802.15.4 network. This network does not need to be connected; it can consist of multiple connected components as long as each of these component have a dedicated mote-PC gateway. The video sensors are based on Logitech QuickCam Pro 4000 cameras attached to OpenBrick-E Linux embedded PCs. These video sensors, along with the mote-PC gateways, the sensor fusion node and the reference broadcaster for time synchronization (see Section III) are all PCs forming a peer-to-peer 802.11b wireless network. Figure 8 shows the conceptual layout of the sensor network.

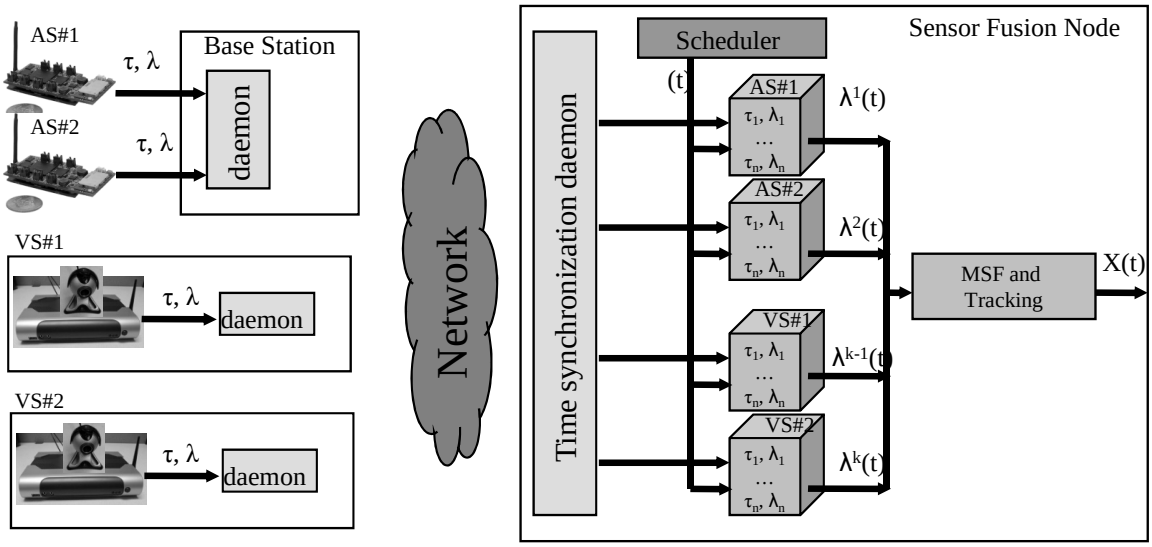


Figure 7: Architecture for the multimodal target tracking system. Here, τ denotes measurement timestamps, λ denotes sensor measurements (also called detection functions that are described in later sections), and t denotes time. The blocks shown inside the sensor fusion node are circular buffers that store timestamped measurements.

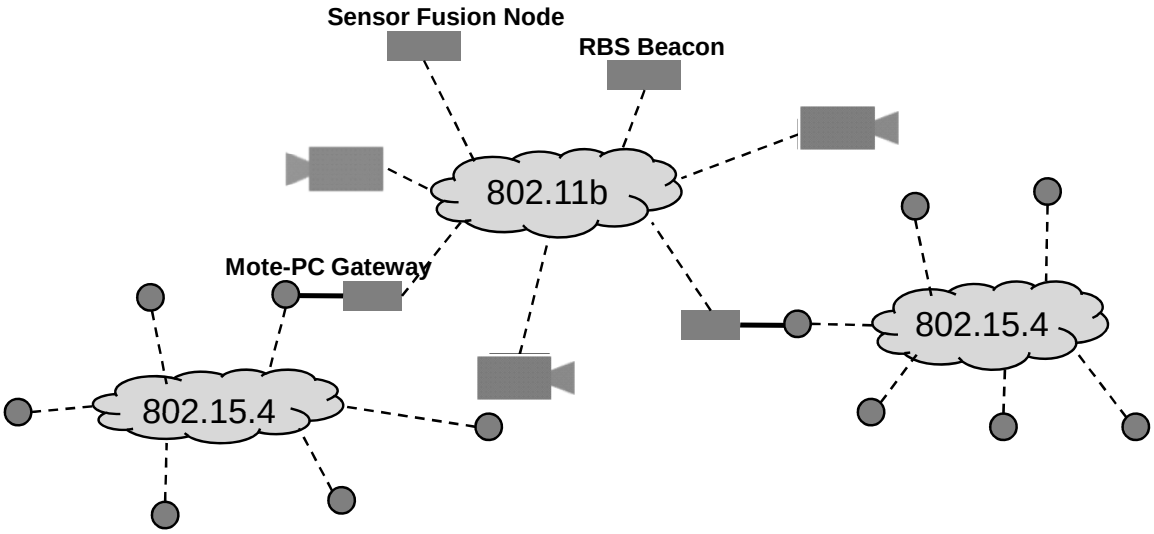


Figure 8: Conceptual layout of sensor network for multimodal target tracking. Circles represent the audio sensors, camera silhouettes represent the video sensors and rectangles represent PCs.

The audio sensors perform beamforming, and transmit the audio detections to the corresponding mote-PC gateway utilizing a multi-hop message routing service [141]. The routing service also performs time translation of the detection timestamps. The video sensors run a motion detection algorithm and compute timestamped video detection functions. Both audio and video detections are routed to the central sensor fusion node. Time translation is also carried out in the 802.11b network

utilizing an RBS-based time synchronization service. This approach ensures that sensor data arrives at the sensor fusion node with timestamps in a common timescale. On the sensor fusion node, the sensor measurements are stored in appropriate sensor buffers, one for each sensor. A sensor fusion scheduler triggers periodically and generates a fusion timestamp. The trigger is used to retrieve the sensor measurements from the sensor buffers with timestamps closest to the generated fusion timestamp. The retrieved sensor measurements are then used for multimodal fusion and target tracking. In this study, we developed and compared target tracking algorithms based on both Sequential Bayesian Estimation (SBE) and MCMCDA. Note that the triggering mechanism of the scheduler decouples the target tracking rate from the audio and video sensing rates, which allows us to control the rate of the tracking application independently of the sensing rate.

Audio Beamforming

Beamforming is a signal processing algorithm for DOA estimation of a signal source. In a typical beamforming array, each of the spatially separated microphones receive a time-delayed source signal. The amount of time delay at each microphone in the array depends on the microphone arrangement and the location of the source. A typical delay-and-sum single source beamformer discretizes the sensing region into directions, or *beams*, and computes a beam energy for each of them. The beam energies are collectively called the *beamform*. The beam with maximum energy indicates the direction of the acoustic source.

Beamforming Algorithm

The data-flow diagram of our beamformer is shown in Fig. 9. The amplified microphone signal is sampled at a high frequency (100 KHz) to provide high resolution for the time delay, which is required for the closely placed microphones. The raw signals are filtered to remove unwanted noise components. The signal is then fed to a tapped delay line (TDL), which has M different outputs to provide the required delays for each of the M beams. The delays are set by taking into consideration the exact relative positions of the microphones so that the resulting beams are steered to the beam angles, $\theta_i = i\frac{360}{M}$ degrees, for $i = 0, 1, ...M-1$. The signal is downsampled and the M beams are formed by adding the four delayed signals together. Data blocks are formed from the data streams (with a typical block length of 5-20ms) and an FFT is computed for each block. The block power

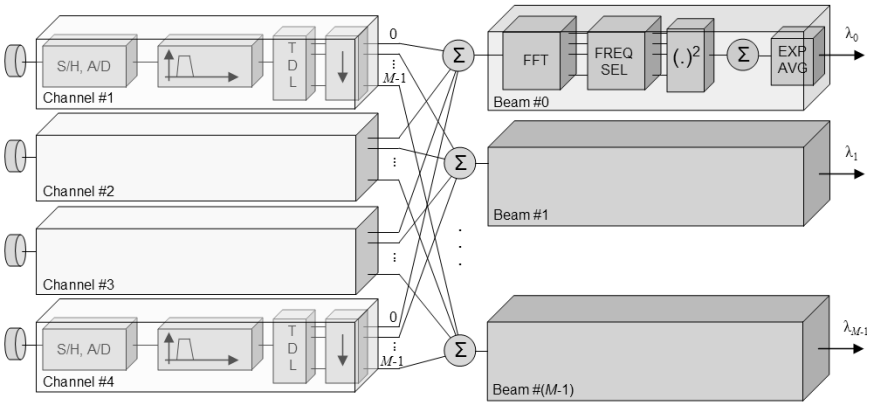


Figure 9: Data-flow diagram of the real-time beamforming sensor.

values, $\mu(\theta_i)$, are smoothed by exponential averaging into the beam energy, $\lambda(\theta_i)$

$$\lambda^t(\theta_i) = \alpha\lambda^{t-1}(\theta_i) + (1-\alpha)\mu(\theta_i) \tag{22}$$

where α is an averaging factor.

Audio Hardware

In our application, the audio sensor is a MICAz mote with an onboard Xilinx XC3S1000 FPGA chip that is used to implement the beamformer [142]. The onboard Flash (4MB) and PSRAM (8MB) modules allow storing raw samples of several acoustic events. The board supports four independent analog channels, featuring an electret microphone each, sampled at up to 1 MS/s (million samples per second). A small beamforming array of four microphones arranged in a $10cm \times 6cm$ rectangle is placed on the sensor node, as shown in Fig. 10(a). Since the distances between the microphones are small compared to the possible distances of sources, the sensors perform far-field beamforming. The sources are assumed to be on the same two-dimensional plane as the microphone array, thus it is sufficient to perform planar beamforming by dissecting the angular space into M equal angles, providing a resolution of $360/M$ degrees. In the experiments, the sensor boards are configured to perform simple delay-and-sum-type beamforming in real time with $M = 36$, and an angular resolution of 10 degrees. Finer resolution increases the communication requirements.

Evaluation

We test the beamformer node outdoors using recorded speech as the acoustic source. Measurements are taken by placing the source at distances of 3, 5, 10, 25, 50, 75, 100, and 150 feet from the sensor,

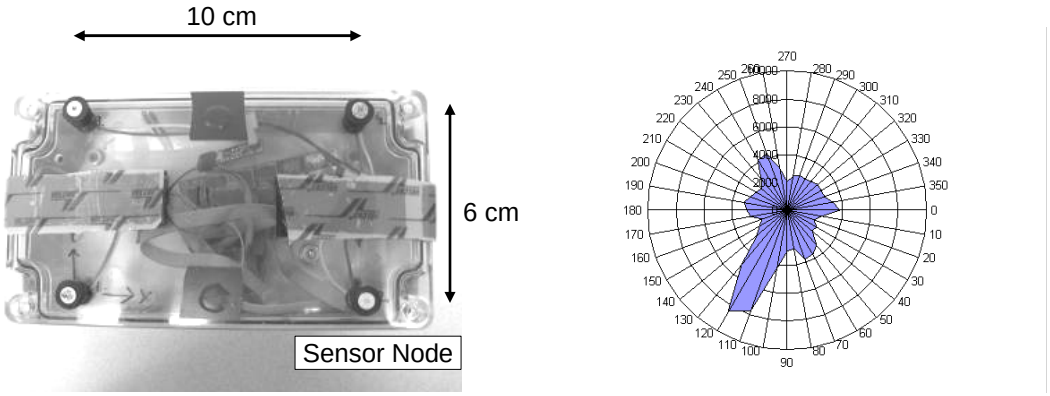


Figure 10: (a) Sensor Node Showing the Microphones, (b) Beamform of acoustic source at a distance of 50 feet and an angle of 120 degrees.

and at an angle from -180° to +180° in 5° increments. Figure 10(b) shows the beamforming result for a single audio sensor when the source was at a distance of 50 feet. Mean DOA measurement error for 1800 human speech experiments is shown in Figure 11. The smallest error was recorded when the source was at a distance of 25 feet from the sensor. The error increases as the source moves closer to the sensor. This is because the beamforming algorithm assumes a planar wavefront, which holds for far-field sources but breaks down when the source is close to the sensor. However, as the distance between the source and sensor grows, error begins accumulating again as the signal-to-noise ratio decreases.

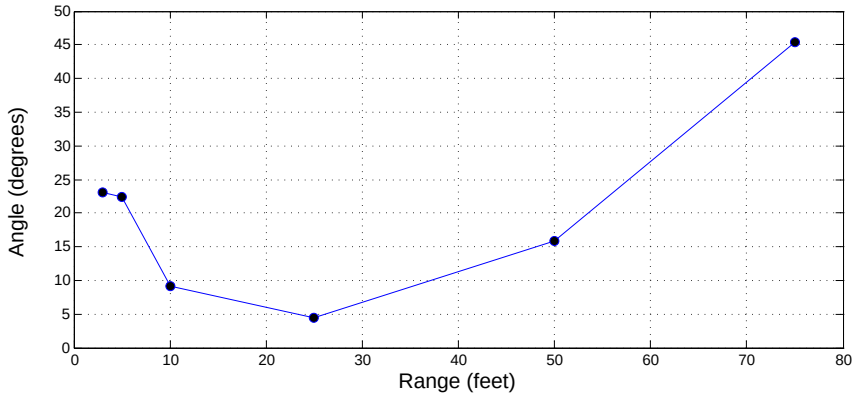


Figure 11: Human speech DOA error for distances of 3, 5, 10, 25, 50, 75 feet between acoustic source and beamformer node.

Messages containing the audio detection functions require 83 bytes, and include node ID, sequence number, timestamps, and 72 bytes for 36 beam energies. These data are transmitted through the network in a single message. The default TinyOS message size of 36 bytes was changed to 96 bytes

to accommodate the entire audio message. The current implementation uses less than half of the total resources (logic cells, RAM blocks) of the selected mid-range FPGA device. The application runs at 20 MHz, which is relatively slow in this domain—the inherent parallel processing topology allows this slow speed. Nonetheless, the FPGA approach has a significant impact on the power budget, the sensor draws 130mA current (at 3.3 V) which is nearly a magnitude higher then typical wireless sensor node power currents. Flash-based FPGAs and smart duty cycling techniques are promising new directions in our research project for reducing the power requirements.

Video Tracking

Video tracking systems seek to automatically detect moving targets and track their movement in a complex environment. Due to the inherent richness of the visual medium, video based tracking typically requires a pre-processing step that focuses attention of the system on regions of interest in order to reduce the complexity of data processing. This step is similar to the visual attention mechanism in human observers. Since the region of interest is primarily characterized by regions containing moving targets (in the context of target tracking), robust motion detection is the first step in video tracking. A simple approach to motion detection from video data is via frame differencing. It compares each incoming frame with a background model and classifies the pixels of significant variation into the foreground. The foreground pixels are then processed for identification and tracking. The success of frame differencing depends on the robust extraction and maintenance of the background model. Performance of such techniques tends to degrade when there is significant camera motion, or when the scene has significant amount of change.

Algorithm

The dataflow in Figure 12 shows the motion detection algorithm and its components used in our tracking application. The first component is background-foreground segmentation of the currently

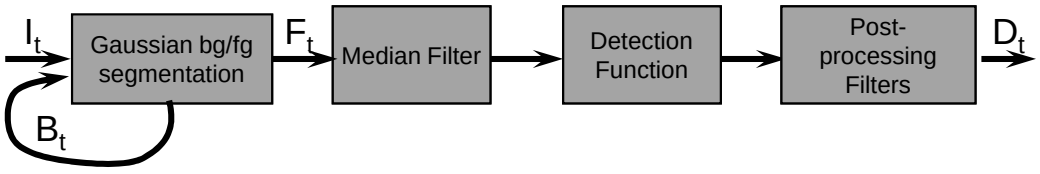


Figure 12: Data-flow diagram of real-time motion detection algorithm

captured frame (I_t) from the camera. We use the algorithm described in [115]. This algorithm uses an adaptive background mixture model for real-time background and foreground estimation.

The mixture method models each background pixel as a mixture of K Gaussian distributions. The algorithm provides multiple tunable parameters for desired performance. In order to reduce speckle noise and smooth the estimated foreground (F_t), the foreground is passed through a median filter. In our experiments, we use a median filter of size 3×3.

Since our sensor fusion algorithm (Section III) utilizes only the angle of moving targets, it is desirable and sufficient to represent the foreground in a simpler detection function. Similar to the beam angle concept in audio beamforming (Section III), the field-of-view of the camera is divided into M equally-spaced angles

$$\theta_i = \theta_{min} + (i-1)\frac{\theta_{max} - \theta_{min}}{M} : i = 1, 2, ..., M \tag{23}$$

where θ_{min} and θ_{max} are the minimum and maximum field-of-view angles for the camera. The detection function value for each angle is simply the number of foreground pixels in that direction. Formally, the detection function for the video sensors can be defined as

$$\lambda(\theta_i) = \sum_{j \in \theta_i}^{W} \sum_{k=1}^{H} F(j,k) : i = 1, 2, ..., M \tag{24}$$

where F is the binary foreground image, H and W are the vertical and horizontal resolutions in pixels, respectively and $j \in \theta_i$ indicates columns in the frame that fall within angle θ_i. Figure 13 shows a snapshot of motion detection.

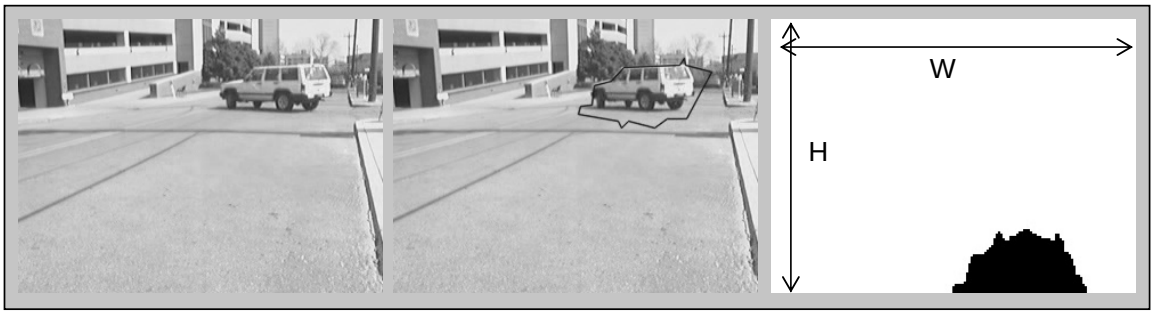


Figure 13: Video detection. The frame on the left shows the input image, the frame in the middle shows the foreground and the frame on the right shows the video detection function.

Video Post-processing

In our experiments, we gathered video data of vehicles from multiple sensors from an urban street setting. The data contained a number of real-life artifacts such as vacillating backgrounds, shadows,

sunlight reflections and glint. The algorithm described above is not able to filter out such artifacts from the detections. We implemented two post-processing filters to improve the detection performance. The first filter removes any undesirable persistent background. For this purpose we keep a background of moving averages which was removed from each detection. The background update and filter equations are

$$b^t(\theta) = \alpha b^{t-1}(\theta) + (1-\alpha)\lambda^t(\theta) \quad (25)$$

$$\lambda^t(\theta) = \lambda^t(\theta) - b^{t-1}(\theta) \quad (26)$$

where $b^t(\theta)$ is the moving average background, and $\lambda^t(\theta)$ is the detection function at time t. The second filter removes any sharp spikes (typically caused by sunlight reflections and glint). For this we convolved the detection function with a small linear kernel to add a blurring effect. This essentially reduces the effect of any sharp spikes in detection function due to glints. The equation for this filter is

$$\lambda^t(\theta) = \lambda^t(\theta) * k \quad (27)$$

where k is a 7×1 vector of equal weights, and $*$ denotes convolution.

We implemented the motion detection algorithm using OpenCV (open source computer vision) library. We use Linux PCs equipped with the QuickCam Pro 4000 as video sensors. The OpenBrick-E has 533 MHz CPU, 128 MB RAM, and a 802.11b wireless adapter. The QuickCam Pro supports up to 640×480 pixel resolution and up to 30 frames-per-second frame rate. Our motion detection algorithm implementation runs at 4 frames-per-second and 320×240 pixel resolution. The number of angles in Equation (23) is $M = 160$.

Time Synchronization

In order to seamlessly fuse time-dependent audio and video sensor data for tracking moving objects, participating nodes must have a common notion of time. Although several microsecond-accurate synchronization protocols have emerged for wireless sensor networks (e.g. [24, 143–145]), achieving accurate synchronization in a *heterogeneous* sensor network is not a trivial task.

Attempting to synchronize the entire network using a single protocol will introduce a large amount of error. Protocols such as TPSN [143], FTSP [144], and RITS [145] are designed to run on the mote-class devices, whereas protocols such as Reference Broadcast Synchronization (RBS) [24] is designed for networks where all nodes have access to a common network medium.

In this work, we adopted the hybrid approach described in [141], which pairs a specific network with the synchronization protocol that provides the most accuracy with the least amount of overhead. Synchronize across the entire network is handled by several synchronization mechanisms running on gateway nodes (i.e., nodes connecting multiple networks).

Multimodal Target Tracking

This section describes the tracking algorithm and the approach for fusing the audio and video data based on sequential Bayesian estimation. We use following notation: Superscript t denotes discrete time ($t \in \mathbb{Z}^+$), subscript $k \in \{1, ..., K\}$ denotes the sensor index, where K is the total number of sensors in the network, the target state at time t is denoted as $x^{(t)}$, and the sensor data at time t is denoted as $z^{(t)}$.

Sequential Bayesian Estimation

We use sequential Bayesian estimation to estimate the target state $x^{(t)}$ at time t, similar to the approach presented in [146]. In sequential Bayesian estimation, the target state is estimated by computing the posterior probability density $p(x^{(t+1)}|z^{(t+1)})$ using a Bayesian filter described by

$$p(x^{(t+1)}|z^{(t+1)}) \propto p(z^{(t+1)}|x^{(t+1)}) \int p(x^{(t+1)}|x^{(t)}) p(x^{(t)}|z^{(t)}) dx^{(t)} \tag{28}$$

where $p(x^{(t)}|z^{(t)})$ is the prior density from the previous step, $p(z^{(t+1)}|x^{(t+1)})$ is the likelihood given the target state, and $p(x^{(t+1)}|x^{(t)})$ is the prediction for the target state $x^{(t+1)}$ given the current state $x^{(t)}$ according to a target motion model. Since we are tracking moving vehicles it is reasonable to use a directional motion model based on the target velocity. The directional motion model is described by

$$x^{(t+1)} = x^{(t)} + v + \mathcal{U}[-\delta, +\delta] \tag{29}$$

where $x^{(t)}$ is the target state time t, $x^{(t+1)}$ is the predicted state, v is the target velocity, and $\mathcal{U}[-\delta, +\delta]$ is a uniform random variable.

Because the sensor models (described later in Subsection III) are nonlinear, the probability densities cannot be represented in a closed form. It is, therefore, reasonable to use a nonparametric representation for the probability densities. The nonparametric densities are represented as discrete grids in 2D space, similar to [146]. For nonparametric representation, the integration term in Equation (28) becomes a convolution operation between the motion kernel and the prior distribution. The

resolution of the grid representation is a trade-off between tracking resolution and computational capacity.

Centralized Bayesian Estimation

Since we use resource constrained mote class sensors, centralized Bayesian estimation is a reasonable approach because of its modest computational requirements. The likelihood function in Equation (28) can be calculated either as a product or weighted summation of the individual likelihood functions. We describe the two methods next.

Product of Likelihood functions Let $p_k(z^{(t)}|x^{(t)})$ denote the likelihood function from sensor k. If the sensor observations are mutually independent conditioned on the target state, the likelihood functions from multiple sensors are combined as

$$p(z^{(t)}|x^{(t)}) = \prod_{k=1,\ldots,K} p_k(z^{(t)}|x^{(t)}) \tag{30}$$

Weighted-Sum of Likelihood functions An alternative approach to combine the likelihood functions is to compute their weighted-sum. This approach allows us to give different weights to different sensor data. These weights can be used to incorporate sensor reliability and quality of sensor data. We define a quality index $q_k^{(t)}$ for each sensor k as

$$q_k^{(t)} = r_k \max_{\theta} (\lambda_k^{(t)}(\theta))$$

where r_k is a measure of sensor reliability and $\lambda_k^{(t)}(\theta)$ is the sensor detection function. The combined likelihood function is given by

$$p(z^{(t)}|x^{(t)}) = \frac{\sum_{k=1,\ldots,K} q_k^{(t)} p_k(z^{(t)}|x^{(t)})}{\sum_{k=1,\ldots,K} q_k^{(t)}} \tag{31}$$

We experimented with both methods in our evaluation. The product method produces more accurate results with low uncertainty in target state. The weighted-sum method performs better in cases with high sensor conflict, though it suffers from high uncertainty. The results are presented in the evaluation section.

Hybrid Bayesian Estimation

In sensor fusion a big challenge is to account for conflicting sensor data. When sensor conflict is very high, sensor fusion algorithms produce false or meaningless fusion results [140]. Reasons for sensor conflict are sensor locality, different sensor modalities, and sensor faults. Selecting and clustering the sensors in different groups based on locality or modality can mitigate poor performance due to sensor conflict. For example, clustering the sensors close to the target and fusing the data from only the sensors in the cluster would remove the conflict caused by distant sensors.

The sensor network deployment in this paper is small and the sensing region is comparable to the sensing ranges of the audio and video sensors. For this reason, we do not use locality based clustering. However, we have multimodal sensors that can report conflicting data. Hence, we developed a hybrid Bayesian estimation framework [59] by clustering sensors based on modalities, and compare it with the centralized approach. Figure 14 illustrates the framework. The likelihood function from each

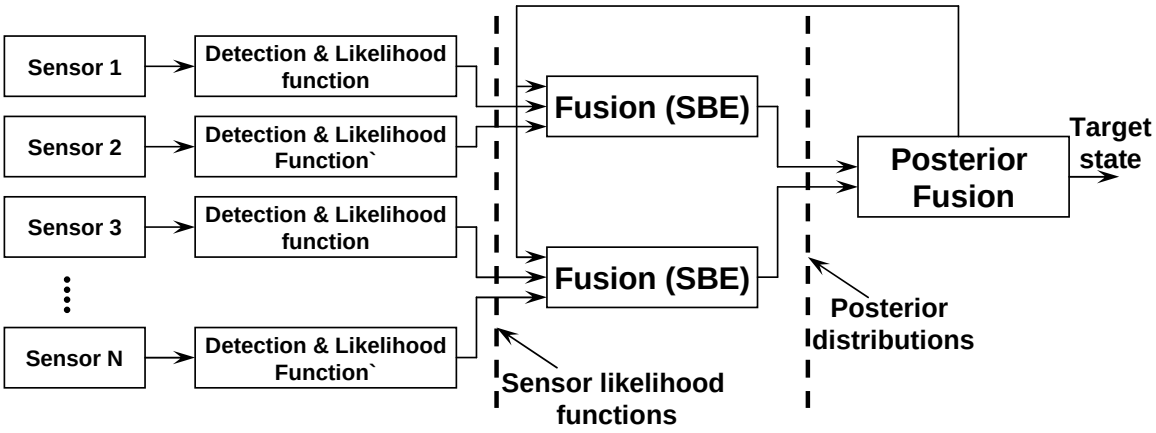


Figure 14: Hybrid Bayesian estimation framework.

sensor in a cluster is fused together using Equation (30) or (31). The combined likelihood is then used in Equation (28) to calculate the posterior density for that cluster. The posterior densities from all the clusters are then combined together to estimate the target state.

For hybrid Bayesian estimation with audio-video clustering, we compute the likelihood functions using Equation (30) or (31). The audio posterior density is calculated using

$$p_{audio}(x^{(t+1)}|z^{(t+1)}) \propto p_{audio}(z^{(t+1)}|x^{(t+1)}) \int p(x^{(t+1)}|x^{(t)}) p(x^{(t)}|z^{(t)}) dx^{(t)}$$

while the video posterior density is calculated as

$$p_{video}(x^{(t+1)}|z^{(t+1)}) \propto p_{video}(z^{(t+1)}|x^{(t+1)}) \int p(x^{(t+1)}|x^{(t)})p(x^{(t)}|z^{(t)})dx^{(t)}$$

The two posterior densities are combined either as (product fusion)

$$p(x^{(t+1)}|z^{(t+1)}) \propto p_{audio}(z^{(t+1)}|x^{(t+1)})p_{video}(z^{(t+1)}|x^{(t+1)})$$

or (weighted-sum fusion)

$$p(x^{(t+1)}|z^{(t+1)}) \propto \alpha p_{audio}(z^{(t+1)}|x^{(t+1)}) + (1-\alpha)p_{video}(z^{(t+1)}|x^{(t+1)})$$

where α is a weighing factor.

Sensor Models

We use a nonparametric model for the audio sensors, while a parametric mixture-of-Gaussian model for the video sensors is used to mitigate the effect of sensor conflict in object detection.

Audio Sensor Model

The nonparametric DOA sensor model for a single audio sensor is the piecewise linear interpolation of the audio detection function

$$\lambda(\theta) = w\lambda(\theta_{i-1}) + (1-w)\lambda(\theta_i), \text{ if } \theta \in [\theta_{i-1}, \theta_i]$$

where $w = (\theta_i - \theta)/(\theta_i - \theta_{i-1})$.

Video Sensor Model

The video detection algorithm captures the angle of one or more moving objects. The detection function from Equation (24) can be parametrized as a mixture-of-Gaussian

$$\lambda(\theta) = \sum_{i=1}^{n} a_i f_i(\theta)$$

where n is the number of components, $f_i(\theta)$ is the probability density, and a_i is the mixing proportion for component i. Each component is a Gaussian density given by $f_i(\theta) = \mathcal{N}(\theta|\mu_i, \sigma_i^2)$, where the component parameters μ_i, σ_i^2 and a_i are calculated from the detection function.

Likelihood Function

The 2D search space is divided into N rectangular cells with center points at (x_i, y_i), for $i = 1, 2, ..., N$ as illustrated in Figure 15. The likelihood function value for k^{th} sensor at i^{th} cell is the average

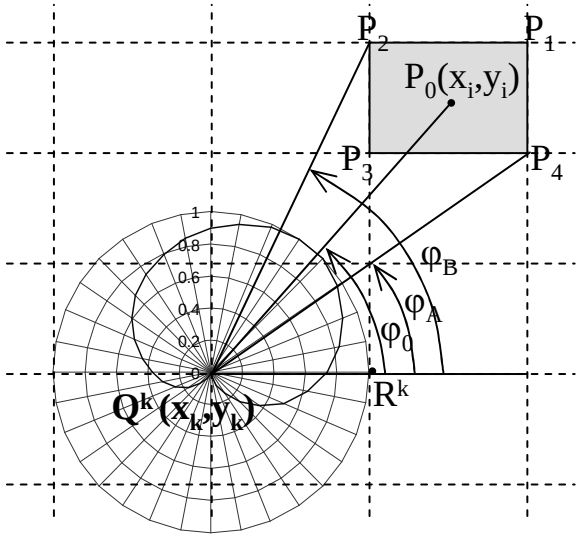


Figure 15: Computation of the likelihood value for k^{th} sensor at i^{th} cell, (x_i, y_i). The cell is centered at P_0 with vertices at P_1, P_2, P_3, and P_4. The angular interval subtended at the sensor due to the i^{th} cell is $\theta \in [\varphi_A^{(k,i)}, \varphi_B^{(k,i)}]$, or $\theta \in [0, 2\pi]$ if the sensor is inside the cell.

value of the detection function in that cell, given by

$$p_k(z|x) = p_k(x_i, y_i) = \frac{1}{(\varphi_B^{(k,i)} - \varphi_A^{(k,i)})} \sum_{\varphi_A^{(k,i)} \leq \theta \leq \varphi_B^{(k,i)}} \lambda_k(\theta)$$

Multiple-Target Tracking

The essence of the multi-target tracking problem is to find a track of each object from noisy measurements. If the sequence of measurements associated with each object is known, multi-target tracking reduces to a set of state estimation problems for which many efficient algorithms are available. Unfortunately, the association between measurements and objects is unknown. The *data association* problem is to work out which measurements were generated by which objects; more precisely, we require a partition of measurements such that each element of a partition is a collection of measurements generated by a single object or noise. Due to this data association problem, the complexity

of the posterior distribution of the states of objects grows exponentially as time progresses. It is well-known that the data association problem is NP-hard [147], so we do not expect to find efficient, exact algorithms for solving this problem.

In order to handle highly nonlinear and non-Gaussian dynamics and observations, a number of methods based on particle filters has recently been developed to track multiple objects in video [42,43]. Although particle filters are highly effective in single-target tracking, it is reported that they provide poor performance in multi-target tracking [42]. This is because a fixed number of particles is insufficient to represent the posterior distribution with the exponentially increasing complexity (due to the data association problem). As shown in [41,42], an efficient alternative is to use Markov chain Monte Carlo (MCMC) to handle the data association problem in multi-target tracking.

For our problem, there is an additional complexity. We do not assume the number of objects is known. A *single-scan* approach, which updates the posterior based only on the current scan of measurements, can be used to track an unknown number of targets with the help of trans-dimensional MCMC [41, 42] or a detection algorithm [43]. But a single-scan approach cannot maintain tracks over long periods because it cannot revisit previous, possibly incorrect, association decisions in the light of new evidence. This issue can be addressed by using a *multi-scan* approach, which updates the posterior based on both current and past scans of measurements. The well-known *multiple hypothesis tracking* (MHT) [39] is a multi-scan tracker, however, it is not widely used due to its high computational complexity.

A newly developed algorithm, called Markov chain Monte Carlo data association (MCMCDA), provides a computationally desirable alternative to MHT [44]. The simulation study in [44] showed that MCMCDA was computationally efficient compared to MHT with heuristics (*i.e.*, pruning, gating, clustering, N-scan-back logic and k-best hypotheses). In this chapter, we use the online version of MCMCDA to track multiple objects in a 2-D plane. Due to the page limitation, we omit the description of the algorithm in this paper and refer interested readers to [44].

Evaluation

In this section, we evaluate target tracking algorithms based on the sequential Bayesian estimation and MCMCDA.

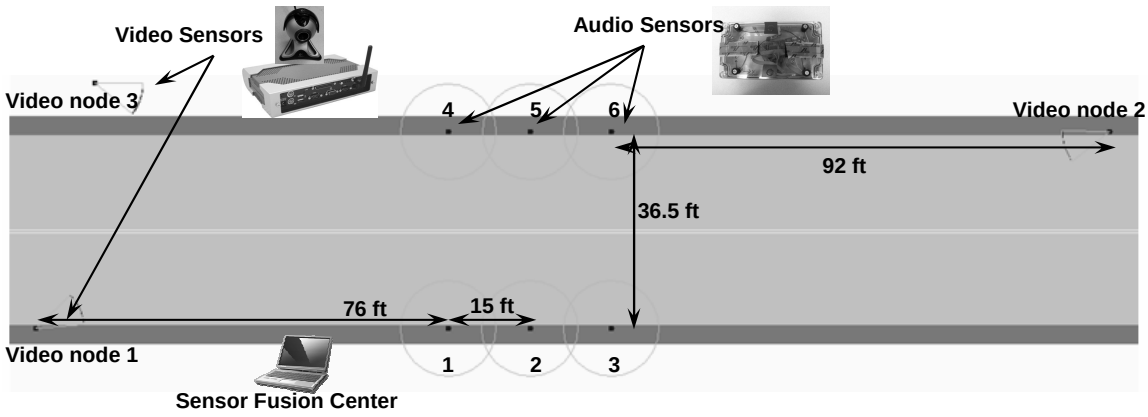


Figure 16: Experimental setup

Table 7: Parameters used in experimental setup

Number of beams in audio beamforming, M_{audio}	36
Number of angles in video detection M_{video}	160
Sensing region (meters)	35×20
Cell size (meters)	0.5×0.5

Sequential Bayesian Estimation

The deployment of our multimodal target tracking system is shown in Figure 16. We employ 6 audio sensors and 3 video sensors deployed on either side of a road. The complex urban street environment presents many challenges including gradual change of illumination, sunlight reflections from windows, glints due to cars, high visual clutter due to swaying trees, high background acoustic noise due to construction and acoustic multipath effects due to tall buildings. The objective of the system is to detect and track vehicles using both audio and video under these conditions.

Sensor localization and calibration for both audio and video sensors are required. In our experimental setup, the sensor nodes are manually placed at marked locations and orientations. The audio sensors are placed on 1 meter high tripods to minimize audio clutter near the ground. An accurate self-calibration technique, e.g. [148, 149], is desirable for a target tracking system. Our experimental setup consists of two wireless networks as described in Section III. The mote network is operating on channel 26 (2.480 GHz) while the 802.11b network is operating on channel 6 (2.437 GHz). Both the channels are non-overlapping and different from the infrastructure wireless network, which operates on channel 11 (2.462 GHz). We choose these non-overlapping channels to minimize interference and are able to achieve less than 2% packet loss.

We gather audio and video detection data for a total duration of 43 minutes. Table 7 presents the parameter values that we use in our tracking system. We run our sensor fusion and tracking

system online using centralized sequential Bayesian estimation based on the product of likelihood functions. We also collect all the audio and video detection data for offline evaluation. This way we are able to experiment with different fusion approaches on the same data set. For offline evaluations, we shortlist 10 vehicle tracks where there is only a single target in the sensing region. The average duration of tracks is 4.25 sec with 3.0 sec minimum and 5.5 sec maximum. The tracked vehicles are part of an uncontrolled experiment. The vehicles are traveling on road at a speed of 20-30 mph speed.

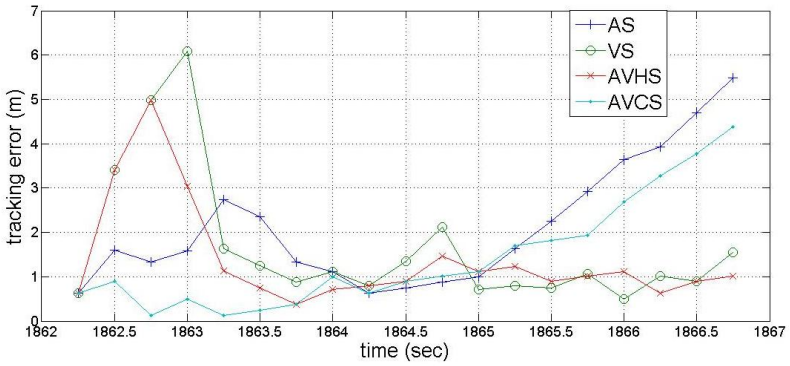
Sequential Bayesian estimation requires a prior density of the target state. We initialize the prior density using a simple detection algorithm based on audio data. If the maximum of the audio detection functions exceeds a threshold, we initialize the prior density based on the audio detection.

In our simulations, we experiment with eight different approaches. We use audio-only, video-only and audio-video sensor data for sensor fusion. For each of these data sets, the likelihood is computed either as the weighted-sum or product of the likelihood function for the individual sensors. For the audio-video data, we use centralized and hybrid fusion. Following is the list of different target tracking approaches.

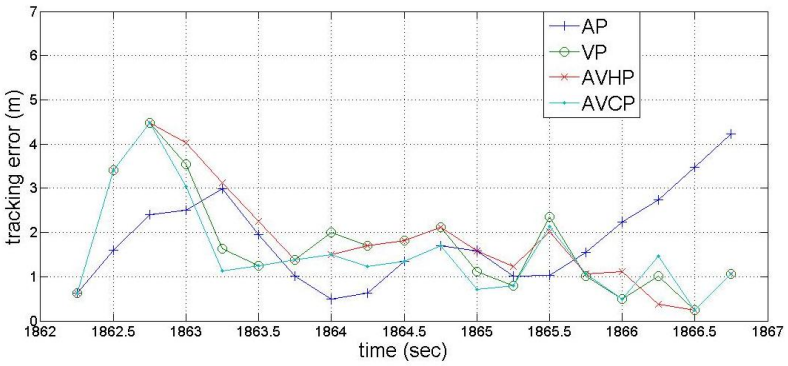
1. audio-only, weighted-sum (AS)
2. video-only, weighted-sum (VS)
3. audio-video, centralized, weighted-sum (AVCS)
4. audio-video, hybrid, weighted-sum (AVHS)
5. audio-only, likelihood product (AP)
6. video-only, likelihood product (VP)
7. audio-video, centralized, likelihood product (AVCP)
8. audio-video, hybrid, likelihood product (AVHP)

The ground truth is estimated post-facto based on the video recording by a separate camera. The standalone ground truth camera is not part of any network, and have the sole responsibility of recording ground truth video. For evaluation of tracking accuracy, the center of mass of the vehicle is considered to be the true location.

Figure 17 shows the tracking error for a representative vehicle track. The tracking error when audio data is used is consistently lower than the case when the video data is used. When we



(a)

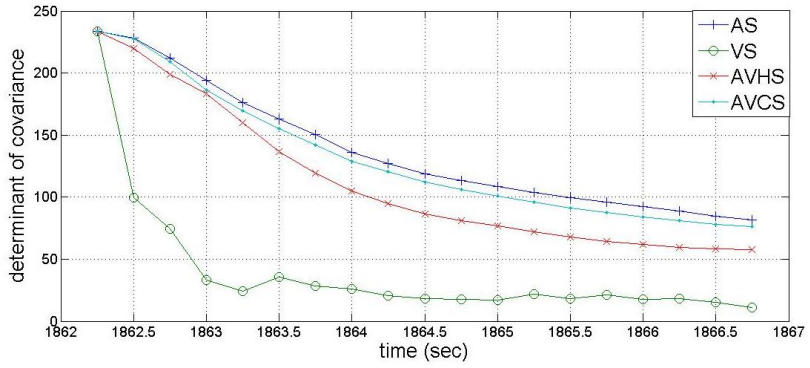


(b)

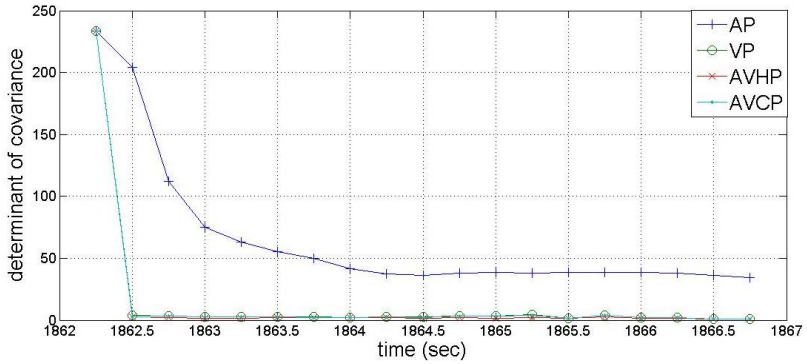
Figure 17: Tracking error (a) weighted sum, (b) product

use both audio and video data, the tracking error is lower than either of those considered alone. Figure 18 shows the determinant of the covariance of the target state for the same vehicle track. The covariance, which is an indicator of uncertainty in the target state is significantly lower for product fusion than weighted-sum fusion. In general, covariance for audio-only tracking is higher than video-only tracking, while using both modalities lowers the uncertainty.

Figure 19(a) shows the tracking error in the case of fusion based on weighted-sum. The tracking error when using only video data shows a large value at time $t = 1067$ second. In this case, the video data has false peaks not corresponding to the target. The audio fusion works fine for this track. As expected, when we use both audio and video data together, the tracking error is decreased. Further, Figure 19(a) shows the tracking error when using 6, 3 and 2 audio sensors for the centralized audio-video fusion. Using as few as two audio sensors can assist video tracking to disambiguate and remove false peaks. Similarly, when there is an ambiguity in the audio data, the video sensors can assist and improve tracking performance. Figure 19(b) shows the tracking error for a track where audio data is poor due to multiple sound sources. When fused with video data the tracking performance



(a)



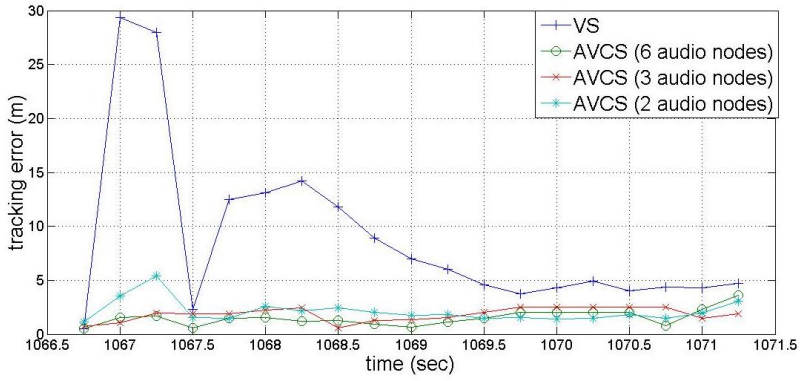
(b)

Figure 18: Tracking variance (a) weighted sum, (b) product

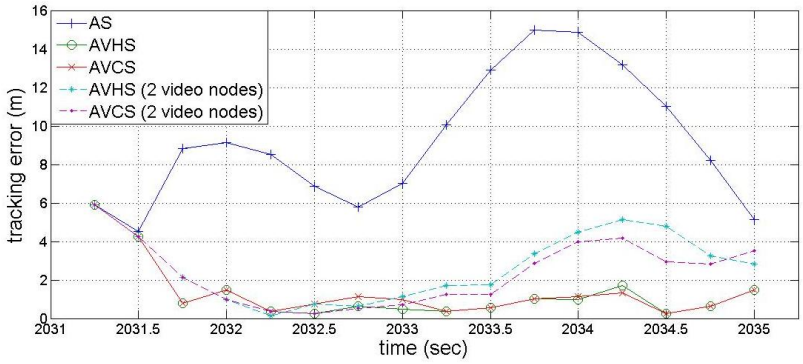
is improved. The figure shows tracking performance when using two and three video sensors for the centralized and hybrid fusion.

Figure 20 shows average tracking errors for all ten vehicle tracks for all target tracking approaches mentioned above. Figure 21(a) averages tracking errors for all the tracks to compare different tracking approaches. Audio and video modalities are able to track vehicles successfully, though they suffer from poor performance in the presence of high background noise and clutter. In general, audio sensors are able to track vehicles with good accuracy, but they suffer from high uncertainty and poor sensing range. Video tracking is not very robust in the presence of multiple targets and noise. As expected, fusing the two modalities consistently gives better performance. There are some cases where audio tracking performance is better than fusion. This is due to poor performance of video tracking.

Fusion based on the product of likelihood functions gives better performance but it is more vulnerable to sensor conflict and errors in sensor calibration. The weighted-sum approach is more robust to conflicts and sensor errors, but it suffers from high uncertainty. The centralized estimation



(a)



(b)

Figure 19: Tracking error (a) poor video tracking, (b) poor audio tracking

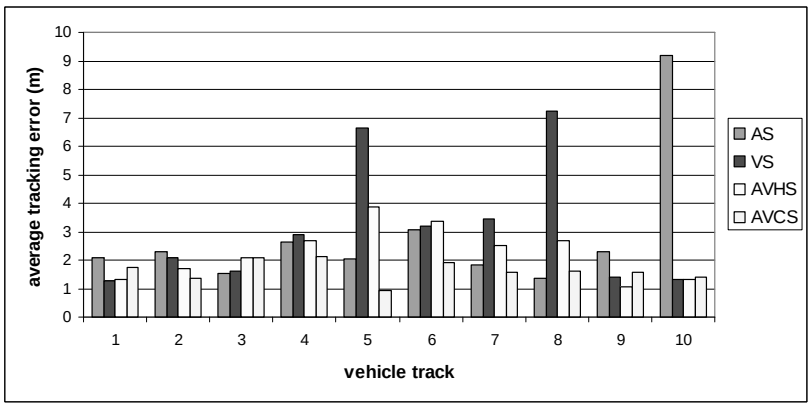
framework consistently performs better than the hybrid framework.

Figure 22 shows the determinant of the covariance for all tracks and all approaches. Figure 21(b) presents averages of covariance measure for all tracks to compare the performance of tracking approaches. Among modalities, video sensors have lower uncertainty than audio sensors. Among the fusion techniques, product fusion produces lower uncertainty, as expected. There was no definite comparison between the centralized and hybrid approach, though the latter seems to produce lower uncertainty in the case of weighted-sum fusion.

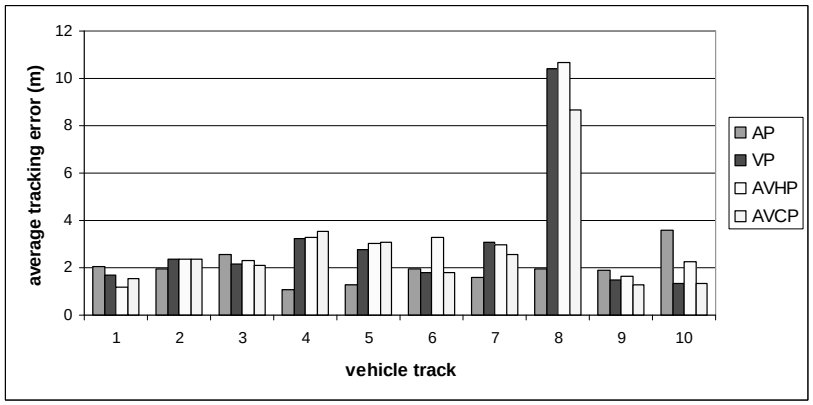
The average tracking error of 2 meters is reasonable considering the fact that a vehicle is not a point source, and the cell size used in fusion is 0.5 meters.

MCMCDA

The audio and video data gathered for target tracking based on sequential Bayesian estimation is reused to evaluate target tracking based on MCMCDA. For MCMCDA evaluation, we experiment with six different approaches. We use audio-only (A), video-only (V) and audio-video (AV) sensor



(a)



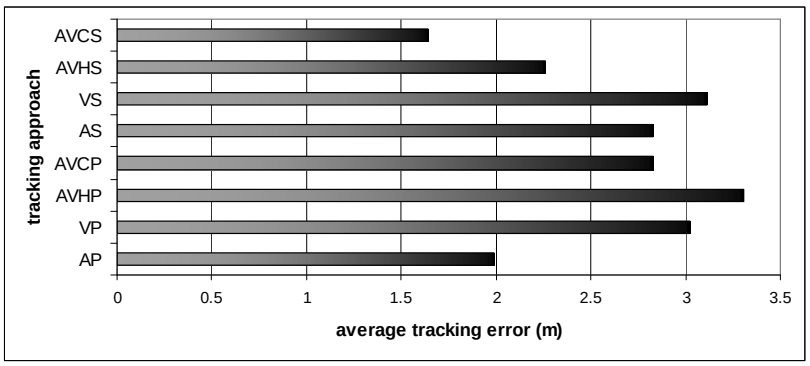
(b)

Figure 20: Tracking errors (a) weighted sum, (b) product

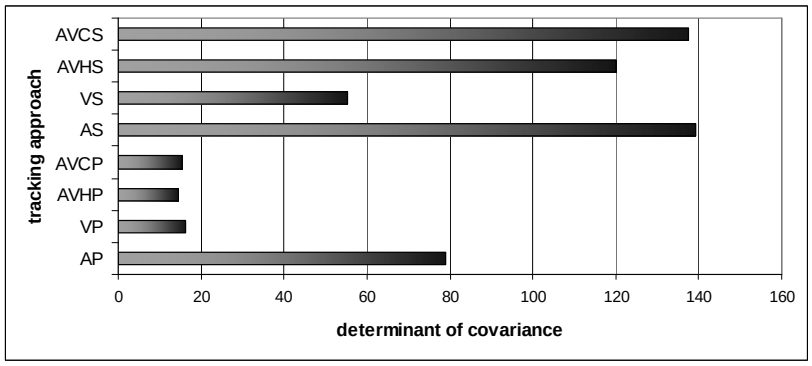
data for sensor fusion. For each of these data sets, the likelihood is computed either as the weighted-sum or product of the likelihood functions for individual sensors.

Single Target

We shortlist 9 vehicle tracks with a single target in the sensing region. The average duration of tracks is 3.75 sec with 2.75 sec minimum and 4.5 sec maximum. Figure 23 shows the target tracking result for two different representative vehicle tracks. The figure also shows the raw observations obtained from the multimodal sensor fusion and peak detection algorithms. Figure 24 shows average tracking errors for all vehicle tracks for the weighted-sum fusion and product fusion approaches. The missing bars indicate that the data association algorithm is not able to successfully estimate a track for the target. Figure 25 averages tracking errors for all the tracks to compare different tracking approaches. The figure also shows the comparison of the performance of tracking based on sequential Bayesian estimation to MCMCDA based tracking. The performance of MCMCDA is consistently better than



(a)



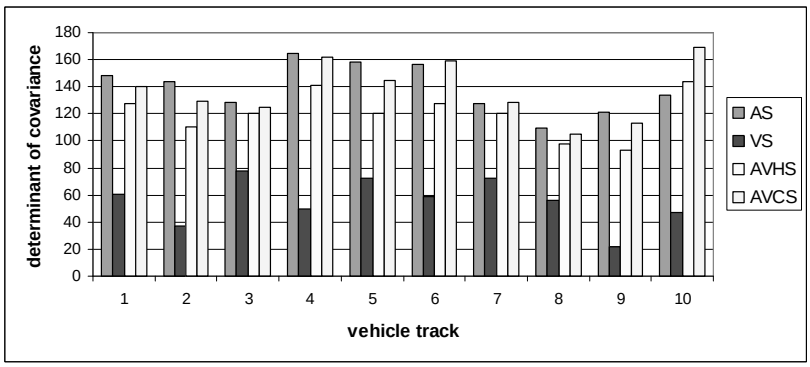
(b)

Figure 21: (a) Average tracking errors and (b) average of determinant of covariance for all tracks

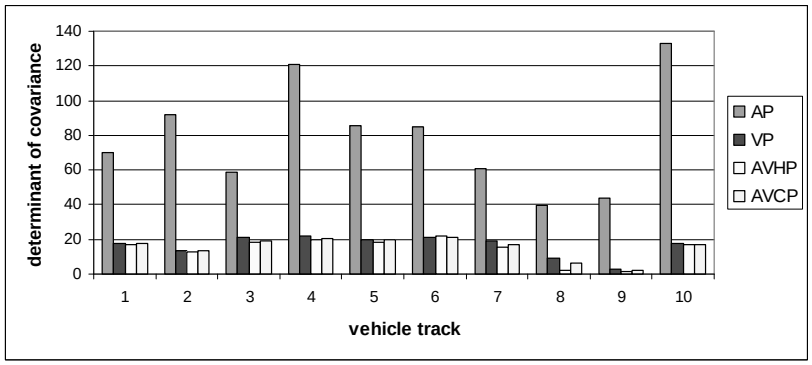
Table 8: Average tracking error and tracking success

		Average error (m)	Tracking success
Fusion	Summ	1.93	74%
	Prod	1.58	59%
Modality	Audio	1.49	89%
	Video	2.44	50%
	AV	1.34	61%

sequential Bayesian estimation. Table 8 compares average tracking errors and tracking success across likelihood fusion and sensor modality. Tracking success is defined as the percentage of correct tracks that the algorithm is successfully able to estimate. Table 9 shows the reduction in tracking error for audio-video fusion over audio-only and video-only approaches. For summation fusion, the audio-video fusion is able to reduce tracking error by an average of 0.26 m and 1.04 m for audio and video approaches, respectively. The audio-video fusion improves accuracy for 57% and 75% of the tracks for audio and video approaches, respectively. For the rest of the tracks, the tracking error either increased or remained same. Similar results are presented for product fusion in Table 9. In general, audio-video fusion improves over either audio or video or both approaches. Video cameras



(a)



(b)

Figure 22: Determinant of covariance (a) weighted sum, (b) product

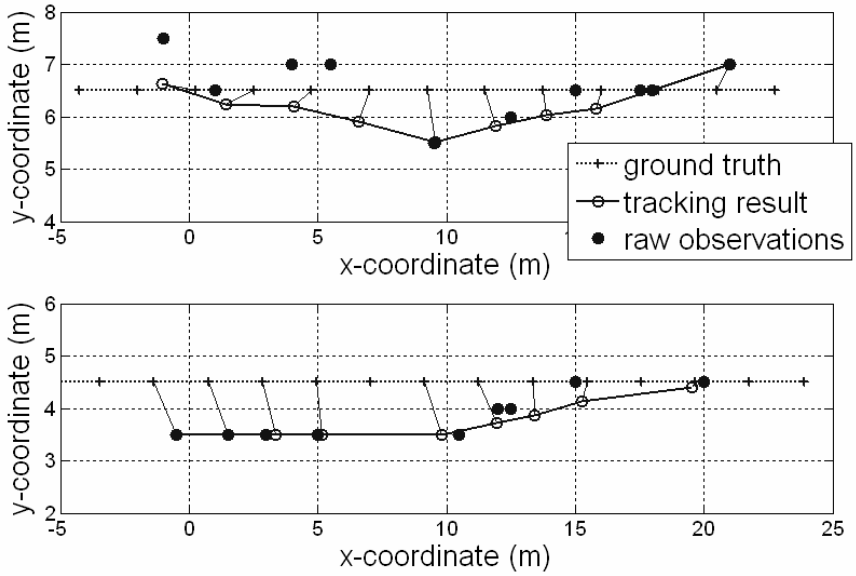
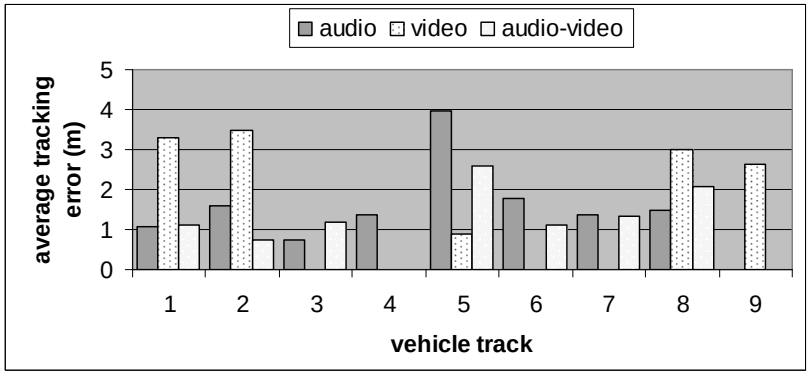
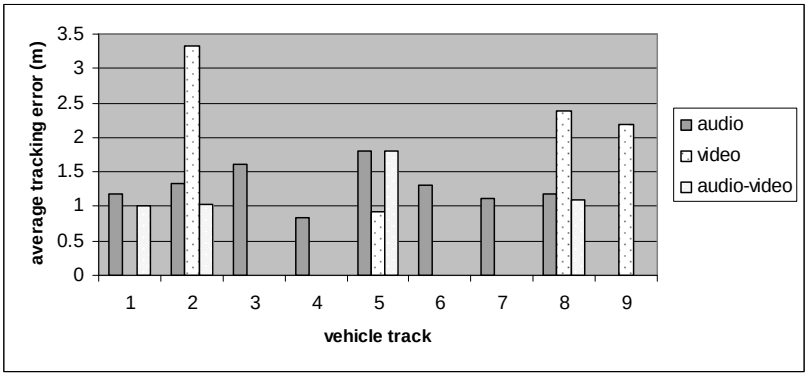


Figure 23: Target Tracking (a) no missed detection (b) with missed detections

were placed at an angle along the road to maximize coverage of the road. This makes video tracking very sensitive to camera calibration errors and camera placement. Also, an occasional obstruction



(a)



(b)

Figure 24: Tracking errors (a) weighted sum fusion, and (b) product fusion

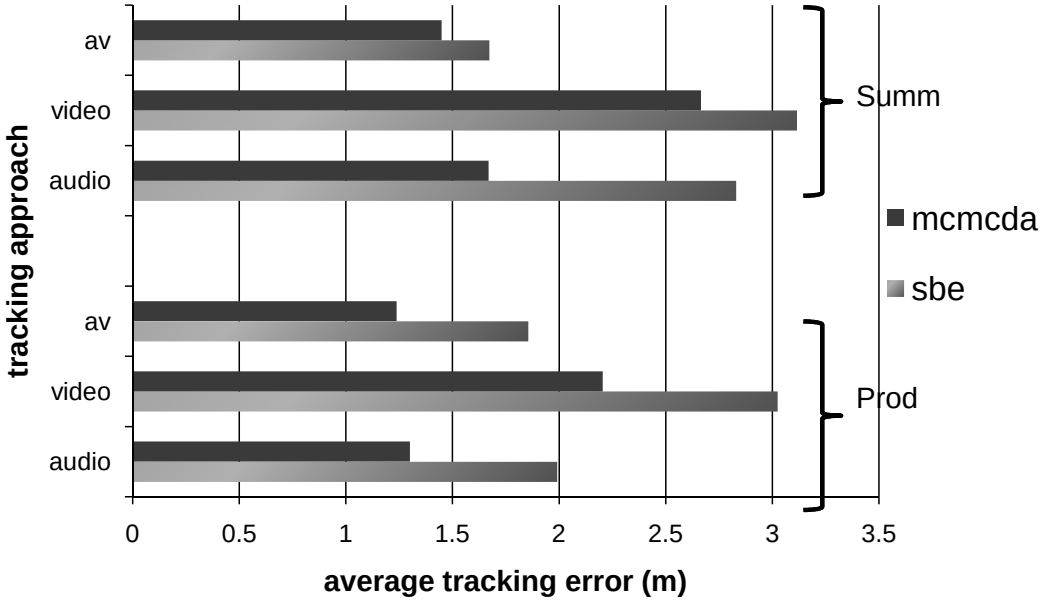


Figure 25: Average tracking errors for all estimated tracks. A comparison with sequential Bayesian estimation based tracking is also shown.

Table 9: Average reduction in tracking error for AV over audio and video-only for all estimated tracks

	Summ		Prod	
	Average error reduction (m)	Tracks improved	Average error reduction (m)	Tracks improved
Audio	0.26	57%	0.14	100%
Video	1.04	75%	0.90	67%

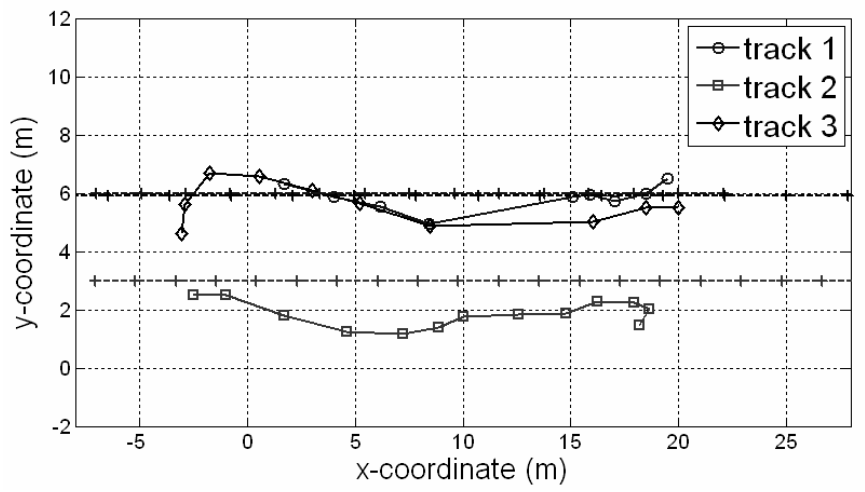
in front of a camera confused the tracking algorithm which took a while to recover. An accurate self-calibration technique, e.g. [148, 149], is desirable for better performance of a target tracking system.

Multiple Targets

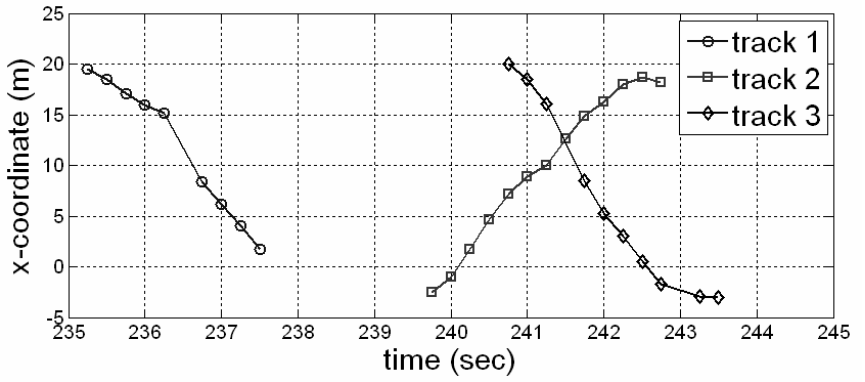
Many tracks with multiple moving vehicles in the sensing region were recorded during the experiment. Most of them have vehicles moving in the same direction. Only a few tracks include multiple vehicles crossing each other. Figure 26 shows the multiple target tracking result for three vehicles where two of them are crossing each other. Figure 26(a) shows the three tracks with the ground truth, while Figure 26(b) shows the x-coordinate of the tracks with time. The average tracking errors for the three tracks are 1.29m, 1.60m and 2.20m. Fig. 26 shows the result when only video data from the three video sensors is used. Multiple target tracking with audio data could not distinguish between targets when they cross each other. This is due to the fact that beamforming is done assuming acoustic signals are generated from a single source. Acoustic beamforming methods exist for detecting and estimating multiple targets [92].

Conclusions

We have developed a multimodal tracking system for an HSN consisting of audio and video sensors. We presented various approaches for multimodal sensor fusion and two approaches for target tracking, which are based on sequential Bayesian estimation and MCMCDA algorithm. We have evaluated the performance of the tracking system using an HSN of six mote-based audio sensors and three PC webcamera-based video sensors. We evaluated and compared the performance for both the tracking approaches. Time synchronization across the HSN allows the fusion of the sensor data. We have deployed the HSN and evaluated the performance by tracking moving vehicles in an uncontrolled urban environment. We have shown that, in general, fusion of audio and video data can improve the tracking performance. Currently, our system is not robust to multiple acoustic sources



(a)



(b)

Figure 26: Multiple target tracking (a) XY plot (b) X-coordinate with time

or multiple moving targets. An accurate self-calibration technique and robust audio and video sensing algorithms for multiple targets are required for better performance. A related challenge is sensor conflict that can degrade the performance of any fusion method and needs to be carefully considered. As in all sensor network applications, scalability is an important aspect that has to be considered as well.

CHAPTER IV

LOCALIZATION AND DISCRIMINATION OF MULTIPLE WIDEBAND/HARMONIC ACOUSTIC SOURCES

Introduction

Acoustic source localization is an important problem in many diverse applications such as military surveillance and reconnaissance, underwater acoustics, seismic remote sensing, and environmental monitoring [2, 150, 151]. Recently, innovative applications such as smart video-conferencing [131], audio-video sensor fusion and target tracking [9,50] have also been proposed to utilize source localization. Traditional acoustic source localization methods were developed for wired sensor networks [70]. In Wireless Sensor Networks (WSNs), collaborative source localization is used to estimate the positions of multiple sources by fusion of observations from multiple sensors. There are two broad classes of methods for collaborative source localization. The first class of approaches, where the estimation is done by fusion of the raw sampled signals, is called *signal-based* or *signal-level fusion*. The second class of approaches, where signal features are extracted from raw data at each sensor and the estimation is done by fusion of the extracted features, is called *feature-based* or *feature-level fusion*. The signal-level fusion methods are not suited for WSNs because they require transmission of the raw signal, which is costly due to limited bandwidth and power. On the other hand, the feature-level fusion methods are appropriate for WSNs due to their lower bandwidth and power requirements.

In this chapter, we present a feature-level fusion approach to collaborative localization and discrimination for multiple harmonic sources in WSNs. Source discrimination involves spatial discrimination where the goal is to separate the sources in space, and frequency discrimination where the goal is to separate the sources in frequency. We use beamforms and Power Spectral Densities (PSDs) as the signal features. The advantage of using the beamform over signal energy is that the beamform captures the angular variation of signal energy, which results in better localization resolution, and hence better spatial source discrimination. Assuming harmonic sources, the use of PSD as another signal feature allows frequency discrimination. Advances in sensor network hardware and FPGA integration has allowed us to implement real-time algorithms for computing such features. Furthermore, the communication bandwidth available in WSNs is sufficient to support wireless transmissions of such features [9].

We use a graphical model to formulate the problem, and employ Maximum Likelihood (ML) and Bayesian estimation for estimating the position of the sources as well as their fundamental and dominant harmonic frequencies. Directed graphical models, which are generalization of Bayesian networks, are directed graphs in which nodes represent random variables, the directed edges represent causality between random variables, and the (lack of) edges capture conditional independence of random variables [152]. We represent the unknown source locations and frequencies as hidden state variables, and the acoustic features as observable variables (or observed data). In directed graphical models, the edges are directed from the hidden state variables to the observed data. Directed graphical models require generative models that describe the observed data in terms of the process that generated them, and the hidden state variables. We present generative models for the beamform and PSD data that describe them in terms of a generative process and the unknown source locations and frequencies.

In our approach, the solution to the collaborative localization and discrimination problem is divided into two steps as source separation and source localization. The idea is to separate the sources in the frequency domain using the PSD data from the sensors, and then use the separated sources for localization and discrimination. We analytically show that source separation is independent of source localization, as long as the sources and the sensors are stationary. We develop a ML estimation method for source separation and use Bayesian estimation for localization. We use Markov Chain Monte Carlo (MCMC) methods, specifically Gibbs sampling and slice sampling [153] for implementing both ML and Bayesian estimation. The advantages of the two step approach instead of joint estimation are twofold. First, estimation in two steps has lower computational complexity than joint estimation. Second, the variances of the likelihood functions for source separation and source localization are significantly different. In context of Monte Carlo methods, joint estimation may cause slower convergence, and require a large number of samples.

We present simulation results for multiple source localization in a grid sensor network. We study three simulation scenarios where (1) we increase the number of sources, (2) we increase the average source SNR of two sources present in the sensing region, and (3) we increase the separation between the two sources. Our results show that as the separation between sources increases, the algorithm is able to achieve higher localization accuracy, comparable to single source localization. We present evaluation of the localization accuracy when the assumptions for the acoustic sources are relaxed; specifically the harmonic and omnidirectional source assumptions. The localization accuracy degrades gracefully when source *harmonicity* is decreased. We also present evaluation of the localization accuracy with PSD data compression. As expected, the localization accuracy

improves as more PSD data is available.

Finally, we implement the feature extraction algorithms on an FPGA chip onboard MICAz sensor nodes, and conduct outdoor experiments with real acoustic sources. Outdoor experimental results reinforce the simulation results. For smaller source separations the average localization error remains low but the algorithm is not able to disambiguate the two sources. For larger separations, the localization error is decreased.

Comparison with Related Work

Signal-based methods for acoustic source localization typically make use of Time Delay of Arrival (TDOA) and Direction of Arrival (DOA). An overview of theoretical aspects of TDOA-based acoustic source localization and beamforming is presented in [92], along with a localization algorithm based on ML estimation. For multiple acoustic sources, an Approximate Maximum Likelihood (AML) algorithm based on alternative projection method is also presented. An empirical study of collaborative acoustic source localization based on an implementation of AML is shown in [151].

Among feature-based methods, Energy-Based Localization (EBL) methods utilize signal energy as the features. Least-squares formulations for EBL have been presented in [83, 85]. A ML formulation with capability for multiple source localization is presented in [84]. They use a multiresolution search algorithm and an expectation-maximization (EM) like iterative algorithm for estimation. We use the beamform instead of the signal energy as the feature. The advantage is that the beamform captures the angular variation of signal energy, which results in better localization resolution, and hence better source discrimination.

The classical approaches to multiple target tracking include data association-based approaches such as Multiple Hypothesis Tracking (MHT) [39] and data association filters [5, 6]. These approaches use a set of exclusive and exhaustive hypotheses either associating measurements with the targets and clutter, called *target-oriented methods*, or associating targets with measurements, called *measurement-oriented methods*. Probabilities are computed for each hypothesis and the most probable hypotheses are used to compute target estimates. The number of hypotheses is combinatorial in the number of targets and measurements, as well as in time.

In data association-based approaches, the *measurements* are noise-corrupted sensor readings related to the state of a target, such as range and/or azimuth from a sensor, etc. The measurements are usually not raw data points, but rather the outputs of signal processing and detection subsystems [6]. The sensor model assumes that each measurement (or detection) corresponds to a single target

(i.e. each measurement originated either due to a single target or due to noise). Also, each target generates a single measurement and the measurement due to one target is *not* affected by the presence of other targets. In other words, a measurement for a target would be same regardless the presence of other targets in the scene. Due to these assumptions, the data-association based approaches are not able to model target interaction and mixed measurements, which results in unresolved targets [154]. The problem of mixed measurements and unresolved targets is even more significant in acoustic sensors because the acoustic source signals from multiple targets are additive.

Due to the fact that measurements are not raw data points, but the outputs of signal processing and detection subsystems, the data association-based tracking can be categorized as high-level (or decision-level) fusion. In high-level fusion, *discriminating* information is lost, especially for sensing models where raw data is a mixture of the signals originating from multiple targets. Alternate approaches can utilize low-level (or signal-level) fusion, or medium-level (or feature-level) fusion. Signal-level fusion methods are not suitable for WSNs because they require transmission of the raw signal, which is costly due to limited bandwidth and power. Feature-level fusion methods include signal processing and feature extraction algorithms that *extract* informative features from the raw data, which are communicated to the sensor fusion node. These methods require explicit target interaction models and observation models that describe the generation of features. Feature-level fusion methods are appropriate for WSNs due to their lower bandwidth and power requirements, at the same time, they maintain sufficient *discriminating* information.

More recent approaches to multitarget tracking include joint tracking using Bayesian inference where the quantity of interest is the joint multitarget state, which is the concatenation of individual target states [45]. Joint Bayesian inference has the advantages of providing a recursive solution with arbitrary target dynamic models and observation models. Several approaches based on Bayesian estimation [48, 49] and graphical models [50, 51] have been also proposed. Sequential Monte Carlo (SMC) implementations for Bayesian estimation and multitarget tracking are presented in [46,47]. A Bayesian approach for tracking the DOA of multiple targets using a passive sensor array is presented in [48]. A Bayesian approach for multiple target detection and tracking, and particle filter-based algorithms are proposed in [49]. A graphical model based approach for audiovisual object tracking that fuses audio and video data from a microphone pair and a camera is presented in [50]. A graphical model formulation for self-localization of sensor networks using a technique called nonparametric belief propagation is presented in [51]. The feature-based localization approach in this paper also uses a graphical model that models multiple target interaction and mixed measurements through generative models for the acoustic features. We use MCMC methods for source separation and

localization, and we analytically show that source separation is independent of source localization.

Acoustic Source Localization & Discrimination

We consider a WSN of K acoustic sensors in planar field and M far-field stationary acoustic sources coplanar with the sensor network. The objective is to estimate the 2D position of all the sources and discriminate them, both in frequency and space, using the received acoustic signals at the sensors. To localize the acoustic sources, we assume each sensor node is equipped with an array of N_{mic} microphones. The acoustic wave front incident on the microphone array is assumed to be planar for far-field sources. Each sensor receives an acoustic signal that is a combination of source signals. The sensors run signal processing algorithms to compute the beamform and PSD features. In the rest of the section, we describe the source assumptions, source model, signal propagation model and signal processing algorithms for feature extraction.

Acoustic Source Model

The main assumptions made for the acoustic sources are: (1) omnidirectional and stationary point sources, (2) emitting stationary signals, (3) the source signals are harmonic, and (4) the cross-correlation between two source signals is negligible compared to the signal autocorrelations. Harmonic signals have a fundamental frequency, also called the first harmonic, and other higher-order harmonic frequencies that are multiples of the fundamental frequency. The energy of the signal is contained in these harmonic frequencies only. The harmonic source assumption is satisfied by a wide variety of acoustic sources [155]. In general, any acoustic signal originating due to the vibrations from rotating machinery will have a harmonic structure. The state for the m^{th} acoustic source is given by: (1) the position $\mathbf{x}^{(m)} = \left[x^{(m)}, y^{(m)}\right]^T$, (2) the fundamental frequency $\omega_f^{(m)}$, and (3) the energies in the harmonic frequencies $\psi^{(m)} = \left[\psi_1^{(m)}, \psi_2^{(m)}, \cdots, \psi_H^{(m)}\right]^T$ where H is the number of harmonic frequencies.

In practice, some of the assumptions may not be always true. For example, the engine sound of a vehicle may not be omnidirectional and will be biased toward the side closer to the engine. The physical size of the acoustic source may be too large to be adequately modeled as a point source for sensors very close to the source. In an outdoor environment, strong background noise, including wind gusts, may be encountered during operation. Perhaps the most restrictive assumption is that the source signals are harmonic. In addition to the harmonic components, the engine sound signal may contain other frequency components, which when not accounted for, may cause localization to

deteriorate. In Section IV, we present empirical evaluation of localization accuracy when the source assumptions are relaxed, specifically the harmonic and omnidirectional sources assumptions, and it is shown that the localization accuracy degrades gracefully.

Signal Propagation Model

The intensity of an acoustic signal emitted omni-directionally from a point sound source attenuates at a rate that is inversely proportional to the distance from the source [84]. The discrete signal received at the p^{th} microphone on a particular microphone array is given by

$$r_p[n] = \sum_{m=1}^{M} \frac{d_0}{\parallel \mathbf{x}_p - \mathbf{x}^{(m)} \parallel} s^{(m)}[n - \tau_p^{(m)}] + w_p[n] \tag{32}$$

for samples $n = 1, \cdots, L$, where L is the length of the acoustic signal, M is the number of sources, $w_p[n]$ is white Gaussian measurement noise such that $w_p[n] \sim \mathcal{N}(0, \sigma_w^2)$, $s^{(m)}[n]$ is the intensity of the m^{th} source measured at a reference distance d_0 from that source, $\tau_p^{(m)}$ is the propagation delay of the acoustic signal from the m^{th} source to the p^{th} microphone, and $\mathbf{x}_p$ denote the microphone position. We define the multiplicative term in Equation (32) as the attenuation factor, $\lambda_p^{(m)}$, given by

$$\lambda_p^{(m)} = \frac{d_0}{\parallel \mathbf{x}_p - \mathbf{x}^{(m)} \parallel}.$$

Acoustic Features

The two acoustic features used for feature-level fusion are beamform and PSD. The details of the feature extraction algorithms are given below. Details related to an FPGA implementation are given in Section IV.

Beamform

Beamforming is a signal processing algorithm for Direction-of-Arrival (DOA) estimation of a signal source using an array of microphone [92]. In a typical delay-and-sum single source beamformer, the 2D sensing region is discretized into directions, or *beams* as $\alpha = i\frac{2\pi}{Q}$, where $i = 0, \cdots, Q-1$ and Q is the number of beams. The beamformer computes the energy of the reconstructed signal at each beam direction. This is achieved by delaying and summing the individual microphone signals. The

delayed and summed signal is given by

$$r[n] = \sum_{p=1}^{N_{mic}} r_p[n + t_{pq}(\alpha)] \tag{33}$$

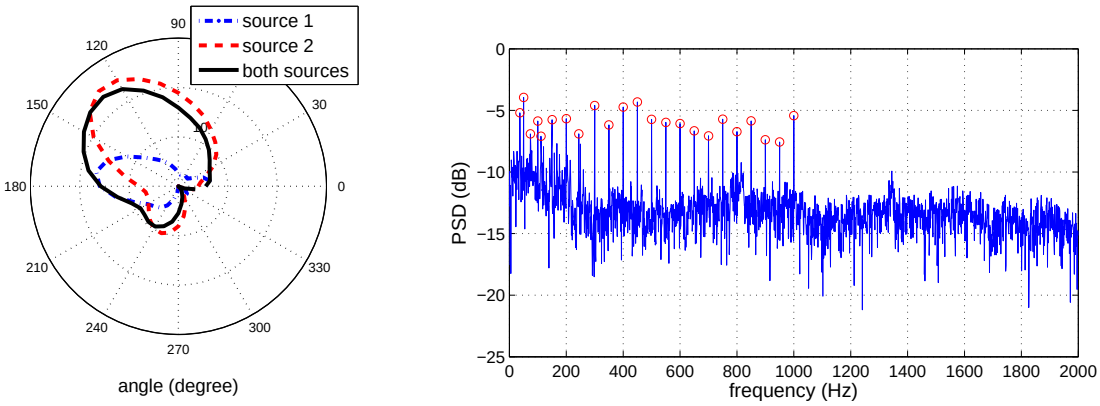
where α is the beam angle, $r_p[\cdot]$ is the received signal at the p^{th} microphone, q is the index of a reference microphone, N_{mic} is the number of microphones, and $t_{pq}(\alpha)$ is the relative time delay for the p^{th} microphone with respect to the reference microphone q, given by

$$t_{pq}(\alpha) = d_{pq} cos(\alpha - \beta_{pq}) f_s / C$$

where d_{pq} and β_{pq} are the distance and angle between the p^{th} and q^{th} microphones, and f_s and C are signal sampling rate and speed of sound, respectively. The beam energy is given by

$$B(\alpha) = \sum_{n=1}^{L} r[n]^2 = \sum_{n=1}^{L} \left[\sum_{p=1}^{N_{mic}} r_p[n + t_{pq}(\alpha)] \right]^2$$

Beam energies are computed for each of the beams, and are collectively called the *beamform*. The beam with maximum energy indicates the DOA of the acoustic source. In case of multiple sources, there might be multiple peaks where the maximum peak would indicate the DOA of the highest energy source. Figure 27(a) shows a beamform for two acoustic sources.



(a) (b)

Figure 27: (a) Acoustic Beamforms; The beamforms for single sources clearly show peaks at the source location but the beamform when both sources are present does not show two peaks. (b) Power spectral density (PSD); the highest PSD values are shown as empty circles. The PSD is compactly represented as pairs of the highest PSD values and corresponding frequencies.

Acoustic PSD

PSD estimation is a signal processing algorithm for estimating the spectrum of the received acoustic signal, which describes how the power of the signal is distributed with frequency [156]. We compute the PSD as the magnitude of the Discrete Fourier Transform (DFT) of the signal

$$P(\omega) = Y(\omega) \cdot \overline{Y(\omega)}$$

where $Y(\omega) = \mathbb{FFT}(r, N_{FFT})$ is the DFT of the signal $r[n]$, N_{FFT} is the length of the transform, and $\overline{Y(\omega)}$ is the complex conjugate of the transform. For real-valued signals, the PSD is real and symmetric; hence we need to store only half of the spectral density. In our implementation, we compress the PSD data for wireless transmission by storing and sending the frequency-power pairs, (ω_j, ψ_j), for N_{PSD} frequencies with the highest power values. Figure 27(b) shows an acoustic PSD estimate for a received signal when two harmonic sources are present.

Graphical Model Overview

Probabilistic graphical models provide a systematic methodology to handle uncertainty and complexity in real systems. They are playing an increasingly important role in the design and analysis of machine learning, bio-informatics, audio processing and image processing algorithms [152, 157]. In a probabilistic graphical model, each node represents a random variable (or a group of random variables), and the edges express probabilistic relationships between these variables. The lack of an edge between nodes represents conditional independence. The graph structure captures the way in which the joint distribution over all the random variables can be factorized into a product of local function defined on nodes and their immediate neighbors (a subset of random variables). Hence, problems involving computation of quantities related to the joint probability model can be profitably cast within a graph theoretic framework. In particular, the underlying structure of the graph is closely related to the underlying computational complexity of an inference problem [152].

The graphical models can be categorized as undirected and directed graphical models. Undirected graphical models capture correlation between random variables, while directed graphical models capture causality between variables. In directed graphical models, also called Bayesian networks, the edges are directed from hidden variables to observed variables. The directed graphical models require *generative models* that describe the observed data in terms of the process that generated them, and the hidden variables.

The model shown in Figure 28 is an example of a directed graphical model. We use this graphical model to formulate the source separation and localization of multiple sources problem. We use plate notation to represent the repetition of the random variables [158]. The plate index, M on the upper plate represents repetition of the hidden variables for M sources, while the index, K on the lower place represents repetition of the observed variables for K sensors. In the figure, the nodes

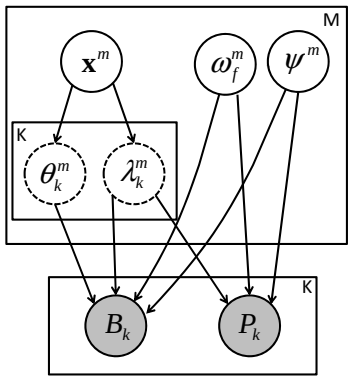


Figure 28: Graphical model for acoustic source localization.

with clear background denote hidden state variables; $\mathbf{x}^{(m)}, \omega_f^{(m)}, \psi^{(m)}$ denote source position, fundamental frequency and harmonic energies for the m^{th} source, respectively. The nodes with shaded backgrounds denote observed variables; B_k and P_k denote the beamform and the PSD received at k^{th} sensor, respectively. Finally, the nodes with dotted outlines denote functions of random variables, or *auxiliary random variables* that capture the functional dependence of the observed variables on the hidden variables. The two auxiliary variables shown in the graphical model are the angle $\theta_k^{(m)}$ and the attenuation factor $\lambda_k^{(m)}$. These variables will be utilized in the generative models for the observed variables.

We perform multiple source localization and discrimination in two steps. First, we use the PSD data only to separate the sources, which in our problem, refers to separating the PSDs of the sources. For harmonic sources, estimation of fundamental frequencies is sufficient for source separation, because all the dominant frequencies in the signal are multiples of the fundamental frequency. An ML estimation method is proposed in the next section for fundamental frequency estimation. It is shown that the ML estimate is independent of the source location, which is intuitive because the dominant frequencies in the source signal are independent of the source location, as long as the source and the sensor are stationary. In the second step, we use the beamform data and the *separated source PSDs* to localize all the sources.

We chose to perform multiple source localization in two steps instead of joint estimation because

of the following two reasons. First, estimation in two steps has lower computational complexity than joint estimation. In the context of Monte Carlo methods, let the number of samples needed be exponential in state dimension, $N \propto D^{\alpha}$, where $D \geq 1$ is the state dimension and $\alpha \geq 1$ is the exponential factor. The joint estimation of two state variables with dimensionality D_1 and D_2 would require $N_{\text{joint}} \propto (D_1 + D_2)^{\alpha}$ samples, while separate estimation would require $N_{\text{separate}} \propto D_1^{\alpha} + D_2^{\alpha}$ samples. For $D_1, D_2, \alpha \geq 1$, one can show that $N_{\text{joint}} \geq N_{\text{separate}}$. Second, the variances of the likelihood functions for source separation and localization are significantly different. In Monte Carlo context, joint estimation may cause slower convergence, and require a large number of samples. Moreover, as mentioned earlier, the ML estimate for source fundamental frequencies is independent of the source locations, further supporting the two step process of source separation and source localization.

Source Separation

In this section, we present the first step of our approach, which is source separation and frequency discrimination. We use the PSD data only to separate the sources.We propose an ML estimation method for source separation. In ML estimation, we need the data likelihood function for the PSD data, which requires a generative model. We begin by presenting the generative model and the likelihood function for the PSD data. We also present a result showing that the likelihood function at the ML estimate of harmonic energies is independent of the source positions. Finally, we present the ML estimate for source fundamental frequency.

Generative Model for PSD Data

For harmonic sources, the PSD can be given by

$$P_s^{(m)}(\omega) = \sum_{h=1}^{H} \psi_h^{(m)} \delta(\omega - h\omega_f^{(m)}) \tag{34}$$

where $m = 1, \cdots, M$ are source indices, ω is the frequency, $\omega_f^{(m)}$ is the fundamental frequency, $\psi_h^{(m)}$ is the energy in the h^{th} harmonic, H is the number of harmonics, and $\delta(\cdot)$ is the Dirac delta function. Using Equation (34), we derive a generative model for the PSD data received at a sensor node. The following proposition states the generative model for the PSD data.

Proposition 1. *For an arbitrary number of acoustic source signals, the power spectral density of the signal received at a sensor is given by*

$$\mathbb{P}(\omega) = \sum_{m=1}^{M} \sum_{n=1}^{M} \lambda^{(m)} \lambda^{(n)} \left(P_s^{(m)}(\omega) P_s^{(n)}(\omega) \right)^{\frac{1}{2}} \cos(\Phi^{(m)}(\omega) - \Phi^{(n)}(\omega)) \tag{35}$$

where M is the number of sources, $\lambda^{(m)}$ is the attenuation factor, and $\Phi^{(m)}(\omega)$ is the phase spectral density, which is given by

$$\Phi^{(m)}(\omega) = \phi^{(m)} - \parallel \mathbf{x}^{(m)} - \mathbf{x}_s \parallel \omega / C$$

where $\phi^{(m)}$ is the phase of the source signal, $\mathbf{x}^{(m)}$ and $\mathbf{x}_s$ are the positions of the source and the sensor, respectively.

Proof. Consider M sources emitting source signals $s_m[n]$, for $m = 1, 2, \cdots, M$. Using Equation (32), the received signal at a microphone is given by

$$y[n] = \sum_{m=1}^{M} \lambda_m s_m[n - \tau_m] + w[n]$$

where τ_m is the propagation delay, and λ_m is the attenuation factor. Taking FFT of the received signal, we have

$$\begin{aligned} Y(\omega) &= \mathbb{FFT}\big(y[n]\big) \\ &= \mathbb{FFT}\left(\sum_{m=1}^{M} \lambda_m s_m[n - \tau_m] + w[n] \right) \\ &= \sum_{m=1}^{M} \lambda_m \mathbb{FFT}\big(s_m[n - \tau_m]\big) + \mathbb{FFT}\big(w[n]\big) \\ &= \sum_{m=1}^{M} \lambda_m S_m(\omega) + W(\omega) \end{aligned}$$

where $S_m(\omega)$ is the Fourier transform of m^{th} source signal, and $W(\omega)$ is the Fourier transform of noise. The power spectral density (PSD) of a signal is given by

$$\begin{aligned}
P(\omega) &= Y(\omega) \cdot \overline{Y(\omega)} \\
&= \left(\sum_m \lambda_m S_m(\omega) + W(\omega) \right) \cdot \overline{\left(\sum_m \lambda_m S_m(\omega) + W(\omega) \right)} \\
&= \left(\sum_m \lambda_m S_m(\omega) + W(\omega) \right) \cdot \left(\sum_m \lambda_m \overline{S_m(\omega)} + \overline{W(\omega)} \right) \\
&= \sum_m \sum_n \lambda_m \lambda_m S_m(\omega) \cdot \overline{S_n(\omega)} + \sum_m \lambda_m S_m(\omega) \cdot \overline{W(\omega)} \\
&\quad + \sum_m \lambda_m \overline{S_m(\omega)} \cdot W(\omega) + W(\omega) \cdot \overline{W(\omega)}.
\end{aligned} \tag{36}$$

The Fourier transform $S(\omega)$ can also be written in terms of the PSD ($P(\omega)$) and phase spectral density ($\Phi(\omega)$), as

$$S(\omega) = P(\omega)^{1/2} e^{+i\Phi(\omega)}$$

which gives

$$\begin{aligned}
S_m(\omega) \cdot \overline{S_n(\omega)} &= \big(P_m(\omega) P_n(\omega)\big)^{1/2} e^{+i(\Phi_m(\omega) - \Phi_n(\omega))} \\
S_m(\omega) \cdot \overline{W(\omega)} &= \big(P_m(\omega) P_\eta(\omega)\big)^{1/2} e^{+i(\Phi_m(\omega) - \Phi_\eta(\omega))} \\
\overline{S_m(\omega)} \cdot W(\omega) &= \big(P_m(\omega) P_\eta(\omega)\big)^{1/2} e^{-i(\Phi_m(\omega) - \Phi_\eta(\omega))} \\
W(\omega) \cdot \overline{W(\omega)} &= \big(P_\eta(\omega) P_\eta(\omega)\big)^{1/2} = P_\eta(\omega)
\end{aligned}$$

where $P_\eta(\omega)$ and $\Phi_\eta(\omega)$ are PSD and phase spectral density of the noise signal. Rewriting Equation (36), we have

$$\begin{aligned}
P(\omega) = &\sum_m \sum_n \lambda_m \lambda_m \big(P_m(\omega) P_n(\omega)\big)^{1/2} e^{+i(\Phi_m(\omega) - \Phi_n(\omega))} + \sum_m \lambda_m \big(P_m(\omega) P_\eta(\omega)\big)^{1/2} e^{+i(\Phi_m(\omega) - \Phi_\eta(\omega))} \\
&+ \sum_m \lambda_m \big(P_m(\omega) P_\eta(\omega)\big)^{1/2} e^{-i(\Phi_m(\omega) - \Phi_\eta(\omega))} + P_\eta(\omega).
\end{aligned}$$

Assuming that PSD for noise is negligible compared to actual source signals, we have

$$P(\omega) = \sum_m \sum_n \lambda_m \lambda_m \big(P_m(\omega) P_n(\omega)\big)^{1/2} e^{+i(\Phi_m(\omega) - \Phi_n(\omega))}. \tag{37}$$

We know that PSD of real-valued signals is real-symmetric, hence the imaginary component in Equation (37) is zero. Hence, we have

$$P(\omega) = \sum_m \sum_n \lambda_m \lambda_m \big(P_m(\omega) P_n(\omega)\big)^{1/2} \cos\left(\Phi_m(\omega) - \Phi_n(\omega)\right).$$

□

The expression in Equation (35) can be approximated using the following observation. Since we do not maintain the phase of the signal in the source model (see Section IV), we assume all the phases to be normally distributed with equal mean. The expected value of the cosine of the difference of two normally distributed angles is one, i.e. $E[cos(\Phi_i - \Phi_j)] = 1$. Therefore, Equation (35) can be approximated as

$$\mathbb{P}(\omega) \approx \left[\sum_{m=1}^{M} \lambda^{(m)} \left(P^{(m)}(\omega)\right)^{1/2}\right]^2 \tag{38}$$

Data Likelihood & ML Estimate

Using Equation (38), the negative log-likelihood for PSD data at the kth sensor is defined as

$$\ell_k(\Omega_f, \Psi, \mathbf{X}) = \frac{1}{\sigma_P^2} \int_\omega \parallel P_k(\omega) - \mathbb{P}_k(\omega) \parallel^2 d\omega \tag{39}$$

where $P_k(\omega)$ is the observed PSD at the kth sensor, and

$$\begin{aligned}
\Omega_f &= \left[\omega_f^{(1)}, \cdots, \omega_f^{(M)}\right]^T \\
\Psi &= \left[\boldsymbol{\psi}^{(1)}, \cdots, \boldsymbol{\psi}^{(M)}\right]^T \\
\boldsymbol{\psi}^{(m)} &= \left[\psi_1^{(m)}, \cdots, \psi_H^{(m)}\right]^T \\
\mathbf{X} &= \left[\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(M)}\right]^T
\end{aligned}$$

Assuming a discrete frequency variable, Equation (39) can be rewritten as

$$\ell_k(\Omega_f, \Psi, \mathbf{X}) = \frac{1}{\sigma_P^2} \sum_{\omega_j} \parallel P_k(\omega_j) - \mathbb{P}_k(\omega_j) \parallel^2$$

The likelihood function for the PSD data at the ML estimate of harmonic energies is independent of the source positions, as stated in the following proposition.

Proposition 2. *The likelihood for the PSD data at the ML estimate of harmonic energies is the likelihood that is given by*

$$\ell_k(\Omega_f, \Psi^{ML}, \mathbf{X}) = \ell'_k(\Omega_f, \mathbf{X}) = \sum_{\omega_j \notin \mathbb{H}(\Omega_f)} \left(P_k(\omega_j)\right)^2 = \ell'_k(\Omega_f) \tag{40}$$

where $\mathbb{H}$ *is the set of all harmonic frequencies for all sources*

$$\mathbb{H}(\Omega_f) = \bigcup_m \left[\omega_f^{(m)}, 2\omega_f^{(m)}, \cdots\right]^T$$

and, the likelihood is a function of the source fundamental frequencies only, and is independent of the source positions.

Proof. The maximum likelihood estimate of $[\Omega_f, \Psi, \mathbf{X}]^T$ can be obtained by minimizing $\ell_k(\Omega_f, \Psi, X)$

$$\frac{\partial}{\partial \psi_h^{(m)}} \ell_k(\Omega_f, \Psi, \mathbf{X}) = 0$$

which leads to the following

$$P_k(h\omega_f^{(m)}) = \mathbb{P}_k(h\omega_f^{(m)}) = \left(\sum_j^M \lambda_k^{(j)} {\psi_{h_j}^{(j)}}^{1/2}\right)^2 \tag{41}$$

where

$$\psi_{h_j}^{(j)} = \begin{cases} > 0 & \text{if } h_j = h\omega_f^{(m)}/\omega_f^{(j)} \in \mathbb{Z}. \\ 0 & \text{otherwise.} \end{cases}$$

If the frequency $h\omega_f^{(m)}$ is *shared* by M' sources (or the number of nonzero $\psi_{h_j}^{(j)}$ is M'), then Equation (41) becomes

$$P_k(h\omega_f^{(m)}) = \left(\sum_j^{M'} \lambda_k^{(j)} {\psi_{h_j}^{(j)}}^{1/2}\right)^2$$

If we assume the energy contribution of all the sources to be same, i.e. $\lambda_k^{(j)} {\psi_{h_j}^{(j)}}^{1/2} = \bar{\psi}_h$, for $j = 1, \cdots, M'$, we have

$$P_k(h\omega_f^{(m)}) = \left(M'\bar{\psi}_h\right)^2 = {M'}^2 {\bar{\psi}_h}^2 = {M'}^2 {\lambda_k^{(m)}}^2 \psi_{h_m}^{(m)} \tag{42}$$

rearranging Equation (42), we have

$$\hat{\psi_h^{(m)}}^{ML} = \frac{P_k(h\omega_f^{(m)})}{M'^2 {\lambda_k^{(m)}}^2}. \tag{43}$$

Substituting the ML estimate for the energies (Equation (43)) in the negative log-likelihood (Equation (40)), we have a modified negative log-likelihood

$$\begin{aligned}\ell_k(\Omega_f, \hat{\Psi}^{ML}, \mathbf{X}) &= \ell'_k(\Omega_f, \mathbf{X}) \\ &= \sum_{\omega_j \notin \mathbb{H}(\Omega_f)} \parallel P(\omega_j) - \mathbb{P}(\omega_j) \parallel^2 \\ &\quad + \sum_{\omega_j \in \mathbb{H}(\Omega_f)} \parallel P(\omega_j) - \mathbb{P}(\omega_j) \parallel^2 \end{aligned} \tag{44}$$

where $\mathbb{H}$ is the harmonic set, which is the set of all harmonic frequencies for all sources

$$\mathbb{H}(\Omega_f) = \bigcup_m \left[\omega_f^{(m)}, 2\omega_f^{(m)}, \cdots\right]^T$$

The value of generative model $\mathbb{P}$ at the frequencies *in* the harmonic set is exactly equal to the observed PSD, hence the second term in Equation (44) goes to zero. On the other hand, the value of generative model $\mathbb{P}$ at the frequencies *not in* the harmonic set is zero, hence

$$\ell'_k(\Omega_f, \mathbf{X}) = \sum_{\omega_j \notin \mathbb{H}(\Omega_f)} \parallel P(\omega_j) - \mathbb{P}(\omega_j) \parallel^2 = \sum_{\omega_j \notin \mathbb{H}(\Omega_f)} \left(P_k(\omega_j)\right)^2. \tag{45}$$

Equation (45) is the negative log-likelihood with the constraint of Equation (43) imposed. Equation (45) implies that the modified likelihood at the ML estimate of energies is independent of the source locations

$$\ell_k(\Omega_f, \Psi^{ML}, \mathbf{X}) = \ell'_k(\Omega_f, \mathbf{X}) = \sum_{\omega_j \notin \mathbb{H}(\Omega_f)} \left(P_k(\omega_j)\right)^2 = \ell'_k(\Omega_f)$$

□

Hence, according to Proposition 2, source separation can be performed independent of source localization. The full negative log-likelihood for all sensors, $\ell'(\Omega_f)$ is defined as

$$\ell'(\Omega_f) = \frac{1}{K} \sum_{k=1}^{K} \ell'_k(\Omega_f)$$

Thus, the ML estimation of the fundamental frequencies can be obtained by minimizing $\ell'(\Omega_f)$

$$\hat{\Omega}_f^{ML} = \arg\min_{\Omega_f} \ell'(\Omega_f) = \arg\min_{\Omega_f} \frac{1}{K} \sum_{k=1}^{K} \sum_{\omega_j \notin \mathbb{H}(\Omega_f)} (P_k(\omega_j))^2 \tag{46}$$

Note that Equation (46) is *not* an explicit expression for Ω_f since the set $\mathbb{H}$ is a function of Ω_f on the righthand side of this equation. This motivates the use of an iterative method for ML estimation. We use a Monte Carlo method described in Section IV.

Source Localization

Source localization is performed by Bayesian estimation in the graphical model shown in Figure 28, and taking the *maximum a-posteriori* (MAP) estimate of the source positions. The posterior, $p(\mathbf{X}|\mathbf{B})$ of the source positions at the ML estimates for source fundamental frequencies and harmonic energies given the beamform data

$$p(\mathbf{X}|\mathbf{B}) \propto \prod_{k=1}^{K} p(B_k|\mathbf{X}, \hat{\Omega}_f^{ML}, \hat{\Psi}^{ML}) p(\mathbf{X})$$

where $p(B_k|\mathbf{X}, \hat{\Omega}_f^{ML}, \hat{\Psi}^{ML})$ is the likelihood function for beamform data, $\mathbf{X}$ represent joint state for all sources, and $\mathbf{B}$ represent the beamforms for all sensors. The likelihood function requires a generative model for the beamform data. In this section, we present the generative model and three intermediate results pertaining to the model. Finally, we present the likelihood and MAP estimation.

Generative Model for Beamform

We start by developing a generative model for a beamform for a two-microphone array, single-source case (Proposition 3). We will show that the beamform for an arbitrary microphone array (Proposition 4) and an arbitrary number of sources (Proposition 5) can be composed from the simple two-microphone array, single-source case.

Proposition 3. *Consider a microphone pair separated by distance d and the angle between the x-axis and the line joining the microphones is β. For an acoustic source at angle θ and range r with power spectral density $P(\omega)$, the beamform B at the microphone pair is given by*

$$B(\alpha) = 2\lambda^2(R_{ss}(0) + R_{ss}(\kappa_\alpha)) + 2R_\eta(0) \tag{47}$$

where $R_{ss}(\tau) = \mathbb{FFT}^{-1}(P(\omega))$ *for* $\tau \in [-\infty, +\infty]$ *is the autocorrelation of the source signal,* $R_{ss}(0)$ *is the signal energy,* $R_{\eta}(0)$ *is the noise energy,* λ *is the attenuation factor, and* $\kappa_{\alpha} = d(\cos(\alpha - \beta) - \cos(\theta - \beta))f_s/C$*, where* $\alpha \in [0, 2\pi]$ *is the beam angle,* f_s *and* C *are sampling frequency and speed of sound, respectively.*

Proof. Consider a source present at an angle θ emitting a source signal $s[n]$. Using Equation (32), the received signals at the microphones are given by

$$r_p[n] = \lambda_p s[n - \tau_p] + w_p[n]$$

for $p = 1, 2$, where τ_p is the propagation delay, and λ_p is the attenuation factor. For far-field case, the distances between the source and the closely-spaced microphones will be approximately same for all microphones, hence $\lambda_1 \approx \lambda_2 = \lambda$.

Using Equation (33), the composite microphone signal for the beam angle α is given by

$$\begin{aligned} r[n] &= r_1[n] + r_2[n + t_{12}(\alpha)] \\ &= \lambda s[n - \tau_1] + \lambda s[n + t_{12}(\alpha) - \tau_2] + w_1[n] + w_2[n + t_{12}(\alpha)] \end{aligned}$$

where $t_{12}(\alpha) = t_2(\alpha) - t_1(\alpha) = d\cos(\alpha - \beta)f_s/C$ is relative sample delay. The beam energy is given by

$$\begin{aligned} B(\alpha) &= \sum_n r[n]^2 \\ &= \sum_n \left(\lambda s[n - \tau_1] + \lambda s[n + t_{12} - \tau_2] + w_1[n] + w_2[n + t_{12}]\right)^2 \\ &= \lambda^2 \sum_n s[n - \tau_1]^2 + \lambda^2 \sum_n s[n + t_{12} - \tau_2]^2 + \sum_n w_1[n]^2 + \sum_n w_2[n + t_{12}]^2 \\ &\quad + 2\lambda^2 \sum_n s[n - \tau_1]s[n + t_{12} - \tau_2] + 2\sum_n w_1[n]w_2[n + t_{12}] \\ &\quad + 2\lambda \sum_n (w_1[n] + w_2[n + t_{12}])(s[n - \tau_1] + s[n + t_{12} - \tau_2]). \end{aligned}$$

Rewriting the above expression in terms of signal and noise autocorrelation and cross-correlation, we have

$$\begin{aligned}B(\alpha) &= \lambda^2 R_{ss}(0) + \lambda^2 R_{ss}(0) + R_{w_1w_1}(0) + R_{w_2w_2}(0) \\ &+ 2\lambda^2 R_{ss}(t_{12} - \tau_2 + \tau_1) + 2R_{w_1w_2}(t_{12}) + 2\lambda R_{w_1s}(-\tau_1) + 2\lambda R_{w_2s}(t_{12} - \tau_1) \\ &+ 2\lambda R_{w_1s}(t_{12} - \tau_2) + 2\lambda R_{w_2s}(\tau_2).\end{aligned}$$

Now, assuming that the noises at the microphones are statistically same (i.e. $R_{w_1w_1}(0) = R_{w_2w_2}(0) = R_\eta(0)$) and the noises are uncorrelated (i.e. $R_{w_1w_2}[m] = 0$), and the noise and signal are also uncorrelated (i.e. $R_{w_ks}[m] = 0$), we have

$$B(\alpha) = 2\lambda^2 R_{ss}(0) + 2R_\eta(0) + 2\lambda^2 R_{ss}(t_{12} - \tau_{12}).$$

Denoting $\kappa_\alpha = t_{12} - \tau_{12} = d(\cos(\alpha - \beta) - \cos(\theta - \beta))f_s/C$ and rearranging, we have

$$B(\alpha) = 2\lambda^2\big(R_{ss}(0) + R_{ss}(\kappa_\alpha)\big) + 2R_\eta(0).$$

□

For an arbitrary microphone-array, the generative model can be extended as follows.

Proposition 4. *For an arbitrary microphone-array of N_{mic} microphones, the beamform is expressed in terms of pairwise beamforms as*

$$B(\alpha) = \sum_{(i,j)\in\mathbf{pa}} B_{i,j}(\alpha) - N_{mic}(N_{mic} - 2)(R_\eta(0) + \lambda^2 R_{ss}(0)) \tag{48}$$

where **pa** *is the set of all microphone pairs, $R_{ss}(0)$ is the signal energy, $R_\eta(0)$ is the noise energy, λ is the attenuation factor, and $B_{i,j}$ is the beamform for the microphone pair (i, j) (Equation (47)).*

Proof. Consider a source present at an angle θ emitting a source signal $s[n]$. Using Equation (32), the received signals at the microphones are given by

$$r_p[n] = \lambda_p s[n - \tau_p] + w_p[n]$$

for $p = 1, 2, \cdots, N_{mic}$, where τ_p is the propagation delay, and λ_p is the attenuation factor. For far-field case, the distances between the source and the closely-spaced microphones will be approximately same for all microphones, hence $\lambda_1 \approx \lambda_2 \approx \cdots = \lambda$.

Using Equation (33), the composite microphone signal for the beam angle α is given by

$$\begin{aligned} r[n] &= \sum_{p=1}^{N_{mic}} r_p[n + t_{1p}(\alpha)] \\ &= \sum_{p=1}^{N_{mic}} \lambda s[n + t_{1p}(\alpha) - \tau_p] + w_p[n + t_{1p}(\alpha)] \end{aligned}$$

where $t_{1p}(\alpha) = t_p(\alpha) - t_1(\alpha) = d_{1p}\cos(\alpha - \beta_{1p})f_s/C$ is relative sample delay between the p^{th} and 1^{st} microphone. Let's denote $\phi_p = n + t_{1p}(\alpha) - \tau_p$ and $\psi_p = n + t_{1p}(\alpha)$ for clarity and brevity. The beam energy is given by

$$\begin{aligned} B(\alpha) &= \sum_n r[n]^2 \\ &= \sum_n \left[\sum_p \lambda s[\phi_p] + w_p[\psi_p]\right]^2 \\ &= \sum_n \left[\left(\sum_p \lambda s[\phi_p]\right)^2 + \left(\sum_p w_p[\psi_p]\right)^2 + 2\sum_p \sum_q \lambda s[\phi_p] w_q[\psi_q]\right] \\ &= \sum_n \left(\sum_p \lambda s[\phi_p]\right)^2 + \sum_n \left(\sum_p w_p[\psi_p]\right)^2 + 2\sum_p \sum_{q,q\neq p} \underbrace{\sum_n \lambda s[\phi_p] w_q[\psi_q]}_{R_{w_q s}(\tau)=0}. \end{aligned} \tag{49}$$

The last term is signal-noise cross-correlation which is zero for uncorrelated signal and noise. The first two term in Equation (49) are expanded to

$$\begin{aligned} \sum_n \left(\sum_p \lambda s[\phi_p]\right)^2 &= \lambda^2 \sum_n \sum_p s^2[\phi_p] + 2\lambda^2 \sum_n \sum_p \sum_{q,q\neq p} s[\phi_p] s[\phi_q] \\ &= \lambda^2 \sum_p \underbrace{\left(\sum_n s^2[\phi_p]\right)}_{R_{ss}(0)} + 2\lambda^2 \sum_p \sum_{q,q\neq p} \underbrace{\left(\sum_n s[\phi_p] s[\phi_q]\right)}_{R_{ss}(\phi_p - \phi_q)} \\ &= \lambda^2 N_{mic} R_{ss}(0) + 2\lambda^2 \sum_{p,q,p\neq q} R_{ss}(\phi_p - \phi_q) \end{aligned} \tag{50}$$

and

$$\begin{aligned}\sum_n \left(\sum_p w_p[\psi_p]\right)^2 &= \sum_n \sum_p w_p^2[\psi_p] + 2\sum\sum_p \sum_{q,q\neq p} w_p[\psi_p]w_q[\psi_q] \\ &= \sum_p \underbrace{\left(\sum_n w_p^2[\psi_p]\right)}_{R_{w_p w_p}(0)} + 2\sum_p \sum_{q,q\neq p} \underbrace{\left(\sum_n w_p[\psi_p]w_q[\psi_q]\right)}_{R_{w_p w_q}[\phi_p-\phi_q]=0} \\ &= N_{mic}R_\eta(0) \end{aligned} \tag{51}$$

The second term in Equation (51) is zero due to uncorrelated noises on different microphones. Substituting Equation (50) and Equation (51) back into Equation (49), we have

$$B(\alpha) = \lambda^2 N_{mic}R_{ss}(0) + 2\lambda^2 \sum_{p,q,p\neq q} R_{ss}(\phi_q - \phi_p) + N_{mic}R_\eta(0).$$

Rearranging the terms and denoting $\kappa_{pq} = \phi_q - \phi_p = t_{pq}(\alpha) - \tau_{pq}$

$$B(\alpha) = N_{mic}(\lambda^2 R_{ss}(0) + R_\eta(0)) + 2\lambda^2 \sum_{p,q,p\neq q} R_{ss}(\kappa_{pq}). \tag{52}$$

Adding and subtracting the term $2\frac{N_{mic}(N_{mic}-1)}{2}(\lambda^2 R_{ss}(0) + R_\eta(0))$, we have

$$\begin{aligned} B(\alpha) &= N_{mic}(\lambda^2 R_{ss}(0) + R_\eta(0)) + \underbrace{2\lambda^2 \sum_{p,q,p\neq q} R_{ss}(\kappa_{pq}] + 2\frac{N_{mic}(N_{mic}-1)}{2}(\lambda^2 R_{ss}(0) + R_\eta(0))} \\ &\quad - 2\frac{N_{mic}(N_{mic}-1)}{2}(\lambda^2 R_{ss}(0) + R_\eta(0)) \\ &= \sum_{p,q,p\neq q} \underbrace{2\left(\lambda^2 R_{ss}(\kappa_{pq}) + \lambda^2 R_{ss}(0) + R_\eta(0)\right)}_{B_{pq}(\alpha) \text{ from Proposition 3}} - N_{mic}(N_{mic}-2)(\lambda^2 R_{ss}(0) + R_\eta(0)) \\ &= \sum_{(p,q)\in\mathbb{P}} B_{pq}(\alpha) - N_{mic}(N_{mic}-2)(\lambda^2 R_{ss}(0) + R_\eta(0)) \end{aligned}$$

□

For an arbitrary number of acoustic sources, the generative model is given by the following proposition.

Proposition 5. *For an arbitrary number of uncorrelated acoustic sources M, the beamform is expressed in terms of single source beamforms as*

$$B(\alpha) = \sum_{m=1}^{M} B_m(\alpha) - N_{mic}(M-1)R_\eta(0) \tag{53}$$

where $R_\eta(0)$ is the noise energy and B_m is the beamform for m^{th} acoustic source (Equation (48)).

Proof. Consider M sources present at angles θ_m emitting source signals $s_m[n]$, for $m = 1, 2, \cdots, M$. Using Equation (32), the received signal at the p^{th} microphone is given by

$$r_p[n] = \sum_{m=1}^{M} \lambda_{mp} s_m[n - \tau_{mp}] + w_p[n]$$

where $p = 1, 2, \cdots, N_{mic}$, τ_{mp} is the propagation delay, and λ_{mp} is the attenuation factor. For far-field case, the distances between a source and the closely-spaced microphones will be approximately same for all microphones, hence $\lambda_{m1} \approx \lambda_{m2} \approx \cdots = \lambda_m$.

Using Equation (33), the composite microphone signal for the beam angle α is given by

$$\begin{aligned} r[n] &= \sum_{p=1}^{N_{mic}} r_p[n + t_{1p}(\alpha)] \\ &= \sum_{p=1}^{N_{mic}} \sum_{m=1}^{M} \lambda_m s_m[n + t_{1p}(\alpha) - \tau_{mp}] + w_p[n + t_{1p}(\alpha)] \end{aligned}$$

where $t_{1p}(\alpha) = t_p(\alpha) - t_1(\alpha) = d_{1p}\cos(\alpha - \beta_{1p})f_s/C$ is relative sample delay between the p^{th} and 1^{st} microphone. Let's denote $\phi_{mp} = n + t_{1p}(\alpha) - \tau_{mp}$ and $\psi_p = n + t_{1p}(\alpha)$ for clarity and brevity.

The beam energy is given by

$$
\begin{aligned}
B(\alpha) &= \sum_n r[n]^2 \\
&= \sum_n \left[\sum_p \sum_m \lambda_m s_m[\phi_{mp}] + w_p[\psi_p] \right]^2 \\
&= \sum_n \left[\left(\sum_p \sum_m \lambda_m s_m[\phi_{mp}] \right)^2 + \left(\sum_p w_p[\psi_p] \right)^2 + 2 \sum_p \sum_q \sum_m \lambda_m s_m[\phi_{mp}] w_q[\psi_q] \right] \\
&= \sum_n \left(\sum_p \sum_m \lambda_m s_m[\phi_{mp}] \right)^2 + \underbrace{\sum_n \left(\sum_p w_p[\psi_p] \right)^2}_{N_{mic} R_\eta(0) \text{ using Equation (51).}} \\
&\quad + 2 \sum_p \sum_q \sum_m \underbrace{\sum_n \lambda_m s_m[\phi_{mp}] w_q[\psi_q]}_{R_{w_q s_m}(\tau)=0} .
\end{aligned}
\tag{54}
$$

The first term in Equation (54) is expanded to

$$
\begin{aligned}
\sum_n \left(\sum_p \sum_m \lambda_m s_m[\phi_{mp}] \right)^2 &= \sum_n \left(\sum_m \sum_p \lambda_m s_m[\phi_{mp}] \right)^2 \\
&= \sum_n \left(\sum_m \left(\sum_p \lambda_m s_m[\phi_{mp}] \right)^2 + 2 \sum_{m_1} \sum_{m_2} \lambda_{m_1} s_{m_1}[\phi_{m_1 p}] \lambda_{m_2} s_{m_2}[\phi_{m_2 p}] \right) \\
&= \sum_m \left(\sum_n \left(\sum_p \lambda_m s_m[\phi_{mp}] \right)^2 \right) + 2 \sum_{m_1} \sum_{m_2} \underbrace{\sum_n \lambda_{m_1} s_{m_1}[\phi_{m_1 p}] \lambda_{m_2} s_{m_2}[\phi_{m_2 p}]}_{R_{s_{m_1} s_{m_2}}(\tau)=0} \\
&= \sum_m \underbrace{\left(\sum_n \left(\sum_p \lambda_m s_m[\phi_{mp}] \right)^2 \right)}_{\text{substitute from Equation (50)}} \\
&= \sum_m \lambda_m^2 \left(N_{mic} R_{s_m s_m}(0) + \sum_{p,q,p \neq q} R_{s_m s_m}(\phi_{mp} - \phi_{mq}) \right)
\end{aligned}
\tag{55}
$$

Denoting $\kappa_{mpq} = \phi_{mq} - \phi_{mp} = t_{pq}(\alpha) - \tau_{mpq}$, and substituting Equation (55) in Equation (54), we have

$$
B(\alpha) = \sum_m \lambda_m^2 \left(N_{mic} R_m(0) + \sum_{p,q,p \neq q} R_m(\kappa_{mpq}) \right) + N_{mic} R_\eta(0)
$$

Adding and subtracting the term $\sum_m N_{mic} R_\eta(0)$, we have

$$
\begin{aligned}
B(\alpha) &= \sum_m \lambda_m^2 \left(N_{mic} R_m(0) + \sum_{p,q,p\neq q} R_m(\kappa_{mpq}) \right) + \sum_m N_{mic} R_\eta(0) - \sum_m N_{mic} R_\eta(0) + N_{mic} R_\eta(0) \\
&= \sum_m \underbrace{\lambda_m^2 \left(N_{mic} R_m(0) + \sum_{p,q,p\neq q} R_m[\kappa_{pq}] \right) + N_{mic} R_\eta(0)}_{B_m(\alpha), \text{ using Equation (52)}} - M N_{mic} R_\eta(0) + N_{mic} R_\eta(0) \\
&= \sum_m B_m(\alpha) - N_{mic}(M-1) R_\eta(0)
\end{aligned}
$$

□

Finally, the generative model for arbitrary microphone array and arbitrary number of sources can be obtained by substituting Equations (47) and (48) into Equation (53), which gives

$$
\mathbb{B}(\alpha) = 2 \sum_{m=1}^{M} {\lambda^{(m)}}^2 \sum_{(i,j)\in \mathbf{pa}} R_{ss}^{(m)}(\kappa_\alpha) + N_{mic} \sum_{m=1}^{M} {\lambda^{(m)}}^2 R_{ss}^{(m)}(0) + N_{mic} R_\eta(0) \tag{56}
$$

Data Likelihood & MAP Estimate

Using Equation (56), the negative log-likelihood for beamform data is given as

$$
-\ln p(B_k|\mathbf{X}) = \ell_k(\mathbf{X}) = \frac{1}{\sigma_B^2} \sum_\alpha \parallel B_k(\alpha) - \mathbb{B}_k(\alpha) \parallel^2
$$

The MAP estimate of the source positions is given by

$$
\hat{\mathbf{X}}^{MAP} = \arg\max_{\mathbf{X}} p(\mathbf{X}|\mathbf{B}) \tag{57}
$$

Since the generative model for beamform in non-linear, an exact method for state estimation in Equation (57) is not possible and we use Monte Carlo method described in the next section for state estimation.

Monte Carlo Estimation

Markov Chain Monte Carlo methods are a class of Monte Carlo methods for sampling complex probability distributions based on constructing a Markov chain that has the desired distribution as

its equilibrium distribution. MCMC approaches are so-named because they use the previous sample values to randomly generate the next sample value, generating a Markov chain (as the transition probabilities between sample values are only a function of the most recent sample value). The state of the chain after a large number of steps is then used as a sample from the desired distribution. The quality of the sample improves as a function of the number of steps. The MCMC methods are more efficient, especially for problems with high-dimensional state-space, than sequential Monte Carlo (SMC) methods, also called particle filters [46]. This is due to the fact that the samples in SMC methods are drawn independently, while samples in MCMC are drawn from a Markov chain. If the desired distribution is highly localized in a high-dimensional state space, most of the independent samples drawn by SMC methods would have low probability. On the other hand, after a sufficient number of steps, samples drawn by MCMC methods would, in fact, be from the desired distribution.

The Metropolis-Hastings (MH) algorithm is the earliest of the MCMC method [159]. The MH algorithm generates a Markov chain using a proposal density which depends on the current sample. The proposed samples are accepted as part of Markov chain according to a acceptance-rejection rule. Another popular MCMC method called the Gibbs sampler is very widely applicable to a broad class of Bayesian problems [160]. The Gibbs sampler is a special case of Metropolis-Hastings sampling wherein the proposed sample is always accepted. This results in lesser rejected samples, hence better efficiency. Gibbs sampling, however, requires that all the conditional distributions of the target distribution can be sampled.

In this chapter, we use Gibbs sampling for estimation. Gibbs sampling algorithm works on the idea that while the joint probability distribution is too complex to draw samples from directly, the *univariate* conditional distributions – the distribution when all but one of the random variables are assigned fixed values – are easier to sample. We denote the state vector as $X = \left[x^{(1)}, x^{(2)}, \cdots, x^{(D)}\right]^{\mathrm{T}}$, where D is the number of state variables. The joint density $p(X_t|X_{t-1}, Y_t)$ is sampled using Gibbs sampler by sequentially sampling univariate conditional densities given by

$$x_t^{(k,j)} \sim p(x^{(j)}|X_t^{(k,-j)}, X_{t-1}, Y_t) \tag{58}$$

where k is the index of the sample, $j = 1, \cdots, n$ is the index of state variable currently being sampled, and $X_t^{(k,-j)} = \left[x_t^{(k,1)}, \cdots, x_t^{(k,j-1)}, x_t^{(k-1,j+1)}, \cdots, x_t^{(k-1,D)}\right]^{\mathrm{T}}$ is the set of all state variables except $x^{(j)}$. In many cases, the univariate conditional distribution can be arbitrary and the choice of one-dimensional sampling algorithm to sample from the univariate distribution determines the speed and convergence of the Gibbs sampler. We select slice sampling for its robustness in parameters such as

step size and applicability toward non-log-concave densities, which is the case in our problem due to multimodal probability distributions [161].

The pseudo code in Algorithm 1 presents the Monte Carlo method for source separation and localization using Gibbs sampling and slice sampling. At time $t = 0$ (line 1), the Gibbs sampling algorithm is initialized with source fundamental frequencies and source locations. For each time $t > 0$, we perform ML estimation for source separation (lines 3–10) and Bayesian estimation for source localization (lines 11–19). The Gibbs sampler draws N samples (line 4 & 12) from the target distribution by sequentially drawing from *univariate* conditional distributions (lines 5–7 & 13–16). Note that slice sampling is used for univariate sampling (lines 6 & 14–15). Notation from Equation (58) is used in lines 6 & 14–15. The state for k^{th} sample is shown in lines 8 & 17. For ML estimation, the sample with minimum negative-log-likelihood is selected as the estimate (line 10). For Bayesian estimation, the sample with maximum *a posteriori* is selected as the state estimate (line 19).

Algorithm 1 Monte Carlo source separation and source localization algorithm

1: At $t = 0$, initialize Gibbs sampler $(\Omega_{f,0}, \mathbf{X}_0)$
2: **for** $t > 0$ **do**
3: `%%% Source Separation (MC-ML Estimation)`
4: **for** $k = 1, \cdots, N$ **do**
5: **for** $m = 1, \cdots, M$ **do**
6: sample $\omega_{f,t}^{(k,m)} \sim p(\omega_f^{(m)} | \Omega_{f,t}^{(k,-m)}, \Omega_{f,t-1}, P_t)$
7: $\Omega_{f,t}^{(k)} = \left[\omega_{f,t}^{(k,1)}, \cdots, \omega_{f,t}^{(k,M)}\right]$
8: ML estimate, $\hat{\Omega}_{f,t}^{ML} = \arg\min_{\Omega_{f,t}^{(k)}} \ell'(\Omega_{f,t}^{(k)})$
9: `%%% Source Localization (MC Bayesian Estimation)`
10: **for** $k = 1, \cdots, N$ **do**
11: **for** $m = 1, \cdots, M$ **do**
12: sample $x_t^{(k,m)} \sim p(x^{(m)} | \mathbf{X}_t^{(k,-m)}, \mathbf{X}_{t-1}, B_t, \hat{\Omega}_{f,t}^{ML})$
13: sample $y_t^{(k,m)} \sim p(y^{(m)} | \mathbf{X}_t^{(k,-m)}, \mathbf{X}_{t-1}, B_t, \hat{\Omega}_{f,t}^{ML})$
14: $\mathbf{X}_t^{(k)} = \left[x_t^{(k,1)}, y_t^{(k,1)}, \cdots, x_t^{(k,M)}, y_t^{(k,M)}\right]$
15: MAP estimate, $\hat{\mathbf{X}}_t^{MAP} = \arg\max_{\mathbf{X}_t^{(k)}} p(\mathbf{X}_t^{(k)} | \mathbf{X}_{t-1}, \mathbf{B})$

Simulation Results

Typically, localization of an acoustic source in WSNs is performed by the sensors that are close to the source because the signal-to-ratio (SNR) is lower for farther sensors. For this reason, we assume that even in a large sensor network, a source will be surrounded by a small number of sensors that will participate in the localization of that source.

Setup and Parameters

In simulations, we consider a sensor network of 4 acoustic sensors arranged in a grid of size $10m \times 5m$, wherein each sensor can detect all the sources. We simulate the sources according to the acoustic source model in Section IV, simulate the data according to the feature extraction algorithms in Section IV, and finally compare the output of source localization against the ground truth. The performance of the approach is measured in terms of the localization error, which is defined as the root mean square (RMS) position error averaged over all the sources

$$E = \frac{1}{M} \sum_{m=1}^{M} ||\mathbf{x}^{(m)} - \tilde{\mathbf{x}}^{(m)}||$$

where M is the number of source, and $\mathbf{x}^{(m)}$ and $\tilde{\mathbf{x}}^{(m)}$ are the estimated and the ground truth positions for the m^{th} source, respectively. Table 10 shows the parameters used in the algorithm.

Table 10: Parameters used in simulations

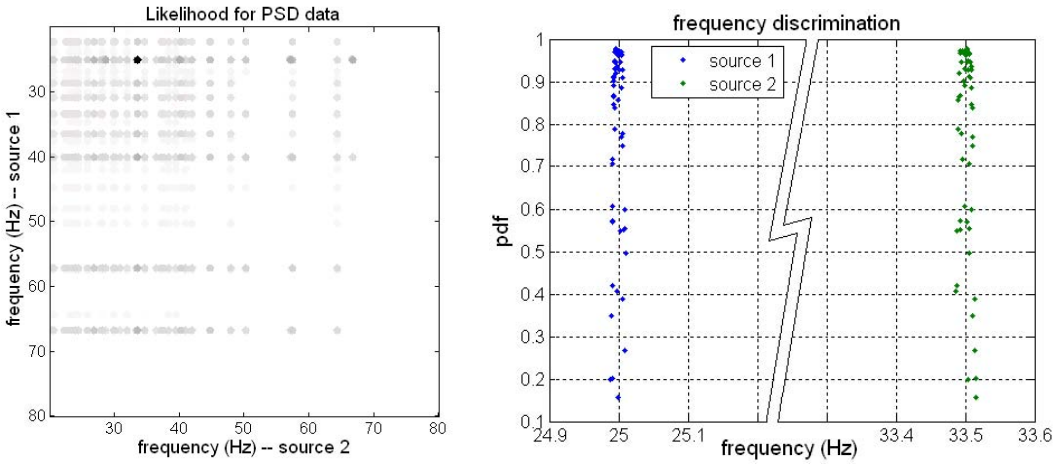
Sampling frequency (f_s)	100kHz
Speed of sound (C)	350 m/sec
Audio data length (time)	1 sec
Maximum harmonic frequency (ω_{max})	1000Hz
SNR (dB)	25
Number of beams	36
Size of Fourier transform (N_{FFT})	4000
Number of Gibbs samples	40

Frequency Discrimination

Figure 29(a) shows the PSD data likelihood for two sources. The data likelihood is highly multimodal but localized. The data likelihood near the true fundamental frequency values is higher than the likelihood for other frequencies. Figure 29(b) shows the localized nature of the PSD data likelihood. The data likelihood is centered around the true fundamental frequency with a standard deviation of less than 0.01 Hz. Due to this localized nature of the data likelihood, we can discriminate fundamental frequencies as close as 0.1 Hz.

Localization and Spatial Discrimination

We study three simulation scenarios. In the first scenario, we increase the number of sources present in the sensing region gradually to see the effect on localization accuracy. In the second scenario, we increase the average source SNR of two sources present in the sensing region. In the third scenario,



(a) (b)

Figure 29: Frequency discrimination results: (a) PSD data likelihood, (b) Frequency discrimination.

we increase the separation between two sources present in the sensing region to evaluate spatial discrimination.

Figure 30(a) shows the localization error for the first scenario when the number of sources is increased from 1 to 4. The localization error increases approximately exponentially with the number of sources. Figure 30(b) shows the average localization error for the second scenario when source SNR for the two sources is increased from 7dB to 52dB. As expected, the localization error decreases with increasing SNR and remains approximately constant above 25dB. Figures 30(c) and 30(d) show the localization error for the third scenario when the source separation between the two sources is increased from $0.1m$ to $8m$. For small source separations ($0.1m$ and $0.2m$), the localization error is of the same order as the separation. This indicates that the two sources cannot be disambiguated at such separation. For higher source separation (above $0.5m$), the localization error is a small fraction of the separation distance. This indicates that the two sources are successfully localized and discriminated. In fact, for larger source separation (above $5m$), the average localization error for the two sources is the same as that of the single sources.

Relaxation of Source Assumptions

We evaluate the localization accuracy when the assumptions for the acoustic sources are relaxed; specifically the harmonic and omnidirectional sources assumptions.

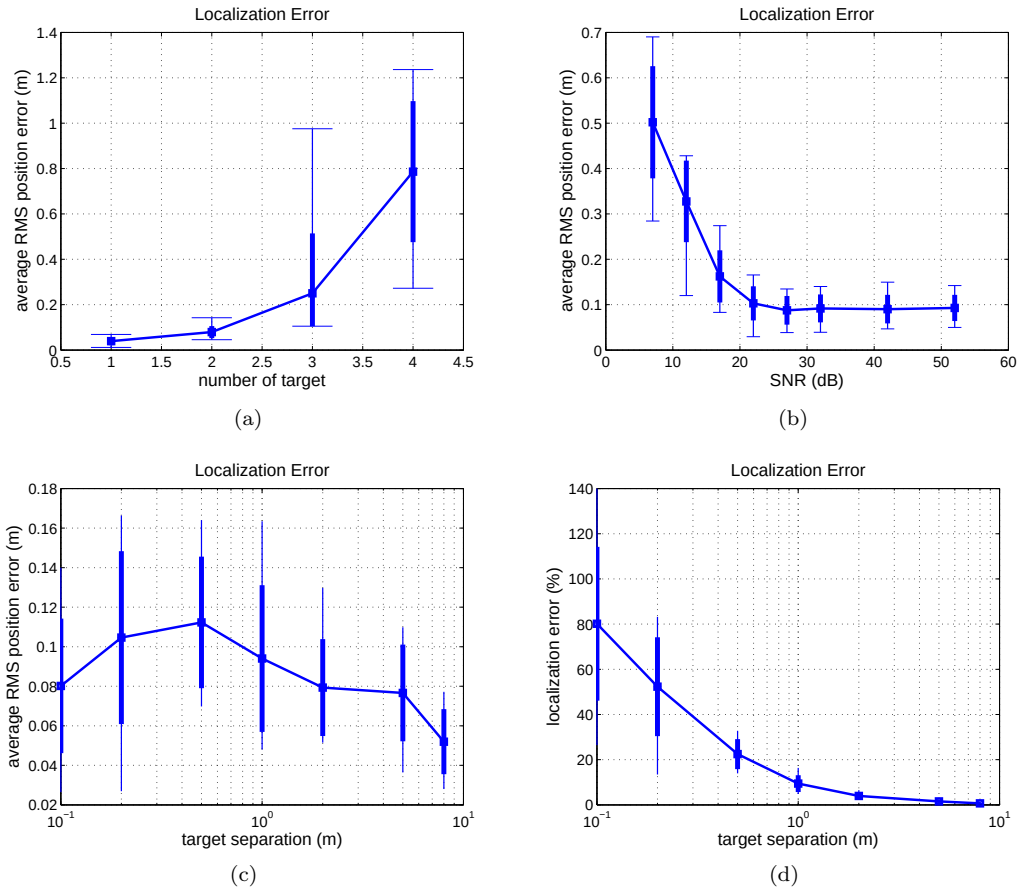


Figure 30: Localization error with (a) Source density, (b) Source SNR, and (c) Source separation. (d) Localization error as a percentage of source separation.

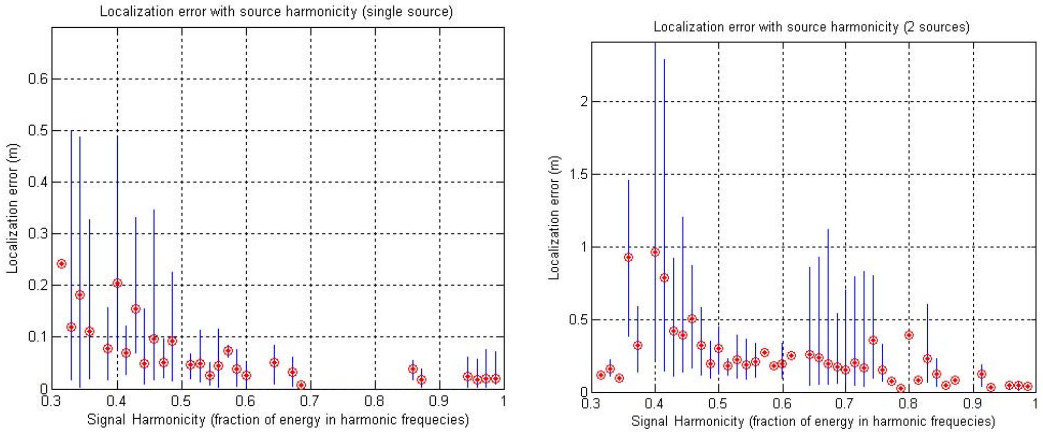
Harmonicity

Harmonicity of a signal represents the degree of acoustic periodicity in the signal. In this analysis, we define signal harmonicity as the ratio of signal energy in the harmonic frequencies over the total signal energy

$$\hbar = \frac{\sum_{\omega \in \mathbb{H}} P(\omega)}{\sum_{\omega} P(\omega)} \tag{59}$$

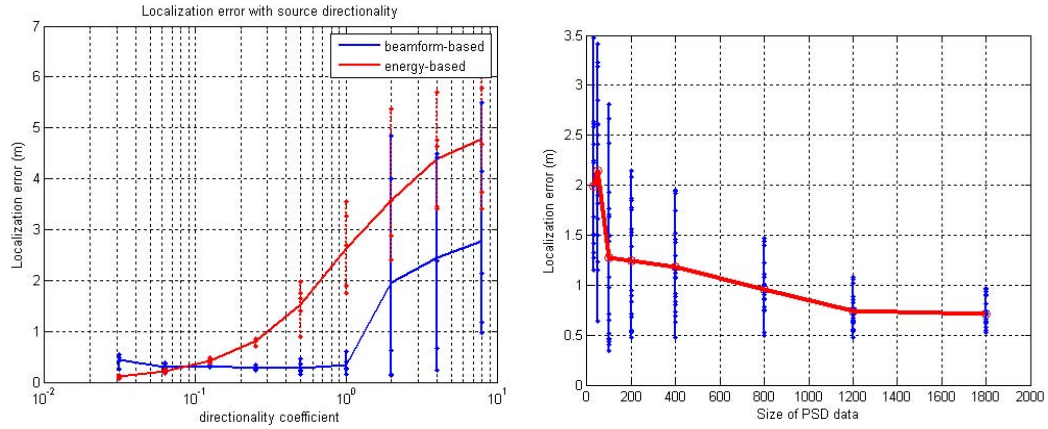
Acoustic signals originating from rotating machinery will have high harmonicity close to unity, while signals due wind or a white noise source will have low harmonicity. The localization accuracy of our approach degrades gracefully when signal harmonicity is decreased. Figure 31(a) and 31(b) show the localization error for single source and two sources, respectively, with signal harmonicity. The localization error decreases as the signal harmonicity is increased. Hence, the signal harmonicity computed using Equation (59) can be also used as an indicator of confidence in the localization

result.



(a) (b)

Figure 31: Localization error with signal harmonicity for (a) Single Source, (b) Two Sources.



(a) (b)

Figure 32: Localization error with (a) Source directionality, (b) Amount of PSD data available.

Directivity

Directivity of an acoustic source is a measure of its directional characteristics. Directivity indicates how much signal energy is directed toward a specific area compared to the total signal energy being transmitted by the source. In this analysis, we express 2D source directivity in terms of a *directionality coefficient* that governs the attenuation of the signal energy with the angle. The directional signal attenuation is given by

$$\lambda_\phi = \left(\frac{2+\cos\phi}{3}\right)^\beta$$

where β is the directionality coefficient and ϕ is the direction. The value of directionality coefficient is zero ($\beta = 0$) for omnidirectional sources (i.e. the signal attenuates uniformly in all directions). Figure 32(a) shows the localization error for two sources when the directionality of the sources is increased. The localization error for beamform-based localization is not affected for directionality coefficient as high as 1.0, after which the error increases rapidly. As compared to energy-based localization, beamform-based localization is able to cope with higher directionality. It is expected that if we increase the number of sensors in beamform-based localization, the effect of source directionality can be further reduced.

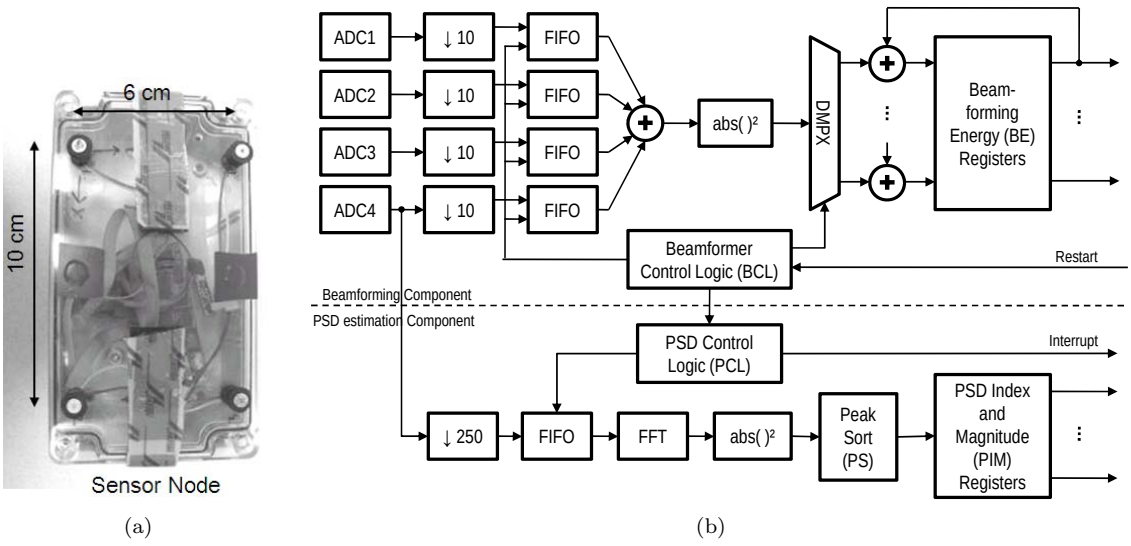
PSD Data Compression

We also present empirical analysis for the effect of PSD data compression to the localization accuracy by increasing the N_{PSD} parameter presented in Section IV. As expected, the localization accuracy improves when more PSD data is available. Figure 32(b) shows the localization error for two sources as a function of size of PSD data available to the fusion algorithm. As expected, the localization error decreases as more PSD data is made available to the base station. The accuracy is poor and degrades rapidly for smaller PSD data size.

Outdoor Experiments

We implemented the beamforming and PSD estimation algorithms described in Section IV on a Xilinx XC3S1000 FPGA based MICAz sensor motes (see Figure 33(a)). The outline of the overall design is as follows. First, the FPGA collects samples of the signal received at the microphones in FIFO buffers and performs beamforming and PSD estimation. Then the FPGA stores the results, i.e. beamforming energies and N_{PSD} highest values of the PSD, in registers easily accessible from the MICAz mote. Once these registers are read the mote transmits their value to the base station where further processing and sensor fusion takes place. The block diagram of the FPGA design is shown in Figure 33(b), where the upper half (above the dashed line) represents the beamforming- and the bottom half shows the PSD estimation component. Beamforming component utilizes 166 msec of audio data each cycle, while the PSD estimation component utilizes 1 sec of data with 75% overlap. The angular resolution of beamforming is 10 degrees while frequency resolution of PSD estimation is 1 Hz. The PSD estimation component returns 30 PSD values.

Beamforming Component. The entire FPGA application runs at 20 MHz and the Analog to Digital Converters (ADCs) are sampling at 1 MSPS. The beamforming component uses all four



(a) (b)

Figure 33: (a) Acoustic sensor node with 4 microphone. (b) Block diagram of Beamforming- and PSD estimation components realized on FPGA.

microphone channels. The 8-bit ADC values are first downsampled by a factor of 10 and stored in a FIFO buffer. FIFO buffers are realized as circular buffers using 2-Kbyte embedded block RAMs that store sample values from last 20 ms. Read access to the FIFO buffers and write access to the Beamforming Energy (BE) registers are controlled by the Beamformer Control Logic (BCL). Elements in the FIFOs are selected for read by a simple state machine that resides in the BCL. The state machine iterates through 36 states each corresponding to a specific direction and accesses the FIFO elements based on a look-up-table that contains the delay information for the different angles. The accessed values of the four FIFOs are summed, squared and accumulated in the corresponding BE register in one FPGA clock cycle for each direction. Once the energies for all the 36 directions have been calculated, BCL waits for the next sample to arrive and starts this procedure over. This procedure is repeated 2^{14} times which takes approximately 160 ms and ensures that the 32-bit BE registers do not overflow. After the last cycle, a trigger signal is generated for the PSD estimation component and the beamforming component is halted.

PSD Estimation Component. The PSD estimation component uses the samples from only one channel. The signal is first re-sampled at 4 kHz. For the decimation process, a 600-tap poly-phase filter with 2 kHz cut-off frequency is used. The decimated 8-bit samples are then fed into a 4-Kbyte FIFO buffer which allows to store up to 1 s sample history. In response to a trigger event received from the beamforming component the FIFO content is loaded into the 4096 point FFT module. Only the magnitude information of the first 2048 FFT results is forwarded to the Peak Sort (PS)

module. The PS module then selects and stores 30 elements with largest magnitudes and their corresponding indices. After all the FFT samples are processed an interrupt signal is generated for the MICAz mote, which in return reads all the register values through an I2C bus, transmits the values through the radio and restarts the beamforming component. The PSD estimation time is negligible compared to the I2C and radio transfers, which take approximately 90 ms together. The beamforming process in conjunction with the PSD estimation and the data transfers takes roughly 250 ms, which results in an approximately 4 Hz update rate at the base station.

Resource Utilization. The resources used by the beamforming- and PSD estimation components and the overall design (including all driver modules) are shown in Table 11. The utilization rates of the different FPGA components compared to available resources found in the Xilinx XC3S1000 FPGA prove the feasibility of this application, but also points out its limitations. Since 75% of block RAMs are already used, memory resources are likely to become a bottleneck when using larger data sets (FIFO sizes).

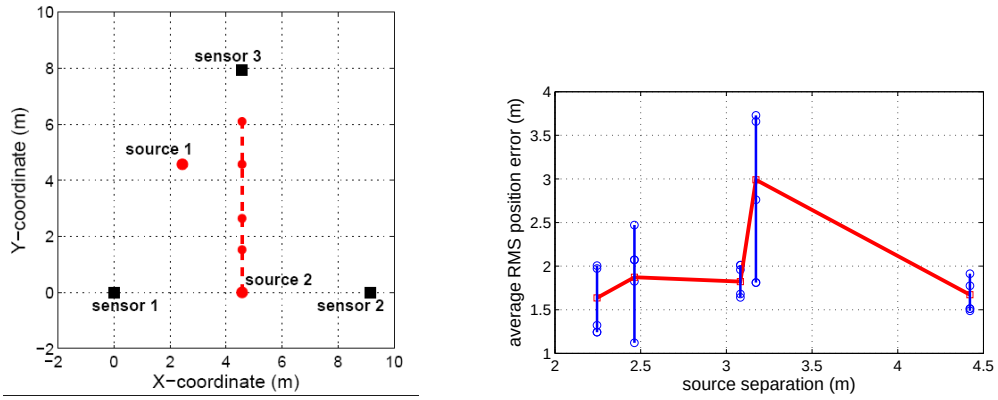
Table 11: FPGA resource utilization of the Beamforming-, PSD Estimation modules and the overall design in absolute numbers and relative to the available resources found in the Xilinx XC3S1000 FPGA.

	Beamforming	**PSD Estimation**	**Overall Design**
Flip Flops	1,258 (8%)	2,018 (13%)	3,843 (8%)
4 input LUTs	1,398 (9%)	2,578 (16%)	6,860 (44%)
Block RAMs	8 (33%)	9 (38%)	18 (75%)
Hardware multipliers	1 (4%)	6 (25%)	7 (29%)

Experiments using an Outdoot Deployment We deployed a sensor network of 3 MICAz-based acoustic sensor nodes in an equilateral triangle of side length 9.144m (15ft). Figure 34(a) shows the experimental setup and the location of the sources. We collected the sensor data and ran the algorithm offline. Figure 34(b) shows the localization error with source separation. The results follow the similar trend as that in Figure 30(c). For smaller source separations, the average error remains low but the algorithm is not able to disambiguate the two sources. For larger separations, the localization error decreases.

Conclusions

In this chapter, we proposed a feature-based fusion method for localization and discrimination of multiple acoustic sources in WSNs. Our approach fused beamforms and PSD data from each sensor.



(a) (b)

Figure 34: (a) Outdoor experimental setup. Source 1 is kept at the same location while source 2 is placed at different locations. (b) Localization error with source separation.

The approach utilized a graphical model for estimating the source positions and the fundamental frequencies. We subdivided the problem into source separation and source localization. We showed in simulation and outdoor experiments that the approach can discriminate multiple sources using the simple features collected from the resource-constrained sensor nodes. As part of an ongoing work, we are working on target dynamics models to extend the approach for multiple source tracking. In the future, the use of graphical models will allow us to extend the approach to multimodal sensors.

CHAPTER V

COLLABORATIVE TARGET TRACKING USING MULTIPLE VISUAL FEATURE IN SMART CAMERA NETWORK

With the evolution and fusion of technologies from sensor networks and embedded cameras, smart camera networks are emerging as useful and powerful systems. In this chapter, we present an approach for tracking multiple targets in 3D space using a wireless network of smart cameras. Existing camera network approaches for target tracking either utilize target handover mechanism between cameras for continuous tracking, or combine results from 2D trackers into 3D target state [125–127]. Such approaches suffer from the drawbacks associated with 2D tracking, such as scale selection, target rotation, and occlusion. In addition, target handover requires robust mechanisms for target (re)initialization as a target (re)enters a camera field-of-view (FoV). On the other hand, in 3D trackers where the target state and the target model are maintained in 3D space irrespective of the camera network parameters, we do not need to do (re)initialization as the target (re)enters a camera FoV. Employment of wireless networks, however, introduces new constraints of limited bandwidth, limited computation and limited power.

In our approach, we use multi-view histograms in different feature-spaces to characterize targets in 3D space. We employ color and texture as the visual features to model targets. The visual features from each camera, along with the target models are used by a probabilistic tracker to estimate the target state. We demonstrate the effectiveness of our proposed base tracker by comparing its performance to a 3D tracker that fuses results of independent 2D trackers. We also present a performance analysis of the base tracker and propose several variations that trade off Quality-of-Service and Quality-of-Information.

Introduction

Smart cameras are evolving on three different evolutionary paths [124]. First, *single smart cameras* focus on integrating sensing with embedded on-camera processing power to perform various vision tasks on-board and deliver abstracted data from the observed scene. Second, *distributed smart cameras* (DSC) introduce distribution and collaboration of smart cameras resulting in a network of cameras with distributed sensing and processing. The main motivations for DSC are to (1)

resolve occlusion, (2) mitigate the single camera handicap, and (3) extend sensing coverage. Finally, *pervasive smart cameras* (PSC) integrate adaptivity and autonomy to DSC.

Single camera tracking algorithms are often applied in the image plane. These image-plane (or 2D) trackers often run into problems such as target scale selection, target rotation, occlusion, view-dependence, and correspondence across views [125]. There are few 3D tracking approaches [125,126] that fuse results from independent 2D trackers to obtain 3D trajectories. These approaches employ decision-level fusion, wherein local decisions made by the node (i.e., 2D tracks) are fused to achieve global decision (i.e., 3D tracks), while discarding the local information (i.e., images captured at nodes). Because of the decision-level fusion, these approaches also suffer from all the problems associated with 2D tracking.

Tracking applications based on distributed and embedded sensor networks are emerging today, both in the fields of surveillance and industrial vision. The above-mentioned problems that are inherent in the image-plane based trackers can be circumvented by employing a tracker in 3D space using a network of smart cameras. Such smart camera networks can be employed using wireline or wireless networks. Wireless networks seem more suited due to their easy deployment in complex environments. In wireless networks, traditional centralized approaches have several drawbacks, due to limited communication bandwidth, computational requirements, and thus limiting the spatial camera resolution and the frame rate. The challenges for wireless smart camera networks include robust target tracking against scale variation, rotation and occlusion, especially in the presence of bandwidth constraints due to the wireless communication medium. We propose an approach for collaborative target tracking in 3D space using a wireless network of smart cameras. Modeling the target in 3D space circumvents the problems inherent in the 2D tracker, or a combination of 2D trackers.

The contributions of this chapter are listed below.

1. We define a new target representation, including target state and target reference model, which is suitable for 3D tracking. The target state consists of the position and orientation of the target in 3D space. The target model consists of its multi-view feature histograms. Such a model would correspond to the actual 3D target that does not change, and hence, unlike the image-plane based trackers, the target model does not need to be updated or learned during tracking. We also extend the definition of similarity measure for our proposed target model.

2. We develop a probabilistic 3D tracker based on our new target representation and implement the tracker using sequential Monte Carlo algorithms.

3. We develop and implement several variations of the base tracker that incur different computational and communication costs at each node, and produce different tracking accuracy. The variations include optimizations such as the use of mixture models, in-network aggregation and the use of image-plane based filtering where it is appropriate. We also present a qualitative comparison of the trackers according to their supported Quality-of-Service (QoS) and Quality-of-Information (QoI).

4. We present quantitative evaluation of the trackers using synthetic targets in simulated camera networks, as well as using real targets (objects and people) in real-world camera network deployments. We also compare the proposed trackers with an implementation of a previous approach for 3D tracking, which is based on 3D ray intersection. The simulation results show robustness against target scale variation and rotation, while working within the bandwidth constraints.

Background – Tracking with Single Camera

The two major components in a typical visual tracking system are the target representation and the tracking algorithm. The target is characterized by a reference target model in a suitable feature space. The reference target model is represented by its probability density function (pdf) in the feature space. The tracking algorithm typically includes searching the current image frame against the reference target model. Typically, the search is limited to a fixed-shape variable size window (also called the target candidate) around the target location in the previous image frame. The target candidate for which the pdf in the feature space closely matches that of the target model is declared as the estimated target location.

Target Representation

The target is characterized by a reference target model in the feature space of interest. Typically, reference target models are obtained by histogramming techniques in the feature space. For example, the model can be chosen to be the color, texture or edge-orientation histogram of the target. In [95], Red-Green-Blue (RGB) colorspace is taken as the feature space, while in [122], Hue-Saturation-Value (HSV) colorspace is taken as the feature space in order to decouple chromatic information from shading effects.

Target Model Consider a target region defined as the set of pixel locations $\{\mathbf{x}_i\}_{i=1\cdots n}$ in an image I. Without loss of generality, consider that the region is centered at $\mathbf{0}$. We define the function $b : \mathbb{R}^2 \to \{1 \cdots m\}$ that maps the pixel at location $\mathbf{x}_i$ to the index $b(\mathbf{x}_i)$ of its bin in the quantized feature space.

Within this region, the target model is defined as, $\mathbf{q} = \{q_u\}_{u=1\cdots m}$, with

$$q_u = C \sum_{i=1}^{n} k(\| \mathbf{x}_i \|^2) \delta[b(\mathbf{x}_i) - u] \tag{60}$$

where δ is the Kronecker delta function, C is the normalization constant such that $\sum_{u=1}^{m} q_u = 1$, and $k(x)$ is a weighting function. For example, in [95], this weighting function is an anisotropic kernel, with a convex and monotonic decreasing kernel profile that assigns smaller weights to the pixels farther from the center. If we set $w \equiv 1$, the target model is equivalent to the standard bin counting.

Target Candidate A target candidate is defined similar to the definition of the target model above. Consider a target candidate at $\mathbf{y}$ as the region which is a set of pixel locations $\{\mathbf{x}_i\}_{i=1\cdots n}$ centered at $\mathbf{y}$ in the current frame. Using the same weighting function, $k(x)$ and feature space mapping function, $b(\mathbf{x})$, the target candidate is defined as, $\mathbf{p}(\mathbf{y}) = \{p_u(\mathbf{y})\}_{u=1\cdots m}$, with

$$p_u(\mathbf{y}) = C \sum_{i=1}^{n} k(\| \mathbf{y} - \mathbf{x}_i \|^2) \delta[b(\mathbf{x}_i) - u] \tag{61}$$

where C is the normalization constant such that $\sum_{u=1}^{m} p_u(\mathbf{y}) = 1$.

Similarity Measure A similarity measure between a target model $\mathbf{q}$ and a target candidate $\mathbf{p}(\mathbf{y})$ plays the role of data likelihood and its local maxima in the frame indicate the target state estimate. Since both the target model and the target candidate are discrete distributions, the standard similarity function is the Bhattacharya coefficient [116] defined as

$$\rho(\mathbf{y}) \equiv \rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] = \sum_{u=1}^{m} \sqrt{p_u(\mathbf{y}) q_u} \tag{62}$$

Tracking Algorithm

The two most common approaches for target tracking using a single camera are mean-shift tracking and particle filter-based tracking.

Mean-Shift Tracker In this tracker, the feature histogram-based target representations are regularized by spatial masking with an isotropic kernel [95]. The masking generates spatially-smooth similarity functions. Finally, the target tracking problem is reformulated as finding the local maxima on this spatially-smooth similarity function. The approach in [95] employs a metric derived from the Bhattacharya coefficient as a similarity measure, and uses the mean-shift mode seeking algorithm for maximization.

Particle Filter In the mean-shift tracker, the search for the current estimate is deterministic. A deterministic search might encounter problems when the background close to the target contains similar features, or when the target is partially, or completely occluded momentarily. Unlike the deterministic search in mean-shift tracking, in probabilistic tracking the search is made probabilistically within a sequential Monte Carlo framework [122]. This requires constructing a feature-space likelihood model, which can be based on the similarity measure such as the Bhattacharya coefficient, or any other similarity measure, for example, the Matusita metric [162]. In [122], the color likelihood based on Bhattacharya distance is used with a dynamical state space model for sequential Bayesian estimation using a particle filter.

Tracking with Camera Network

Since single camera tracking algorithms are applied in the image-plane, such image-plane (2D) trackers often run into problems such as target scale selection, target rotation, occlusion, view-dependence, and correspondence across views [125]. There are a few 3D tracking approaches [125,126] that fuse results from independent 2D trackers to obtain 3D trajectories. These approaches employ decision-level fusion, wherein local decisions made by the node (i.e. 2D tracks) are fused to achieve global decision (i.e. 3D tracks), while discarding the local information (i.e. images captured at nodes). Because of the decision-level fusion, these approaches also suffer from all the problems associated with 2D tracking.

Target Hand-off

Another problem in 2D image-plane based trackers is the (re)initialization of a target when it (re)enters a camera field-of-view. In current state-of-the-art approaches, (re)initialization is performed by handing over target state to an adjacent camera node, which maintains the target state until it hands over the state to some other camera node.

An autonomous multicamera tracking approach based on a fully decentralized handover mechanism between adjacent cameras is presented in [127]. The system automatically initiates a single tracking instance for each target of interest. The instantiated tracker for each target follows the target over the camera network, migrating the target state to the camera which observed the object. This approach, however, utilizes data from only a single camera node at any given time. The authors do admit that the effectiveness of their handover mechanism introduces some requirements for the tracker. First, the tracker must have a short initialization time to build a target model. Second, the tracker on the new camera node must be able to initialize itself from a previously saved state, or handed-over state. Finally, the tracker must be robust with respect to the position and orientation of a target such that it must be able to identify the same target on the next camera node. These requirements need sophisticated and fine-tuned algorithms for 2D image-plane based tracker. On the other hand, in 3D trackers, the target state and the target model are maintained in 3D space. The target state and the target model are not tied to the camera network parameters, such as the number of cameras, position and orientation of the cameras, etc. Once initialized, the target model does not need to be re-initialized as it moves in the sensing region and enters, or re-enters a camera field-of-view.

3D Tracking using Ray Intersection

The classical and most naive approach for 3D collaborative target tracking is to combine the 2D tracking results from individual camera nodes for 3D tracking. This can be done by projecting rays from the camera center to the image affine coordinate in the world coordinate system and finding the intersection of multiple such rays from multiple cameras. This approach requires each camera node to maintain a 2D target model and a feature histogram. Hence the problems of scale variation, rotation and occlusion are not alleviated. As soon as the target moves along the camera principal axis, or rotates around its axis, the target model including the size and feature-histogram would become invalid. A target model learning algorithm during target tracking can help mitigate the problem but any sudden change which is faster than the model learning would cause the tracker to loose the target.

Visual Features for Tracking

The most desirable property of a visual feature, also called a visual cue, is its uniqueness and discernibility, so that the targets can be easily distinguished in the feature space. Feature selection

is closely related to target representation. For example, color is used as a feature for histogram-based appearance representations, while for contour-based representation, object edges are usually used as features.

Most of the visual tracking systems use a single visual feature and are often limited to a particular, simple visual environment. In complex environments, due to frequent changes, it is likely that no single visual feature will be robust and discriminant enough to successfully handle various scenes in the real-world. In such scenarios, a tracking approch utlizing multiple features is desirable. The visual features used by our approach are introduced below.

Color

The apparent color of an object is influenced primarily by two physical factors, (1) the characteristics of the light source and (2) the surface reflectance properties of the object. In image processing, the RGB colorspace is usually used to represent color. However, the RGB space is not a perceptually uniform color space and is highly sensitive to illumination changes. HSV colorspace, on the other hand, is approximately uniform in perception. The hue parameter in HSV space represents color information, which is illumination invariant as long as the following two conditions hold, (1) the light source color can be expected to be almost white, and (2) the saturation value of object color is sufficiently large [98].

In our tracking algorithm, we use the color model developed in [122]. The color model is obtained by histogramming techniques in the HSV color space in order to decouple chromatic information from shading effects. Since the color information is only reliable when both the saturation and the value are not too small, an HS histogram is populated with $N_h N_s$ bins using only the pixels with saturation and value larger than the hue and saturation thresholds. The remaining *color-free* pixels can however retain a crucial information when tracked regions are mainly black and white. Thus N_v additional value-only bins are populated with them. The resulting complete histogram is thus composed of $N = N_h N_s + N_v$ bins. The function $b_t(\mathbf{x}) \in \{1, \cdots, N\}$ denotes the bin index associated with the color of the pixel at location $\mathbf{x}$ in the current frame. We used the default setting $N_h = N_s = N_v = 8$ in all experiments.

Texture

Visual textures are the patterns in the intensity variations of a surface. The patterns can be the result of physical surface properties such as roughness, or they could be the result of reflectance differences

such as the color on a surface. Image texture is typically defined as a function of the spatial variation in pixel intensities. In our tracking algorithm, we use the texture model based on Local Binary Patterns (LBP) developed in [163]. The methodology has gained increasing attention and use, especially in applied research due to its performance and real-time processing capability. According to the authors, the most important property of the LBP operator in real-world applications is its tolerance against illumination changes, and its computational simplicity, which makes it possible to analyze images in real-time settings.

Probabilistic 3D Tracker

In this section, we present the details of our proposed probabilistic 3D tracker. First, we describe the target representation including the target state and the target model. Then, we define the similarity measure for target localization. We then present an algorithm to estimate target orientation, and finally we present the details of the proposed tracker based on particle filtering.

Target Representation

A target is characterized by a state vector and a reference model. The target state consists of the position, velocity and orientation of the target in 3D space. The reference target model, described below, consists of the 3D shape attributes, and the multi-view histograms of the target object in a suitable feature-space. Such a reference target model would correspond to the actual 3D target which does not change with scale variation and rotation. Once learned during the initialization phase, the model does not need to be updated or learned during tracking.

Target State

The state of a target is defined as

$$\boldsymbol{\chi} = [\mathbf{x}, \mathbf{v}, \boldsymbol{\theta}] \tag{63}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the position, $\mathbf{v} \in \mathbb{R}^3$ is the velocity, and $\boldsymbol{\theta}$ is the orientation of the target in 3D space. Specifically, we represent the target orientation as a unit quaternion, $\boldsymbol{\theta}$ [164]. Target orientation can also be represented using Direction Cosine Matrix (DCM), rotation vectors, or Euler angles. Standard conversions between different representations are available. We chose unit quaternions due to their intuitiveness, albegraic simplicity, and robustness. The target state evolution (the

target dynamics) is given by

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{v}_{t-1} \cdot dt + w_{\mathbf{x}} \\ \mathbf{v}_t &= \mathbf{v}_{t-1} + w_{\mathbf{v}} \\ \boldsymbol{\theta}_t &\equiv \boldsymbol{\theta}_{t-1} + w_{\boldsymbol{\theta}} \end{aligned} \tag{64}$$

where $w_{\mathbf{x}}$, $w_{\mathbf{v}}$, and $w_{\boldsymbol{\theta}}$ are the additive noise in target position, velocity and orientation, respectively.

Target Model

Since we want to model a 3D target, the definition of target model (see Equation (60)) as a single histogram on an image-plane is not sufficient. We extend the definition of the target model to include multiple histograms for a number of different viewpoints. This is called a multi-view histogram. Our target model is based on multi-view histograms in different feature-spaces.

The 3D target is represented by an ellipsoid region in 3D space. Without loss of generality, consider that the target is centered at $\mathbf{x}_0 = [0\ 0\ 0]^{\mathrm{T}}$, and the target axes are aligned with the world coordinate frame. The size of the ellipsoid is represented by the matrix

$$A = \begin{bmatrix} 1/l^2 & 0 & 0 \\ 0 & 1/w^2 & 0 \\ 0 & 0 & 1/h^2 \end{bmatrix} \tag{65}$$

where l, w, h represent the length, width and height of the ellipsoid. A set $\mathcal{S} = \{\mathbf{x}_i \ : \ \mathbf{x}_i^{\mathrm{T}} A \mathbf{x}_i = 1;\ \mathbf{x}_i \in \mathbb{R}^3\}$, is defined as the set of 3D points on the surface of the target. A function $b(\mathbf{x}_i) : \mathcal{S} \rightarrow \{1 \cdots m\}$ maps the surface point at location $\mathbf{x}_i$ to the index $b(\mathbf{x}_i)$ of its bin in the quantized feature space.

Let $\{\hat{\mathbf{e}}_j\}_{j=1\cdots N}$ be the unit vectors pointing away from the target center. These unit vectors are the viewpoints from where the target is viewed and the reference target model is defined in terms of these viewpoints. Finally, the reference target model is defined as

$$\mathbf{Q} = [\mathbf{q}_{\hat{\mathbf{e}}_1}^{\mathrm{T}}, \mathbf{q}_{\hat{\mathbf{e}}_2}^{\mathrm{T}}, \cdots \mathbf{q}_{\hat{\mathbf{e}}_N}^{\mathrm{T}}] \tag{66}$$

where $\mathbf{q}_{\hat{\mathbf{e}}_j}$ is the feature-histogram for viewpoint $\hat{\mathbf{e}}_j$, and N is the number of viewpoints. The feature histogram from viewpoint $\hat{\mathbf{e}}_j$ is defined as $\mathbf{q}_{\hat{\mathbf{e}}_j} = \{q_{\hat{\mathbf{e}}_j,u}\}_{u=1\cdots m}$

$$q_{\hat{\mathbf{e}}_j,u} = C \sum_{\mathbf{x}_i \in \mathcal{R}(\hat{\mathbf{e}}_j)} \kappa\big(d(\mathbf{y}_i)\big)\ \delta[b(\mathbf{x}_i) - u] \tag{67}$$

where δ is the Kronecker delta function, C is the normalization constant such that $\sum_{u=1}^{m} q_{\hat{\mathbf{e}}_j,u} = 1$, $\kappa(\cdot)$ is a weighting function, and

$$\mathcal{R}(\hat{\mathbf{e}}_j) = \{\mathbf{x}_i \ :\ \mathbf{x}_i \in \mathcal{S}, \mathbf{x}_i^{\mathrm{T}} A \hat{\mathbf{e}}_j \geq 0, \forall i \neq j \rightarrow \mathbf{y}_i \neq \mathbf{y}_j\} \tag{68}$$

is the set of points on the surface of the target that are visible from the viewpoint $\hat{\mathbf{e}}_j$. In equation (67), $\mathbf{y}_i = P_{\hat{\mathbf{e}}_j}\mathbf{x}_i$ denotes the pixel location corresponding to the point $\mathbf{x}_i$ projected on the image plane, where $P_{\hat{\mathbf{e}}_j}$ is the camera matrix for a hypothetical camera placed on vector $\hat{\mathbf{e}}_j$ with principal axis along $-\hat{\mathbf{e}}_j$. This camera matrix is defined as $P_{\hat{\mathbf{e}}_j} = K\big[R|t\big]$, where $R, \mathbf{t}$ are the rotation and translation given as

$$R = R_{\mathbf{x}}(\theta) R_{\mathbf{y}}(\phi) R_0$$

$$\theta = sin^{-1}(\hat{\mathbf{e}}_{j,z})$$

$$\phi = tan^{-1}\left(\frac{\hat{\mathbf{e}}_{j,y}}{\hat{\mathbf{e}}_{j,x}}\right)$$

$$R_0 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

where $R_{\mathbf{x}}(.), R_{\mathbf{y}}(.)$ are the basic rotation matrices along $x-$ and $y-$axis, θ and ϕ are zenith and azimuth angles, respectively, and R_0 is the base rotation. The translation vector $\mathbf{t}$ is given as

$$\mathbf{t} = -R\mathbf{x}_p$$

$$\mathbf{x}_p = L\hat{\mathbf{e}}_j$$

where $\mathbf{x}_p$ is the position of the hypothetical camera places on unit vector $\hat{\mathbf{e}}_j$ at a distance L from the target. The function $d(\mathbf{y}_i)$ in equation (67) computes pixel distance between pixel locations $\mathbf{y}_i$

and $\mathbf{y}_0$ as

$$d(\mathbf{y}_i) = \big(\mathbf{y}_i - \mathbf{y}_0\big)^{\mathrm{T}} B \big(\mathbf{y}_i - \mathbf{y}_0\big) \tag{69}$$

where $B \in \mathbb{R}^{2\times 2}$ is the representation of size of the ellipse-like shape when the target ellipsoid is projected on the image plane, and $\mathbf{y}_0 = P_{\hat{\mathbf{e}}_j}\mathbf{x}_0$.

Similarity Measure and Localization

Below, we describe the algorithm to compute the similarity measure between the reference target model and a target candidate state using the camera images from a network of cameras.

Consider a camera network of N cameras, where the cameras are denoted as C_n. The camera matrices are denoted as $P_n = K\big[R_n|\mathbf{t}_n\big]$, where K is the internal calibration matrix, R_n is the camera rotation and $\mathbf{t}_n$ is the camera translation. Consider an arbitrary target candidate state $\boldsymbol{\chi} = [\mathbf{x}, \mathbf{v}, \boldsymbol{\theta}]$, and let $\{I_n\}_{n=1\cdots N}$ be the images taken at the cameras at current time-step.

For the target candidate state $\boldsymbol{\chi}$, the *similarity measure* between the target candidate and the reference target model is computed based on the Bhattacharya Coefficient. The similarity measure is defined as

$$\rho(\boldsymbol{\chi}) = \prod_{n=1}^{N} \rho_n(\boldsymbol{\chi}) = \prod_{n=1}^{N} \rho\big(\mathbf{p}_n(\mathbf{x}), \mathbf{q}_{\hat{\mathbf{e}}_n}\big) \tag{70}$$

where N is the number of cameras, $\mathbf{p}_n(\mathbf{x})$ target candidate histogram at $\mathbf{x}$ from camera n, and $\mathbf{q}_{\hat{\mathbf{e}}_n}$ is the target model for the viewpoint $\hat{\mathbf{e}}_n$, where $\hat{\mathbf{e}}_n$ is the viewpoint closest to camera C_n's point-of-view. This is computed as

$$\hat{\mathbf{e}}_n = \arg\max_{\hat{\mathbf{e}}_j} \ \hat{\mathbf{e}}_{\text{target}}^{\mathrm{T}} R(\boldsymbol{\theta}) \hat{\mathbf{e}}_j \tag{71}$$

where $\hat{\mathbf{e}}_{\text{target}}$ is the camera viewpoint towards the target, $\boldsymbol{\theta}$ is the target orientation, and $R(\boldsymbol{\theta})$ is the rotation matrix for the target orientation given as

$$R(\boldsymbol{\theta}) = \begin{bmatrix} 1 - 2{q_2}^2 - 2{q_3}^2 & 2q_1q_2 - 2q_3q_4 & 2q_1q_3 + 2q_2q_4 \\ 2q_1q_2 + 2q_3q_4 & 1 - 2{q_1}^2 - 2{q_3}^2 & 2q_2q_3 - 2q_1q_4 \\ 2q_1q_3 - 2q_2q_4 & 2q_2q_3 + 2q_1q_4 & 1 - 2{q_1}^2 - 2{q_2}^2 \end{bmatrix} \tag{72}$$

where $\boldsymbol{\theta} \equiv [q_1, q_2, q_3, q_4]^{\mathrm{T}}$ is the unit quaternion. The unit vector $\hat{\mathbf{e}}_{\text{target}}$ is given by

$$\hat{\mathbf{e}}_{\text{target}} = \frac{\mathbf{x}_n - \mathbf{x}}{\| \mathbf{x}_n - \mathbf{x} \|} \tag{73}$$

where $\mathbf{x}_n$ is the camera position.

The target candidate histogram $\mathbf{p}_n(\mathbf{x})$ in Equation (70), is computed in a similar way as that for the target model histogram. The target candidate histogram for camera C_n is given by

$$\mathbf{p}_n(\mathbf{x}) = \{p_{n,u}(\mathbf{x})\}_{u=1\cdots m} \tag{74}$$

where

$$p_{n,u}(\mathbf{x}) = C \sum_{\mathbf{y}_i \in \mathcal{R}(\mathbf{x})} \kappa\big(d(\mathbf{y}_i, \mathbf{y})\big)\ \delta[b_I(\mathbf{y}_i) - u] \tag{75}$$

where C is the normalization constant such that $\sum_{u=1}^{m} p_{n,u} = 1$, $\kappa(.)$ is the weighting function, and

$$\mathcal{R}(\mathbf{x}) = \{\mathbf{y}_i \ :\ \mathbf{y}_i \in \mathcal{I}, (\mathbf{y}_i - \mathbf{y})^{\mathrm{T}} B(\mathbf{x})(\mathbf{y}_i - \mathbf{y}) \leq 1, \forall i \neq j \rightarrow \mathbf{y}_i \neq \mathbf{y}_j\}$$

is the set of pixels in the region around $\mathbf{y}$, defined as $B(\mathbf{x})$. Here, $\mathbf{y} = P_n\mathbf{x}$ is the projection of the target position on the camera image plane. The function $d(\mathbf{y}_i, \mathbf{y})$ computes pixel distance between pixel locations $\mathbf{y}_i$ and $\mathbf{y}$ as follows,

$$d(\mathbf{y}_i, \mathbf{y}) = \big(\mathbf{y}_i - \mathbf{y}\big)^{\mathrm{T}} B(\mathbf{x}) \big(\mathbf{y}_i - \mathbf{y}\big) \tag{76}$$

where $B(\mathbf{x}) \in \mathbb{R}^{2\times 2}$ is the representation of the size of the ellipse-like shape when the target ellipsoid is projected on the camera image plane.

Estimation of Target Orientation

Target orientation is estimated separately from the target position. Below, we describe our algorithm to estimate the target quaternion using the data from multiple cameras. In the first step, we estimate the target quaternion at each camera separately. In the second step, the individual target quaternions are fused together to get a global estimate of the target quaternion.

In the first step, on each camera, we compute the similarity measure of the target candidate histogram, $\mathbf{p}_n(\mathbf{x})$, with each of the histograms in the target reference model (Equation (66))

$$\boldsymbol{\rho}(\boldsymbol{\chi}) \equiv \big[\rho_1(\boldsymbol{\chi}), \rho_2(\boldsymbol{\chi}), \cdots \rho_N(\boldsymbol{\chi})\big]$$
$$\rho_j(\boldsymbol{\chi}) \equiv \rho\big(\mathbf{p}(\mathbf{x}), \mathbf{q}_{\hat{\mathbf{e}}_j}\big)$$

where $\rho\big(\mathbf{p}(\mathbf{x}), \mathbf{q}_{\hat{\mathbf{e}}_j}\big)$ is the Bhattacharya Coefficient. Now, we have viewpoints $(\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \cdots, \hat{\mathbf{e}}_N)$ and similarity measures $(\rho_1, \cdots, \rho_N)$ along each viewpoint. We take the weighted average of all the viewpoints to get the most probable direction of the camera with respect to the target

$$\hat{\mathbf{e}}_{\text{avg}} = \frac{\sum_j \rho_j \hat{\mathbf{e}}_j}{\sum_j \rho_j}$$

The unit vector $-\hat{\mathbf{e}}_{\text{avg}}$ is the estimate of the camera principal axis in the target's frame of reference. To estimate the target rotation vector, we need to compute the transformation between $-\hat{\mathbf{e}}_{avg}$ and $\hat{z}_{\text{CAM}}$, where $\hat{z}_{\text{CAM}}$ is the actual camera principal axis, and apply the same transformation to the target axes, $\mathbf{T} \equiv \mathbf{I}_{3\times 3}$, where $\mathbf{I}_{3\times 3}$ is the identity matrix of size 3.

The transformation between the two unit vectors can be computed as follows,

$$\hat{\mathbf{a}} = \frac{-\hat{\mathbf{e}}_{avg} \times \hat{z}_{\text{CAM}}}{\parallel \hat{\mathbf{e}}_{avg} \times \hat{z}_{\text{CAM}} \parallel}$$
$$\phi = \cos^{-1}\big(-\hat{\mathbf{e}}_{\text{avg}} \cdot \hat{\text{z}}_{\text{CAM}}\big)$$

where $\hat{\mathbf{a}}$ is the Euler axis and ϕ is the rotation angle. Using this transformation, the transformed target axes are

$$T \equiv R_{\hat{\mathbf{a}}}(\phi) = [\hat{\mathbf{e}}_{x'}^{\text{T}}\ \hat{\mathbf{e}}_{y'}^{\text{T}}\ \hat{\mathbf{e}}_{z'}^{\text{T}}] \tag{77}$$

The target orientation on each node is computed using the following conversion from Euler axis and rotation angle to quaternion

$$\hat{\boldsymbol{\theta}}_n = \begin{bmatrix} a_{n,x}\sin(\phi_{\text{n}}/2) \\ a_{n,y}\sin(\phi_{\text{n}}/2) \\ a_{n,z}\sin(\phi_{\text{n}}/2) \\ \cos(\phi_{\text{n}}/2) \end{bmatrix} \tag{78}$$

In the second step, after we have estimated the target quaternions on each of the cameras, we fuse the quaternions together to get a global estimate of the target quaternion. Given target quaternion estimates $\{\hat{\boldsymbol{\theta}}_n\}_{n=1\cdots N}$ and weights $\{w_n\}_{n=1\cdots N}$ from N cameras, we estimate the global target quaternion by taking the weighted average

$$\hat{\boldsymbol{\theta}}_{all} = \frac{\sum_n w_n \hat{\boldsymbol{\theta}}_n}{\| \sum_n w_n \hat{\boldsymbol{\theta}}_n \|}$$

The current target orientation is updated using the global target orientation estimated from the camera images as

$$\hat{\boldsymbol{\theta}} = \alpha \hat{\boldsymbol{\theta}}_{all} + (1-\alpha)\hat{\boldsymbol{\theta}}_{prior}$$

where $\hat{\boldsymbol{\theta}}_{prior}$ is the prior target orientation and α is an update factor.

Tracking Algorithm

In this section, we discuss the implementation of our base tracker.

Base Tracker (T0)

Our probabilistic 3D tracker is based on sequential Bayesian estimation. In Bayesian estimation, the target state is estimated by computing the posterior probability density $p(x_{t+1}|z_{0:t+1})$ using a Bayesian filter described by

$$p(x_{t+1}|z_{0:t+1}) \propto p(z_{t+1}|x_{t+1}) \int_{x_t} p(x_{t+1}|x_t) p(x_t|z_{0:t}) dx_t \tag{79}$$

where $p(x_t|z_{0:t})$ is the prior density, $p(z_{t+1}|x_{t+1})$ is the likelihood given the target state, and $p(x_{t+1}|x_t)$ is the prediction for the target state x_{t+1} given the current state x_t according to a target state evolution model. Here $z_{0:t} \equiv (z_0, \cdots, z_t)^{\mathrm{T}}$ denote all the measurements up until t.

In sequential Bayesian estimation, the target state estimate can be updated as the data is made available using only the target state estimate from the previous step, unlike Bayesian estimation, where the target state estimate is updated using all the data collected up until the current time-step. In sequential Bayesian estimation, the target state is estimated by computing the posterior

probability density $p(x_{t+1}|z_{t+1}, x_t)$ using a sequential Bayesian filter described by

$$p(x_{t+1}|z_{t+1}, x_t) \propto p(z_{t+1}|x_{t+1})p(x_{t+1}|x_t)p(x_t) \tag{80}$$

where $p(x_t)$ is the prior density from the previous step, $p(z_{t+1}|x_{t+1})$ is the likelihood given the target state, and $p(x_{t+1}|xt)$ is the prediction for the target state x_{t+1} given the current state x_t according to a target state evolution model.

In visual tracking problems the likelihood is non-linear and often multi-modal. As a result linear filters such as the Kalman filter, and its approximations are usually not suitable. Our base tracker is implemented using particle filters. Particle filters can handle multiple hypotheses and non-linear systems. Our probabilistic base tracker is summarized in Algorithm 2.

Algorithm 2 Base tracker

Input: The reference target model $\mathbf{Q}$ (Equation (66)), and target state $\hat{\boldsymbol{\chi}}_0 = [\hat{\mathbf{x}}_0, \hat{\mathbf{v}}_0, \hat{\boldsymbol{\theta}}_0]$ in previous time-step.
On Each Camera Node: $\{C_n\}_{n=1\cdots N}$
    target position estimation
Generate *synchronized* particle set for the target position, $\{\tilde{\mathbf{x}}_i\}_{i=1\cdots M} \sim \mathcal{N}(\hat{\mathbf{x}}_0 + \hat{\mathbf{v}}_0, \Sigma)$,
For $i = 1 \cdots M$, compute target candidate histogram, $\mathbf{p}_n(\tilde{\mathbf{x}}_i)$ according to Equation (74),
For $i = 1 \cdots M$, compute weights $w_{n,i} = \rho_n(\tilde{\mathbf{x}}_i)$ according to Equation (70),
    target orientation estimation
Estimate target orientation $\hat{\boldsymbol{\theta}}_n$ according to Equation (78)
On Base Station:
For $i = 1 \cdots M$, combine weights from each camera, $w_i = \prod_n w_{n,i}$,
Compute target position estimate, $\hat{\mathbf{x}}$
Compute target velocity estimate, $\hat{\mathbf{v}}$ using Kalman filter,
Estimate target orientation $\boldsymbol{\theta}$ according to Equation (79).

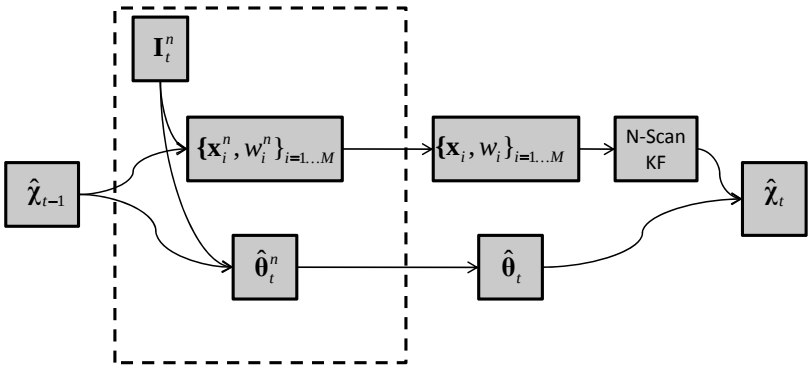


Figure 35: Proposed base tracker T0 for video tracking.

Figure 35 illustrates the base tracker operation for a single time-step. At each time step, each camera node performs position estimation and orientation estimation separately. For position es-

timation, we generate a set of *synchronized* particles for the predicted position. The synchronized particle set is generated using a synchronized random number generator, which can be achieved by seeding the random number generator on each node with the same seed. Then, target candidate histograms are computed for each of the proposed particles. In our framework, we use color features, specifically HS colorspace, and texture features, specifically LBP. After we compute the target candidate histograms in HS-space and LBP-space for each particle, we compute the weights according to the following

$$\rho(\mathbf{x}) = \alpha_{\text{HS}}\rho_{\text{HS}}(\mathbf{x}) + (1 - \alpha_{\text{HS}})\rho_{\text{LBP}}(\mathbf{x}) \tag{81}$$

where $\rho_{\text{HS}}(\mathbf{x})$ and $\rho_{\text{LBP}}(\mathbf{x})$ are the similarity measures for the target candidate histograms in HS- and LBP-spaces, respectively (computed using Equation (70)), and $0 \leq \alpha_{\text{HS}} \leq 1$ is the weighting factor. Target orientation estimation is performed on each camera node according to the algorithm described earlier in the section.

Then, the weights of the *synchronized* particle set from each of the camera nodes are sent to the base-station, and are combined together. An MMSE, MLE or MAP estimate is estimated as

$$\text{MMSE: } \hat{\mathbf{x}} = \frac{\sum_i w_i \mathbf{x}_i}{\sum_i w_i} \tag{82}$$

$$\text{MLE: } \hat{\mathbf{x}} = \arg\max_{\mathbf{x}} w_i \tag{83}$$

$$\text{MAP: } \hat{\mathbf{x}} = \arg\max_{\mathbf{x}} w_i \mathcal{N}(\mathbf{x}_i | \mathbf{x}_0, \Sigma_0) \tag{84}$$

The target position estimate is then used in a *N-scan Kalman smoother* [165] to smooth the position estimates, as well as to estimate the target velocity. Finally, target orientation esimates from each camera node are combined according to Equation (79) to estimate global target orientation.

Computational Cost Each camera node generates a synchronized particle set of size M and computes weights for each particle. Computation of weight for a single particle includes computing the target candidate histogram and the Bhattacharya coefficient with the reference target model. Hence, the total computational cost at each time-step on each camera node is $O(Mmn_p)$, where M is the number of particles, m is the number of bins in the quantized feature-space, and n_p is the number of pixels inside the target candidate region.

Communication Cost The base tracker algorithm requires the base station to transmit the current target state to each camera node. Each camera node then transmits the weights for the *synchronized* particle set back to the base station, as well as the target orientation estimates. During a single time-step, the total cost of communication can be computed as follows. Let the size of the target state $\chi = \{\mathbf{x}, \mathbf{v}, \boldsymbol{\theta}\}$ be $24 + 24 + 32 = 80$ bytes, and each particle weight be represented by a 8 byte double. Then, the total cost of communication during a single step is $C = |\chi| + MN|w| = 80 + 8MN$ bytes, where N is the number of cameras and M is the number of particles in the synchronized particle filter. For example, if $N = 4$ and $M = 1000$, then the total cost is $C = 32,080$ bytes, or approximately 32 kb-per-time-step. If the application is running at 4 Hz, then the total bandwidth consumption would be around 128 kbps.

Tracker Variations

In this subsection, we introduce four variations to the base tracker. The approaches incur different computational and communication costs on each node, and produce different tracking accuracies. The computational and communication costs associated with a tracker directly affects the tracking rate, network size, bandwidth consumption, latency, etc. In other words, different trackers support different Quality-of-Service (QoS). Since the objective of a tracking algorithm is to effectively track a target, which is the information that is desired from a tracking algorithm, we can say that different trackers support different Quality-of-Information (QoI). After the description of all the tracker variations, we will qualitatively classify the trackers according to their supported QoS and QoI.

Tracker T1: 3D Kernel Density Estimate

In the base tracker, the communication cost is very high because we send weights for the synchronized particle set from each node to the base station. In this variation of the base tracker, instead of sending all the weights, each node computes a 3D kernel density estimate, approximates the kernel density estimate using a Gaussian Mixture Model (GMM), and sends only the mixture model parameters to the base station, thereby reducing the communication cost by a large factor. Figure 36 shows the tracker. In tracker T1, the main differences from tracker T0 are: 1) computation of the 3D kernel density from the particle set, 2) GMM approximation of the kernel density, and 3) state estimation at the base station using GMM parameters from all nodes.

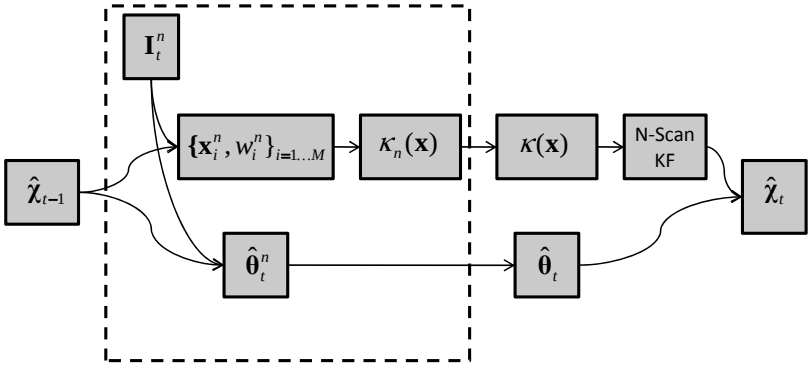


Figure 36: Tracker T1. The tracker uses 3D kernel density estimate.

The kernel density in 3D space is computed as follows

$$\kappa(\mathbf{x}) = \sum_{i=1}^{N} k\left(\frac{\| \mathbf{x} - \mathbf{x}_i \|^2}{h^2}\right) \tag{85}$$

where $k(x) = \exp(-x/2) : [0, \infty) \rightarrow \mathbb{R}$ is the kernel profile function. The 3D kernel density $\kappa(\mathbf{x})$ is approximated as a 3D-GMM of appropriate model-order (number of mixture components). A model-order selection algorithm is used to select an optimal model-order that best matches the kernel density. The best matching is done according to KL-divergence as follows

$$m_{opt} = \arg \min_{m \leq m_{\text{MAX}}} \text{KL}\big(\kappa(\mathbf{x}) || g_m(\mathbf{x})\big) \tag{86}$$

where $g_m(\mathbf{x}) \equiv \{\alpha_i, \mu_i, \Sigma_i\}_{i=1\cdots m}$ is the 3D-GMM of order m (estimated using the EM algorithm [166]), $\text{KL}(\kappa(\mathbf{x})||g_m(\mathbf{x}))$ is the KL-divergence of $g_m(\mathbf{x})$ from $\kappa(\mathbf{x})$, and m_{opt} is the optimal model-order. Finally, the state estimation is done at the base station by mode estimation on the combined kernel density from all the nodes

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}} \kappa(\mathbf{x}) \equiv \arg \max_{\mathbf{x}} \sum_{i=1}^{m_{opt}} \alpha_i \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i) \tag{87}$$

Tracker T2: In-Network Aggregation

Further improvements in terms of the communication cost can be made by in-network aggregation. In this tracker, instead of each camera node sending the mixture model parameters to the base station, *in-nodes* in the routing tree aggregate the mixture model parameters and forward a fewer number of parameters.

Figure 37 shows the tracker. In-network aggregation is done in two-steps. In the first step, the

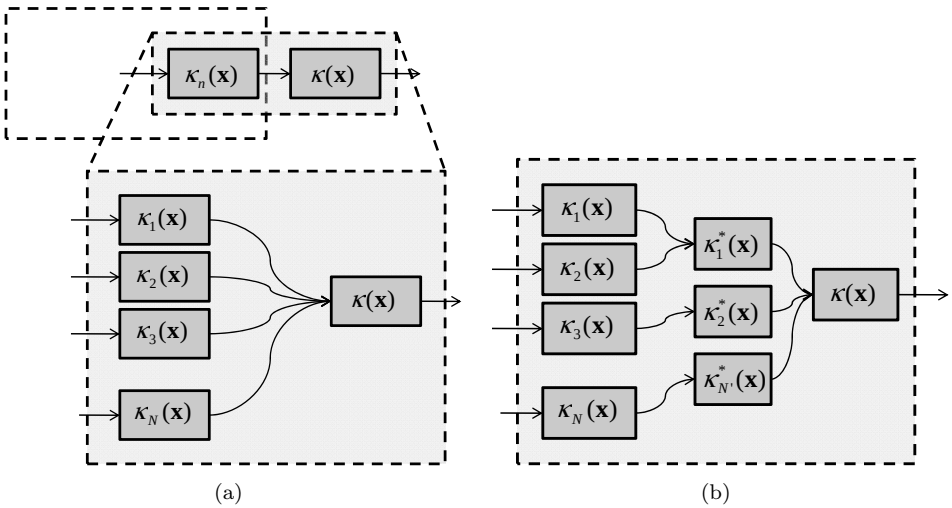


Figure 37: Tracker T2 design. (a) Tracker T1 does not include in-network aggregation, whereas (b) tracker T2 includes in-network aggregation.

GMMs from multiple camera nodes are combined by taking the product of the GMMs. In the second step, we reduce the number of mixture components in the resulting GMM from the first step to fit the size of a message of fixed size.

Step 1: Product of GMMs.

Let $\{\kappa_j(\mathbf{x})\}_{j=1\cdots N}$ be the kernel densities available at a node from its children and itself. Then, the combined kernel density is given by

$$\kappa(\mathbf{x}) = \kappa_1(\mathbf{x}) \cdot \kappa_2(\mathbf{x}) \cdots \kappa_N(\mathbf{x}) \tag{88}$$

Without loss of generality, we can perform successive pairwise product operations to obtain the combined density. Lets consider the product of two GMMs, $\kappa_1(\mathbf{x})$ and $\kappa_2(\mathbf{x})$ given as

$$\kappa_1(\mathbf{x}) = \sum_{i=1}^{N_1} \alpha_i \mathcal{N}(\mu_i, V_i)$$
$$\kappa_2(\mathbf{x}) = \sum_{j=1}^{N_2} \beta_j \mathcal{N}(\lambda_j, W_j)$$

The product GMM is given by

$$\begin{aligned}\kappa(\mathbf{x}) &= \kappa_1(\mathbf{x})\kappa_2(\mathbf{x}) \\ &= \sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\alpha_i\beta_j\mathcal{N}(\mu_i, V_i)\mathcal{N}(\lambda_j, W_j) \\ &= \sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\gamma_{ij}\mathcal{N}(\xi_{ij}, \Sigma_{ij})\end{aligned}$$

where,

$$\begin{aligned}\Sigma_{ij} &= \left(V_i^{-1} + W_i^{-1}\right)^{-1} \\ \xi_{ij} &= \Sigma_{ij}\left(V_i^{-1}\mu_i + W_j^{-1}\lambda_j\right) \\ \gamma_{ij} &= \left[\frac{|\Sigma_{ij}|}{|V_i||W_j|}\right]^{1/2}\frac{\exp(-z_c/2)}{(2\pi)^{D/2}}\alpha_i\beta_j \\ z_c &= \mu_i^{\mathrm{T}}V_i^{-1}\mu_i + \lambda_j^{\mathrm{T}}W_j^{-1}\lambda_j - \xi_{ij}^{\mathrm{T}}\Sigma_{ij}^{-1}\xi_{ij}\end{aligned}$$

So, given two GMMs with N_1 and N_2 mixture components, the product will be a GMM with N_1N_2 mixture components.

Step 2: Model-Order Reduction.

To reduce the number of components in the product GMM such that the mixture model parameters fit a communication message of fixed size is achieved by using a modified k-means algorithm. The model-order reduction problem can be stated as follows. Given a GMM with N components, we want to estimate parameters of a GMM with K components ($K < N$) such that the reduced-order GMM *faithfully* represents the original GMM. In other words,

$$\kappa(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i\mathcal{N}(\mu_i, V_i) \equiv \sum_{j=1}^{K}\beta_j\mathcal{N}(\lambda_j, W_j) \tag{89}$$

In the modified k-means algorithm, we want to cluster N points, which are the mixture model components, in K clusters. In the description of the algorithm, we will interchangeably use the terms points and mixture model components. The two key modifications in the standard k-means algorithm are: 1) computation of the distance between points, and 2) the clusterhead update algorithm. First, initialize the k-means algorithm using K random points, $\left(\beta_j^0, \lambda_j^0, W_j^0\right) = (\alpha_i, \mu_i, V_i)$, where $j = 1 \cdots K$ and $i = \textsc{random}(N)$. Then, compute the modified distance of all the points, $i = 1 \cdots N$,

with the K clusterheads as

$$d_{ij} = (\mu_i - \lambda_j^0)^{\mathrm{T}}(V_i^{-1} + W_j^{0^{-1}})(\mu_i - \lambda_j^0) \tag{90}$$

and associate each point with the clusterhead closest to it, $m_i = \arg\min_j d_{ij}$, where m_i is the index of the clusterhead closest to the i^{th} point. Then, move the clusterheads to the centroid of the cluster (defined as the collection of all the points associated with the cluster). For $j = 1 \cdots K$, let $\mathcal{C}_j$ represent the set of points in the j^{th} cluster. Then, update the clusterheads according to

$$\beta_j = \sum_{i \in \mathcal{C}_j} \alpha_i \tag{91}$$

$$\lambda_j = \frac{1}{\beta_j} \sum_{i \in \mathcal{C}_j} \alpha_i \mu_i \tag{92}$$

$$W_j = \frac{1}{\beta_j} \sum_{i \in \mathcal{C}_j} \alpha_i (V_i + \mu_i^{\mathrm{T}} \mu_i) - \lambda_j^{\mathrm{T}} \lambda_j \tag{93}$$

Finally, the termination criteria is to stop when the clusterheads are converged, $\sum_j \parallel \beta_j - \beta_j^0 \parallel \leq \epsilon$, or the algorithm has exceeded a maximum number of iterations.

Tracker T3: Image-Plane Particle Filter & 3D Kernel Density

In trackers T1 and T2, we made improvements in terms of the communication cost by approximating data likelihood by 3D kernel density estimate which is implemented and approximated as GMMs. The computational cost of computing such a kernel density is still high because the particle filter is run in 3D space. An improvement in terms of the computational cost can be made by employing a 2D particle filter (in the camera image-plane) and computing a 3D kernel density from it.

In this variation, instead of generating the particle set of 3D target positions, each camera node generates a particle set of 2D pixel positions in the image-plane. Each camera then computes a 3D kernel density from the image-plane particle set and proceeds as in the case of tracker T2. Figure 38 shows the tracker. The main differences of tracker T3 with tracker T2 are: 1) the 2D particle filter, and 2) the algorithm to compute 3D kernel density using 2D particles. The algorithm for 2D particle filter is described below.

The target candidate histogram $\mathbf{p}_n(\mathbf{y})$, and hence the weights for image-plane particle filter are computed a little differently from that in the case of the 3D particle filter. The target candidate

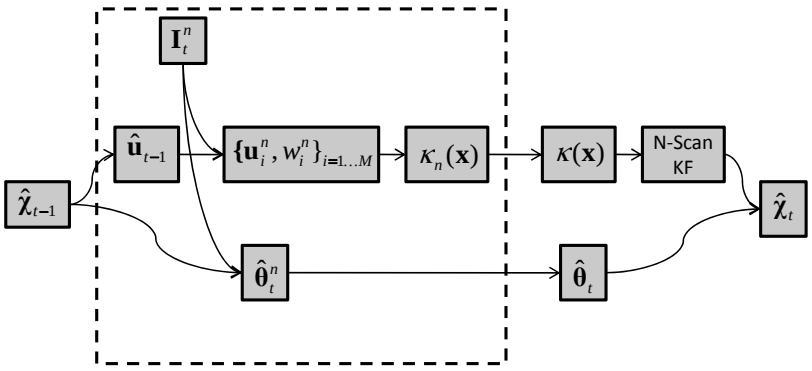


Figure 38: Tracker T3 design. It includes the use if 2D particle filter, or image-plane filtering, and & 3D kernel density estimate.

Algorithm 3 T3 tracker

Input: The reference target model $\mathbf{Q}$ (Equation (66)), and target state $\hat{\boldsymbol{\chi}}_0 = [\hat{\mathbf{x}}_0, \hat{\mathbf{v}}_0, \hat{\boldsymbol{\theta}}_0]$ in previous time-step.
On Each Camera Node: $\{C_n\}_{n=1\cdots N}$
target position estimation
Project target position on image plane, $\hat{\mathbf{y}}_0 = P_n(\hat{\mathbf{x}}_0 + \hat{\mathbf{v}}_0)$.
Generate particle set for the target position (in pixel locations), $\{\tilde{\mathbf{y}}_i\}_{i=1\cdots M} \sim \mathcal{N}(\hat{\mathbf{y}}_0, \Sigma)$,
For $i = 1 \cdots M$, compute target candidate histogram $\mathbf{p}_n(\tilde{\mathbf{y}}_i)$ according to Equation (94),
For $i = 1 \cdots M$, compute weights $w_{n,i} = \rho_n(\tilde{\mathbf{y}}_i)$ according to Equation (98),
target orientation estimation
Estimate target orientation $[\hat{\boldsymbol{\theta}}_n]$ according to Equation (78)
On Base Station:
Compute target position estimate $\hat{\mathbf{x}}$.
Compute target velocity estimate $\hat{\mathbf{v}}$ using Kalman filter,
Estimate target rotation vector $\boldsymbol{\theta}$ according to Equation (79).

feature histogram for camera C_n is given by

$$\mathbf{p}_n(\mathbf{y}) = \{p_{n,u}(\mathbf{y})\}_{u=1\cdots m} \tag{94}$$

where

$$p_{n,u}(\mathbf{y}) = C \sum_{\mathbf{y}_i \in \mathcal{R}(\mathbf{y})} \kappa\big(d(\mathbf{y}_i, \mathbf{y})\big)\ \delta[b_I(\mathbf{y}_i) - u] \tag{95}$$

and C is the normalization constant such that $\sum_{u=1}^{m} p_{n,u} = 1$, and

$$\mathcal{R}(\mathbf{y}) = \{\mathbf{y}_i\ :\ \mathbf{y}_i \in \mathcal{I}, \mathbf{y}_i^{\mathrm{T}} B(\mathbf{y}) \mathbf{y}_i \leq 1, \forall i \neq j \rightarrow \mathbf{y}_i \neq \mathbf{y}_j\} \tag{96}$$

is the set of pixels in the camera image I in the region defined by $B(\mathbf{y})$ around the pixel location $\mathbf{y}$. The function $d(\mathbf{y}_i, \mathbf{y})$ computes pixel distance between pixel locations $\mathbf{y}_i$ and $\mathbf{y}$ as follows,

$$d(\mathbf{y}_i, \mathbf{y}) = \big(\mathbf{y}_i - \mathbf{y}\big)^{\mathrm{T}} B(\mathbf{y}) \big(\mathbf{y}_i - \mathbf{y}\big) \tag{97}$$

where $B(\mathbf{y}) \in \mathbb{R}^{2\times 2}$ is the representation of the size of the elliptical region on the image plane around the pixel location $\mathbf{y}$. Finally, the weights are computed as

$$w_{i,n} = \rho_n(\tilde{\mathbf{y}}_i) = \rho(\mathbf{p}_n(\tilde{\mathbf{y}}_i), \mathbf{q}_{\hat{\mathbf{e}}_n}) \tag{98}$$

where $\mathbf{q}_{\hat{\mathbf{e}}_n}$ is the target model for the viewpoint $\hat{\mathbf{e}}_n$ (see Equation (71)) that is closest to camera C_n's point-of-view.

Tracker T4: Image-Plane Kernel Density

A different variation of tracker T3 is possible if, instead of computing the 3D kernel density from the image-plane particle set, each camera node computes the image-plane (2D) kernel density. Then, on the base-station, the target state is estimated using image-plane kernel densities from each camera node. In this chapter, we call it image-plane filtering. Figure 39 shows the tracker. The main differences from tracker T3 are: 1) computation of image-plane kernel density from the image-plane particle set, and 2) target state estimate using image-plane kernel densities from multiple cameras.

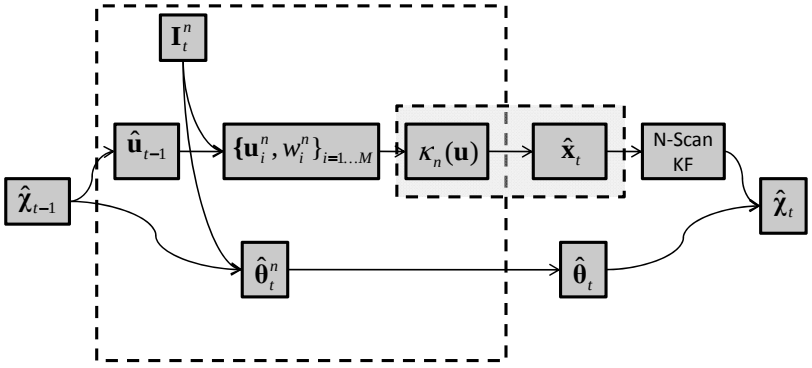


Figure 39: Tracker T4 design. It includes image-plane filtering, as well as mode-estimation using the image-plane kernel densities.

The target state estimation using image-plane kernel densities is performed using a mode estimation algorithm which is described below. First, let us denote $\mathbf{x} \in \mathbb{R}^4$ as the target position in the homogeneous 3D world coordinate and $\mathbf{y} \in \mathbb{R}^3$ as the target position in the image affine coordinate system. Then, it follows that $\mathbf{y} = P\mathbf{x}$, where P is the camera matrix. The posterior density for

target position is given by

$$p(\mathbf{x}) = p_0(\mathbf{x}|\mathbf{x}_0, \Sigma_0) \prod_{i=1}^{N} \gamma_i p_i(\mathbf{x}) \tag{99}$$

where $p_i(\mathbf{x})$ is the likelihood density at camera node C_i, which is a mixture model given by

$$p_i(\mathbf{x}) = \sum_{j=1}^{K_i} \alpha_{ij} p_{ij}(\mathbf{x}) \tag{100}$$

where $p_{ij}(\mathbf{x})$ are mixture components given by the following Gaussian density

$$p_{ij}(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma_{ij}|^{D/2}} \exp\left(-\frac{1}{2}(\mathbf{y}_i^\dagger - \mathbf{u}_{ij})^{\mathrm{T}} \Sigma_{ij}^{-1} (\mathbf{y}_i^\dagger - \mathbf{u}_{ij})\right) \tag{101}$$

where $\mathbf{y}_i^\dagger = \frac{P_i \mathbf{x}}{p_{i3}^{\mathrm{T}} \mathbf{x}}$ is the target position in the image plane of camera node C_i. Rewriting the expression for the above density, we have

$$\begin{aligned} p_{ij}(\mathbf{x}) &= K_{ij} \exp\left(-\frac{1}{2} \frac{(P_i \mathbf{x} - \mathbf{u}_{ij} p_{i3}^{\mathrm{T}} \mathbf{x})^{\mathrm{T}}}{p_{i3}^{\mathrm{T}} \mathbf{x}} \Sigma_{ij}^{-1} \frac{(P_i \mathbf{x} - \mathbf{u}_{ij} p_{i3}^{\mathrm{T}} \mathbf{x})}{p_{i3}^{\mathrm{T}} \mathbf{x}}\right) \\ &= K_{ij} \exp\left(-\frac{1}{2} (Q_{ij} \mathbf{x})^{\mathrm{T}} \Sigma_{ij}^{-1} (Q_{ij} \mathbf{x})\right) \\ &= K_{ij} \exp\left(-\frac{1}{2} \mathbf{x}^{\mathrm{T}} Q_{ij}^{\mathrm{T}} \Sigma_{ij}^{-1} Q_{ij} \mathbf{x}\right) \end{aligned}$$

where

$$Q_{ij} = \frac{P_i - \mathbf{u}_{ij} p_{i3}^{\mathrm{T}}}{p_{i3}^{\mathrm{T}} \mathbf{x}}$$

The MAP estimate can be obtained by maximizing the posterior density. Taking the derivative of the posterior density, we get

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} p_0(\mathbf{x}) \prod_{i=1}^{N} \gamma_i p_i(\mathbf{x}) \\ + p(\mathbf{x}) \sum_{i=1}^{N} \frac{1}{p_i(\mathbf{x})} \frac{\partial}{\partial \mathbf{x}} p_i(\mathbf{x}) \end{aligned} \tag{102}$$

where

$$\frac{\partial}{\partial \mathbf{x}} p_0(\mathbf{x}) = -p_0(\mathbf{x}) \left[(\mathbf{x} - \mathbf{x}_0)^{\mathrm{T}} \Sigma_0^{-1} \right]$$
$$\frac{\partial}{\partial \mathbf{x}} p_i(\mathbf{x}) = -\sum_{j=1}^{K_i} \alpha_{ij} p_{ij}(\mathbf{x}) \left[\mathbf{x}^{\mathrm{T}} Q_{ij}^{\mathrm{T}} \Sigma_0^{-1} Q_{ij} (I_4 - \frac{\mathbf{x} p_{i3}^{\mathrm{T}}}{p_{i3}^{\mathrm{T}} \mathbf{x}}) \right]$$

Simplifying the expression for $\frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}) = 0$, and substituting the following

$$\beta_{ij} = \alpha_{ij} p_{ij}(\mathbf{x})$$
$$R_{ij} = Q_{ij}^{\mathrm{T}} \Sigma_{ij}^{-1} Q_{ij} \big(I_4 - \frac{\mathbf{x} p_{i3}^{\mathrm{T}}}{p_{i3}^{\mathrm{T}} \mathbf{x}} \big)$$

we have

$$\frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}) = -p_0(\mathbf{x}) \left[(\mathbf{x} - \mathbf{x}_0)^{\mathrm{T}} \Sigma_0^{-1} \right] \prod_{j=1}^{N} \gamma_i p_i(\mathbf{x}) - p_0(\mathbf{x}) \prod_{i=1}^{N} \gamma_i p_i(\mathbf{x}) \sum_{i=1}^{N} \frac{1}{p_i(\mathbf{x})} \sum_{j=1}^{K_i} \beta_{ij} \mathbf{x}^{\mathrm{T}} R_{ij} = 0$$

Further simplifying and substituting the following

$$R_i = \gamma_i \sum_{j=1}^{K_i} \beta_{ij} R_{ij}$$

we have

$$(\mathbf{x} - \mathbf{x}_0)^{\mathrm{T}} \Sigma_0^{-1} + \sum_{i=1}^{N} \mathbf{x}^{\mathrm{T}} \frac{R_i}{p_i(\mathbf{x})} = 0$$

Solving for $\mathbf{x}^{\mathrm{T}}$, we have the approximate MAP estimate of the target location as

$$\hat{\mathbf{x}}^{\mathrm{T}} = \mathbf{x}_0^{\mathrm{T}} \Sigma_0^{-1} \left[\Sigma_0^{-1} + \sum_{i=1}^{N} \frac{R_i}{p_i(\mathbf{x}_0)} \right]^{-1} \tag{103}$$

Comparison of all trackers

Table 12 provides a qualitative comparison of all the trackers. A quantitative comparison of the trackers is included in the evaluation section. The table compares the trackers in terms of supported QoI, which includes the tracking accuracy, in terms of supported QoS, which includes the computational and communication costs, and their robustness to the size of the target in pixels.

Table 12: Qualitative comparison proposed video trackers.

Tracker	Quality-of-Information (QoI)	Quality-of-Service (QoS)		Robustness to size in pixels
	tracking accuracy	number of particles (computational cost)	message size (communication cost)	
Tracker P: (2D-to-3D)	POOR	LOW	LOW	NO
Base Tracker T0: (Sync3DPF)	GOOD	HIGH	VERY HIGH	YES
Tracker T1: (3DPF & 3DKD)	MEDIUM	HIGH	MEDIUM	YES
Tracker T2: (3DPF & 3DKD & NetAggr)	MEDIUM	HIGH	LOW	YES
Tracker T3: (2DPF + 3DKD + NetAggr)	GOOD	MEDIUM	LOW	NO
Tracker T4: (2DPF & 2DKD)	GOOD	MEDIUM	MEDIUM	NO

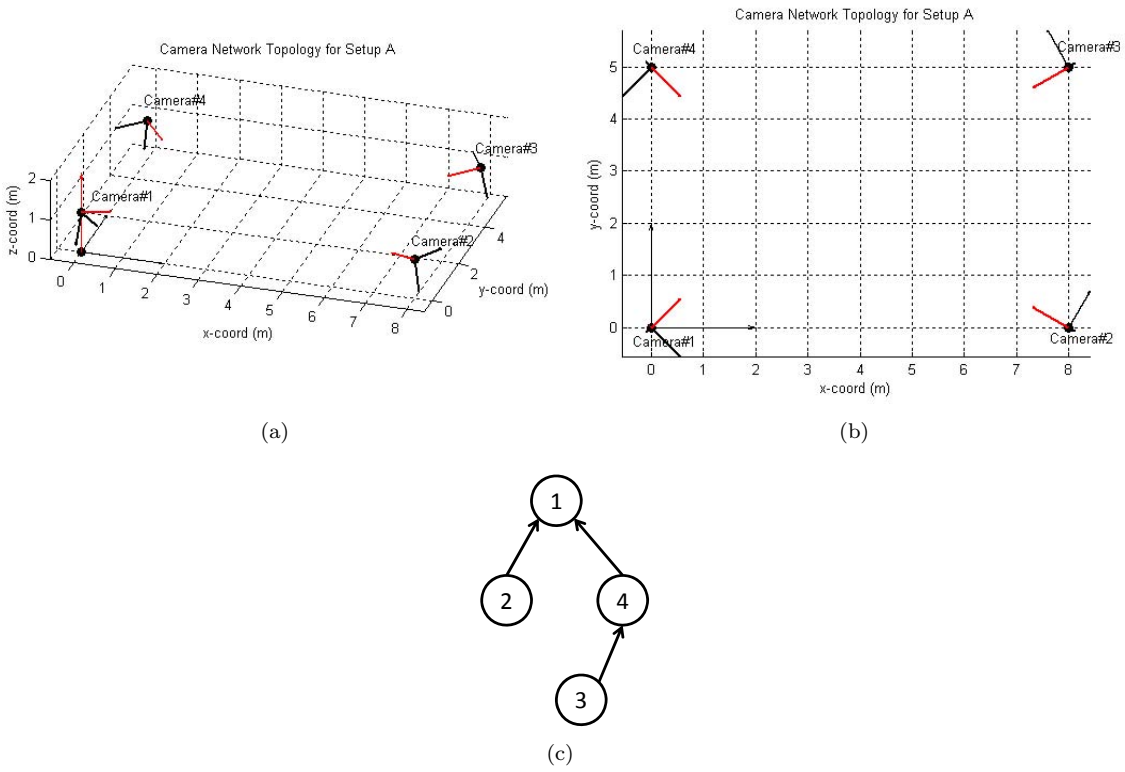
Performance Evaluation

In this section, we evaluate the proposed base tracker, as well as trackers described earlier, and compare them with the tracker based on 3D ray intersection method. We evaluate the trackers for four different camera network setups; two simulated camera networks and two real-world camera network deployments.

Simulated Camera Network

To demonstrate the effectiveness and efficiency of the proposed approaches, we performed experiments on synthetic video data. The scenarios considered here involve a wireless network of smart cameras arranged in a simulated topology. We consider two camera network topologies in this section. Camera network setup A consists of 4 cameras in a topology illustrated in Figure 40, while camera network setup B consists of 10 cameras in a topology shown in Figure 50. Setup B covers a much larger area as compared to setup A. We will see later that the optimal tracking approach depends on the camera topology, specifically tracker T3 performs better if the target is close to the camera and occupies a significant number of pixels, while tracker T2 performs better if the target is farther away from the camera and occupies a very small number of pixels. Figure 40(c) shows the network routing tree for in-network aggregation as proposed in trackers T2 and T3.

Synthetic Target A synthetic target with multiple colors moves in the 3D space, moving in and out of cameras field-of-view. Depending on the current target position and orientation, and the camera network geometry, the target's projection on each camera image plane is computed and superimposed on a pre-recorded background video with visual clutter. The ellipsoid synthetic



(a) (b) (c)

Figure 40: Camera network topology for setup A (a) 3D-view, (b) top-view, and (c) the routing tree.

target is shown in Figure 41. As the target moves up and down a camera principle axis and rotates

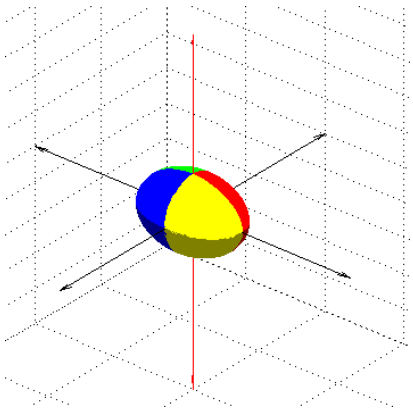


Figure 41: Synthetic target.

around its axis, the correct target projection, with correct pixel color values are computed. Figure 42 shows the image captured by multiple cameras in setup A. For each setup, we generated two types of simulated target trajectories, one when the target orientation is fixed and one when the target is rotating. Below, we present target tracking results for the simulated target and present a

Figure 42: Camera frames for a typical synthetic video.

quantitative comparison of various trackers.

Camera Network Setup A In Figure 43, 44, 45, and 46, we show camera frames with the estimated target position superimposed as a blue ellipse for trackers P, T0, T2 and T3, respectively. Qualitatively, trackers T1 and T2, and trackers T3 and T4 are similar except for the communication cost they incur due to in-network aggregation in trackers T2 and T3. Figure 43 shows camera frames at all four cameras at time-step 10, 20 and 30 during the execution of tracker P. The camera frames for the first time-step with manual target initialization is shown in Figure 42. Figure 44 shows

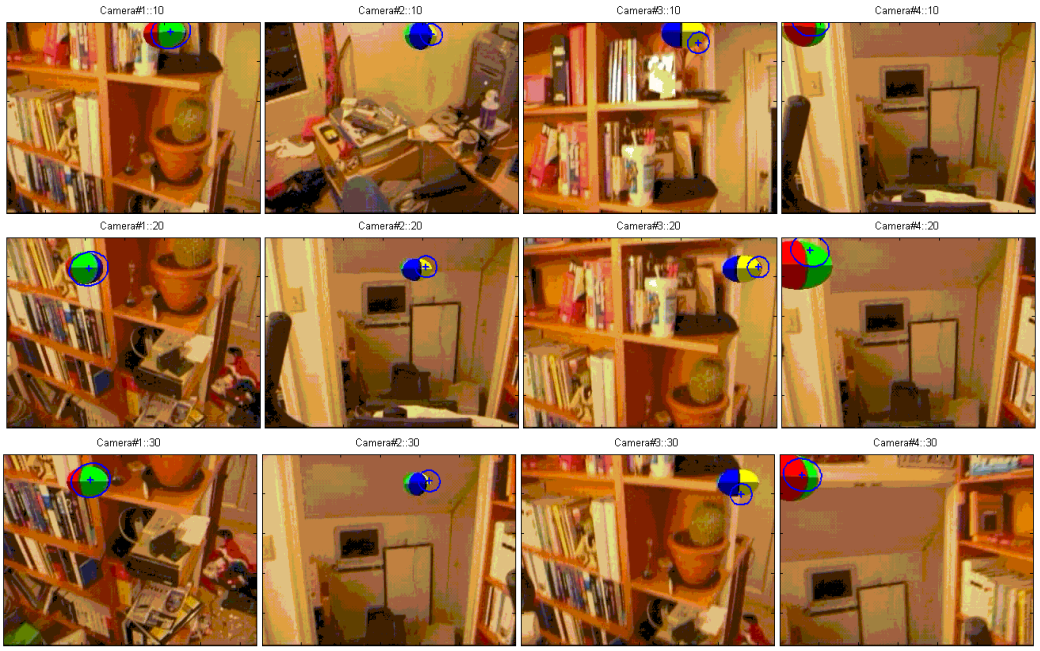


Figure 43: Camera images with estimated target state as blue ellipse for tracker P.

camera frames at all four cameras at time-step 10, 20 and 30 during the execution of tracker T0. The camera frames for the first time-step with manual target initialization are shown in Figure 42.

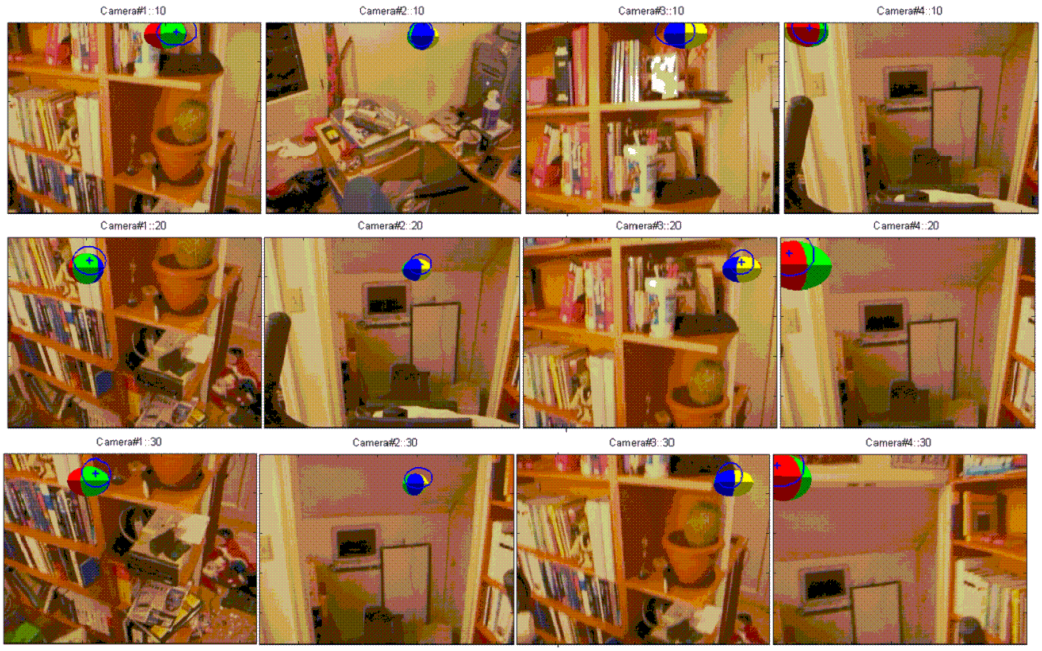


Figure 44: Camera images with estimated target state as blue ellipse for tracker T0.

Figure 45 shows camera frames at all four cameras at time-step 10, 20 and 30 during the execution of tracker T2. The camera frames for the first time-step with manual target initialization are shown in Figure 42. Figure 46 shows camera frames at all four cameras at time-step 10, 20 and 30 during the

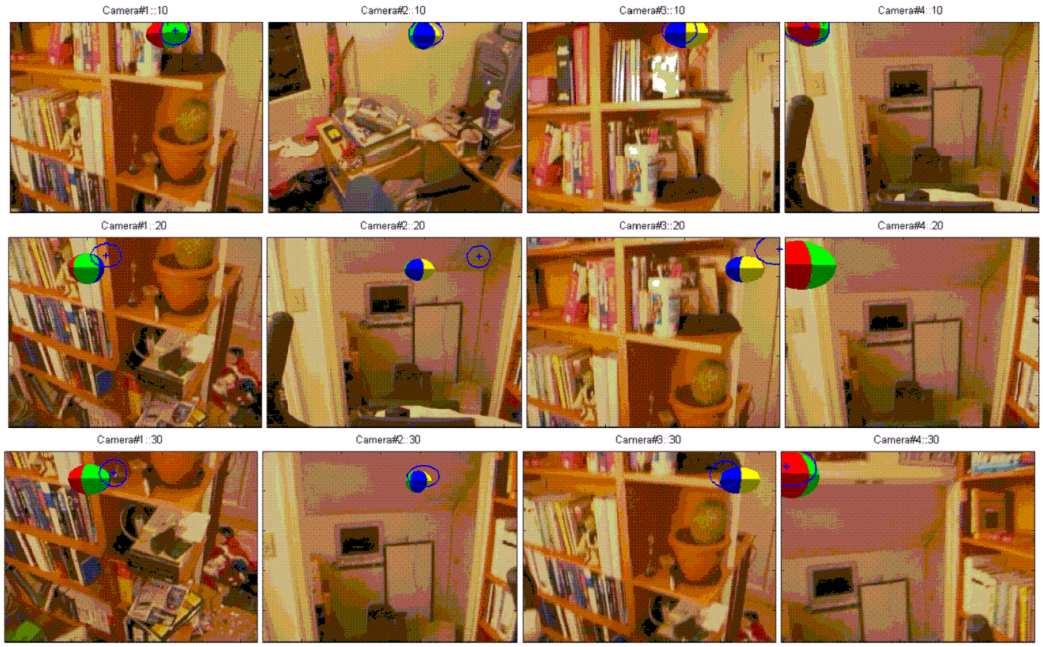


Figure 45: Camera images with estimated target state as blue ellipse for tracker T2.

execution of tracker T3. The camera frames for the first time-step with manual target initialization

is shown in Figure 42.

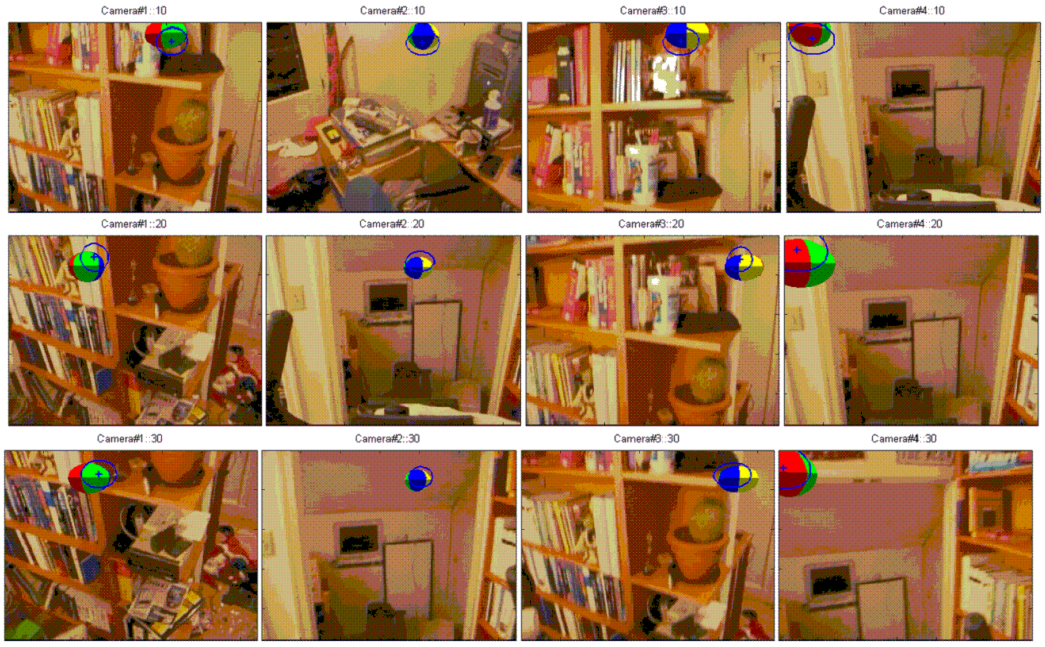
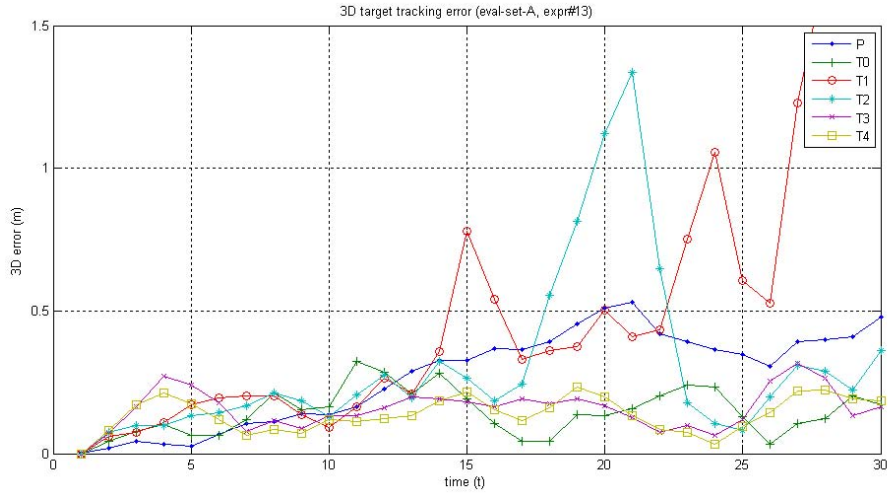


Figure 46: Camera images with estimated target state as blue ellipse for tracker T3.

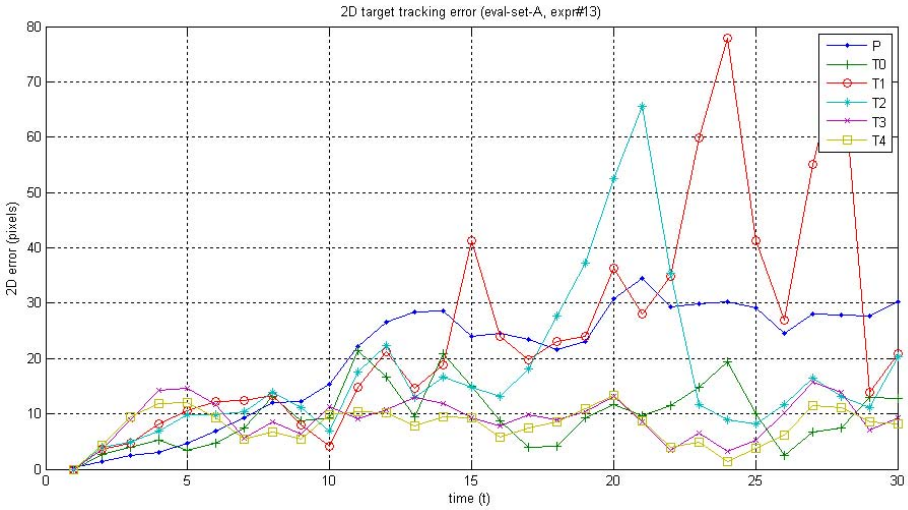
Figure 47(a) and 47(b) show the 3D tracking error and 2D pixel reprojection error for all the trackers.

Tracker P suffers from poor performance when presented with significant target scale change and rotation. The performance degradation will be clearer and more pronounced when we evaluate camera network setup B. The performance of tracker P is poor because the target model consists of a 2D position on the image plane, 2D shape description and a color feature histogram in the current point-of-view. The proposed trackers T0, T2, T3 do not suffer from poor tracking performance in such conditions because the target is modeled in 3D space. The proposed trackers T0 and T2, however, require a greater number of particles than the tracker T3 because they sample the 3D space directly, whereas T3 samples only the image-plane of each camera.

Figure 48 and Figure 49 show the quantitative evaluation of different trackers. Figure 48(a) shows the average 3D tracking errors for a set of 10 simulated experiments for all trackers. Of these 10 experiments, the target orientation is fixed for the first 5 experiments whereas the target is rotating for the next 5 experiments. Figure 48(b) shows the 3D tracking results for rotating and non-rotating targets. Tracker P is outperformed by all other trackers. For almost all trackers, the performance for a rotating target is poorer than that for a non-rotating target. Figures 49(a) and 49(b) show the average 3D tracking error and average 2D pixel reprojection error for all the trackers over all



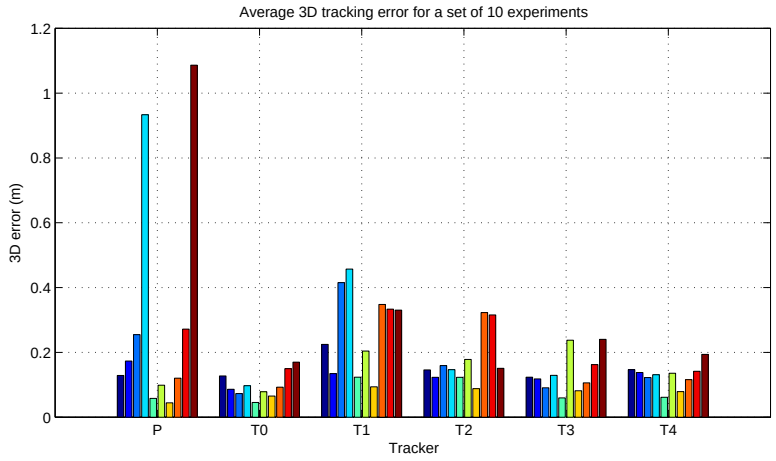
(a)



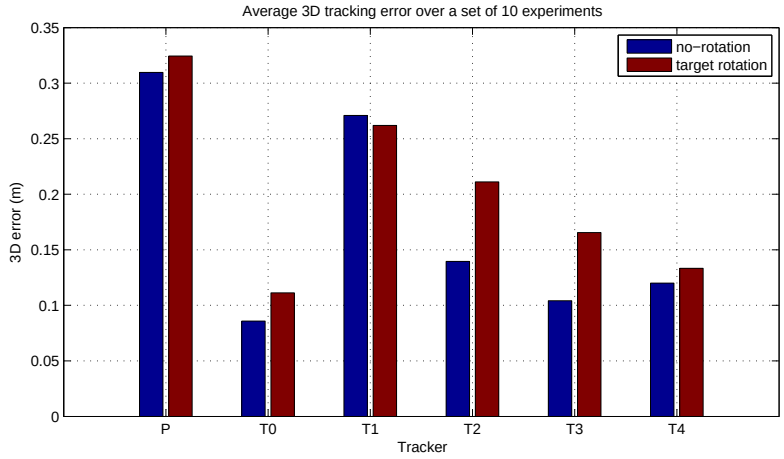
(b)

Figure 47: Tracking error for a typical simulated trajectory (a) 3D tracking error, and (b) 2D pixel reprojection error.

experiments. The trend is the same for both of them. Tracker P is outperformed by all other trackers. The base tracker, T0 performs best followed by the trackers based in image-plane filtering, which is followed by kernel-based filters. In-network aggregation does not have any significant bearing on the image-plane based trackers (tracker T3 over T4), however in-network aggregation does improve performance slightly for 3D particle filter based trackers (tracker T2 over T1). We will make few more observations while evaluating results for camera network setup B.

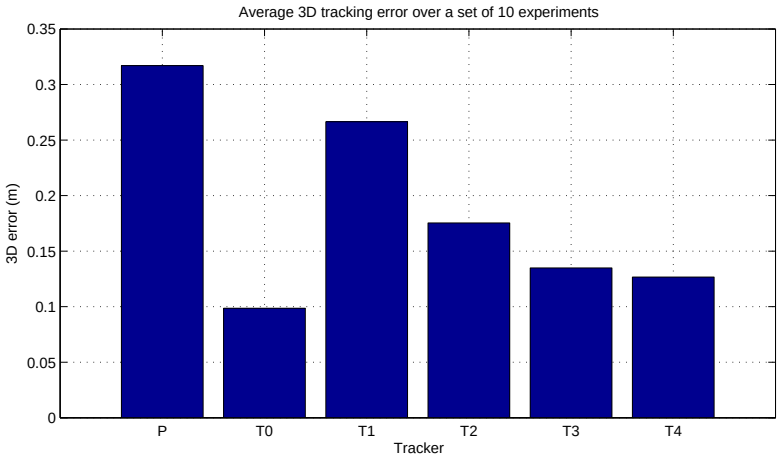


(a)

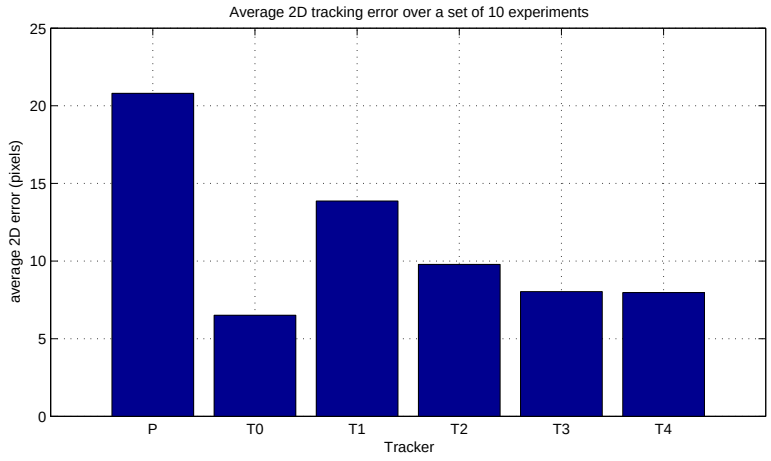


(b)

Figure 48: Quantitative comparison of all trackers for setup A, (a) average 3D tracking error for 10 experiments, and (b) average 3D tracking error for experiments with rotating and non-rotating target.



(a)



(b)

Figure 49: Quantitative comparison of all trackers for setup A, (a) average 3D tracking error over 10 experiments, and (b) average 2D pixel reprojection error over 10 experiments.

Camera Network Setup B Figure 50 shows the simulated camera network setup B, which consists of 10 cameras. This setup covers a much larger area as compared to setup A. Figure 50(c) also shows the network routing tree for in-network aggregation as proposed in trackers T2 and T3.

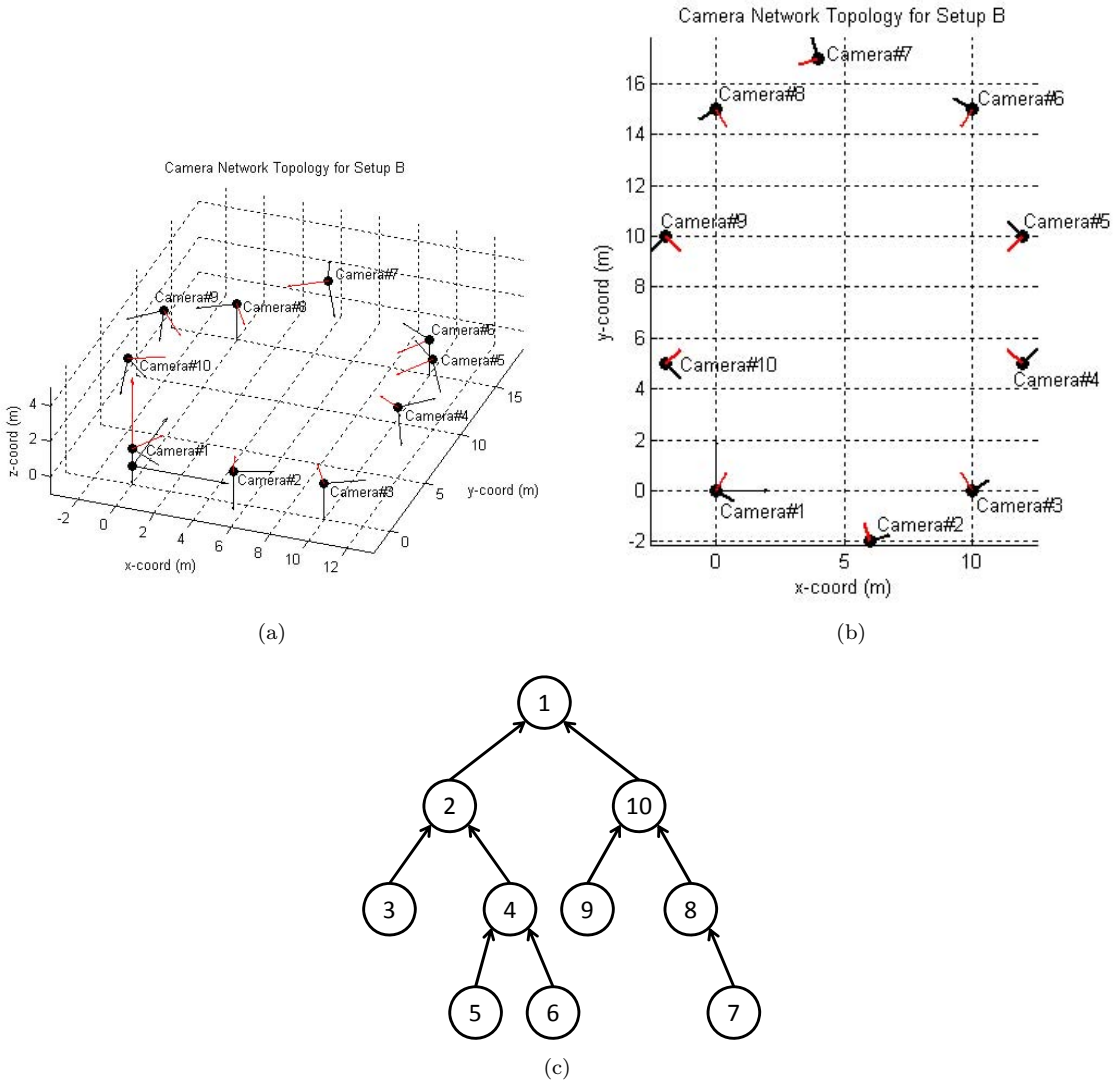


Figure 50: Camera network topology for setup B (a) 3D-view, (b) top-view, and (c) the routing tree.

Figure 51 shows the camera frames at all ten cameras at time-step 1, 10, 20 and 30 during the execution of tracker T2. Similar to camera network setup A, target initialization is performed manually at first time-step. The estimated target position is shown as a blue ellipse superimposed on camera frames. Unlike setup A, the target size in the image-plane for setup B is much smaller due to a much wider coverage. This results in a fewer number of pixels occupied by the target in each camera frame. A larger setup also means that the target will not remain visible in any given

Figure 51: Camera images with estimated target state as blue ellipse for tracker T2.

camera field-of-view for a large number of time-steps.

Figures 52(a), 52(b), 52(c), and 52(d) show the 3D tracking result for trackers P, T0, T2 and T3, respectively (tracker T1 had similar performance as T2, while tracker T4 had similar performance as T3). The figures show the tracking performance as a *patch* graph, where the *blue* edge of the patch is the ground truth trajectory, while the *red* edge of the patch is the estimated target trajectory. The filled patches between the two edges represent the 3D tracking error at each time-step. As the target moves in the sensing region, it comes in and out of the fields-of-views of different cameras. Figure 53(a) shows 3D tracking errors with the number of cameras that contain the target in their camera field-of-view (called *participating cameras*). At the beginning of the experiment, there are 4 cameras that can see the target. It drops to a single camera at time-step 21, with one more camera picking up the target at time-step 23. The top part of the figure shows the 3D tracking errors for the four approaches. Both trackers T0 and T2 performed quite well till time-step 21, when both of them started diverging because of a single participating camera. Unlike tracker T0, tracker T2 converged back to the ground truth as soon as there was one more participating camera. Trackers P and T3 performed poorly, even in the presence of as many as six participating cameras. This is explained in Figure 53(b), which shows the 3D tracking errors with the percentage of image pixels occupied by the target averaged over the number of participating cameras. As we can see, almost at all time-steps the percentage of image pixels is below 1% of the total pixels, i.e. the target occupied less than 800 pixels in a QVGA (320×240) image (equivalent of a circular region of radius 16 pixels). Since both trackers P and T3 are based on image-plane filterig, they fared poorly due to a fewer number of usable pixels.

Figure 54(a) and Figure 54(b) show quantitative evaluation of different trackers for setup B over a set of 15 simulated experiments. Of these 15 experiments, the target orientation is fixed for the

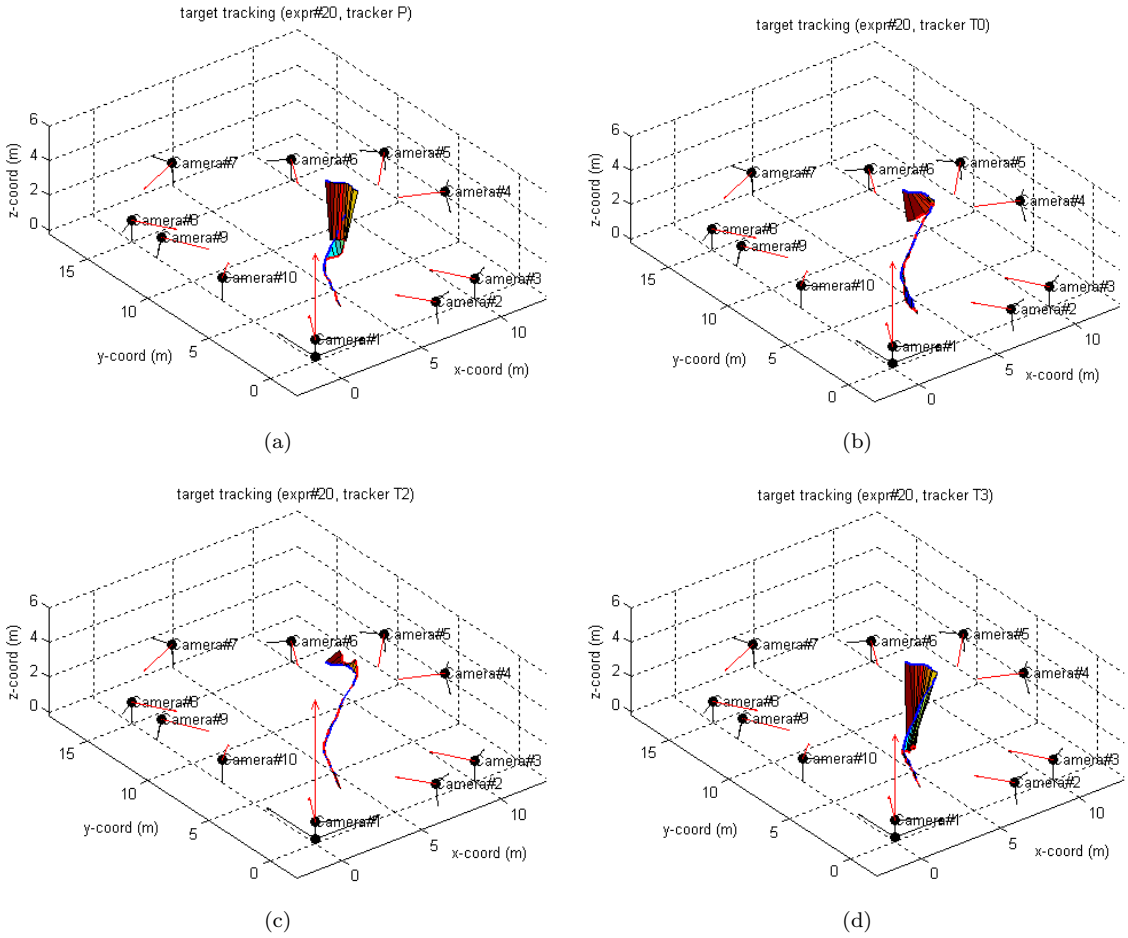
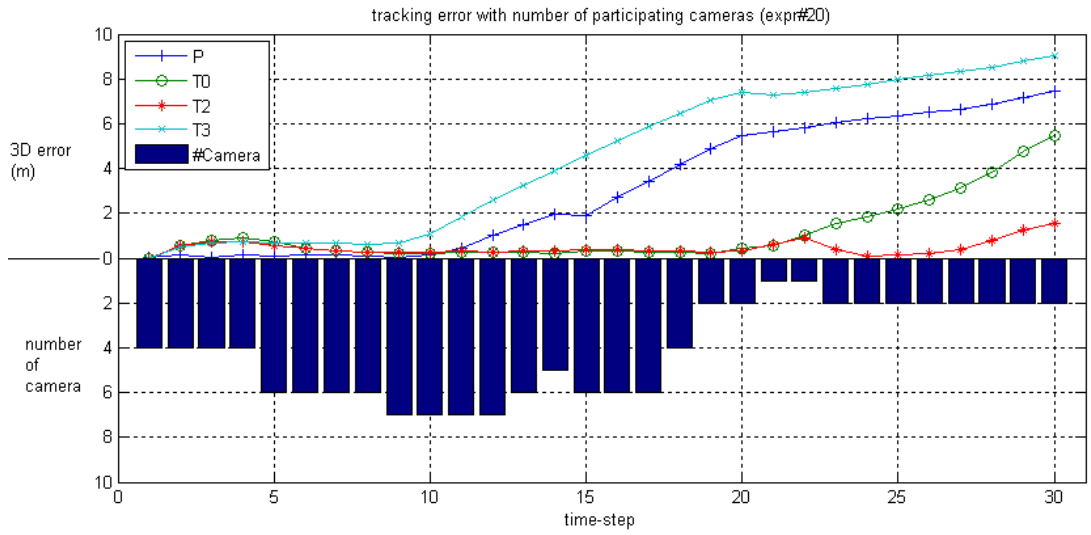


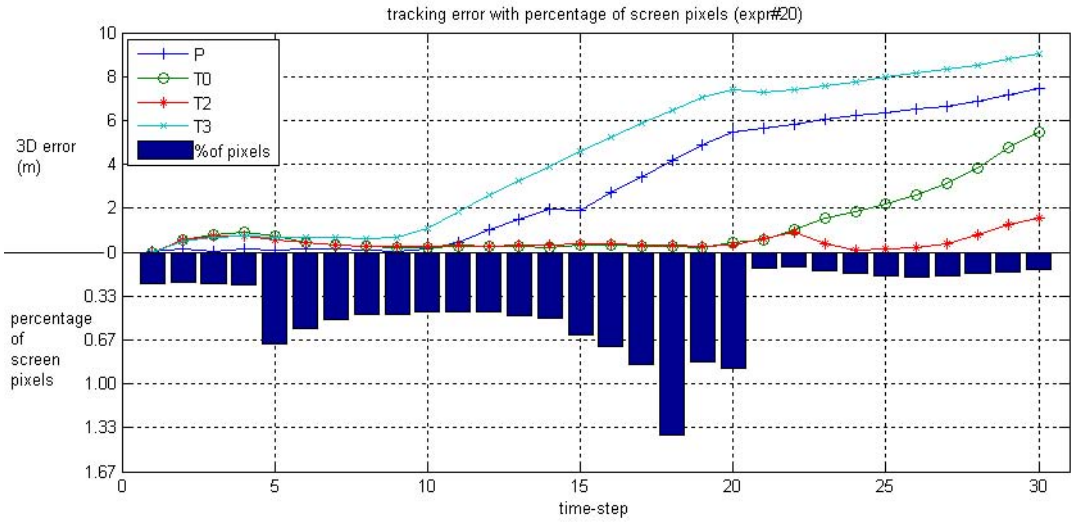
Figure 52: Target tracking performance as a *patch* graph for (a) tracker P, (b) tracker T0, (c) tracker T2, and (d) tracker T3.

first 10 experiments, whereas the target is rotating for the next 5 experiments. Figures 54(a) and 54(b) show the average 3D tracking errors and the average 2D pixel reprojection errors for all the trackers. Figure 55(a) and 55(b) show the average 3D tracking errors and the average 2D pixel reprojection errors for the rotating and non-rotating targets. It seems that target rotation does decrease the performance in 3D tracking by a small amount, while it does not have any significant bearing on the 2D pixel reprojection error.

Figures 56(a) and 56(b) show the average 3D tracking error and the average 2D pixel reprojection error for all the trackers averaged over all experiments. The trend is the same for both of them. Trackers based on 3D kernel density, namely trackers T1/T2 perform far better than any other tracker followed by the base tracker T0. The trackers based on image-plane filtering, namely trackers P/T3/T4 perform poorly for this setup, due to the reasons mentioned above. In-network aggregation does not have any significant bearing on either 3D kernel density based trackers, or the image-plane



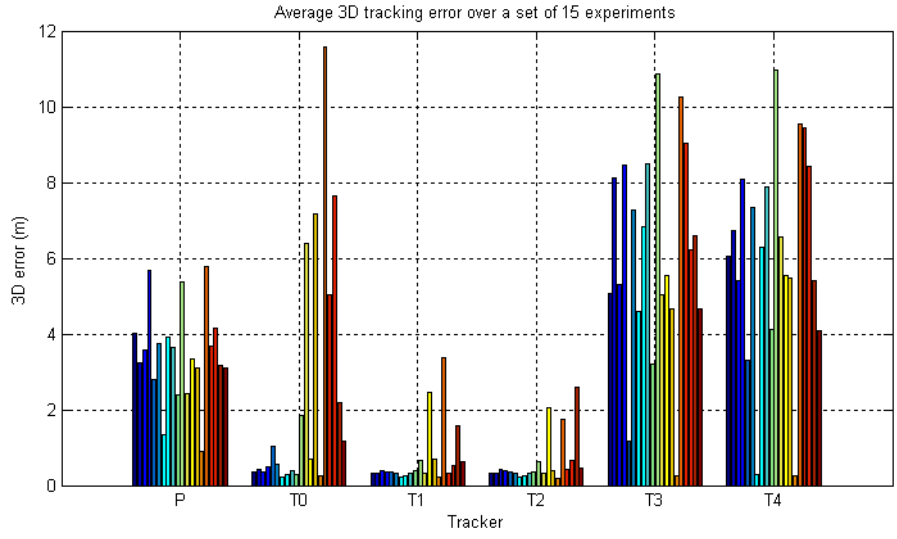
(a)



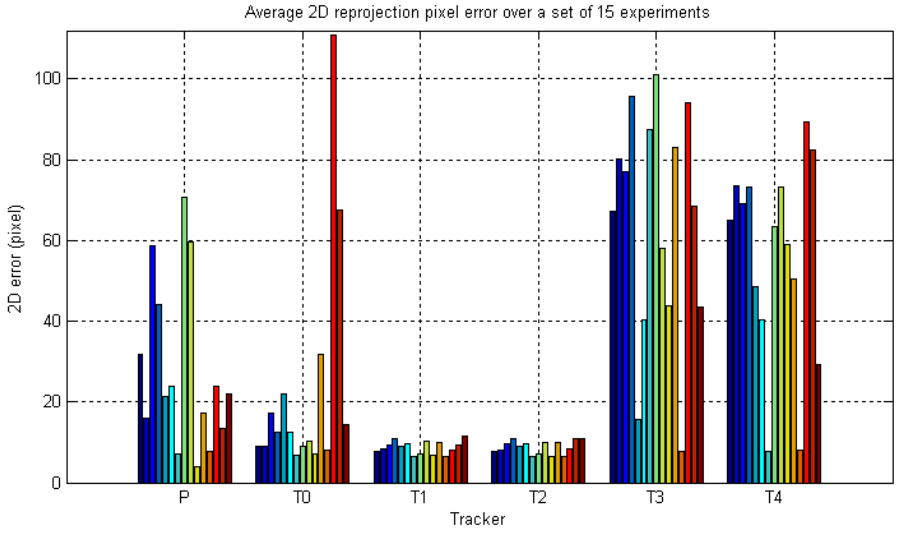
(b)

Figure 53: 3D tracking errors for trackers P, T0, T2, and T3 with (a) number of *participating cameras*, and (b) percentage of image pixels occupied by the target.

based trackers.

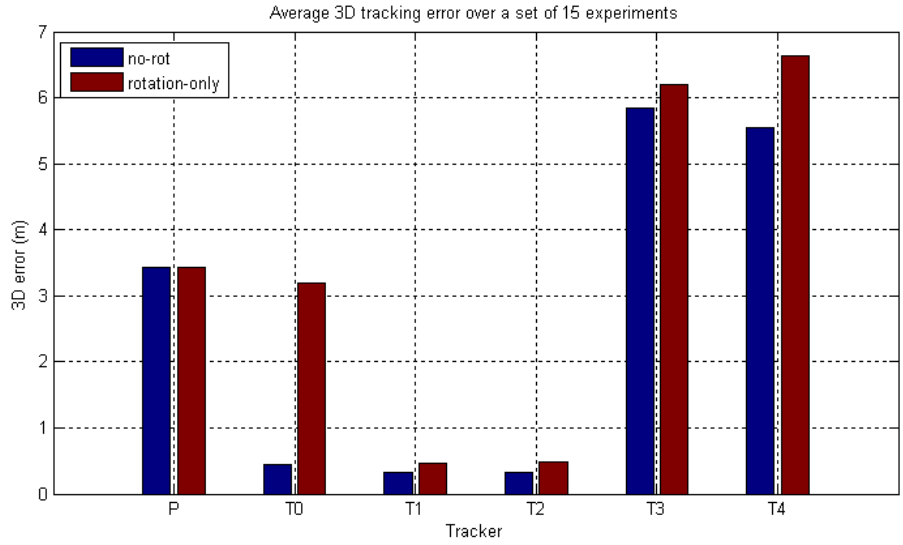


(a)

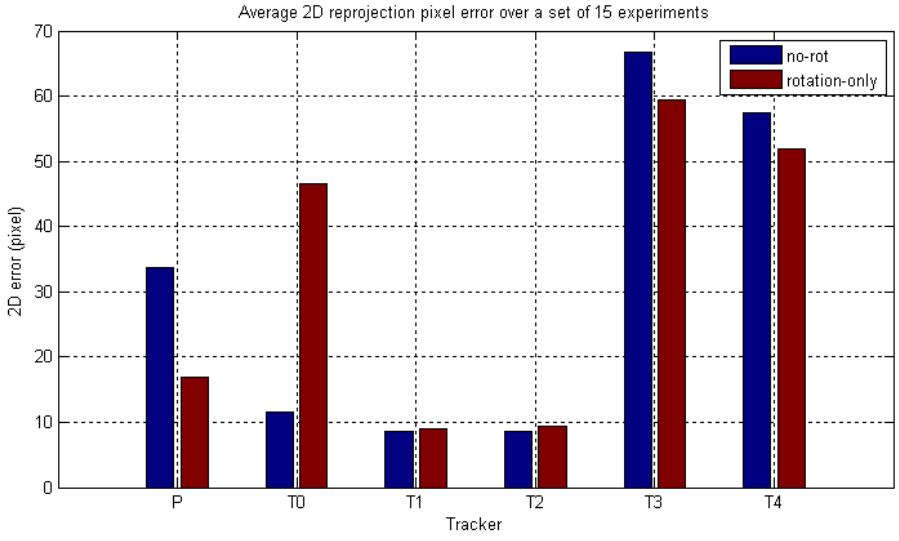


(b)

Figure 54: Quantitative comparison of trackers for setup B over a set of 15 simulated experiments, (a) average 3D tracking error, and (b) average 2D pixel reprojection error.

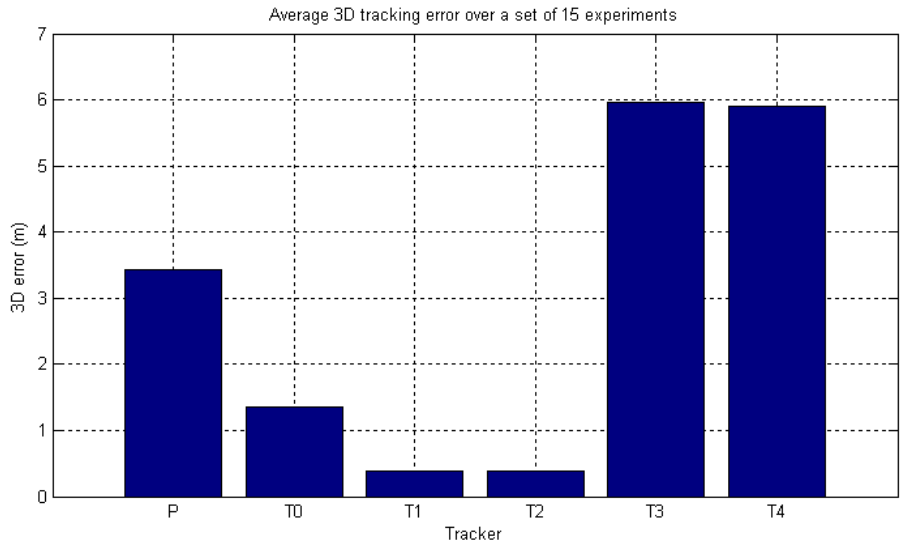


(a)

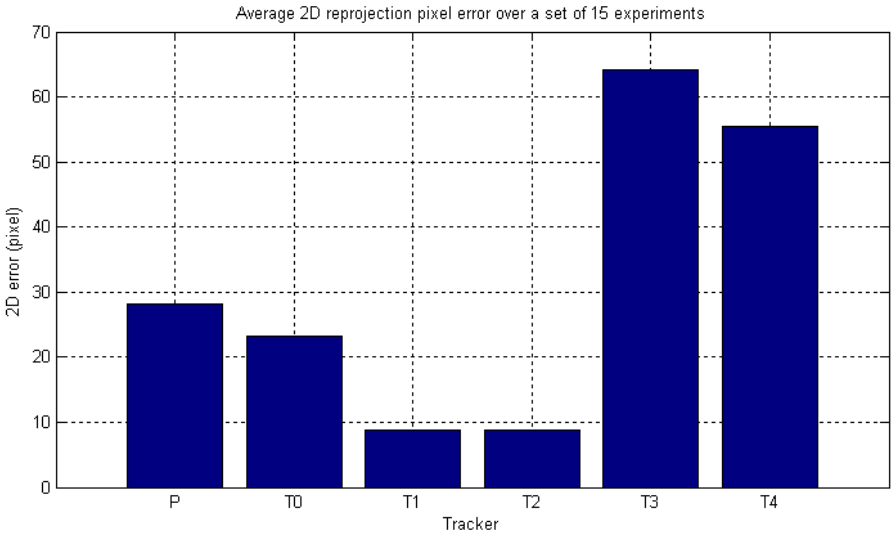


(b)

Figure 55: Quantitative comparison of all trackers for setup B for rotating and non-rotating targets, (a) average 3D tracking error, and (b) average 2D pixel reprojection error.



(a)



(b)

Figure 56: Quantitative comparison of all trackers for setup B averaged over the set of 15 simulated experiments, (a) average 3D tracking error, and (b) average 2D pixel reprojection error.

LCR Experiments

In this section, we present results for a real camera network deployment inside a Large Conference Room (LCR setup). The setup consists of 4 camera nodes in a topology as shown in Figure 57. A real target, in this case a typical box, is moved in the 3D space, moving in and out of coverage of cameras. Figure 57(c) also shows the network routing tree for in-network aggregation as proposed in trackers T2 and T3.

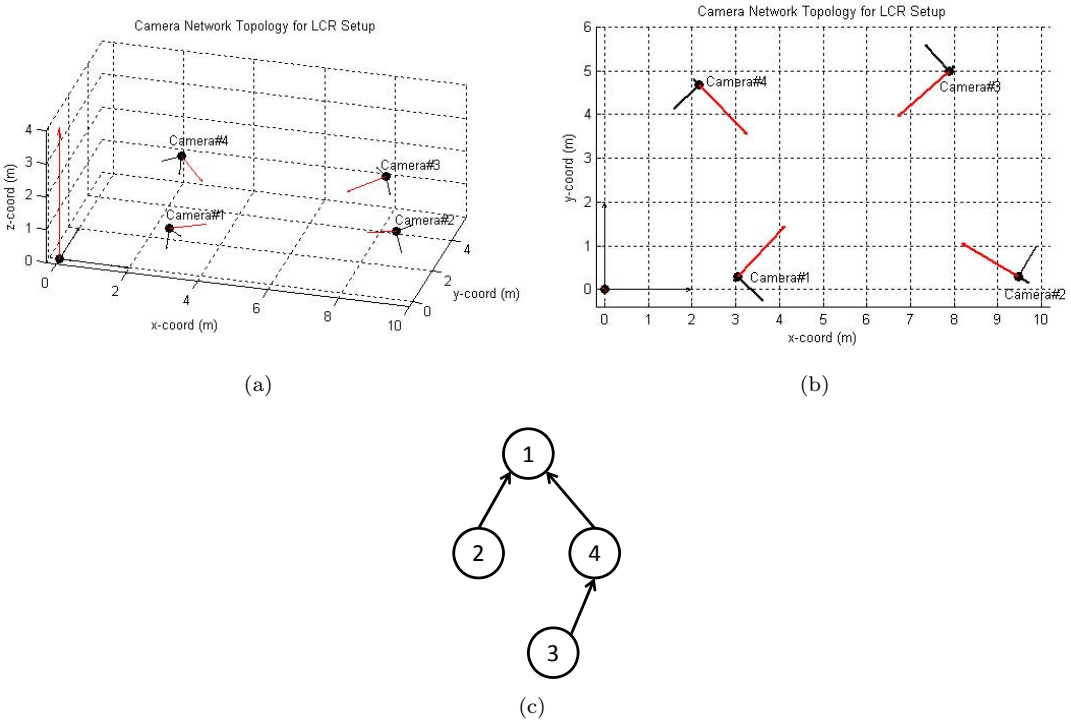


Figure 57: Camera network topology – LCR setup (a) 3D-view, (b) top-view, and (c) the routing tree.

Figure 58 shows the camera frames at the four cameras at time-step 1, 15, 30, 50, 70 and 90 during the execution of tracker T3. Target initialization is performed manually at the first time-step. The estimated target positions are shown as blue ellipses superimposed on the camera frames. Figure 59 shows the 3D target trajectory as estimated by the tracker. In this experiment, as in all other experiments for this setup, the moving target was in the fields-of-view of four cameras.

Figure 60 shows the percentage of image pixels occupied by the target averaged over the number of participating cameras. As we can see, at all time-steps the percentage of image pixels is above 1% of the total pixels. This is perhaps the reason why tracker T3, which is based on image-plane filtering, works fine.

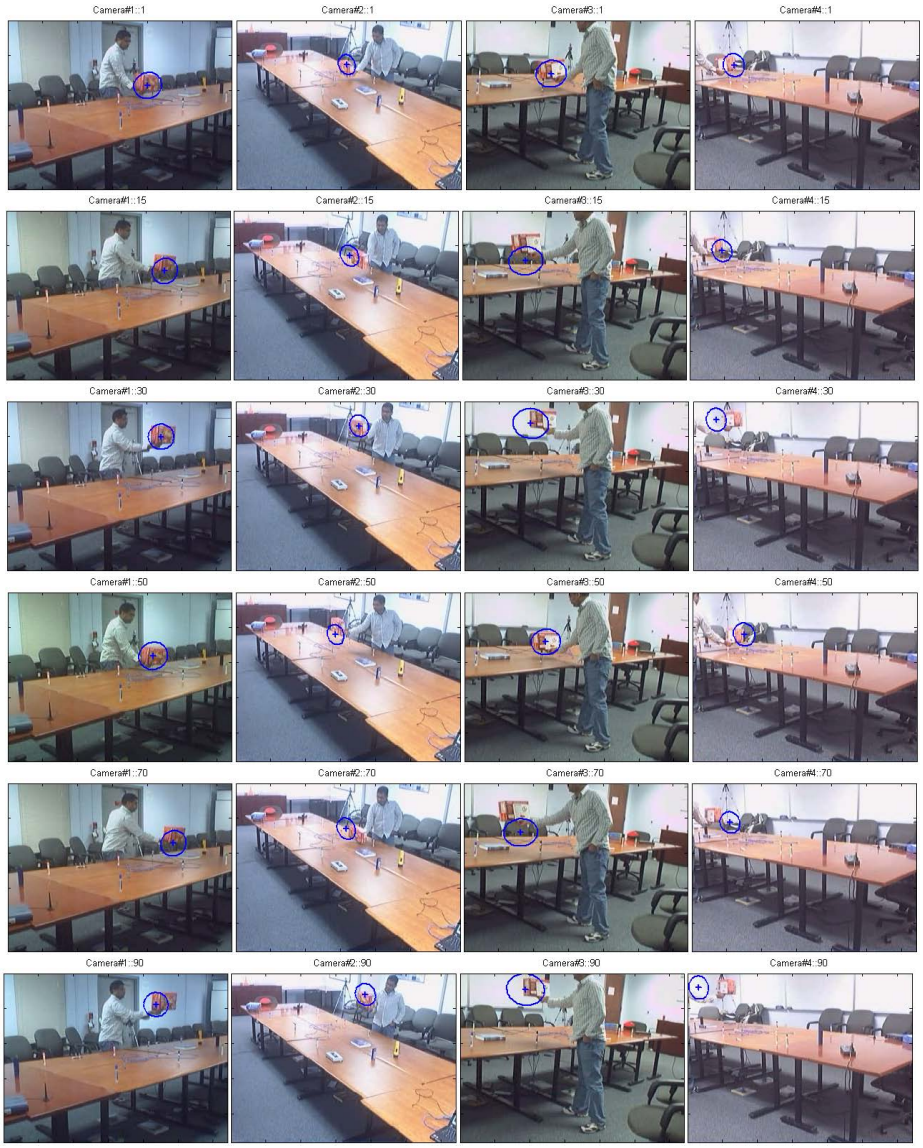


Figure 58: Video target tracking using tracker T3 for experiment#1 for LCR setup.

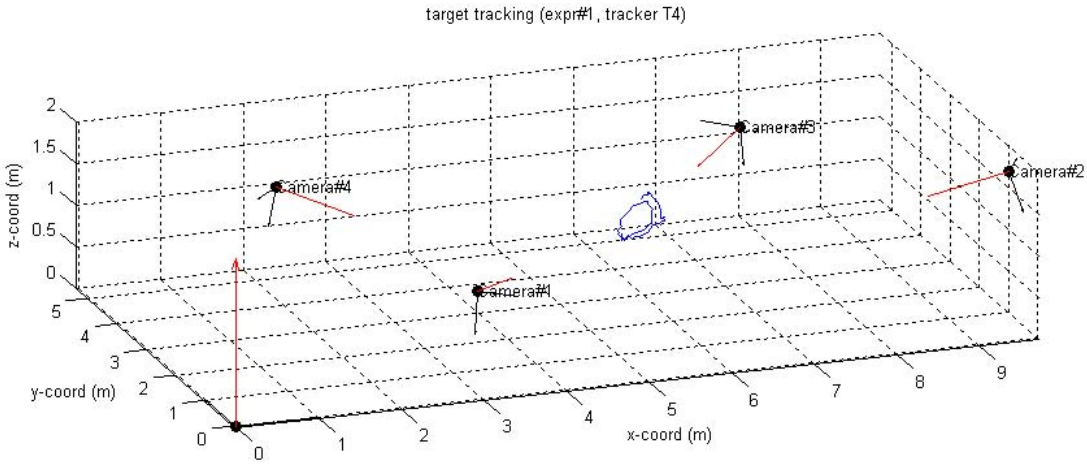


Figure 59: Estimated target trajectory shown with camera network topology.

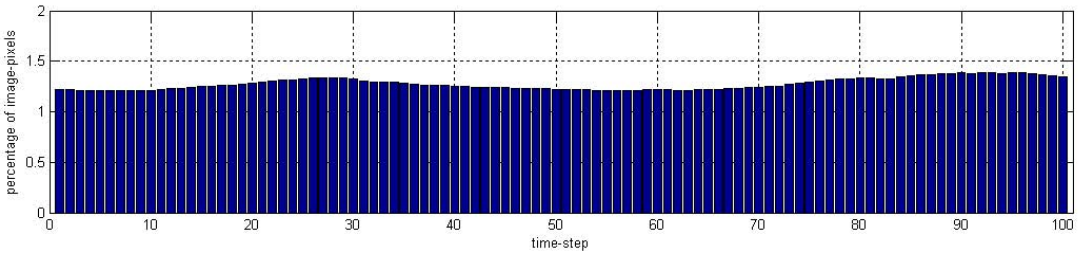


Figure 60: Average fraction of image pixels occupied by the target.

FGH Experiments

In this section, we present results for a real camera network deployment inside our department building (FGH). The setup consists of 6 camera nodes as shown in Figure 61. In this case, the targets to be tracked are the people moving in the FGH atrium. Figure 61(c) also shows the network routing tree for in-network aggregation as proposed in trackers T2 and T3.

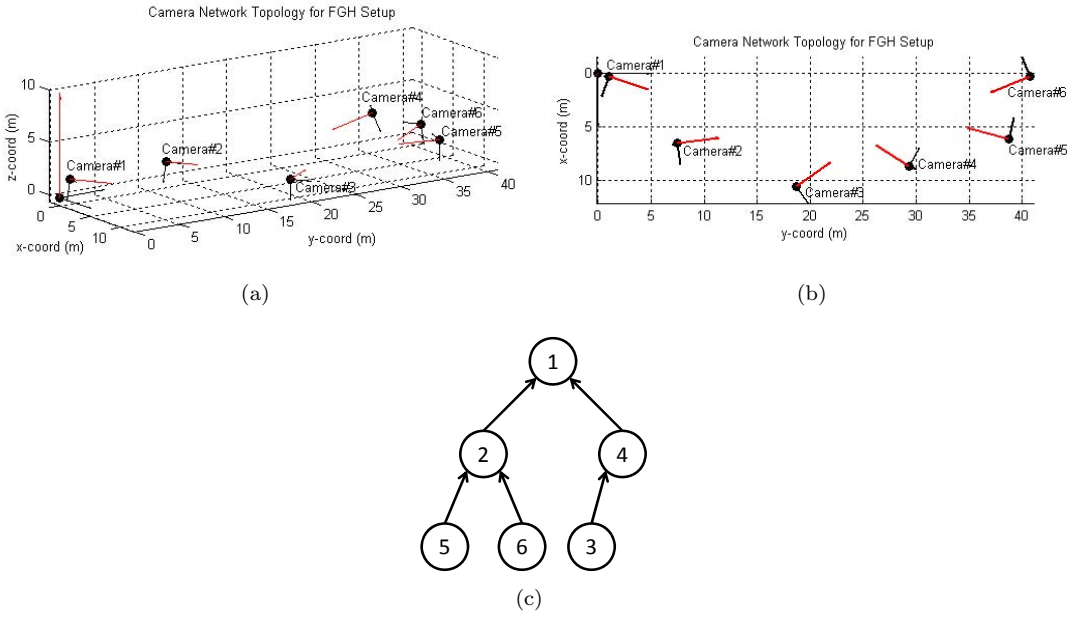


Figure 61: Camera network topology – FGH setup (a) 3D-view, (b) top-view, and (c) the routing tree.

Figure 62 shows the camera frames at all six cameras at time-step 1, 31, 50, 75, 100, 150 and 175 during the execution of tracker T2 for experiment#1. As with other setups, target initialization is performed manually at the first time-step. The estimated target positions are shown as blue ellipses superimposed on the camera frames. This experiment demonstrates that even for an extended number of frames (192 frames) the tracker is successfully able to follow the target. During the length of this experiment, the target dramatically changes scale in camera images, and comes in and out of different camera fields-of-view. This experiment demonstrates the effectiveness of a 3D tracker over state-of-the-art 2D trackers by: 1) not having to learn or update the target model even in case of dramatic scale change and target rotation, and 3) not having to reinitialize a target when it (re-)enters a camera field-of-view.

Figure 63 shows the 3D target trajectory as estimated by the tracker. Since the sensing region in this setup is large, the target invariably moves in and out of the camera fields-of-view. We have also put a threshold on the size of the projection of the target on the camera image plane. If the

Figure 62: Video target tracking using tracker T2 for experiment#1 for FGH setup.

pixels occupied by the target in a particular camera image is below the threshold, we deem that frame unusable. Figure 64(a) shows the number of participating cameras at each time-step. At the beginning of the experiment, there are 2 cameras that participate in tracking. It grows up to six cameras for a single time-step before dropping to three participating cameras in the end. At all time-steps, there are at least 3 cameras that contain the target in their fields-of-view, such that the camera projection size exceeds the threshold. Figure 64(b) shows the percentage of image pixels occupied by the target averaged over the number of participating cameras. As we can see, for most of the time the percentage of image pixels is below 1% of the total pixels. The share of pixels for the target grows above 1% when it is in the middle of the sesing region. Since both trackers P and

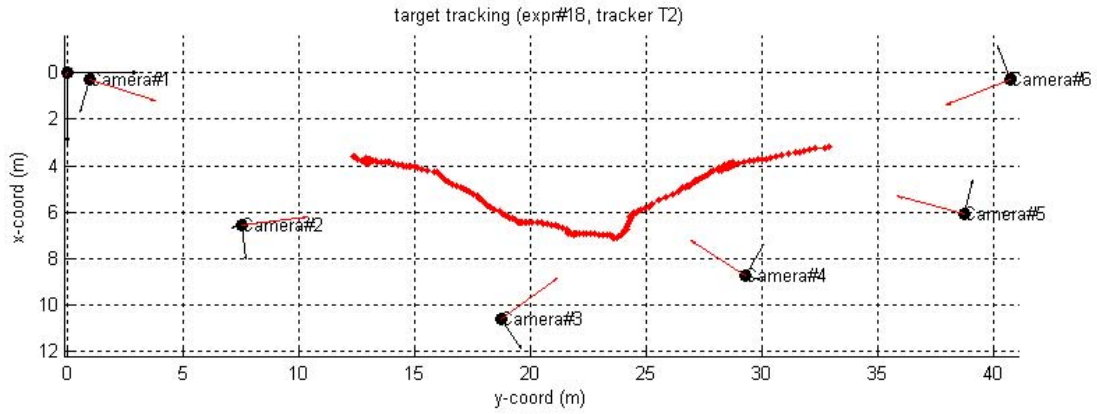
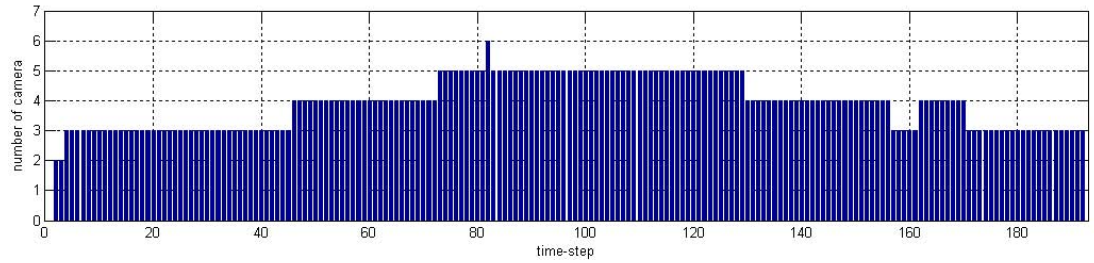
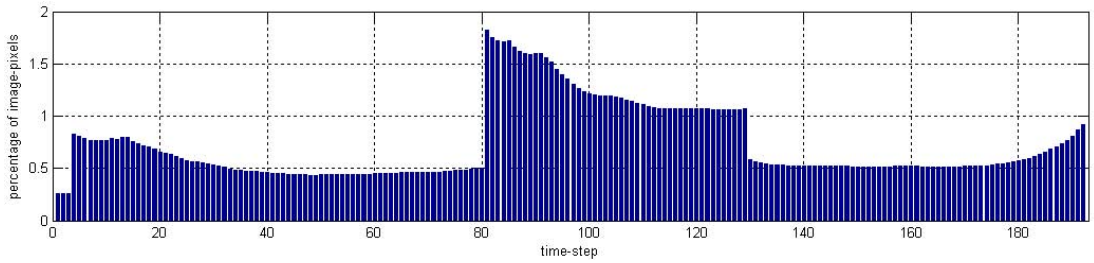


Figure 63: Estimated target trajectory shown with camera network topology.

T3 are based on image-plane filtering, they fared poorly due to the fewer number of usable pixels. This is why we only show the results for the best tracker in this setup, which is tracker T2.



(a)



(b)

Figure 64: Number of *participating* cameras, and average fraction of image pixels occupied by the target.

Figure 65: Target tracking, FGH Setup, Experiment#1

We show tracking results for one more experiment in this dissertation. For more tracking results and videos we encourage the reader to go online at http://sites.google.com/site/manishkushwaha79/researchex/videos-for-target-tracking-experiments. Figure 66 shows the camera frames at all six cameras at time-step 1, 31, 50, 75, 100, 150 and 175 during the execution of tracker T2 for experiment#2. Again, target initialization is performed manually at the first time-step, and the estimated

target positions are shown as blue ellipses superimposed on the camera frames. Along with other points made in the analysis of the previous experiment, this experiment demonstrates robustness of the tracker in the presence of target occlusion. As we can see between frame 41 and 46, the target is occluded by another person. Since we are using discriminatory features, i.e. color and texture, and we are performing tracking in 3D space, our tracker is successfully able to handle such occlusions.

Figure 66: Video target tracking using tracker T2 for experiment#2 for FGH setup.

Figure 67 shows the 3D target trajectory as estimated by the tracker. Similar to previous experiment analyses, Figure 68(a) shows the number of participating cameras at each time-step. At the beginning of the experiment, there are 3 cameras that participate in tracking, which grows to 5 participating cameras in the end. Figure 68(b) shows the percentage of image pixels occupied by

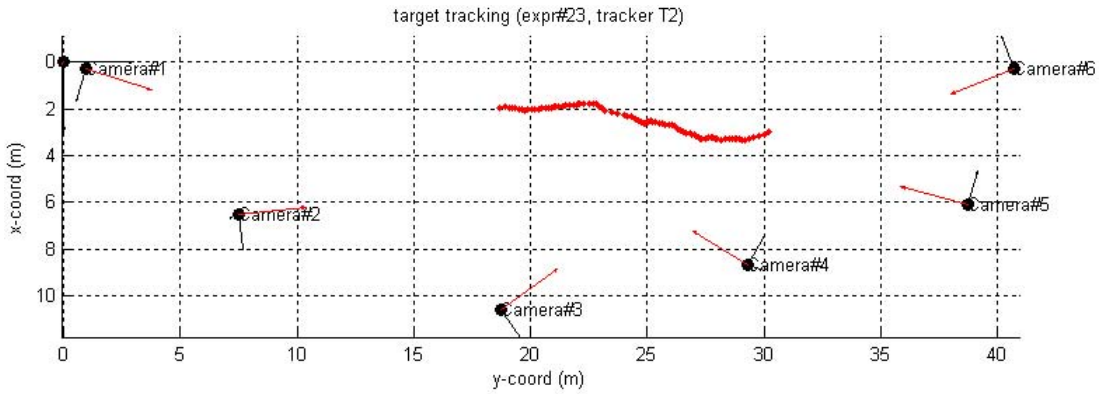
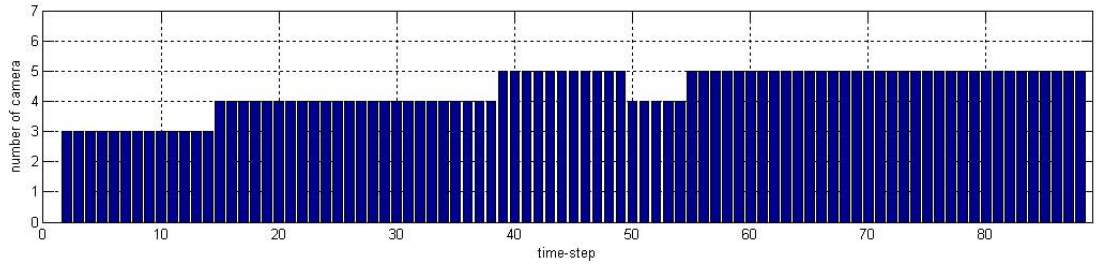
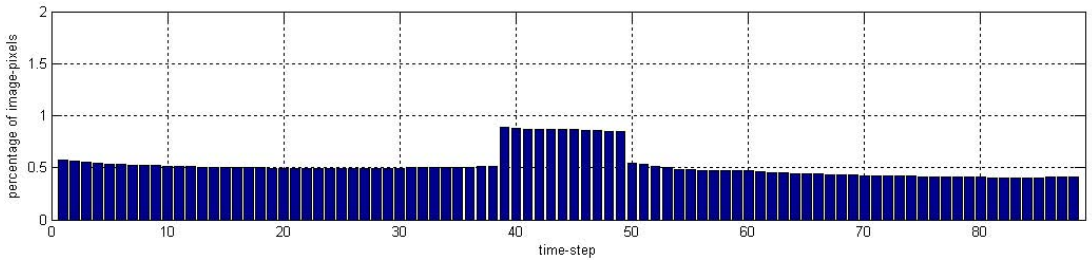


Figure 67: Estimated target trajectory shown with camera network topology.

the target averaged over the number of participating cameras. At all time-steps, the percentage of image pixels is below 1% of the total pixels.



(a)



(b)

Figure 68: Number of *participating* cameras, and average fraction of image pixels occupied by the target.

Conclusion

In summary, we proposed a number of probabilistic approaches for collaborative target tracking in 3D space using a wireless network of smart cameras. We established that modeling the target in 3D space circumvents the problems inherent in the 2D tracker, or combination of 2D trackers.

CHAPTER VI

CONCLUSION

Target tracking is an important class of application in HSNs. With the proliferation of cost-effective sensors and ubiquity of HSNs, target tracking and various surveillance systems will pervade our environment. Classical target tracking approaches, such as those based on decision-level fusion, suffer from poor discrimination and exponential complexity. In case of multiple targets, when they are in close proximity and interacting with each other, the classical approaches have poor performance. The alternatives to decision-level fusion are signal-level and feature-level fusion methods. However, signal-level fusion methods are not feasible in HSNs due to limited communication bandwidth. The feature-level fusion methods for tracking are much needed and they hold great promise to large-scale target tracking and surveillance systems.

In summary, the contributions of this Ph.D. dissertation address some of these research challenges. The contribution lies in using the capabilities of HSNs to minimize or mitigate the handicap of sensor and target models, due to violation of assumptions. This is achieved by redundant, complementary and cooperative sensor fusion. A WSN can have a number of sensors of different modalities spatially distributed over the sensing environment. The advantage of HSNs is that when one sensor, or a group of sensors are handicapped due to violation of some sensor or target assumption, other sensors at different locations and of modalities, might be able to carry out tracking.

Along with handicap mitigation, we attempt to address the real-time processing requirements and limited communication bandwidth. We use simple target models and simple features for fast processing. We also focus on concise features for efficient communication. In addition, this research proposal presents extensive related work in the area of general target tracking approaches, and specific approaches for target localization and tracking using audio and video sensors.

The specific contributions of this dissertation are listed below,

1. To illustrate the approach, we have designed and implemented a multimodal multisensor information fusion system for target tracking in an urban environment using an HSN [9–13]. The HSN consists of mote class devices equipped with microphone arrays as audio sensors and embedded PCs equipped with web cameras as video sensors. The system operates online in real-time at 4Hz, thus addressing the real-time processing requirement. The audio and video

sensors compute local features and communicate them with the fusion node, thus addressing the limited communication bandwidth.

2. Further, we proposed a feature-based approach to collaborative source localization of multiple acoustic sources in WSNs [14–16]. Acoustic beamform and power spectral density (PSD) extracted from sensor nodes equipped with microphone array are used as acoustic features.

3. Finally, we propose an approach for collaborative target tracking in 3D space using a wireless network of smart cameras. We model the targets in 3D space thus circumvent the problems inherent in the tracker based on 2D target models. In addition, we use multiple visual features, specifically, color and texture to model the target. We propose a number of probabilistic 3D trackers and implement them using sequential Monte Carlo method. We evaluate the trackers using synthetic targets in simulated camera networks, as well as using real targets (objects and people) in real-world camera network deployments.

CHAPTER VII

LIST OF PUBLICATIONS

Journal Papers

1. Manish Kushwaha, Xenofon Koutsoukos, Sandor Szilvasi, *Feature-based Collaborative Localization and Discrimination in Wireless Sensor Networks*, ISIF Journal of Advances in Information Fusion (Submitted).

2. Songhwai Oh, Phoebus Chen, Marci Meingast, Manish Kushwaha, Inseok Hwang, *Data Association and Its Applications to Intelligent Systems: A Bayesian Approach*, Preprint submitted to Robotics and Autonomous Systems.

Book Chapters

1. Manish Kushwaha, Songhwai Oh, Isaac Amundson, Xenofon Koutsoukos, Akos Ledeczi, *Tracking in Urban Environments Using Sensor Networks Based on Audio-Video Fusion*, Handbook of Ambient Intelligence and Smart Environments (AISE), pp. 117–148 Springer, 2009.

Conference Papers

1. Manish Kushwaha, Xenofon Koutsoukos, Peter Volgyesi and Akos Ledeczi, *Acoustic Source Localization and Discrimination in Urban Environments*, International Conference on Information Fusion, pp. 1859–1866, July 2008.

2. Manish Kushwaha, Songhwai Oh, Isaac Amundson, Xenofon Koutsoukos, and Akos Ledeczi, *Target Tracking in Urban Environments using Audio-Video Signal Processing in Heterogeneous Wireless Sensor Networks*, 42nd IEEE Asilomar Conference on Signals, Systems and Computers, pp. 1606-1610, October 2008.

3. Marci Meingast, Manish Kushwaha, Songhwai Oh, Xenofon Koutsoukos, Akos Ledeczi, Shankar Sastry, *Fusion-based localization for a Heterogeneous camera network*, In ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-08), Stanford University, California, USA, September 2008.

4. Manish Kushwaha, Songhwai Oh, Isaac Amundson, Xenofon Koutsoukos, Akos Ledeczi, *Target Tracking in Heterogeneous Sensor Networks Using Audio and Video Sensor Fusion*, In IEEE

International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2008), Seoul, Korea, pp. 14–19, August 2008.

5. Manish Kushwaha, Isaac Amundson, Peter Volgyesi, Parvez Ahammad, Gyula Simon, Xenofon Koutsoukos, Akos Ledeczi, Shankar Sastry, *Multi-modal Target Tracking using Heterogeneous Sensor Networks*, In International Conference on Computer Communications and Networks (ICCCN 2008), St. Thomas, US Virgin Islands, August 2008.

6. Manish Kushwaha, Isaac Amundson, Xenofon Koutsoukos, Sandeep Neema, and Janos Sztipanovits, *OASiS: A Programming Framework for Service-Oriented Sensor Networks*, In International Conference on Communication System Software and Middleware (COMSWARE 2007), Bangalore, India, January 2007.

7. Manish Kushwaha, Karoly Molnar, Janos Sallai, Peter Volgyesi, Miklos Maroti, Akos Ledeczi, *Sensor Node Localization Using Mobile Acoustic Beacons*, The 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2005), Washington DC, pp. 483–491, November 2005.

Workshop Papers

1. Isaac Amundson, Manish Kushwaha, Xenofon Koutsoukos, *On the Feasibility of Determining Angular Separation in Mobile Wireless Sensor Networks.* In 2nd International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT), Orlando, FL, 2009.

2. Xenofon Koutsoukos, Manish Kushwaha, Isaac Amundson, Sandeep Neema, and Janos Sztipanovits, *OASiS: A Service-Oriented Architecture for Ambient-Aware Sensor Networks*, F. Kordon and O. Sokolsky (Eds.) Monterey Workshop 2006, LNCS 4888, pp. 125-149, 2007, Springer-Verlag.

3. Isaac Amundson, Manish Kushwaha, Branislav Kusy, Peter Lolgyesi, Gyula Simon, Xenofon Koutsoukos, and Akos Ledeczi, *Time Synchronization for Multi-Modal Target Tracking in Heterogeneous Sensor Networks*, In Workshop on Networked Distributed Systems for Intelligent Sensing and Control, Kalamata, Greece, June 30, 2007.

4. Isaac Amundson, Manish Kushwaha, Xenofon Koutsoukos, Sandeep Neema, and Janos Sztipanovits, *Efficient Integration of Web Services in Ambient-aware Sensor Network Applications*, IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (BaseNets 2006), San Jose, CA, Oct. 2006

Conference Posters

1. Manish Kushwaha, Isaac Amundson, Xenofon Koutsoukos, Sandeep Neema, and Janos Sztipanovits, *OASiS: A Programming Framework for Service-Oriented Sensor Networks*, NSF ITR Fall Review Meeting, October 2006.

2. Manish Kushwaha, Isaac Amundson, Xenofon Koutsoukos, Sandeep Neema, Janos Sztipanovits, Chang Hong Lin, Wayne Wolf, *An Object-Centric Programming Framework for Ambient-Aware, Service-Oriented Sensor Networks*, The Fifth International Conference on Information Processing in Sensor Networks (IPSN 2006) (WIP Track), Nashville TN, April 2006.

Technical Reports

1. Manish Kushwaha, Xenofon Koutsoukos, *A Graphical Model Approach to Source Localization in Wireless Sensor Networks*, Technical Report ISIS-09-101, ISIS, Vanderbilt University, 2009.

2. Isaac Amundson, Manish Kushwaha, Xenofon Koutsoukos, Sandeep Neema, Janos Sztipanovits, *OASiS: A Service-Oriented Middleware for Pervasive Ambient-Aware Sensor Networks*, Technical Report ISIS-06-706, ISIS, Vanderbilt University, 2006.

REFERENCES

[1] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *SIGPLAN Not.*, vol. 35, no. 11, 2000, pp. 93–104.

[2] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, "Connecting the physical world with pervasive networks," in *Pervasive Computing, IEEE*, vol. 1, no. 1, Jan-Mar 2002, pp. 59–69.

[3] C.-Y. Chong and S. Kumar, "Sensor networks: evolution, opportunities, and challenges," in *Proceedings of the IEEE*, vol. 91, no. 8, Aug. 2003, pp. 1247–1256.

[4] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks," in *IEEE INFOCOM*, 2005.

[5] S. S. Blackman, *Multiple-Target Tracking with Radar Applications.* Artech House, 1986.

[6] Y. Bar-Shalom, *Tracking and data association.* Academic Press Professional, Inc., 1988.

[7] P. W.-C. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. J. Lobaton, M. L. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, J. D. Tygar, and S. S. Sastry, "CITRIC: A low-bandwidth wireless camera network platform," in *Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, September 2008.

[8] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The emergence of networking abstractions and techniques in TinyOS," in *NSDI*, 2004.

[9] M. Kushwaha, I. Amundson, P. Volgyesi, P. Ahammad, G. Simon, X. Koutsoukos, A. Ledeczi, and S. Sastry, "Multi-modal target tracking using heterogeneous sensor networks," in *In International Conference on Computer Communications and Networks (ICCCN 2008)*, 2008.

[10] M. Kushwaha, S. Oh, I. Amundson, X. Koutsoukos, and A. Ledeczi, "Target tracking in heterogeneous sensor networks using audio and video sensor fusion," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, August 2008.

[11] ——, "Target tracking in urban environments using audio-video signal processing in heterogeneous wireless sensor networks," in *42nd IEEE Asilomar Conference on Signals, Systems and Computers*, October 2008, pp. 26–29.

[12] ——, "Tracking in urban environments using sensor networks based on audio-video fusion," in *Handbook of Ambient Intelligence and Smart Environments (AISE).* Springer, June 2009.

[13] I. Amundson, M. Kushwaha, B. Kusy, P. Lolgyesi, G. Simon, X. Koutsoukos, and A. Ledeczi, "Time synchronization for multi-modal target tracking in heterogeneous sensor networks," in *Workshop on Networked Distributed Systems for Intelligent Sensing and Control*, June 2007.

[14] M. Kushwaha, X. Koutsoukos, P. Volgyesi, and A. Ledeczi, "Acoustic source localization and discrimination in urban environments," in *International Conference on Information Fusion*, 2009.

[15] M. Kushwaha and X. Koutsoukos, "A graphical model approach to source localization in wireless sensor networks," ISIS, Vanderbilt University, Tech. Rep. ISIS-09-101, 2009.

[16] M. Kushwaha, X. Koutsoukos, and S. Szilvasi, "Feature-based collaborative localization and discrimination in wireless sensor networks," in *ISIF Journal of Advances in Information Fusion (submitted)*, 2009.

[17] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *SenSys '04*, 2004.

[18] M. Castillo-Effer, D. Quintela, W. Moreno, R. Jordan, and W. Westhoff, "Wireless sensor networks for flash-flood alerting," in *Devices, Circuits and Systems, 2004. Proceedings of the Fifth IEEE International Caracas Conference on*, vol. 1, Nov. 2004, pp. 142–146.

[19] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton, "Sensor networks for emergency response: challenges and opportunities," in *Pervasive Computing, IEEE*, vol. 3, no. 4, Oct.-Dec. 2004, pp. 16–23.

[20] T. Gao, D. Greenspan, M. Welsh, R. Juang, and A. Alm, "Vital signs monitoring and patient tracking over a wireless network," in *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, Jan. 2005, pp. 102–105.

[21] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," in *IEEE Internet Computing*, vol. 10, no. 2, 2006, pp. 18–25.

[22] M. Kushwaha, K. Molnar, J. Sallai, P. Volgyesi, M. Maroti, and A. Ledeczi, "Sensor node localization using mobile acoustic beacons," in *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, Nov. 2005.

[23] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler, "The effects of ranging noise on multihop localization: an empirical study," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, April 2005, pp. 73–80.

[24] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Operating Systems Design and Implementation (OSDI)*, 2002.

[25] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, and D. Culler, "Elapsed time on arrival: A simple and versatile primitive for time synchronization services," vol. 2, no. 1, January 2006.

[26] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," in *Wireless Communications, IEEE*, vol. 11, no. 6, Dec. 2004, pp. 6–28.

[27] M. Demirbas, "Scalable design of fault-tolerance for wireless sensor networks," Ph.D. dissertation, The Ohio State University, 2004.

[28] B. Dasarathy, "What, where, why, when, and how?" in *Inform. Fus. (Editorial)*, vol. 2, no. 2, January 2001, pp. 75–76.

[29] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," in *ACM Computing Surveys*, vol. 39, no. 3, 2007, p. 9.

[30] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," in *Commun. ACM*, vol. 47, no. 6, 2004, pp. 53–57.

[31] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," in *ACM Trans. Embed. Comput. Syst.*, vol. 6, no. 4, 2007.

[32] L. Luo, T. F. Abdelzaher, T. He, and J. A. Stankovic, "Envirosuite: An environmentally immersive programming framework for sensor networks," in *ACM Transactions on Embedded Computing Systems*, vol. 5, 2006.

[33] M. Kushwaha, I. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits, "Oasis: A programming framework for service-oriented sensor networks," in *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on*, Jan. 2007, pp. 1–8.

[34] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 28–39.

[35] N. H. Cohen, A. Purakayastha, J. Turek, L. Wong, and D. Yeh, "Challenges in flexible aggregation of pervasive data," IBM Research Division, Tech. Rep. RC 21942 (98646), January 2001.

[36] R. Kumar, V. Tsiatsis, and M. B. Srivastava, "Computation hierarchy for in-network processing," in *Proc. of the 2nd Intl. Workshop on Wireless Networks and Applications (WSNA03)*, 2003.

[37] S. Rhee, D. Seetharam, and S. Liu, "Techniques for minimizing power consumption in low data-rate wireless sensor networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC04)*, 2004.

[38] E. Duarte-Melo and M. Liu, "Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks," in *IEEE Globecom*, 2002.

[39] D. Reid, "An algorithm for tracking multiple targets," vol. 24, no. 6, pp. 843–854, December 1979.

[40] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina at Chapel Hill, Tech. Rep. TR 95-041, July 2006.

[41] T. Zhao and R. Nevatia, "Tracking multiple humans in crowded environment," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[42] Z. Khan, T. Balch, and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," vol. 27, no. 11, pp. 1805–1918, Nov. 2005.

[43] K. Okuma, A. Taleghani, N. de Freitas, J. Little, and D. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *European Conference on Computer Vision*, 2004.

[44] S. Oh, S. Russell, and S. Sastry, "Markov chain Monte Carlo data association for general multiple-target tracking problems," in *Proc. of the IEEE Conference on Decision and Control*, Paradise Island, Bahamas, Dec. 2004.

[45] L. D. Stone, T. L. Corwin, and C. A. Barlow, "Bayesian multiple target tracking." Artech House, 1999.

[46] M. S. Arulampalam, S. Maskell, and N. Gordon, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," pp. 174–188, 2002.

[47] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[48] M. Orton and W. Fitzgerald, "A bayesian approach to tracking multiple targets using sensor arrays and particle filters," in *IEEE Transactions on Signal Processing*, vol. 50, 2002, pp. 216–223.

[49] M. R. Morelande, C. M. Kreucher, and K. Kastella, "A bayesian approach to multiple target detection and tracking," in *IEEE Transactions on Signal Processing*, vol. 55, 2007, pp. 1589–1604.

[50] M. J. Beal, N. Jojic, and H. Attias, "A graphical model for audiovisual object tracking," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, 2003, pp. 828–836.

[51] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," in *IEEE Journal on Selected Areas in Communications*, vol. 23, 2005, pp. 809–819.

[52] J. Goutsias, R. Mahler, and H. T. Nguyen, *Random Sets Theory and Applications*. Springer-Verlag New York, 1997.

[53] I. Goodman, R. Mahler, and H. T. Nguyen, *Mathematics of Data Fusion*. Kluwer Academic Publishers, 1997.

[54] R. Mahler, *An Introduction to Multisource-Multitarget Statistics and Applications*. Lockheed Martin Technical Monograph, 2000.

[55] ——, "Multi-target bayes filtering via first-order multi-target moments," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, 2003.

[56] R. Maher, "A survey of phd filter and cphd filter implementations," in *SPIE Signal Processing, Sensor Fusion, and Target Recognition XVI*, vol. 6567, 2007.

[57] B. ngu Vo, S. Singh, and A. Doucet, "Sequential monte carlo implementation of the phd filter for multi-target tracking," in *In Proceedings of the Sixth International Conference on Information Fusion*, 2003.

[58] T. Zajic, R. Ravichandran, R. Mahler, R. Mehra, and M. Noviskey, "Joint tracking and identification with robustness against unmodeled targets," in *Signal Processing, Sensor Fusion and Target Recognition XII*, vol. 5096, 2003.

[59] D. Hall and J. Llinas, "An introduction to multisensor data fusion," in *Proceedings of the IEEE*, vol. 85, 1997, pp. 6–23.

[60] R. Luo, C.-C. Yih, and K. L. Su, "Multisensor fusion and integration: approaches, applications, and future research directions," in *Sensors Journal, IEEE*, vol. 2, no. 2, Apr 2002, pp. 107–119.

[61] H. F. Durrant-Whyte, "Sensor models and multisensor integration," in *International Journal of Robotics Research*, vol. 7, no. 6, 1988, pp. 97–113.

[62] B. V. Dasarathy, "Sensor fusion potential exploitation-innovative architectures and illustrative applications," in *Proceedings of the IEEE*, vol. 85, no. 1, 1997, pp. 24–38.

[63] A. P. Dempster, "A generalization of bayesian inference," in *J. Royal Stat. Soc.*, vol. Series B, no. 30, 1968, pp. 205–247.

[64] V. Novk, I. Perfilieva, and J. Mockor, *Mathematical Principles of Fuzzy Logic*. Springer, 1999.

[65] P. P. Bonissone, "Soft computing: The convergence of emerging reasoning technologies," in *Soft Computing*, vol. 1, 1997, pp. 6–18.

[66] T. Kohonen, *Self-Organizing Maps*. Springer, 1997.

[67] I. Liggins, M.E., C.-Y. Chong, I. Kadar, M. Alford, V. Vannicola, and S. Thomopoulos, "Distributed fusion architectures and algorithms for target tracking," in *Proceedings of the IEEE*, vol. 85, no. 1, Jan. 1997, pp. 95–107.

[68] R. Murphy, "Biological and cognitive foundations of intelligent sensor fusion," in *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 26, no. 1, Jan 1996, pp. 42–51.

[69] T. G. R. Bower, *Perception: Essays in Honor of James J. Gibson*. Cornell Univ. Press, 1974, ch. The evolution of sensory systems, pp. 141–153.

[70] S. S. Haykin and J. H. Justice, *Array Signal Processing*. Prentice-Hall, Inc., 1985.

[71] H. Krim and M. Viberg, "Two decades of array signal processing research: the parametric approach," in *IEEE Signal Processing Magazine*, vol. 13, no. 4, July 1996.

[72] H. Wang and M. Kaveh, "Coherent signal-subspace processing for the detection and estimation of angles of arrival of multiple wide-band sources," in *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 33, no. 4, Aug 1985, pp. 823–831.

[73] S. Valaee and P. Kabal, "Wideband array processing using a two-sided correlation transformation," in *Signal Processing, IEEE Transactions on*, vol. 43, no. 1, Jan 1995, pp. 160–172.

[74] H. Schau and A. Robinson, "Passive source localization employing intersecting spherical surfaces from time-of-arrival differences," in *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, no. 8, Aug 1987, pp. 1223–1225.

[75] J. Smith and J. Abel, "Closed-form least-squares source location estimation from range-difference measurements," in *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, no. 12, Dec 1987, pp. 1661–1669.

[76] S. M. Kay, *Fundamentals of Statistical Processing, Volume I: Estimation Theory.* Prentice Hall PTR, 1997.

[77] Y. Chan and K. Ho, "A simple and efficient estimator for hyperbolic location," in *Signal Processing, IEEE Transactions on*, vol. 42, no. 8, Aug 1994, pp. 1905–1915.

[78] M. S. Brandstein, J. E. Adcock, and H. F. Silverman, "A closed-form location estimator for use with room environment microphone arrays," in *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 5, no. 1, Jan 1997, pp. 45–50.

[79] J. Vermaak and A. Blake, "Nonlinear filtering for speaker tracking in noisy and reverberant environments," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001*, 2001, pp. 3021–3024.

[80] D. B. Ward and R. C. Williamson, "Particle filter beamforming for acoustic source localization in a reverberant environment," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2002*, 2002, pp. 1777–1780.

[81] S. Birchfield and D. Gillmor, "Fast bayesian acoustic localization," in *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, vol. 2, 2002, pp. 1793–1796.

[82] S. T. Birchfield, "A unifying framework for acoustic localization," in *Proceedings of the 12th European Signal Processing Conference (EUSIPCO)*, Vienna, Austria, September 2004.

[83] D. Li and Y. H. Hu, "Energy-based collaborative source localization using acoustic microsensor array," in *EURASIP Journal on Applied Signal Processing*, vol. 2003, 2003, pp. 321–337.

[84] X. Sheng and Y.-H. Hu, "Maximum likelihood multiple source localization using acoustic energy measurements with wireless sensor networks," in *IEEE Transactions On Signal Processing*, vol. 53, 2005, pp. 44–53.

[85] C. Meesookho, U. Mitra, and S. Narayanan, "On energy-based acoustic source localization for sensor networks," in *IEEE Transactions On Signal Processing*, vol. 56, 2008, pp. 365–377.

[86] T. Manickam, R. Vaccaro, and D. Tufts, "A least-squares algorithm for multipath time-delay estimation," in *Signal Processing, IEEE Transactions on*, vol. 42, no. 11, Nov 1994, pp. 3229–3233.

[87] J. J. Fuchs, "Multipath time-delay detection and estimation," in *Signal Processing, IEEE Transactions on*, vol. 47, no. 1, 1999, pp. 237–243.

[88] S. Doclo and M. Moonen, "Robust adaptive time delay estimation for speaker localization in noisy and reverberant acoustic environments," in *EURASIP J. Appl. Signal Process.*, vol. 2003, 2003, pp. 1110–1124.

[89] R. Tremblay, G. Carter, and D. Lytle, "A practical approach to the estimation of amplitude and time-delay parameters of a composite signal," in *Oceanic Engineering, IEEE Journal of*, vol. 12, no. 1, Jan 1987, pp. 273–278.

[90] A. Ledeczi, A. Nadas, P. Volgyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dora, K. Molnar, M. Maroti, and G. Simon, "Countersniper system for urban warfare," vol. 1, no. 2, 2005.

[91] J. C. Chen, R. E. Hudson, and K. Yao, "Maximum-likelihood source localization and unknown sensor location estimation for wideband signals in the near-field," in *IEEE Trans. Signal Processing*, vol. 50, August 2002, pp. 1843–1854.

[92] J. C. Chen, K. Yao, and R. E. Hudson, "Acoustic source localization and beamforming: theory and practice," in *EURASIP Journal on Applied Signal Processing*, April 2003, pp. 359–370.

[93] P. Bergamo, S. Asgari, H. Wang, D. Maniezzo, L. Yip, R. E. Hudson, K. Yao, and D. Estrin, "Collaborative sensor networking towards real-time acoustical beamforming in free-space and limited reverberence," in *IEEE Transactions On Mobile Computing*, vol. 3, no. 3, 2004, pp. 211–224.

[94] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," in *ACM Computing Surveys*, vol. 38, no. 4, 2006.

[95] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," in *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, May 2003, pp. 564–577.

[96] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," in *International Journal of Computer Vision*, vol. 26, 1998, pp. 63–84.

[97] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *IEEE International Conference on Computer Vision*, 1999.

[98] K. Ohba, K. Ikeuchi, and Y. Sato, "Appearance-based visual learning and object recognition with illumination invariance," in *Mach. Vision Appl.*, vol. 12, no. 4, 2000, pp. 189–196.

[99] J. Triesch and C. Von Der Malsburg, "Democratic integration: Self-organized integration of adaptive cues," in *Neural Computation*, vol. 13, no. 9, 2001, pp. 2049–2074.

[100] E. Hayman and J.-O. Eklundh, "Probabilistic and voting approaches to cue integration for figure-ground segmentation," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, 2002, pp. 469–486.

[101] J. Canny, "A computational approach to edge detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, 1986, pp. 679–698.

[102] H. Moravec, "Visual mapping by a robot rover," in *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, August 1979, pp. 599–601.

[103] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[104] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," in *Int. J. Comput. Vision*, vol. 60, 2004, pp. 91–110.

[105] B. K. P. Horn and B. G. Schunck, "Determining optical flow," in *Artificial Intelligence*, vol. 17, 1981, pp. 185–203.

[106] J. Y. Aloimonos and D. Shulman, *Integration of visual modules: an extension of the Marr paradigm*. San Diego, CA, USA: Academic Press Professional, Inc., 1989.

[107] R. Filipovych and E. Ribeiro, "Probabilistic combination of visual cues for object classification," in *LNCS Advances in Visual Computing*, vol. 4841, 2007, pp. 662–671.

[108] D. Kragic and H. Christensen, "Cue integration for visual servoing," in *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 1, Feb 2001, pp. 18–27.

[109] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, 1997, pp. 780–785.

[110] X. Gao, T. Boult, F. Coetzee, and V. Ramesh, "Error analysis of background adaption," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, 2000, pp. 503–510.

[111] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Towards robust automatic traffic scene analysis in realtime," in *IEEE Conference on Decision and Control*, 1994.

[112] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," in *Conference on Uncertainty in Artificial Intelligence*, 1997.

[113] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *IEEE ICCV'99 Frame-Rate workshop*, 1999.

[114] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[115] P. KaewTraKulPong and R. B. Jeremy, "An improved adaptive background mixture model for realtime tracking with shadow detection," in *Workshop on Advanced Video Based Surveillance Systems (AVBS)*, 2001.

[116] D. Comaniciu, P. Meer, and S. Member, "Mean shift: A robust approach toward feature space analysis," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, 2002, pp. 603–619.

[117] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[118] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, vol. 2, Oct. 2003, pp. 734–741.

[119] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *IEEE Conference on Computer Vision and Pattern Recognition, 1998.*, 1998, pp. 232–237.

[120] H. Tao, H. Sawhney, and R. Kumar, "Object tracking with bayesian estimation of dynamic layer representations," in *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 1, Jan 2002, pp. 75–89.

[121] M. Isard and J. MacCormick, "Bramble: a bayesian multiple-blob tracker," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, 2001, pp. 34–41.

[122] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, 2002, pp. 661–675.

[123] Q. Delamarre and O. Faugeras, "3d articulated models and multi-view tracking with silhouettes," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, pp. 716–721.

[124] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf, "The evolution from single to pervasive smart cameras," in *Second ACM/IEEE International Conference on Distributed Smart Cameras, 2008 (ICDSC 2008)*, 2008.

[125] A. Tyagi, M. Keck, J. Davis, and G. Potamianos, "Kernel-based 3d tracking," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[126] S. Fleck, F. Busch, and W. Straßer, "Adaptive probabilistic tracking embedded in smart cameras for distributed surveillance in a 3d model," in *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, 2007, pp. 24–24.

[127] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl, "Autonomous multicamera tracking on embedded smart cameras," in *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, 2007, pp. 35–35.

[128] A. V. Nefian, L. H. Liang, T. Fu, , and X. X. Liu, "A bayesian approach to audio-visual speaker identification," in *Audio- and Video-Based Biometric Person Authentication*, vol. 2688, June 2003, pp. 761–769.

[129] M. Kaynak, Q. Zhi, A. Cheok, K. Sengupta, Z. Jian, and K. C. Chung, "Analysis of lip geometric features for audio-visual speech recognition," in *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 34, no. 4, July 2004, pp. 564–570.

[130] D. N. Zotkin, R. Duraiswami, and L. S. Davis, "Joint audio-visual tracking using particle filters," in *EURASIP J. Appl. Signal Process.*, vol. 2002, no. 1, 2002, pp. 1154–1164.

[131] B. H. Yoshimi and G. S. Pingali, "A multimodal speaker detection and tracking system for teleconferencing," in *ACM Multimedia '02*, 2002.

[132] N. Checka, K. Wilson, V. Rangarajan, and T. Darrell, "A probabilistic framework for multimodal multi-person tracking," in *IEEE Workshop on Multi-Object Tracking*, 2003.

[133] B. Bhanu and X. Zou, "Moving humans detection based on multi-modal sensor fusion," in *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop*, 2004, p. 136.

[134] W. Zajdel, J. Krijnders, T. Andringa, and D. Gavrila, "Cassandra: audio-video sensor fusion for aggression detection," in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, Sept. 2007, pp. 200–205.

[135] K. Kalgaonkar, P. Smaragdis, and B. Raj, "Sensor and data systems, audio-assisted cameras and acoustic doppler sensors," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1–2.

[136] J. Houser and L. Zong, "The arl multi-modal sensor: A research tool for target signature collection, algorithm validation, and emplacement studies," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1–2.

[137] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb, "Plan-view trajectory estimation with dense stereo background models," in *Proceedings. Eighth IEEE International Conference on Computer Vision (ICCV 2001)*, 2001.

[138] Z. Zhu and T. S. Huang, *Multimodal Surveillance: Sensors, Algorithms, and Systems.* Artech House Publishers, 2007.

[139] M. Ding, A. Terzis, I.-J. Wang, and D. Lucarelli, "Multi-modal calibration of surveillance sensor networks," in *Military Communications Conference, MILCOM 2006*, 2006.

[140] H. Y. Hau and R. L. Kashyap, "On the robustness of Dempster's rule of combination," in *IEEE International Workshop on Tools for Artificial Intelligence*, 1989.

[141] I. Amundson, B. Kusy, P. Volgyesi, X. Koutsoukos, and A. Ledeczi, "Time synchronization in heterogeneous sensor networks," in *International Conference on Distributed Computing in Sensor Networks (DCOSS 2008)*, 2008.

[142] P. Volgyesi, G. Balogh, A. Nadas, C. Nash, and A. Ledeczi, "Shooter localization and weapon classification with soldier-wearable networked sensors," in *Mobisys*, 2007.

[143] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *ACM SenSys*, 2003.

[144] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *ACM SenSys*, 2004.

[145] J. Sallai, B. Kusy, A. Ledeczi, and P. Dutta, "On the scalability of routing integrated time synchronization," in *Workshop on Wireless Sensor Networks (EWSN)*, 2006.

[146] J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," in *EURASIP, Journal on Applied Signal Processing*, 2002.

[147] A. Poore, "Multidimensional assignment and multitarget tracking," in *Partitioning Data Sets*, I. J. Cox, P. Hansen, and B. Julesz, Eds. American Mathematical Society, 1995, pp. 169–196.

[148] L. Girod, M. Lukac, V. Trifa, and D. Estrin, "The design and implementation of a self-calibrating distributed acoustic sensing platform," in *SenSys '06*, 2006.

[149] M. Meingast, M. Kushwaha, S. Oh, X. Koutsoukos, and S. S. Akos Ledeczi, "Heterogeneous camera network localization using data fusion," in *In ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-08)*, 2008.

[150] C. Savarese, J. Rabaey, and J. Beutel, "Location in distributed ad-hoc wireless sensor networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001*, vol. 4, 2001, pp. 2037–2040.

[151] A. M. Ali, K. Yao, T. C. Collier, C. E. Taylor, D. T. Blumstein, and L. Girod, "An empirical study of collaborative acoustic source localization," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 41–50.

[152] M. I. Jordan, *Learning in Graphical Models*. The MIT Press, 1998.

[153] C. P. Robert and G. Casella, *Monte Carlo statistical method*. (Springer Texts in Statistics) Springer-Verlag, 2004.

[154] R. Mahler, "PHD filters for nonstandard targets, II: Unresolved targets," in *12th International Conference on Information Fusion*, 2009, pp. 922–929.

[155] C. Serviere and P. Fabry, "Blind source separation of noisy harmonic signals for rotating machine diagnosis," vol. 272, no. 1-2, 2004.

[156] B. Porat, *Digital processing of random signals: theory and methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994.

[157] F. V. Jensen, *Bayesian Networks and Decision Graphs*. Springer, 2001.

[158] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," in *The Journal of Machine Learning Research*, vol. 3, 2003, pp. 993–1022.

[159] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," in *Biometrika*, vol. 57, no. 1, 1970, pp. 97–109.

[160] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, 1984, pp. 721–741.

[161] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2002.

[162] K. Matusita, "Decision rules, based on the distance, for problems of fit, two samples, and estimation," vol. 26, no. 4, 1955, pp. 631–640.

[163] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, 2002, pp. 971–987.

[164] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, August 2002.

[165] R. G. Brown and P. Y. Hwang, *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises and Solutions*, 3rd ed. John Wiley & Sons, November 1996, ch. 8: Smoothing.

[166] J. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," ICSI, Tech. Rep. TR-97-021, 1997.