This paper was accepted for publication in the International Journal of Sensor Networks. The final accepted version is currently available to subscribers and pay-per-view users at http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijsnet

Resource-Aware Distributed Particle Filtering for Cluster-Based Object Tracking in Wireless Camera Networks

Kihyun Hong

Creative Innovation Center
LG Electronics
Seoul, Korea
E-mail: kihyun.hong@lge.com

Henry Medeiros

Department of Electrical and Computer Engineering
Marquette University
Milwaukee, WI 53210, USA
E-mail: henry.medeiros@marquette.edu

Paul J. Shin

VM Labs
VideoMining Corporation
State College, PA 16801, USA
E-mail: pshin@videomining.com

Johnny Park

Department of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906, USA
E-mail: jpark@purdue.edu

Abstract This paper presents a novel resource-aware framework for the implementation of distributed particle filters in resource-constrained wireless camera networks (WCNs). WCNs often suffer from communication failures caused by physical limitations of the communication channel as well as network congestion. Unreliable communication degrades the visual information shared by the cameras, such as visual feature data, and consequently leads to inaccurate vision processing at individual camera nodes. This paper focuses on the effects of communication failures on object tracking performance and presents a novel communication resource-aware tracking methodology, which adjusts the amount of data packets transmitted by the cameras according to the network conditions. We demonstrate the performance of the proposed framework using three different mechanisms to share the particle information among nodes: synchronized particles, Gaussian mixture models, and Parzen windows. The experimental results show that the proposed resource-aware method makes the distributed particle filters more tolerant to packet losses and also more energy efficient.

Keywords: wireless camera networks, particle filters, object tracking, resource allocation, collaborative processing.

Reference to this paper should be made as follows: Hong, K., Medeiros, H., Shin, P. and Park, P. (2014) 'Resource-Aware Distributed Particle Filtering for Cluster-Based Object Tracking in Wireless Camera Networks' *Int. J. Sensor Networks*, Vol. x, No. x, pp xx-xx.

Biographical notes: Kihyun Hong is currently a chief research engineer in the Smart Car R&D Lab, Creative Innovation Center at LG Electronics, Seoul, Korea. His research interests are in the development and application of sensor/data fusion, multiple camera system, computer vision, and video/image signal processing. He received the B.Eng. degree from Kwangwoon University, Seoul, Korea, the M.S. in electrical engineering from Stanford University, CA and Ph.D. degree from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, in 2013.

Henry Medeiros is an Assistant Professor of Electrical and Computer Engineering at Marquette University. Before joining Marquette, he was a Research Scientist at the School of Electrical and Computer Engineering at Purdue University and the Chief Technology Officer of Spensa Technologies, a high tech start up company located at the Purdue Research Park. He received his Ph.D. from the School of Electrical and Computer Engineering at Purdue University in 2010. His research interests include sensor networks, computer vision, robotics, and embedded systems.

Dr. Paul J. Shin received his Ph.D. in Electrical and Computer Engineering from Purdue University, West Lafayette in 2013 and his B.E. in Electrical Engineering from Korea University. He is currently director of research in VideoMining Corp, where he leads the company's R&D activities for future products. His research interests lie in the broad area of building real-world Cyber-Physical Systems based on resource-constrained networked embedded devices and real-time information processing and network protocols in large-scale distributed networks. He won the Best Poster Award in ACM/IEEE International Conference on Distributed Smart Cameras in 2008.

Johnny Park received the B.S., M.S., and Ph.D. degrees from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, in 1998, 2000, and 2004, respectively. He is currently a Research Assistant Professor in the School of Electrical and Computer Engineering at Purdue University. His research interests span various topics in distributed sensor networks, computer graphics, computer vision, and robotics.

1 Introduction

Along with an increasing demand on networked camera applications such as security and surveillance, the recent advances of wireless camera hardware, e.g., Imote2 (Nachman et al. [2008]) equipped with a camera, has brought on much interest in the development of network-based computer vision algorithms, especially object tracking algorithms for wireless camera networks (WCNs). Previously, in Medeiros et al. [2007], we introduced an event-driven camera clustering protocol for collaborative object tracking in camera networks. We also presented a Kalman filter based object tracking approach in a wireless network of multiple cameras using the clustering protocol in Medeiros et al. [2008a]. However, even though the tracking algorithm was designed to run in WCNs, it did not explicitly account for the effects of communication failures, network congestion, and computational load on the performance of more sophisticated collaborative tracking methods.

A number of previous works (Coates [2004], Sheng and Hu [2005], Huang et al. [2008]) on wireless sensor networks have presented object tracking algorithms which consider constraints related to sensor hardware such as low computational and communication capabilities. These works proposed distributed tracking methods for wireless sensor networks, which perform complicated tracking processing by spreading the computational load to local network nodes. They also presented several mechanisms to reduce the amount of data communication, e.g., incorporating data compression or parametric data representation of the communication data for the distributed tracking systems. Although these algorithms were not especially designed for networks of wireless cameras, they showed tracking algorithm implementation methodologies under such hardware limitations.

Besides the computational hardware constraints, far too little attention has been paid to communication and networking issues, such as communication channel characteristics and network traffic, in the design of object tracking methods for WCNs. These communication and network issues are even more critical in a WCN because vision applications tend to generate heavy network traffic and communication failures (Shin et al. [2011]). Specifically, communication failures degrade the information shared by local sensor nodes for collaborative tracking and have severe effects on object tracking performance.

This paper focuses on the effects of communication failures on object tracking performance and presents a communication resource-aware tracking methodology, which adjusts the amount of data packet transmission according to the network conditions. In this paper, valid communication packets, which are not dropped in data communications, are considered as an available communication resource for distributed tracking. Our approach allows sensor nodes to consume communication energy efficiently and reduces tracking performance degradation due to communication failures.

For our implementation we employed a widely used object tracking approach, the particle filter, using the clustering algorithm in Medeiros et al. [2007] as the underlying framework for collaborative processing. The key

features of our system are as follows: (1) particle filtering is performed by dynamic clusters of cameras in a distributed manner, and (2) the amount of data representing particle probabilities is adaptively adjusted based on the conditions of the network on which the proposed tracker operates.

This paper begins by discussing design challenges imposed by WCNs in details in Section 2. It then goes on to introduce the proposed distributed particle filtering framework, which is the main building block of our implementation, in Section 3. The implementation of the resource-aware and distributed particle filter for cluster-based WCNs is described in Section 4. Experimental results are shown in Section 5 and Section 6 concludes the paper.

2 Challenges in Wireless Camera Networks

Design specifications for computer vision algorithms such as object tracking for WCNs are constrained by the characteristics of the camera node hardware, communication channel, network topology, network traffic, etc. In this section, we review the issues which need to be considered when developing vision algorithms for WCNs.

Let us first consider design challenges caused by hardware. Wireless smart cameras consisting of sensing, data processing, and communication units impose constraints on image quality, computational complexity of the vision algorithm, and data communication bandwidth. This is particularly true when severely resource-constrained mote-based embedded systems such as the Imote2 are utilized as wireless smart cameras. Computational limitations prevent smart camera systems from executing complicated vision algorithms, including most current state-of-the-art approaches. Low communication bandwidth and limited transmission energy available in a sensor node make it difficult to transmit raw image data and even sets of visual feature data. We can consider lightweight and distributed computer vision processing as one of the most promising approaches for reducing the computational burden in individual nodes. However, it is crucial to develop distributed methods that require only a small amount of data communication.

WCNs often suffer from unreliable communication, as wireless communication channels suffer from effects such as Rayleigh fading and inherently impose packet losses (Rappaport [2001]). Unreliable communication causes data losses and degrades the overall quality of the received visual information, consequently leading to inaccurate vision processing at the camera nodes. Hence, in designing computer vision algorithms for WCNs, it is necessary that the distributed vision processing be robust to imperfectly communicated data.

In applying a vision task for WCNs, it is also important to consider mechanisms for collaborative processing. A cluster-based approach as suggested in Medeiros et al. [2007, 2008a] allows camera networks to work collaboratively. This collaborative processing causes additional

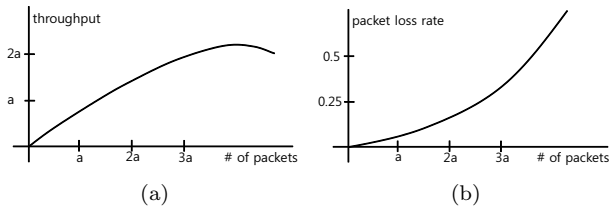


Figure 1: Typical characteristics of wireless network traffic. (a) throughput vs. number of transmitted packets and (b) packet loss rate vs. transmitted packets.

data traffic on the clustered nodes when an event of interest is detected by a cluster. Intensive data traffic in a cluster tends to cause packet collisions and additional packet losses, and has a degrading effect on the performance of the vision tasks. Fig. 1 shows typical throughput and packet loss rate curves in wireless networks. As shown in Fig. 1(a), there is a limit to the throughput that can be obtained by the network. This is essentially caused by the fact that, as shown in Fig. 1(b), the packet loss rate increases sharply as more packets are transmitted in the network.

Data congestion and data loss are handled, in general, by reliable transport protocols, quality of service (QoS) mechanisms or by over-provisioning network capacity. Transport protocols such as TCP, however, are not suitable for delay sensitive applications, e.g., real time tracking applications, because such protocols as well as scheduling-based congestion control protocols can cause excessive latency (Khan et al. [2012]). An approach of over-provisioning network bandwidth for the expected peak traffic load is not usually applicable to resource constrained wireless camera nodes either, since the nodes have tightly limited hardware capabilities. In a WCN in which network capacity is limited, data loss during communication is inevitable. Vision applications for the wireless network need to be dynamically adaptable under varying traffic conditions, for example, by controlling the amount of data transmitted without exceeding the available bandwidth in the network.

3 Distributed Particle Filtering Algorithms

In the particle filtering procedure, probabilities are approximated with discrete particle samples. Many previous works showed that this discrete approximation can effectively represent object probabilities for single camera based object tracking (Isard and Blake [1998], Perez et al. [2002], Nummiaro et al. [2002]). For distributed particle filtering using multiple cameras, each camera computes its object probability locally, and the global object probability is obtained by fusing the local object probabilities of the camera nodes. In this case, however, the discrete nature of the method requires particle sample synchronization in order to compute the joint

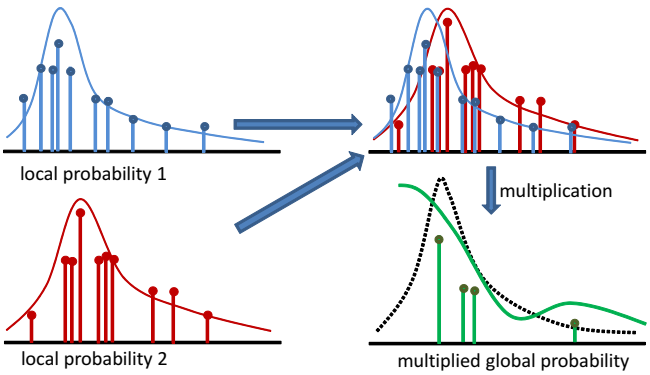


Figure 2: Particle synchronization problem to compute a joint probability. The top-right plot shows the overlapped local probabilities and the bottom-right plot illustrates the multiplied probability of the two local probabilities. In the global probability plot, the black dotted line indicates the desired probability density and the solid green line shows the global probability computed by the multiplication of two asynchronous discrete local probabilities.

(global) probability (Coates [2004], Ridley et al. [2004], Ong et al. [2006b, 2008]). Note that we use the term *synchronization* to describe the fact that all local particle sets have the same support points as defined in Coates [2004]. Fig. 2 shows the synchronization problem when two different camera nodes must build a joint probability. In the figure, each vertical bar represents a weighted particle sample. As shown on the left side of the figure, local probabilities corresponding to asynchronous particles from different cameras, which have different support points in their probability representations, lead to inaccurate joint probability estimates. Furthermore, as the erroneous probability propagates frame by frame, it deteriorates the recursive distributed particle filtering and causes severe tracking performance degradation. In distributed particle filter implementations, one of the main challenges is to make all the camera particles synchronized (Coates [2004], Ridley et al. [2004], Ong et al. [2005, 2006a]).

We consider two approaches for handling the particle synchronization issue. The first approach, which is an academic exercise to show the effects of perfect particle synchronization, synchronizes all local particles by forcing the local random number generators to use the same random seed. The second and more practical approach is to convert discrete probabilities to continuous forms that do not require synchronization. We will refer to the first approach as *synchronized particle filtering*, and to the second as *probability conversion based particle filtering*. Before describing these two approaches, we first introduce a general multiple camera particle filtering framework, which lays the groundwork for our distributed particle filter implementations.

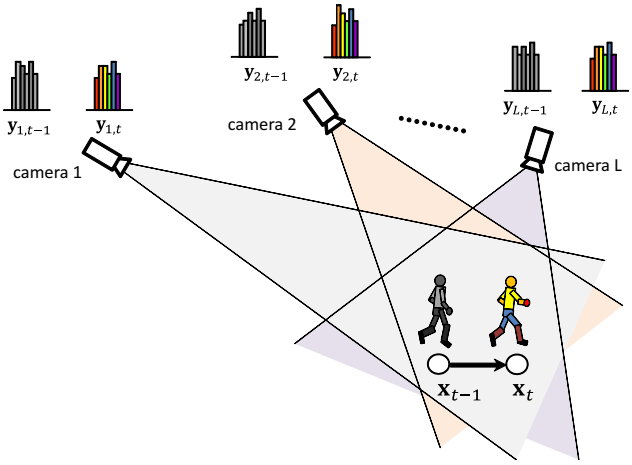


Figure 3: Overview of a multiple camera based object tracking system.

3.1 Multiple Camera Particle Filter

Given L cameras, we denote the observation random vector from the i^{th} camera at time t as $\mathbf{Y}_{i,t}$ and its realization as $\mathbf{y}_{i,t}$. The random process of an object state at time t is denoted as $\mathbf{X}_t$ and its realization as $\mathbf{x}_t$. Fig. 3 illustrates the overview of a multiple camera based object tracking system. In the figure, $\mathbf{y}_{i,t}$ represents an extracted object feature, e.g., a color histogram, from an image captured at the i^{th} camera at time t. The dynamic model of the object is defined by a state process model:

$$\mathbf{X}_{t+1} = f_t(\mathbf{X}_t, \mathbf{U}_t). \tag{1}$$

The observation model, assuming the observations are independent given the state, is defined as

$$\begin{aligned} \mathbf{Y}_{1,t} &= h_{1,t}(\mathbf{X}_t, \mathbf{V}_{1,t}) \\ &\vdots \\ \mathbf{Y}_{L,t} &= h_{L,t}(\mathbf{X}_t, \mathbf{V}_{L,t}), \end{aligned} \tag{2}$$

where $\mathbf{U}_t$ and $\mathbf{V}_{i,t}$ are independent and identically distributed (i.i.d.) white noise processes, and f_t and $h_{i,t}$ are assumed to be known functions. Also, each observation model can be characterized in probability by the associated likelihood

$$\mathbf{Y}_{i,t} = h_{i,t}(\mathbf{X}_t, \mathbf{V}_{i,t}) \Leftrightarrow p(\mathbf{y}_{i,t}|\mathbf{x}_t). \tag{3}$$

Then, the likelihood probability density function (PDF) given all the observations has the form

$$p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^{L} p(\mathbf{y}_{i,t}|\mathbf{x}_t), \tag{4}$$

where $\mathbf{y}_t = ((\mathbf{y}_{1,t})^T, \cdots, (\mathbf{y}_{L,t})^T)^T$.

Fig. 4 shows the graphical model of the dependencies between the object state and the L camera observations. The graphical model illustrates the evolution of the system over time as a hidden Markov dynamic model. The directed link from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ represents the state transition process in Eq. (1) with its associated probability,

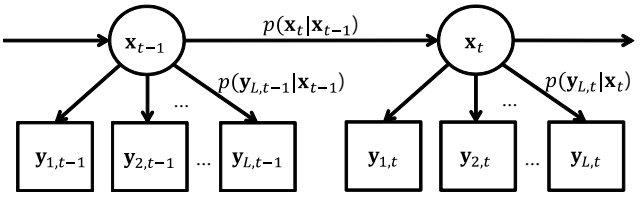


Figure 4: Dynamic Markov model for multiple observation based tracking.

Algorithm 1 Multiple observation based particle filter.

Given: Random measure of previous posterior PDF, $\{\tilde{\mathbf{x}}_{t-1}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$.
Given: Observations at time t, $\mathbf{y}_{1,t}, \cdots, \mathbf{y}_{L,t}$.
Output: Random measure of current posterior PDF, $\{\tilde{\mathbf{x}}_{t}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$.
Output: MMSE estimate of current state, $E(\mathbf{X}_t|\mathbf{y}_{0:t})$.

1: Do sampling: $\mathbf{x}_t^{(k)} = f_t(\tilde{\mathbf{x}}_{t-1}^{(k)}) + \mathbf{u}_t$.
2: Compute (local) likelihood weights:
$w_{i,t}^{(k)} = p(\mathbf{y}_{i,t}|\mathbf{x}_t^{(k)})$.
3: Compute joint (global) likelihood weights:
$w_t^{(k)} = \prod_{i=1}^{L} w_{i,t}^{(k)}$.
4: Compute MMSE estimate:
$E(\mathbf{X}_t|\mathbf{y}_{0:t}) = \sum_{k=1}^{K} w^{(k)}\mathbf{x}^{(k)}$.
5: Do resampling: $\left\{\tilde{\mathbf{x}}_t^{(k)}, \frac{1}{K}\right\}_{m=1}^{K} \Leftarrow \{\mathbf{x}_t^{(k)}, w_t^{(k)}\}_{k=1}^{K}$.

$p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The directed link from $\mathbf{x}_t$ to $\mathbf{y}_{i,t}$ represents the local observation process at the i^{th} camera in Eq. (3).

Particle filtering can be implemented through several approaches, such as sequential importance sampling (SIS) (Gordon et al. [1993], Isard and Blake [1998]) and sequential importance resampling (SIR) (Gordon et al. [1993]). In this work, the SIR particle filter is utilized because it is simple, avoids the particle degeneration problem, and is commonly used in many visual object tracking applications. At each time instant, a tracking result is obtained by the minimum mean squared error (MMSE) estimate of the target state. Alg. 1 describes the SIR particle filter utilizing multiple independent observations, where K is the number of particles. In Alg. 1, $\mathbf{x}_t^{(k)}$ and $\tilde{\mathbf{x}}_t^{(k)}$ are the k^{th} particle and resampled particle samples at time t, respectively. A random measure, $\{\mathbf{x}^{(k)}, w^{(k)}\}_{k=1}^{K}$, represents a probability in which

$$p(\mathbf{x}) \approx \sum_{k=1}^{K} w^{(k)}\delta(\mathbf{x} - \mathbf{x}^{(k)}). \tag{5}$$

After resampling, all the particles $\tilde{\mathbf{x}}_t^{(k)}$ have the same probability $\frac{1}{K}$, but there is a larger concentration of particles in regions of higher probability.

3.2 Synchronized Particle Filter

As shown in Alg. 1, the collaborative work of the multiple observation based particle filter is carried out by computing the joint likelihood weights $w_t^{(k)}$. Let us say the k^{th} weight of the i^{th} camera, $w_{i,t}^{(k)}$, represents the likelihood at the k^{th} support point, $\mathbf{x}_{i,t}^{(k)}$. The k^{th} support points of all cameras should be the same to obtain the joint probability by the multiplication of all camera weights. If any of the local likelihood weights have different support points, the multiplication of local weights does not guarantee the correct joint probability.

Algorithm 2 Synchronized particle filter at node i.

Given: A network with L sensors.

1: Do sampling with a common random seed.
2: Compute local likelihood weights:
$w_{i,t}^{(k)} = p(\mathbf{y}_{i,t}|\mathbf{x}_t^{(k)})$.
3: Transmit local random measure (weights):
$\{w_{i,t}^{(k)}\}_{k=1}^{K}$.
4: Receive other sensors' random measures (weights):
$\{w_{j,t}^{(k)}\}_{k=1}^{K}$, $j \neq i$.
5: Compute joint global weights: $w_t^{(k)} = \prod_{j=1}^{L} w_{j,t}^{(k)}$.
6: Compute posterior PDF.
7: Compute MMSE estimate.
8: Do resampling with the same random seed.

In the synchronized distributed particle filter (Farahmand et al. [2011]), all local particle filters are synchronized to have the same support points. To make all camera particles synchronized, all cameras are restricted to use a common random seed to initialize the random number generators responsible for the sampling and resampling processes, which is equivalent to operating all local cameras with a single set of particles. When all local particle filters are synchronized with the same random seed, multiple observation based particle filtering can be accomplished in a distributed way by communicating only local weights. Alg. 2 shows the general procedure of the synchronized particle filter at a given node.

3.3 Probability Conversion Based Particle Filter

When the synchronized particle filter is utilized in a camera network, the amount of data transmitted by a camera node is solely dependent upon the number of particles. For networks that have a tight communication bandwidth, it is indispensable that all nodes operate with a small number of particles to prevent tracking performance degradation. Also, when an event breaks the particle synchronization, e.g. a failure to share the seed due to communication loss, the multiple node collaboration cannot take place properly, which deteriorates the tracking performance. Hence, in many cases, it is preferable to encode the probability representation into a continuous form that is independent of the number of particles and does not require particle synchronization. Gaussian mixture model (GMM) (Ma and Ng [2006], Zuo et al. [2006], Gu [2007], Song et al. [2009], Gao et al. [2009], Huang et al. [2008], Sheng et al. [2005]) and Parzen (Ridley et al. [2004], Ong et al. [2005, 2006b, 2008]) based probability

conversion methods were proposed for converting particle probabilities into continuous forms. In this section, we show the derivation of a distributed particle filtering framework using the aforementioned probability conversion approaches, which will be utilized for the proposed tracking system.

For convenience of presentation, we first define some additional notation before describing the probability conversion based particle filters. Let us say $\Gamma(\mathbf{x}_t)$ is the prior PDF, $p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$, in the prediction step

$$\Gamma(\mathbf{x}_t) \triangleq p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1}). \tag{6}$$

Additionally, let us denote a local posterior PDF constructed from a local likelihood PDF at the i^{th} node (camera) as $\Delta_i(\mathbf{x}_t)$:

$$\Delta_i(\mathbf{x}_t) \triangleq \frac{1}{Z}p(\mathbf{y}_{i,t}|\mathbf{x}_t)\Gamma(\mathbf{x}_t), \tag{7}$$

where Z is the normalization constant. Then, the global posterior probability can be computed at camera i in an L camera network as

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \Delta_i(\mathbf{x}_t)\prod_{j\neq i}^{L} p(\mathbf{y}_{j,t}|\mathbf{x}_t). \tag{8}$$

For all $j \neq i$, the local likelihood $p(\mathbf{y}_{j,t}|\mathbf{x}_t)$ can be computed at node i if the posterior $\Delta_j(\mathbf{x}_t)$ and the prior PDF $\Gamma(\mathbf{x}_t)$ are known by node i, since

$$p(\mathbf{y}_{j,t}|\mathbf{x}_t) \propto \frac{\Delta_j(\mathbf{x}_t)}{\Gamma(\mathbf{x}_t)}. \tag{9}$$

By exchanging the local posterior PDFs, $\Delta_j(\mathbf{x}_t)$, each node can compute the likelihoods of the other cameras using Eq. (9), and then build the global posterior PDF using Eq. (8). This procedure effectively allows for object tracking to be performed in a distributed manner.

Assuming we have a global random measure $\{\tilde{\mathbf{x}}_{i,t-1}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$ of the previous global posterior PDF at the i^{th} camera, we can compute the prior PDF at the i^{th} camera by the following sampling procedure:

$$\mathbf{x}_{i,t}^{(k)} = f_t(\tilde{\mathbf{x}}_{i,t-1}^{(k)}) + \mathbf{u}_{i,t},\ k = 1, \cdots, K, \tag{10}$$

$$\Gamma_i(\mathbf{x}_t) \approx \frac{1}{K}\sum_{k=1}^{K}\delta(\mathbf{x}_t - \mathbf{x}_{i,t}^{(k)}), \tag{11}$$

where $\Gamma_i(\mathbf{x}_t)$ is the prior PDF at the i^{th} camera. Note that $\Gamma_i(\mathbf{x}_t)$ is equivalent to $\Gamma(\mathbf{x}_t)$ for all i, since all $\Gamma_i(\mathbf{x}_t)$ represent the same prior PDF with different support points (recall that in the probability conversion approaches we no longer require synchronized particles). The local likelihood weights at camera i are formed as

$$w_{i,t}^{(k)} = \frac{1}{Z}p(\mathbf{y}_{i,t}|\mathbf{x}_{i,t}^{(k)}), \tag{12}$$

where Z is the normalization constant. Then, the local posterior PDF, which is characterized by the random measure $\{\mathbf{x}_{i,t}^{(k)}, w_{i,t}^{(k)}\}_{k=1}^{K}$, is expressed as

$$\Delta_i(\mathbf{x}_t) \approx \frac{1}{K}\sum_{k=1}^{K} w_{i,t}^{(k)}\delta(\mathbf{x}_t - \mathbf{x}_{i,t}^{(k)}). \tag{13}$$

After computing the local posterior PDF, it is necessary to convert this discrete local posterior PDF form into a continuous representation. In the GMM representation, the local posterior PDF is converted to an equally weighted random measure by the resampling process; then GMM parameters (mixture component weights, means, and variances) are computed to form the GMM representation in which

$$\Delta_i(\mathbf{x}_t) \approx \frac{1}{Z_c}\sum_{n=1}^{N_{GMM}} c_{i,t}^{(n)}\mathcal{N}(\mathbf{x}_t - \boldsymbol{m}_{i,t}^{(n)}, \boldsymbol{\Sigma}_{i,t}^{(n)}), \tag{14}$$

where Z_c is the normalization constant, N_{GMM} is the number of GMM mixture components, $c_{i,t}^{(n)}$ is the n^{th} mixture component weight, and $\mathcal{N}(m, \Sigma)$ is a Gaussian PDF with mean m and variance Σ.

An alternative approach to represent the particles using a continuous distribution is through a Parzen window representation. For Parzen form conversion, we draw N_{PZ} samples $\{\bar{\mathbf{x}}^{(n)}\}_{n=1}^{N_{PZ}}$ from the local posterior PDF. Then, we have the random measure $\{\bar{\mathbf{x}}_{i,t}^{(n)}, \frac{1}{N_{PZ}}\}_{n=1}^{N_{PZ}}$ that is directly transformed to a continuous PDF representation. Note that the number of samples N_{PZ} does not need to be the same as the number of particles K to represent the local posterior PDF in continuous form. In fact, the continuous PDF representation of particle samples using Parzen windows requires significantly fewer samples than the corresponding discrete approximation, i.e., $N_{PZ} \ll K$. The continuous form of the local posterior PDF from the samples is expressed as

$$\Delta_i(\mathbf{x}_t) \approx \frac{1}{Z_k}\sum_{n=1}^{N_{PZ}} P_i(\mathbf{x}_t - \bar{\mathbf{x}}_{i,t}^{(n)}), \tag{15}$$

where Z_k is the normalization constant and $P_i(\cdot)$ is a Parzen's kernel.

After the conversion to continuous form of the local posterior PDF, each camera exchanges the Parzen's samples $\{\bar{\mathbf{x}}_{i,t}^{(n)}\}_{n=1}^{N_{PZ}}$ or the GMM parameters $\{c_{i,t}^{(n)}, \boldsymbol{m}_{i,t}^{(n)}, \boldsymbol{\Sigma}_{i,t}^{(n)}\}_{n=1}^{N_{GMM}}$. Then, the local likelihood weights of other nodes are computed at the support points of the i^{th} camera prior PDF as shown in Eq. (9), with $\Delta_j(\mathbf{x}_{i,t}^{(k)})$ given by Eqs. (14) or (15), i.e.:

$$w_{j,t}^{(k)} \triangleq \frac{\Delta_j(\mathbf{x}_{i,t}^{(k)})}{\Gamma(\mathbf{x}_{i,t}^{(k)})},\ \text{for } j \in \{1, \cdots L\}. \tag{16}$$

Then, the global likelihood weight can be computed as

$$w_t^{(k)} = \frac{\prod_{j=1}^{L} w_{j,t}^{(k)}}{\sum_{k=1}^{K}\prod_{j=1}^{L} w_{j,t}^{(k)}}. \tag{17}$$

The new random measure, $\{\mathbf{x}_{i,t}^{(k)}, w_t^{(k)}\}_{k=1}^{K}$, characterizes the global posterior PDF. Finally, by applying resampling, each camera obtains the equally weighted random measure $\{\tilde{\mathbf{x}}_{i,t}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$ for the next estimation. The detailed algorithm of the cluster-based probability conversion based particle filter is shown in the next section.

4 Resource-Aware Distributed Particle Filter

When developing a distributed tracking method, it is important to maximize the tracking system efficiency under given resource constraints such as communication bandwidth, processor's computational power, and energy consumption. Let us consider the constraints that affect the tracking algorithm design and application. While available communication bandwidth is dependent on network traffic, computational capacity for running a tracking algorithm is specified by hardware, which can be treated, to a great extent, as a static resource. Hence, the number of particle samples, which is the main factor influencing computational power consumption, can be set as a preassigned value according to the hardware specification. The computational resources required for carrying out a tracking algorithm typically do not change significantly over the course of tracking. Communication resources, on the other hand, are much more dynamic.

In this paper, we consider dynamically available communication resources and propose a resource-aware method, which reduces communication failures thereby improving the distributed tracking performance. Even though existing network protocols for wireless sensor networks, such as STCP (Iyer et al. [2005]), Fusion (Hull et al. [2004]), CODA (Wan et al. [2003]), and PCCP (Wang et al. [2006]), control traffic congestion and reduce the chance of communication failures by adjusting the data transmission rate, these methods do not affect the number of data packets generated by a sensor node. Instead, they control the transmission rate of queued data. The proposed resource-aware method computes the number of communication packets that are available for data transmission at a given time according to the network data traffic conditions and hence allows the particle filter to dynamically adjust the quality (resolution) of the tracking data to be communicated.

In this section, we introduce a novel distributed particle filter approach that is based on a resource-aware method for cluster-based WCNs. We utilize the dynamic clustering protocol in Medeiros et al. [2007] as the collaborative processing framework for our implementation. The clustering protocol describes a mechanism to form clusters for camera collaborations. Let us say a camera cluster is formed with the purpose of tracking an object so that a camera node in the cluster is elected as the cluster head and the other camera nodes are assigned as cluster members. As described in Medeiros et al. [2007, 2008a], we assume that cluster members are one hop neighbors of the cluster head but are not necessarily within single hop communication range of one another. In this environment, the cluster head estimates the global posterior probability of the target object by combining the local information transmitted from member cameras. Additionally, the cluster head broadcasts the joint posterior probability to its members, so that all the cluster members can perform the object tracking task with the same global information. An overview of the cluster-based tracking approach is shown in Fig. 5.

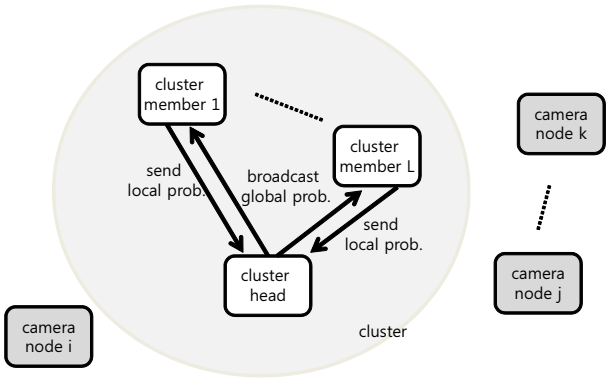


Figure 5: Data communication in cluster-based distributed particle filter.

4.1 Resource-Aware Packet Allotment

Under the assumption that tracking accuracy increases with the number of observations, tracking performance can be maximized when we utilize the information provided by all the members of a cluster. However, if we allow a large number of cluster members to transmit information at the maximum attainable data packet load, then the data traffic within the cluster increases, which may cause severe data loss, as mentioned in Section 2.

In this section, we present a resource-aware method that recognizes the data traffic conditions and computes the optimal amount of data that should be transmitted in a cluster. The optimal data packet load is obtained by maximizing the total amount of data transmitted in a cluster given the maximal allowed packet loss rate. This procedure also confines the energy waste level of data transmission to the user-defined missing rate when transmitting local data with the optimal data packet load.

Let us say that R is the total data packet load, i.e. the total number of packets containing particle weights, Parzen samples or GMM parameters in a frame, and M is the packet loss rate. Then, the plot in Fig. 1 (b) can be expressed as a function $M(R)$ shown in Fig. 6. We obtain the optimal data packet load, R_{opt}, by solving the following problem:

$$R_{opt} = max\{R(M)\} \quad s.t. \quad M < M_{max}, \tag{18}$$

where $R(M)$ is the inverse function of $M(R)$, and M_{max} is a user-defined packet loss rate. When $M(R)$ is a monotonically increasing function as shown in Fig. 6, $R(M)$ also becomes a monotonically increasing function. Then, the solution R_{opt} is in fact $R(M_{max})$. However, the rate function $R(M)$ is generally not available in practice but

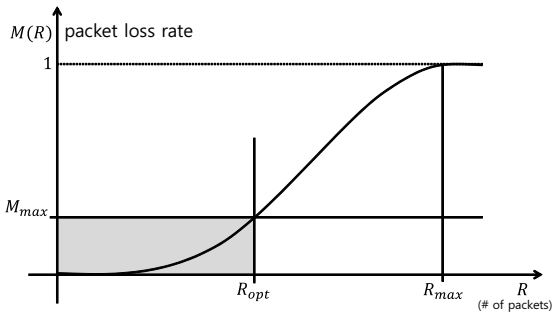


Figure 6: Optimal data packet load point in the packet loss rate function.

the packet loss rate can be measured at each time frame as

$$M(R_{TX}) = (R_{TX} - R_{RX})/R_{TX}, \tag{19}$$

where R_{TX} and R_{RX} are the number of transmitted and received packets, respectively. Thus, it is preferable to re-formulate the problem using the packet loss rate function.

Since $M(R_{opt}) = M_{max}$ as described in Fig. 6, we can re-formulate the optimization problem in Eq. (18) to find the optimal data packet load, R_{opt}, for M to be our target value, M_{max}. Using a convex cost function such as a squared difference function, the optimal data packet load problem can be expressed as

$$\begin{aligned} R_{opt} &= \arg\min \left\{ (M(R) - M_{max})^2 \right\} \\ s.t. \quad & 0 < R < R_{max}, \end{aligned} \tag{20}$$

where R_{max} is the data packet load for which all the data is missed (i.e., M_{max} becomes 1). We assume that as R increases, the packet loss rate monotonically increases as in (Zhao and Govindan [2003]):

$$M(R_{opt} - \delta) < M(R_{opt}) < M(R_{opt} + \delta), \tag{21}$$

where δ is a small number. Then, the cost function $(M(R) - M_{max})^2$ becomes a unimodal function, which has a single optimum. Hence, the minimizer R_{opt} can be obtained by applying a gradient descent method (Chong and Zak [2008]) and the iterative solution is

$$R_{opt}^{(t+1)} = R_{opt}^{(t)} - \mu_t \nabla_t, \tag{22}$$

where $\nabla_t = M'(R_{opt}^{(t)})(M(R_{opt}^{(t)}) - M_{max})$, μ_t is the step size of the iteration, and $\mu_t > 0$. Since we cannot access the derivative of the packet loss rate function, $M(\cdot)$, we assign μ_t as $\beta_t / M'(R_{opt}^{(t)})$, where $\beta_t > 0$. Note that $M'(R_{opt}^{(t)})$ is positive since the packet loss rate monotonically increases in the operational range such that $0 < R < R_{max}$. Then the iterative equation becomes

$$R_{opt}^{(t+1)} = R_{opt}^{(t)} - \beta_t (M(R_{opt}^{(t)}) - M_{max}). \tag{23}$$

Algorithm 3 Packet allotment procedure.

Given: the target packet loss rate, M_{max}, the minimum data packet load, R_{min}, and the data packet load at t, $R_{opt}^{(t)}$.

1: Count received data packets: $R_{RX}^{(t)}$.
2: Compute the packet loss rate:
$$M(R_{opt}^{(t)}) = (R_{opt}^{(t)} - R_{RX}^{(t)})/R_{opt}^{(t)}.$$
3: Update data packet load:
$$R_{opt}^{(t+1)} = R_{opt}^{(t)} + \beta(M(R_{opt}^{(t)}) - M_{max}).$$
4: Compute individual node data packet load:
$$R_l^{(t+1)} = max \left\{ \left\lfloor R_{opt}^{(t+1)}/L \right\rfloor, R_{min} \right\}.$$

When we assign β_t as a constant, we have the following condition for convergence (Chong and Zak [2008])

$$\beta < M'(R). \tag{24}$$

This condition indicates that β should be a small number if the slope of $M(R_{max})$ is small, and that there is a limit in the rate of change of $M(R)$ that the system is capable of handling.

Let us say the solution at the current frame, R_{opt}, is the available communication resource in a cluster. It is necessary to assign a proper data packet load for each node, R_l, where $\sum_{l=1}^{L} R_l = R_{opt}$. One may utilize a rate-distortion optimization approach (Cover and Thomas [2006]), which assigns each node communication rate according to the difference between the transmitted and received probabilities of each node, caused by packet losses. However, in hardware and bandwidth constrained smart camera applications, it may not be feasible to run the rate-distortion optimization process, since it requires both monitoring the data packet loads of all nodes and measuring their probability differences. Assuming the communication channels and packet loss rates of all camera nodes are approximately the same, we use a fair packet allotment approach, which is

$$R_l = max \left\{ \left\lfloor \frac{R_{opt}}{L} \right\rfloor, R_{min} \right\}, \tag{25}$$

whereR_{min} is a minimum packet load for data transmission, which prevents the rate from becoming zero. Note that the floor operator,$\lfloor \cdot \rfloor$, is used to ensure that the resulting packet load is an integer value. Alg. 3 describes the procedure of packet loss measurement and packet allotment.

4.2 Cluster-Based Framework for Particle Filter Tracking

In this section, we describe the proposed cluster-based distributed particle filter implementations. Alg. 4 and Alg. 5 illustrate the particle filtering procedures for the cluster members and the cluster head, respectively. For the sake of convenience, we divide the filtering procedure of a node into three sub phases: *pre-processing*, *data communication*, and *post-processing*.

Algorithm 4 Cluster-based distributed particle filter at the i^{th} cluster member.

Given: the previous posterior PDF random measure, $\left\{\tilde{\mathbf{x}}_{i,t-1}^{(k)}, \frac{1}{K}\right\}_{k=1}^{K}$ and data packet load, $R_l^{(t-1)}$.

Pre-Processing

1: Do sampling: $\mathbf{x}_{i,t}^{(k)} = f_t(\tilde{\mathbf{x}}_{i,t-1}^{(k)}) + \mathbf{u}_{i,t}$.
2: Compute local weights: $w_{i,t}^{(k)}$.

Data Communication

1: Convert local probability into continuous form according to the data packet load $R_l^{(t-1)}$.
2: Transmit local probability to cluster head.
3: Receive global probability and the next data packet load $R_l^{(t)}$ from cluster head.
4: Reconstruct global posterior PDF (update weights): $w_t^{(k)}$.

Post-Processing

1: Do resampling: $\left\{\tilde{\mathbf{x}}_{i,t}^{(k)}, \frac{1}{K}\right\}_{m=1}^{K} \Leftarrow \{\mathbf{x}_{i,t}^{(k)}, w_t^{(k)}\}_{k=1}^{K}$.

Algorithm 5 Cluster-based distributed particle filter at the cluster head.

Given: the previous posterior PDF random measure, $\left\{\tilde{\mathbf{x}}_{i,t-1}^{(k)}, \frac{1}{K}\right\}_{k=1}^{K}$

Pre-Processing

1: Do sampling: $\mathbf{x}_{i,t}^{(k)} = f_t(\tilde{\mathbf{x}}_{i,t-1}^{(k)}) + \mathbf{u}_{i,t}$.
2: Compute local weights: $w_{i,t}^{(k)}$.

Data Communication

1: Receive local probabilities from cluster members.
2: Reconstruct local likelihood weights: $w_{j,t}^{(k)}$, $j \neq i$.
3: Compute joint global weights: $w_t^{(k)} = w_{i,t}^{(k)} \prod_{j \neq i}^{L} w_{j,t}^{(k)}$
4: Compute data packet load for next frame according to Alg. 3.
5: Transmit global probability and the data packet load to cluster members.

Post-Processing

1: Do resampling: $\left\{\tilde{\mathbf{x}}_{i,t}^{(k)}, \frac{1}{K}\right\}_{k=1}^{K} \Leftarrow \{\mathbf{x}_{i,t}^{(k)}, w_t^{(k)}\}_{k=1}^{K}$.
2: Compute MMSE estimate: $E(\mathbf{X}_t|\mathbf{y}_{0:t})$.

In pre-processing and post processing, the nodes perform common particle filtering operations consisting of sampling, computing object likelihoods, resampling, and estimating the target position. At the data communication phase, the cluster head and cluster members carry out different operations. First, cluster members prepare for data communication by computing the amount of data load according to Eq. (25) and, when applicable, converting their probabilities into continuous forms (e.g., Parzen samples or GMM parameters) according to the data packet load; the number of Parzen samples or GMM parameters to be transmitted is proportional to the packet rate. Then the cluster head receives the local data from the cluster members and builds a joint probability. The cluster head also updates the total packet rate of the cluster by inspecting the number of missed packets at the current frame, as described in Alg. 3. Then, the cluster head broadcasts the global probability to the cluster members, which execute their particle filters with the received global information. The timing diagram of the cluster-based distributed particle filter, as shown in Fig.7, describes one cycle of processing, including image capturing and data processing.

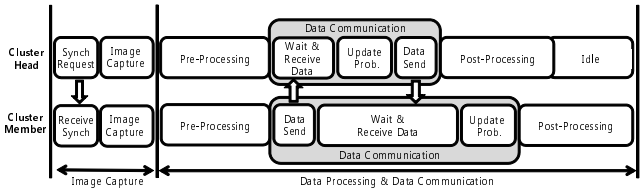


Figure 7: Distributed particle filter timing diagram.

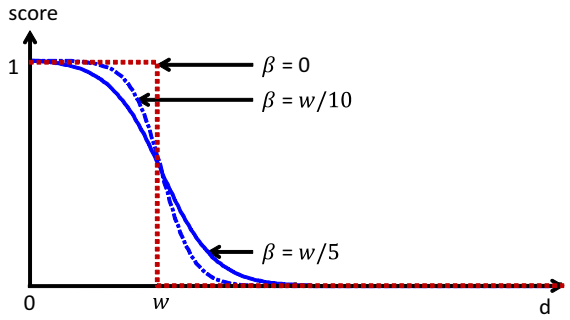


Figure 8: Score function for tracking result evaluation.

5 Experiments

In this section, we present experiments that demonstrate the feasibility of the proposed method. We first describe the evaluation metrics we utilize for analyzing the proposed approaches. Then, the detailed experimental setting, including parameters of the particle filters, communication payloads, and packet loss models are introduced so that we can finally show the experimental results.

5.1 Evaluation Metrics

The proposed distributed tracking systems have two main functionalities: One is object tracking based on particle filtering, and the other is packet allotment, which is independent of the tracking mechanism. To evaluate the performance of the object tracking methods, two metrics are utilized, *tracking error* and *score*. In order to show the performance of the resource-aware packet allotment mechanism, the *delivery energy efficiency* metric is introduced. These metrics are described in detail in the following sections.

5.1.1 Tracking Evaluation Metrics

To evaluate the performance of a visual tracking system, a human visually verifies the tracking results by manually creating bounding circles or boxes surrounding the objects being tracked at each image frame. When such ground-truth data is available, the tracking performance can be measured by tracking errors, which are computed utilizing a root squared difference function between the ground-truth data $\mathbf{x}_{gt}$ and the tracking results $\mathbf{x}_{est}$:

$$error \triangleq \|\mathbf{x}_{est} - \mathbf{x}_{gt}\|_2 \tag{26}$$

Eq. (27) shows the average tracking error (ATE) measurement that we will utilize to evaluate tracking performance.

$$ATE = \frac{\sum_{t=1}^{T} error_t}{T}, \tag{27}$$

where t is the frame number and T is the total number of frames. However, in this paper, we are interested in measuring the accuracy and persistence of target tracking in a camera network. After a tracker loses track of the target object, the algorithm may present erratic behavior. Hence, it is difficult to show the tracking performance exclusively based on tracking errors, since the errors could indicate largely different values depending upon where the estimated tracks are lost. Here, we use an additional measurement to evaluate tracking performance. We measure to what extent a tracker tracks the target object successfully. Successful tracking is measured with a scoring function, which is a thresholding or decreasing function of distance between the target track and the ground-truth at the t^{th} frame. The average success score is defined in Eq. (28).

$$ATS = \frac{\sum_{t=1}^{T} score_t}{T}, \tag{28}$$

where t is the frame number, T is the total number of frames, and $score_t$ is a thresholding or decreasing function of the distance between the target track and the ground-truth at the frame t. Fig. 8 shows how the $score_t$ is assigned according to the Euclidean distance. When we use a thresholding score shown as the red dotted line, it assigns one if a tracking result is within a certain range; otherwise it assigns zero to indicate tracking failure. When a decreasing function (blue lines) is utilized as a tracking score, it penalizes tracking results according to their accuracy. Hence, the tracking score can also show the accuracy and success of target tracking.

To evaluate our tests, we set the distance threshold as the target object width, w, and the scoring function as a sigmoid type function as in Eq. (29).

$$score \triangleq \frac{1 + \exp(-w/\beta)}{1 + \exp((d-w)/\beta)}, \tag{29}$$

where β is a control parameter for the transition width of the scoring function and d is the Euclidean distance between the estimated target position and the ground-truth at a given frame. In our experiments, we set β as $w/5$, as depicted in Fig. 8. As we can see, the thresholding approach is just a special case when $\beta = 0$.

	1 packet (12 bytes)
Synch	6 particle weights
GMM	1 set of GMM parameters
Parzen	3 Parzen samples

Table 1 Data payload in a packet.

5.1.2 Delivery Energy Efficiency

In lossy networks, transmitted data packets are frequently missed. The non-received packets also consume transmission energy, which is essentially wasted. If a tracker shows good tracking results with large packet loss (typically under the situation that it transmits a very large number of redundant packets), the tracking system requires high cost and may not be suitable for wireless systems. We measure the energy efficiency of delivered packets using *delivery energy efficiency* (DEE), which is defined as the non-wasted portion of the total transmission energy. Let us say e is the energy required to transmit a packet. Then, the total transmission energy E_{TX} is expressed as

$$E_{TX} = N_{TX} \times e, \tag{30}$$

where N_{TX} is the number of transmitted packets. Similarly, the amount of wasted energy E_{NR} is computed as

$$E_{NR} = N_{NR} \times e, \tag{31}$$

where N_{NR} is the number of lost packets. Then, DEE has the following form:

$$DEE \triangleq \frac{E_{TX} - E_{NR}}{E_{TX}} = \frac{N_{TX} - N_{NR}}{N_{TX}}. \tag{32}$$

We will utilize DEE as a measurement of energy efficiency for tracking data communication.

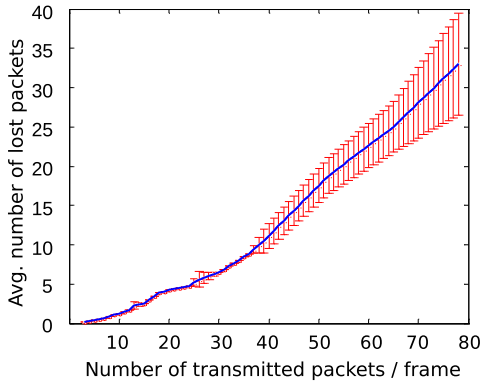
5.2 Simulations and Experiments in a Wireless Camera Network

In this section, simulated results using previously recorded image sequences are carried out to analyze the tracking performance of the proposed resource-aware method. Then, real-time experiments are shown on a WCN implementation.

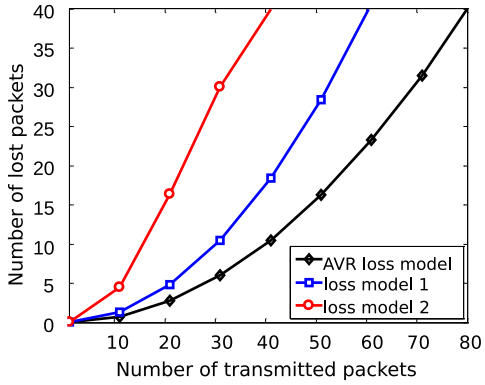
5.2.1 Particle Filter Settings

In our experimental setup, we set the target state, $\mathbf{x}_t$, at time t as the object position (x_w, y_w) in the world coordinate plane:

$$\mathbf{x}_t = [x_w, y_w]^T. \tag{33}$$



(a)



(b)

Figure 9: Average packet loss under AVR simulator (a) and packet loss models (b). (a) Red bars indicate standard deviations of the packet loss. (b) The black line model is obtained from the AVR simulation and the red and blue lines are set for a moderate and a severe loss case for the resource-aware tracking test, respectively. In the packet loss model 2, as the number of transmitted packets increases, most transmitted packets are lost.

For the state process model, we utilize a linear transition matrix corrupted by Gaussian noise:

$$\mathbf{x}_{t+1} = \boldsymbol{F}\mathbf{x}_t + \mathbf{u}_t, \tag{34}$$

where we set the identity matrix as the transition matrix, $\boldsymbol{F} = \boldsymbol{I}_{2\times 2}$, since, for simplicity, we do not accommodate object velocity in the state vector. The transition noise, $\mathbf{u}_t$, is set as a Gaussian noise model. Also, we consider $\mathbf{y}_t$ as the target object (or reference) feature at time t and $h_t(\mathbf{x}_t)$ as the extracted feature at a support point $\mathbf{x}_t$ in the image at time t. In our implementation, we use color histogram features as described in (Nummiaro et al. [2002], Perez et al. [2002], Medeiros et al. [2008b, 2010]). We compute the object histogram at the object position on the image coordinate, which is transformed from the object position in the world coordinate frame by the precomputed homography obtained from the camera calibration information. The likelihood probability is set to

$$p(\mathbf{y}_{i,t}|\mathbf{x}_t) = \frac{1}{Z}\exp\left(-\frac{d\big(h_{i,t}(H_i\mathbf{x}_t), \mathbf{y}_{i,t}\big)}{\lambda}\right), \tag{35}$$

where Z is the normalization constant, H_i is the homography between the world coordinate ground plane and the i^{th} camera image plane, λ is the observation noise power, and $d(x, y)$ is the Euclidean distance function between x and y. Note that the distance can be computed instead with the Bhattacharyya distance utilized in (Nummiaro et al. [2002], Perez et al. [2002]) as shown in (Hager et al. [2004]) or with any other suitable distance metric. The color histograms are computed in the RGB color space with 38 bins in each dimension. The noise level (power) was set as a fixed constant, $\lambda = 0.06$.

5.2.2 Communication Packet Payloads and Packet Loss Models

In our implementation, we describe a floating point variable such as a particle weight with 2-byte precision. As for the size of the communication packets, we have chosen to allow for 12 bytes of payload so that the particle information as well as additional 17 bytes of headers required by lower level protocols (such as the clustering protocol) can be transmitted within a single TinyOS packet, which has a default length of 29 bytes. For the synchronized particle filter implementation, 6 particle weights can be transmitted in a single packet, as a particle weight requires 2 bytes to be described. In the Parzen particle filter implementation, 4 bytes are needed to describe a Parzen sample composed of x and y positions, hence 3 Parzen samples are transmitted in a packet. A set of GMM mixture parameters consists of 6 floating point variables, which are a GMM component weight, x and y positional means, and the variances and covariance between x and y. This set of GMM parameters requires 12 bytes, and hence in the GMM implementation, a single set of GMM parameters is loaded in a packet. Table 1 shows the summary of the different data payloads in a packet.

We set the number of particles $K = 300$ for the probability conversion based particle filter methods. However, for the synchronized particle filter, the number of particles is determined by the number of packets available to a local node. For example, when 10 packets are assigned to a node for particle data communication, the number of particles in the synchronized particle filter is 60 (10 multiplied by 6).

To obtain a packet loss model, the clustering (Medeiros et al. [2007]) and the Parzen based tracking algorithms with a sensing rate of 0.5 seconds were executed under the Avrora (AVR) simulator (Titzer et al. [2005]) with the CSMA (carrier sense multiple access with collision avoidance) protocol (IEEE [1998]). We varied the numbers of packets per frame assigned to each node as well as the number of cluster members in each cluster (2 to 7 cluster members, including the cluster head) so that

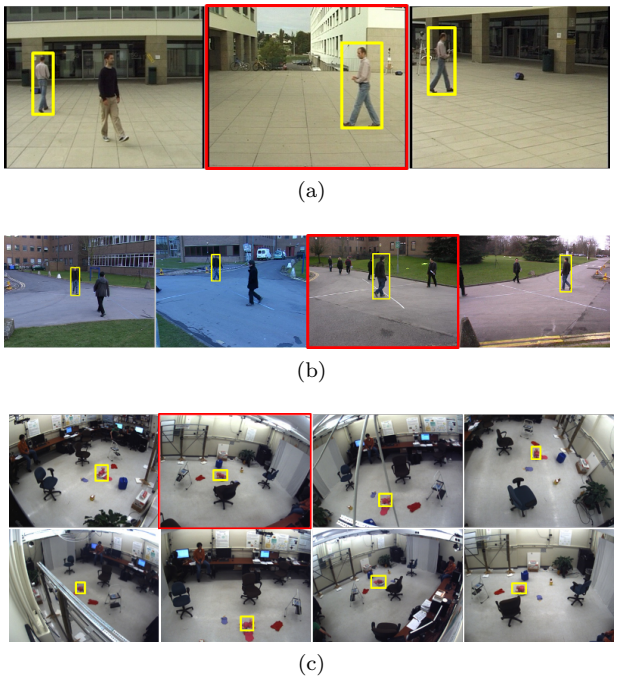


Figure 10: Test image sequences. (a) Campus, (b) PETS and (c) Lab sequences. The red image box indicates the image captured at the cluster head; the small yellow boxes in all images show the target object.

the total number of packets transmitted per frame varied between approximately 0 and 80. Fig. 9 (a) shows the average packet loss of the simulations, which were repeated 30 times on each combination of number of packets per frame and cluster members. Based on the simulation results, we created two packet loss models (shown in Fig. 9 (b)) to investigate the performance of the proposed resource-aware approach. Note that the two models we utilized induce more packet losses than the model obtained from the AVR simulation. The purpose of the additional packet losses is to account for potential sources of communication failures that may not have been included in the simulations. These models are supposed to represent severe and moderate communication failure scenarios in our experimental evaluation in the following sections.

5.2.3 Simulated Experiments

We tested three multiple image sequence sets in our simulations: Campus (3 image sequences) (Ecole Polytechnique Federale de Lausanne [2008]), PETS (4 image sequences) (Reading University [2009]), and Lab (8 image sequences) image sets. Examples of the publicly available Campus and PETS image sequences, which are captured in outdoor environments, are shown in Fig. 10 (a) and (b), respectively. Fig. 10 (c) shows examples of the Lab image sequences, which were captured at the Purdue Robot Vision Lab, an indoor environment. In each image sequence set, the red boxes indicate the images captured by the cluster heads. The target objects in the image sequences are marked by a small yellow box. We select the test image sequences in which all cameras have common object views so that all the cameras take part in the tracking process, one as the cluster head and the others as cluster members. We assume that all cameras are located in single hop communication range to the cluster head. All of the tests were done under the same transition dynamic model and observation noise level. The experiments were carried out using two different target packet loss rates, $M = 0.1$ and $M = 0.2$, for the resource-aware distributed particle filters under the two loss models described in Fig. 9 (b). For the non resource-aware distributed particle filters, 4 to 10 packet loading scenarios were tested under the same two loss models. In these experiments, ATSs and ATEs are computed by considering the tracking results of a centralized particle filter as the ground-truth.

We first looked at the performance of the standard distributed particle filters, in which the resource-aware method is not employed. Fig. 11 shows the average tracking scores and the average tracking errors for the Campus, PETS, and Lab sequences under lossless packet communication. All three distributed particle filters, the synchronized, the GMM, and the Parzen approaches, show reliable tracking performances. As the figure shows, all ATSs are close to 1 and the ATEs are within few centimeters, since the probability representations are not subject to any degradation caused by dropped packets. As the number of assigned transmission packets increases, the tracking performance does improve but not noticeably (ATSs increase and ATEs decrease) as shown in Fig. 11. Note that the three different approximations of the object probability distribution cause a marginal difference in error values.

However, when packet losses occur, the tracking performances of the distributed particle filters are degraded. Fig. 12 and 13 show the average tracking scores and the average tracking errors for the Campus, PETS, and Lab sequences under loss model 1 and 2. Fig. 14 and 15 depict the corresponding delivery energy efficiencies and transmitted and received packets. As the number of transmitted packets in the network increases (from 4 to 10 packets), the number of lost packets also increases as shown in Fig. 15. The synchronized particle filter shows that it is extremely vulnerable to packet losses; the tracking performance degrades rapidly as the number of lost packets increases (see Figs. 12 and 13). In the synchronized particle filter, the collaborative processing, which consists of computing the joint probability, requires particle synchronization. When data losses occur, the particle weights in the lost support points become unavailable, which leads to a distorted joint probability and incorrect object track estimation. On the other hand, the GMM and Parzen particle filters show robust performances to lossy data communication. In the Campus and PETS sequence experiments, which utilized 3 and 4 cluster members, the GMM and Parzen tracking algorithms present around 90% tracking success rate (0.9 ATS) under the packet loss model 1, in which the packet loss rates are less than 30%. However, increasing the packet loss rate,

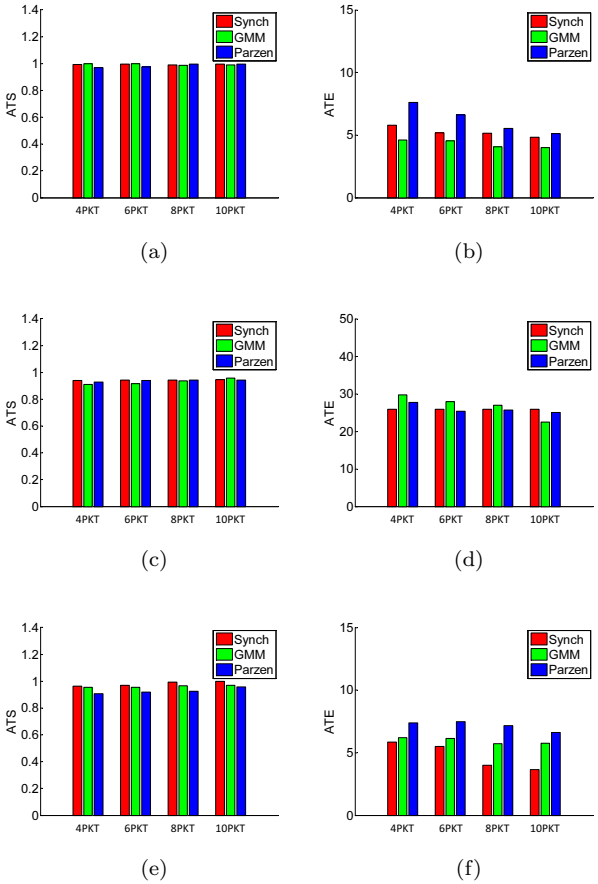


Figure 11: Average tracking scores (ATSs) and average tracking errors (ATEs) under lossless packet communication. (a), (c), and (e): ATS for Campus, PETS, and Lab sequences. (b), (d), and (f): ATE for Campus, PETS, and Lab sequences.

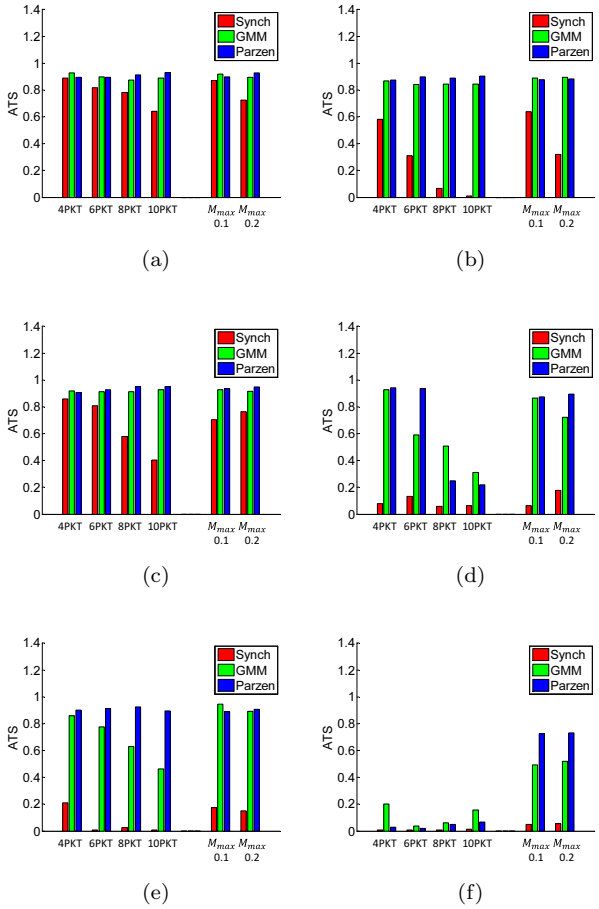


Figure 12: Average tracking scores (ATSs) for the three test sequences. From top to bottom, each row shows ATSs for Campus, PETS, and Lab sequences, respectively. From left to right, each column depicts ATSs under loss model 1 and 2, respectively.

which happens in the Lab sequence experiments with the packet loss models 1 and 2 and the PETS sequence experiments with the packet loss model 2, the GMM particle filter shows degraded performances as shown in Fig. 12 (d), (e), and (f), as data traffic increases as shown in Fig. 15 (d), (e), and (f). The Parzen particle filter shows additional robustness with respect to the GMM approach in at least one of these three severe scenarios (plot (e)).

When the resource-aware method is applied to the distributed particle filters, it confines the packet loss rate to a pre-assigned level M_{max}. For the synchronized particle filter, the resource-aware method does not improve the tracking performance much, as it does not guarantee lossless packet transmission but rather a certain rate of packet loss, and the synchronized particle filter cannot operate properly even under modest communication failures. However, the resource-aware method drastically improves the tracking performances of the GMM and the Parzen distributed particle filters as shown in Fig. 12 and 13, and increases the packet delivery energy efficiencies as shown in Fig. 14. For the most severe packet loss case, shown in Fig. 15 (f), the resource-aware method still preserves the requested packet loss rate and improves the tracking performances of the GMM particle filter from a mean ATS of 0.11 to an ATS of 0.52 and that of the Parzen particle filter from a mean ATS of 0.04 to an ATS of 0.74, as shown in Fig. 12 (f) (the mean ATS of the non-resource aware particle filters is computed by averaging the ATSs for 4 to 10 packet transmissions).

Selected object trajectories under loss model 1 and 2 are shown in Fig. 16 and 17, respectively. Note that in the trajectory plots, the world plane is represented in the x and y axes, and the frame number is indicated in the z axis. As previously shown, the synchronized particle filter shows high tracking errors even in the presence of modest communication failures. Therefore, the object trajectories estimated by the synchronized particle filter are far deviated from the ground-truth, and hence, to better visualize the performance of the resource-aware method, we chose to plot only the trajectories of the GMM and the Parzen particle filters in the trajectory plots. Fig. 16 and 17 show that the estimated trajectories become significantly more sta-

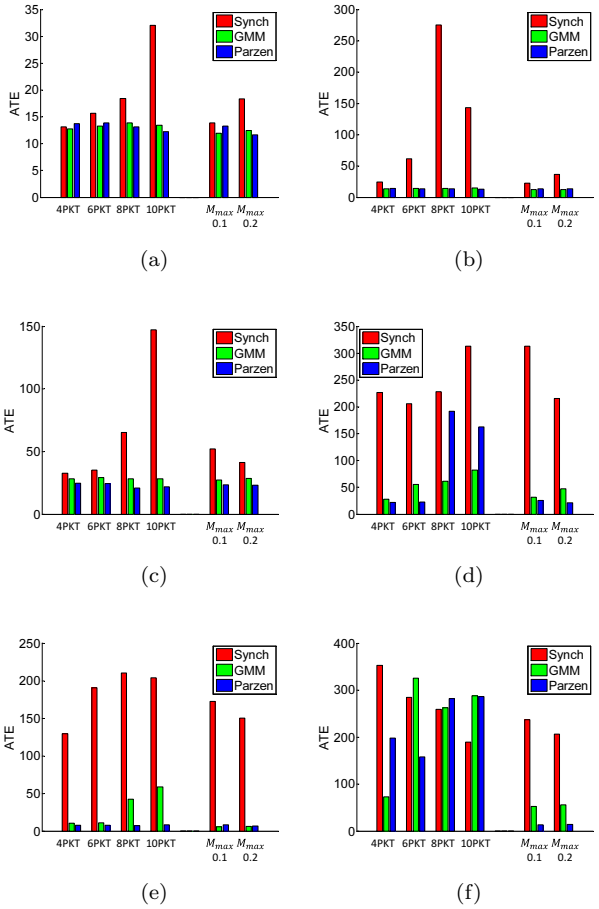


Figure 13: Average tracking errors (ATEs) for the three test sequences. From top to bottom, each row shows ATEs for Campus, PETS, and Lab sequences, respectively. From left to right, each column depicts ATEs under loss model 1 and 2, respectively.

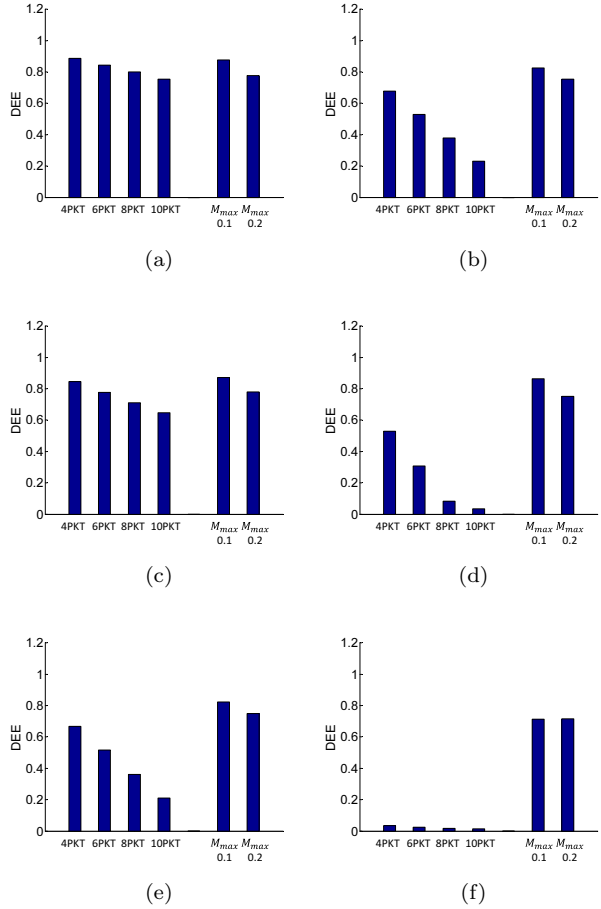


Figure 14: Delivery energy efficiencies (DEEs) for the three test sequences. From top to bottom, each row shows DEEs for Campus, PETS, and Lab sequences, respectively. From left to right, each column depicts DEEs under loss model 1 and 2, respectively.

ble when the resource-aware method is applied to the GMM and the Parzen particle filters, confirming that the resource-aware method improves the object tracking performance, as previously shown in Fig. 12 (d) and (f). For the most severe packet loss case (the Lab sequence test with the loss model 2), since almost no data is received, all the distributed particle filters lose track of the object, as shown in Fig. 17 (e). The resource-aware method reduces the number of packets to be transmitted in this case and it allows a certain portion of packets to be communicated (as shown in Fig. 15 (f)), thereby restoring to a great extent the functionality of the GMM particle filter and allowing the Parzen particle filter to operate almost perfectly, as shown in Fig. 17 (f). Snapshots of the tracking results are shown in Fig. 18.

To show a more concise and integrative view of the distributed particle filter performances, we introduce a new metric that combines the ATS and DEE metrics. Fig. 19 shows the combined measure, which is computed by multiplying the ATS and DEE. The resource-aware methods show improved performances for the GMM and Parzen particle filters when packet loss is higher as shown in Fig. 19. Although the resource-aware methods tend to present slightly better performance when $M_{max} = 0.1$, when packet loss is severe as in the case of the particle filters with 8 cluster members running under the loss model 2, the performances of the two resource-aware methods ($M_{max} = 0.1$ and $M_{max} = 0.2$) does not show noticeable difference as shown in Fig. 19 (f), since the best packet loss rate that can be achieved by either resource-aware method is limited to the minimal packet loss condition. In this experiment, the achieved packet loss rates of both resource-aware methods using $M_{max} = 0.1$ and $M_{max} = 0.2$ were approximately 0.28 packet loss rate (i.e., $M = 0.28$). Note that in the other experiments the resource-aware method for the GMM and Parzen particle filters with 0.2 maximum loss rate ($M_{max} = 0.2$) show lower values than with 0.1 maximum loss rate ($M_{max} = 0.1$) in the combined measure because although their ATSs are comparable there is a difference of 0.1 between the two maximum loss rates.

To understand how the performances of the particle filters are affected by the number of nodes in the network and by the amount of traffic generated by each

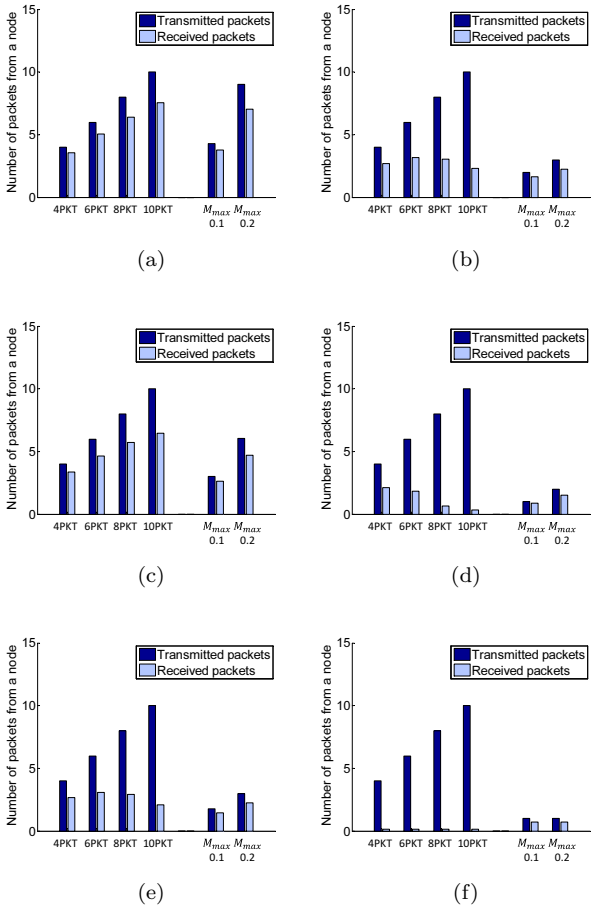


Figure 15: Transmitted (TX) and received (RX) packets for the three test sequences. From top to bottom, each row shows RX and TX packets for Campus, PETS, and Lab sequences, respectively. From left to right, each column depicts RX and TX packets under loss model 1 and 2, respectively.

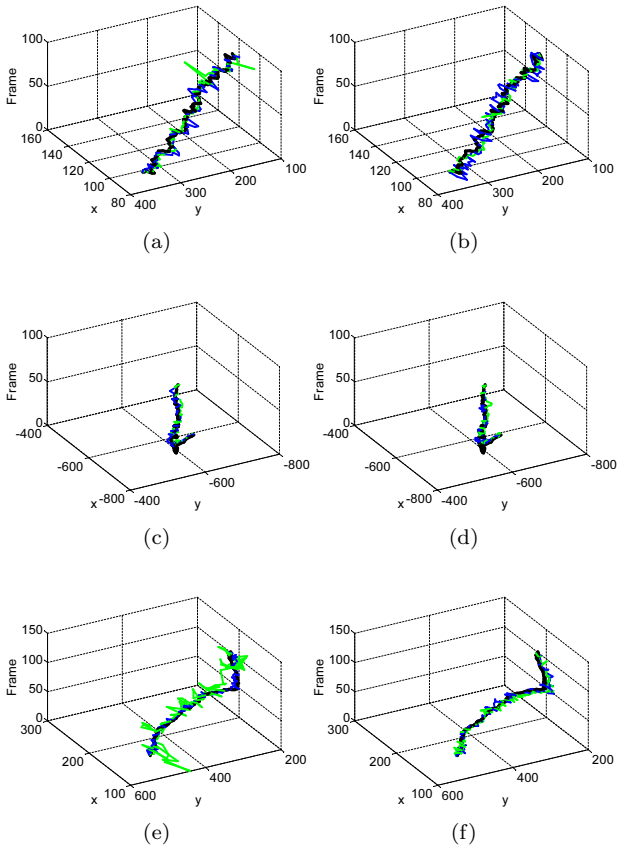


Figure 16: Trajectories of estimated object positions under the loss model 1 (unit: cm). (a), (c), and (e): Trajectories of non resource-aware methods when 8 packets per node were transmitted for the Campus, PETS, and Lab sequences. (b), (d) and (f): Trajectories of resource-aware methods when $M_{max} = 0.1$ for Campus, PETS, and Lab sequences. The blue, green, and black lines indicate the object trajectories estimated by Parzen, GMM, and centralized particle filters, respectively.

node, we consider scenarios in which clusters consisted of 3 to 8 members and the number of packets generated by each cluster member varied from 4 to 10. For these experiments, we used only the Lab sequences as they correspond to the network with the largest number of cameras and the greatest amount of overlap between camera views. Fig. 20 shows the delivery energy efficiencies and received packets. Note that packet losses and the packet control mechanism of the resource-aware method are independent of the particle filters. Hence, we show common DEEs and RX packets for all the distributed particle filters in Fig. 20. The resource-aware method preserves the DEEs of the distributed particle filters by monitoring the rate of lost packets and adjusting the number of data packets, whereas non-resource-aware particle filters have degraded DEEs as they do not have any adaptive mechanism for lossy communication environments.

Fig. 21 and 22 depict the tracking performances with ATSs and ATEs. As the figures indicate, the resource-aware methods perform significantly better than the non-resource aware particle filters in the majority of the scenarios, especially when the number of cluster members is higher (and consequently the network traffic is heavier). Again, the synchronized particle filter cannot benefit much from the resource-aware method. The experiments shown in Fig. 21 (a) and (b) demonstrate that the tracking performance is dependent on the number of packet losses as shown in Fig. 20 (a) and (c). The resource-aware method does not help the performance much, since it keeps a certain level of packet loss rate as shown in Fig. 20.

The GMM components are weighted as described in Section 3.3. Therefore, data packets of the distributed GMM particle filter are not equally important, since a packet consists of a mixture weight. When highly weighted packets are lost during communication, the converted probability from the received packets represents the probability estimated by that node poorly, which causes overall unreliable tracking. The experimental results in Fig. 21 and 22 show a tendency that higher packet loss rates caused by increasing the num-

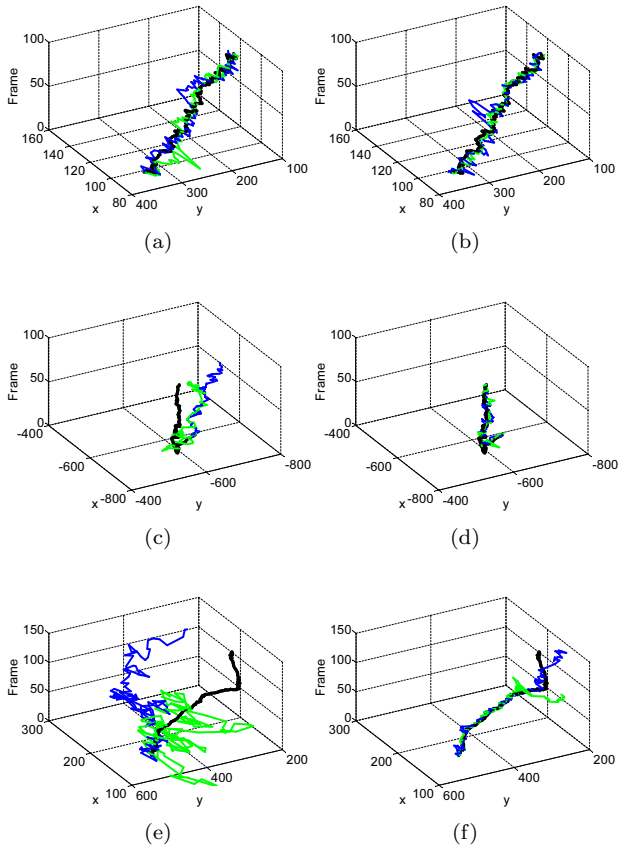


Figure 17: Trajectories of estimated object positions under the loss model 2 (unit: cm). (a), (c), and (e): Trajectories of non resource-aware methods when 8 packets per node were transmitted for the Campus, PETS, and Lab sequences. (b), (d) and (f): Trajectories of resource-aware methods when $M_{max} = 0.1$ for Campus, PETS, and Lab sequences. The blue, green, and black lines indicate the object trajectories estimated by Parzen, GMM, and centralized particle filters, respectively.

ber of cluster members and packet transmissions degrade the tracking performance of the GMM particle filter. This tendency is explained by the fact that higher loss rates increase the odds of dropping important packets. The resource-aware method confines the packet loss rate, which reduces the chance of missing highly weighted packets. Hence, the performance of the GMM particle filter is improved with the resource-aware method under lossy communication environments. The GMM particle filter shows better tracking performance with a smaller maximum missing rate ($M_{max} = 0.1$ in the experiments).

In the Parzen distributed particle filter, Parzen samples are packed into transmission packets. Communication failures cause packets to be dropped at random, which is essentially equivalent to reducing the resolution of Parzen sampling since Parzen samples are generated by a random selection from the equally weighted particles. Furthermore, Parzen samples are distributed densely at the support points that have high particle weights. When the resolution of the Parzen representation is not at a critical level (i.e., the number of received Parzen samples is not too small) to represent an object probability, the Parzen particle filter can provide reliable tracking performance. Hence, the Parzen particle filter is less vulnerable to packet losses than the synchronized and the GMM particle filters, as our experiments demonstrate. In terms of tracking accuracy, the resource-aware method does not improve the performance of the Parzen particle filter much as shown in Fig. 21 (e) if packet loss rate is not critical as shown in Fig. 20 (a). Note that the resource-aware method enables a small number of packet communications even in an extremely lossy environment as shown in Fig. 20 (c), and the Parzen particle filter takes advantage of the resource-aware method in this case, as Fig. 21 (f) shows, especially when the number of cluster members is more than 6. In general, even for lower packet loss rates, the resource-aware method significantly increases the DEE without degrading the tracking performance of the Parzen particle filter, as shown in Fig. 20 (a) and (c).

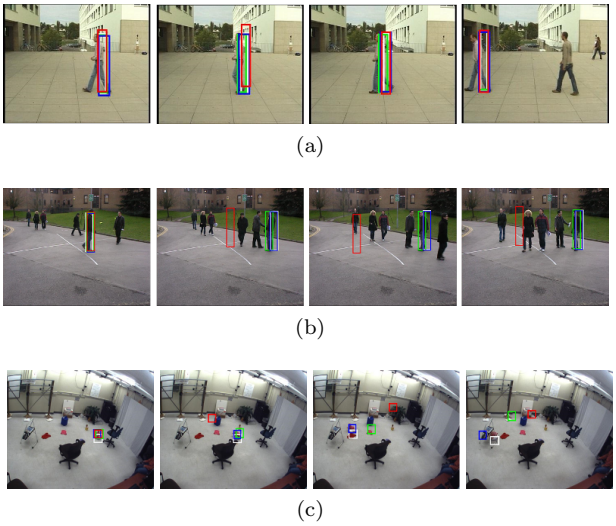


Figure 18: Tracking results of (a) Campus, (b) PETS, and (c) Lab sequences using resource-aware method with $M_{max} = 0.1$ under the loss model 2. The red, green, blue, and white boxes indicate the object tracks estimated by the synchronized, GMM, Parzen, and centralized particle filters, respectively. The tracking results were captured at the cluster head.

5.2.4 Experiments on a Wireless Camera Network Testbed

The proposed resource-aware method was evaluated on an Imote2-based real wireless camera network testbed as shown in Fig. 23. The Robot Vision Lab (RVL) testbed consists of 13 Imote2 nodes, shown in Fig.24 (a), and covers a doorway across three consecutive rooms where each room size is roughly $20ft \times 20ft$. The sensing rate of the camera nodes was set to capture a 320 by 240 image at every 1.6 second; the extremely low sensing rate is due to hardware limitations. The target object (an iRobot Create with color markers), shown in Fig. 24 (b), was

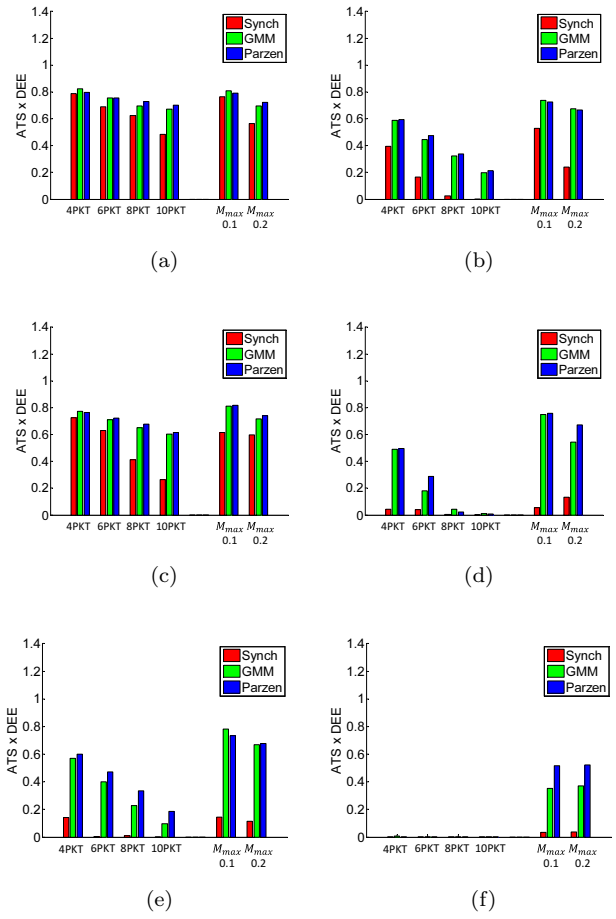


Figure 19: Average tracking scores (ATSs) with delivery energy efficiency (DEE) for the three test sequences. From top to bottom row, each row shows ATS×DEEs for Campus, PETS, and Lab sequences, respectively. From left to right column, each column depicts ATS×DEEs under loss model 1 and 2, respectively.

navigated on a pre-determined track (the ground-truth) using a remote control. We compared the distributed particle filter without the resource-aware capability using 2, 4, and 6 packet loads with the resource-aware particle filter with $M_{max} = 0.2$. The Parzen distributed particle filter, which is robust to packet losses during data aggregation at the cluster head, is applied to all the implementations. The network protocol stack that supports the object tracking application is built with typical wireless sensor network protocols for realistic experimental environments, which includes the energy-efficient random-access MAC protocol T-MAC (Van Dam and Langendoen [2003]) with 30% duty cycle. We also utilized a dynamic camera clustering protocol (Medeiros et al. [2007]) to enable mobile object tracking with static wireless cameras.

As we demonstrated in the previous section, all the implementations of the Parzen particle filters achieve similar tracking performance as measured by ATS and ATE. We can observe the same behavior in Fig. 25 (a) and (b), which show the ATS and ATE measured in the

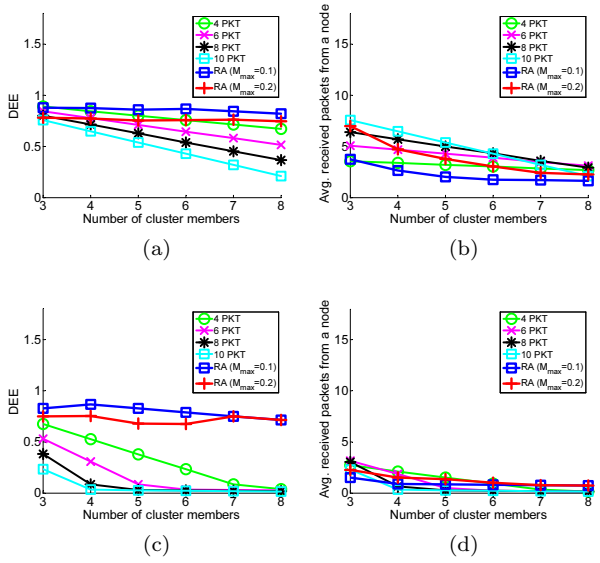


Figure 20: Delivery energy efficiencies (DEEs) and RX packets. (a) and (c): DEEs of 3 - 8 Lab sequences under the loss model 1 and 2, respectively. (b) and (d): RX packets for 3 - 8 Lab sequences under the loss model 1 and 2, respectively.

experiment on the testbed. As shown in Fig. 25 (d), as more transmission packets are assigned per node, the contention for medium becomes more severe, resulting in increased packet losses. It is obvious that such packet losses cause unnecessary energy consumption, and more importantly the increased contention could potentially cause the loss of more critical packets, such as the packets required for the operation of the clustering algorithm. For example, if the packets used for cluster propagation were dropped as a consequence of the increased contention, more frequent tracking failures would take place. When the resource-aware method is applied, however, the packet loss rate is reduced by controlling the packet generation rate at the cluster members, preventing the medium from being saturated and thus leaving enough bandwidth available for the other protocols such as the clustering protocol. Such performance gain of the proposed resource-aware approach is well-captured by the improved DEE as shown in Fig. 25 (c). Fig. 26 shows the estimated target trajectories of the non resource-aware particle filter with 6 packet load and the resource-aware particle filter. Note that the tracking errors are caused by both communication failures and calibration inaccuracies.

6 Conclusions

In this paper, we proposed a resource-aware particle filtering scheme for WCNs. The resource-aware method utilizes an optimization process to achieve a certain level of packet loss rate for the communication of tracking information and to preserve packet delivery energy ef-

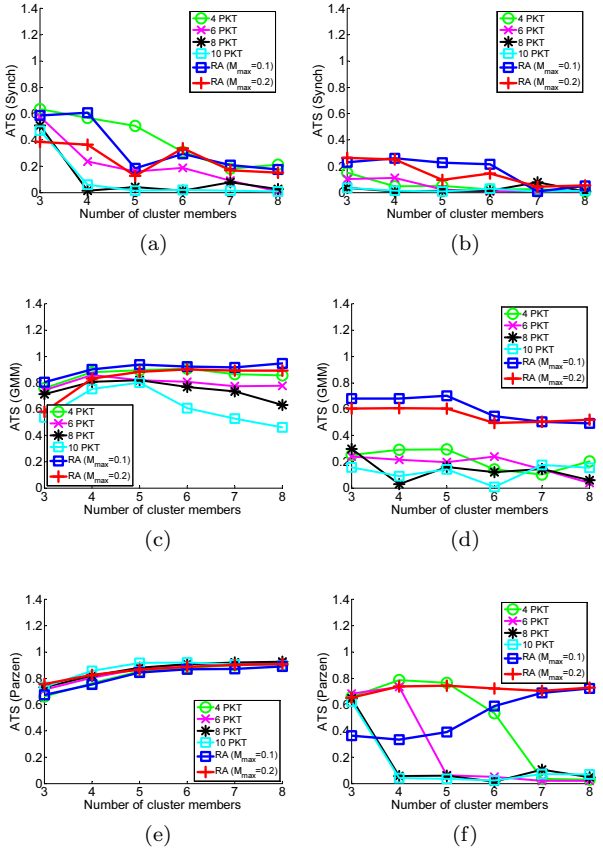


Figure 21: Average tracking scores (ATSs) of 3 - 8 Lab sequences. From top to bottom, each row shows ATSs for the synchronized, GMM, and Parzen particle filters, respectively. From left to right, each column depicts ATSs under loss model 1 and 2, respectively.

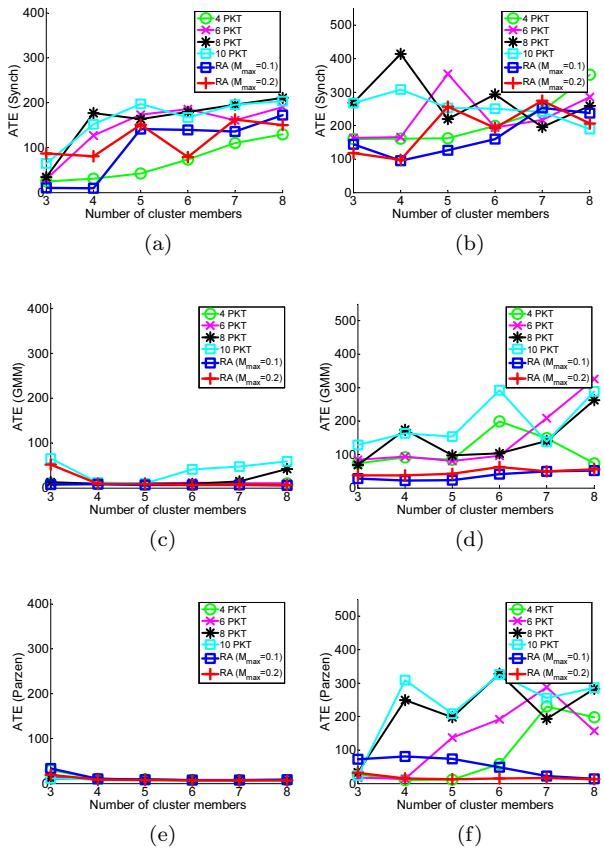


Figure 22: Average tracking errors (ATEs) of 3 - 8 Lab sequences. From top to bottom, each row shows ATEs for the synchronized, GMM, and Parzen particle filters, respectively. From left to right, each column depicts ATEs under loss model 1 and 2, respectively.

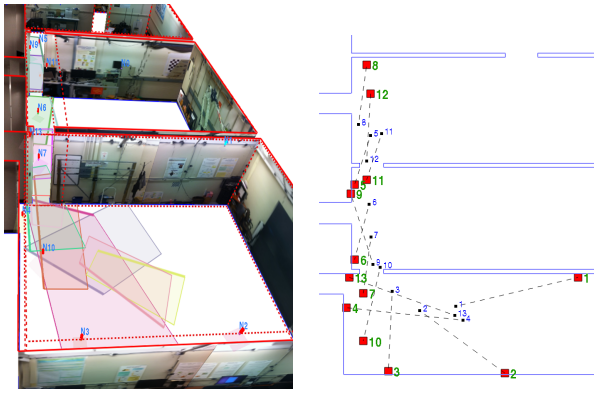
ficiency. This packet allocation procedure alleviates the data loss effects in particle filtering by adjusting the amount of packets generated and transmitted by each camera, thereby reducing the amount of collisions due to a saturated communication medium. This procedure allows collaborative tracking to be carried out using as much data as possible and ultimately leads to more accurate tracking. In addition, we presented three different mechanisms for the exchange of particle information: the synchronized, the GMM, and the Parzen particle filters. We extensively evaluated the performance of these particle filters and analyzed their behaviors under different network traffic conditions. Our experimental results showed that whereas the synchronized particle filter cannot tolerate even small amounts of communication failures, both the GMM and the Parzen particle filters can be employed in lossy environments. The Parzen particle filter is especially robust to communication failures and performs relatively well even when communication quality is extremely low. Our proposed resource-aware packet allocation mechanism improved both the tracking performance and the energy efficiency of the GMM particle filter. As for the Parzen particle filter, because tracking performance is generally robust even in the presence of severe communication problems, the main benefit of the resource-aware approach is to significantly increase its energy efficiency. The resource-aware distributed particle filters are an important tool for the development of effective WCNs that can be employed in real-world surveillance applications.

As for future work, we will consider (1) a resource-aware approach to compute the optimal number of cameras that should join a cluster in the form of cluster members according to the current network conditions and (2) a resource-allocation methodology to allow weighted packet allocation to individual cluster members given the available resources.

References

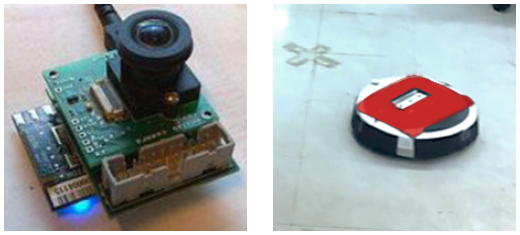
Edwin K. P. Chong and Stanislaw H. Zak. *An Introduction to Optimization*. Wiley-Interscience, 2008.

Mark Coates. Distributed particle filters for sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 99–107, 2004.



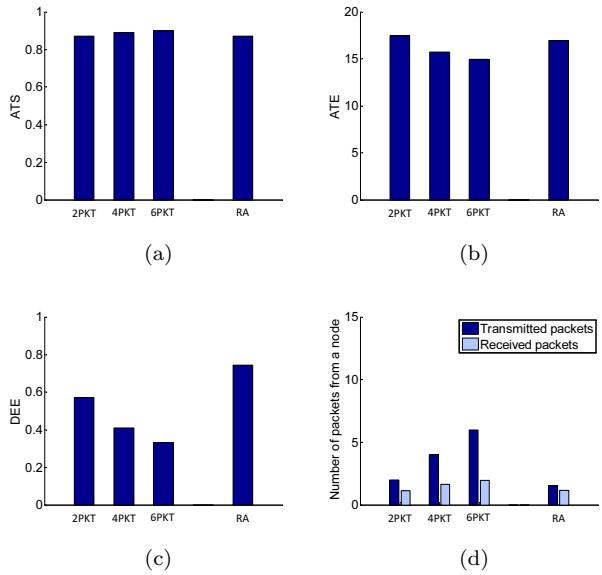
(a) (b)

Figure 23: The RVL wireless camera network testbed. (a) A 3D view of the testbed: The camera sensing range of each camera is shown as a colored polygon. (b) A floor plan view of the testbed: The location of the camera nodes are indicated by the red boxes and the center of each camera sensing range is shown with a small black dot (a camera and its corresponding sensing center are connected with dotted lines).



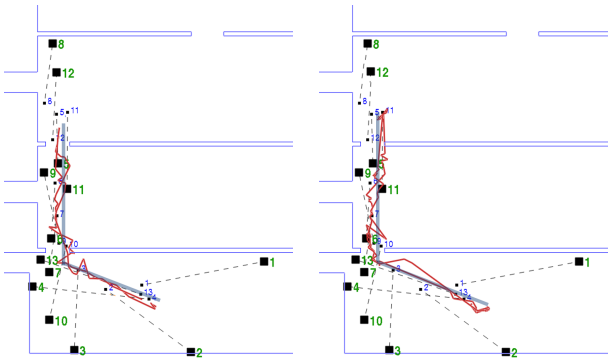
(a) (b)

Figure 24: Network camera and target object. (a) the Imote2 smart camera and (b) the target object.



(a) (b) (c) (d)

Figure 25: Distributed particle filter performance on the RVL wireless camera network testbed. (a) Average tracking scores (ATSs), (b) average tracking errors (ATEs), (c) delivery energy efficiencies (DEEs), and (d) TX and RX packets.



(a) (b)

Figure 26: Object trajectories. The blue lines show the ground-truth of the target trajectories and the red lines indicate the estimated target trajectories; (a) non resource-aware particle filter with 6 packet load and (b) resource-aware particle filter with M_{max}= 0.2.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

Ecole Polytechnique Federale de Lausanne. Campus sequences. [Online]. Available: http://cvlab.epfl.ch/data/pom, 2008.

Shahrokh Farahmand, Stergios I. Roumeliotis, and Georgios B. Giannakis. Set-membership constrained particle filter: Distributed adaptation for sensor networks. *IEEE Transactions on Signal Processing*, 59(9):4122–4138, September 2011.

Wei Gao, Hai Zhao, Chunhe Song, and Jiuqiang Xu. A new distributed particle filtering for WSN target tracking. In *Proceedings of International Conference on Signal Processing Systems*, pages 334–337, 2009.

N. Gordon, D. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F, Radar and Signal Processing*, volume 140, pages 107–113, April 1993.

Dongbing Gu. Distributed particle filter for target tracking. In *Proceedings of IEEE International Conference in Robotics and Automation*, pages 3856–3861, April 2007.

Gregory D. Hager, Maneesh Dewan, and Charles V. Stewart. Multiple kernel tracking with SSD. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 790–797, 2004.

Yan Huang, Wei Liang, Hai-Bin Yu, and Yang Xiao. Target tracking based on a distributed particle filter in underwater sensor network. *Wireless Communications and Mobile Computing*, 8(8):1023–1033, October 2008.

Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 134–147, 2004.

IEEE. Wireless LAN medium access control (MAC) and physical layer (PHY) spec, 1998.

Michael Isard and Andrew Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

Yogesh G. Iyer, Shashidhar Gandham, and S. Venkatesan. STCP: A generic transport layer protocol for wireless sensor networks. In *Proceedings of International Conference on Computer Communications and Networks*, pages 449–454, October 2005.

Shafiullah Khan, Al-Sakib Khan Pathan, and Nabil Ali Alrajeh. *Wireless Sensor Networks: Current Status and Future Trends*. CRC Press, 2012.

Hui Ma and Brian W.-H. Ng. Collaborative data and information processing for target tracking in wireless sensor networks. In *Proceedings of IEEE International Conference on Industrial Informatics*, pages 647–652, August 2006.

Henry Medeiros, Johnny Park, and Avinash Kak. A lightweight event-driven protocol for sensor clustering in wireless camera networks. In *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, pages 203 – 210, September 2007.

Henry Medeiros, Johnny Park, and Avinash Kak. Distributed object tracking using a cluster-based Kalman filter in wireless camera networks. *IEEE Journal of Selected Topics in Signal Processing*, 2(4):448–463, August 2008a.

Henry Medeiros, Johnny Park, and Avinash Kak. A parallel color-based particle filter for object tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008b.

Henry Medeiros, German Holguin, Paul J. Shin, and Johnny Park. A parallel histogram-based particle filter for object tracking on SIMD-based smart cameras. *Computer Vision and Image Understanding*, 114(11):1264 – 1272, 2010.

Lama Nachman, Jonathan Huang, Junaith Shahabdeen, Robert Adler, and Ralph Kling. IMOTE2: Serious computation at the edge. In *Proceedings of Wireless Communications and Mobile Computing Conference*, pages 1118–1123, August 2008.

Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. A color-based particle filter. In *Proceedings of International Workshop on Generative-Model-Based Vision*, volume 2002/01, pages 53–60, 2002.

Lee-Ling Ong, Ben Upcroft, Matthew Ridley, Tim Bailey, Salah Sukkarieh, and Hugh Durrant-Whyte. Decentralised data fusion with particles. In *Proceedings of Australasian Conference on Robotics and Automation*, 2005.

Lee-Ling Ong, Ben Upcroft, Tim Bailey, Matthew Ridley, Salah Sukkarieh, and Hugh Durrant-Whyte. A decentralised particle filter algorithm for multi target tracking across multiple flight vehicles. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4539–4544, 2006a.

Lee-Ling Ong, Ben Upcroft, Matthew Ridley, Tim Bailey, Salah Sukkarieh, and Hugh Durrant-Whyte. Consistent methods for descentralised data fusion using particle filters. In *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 85–91, September 2006b.

Lee-Ling Ong, Tim Bailey, Hugh Durrant-Whyte, and Ben Upcroft. Decentralised particle filtering for multiple target tracking in wireless sensor networks. In *Proceedings of International Conference on Information Fusion*, pages 1–8, 2008.

Patrick Perez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color-based probabilistic tracking. In *Proceedings of European Conference on Computer Vision*, pages 661–675, 2002.

Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2001.

Reading University. PETS 2009 data sequence. [Online]. Available: http://www.cvg.rdg.ac.uk/PETS2009, 2009.

Matthew Ridley, Ben Upcroft, Lee-Ling Ong, Suresh Kumar, and Salah Sukkarieh. Decentralised data fusion with Parzen density estimates. In *Proceedings of Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 161–166, 2004.

Xiaohong Sheng and Yu-Hen Hu. Distributed particle filters for wireless sensor network target tracking. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 845–848, March 2005.

Xiaohong Sheng, Yu-Hen Hu, and Parameswaran Ramanathan. Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, pages 181–188, April 2005.

Paul J. Shin, Henry Medeiros, Johnny Park, and Avinash Kak. Predictive duty cycle adaptation for wireless camera networks. In *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–6, 2011.

Chunhe Song, Hai Zhao, and Wei Jing. Asynchronous distributed PF algorithm for WSN target tracking. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pages 1168–1172, June 2009.

Ben Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: Scalable sensor network simulation with precise timing. In *Proceedings of International Symposium on Information Processing in Sensor Networks*, pages 25–27, April 2005.

Tijs Van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 171–180, 2003.

Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell. CODA: Congestion detection and avoidance in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 266–279, November 2003.

Chonggang Wang, Kazem Sohraby, Victor Lawrence, Bo Li, and Yueming Hu. Priority-based congestion control in wireless sensor networks. In *Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pages 22–31, 2006.

Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 1–13, 2003.

Long Zuo, Kishan Mehrotra, Pramod K. Varshney, and Chilukuri K. Mohan. Bandwidth-efficient target tracking in distributed sensor networks using particle filters. In *Proceedings of International Conference on Information Fusion*, pages 1–4, July 2006.