AN ENERGY-EFFICIENT AND REACTIVE REMOTE SURVEILLANCE FRAMEWORK USING WIRELESS MULTIMEDIA SENSOR NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HAKAN ÖZTARAK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

MAY 2012

Approval of the thesis:

AN ENERGY-EFFICIENT AND REACTIVE REMOTE SURVEILLANCE FRAMEWORK USING WIRELESS MULTIMEDIA SENSOR NETWORKS

submitted by **HAKAN ÖZTARAK** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering**

Prof. Dr. Adnan Yazıcı
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Assoc. Prof. Dr. İbrahim Körpeoğlu
Department of Computer Engineering, Bilkent University

Prof. Dr. Adnan Yazıcı
Computer Engineering, METU

Assoc. Prof. Dr. Kemal Akkaya
Computer Science, Southern Illinois University Carbondale, USA

Assoc. Prof. Dr. Ahmet Coşar
Computer Engineering, METU

Assist. Prof. Dr. Selim Temizer
Computer Engineering, METU

Date:

Name, Last Name:	HAKAN ÖZTARAK
Signature :	

ABSTRACT

AN ENERGY-EFFICIENT AND REACTIVE REMOTE SURVEILLANCE FRAMEWORK USING WIRELESS MULTIMEDIA SENSOR NETWORKS

Öztarak, Hakan
Ph.D., Department of Computer Engineering
Supervisor : Prof. Dr. Adnan Yazıcı

May 2012, 135 pages

With the introduction of Wireless Multimedia Sensor Networks, large-scale remote outdoor surveillance applications where the majority of the cameras will be battery-operated are envisioned. These are the applications where the frequency of incidents is too low to employ permanent staffing such as monitoring of land and marine border, critical infrastructures, bridges, water supplies, etc. Given the inexpensive costs of wireless resource constrained camera sensors, the size of these networks will be significantly larger than the traditional multi-camera systems. While large number of cameras may increase the coverage of the network, such a large size along with resource constraints poses new challenges, e.g., localization, classification, tracking or reactive behavior. This dissertation proposes a framework that transforms current multi-camera networks into low-cost and reactive systems which can be used in large-scale remote surveillance applications. Specifically, a remote surveillance system framework with three components is proposed: 1) Localization and tracking of objects; 2) Classification and identification of objects; and 3) Reactive behavior at the base-station. For each component, novel lightweight, storage-efficient and real-time algorithms both at the computation and communication level are designed, implemented and tested under a variety of conditions.

The results have indicated the feasibility of this framework working with limited energy but having high object localization/classification accuracies. The results of this research will facilitate the design and development of very large-scale remote border surveillance systems and improve the systems' effectiveness in dealing with the intrusions with reduced human involvement and labor costs.

Keywords: Wireless Multimedia Sensor Networks, Active Rules, Object Localization, Object Classification, Object Tracking, Complex Event Processing

ÖZ

KABLOSUZ ÇOKLU ORTAM SENSÖR AĞLARI KULLANAN, ENERJİ YÖNÜNDEN VERİMLİ VE TEPKİSEL, UZAKTAN KONTROL EDİLEN GÖZETLEME YAPISI

Öztarak, Hakan
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Adnan Yazıcı

Mayıs 2012, 135 sayfa

Kablosuz Çoklu Ortam Sensör Ağlarının takdimiyle, kameraların büyük çoğunluğu pil ile işleyecek olan, büyük ölçekli uzaktan kontrollü dış mekan gözetleme uygulamaları öngörülmektedir. Bu uygulamalar, hadiselerin oluşma sıklığı devamlı personel görevlendirilmesi için çok düşük olan, kara veya deniz sınırlarını, kritik yapıları, köprüleri ve su kaynaklarını vb. izleme uygulamalarıdır. Kaynakları sınırlı, ucuz maliyetli kamera sensörleri göz önüne alındığında, bu ağların büyüklüğü, geleneksel çok kameralı sistemlerden daha büyük olacaktır. Çok fazla kamera, ağın kapsamını genişletebilirken, kaynak sıkıntısı ile birlikte böyle büyük bir boyut, yeni sorunlar ortaya çıkarmaktadır, örneğin, yer tespiti, sınıflandırma, izleme veya tepkisel davranışlar. Bu doktora tezi, günümüzdeki çok kameralı ağları, büyük boyutlu uzaktan kontrollü gözetleme uygulamalarında kullanılabilen, düşük maliyetli ve tepkisel sistemlere dönüştüren bir yapıyı önermektedir. Özellikle, üç bileşene sahip uzaktan kontrollü gözetleme yapısı önerilmektedir: 1) Nesnelerin yerlerinin tespit edilmesi ve izlenmesi 2) Nesnelerin tanımlanması ve sınıflandırılması; ve 3) Üs-istasyonundaki tepkisel davranışlar. Her bir bileşen için, hesaplama ve iletişim seviyelerinin her ikisinde de özgün ve hafif, depolama yönünden verimli ve gerçek zaman kabiliyetine sahip algoritmalar tasarlanmış, geliştirilmiş ve çeşitli koşullar altında test edilmiştir. Sonuçlar, yüksek nesne yer tespiti ve sınıflandırması doğruluklarına sahip olan, fakat limitli enerji gideri ile çalışan bu yapının uygulanabilirliğini göstermektedir. Bu araştırmanın sonuçları, çok büyük ölçekli uzaktan kontrollü sınır gözetleme sistemlerinin tasarım ve geliştirmesi ile birlikte, işçilik maliyetleri ve insan müdahelesi düşürülmüş, davetsiz misafirlerle başa çıkan sistemlerin etkiliklerinin artırılmasına olanak sağlayacaktır.

Anahtar Kelimeler: Kablosuz Çoklu Ortam Sensör Ağları, Aktif Kurallar, Nesnelerin Yerlerinin Tespit Edilmesi, Nesnelerin Sınıflandırılması, Nesnelerin İzlenmesi, Karmaşık Olay İşleme

To My Family...

ACKNOWLEDGMENTS

I would like to express my inmost gratitude to my supervisor Prof. Dr. Adnan Yazıcı for his guidance, advice, insight throughout the research and trust on me. It is an honor for me to share his knowledge and wisdom.

This dissertation would not have been possible without continuous support of Assoc. Prof. Dr. Kemal Akkaya. I would like to record my gratitude to him for his supervision from the very early stage of this research as well as giving me extraordinary experiences through out the work.

Additionally, I am grateful to Turgay Yılmaz for his support during the "Efficient and Accurate Object Classification" part of this dissertation.

I am indebted to my parents for all their support and self-sacrifice on my behalf.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Wireless Multimedia Sensor Networks (WMSN) receive a great deal of attention because of their potential to be deployed flexibly in various outdoor applications relatively cheaply [Akyildiz07]. Such networks deploy a large number of camera (video) sensors with different capabilities and can collect/process multimedia data [Rahimi05][Hengstler07]. Typical applications of WMSNs include multimedia surveillance, target tracking, habitat monitoring, and intrusion detection [Akyildiz07][Akyildiz08]. In this dissertation, we focus on surveillance applications, where the frequency of incidents is too low to employ permanent staffing (such as monitoring of land and marine border, critical infrastructures, bridges, water supplies, etc.) and consequently need a reactive framework to reduce the human involvement.

In surveillance applications, large number of battery operated image/video camera sensors are deployed to capture the occurring events. These cameras are typically low-resolution and severely resource constrained in terms of CPU, memory and storage which makes them cheaper and convenient to be deployed. As a result of these constraints, current WMSN researches focus on energy-efficiency issues in terms of video data routing, quality of service (QoS) and coverage for increasing the network lifetime [Akyildiz07][Soro09]. Most of these studies, however, are at the network layer or lower which do not take into account active data processing related research at the camera sensors. Nevertheless, there are several traditional image/video processing problems specific to multi-camera surveillance systems that may consume significant energy and yet to be tackled under active framework of WMSNs. For instance, localization, tracking, classification and identification of objects as well as events are such traditional problems whose active solutions may directly affect the WMSN lifetime. This is because, with the reduction or compression of multimedia data at the camera sensor nodes, bandwidth and transmission requirements can be relieved which in turn increases the lifetime of the nodes. On the other hand, with the reduction of data, the quality of object classification and event identification should not be compromised. This relationship of processing and communication indicates the need for WMSNs to be able to perform image/video data processing with limited/uncertain resources in an energy-efficient manner. In this way, the gap between networking and data processing research can be filled so as to realize the deployment of WMSNs in a wide variety of real-life applications. This dissertation proposes a framework to transform current multi-camera networks into efficient and low-cost active systems that can be used in large-scale remote surveillance. We explore the main challenges of this reactive framework in the following paragraphs:

Localization and Tracking of Objects: One of the major traditional problems in surveillance systems is to locate and track multiple objects in real-time. This is very crucial given that surveillance applications are geared for security and safety. For instance, in a border surveillance application, the location of the intruder needs to be communicated to an officer in time so that the officer can take the necessary actions immediately. Similarly, in a target tracking application, the location information is needed to determine the path of the intruder or to prepare a defense which might even include shooting the intruder. In addition, if there are multiple objects in the area, accurate tracking of each object is needed not to miss any events associated with them.

While the problems of object localization and tracking in surveillance applications are well-studied in the literature [Spors01][Huiyu08][Qi07][Bramberger06][Guo05][Javed03], these earlier studies are conducted in the context of traditional multimedia systems where powered video and image cameras with significant processing capabilities are available. While a large number of cameras in WMSNs may increase the coverage of the network, increased size along with resource constraints pose new challenges for both localization and tracking. To be more explicit, as far as the localization problem is concerned, the schemes proposed in the past for traditional multimedia systems require complex signal processing, meaning a powerful processing capability and the compulsory involvement of multiple cameras. These approaches cannot be directly applied in our context for resource constrained WMSNs. Furthermore, energy overhead due to processing and communication becomes a critical issue in addition to the real-time nature of the process.

For object tracking, there are even more challenges: In the first place, because we intend to keep the energy consumption low, the individual cameras cannot perform tracking by themselves. Besides, individual cameras' Field of Views (FoV) are not enough to track the detected objects throughout the entire area under surveillance. Hence, a base-station, (which will be called afterwards as "sink"), may collect information from multiple cameras and performs the function of tracking instead of individual sensors. Secondly, the use of an increased number of cameras introduces the problem of overlapping areas, meaning that the FoVs of multiple cameras can overlap and thus several cameras may report the same objects to the sink. Under these circumstances, if there are multiple objects being tracked in the region, the reported object locations and the number of objects in the monitored region may not be accurate. This not only generates false alarms for surveillance personnel but also renders accurate real-time tracking impossible. Finally, if the camera sensors are deployed randomly, the distribution of cameras' FoVs may create blind areas where objects become invisible to the WMSN. In light of these challenges, novel approaches for object localization and tracking in WMSNs are needed.

Classification and Identification of Objects: Another traditional problem which can also be employed within object tracking problem is the object classification problem. Deploying a large number of battery-operated image/video camera sensors in the surveillance region helps to alleviate issues regarding possible obstacles (blocking cameras) and thus provides great convenience in terms of object/event coverage. However, accurate classification of the detected objects regardless of the WMSN size and area coverage percentage is still a challenge. If the detected objects are classified correctly on site, then the sink may be alarmed based on the received object information. This is very crucial given that surveillance applications are geared for security and safety. For instance, in a power-plant surveillance application, only human intruders who are strangers should trigger an alarm for the guards. In case of a detected animal or an employee, no alarm may be necessary.

In order to perform an accurate object classification, an effective set of features should be selected and a robust classifier should be constructed [Datta05][Smeulders00][Jain00]. Nevertheless, limited resources of WMSNs restrict the options for choosing the features and the classifier. Specifically, features and classifiers which are lightweight in terms of processing, energy consumption and storage are needed. In addition, real-time applicability of the classifier is crucial considering the fact that the classification process is performed at the camera sensors when an object is detected. Finally, the flexibility of the classifier for adding new features and object classes and making it applicable for other domains is also a big plus.

Event Identification and Reactive Behavior at the Sink: In surveillance applications having large number of cameras deployed, when the objects are classified and tracked, there is also a need for a large number of personnel in order to monitor the received data from the cameras. This is not feasible in large-scale applications like border monitoring [WEB:BorderReport05]. Therefore, a reactive system, which can identify the events and take certain actions, is a convenient solution in reducing the human involvement.

However, due to the goal of data reduction at the camera sensors to save energy, there is a limited/uncertain/imprecise data at the sink regarding detected objects. Deciding which actions should be taken at the sink becomes more challenging with such data. Specifically, the reactive system should have the capability of reasoning when it has only uncertain and incomplete information. In addition, there needs to be a fusion mechanism at the sink before the collected data are processed. This is due to the redundancy of the received data. Typically, due to the availability of a large number of low-cost wireless camera sensors, an event may be captured by different camera sensors (i.e., the same event may be captured under the Field-of-Views (FoV) of several camera sensor nodes), creating redundancy at the sink. Hence before taking an action, the detected objects and their corresponding events should be analyzed and identified. Finally, due to the complexity of events and the multi-modal nature of the data gathered at the sink, fuzziness in events needs to be supported.

1.1 Contributions of the Dissertation

This dissertation proposes an energy-efficient and reactive remote surveillance framework using WMSNs. Specifically; the proposed framework is compromised from novel, lightweight, storage-efficient and real-time algorithms for WMSNs to address the aforementioned problems [Oztarak07]. Our contributions on this framework can be listed as follows:

- We propose a novel lightweight object localization and tracking scheme for WMSNs [Oztarak09][Oztarak12-2]. The localization scheme extracts the objects from video frames using frame differencing. Once the object is extracted, the distance between object and sensor is calculated by using the height of the object and camera sensor properties. Then, its location is calculated by using the camera's location and the distance between the object and the camera. The distance along with the minimum bounding box of the objects are sent to the sink for tracking purposes. The tracking is performed at the sink using a fuzzy similarity algorithm to identify the objects located.

- We propose an object classification algorithm based on genetic algorithms [Oztarak12]. We choose two simple but effective features which are extracted from the video frames in the phase of object localization: "Shape_Ratio" and "Speed" of the detected objects. The idea is to eliminate the cost of the extraction of the features to save energy and improve the WMSN lifetime. For the classifier, we employ a genetic algorithm based classifier for the best accuracy. The training of the classifier is done offline with a set of sample objects and then the classifier is stored at each camera sensor before the online data collection starts.

- Considering the surveillance applications using WMSNs, in this dissertation, we also propose a reactive mechanism for not only fusing uncertain data at the sink but also automated processing of data using active rules extending the classical Event-Condition-Action (ECA) structure [Oztarak11]. In this mechanism, data fusion is performed using fuzzy logic to handle impreciseness and uncertainty in the received data. The fused data is then processed to identify simple and complex events and then to infer predefined actions based on these events. This step is referred to as the *WHEN* part of the extended active rules. We propose that this syntax be employed as WECA rules. For instance, based on the action part of the rule, raw video data may be requested from the appropriate camera node only when some events and conditions are met. In this way, one can get rid of the burden of receiving huge raw video data at all times from the camera nodes.

1.2 Organization of the Dissertation

The rest of this dissertation is organized as follows: Chapter 2 gives a review of the literature that is related to our work. Chapter 3 presents WMSNs background and explores our system definition. Chapter 4, 5, 6 and 7 initially define the problems of "Lightweight Object Localization with a Single Camera", "Efficient and Accurate Object Classification", "Efficient Object Tracking" and "Active Rule Processing" consecutively. Each chapter then gives details about our solution to the corresponding problems, solution analysis and performed experiments with their results. Lastly, Chapter 8 states the conclusions and presents the details about the future works. As an appendix, implementation details are presented.

CHAPTER 2

RELATED WORK

In this chapter, we survey the state of the art under several categories that are related to our studies.

2.1 Video Surveillance

In the literature, video surveillance focusing on object extraction is studied for more traditional multimedia systems concentrating on the automatic data extraction problem such as moving object extraction, object classification, object identification, object trajectory finding or activity classification [Adali96] [Black04] [Hampapur05] [Lyons00] [Garofalakis02] [Orten05] [Rangaswami04] [Li97]. In video surveillance applications, besides the ability to store and process the retrieved multimedia data, multimedia sensors should also infer and record potentially relevant activities (thefts, car accidents, traffic violations, border intrusions etc.) and make video/audio streams or reports available for future query [Akyildiz07]. For instance, IBM smart surveillance engine [Hampapur05] detects moving objects, tracks multiple objects, classifies objects and events, and then executes event based queries. Similar to this study, in [Lyons00] video sequences are retrieved for content analysis for surveillance applications. The cooperation of multiple camera outputs to track moving objects is studied in [Black04][Rangaswami04]. However, in the context of resource-constrained WMSNs, little work is done in terms of video surveillance.

In WMSNs, the nature of the data that needs to be communicated to the data collection point, i.e., the sink node, presents novel research challenges. Unlike traditional multimedia approaches where resourceful sources, e.g., servers, communicate to potentially resource-constrained receivers, e.g., mobile devices; there is a pressing need to reverse the processing complexity. In this regard, it is preferable to deploy efficient processing at the source to decide on what to transmit at the possible cost of increased processing complexity at the receiver, e.g., the sink node. Recent studies on WMSNs focus on the processing and compression of multimedia (video/image) data for increasing network lifetime, providing Quality of Service (QoS) for data communication and placement of cameras for desired coverage [Akyildiz07][Akkaya08][Akkaya09][Erdem06].

2.2 Object Detection and Localization

The localization problem receives a lot of attention particularly in traditional multimedia systems. The majority of these works study the localization problem with the involvement of multiple cameras. For instance, [Spors01] proposes a solution by integrating different types of audiovisual sensors. In this approach, a video-based face tracker and a microphone array are used to locate the object. The authors try to reduce their error rate by using multiple sensors. [Huiyu08] presents another approach that uses both visual and audio sensors. The target is located through color-based change detection in the video modality and through estimation of the time difference of arrival between the two microphones in the audio modality. Note that both of these methods work indoors with single-hop powered multi-camera networks, and the challenges of WMSNs do not exist. The method proposed in [Qi07] is based on the use of man height and uses a single camera similar to our approach. In this method, the cameras are assumed to be located above the scene (i.e., ceiling or top of the building) and geared for indoor applications. However, the approach is geared for human recognition applications and does not consider any other objects. In addition, the calculations used in [Qi07]'s algorithm consider both maximum and minimum points seen in the grounds, which is not suitable for WMSNs where cameras may have varying orientations. Nonetheless, the height estimation in this approach can be used in our proposed approach. Camera positioning assumption is not suitable for WMSNs where the camera sensors may be located parallel to the ground and directed to the horizon.

Other pursued approaches related to camera sensors strive to determine the location of the camera sensors rather than intruders or objects. For instance, in [Kassebaum08], the authors try to determine the projection matrix for a camera viewing a 3-D target. Since the location and orientation of the target is known in advance, any two cameras that locate the same target position automatically locate the target's geometry. In order to locate an entire view-connected smart camera network, the system described in [Kassebaum08] requires moving the target through the overlapping views of all pairs of cameras. Another work that studies the determination of location of a camera sensor is reported in [Ravi05]. In this work, the authors use an image database to compare video frames in order to find where the sensor is located.

In addition to these approaches, it is worth noting that there is also a huge literature on WSN localization schemes based on various techniques like [Patwari05]. These approaches as well as the approaches for determining the locations of camera sensors are different from ours. Basically, we try to locate the objects/intruders in the environments using cameras, assuming that the camera locations are already known. Therefore, sensor/camera localization techniques complement our work.

2.3 Object Classification

Object classification is also part of surveillance applications which can be performed in conjunction with object detection and tracking. The main challenges with object classification stay with video processing, data elimination and data reduction so as to minimize communication and coordination needs of the involved camera sensors for saving energy. However, currently, there is not much research focusing solely on this aspect of WMSNs which indirectly affects the networking/communication costs. The works in resource constraint WSNs (such as [Brooks03]) do not apply as they work only on scalar sensors not on cameras. Our work is one of the first to address this issue in the context of WMSNs at resource constraint cameras rather than the resource rich sink node. Given that there is no work in WMSNs on classification, next we look at the related literature on object classification and feature selection in general surveillance applications without any resource constraints.

Until now, a significant amount of research has been done on classification problem in general. A detailed discussion on several types of classifiers and an extensive comparison of them can be found in [Jain00] and [Michie94]. Below, we present a broad categorization of classifiers according to their decision units. Also, we analyze their applicability to WMSNs according to their complexity and flexibility.

The classifiers in literature can be grouped into three different approaches: Instance-based, probabilistic and decision boundary-based [Jain00]. In an instance-based classifier, objects are classified according to the similarities with the prototypes. The classifier model holds the prototype classes, prototype objects in each class and their features. In order to classify a given object (i.e., query object), similarity of the object with each prototype class is calculated. Thus, the complexity of the classifier in terms of processing, energy and time depends on the similarity function and the features chosen. If reasonable distance functions with some simple features are utilized, this approach provides a lightweight solution. Moreover, considering that the raw features are hold in the classifier model, it is easy to extend the classifier by including new features and objects classes. However, this approach suffers from the risk of low accuracy if its model contains a small number of prototypes for each class and they do not represent the classes well or noise exists in the prototypes. There may be also a high storage requirement if the model is constructed with a large number of prototypes in order to represent the prototype classes better. Besides, the probabilistic and the decision boundary-based approaches perform better than the instance-based approaches at the expense of more complex operations. Some of the well-known classifiers of this type are as follows: Template matching classifier, Nearest mean classifier, Subspace method, 1-Nearest neighbor rule [Duda01].

Probabilistic classifiers perform classification by using the generative probability models obtained via the distributions of object classes over the feature space. The classifier model does not hold the features, but holds the probability models. Thus, the storage requirement is less for this type. However, it may not be possible to obtain the flexibility as in the instance-based approaches since the probability models should be recalculated when new features and classes are included. The accuracy of probabilistic classifiers is reasonably high considering the low complexity of the classifier model, but it is sensitive to the outliers and class density estimation errors. Some of the well-known classifiers of this type are: Naive Bayes classifier, K-nearest neighbor rule, Logistic classifier, Parzen classifier [Duda01].

The decision boundary-based classifiers construct decision boundaries by optimizing some certain error criteria and perform classification based on these boundaries. Decision boundary approach is the most complex one among the three approaches with respect to the processing. However, the accuracy of this approach is usually better. As in the case of the probabilistic approaches, it is not easy to include new features and classes. The decision boundary should be recalculated. Some of the well-known classifiers of this type are as follows: Fisher linear discriminant, Perceptron, Binary decision tree, Neural networks, Support vector classifier [Duda01].

Considering these three approaches, the third one is the least convenient for WMSNs because of its complexity and inflexibility though it provides the best accuracy. Thus, we prefer to use a classifier that benefits from the advantages of both the instance-based classifiers and the probabilistic classifiers, as described in [Yilmaz11]. The classifier is first constructed like an instance based classifier by respecting the low-complexity requirements of the WMSNs. Yet, the classifier model is constructed with a large number of prototypes. Then, by applying genetic algorithm operations, the prototypes are regulated according to their fitness and the probability knowledge present in the training data is absorbed so that the classifier is enhanced.

2.3.1 Feature Selection

One important issue for classification of objects is the representation of images. There exist an excessive number of visual features proposed for the representation of images. The features can be classified as global and local features. The former include color [Manjunath01], shape [Bober01] and texture [Manjunath01] descriptors on the whole image while the latter is based on some special local points in the image (SIFT [Lowe99], SURF [Bay08]). However, most of these features are low-level features. Extraction and utilization of such low-level features impose a high processing complexity as far as the WMSN applications are concerned.

As a result, we prefer utilizing more semantic features which can be easily extracted during normal video processing for tracking purposes (e.g., object detection or localization). The goal is to reduce the burden on camera sensors when extracting the features. However, there is a tradeoff here regarding the number and type of the features used and the classification accuracy. Our features of object shape and object speed are unique and novel in the sense that they do not require additional processing while they can still represent the objects for accurate classification.

2.4 Object Tracking

Object tracking, identification, re-identification (i.e., deciding if a presently visible object has already been observed somewhere else before), and classification have been well-studied in the past [Spors01][Huiyu08][Qi07][Bramberger06][Guo05][Javed03][Collins01] [Sharif09] using traditional multimedia systems where powered video and smart image cameras were available with sufficient processing capabilities. The solutions presented in these studies cannot, therefore, be directly applied to our case as we have different constraints and characteristics. Nonetheless, we will discuss how our classification and tracking processes are different from these previous studies.

Due to the resource constraints in WMSNs, object tracking and identification processes require that objects should be detected from the frames and their locations should be determined in an energy-efficient manner. While [Alaei09] addresses the energy problem with a periodic wake up of the cameras, the approach still transmits the whole frame and such periodicity cannot provide real-time tracking of the object. However, our approach utilizes the coordinates of the objects on the frame, applying a fuzzy classification method for determining the type of the detected object at the camera sensor, without sending any raw video data to the sink. Moreover, since there is no periodicity, real-time tracking is possible.

In order to track the vehicle objects, some studies have made pre-assumptions on where the vehicle is going [Coifman98]. In this way, it is much easier to track and re-identify the vehicles. Our work, however, does not include such assumptions about the behavior of objects. Some of the studies on WMSNs propose distributed or collaborative sensor processing for object tracking. For instance, [Liu09] proposes a nonlinear localization-oriented sensing model for camera sensors. The authors propose a dynamic node collaboration scheme for target tracking. Nevertheless, this approach cannot handle multiple objects that leave the region or disappear temporarily due to obstacles. For tracking of multiple objects, more challenges are identified in [Pasula99]. For example, sensors may be noisy; objects may look similar to each other or object behaviors may be unpredictable. Our fuzzy solution handles such difficult situations as shown in Chapter 6.

While we exploit identification for real-time tracking, it has typically been used for different purposes in the past. For instance, a person identification scheme is investigated in

[Hamdoun08] for a multi-camera surveillance system. This system uses interest-points descriptors to identify individuals appearing at different times throughout the network. Since the system is used in urban areas, the accuracy of person identification depends on the number of interest-points. In [Arth07], the authors identify and describe the vehicle objects by using their local features on traffic scenarios. The matching algorithm compares object signatures for reacquisition and tracking of the objects through the entire camera network. In [Javed02], the authors use the feature vector called a ‘Recurrent Motion Image’ (RMI) to calculate repeated motions of objects. As a result, they are able to distinguish between a single person, a group of people and vehicles. Some of the powered systems even use the interaction of a human operator for these purposes [Park06]. In contrast to these, our identification algorithm is fully automated, simple, and does not require complicated person or vehicle identification algorithms.

Classification is considered a good solution to the problem of identifying the different objects, [Comaniciu00]. For example, people undergo repeated changes in shape while walking, meaning that they are non-rigid. Vehicles, on the other hand, are rigid bodies and do not exhibit repeating change in shape while moving. We exploit a similar but simpler approach in this work. Since the focused application is border monitoring, classifying humans, vehicles or animals in a simple way is sufficient for differentiating one object from the others.

As far as the WSNs are concerned, there are several studies on collaborative signal processing for target classification and tracking [Brooks03][Zhang04][Clouqueur01][Li07]. However, since there are no cameras involved, the event detection techniques in these works are based on numerical sensor readings and thus they are different from those of WMSNs. For instance, [Li07] collects a time series of data maps from the sensor network and detects complex events through matching the gathered data to spatio-temporal data patterns. In our work, we use multimedia events and investigate them in the space and time domain to identify multiple objects at the sink.

2.5 Active Rule Processing

2.5.1 Sensor-Actuator Applications

In the literature, there exist two different working models for sensor-actuator applications [Zoumboulakis04]:

- Demand-Driven Model: The system initiates the action when an external request/query is injected into the system. For instance, in order to do environmental monitoring, a query may be injected into the system periodically [Zaniolo93]. Each sensor in this model is considered as a data source to the database virtual table stored at sink [Gehrke04]. An example query for this may be as follows: *Tell me the average temperature in the region once every five seconds*. These types of applications are not very prevalent due to the lack of possible applications and thus we will not follow this approach.
- Event-Driven Model: An individual sensor node produces an event and propagates it through the network. The system then initiates an action against the produced event. An example application may be detecting forest fires with temperature sensors. This approach is used in reactive applications, [Zaniolo93], which is our focus in this work. Typically, Active Database approach (containing ECA Rules) [Zoumboulakis04] is followed in these applications. The aim of ECA systems is to react against the events occurring in the area under control of the active rules asynchronously [Ceri94][Zaniolo93] [Aiken95][Ghanem93].

While these approaches can respond to the data collected from a WSN, they cannot be applied directly to a WMSN due to the nature of the data produced, as explained in Chapter 1. The data contains both redundancy and imprecision. Thus ECA rules need to be extended to provide data fusion before the rules can be processed. None of the above works consider these issues.

In [Korpeoglu06], by considering the possible outliers in the processed WSN data, the authors propose Fuzzy ECA rules. While processing the Fuzzy ECA Rules, events coming from the sensors are processed directly through a mechanism referred to as composition. In this work, all the collected data are assumed to be scalar and include single modalities like temperature.

In our context, due to the ability of camera sensors to capture objects in a larger area (i.e., FoV is much larger compared to the sensing ranges of simple temperature sensors), when multiple camera sensors send multimedia information regarding the detected objects, the system cannot directly infer whether the sent multimedia information belongs to the same object. In addition, there may be several modalities available on the used sensors, which complicate the use of a mechanism based on composition. Therefore, the fuzzy ECA rule syntax proposed in [Korpeoglu06] is not adequate to deal with the data in WMSNs. We need to apply data fusion and extract possible events before the ECA rules can be applied.

2.5.2 Video Event Identification

In WMSNs, different than WSN, the concept of video event plays an important role. A video event is a high-level semantic concept in videos. In the literature, the translation of low-level content features in video sequences into video events is a popular research topic [Lavee09][Ballan11]. In addition, video event extraction is the highest level task in computer vision. It relies on solutions that use many lower level tasks such as edge detection, optical flow estimation, object recognition, object classification, and tracking. For instance, in [Medioni01], the authors work on Unmanned Airborne Vehicle (UAV) videos. They try to infer whether human beings (small moving objects) or vehicles (big moving objects) present a threat that should be signaled to a human operator. In order to achieve this, they use a detection module to detect the moving parts of the objects, a tracking module to conclude with the path information of individual objects and, lastly, a behavior analysis module to extract the events of the detected objects. Most of their events rely on single moving object and its behaviors according to static objects on the map which are defined by a human operator.

Different than [Medioni01], [Hongeng04] presents an approach that uses spatial events about multiple objects. They have just one object class, whose instances are human beings, and they try to achieve complex events such as “converse” from simple events such as “a person approaches a reference person from a distance” and then “stops at the reference person when he arrives”. All the events they define are spatial events. In the literature, in addition to spatial events, temporal events are also used such as in [Hakeem05], in which they define the world as a complex network of agents and their temporal interactions which are the events. These interactions can be visualized in the form of a hierarchy of events and sub-events. The authors propose a solution to learn and extract these sub-events temporally in video frames.

The solutions proposed for video event understanding, such as mentioned above, firstly include a video event extraction part and then a video event modeling part if needed [Oztarak06]. Since we have limited/incomplete/uncertain data gathered from camera nodes having limited resources, we work on simple video events which discard the video event modeling part. Using ideas similar to those presented in [Hongeng04] and [Hakeem05], we identify events based on the spatial and temporal features of detected objects. However, the goal of this dissertation is not to extract and model complex events, instead we extract simple events to conclude actions in an efficient manner and also to test our reactive system.

2.6 Complex Event Processing

In the literature, complex event processing is defined as extracting information from distributed message systems [Luckham98] and makes it possible to discover more complex events by correlating the simple ones [Flatin07]. In [Etzion10], the complex event processing correlation techniques are listed as follows:

- Event-pattern detection [Li10],
- Event abstraction [Giambiasi06],
- Modeling event hierarchies [Luckham08],
- Detecting relationships between events (for instance causality, membership or timing) [Wang08],
- Abstracting event-driven processes [Dong06].

One of the methods that implements the listed correlation techniques is to use the fuzzy rule based systems. Complex event processing systems use fuzzy rule based systems to model and detect the event relations. In the literature, fuzzy rule based systems are used in a wide variety of application areas such as decision making in medicine [Shtovba00][Lin02], taxi time estimation in airports [Chen11], musical genre classification [Fernandez12] or to extract features from large data sets [Chaudhari08]. There also exist researches that try to use fuzziness in event modeling systems, such as transforming crisp systems into fuzzy systems is also a research area. For instance, in [Lin02], the authors propose to generalize (crisp) discrete event systems (DES) to fuzzy DES. For that purpose, they transform the traditional crisp finite automaton model that models the DES system into fuzzy automaton model. They use biomedicine for their application area. For example, they classify the lung condition states for a patient as excellent, fair and poor; then, they define the transitions between these condition states with a crisp finite automaton. Afterwards, they convert the states into fuzzy ones and convert the transition function to the matrix multiplication that calculates the fuzzy probability for new state values. They use fuzziness well in their work; however they did not work on complex events.

Although, the complex event processing systems are used in literature for Wireless Sensor Networks [Wang10][Dunkel09], there exists no complex multimedia semantic event processing system for Wireless Multimedia Sensor Networks. In our context, semantic events in multimedia are the objects' spatio-temporal relations in multimedia data [Khatib99][Snoek05] [Leonardi02][Oztarak06]. On the other hand, the usage of fuzzy rule based systems in Wireless Multimedia Sensor Networks area is also very limited. For instance, in [Wang10], the authors use the fuzzy rules for scheduling the sensors that send multimedia data to the sink by using the future target position. The low or medium residual energy remaining on camera nodes block them to be awoken. Different than [Wang10], in our application which uses Wireless Multimedia Sensor Network, the simple events are correlated with each other by using fuzzy rules to produce more complex events.

CHAPTER 3

WMSN BACKGROUND AND SYSTEM MODEL

This chapter firstly presents some background information about WMSNs, i.e., camera sensors and their features, architectures and application examples. Then, we define our system model with the assumptions and explain its features in the last section of this chapter.

3.1 WMSN Background

Wireless sensor nodes are used in a wide range of applications such as scientific research, military, healthcare, and environmental monitoring [Akyildiz02]. Sensor nodes collect information about the environment and communicate their observations to a data collection point from where users can access the collected data without the need to travel to the monitored area.

With the availability of low cost and also miniature size cameras, multimedia sensor networks are becoming highly popular to extract more descriptive information about the environment. In Figure 3.1, we depict a sample deployment from the BWN research lab at Georgia Tech [WEB:BWN], Stargate board with an 802.11 card and a MICAz mote interfaced with a Logitech QuickCam Pro 4000 webcam. Cyclops, [WEB:CYC] depicted in Figure 3.2, attaches miniaturized cameras similar to those used in cell phones to Mica motes. These examples enable the nodes to process the images in context and report any new information.

One of the most common used camera sensors is CMUCam3 camera sensor [Rowe07]. It is an intelligent sensor having vision capabilities and used in different applications such as surveillance (as shown in Figure 3.3), robotics (as shown in Figure 3.4), sensor networks, education, interactive toys etc. The features of CMUCam3 are listed as follows in [WEB:CmuCam]:

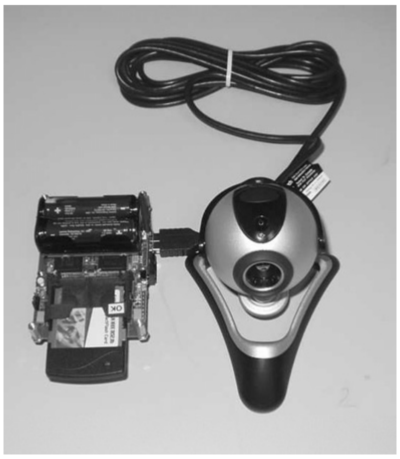


Figure 3.1: Logitech webcam interfaced with Stargate platform

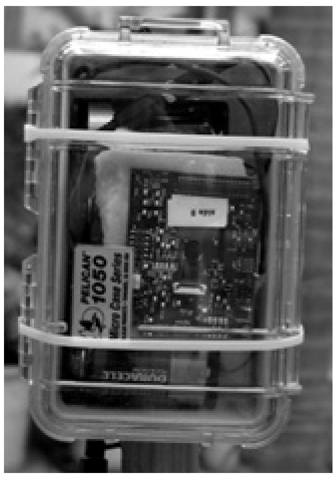


Figure 3.2: Cyclops with a miniature camera

- Data format of CIF (Common Intermediate Format) with resolution (352x288) on RGB color sensor,
- For storage, MMC/SD flash slot with FAT16 driver support (The following cards are known to work with the CMUcam3: PQI 128MB MMC, SanDisk 2GB / 1GB / 512MB SD cards, SanDisk 512MB MMC cards),
- Image processing rate of 26 frames per second,
- Software JPEG compression,
- Basic image manipulation library,

    - Arbitrary image clipping
    - Image down sampling
    - Threshold and convolution functions
    - RGB, YCrCb and HSV color space
    - CMUcam2 emulation
        * User defined color blobs
        * Frame differencing
        * Mean and variance data collection
        * Raw images dumps over serial
        * Histogram generation

- Analog video output (PAL or NTSC),
- FIFO image buffer for multiple pass hi-res image processing,
- Wireless Mote networking interface (802.15.4),
- CMUcam3 Frame Grabber for viewing images on the PC.

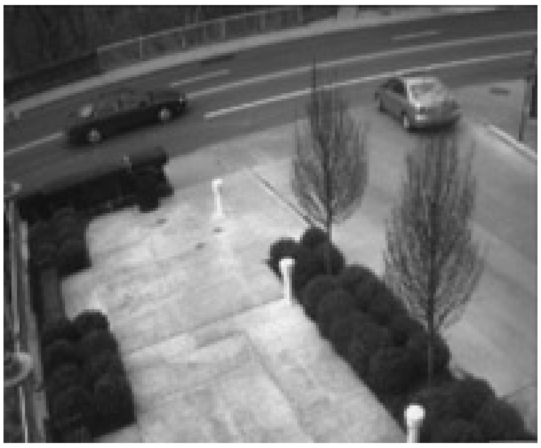


Figure 3.3: Sample frame for the CMUCam3 used in surveillance (original frame is in color)

Multimedia sensor networks can be deployed in a wide range of applications, such as surveillance sensor networks, law-enforcement reports, traffic control systems, advanced health care delivery, automated assistance to elderly, etc. [Akyildiz07]. In these applications multimedia support has the potential of enhancing the level of information collected and enlarging the range of coverage (i.e., in comparison to point measurements of scalar values).

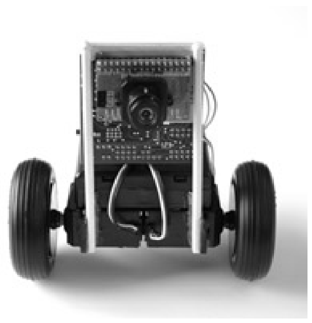


Figure 3.4: A CMUCam3 used in robotics

In [Akyildiz07], the reference architecture for WMSNs is given as shown in Figure 3.5. In Figure 3.5(a), an example of single-tier network of homogeneous camera sensors is illustrated. In this network, some of the camera nodes may have higher processing capabilities and they are referred as processing hubs. The second part, which is shown Figure 3.5(b), is an example of heterogeneous sensor networks which constitutes scalar sensors and camera sensors as a mixture. Additionally, an example of multi-tier heterogeneous sensor networks is shown in Figure 3.5(c). In this kind of networks, lower tier may include low-resolution imaging sensors and the higher tier may be composed of high-end pan-tilt-zoom cameras.

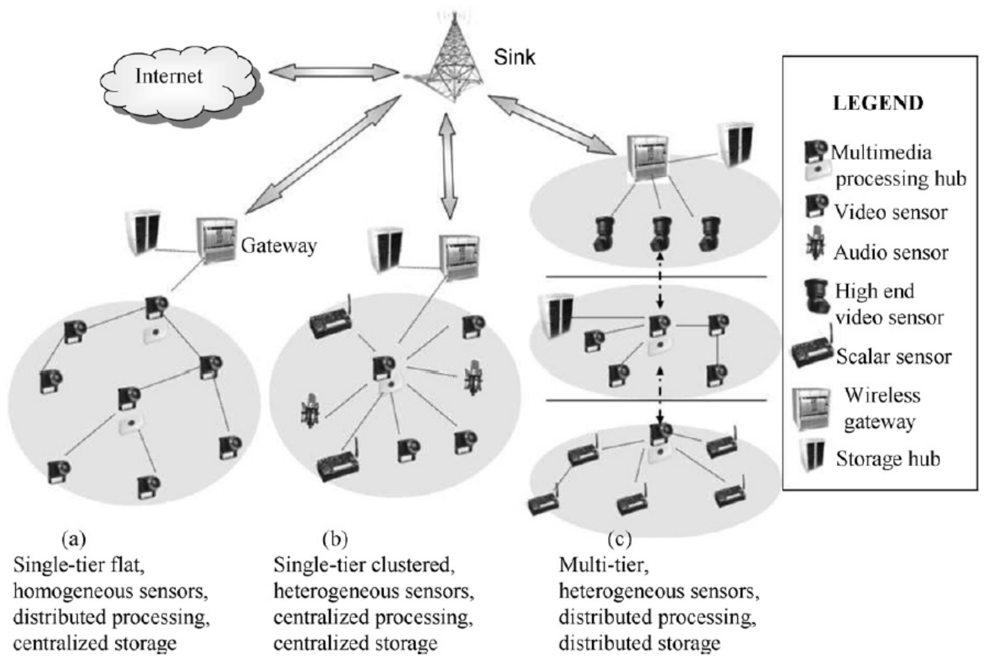

Internet
Sink
Gateway
LEGEND
Multimedia processing hub
Video sensor
Audio sensor
High end video sensor
Scalar sensor
Wireless gateway
Storage hub
(a)
Single-tier flat, homogeneous sensors, distributed processing, centralized storage
(b)
Single-tier clustered, heterogeneous sensors, centralized processing, centralized storage
(c)
Multi-tier, heterogeneous sensors, distributed processing, distributed storage

Figure 3.5: Reference architecture for WMSNs

3.2 System Definition

In this dissertation, the WMSN system is single-tier flat and contains homogeneous camera sensors (e.g., CmuCam3 [Rowe07]), placed randomly in a surveillance application (e.g., at borders, bridges, water supplies) to monitor moving objects [Akyildiz07].

Each camera sensor in our WMSN has a certain Field-of-View (FoV) β and Depth-of-Field (DoF) d, as shown in Figure 3.6, which are the angle and the distance respectively. FoV is the area within which the camera sensor can capture an accurate image or video. The camera sensors are assumed to have a random but fixed position and orientation. Location and orientation can be determined initially using one of the techniques described in the literature [Patwari05]. Since this is done only once, the energy cost of this process would be negligible. Camera sensors can communicate with one another as long as they are within the transmission range of each other. Camera sensors discover their neighbors automatically and use Dijkstra's shortest path algorithm to send their messages to the sink. The sink node is assumed to have more resources than the camera sensors in terms of processing and power.

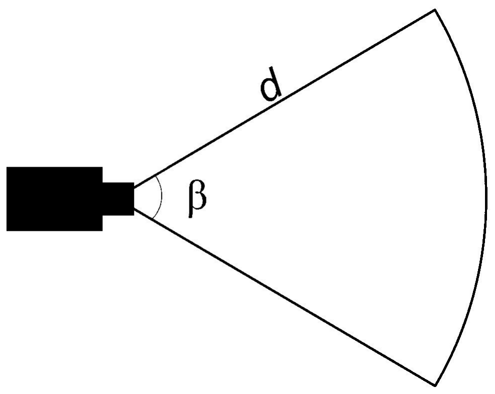


Figure 3.6: A camera sensor with its FoV (β) and DoF (d)

Finally, the monitored region may have many intersected (overlapped) areas as well as some blind areas, due to the random camera deployment. A *Blind Area* is the area which is not covered by any of the cameras' FoVs. An example of a blind area is shown in Figure 3.7, as the whole area around FoVs in black. On the other hand, an *Intersected Area* is the area which is covered by more than one camera FoV. An example of an intersected area is shown in Figure 3.7, as the intersection of the FoVs in gray.

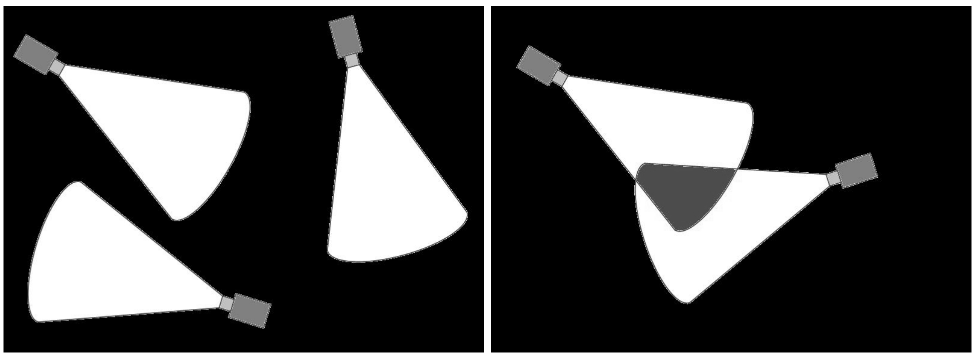


Figure 3.7: Blind and overlapped areas shown in black and gray respectively

In our surveillance application (border surveillance or power plant surveillance), we assume three general object classes that may be captured: *human*, *animal* and *vehicle*. In this application, the purpose of each object is assumed to be entering the area and then moving towards the sink as if it aims to capture an important structure positioned at the sink. We assume that these objects move individually and are not very close to each other (i.e., they do not stick together as a group).

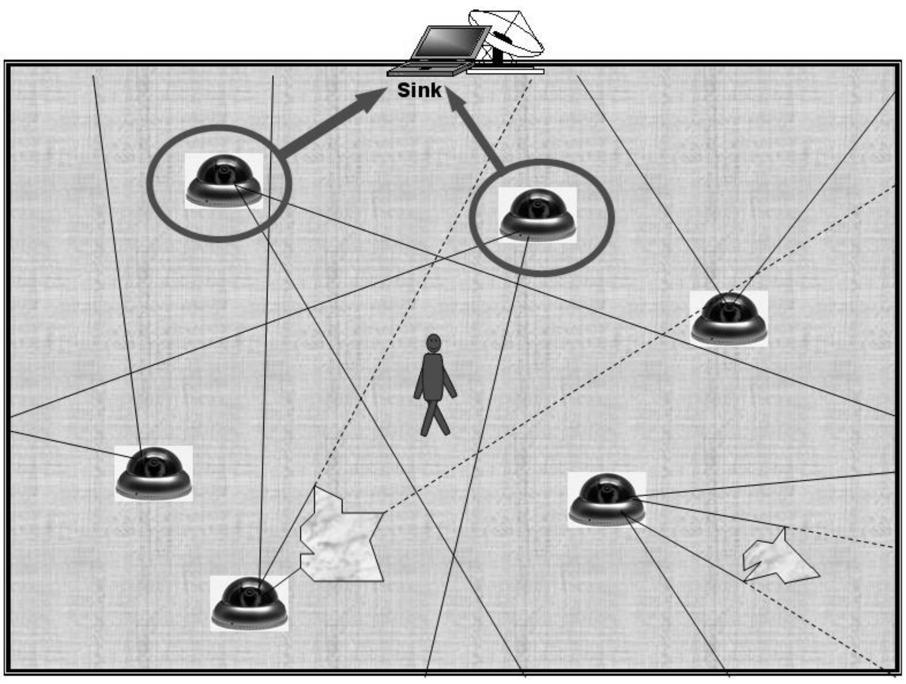


Figure 3.8: An example scenario where nodes at random locations and an intrusion occurs

An example scenario is depicted in Figure 3.8. In the figure we demonstrate an ad-hoc deployment of sensor nodes equipped with cameras represented by their Field of View (FoV) based on the direction of the deployed camera. Even though the FoV of each camera is limited, the distributed deployment of multiple cameras enables perception of the environment from multiple disparate viewpoints. In the figure, we demonstrate an emergency in field marked as the event in the middle of the figure, and nodes detecting the emergency are circled to distinguish them from others. Some obstacles that might block the views of cameras are depicted by the two gray constructions. In the figure, two nodes with a circle are able to observe the event while the other nodes are unable to observe the event even though they are much closer to the event. In this regard, we should not rely on collaboration of neighboring nodes as in traditional sensor networks to reduce the complexity of multimedia data processing.

CHAPTER 4

LIGHTWEIGHT OBJECT LOCALIZATION WITH A SINGLE CAMERA

In this chapter, we describe our proposed protocol for object localization in WMSNs using a single camera. We first define the problem, then give the solution and lastly explore the experiments.

4.1 Object Localization Issues in WMSNs

While object localization has been well-studied in the past [Spors01][Huiyu08][Qi07], this was under the context of traditional multimedia systems where powered video and image cameras are available with enough processing capabilities. The applications included indoor/outdoor building surveillance with the cameras mounted on the walls/ceilings of the buildings. However, with the introduction of WMSNs, we envision surveillance applications in outdoor contexts where the majority of the cameras will be battery operated. These are the applications where the frequency of incidents is too low to employ permanent staffing such as monitoring of critical infrastructures, bridges, water supplies, borders, etc. As the cameras will be battery operated in such setups, energy overhead due to processing and communication becomes a critical issue in addition to the accuracy of object localization. Given that the object localization schemes proposed for traditional multimedia systems in the past required complex signal processing and the involvement of multiple cameras, those approaches cannot be directly applied in our context for resource constrained WMSNs. Therefore, novel approaches for object localization in WMSNs are needed.

One possible and simple solution for object localization in WMSNs is to utilize camera sensor locations and employ triangulation. However, this solution may not be appropriate since it may not always be guaranteed that an object will be detected by more than one camera sensor simultaneously. In addition, this approach will require message exchanging among the participating nodes which may unnecessarily increase the localization time. Note that minimizing the object localization time is also of paramount importance given that the effectiveness of the actions need to be taken depends on the timeliness of object localization. Another solution would be collecting video data at a sink and perform the calculations there to locate the object. This approach not only introduces unnecessary transmission of video frames to the sink but also creates the same concerns regarding timeliness as mentioned above.

Considering the issues of overhead and timeliness, in this chapter we propose a novel lightweight object localization scheme for WMSNs.

4.2 Problem Definition

Our problem for object detection and localization can be formally defined as follows: "Given a video camera sensor which is generating video frames within a WMSN, determine the location of an object detected by such camera with maximum accuracy and minimum overhead in terms of energy and time".

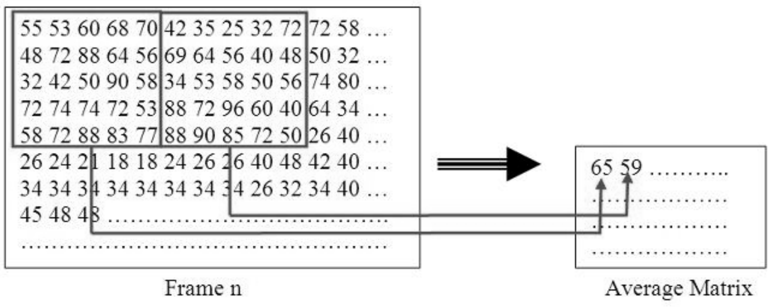
Our solution consists of two steps. First, we describe how the objects are extracted using frame differencing. Second, we determine the object location by calculating its distance from the camera and using camera orientation. Our goal while solving the above problem is to perform calculations at the camera sensors with maximum accuracy and minimum overhead in terms of energy.

4.3 Lightweight Object Localization with a Single Camera

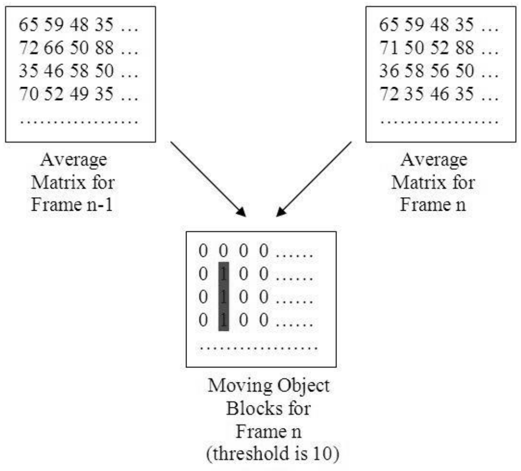
4.3.1 Object Detection

When an unusual event occurs in the monitored area, the consecutive frames of the camera sensor are processed to extract the moving objects. Different moving object detection techniques may be used for this purpose. The simplest technique is frame differencing where moving objects on the frame are found by comparing the current frame with the previous frame, pixel by pixel, according to a given threshold value [Bovik00]. If the change of the intensity at a processed pixel is above a pre-specified threshold, then this pixel is considered as part of the foreground. Although this is a simple method, it may not work well in noisy situations such as an area containing dense tree leaves or branches waving in the wind.

In order to eliminate noisy background, we propose to apply two techniques. First, we reduce the frame size by dividing it into image blocks with a determined size (i.e., 5x5 pixels). Hence we obtain pixel blocks rather than individual pixels to process. Secondly, the average color value for every block is calculated so that the image size can further be reduced and the noise can be eliminated. These steps are shown in Figure 4.1(a).



(a)



(b)

Figure 4.1: Elimination of noise and calculation of the average color values for frame *n*

Block size should be determined according to the dynamic background of the frame and it is dependent to the application area where the camera sensor nodes are used. For instance, if the background has dense and small leaves waving in the wind, i.e., a forest application, the block size may be chosen large to eliminate the noisy background more appropriately, if the background is an open area having no dynamic background, block size may be chosen as small to be more sensitive to the moving object accordingly.

After the creation of average matrices, the moving object's blocks are detected by comparing block averages for each frame n with the block averages of the previous frame $n-1$ according to a defined threshold. If the difference is bigger than the threshold, that block is noted as part of the moving object. This process is shown in Figure 4.1(b). At the end of this step we get a $0/1$ matrix, in which the blocks of moving objects are noted as $1s$ and other parts of the matrix are noted as $0s$. Thus, we can find the moving object of the original image just by using this matrix. For instance, in Figure 4.1(b), the moving object consists of 3 moving blocks, which constitute the frame part having a size of 5x15 pixels in the real frame.

Of course the first possible attempt after detecting the moving object would be sending the current frame to the sink for further process. For instance, the sink may guide the camera sensors to zoom in/out to clearly identifying the moving objects. However, firstly, the biggest gain in our work is the less energy consumption. Secondly, simple cameras do not have zooming properties due to energy/cost concerns. Such cameras may exist (e.g., PTZ cameras) but they require significant power (i.e., cannot work with batteries for long) and are expensive, which makes them infeasible to use in large scale in outdoor contexts. Therefore, identifying the objects in this way is not applicable for the targeted applications using WMSNs. Thirdly, even though we assume such cameras with zooming capability, the decision for sending which frame/frames to the sink is not an easy process. As an example, let us assume that an object passes through the front of the camera in 10 seconds. This means that, in 260 frames (for a video having 26 frames/sec), the object is detected as a moving object. Let us assume that the camera sends the 10^{th} frame to the sink and the sink decides to zoom into the object. However, again let us assume that at frame 50, another object enters to the scene which is far from the first object. Since the camera node has zoomed into the first object, it may miss the second one. If the camera node decided to send the 50^{th} frame to the sink, it would be a better decision. Hence, instead of huge sized raw video frame or raw video data, we perform classification and localization as will be explained in next sections and send only this extracted information to the sink.

4.3.2 Object Localization

Our object localization approach requires the distance between the extracted object and the camera. In order to find the distance between the object and the camera sensor, we use camera properties such as focus distance and sensor dimensions (e.g., the complementary metal oxide semiconductor (CMOS) sensor which is used in many digital cameras today). By using the size of the frame and size of the object on the frame, we can determine the size of the object on the CMOS sensor. We can then use this information to estimate the distance between the camera sensor and the detected object, as shown in Figure 4.2.

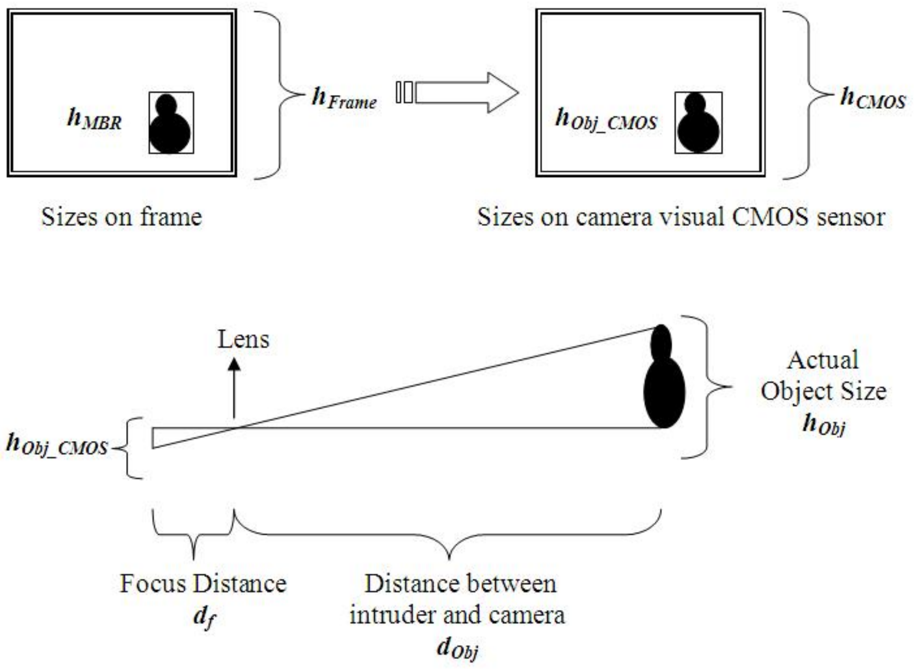

h_{MBR}
h_{Frame}
h_{Obj_CMOS}
h_{CMOS}
Sizes on frame
Sizes on camera visual CMOS sensor
Lens
h_{Obj_CMOS}
Actual Object Size h_{Obj}
Focus Distance d_f
Distance between intruder and camera d_{Obj}

Figure 4.2: Process to find the distance between the intruder and the camera sensor

The relationship between the real object size and the size of the same object on the frame can be determined as explained next, using the notation in Table 4.1.

We would like to note that the estimation of the height of the object is application dependent and it should be chosen accordingly. The type of the object in our border surveillance application is determined in the first step and the height of the object can be estimated assuming average height values for *human*, *animal* and *vehicle* intruders respectively. In our application object height estimation is performed by looking its *Shape_Coefficient* value explained in the

Table 4.1: Notation for Finding the Object Distance

h_{Obj_CMOS}	Height of the object on CMOS sensor
h_{CMOS}	Height of the CMOS sensor
h_{Obj}	Estimated height of the object in reality
h_{MBR}	Height of minimum bounding rectangle for the object
h_{Frame}	Height of the frame
d_f	Focus distance of the CMOS camera
d_{Obj}	Distance between the object and sensor

next section. In order to find the distance between the intruder and the camera sensor, first we need the height of the intruder on the CMOS sensor which can be found using Equation 4.1.

$$h_{Obj\_CMOS} = h_{CMOS} * \frac{h_{MBR}}{h_{Frame}} \tag{4.1}$$

We then use the height of the intruder on the CMOS sensor found in Equation 4.1, to find the distance between the object and camera using Equation 4.2.

$$d_{Obj} = h_{Obj} * \frac{d_f}{h_{Obj\_CMOS}} \tag{4.2}$$

By using only Equation 4.1 and Equation 4.2, we can calculate the raw distances given in Figure 4.3 for each frame. In our experiments, we have observed that, at every frame, since there may be one or more block count changes due to the noise on image, the calculated distances may differ significantly (The reasons for this are discussed in object classification section). Therefore, we average the previous distances in a windowed fashion (i.e., using the past 5 frames) to estimate the current distance of the object by smoothing out the raw distances. Sample results are shown in Figure 4.3 as estimated averaged results for all frames.

Once the distance of the object to the camera sensor is found, the next step is to find the location of the detected object by exploiting this distance. Here, we use the location of the camera sensor with its orientation as shown in Figure 4.4.

In order to find the orientation of the detected object, we use a stripping method on the frame. In this method, we divide the frame into $2k$ strips. (In our implementation, in order to decrease

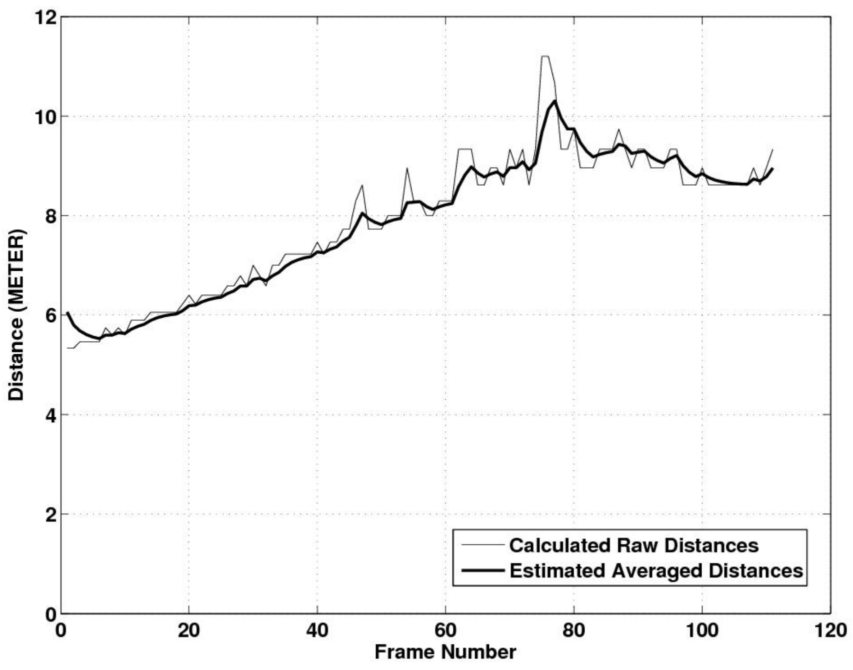


Figure 4.3: The results of calculating the distance of the intruder to the camera in our experiment setup

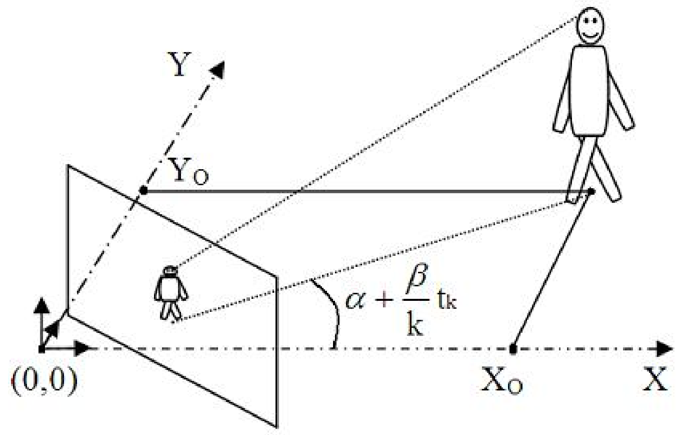


Figure 4.4: Calculating the location of the object

the localization error to the low levels, we use the average matrix's width as the $2k$ value) The middle of the frame shows the same orientation as the camera's middle point. The left and right strips give plus or minus orientation angles for the view. This is shown in Figure 4.5.

We use the following notation in Table 4.2 to describe the computation of object location. Based on the notation, the x and y coordinates of the detected object can be found by using

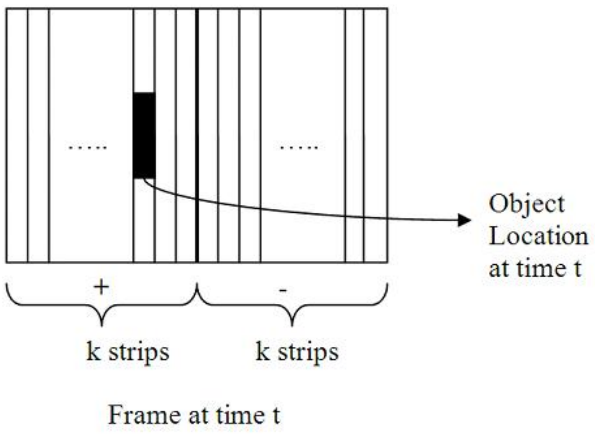


Figure 4.5: Using object location on the video frame

Equation 4.3 and Equation 4.4.

$$x_o(t) = x_s + \cos(\alpha + \frac{\beta}{\kappa} t_k) d_t \tag{4.3}$$

$$y_o(t) = y_s + \sin(\alpha + \frac{\beta}{\kappa} t_k) d_t \tag{4.4}$$

Table 4.2: Notation for Localization

(x_s, y_s)	The Location of the Sensor
α	The Orientation of the Sensor
β	The Field of View Angle (±)
d_t	The Distance of the Object at time t
t_k	The Frame Strip of the Object at time t
$2k$	The Number of Strips at Frame at time t

The method is tested in a real scenario with a frame size of 640 X 480 on a human intruder, as seen in Figure 4.6 where one of the frames's moving object is shown in Figure 4.7 (original video is in color). We have calculated the object location for each frame and produced the human's path. This path is compared with the real object path as seen in Figure 4.8. With small estimation error (i.e., up to 5%) in object height, the approach can provide 98% accuracy in localization.

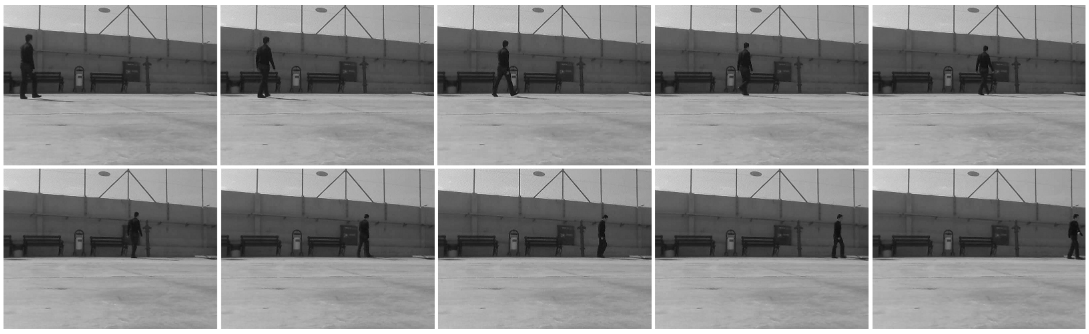


Figure 4.6: Sample frames for moving object detection

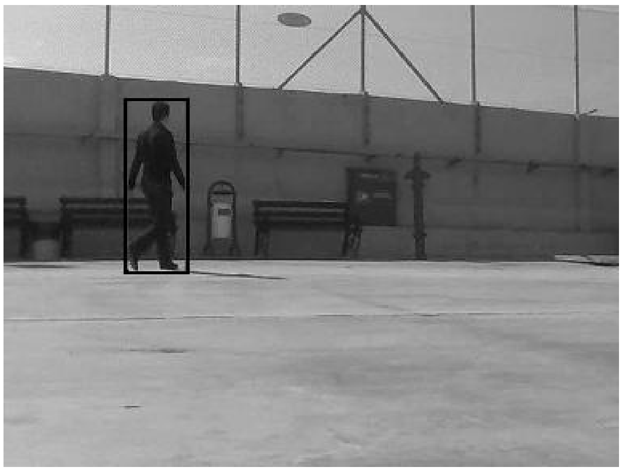


Figure 4.7: Moving object detection for an individual frame (minimum bounding rectangle is shown with black)

4.4 Algorithm Analysis

The object localization algorithm explained in this dissertation works on every frame of the video taken by the individual camera sensor. The main steps of the algorithm are listed as follows:

1. Detect the moving object on the frame,
2. Calculate the distance between the moving object and the camera sensor,
3. Locate the object in 2-D environment.

In order to detect the moving object, first the algorithm runs on every pixel of the frame to reduce the noise and size of the image. The time requires to process the whole image is $O(hw)$

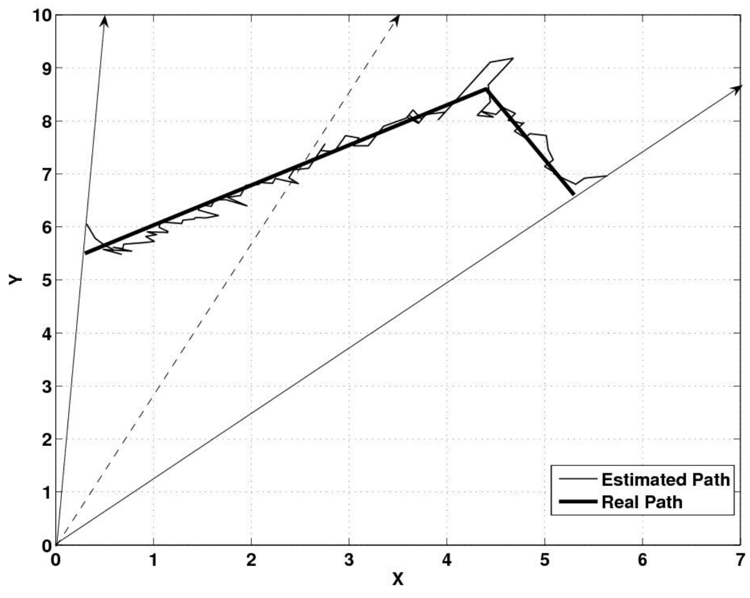


Figure 4.8: Estimated object path and real path for scenario in Figure 4.9. The straight vectors indicate the borders of the FoV and dashed vector indicates the direction of the sensor.

where h is the height and w is the width of the frame in pixels respectively. Then, the reduced frame is used at other stages which is smaller than original frame in size. Since detecting the moving object requires looking at the previous frame and current frame averages matrices, the time needed for this comparison is not greater than $O(hw)$. After detecting the moving object, computations to find the distance between object and sensor require constant time which is $O(1)$. The last step is to locate the object in the environment using the distance found in the previous step. This also requires constant time, $O(1)$. Hence overall computational complexity of the algorithm is $O(hw)$.

Note that in case of doing the processing at the sink, the frames need to be transmitted to the sink. Therefore, there will be extra delay for transmission of one frame over k hops assuming that camera sensor is k-hop away from the sink. Let the transmission and propagation delays for a frame of mentioned size above (i.e., hw) are t_d and t_p respectively, then total communication time complexity is $O(k(t_d+t_p))$. Note that $O(k(t_d+t_p) >> O(hw)$. To determine the object location, the sink would need to process the data. If our approach is used for such processing, there is still a timing cost of $O(hw)$ in addition to the communication cost of $O(k(t_d + t_p))$. Therefore, total time complexity of a centralized approach would be $O(hw + k(t_d + t_p))$. Our approach saves us a time of $O(k(t_d + t_p))$ which is invaluable for critical surveillance applications that depend on the timeliness of the object detection and localization.

Table 4.3: Notation Used in Time Complexity Analysis

h	Height of the processed frame
w	Width of the processed frame
k	The distance between the camera sensor performing the processes and the sink in hops
t_d	Transmission delay for the processed frame
t_p	Propagation delay for the processed frame

4.5 Experimental Evaluation

4.5.1 Experiment Setup and Performance Metrics

In the experiments for object detection and localization, we have captured videos of a mobile intruder in different scenarios and calculated the location of the intruder. These locations are then compared with the real 2-D locations of the intruder. We have used sample videos which have frame size 640 x 480 and every frame is 900 KB. Two sample frames are shown in Figure 4.9.

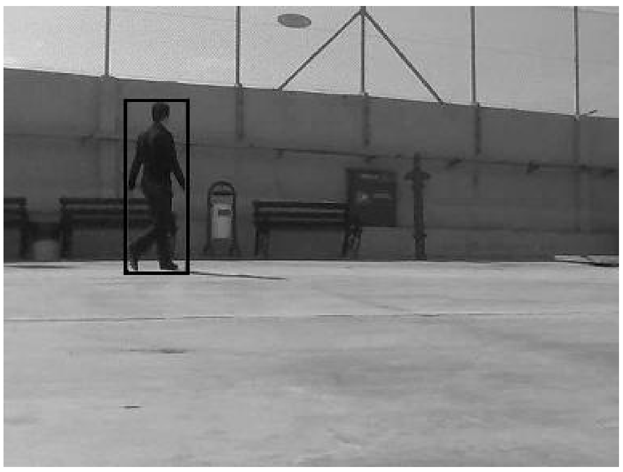
We have considered two metrics:

- Localization Error: This metric shows the error in estimating the distance of the intruder. The less is the error, the better is the quality of the approach.
- Energy Overhead: This metric constitutes total energy in processing and transmitting the frames (if needed). Our goal is to minimize communication overhead in order to maximize the lifetime of the cameras.

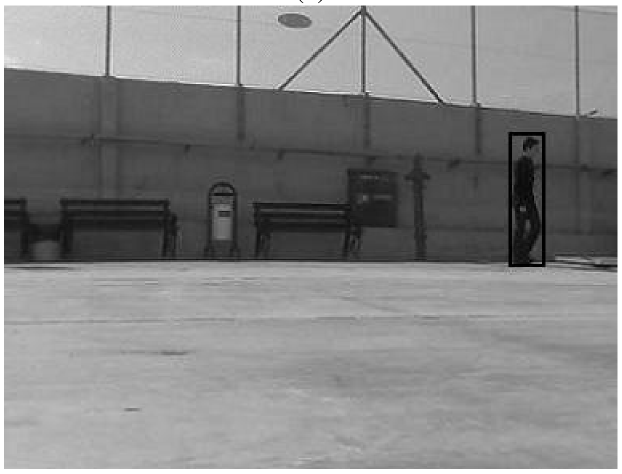
4.5.2 Performance Results

4.5.2.1 Localization Error

We have calculated the detected object location for each frame and produce its path. We have repeated this under different scenarios, where intruder is closer to the camera or vice-versa. These paths are compared with real object paths and the results for them shown in Figure



(a)



(b)

Figure 4.9: Moving object detection for a sample video where MBR of the object is shown on two sample frames for Exp-1

4.10 and 4.11 have revealed that our approach can locate the objects with an error of 2% in the worst case.

4.5.2.2 Energy Overhead

In order to prove the efficiency of the proposed algorithm, we have also performed experiments to assess the energy consumption on the camera sensors. We have used the AVR Simulation and Analysis Framework (AVRORA) [WEB:AVRORA] to calculate the energy costs. AVRORA is an emulator which can provide realistic results as if the approach is run on a typical CMOS sensor. It has built in functions that can compute the processing and communication costs.

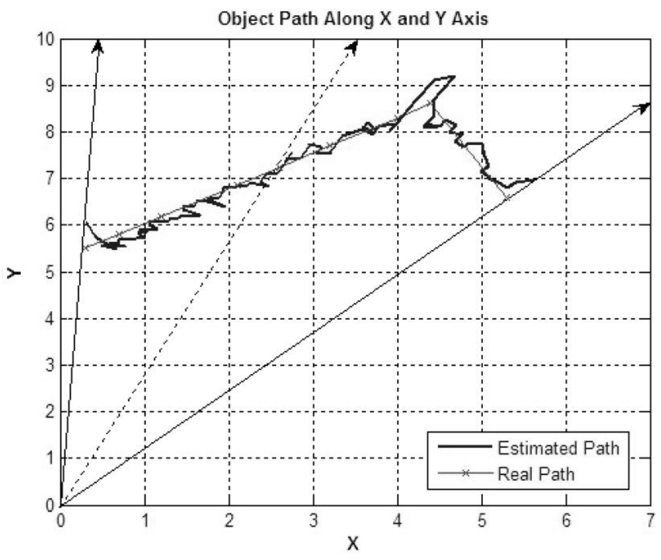


Figure 4.10: Estimated object path and the real path for Exp-1. The straight vectors indicate the borders of the FoV and dashed vector indicates the direction of the camera sensor

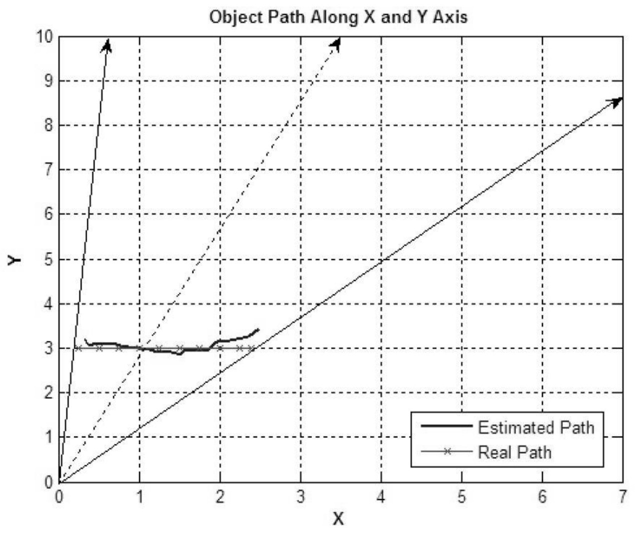


Figure 4.11: Estimated object path and the real path for Exp-2 where the same intruder of Exp-1 passes closer to the camera sensor

We have used a baseline approach which processes the frames at the sink and determines the location of the objects. In that case, the frames are sent to the sink traveling through multiple hops (i.e., *k*). This is referred to as '*Traditional Method*' in the graphs. Our approach performs the localization on site and does not send any data to the sink for localization purposes. However, it may need to send an alarm (i.e., one simple message) to the sink when an intruder is detected and located.

Since AVRORA uses very limited hardware simulation, instead of whole frame, only reduced part (512 byte) is used to approximate the results. The results are given in Table 4.4 and 4.5.

Table 4.4: Energy Costs for Different Tasks

Task	**Cost in Joule**
C: One-time CPU cost to process the frame to extract the moving object	0.0180
M: Transmission cost of the whole frame for 1 hop	0.0700
T: Transmission cost of the alarm for 1 hop	0.0007
Taking the video data	Same for both cases

Table 4.5: Total Energy Costs in Joules

Process	**Traditional Method**	**Proposed Method**
For 1 Hop	$M = 0.0700$	$C + T = 0.0187$
For k Hops	$M * k = 0.0700 * k$	$C + T * k = 0.018 + 0.070 * k$

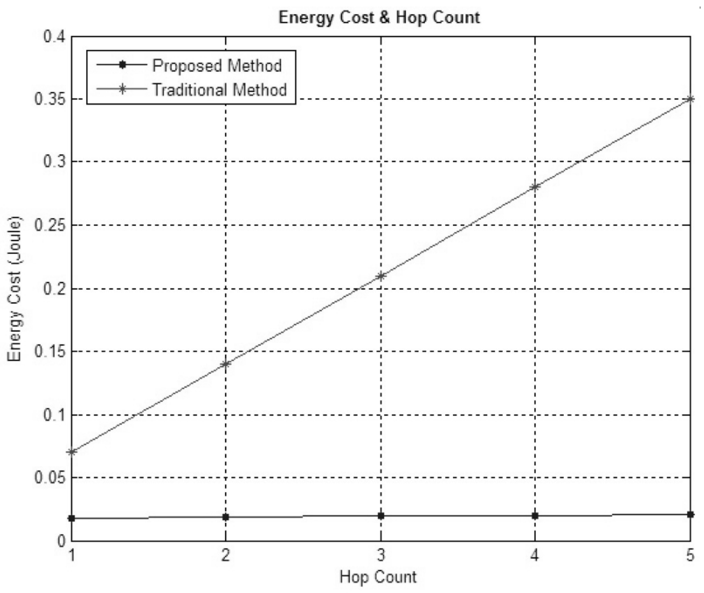


Figure 4.12: Energy costs of two different methods

The results for varying *k* (*Hop Count*) values are depicted in Figure 4.12. As can be seen from this figure, energy overhead for our approach is almost constant and significantly smaller than the traditional method. We would like to note that in this experiment every moving object detection event is sent as an alarm to the sink. However, if we define some alarm criteria for the proposed method, the energy consumption would be further reduced (i.e., alarms are only sent when needed).

4.6 Summary

In this chapter, we present a novel object localization approach which can work on-site at a camera sensor without the need for collaboration with its neighbors. The approach utilizes frame differencing in order to extract the object. Once the object is detected, its distance to the camera sensors is estimated using the size of the object on the frame and on the camera. The distance is then used to determine the exact location of the object by using the orientation of the camera and the object.

The experimental evaluation reveals that our approach can effectively determine the location of an object with an error rate of 2% in the worst case. We also assess the energy overhead of our approach on the individual camera sensors. The energy consumption is significantly reduced compared to the cases where the localization is performed at the sink due to communication overhead.

CHAPTER 5

EFFICIENT AND ACCURATE OBJECT CLASSIFICATION

In this chapter, we explain how we solve the object classification problem in WMSNs. We give information about feature selection process and the classifier we use. At the end, we present the experiments in detail.

5.1 Object Classification Issues in WMSNs

Current WMSN research has focused on the issues regarding energy-efficient video data routing, quality of service (QoS) and intelligent camera actuation for increased network lifetime, which are mostly related to networking aspects of the application [Akyildiz07][Soro09]. However, WMSNs are multi-camera systems and they also deal with traditional problems of image/video processing such as object classification. Such processing is associated with networking issues as it may provide in-network processing to improve the network lifetime and reduce the bandwidth requirements. For this purpose, in this chapter, we propose an efficient and accurate object classification solution for WMSNs.

5.2 Problem Definition

Our problem for object classification can be formally defined as follows: "Given a video camera sensor which is generating video frames within a WMSN,determine the class of a detected object by such a camera with maximum accuracy and minimum overhead in terms of energy, time and storage". Our goal is to minimize the need for human intervention on the WMSN applications with automated and accurate object classification.

Our goal while solving the above problem is to perform the calculations at the camera sensors with maximum accuracy and minimum overhead in terms of energy, time and storage to achieve high data reduction rate. It is desirable to reduce the amount of data that needs to be transmitted such that the amount of energy during data communications is kept at a minimal level. This also helps controlling the data traffic to reduce interference problems and data collision.

5.3 Efficient and Accurate Object Classification

In our border surveillance application, we assume that camera sensors may sense three different object types, namely *Human*, *Vehicle* and *Animal*. In the literature, there are many works using the shape and the motion properties of moving objects in order to classify them into one of these categories (human, animal, vehicle), such as [Rivlin02][Bogomolov03][Javed06][Landabaso04]. However, all of these works use powerful cameras or machines that do not have any energy constraints. Given the resource constraints of camera sensors, we opt to send as minimal data as we can to the sink. Hence we take from these works the idea of using the features of the moving objects' shape and motion (velocity) in order to classify them. We implement this idea in a simpler way to address the energy and processing problems in WMSNs.

The event associated with the object and sent by camera sensors to the sink is named the *Moving Object Event (MOE)*. Since, instead of the complete image or video of the object and only MOEs flow to the sink, there is no way of performing object identification on raw video data using powerful image processing techniques to differentiate between different objects at the sink. We therefore propose to utilize a simple object classification algorithm at the camera sensors in order to be able to identify and track these objects at the sink. The classification algorithm is fuzzy logic based and does not provide a crisp classification. It depends on very limited information since it is performed under the energy and processing constraints. The motivation behind using fuzzy logic is based on the fact that, even though some processing is allowed at the camera sensors, a single camera may not provide enough information to recognize and classify an object accurately. Hence, fuzzy logic is utilized in order to incorporate the uncertainty in the decision of classification at the camera sensors.

The classification algorithm works as follows: In order to detect the moving objects in the monitored area, we extract the *Minimum Bounding Rectangle (MBR)* of the moving object in the captured frame. We realize that MBR behaviors of different types of objects differ from each other. For instance, a moving human intruder shows somewhat slim-tall MBR as shown in Figure 5.1 (with a black box). On the other hand, an animal shows a wider but mostly thicker MBR. A vehicle shows similar MBR behavior to that of an animal however they differ in their speeds throughout the video frames. While it is true that some of the animals can move relatively fast (e.g., horse, dogs, cats), this is only possible when they are escaping or hunting at their maximum speeds. Assuming a typical walking pace, most of the time, an animal will be slower than a typical vehicle. For the extreme cases, we introduce possible errors in classification which will be tested in Chapter 5.5 (Experimental Evaluation).

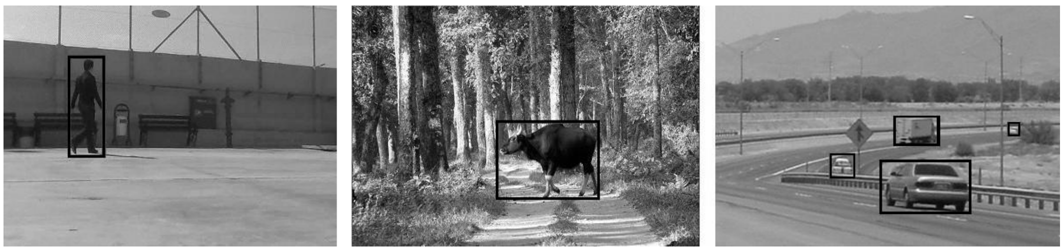


Figure 5.1: A human intruder, an animal and some vehicles with their minimum bounding rectangles (MBRs), shown in black

For every detected moving object, a *Shape_Coefficient* is calculated by using Equation 5.1. The membership values of the objects for the three object classes are found by using the example fuzzy sets shown in Figure 5.2. We use trapezoidal and triangular membership functions for fuzzy input variable *Shape_Coefficient*. The example fuzzy sets formed by this membership function and corresponding linguistic states are shown in Figure 5.2. Note that this *Shape_Coefficient* membership figure is for our specific border surveillance application and is given as an example. A more specific and accurate fuzzy set can be created and adapted based on the specifics of the applications.

$$Shape\_Coefficient = \frac{width\_MBR}{height\_MBR} \tag{5.1}$$

As an example, the *Shape_Coefficient* for the human's MBR in Figure 5.1 is 0.36 (width: 40 pixels, height: 110 pixels). This means that this MBR has the classification vector of {Human:1.0, Vehicle:0.0, Animal:0.0}. On the other hand, the *Shape_Coefficient* for the animal's MBR shown in Figure 5.1 is 1.35 (width: 230 pixels, height: 170 pixels). This means that this MBR has the classification vector of {Human:0, Vehicle:0.25, Animal:0.75}.

Since animals and vehicles have similar MBRs and consequently similar *Shape_Coefficient*s, the animals and the vehicles can also be separated from each other with the aid of their velocities when they are moving. Therefore, we add one more step to our classification algorithm to use the velocities of the detected moving objects. The velocity of the objects can be determined by finding two consecutive locations (which will be explained shortly) and by looking at the time it takes to travel between those two locations.

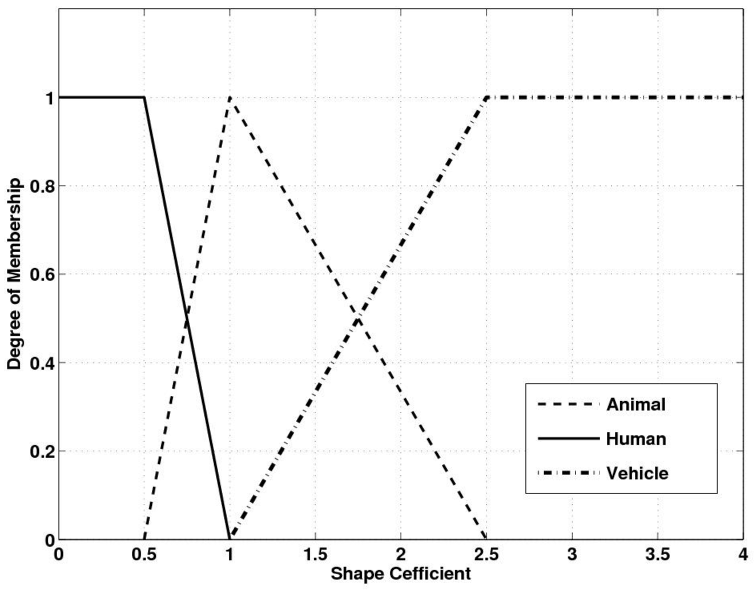


Figure 5.2: Example fuzzy sets produced for our application used in simple object classification according to Shape Coefficient

We would like to note that there may be some situations that affect the accuracy of the computation of the *Shape_Coefficient*. For instance, an object may be seen in the frame partially, for following reasons:

1. The object may disappear partially due to an obstacle and thus only some part of it may be visible;
2. The object may change its shape naturally while moving (i.e., if it is a human being or animal);
3. There may be a shadow of the object;

4. The algorithm may not capture the whole moving object at all frames.

Therefore, to prevent inconsistencies in the coefficient, we consider the results of both the current frame and its previous frames. We average the previous coefficient in a windowed fashion (i.e., using the past 5 frames) to estimate the coefficient of the object more accurately.

5.3.1 Feature Selection & Extraction for Genetic Algorithm Based Classifier

In visual object classification task, extracting features from images and performing classification with these features are usually complex and costly operations. In order to provide a lightweight classification mechanism in terms of processing and energy consumption, it is crucial to adapt an energy-efficient and effective solution for the choice of features and the classification process. Therefore, in this dissertation, we propose to use two simple visual features that are related to the objects rather than the whole image: *Shape_Ratio* and *Speed*. The *Shape_Ratio* feature is the ratio of width and height of the detected object's minimum bounding rectangle (MBR), whereas *Speed* feature gives the speed of the object. Note that while these features are easy to extract, the classification using such limited features (i.e., in terms of the number and the characteristics of features) will be challenging. We will address this challenge when we employ a suitable classifier as will be detailed in the next subsection.

Shape_Ratio can be easily obtained from a detected object. To do this, first the object is extracted from the video frames using frame differencing at individual camera sensors. MBR on the image can then be easily calculated. This process does not require a lot of video processing as it only depends on the frame differencing which was shown to be energy-efficient in [Oztarak09]. MBR has the height and width of the object which are used to calculate $Shape\_Ratio = width/height$

The speed can also be calculated based on some pre-processing using camera properties and camera locations. Specifically, once the object is extracted, the distance between the object and the camera sensor is calculated. The object location is then calculated by using the camera's location and direction, the object position on the frame and the distance between the object and the camera. At the end, previous frame's result for localization is used to calculate the speed of the object in the area under surveillance. This process has also been shown to be feasible and cost-efficient in [Oztarak09] for object tracking applications.

5.3.2 Object Classification Using Genetic Algorithm Based Classifier

In this dissertation, an instance-based classification mechanism is adopted. The instance-based mechanism requires some prototype images for each class as the classifier model and makes a decision by using the dissimilarities of the prototypes with the query object. To avoid the large space requirements of such a classifier model (i.e., it can be used on resource constrained cameras), a pre-defined number of prototypes are selected by using a Genetic Algorithm (GA) based approach [Yilmaz11] on the training data. In addition to the GA support, the classifier utilizes Class Specific Feature (CSF) approach [Yilmaz11-2] in order to define prototype objects with the most representative and discriminative features.

The classifier model is constructed offline and can be deployed to the camera sensors to be used in classification process. Note that when a new and improved classifier is available, this classifier can also be deployed to the camera sensors to replace the existing classifier.

Before starting the actual classification, a training phase is performed to construct the classifier model. The goal in this phase is to train the algorithm by using a set of prototype images. In order to do this, two things need to be performed: 1) Objects features that best represent each class need to be identified by using CSF indices; and 2) A genetic algorithm (GA) should be used to classify candidate prototype objects from all images available. Thus, the classifier model contains two major parts: CSF Model and GA Model. Note that the object features used here are different than the features that will be used in the classification of the objects after the training is done.

As introduced in [Yilmaz11-2], the CSF model is built on the idea that different object types can be represented better by different visual features. For example, a ‘car’ object can be represented better by its ‘shape’, whereas a ‘sea’ object can be represented with its ‘color’. In this study, the class-specific feature selection mechanism, which is proposed in [Yilmaz11-2], is utilized. The method finds out the representative and discriminative features for each image class. The representative characteristics of features are calculated according to the dissimilarities of images within the same class, and discriminative characteristics are calculated according to the dissimilarities of images between different image classes. Using these representative and discriminative characteristics, the weight values of features for each image class are calculated. These weight values are referred to as CSF indices and are used during the decision making in the classification process.

GA model contains the prototype images for each class. These prototype images are obtained by using the GA based approach in [Yilmaz11]. Based on the GA notations, each class is represented with a set of chromosomes (prototypes) which are representative images of the class. Each chromosome holds an *Effectiveness Value* that is used as a weight during the decision making. In addition, each chromosome is represented with a set of genes which are the representative features for that class in correspondence with the CSF model.

To utilize a GA based approach and perform genetic operations, the training phase is divided into two parts: one training phase for obtaining the definitions of the training objects (First-Training) and one for making improvements on the definitions by using genetic operations (Second-Training). The initial population of genetic algorithm is generated by using the First-Training data. In other words, the classes in the classifier model gains chromosomes by identifying new objects in the First-Training phase. Then, the genetic operations are applied to the system during Second-Training. These genetic operations are: *Effectiveness Correction*, *Crossover* and *Mutation*. By applying these operations, "survival of the fittest" principle of genetic algorithm is provided and the fittest chromosomes (prototypes) are selected as the classifier model.

For each image obtained during the Second-Training, firstly the decisions of all categories are calculated. Also, fitness values of all chromosomes on all categories are computed by using the *Fitness Function*. Then, *Effectiveness Correction*, *Crossover* and *Mutation* operations are performed on the chromosomes of all categories. The *Fitness Function* is used to understand how much a given decision is fit according to the ground truth. The *Effectiveness Correction* operation updates the *Effectiveness Value*'s of the chromosomes according to their fitness. The *Crossover* operation mates the currently available chromosomes, combines the information they contain and tries to obtain fitter chromosomes. Similarly, the *Mutation* operation mates the currently available chromosomes with the newly obtained chromosomes during Second-Training.

Our proposed classification method performs multiple labeling on each object by providing a fuzzy membership decision for each class in the range [0,1]. The classifier makes a decision on the fuzzy membership of a candidate object for each of the contained classes. During such calculation, decision of a gene is based on the similarity of the query object with the corresponding prototype object to which the feature belongs. Decisions of genes in a chromosome are combined by applying a weighted sum where the weights are the CSF indices for the decision-giving class. Decisions of chromosomes are also combined by using a weighted sum in order to obtain the decision of the class. In such calculation, the weights of the chromosomes are the effectiveness values of chromosomes that are updated during the genetic operations according to the fitness of the chromosomes.

Our proposed classification method performs multiple labeling on a query object by providing a fuzzy membership decision for each class in the range [0,1]. For instance, if there are two classes as human and animal, an object's fuzzy membership value can be [0.7,0.3] which indicates that the object is classified as a human with 0.7 fuzzy membership value and as an animal with 0.3 fuzzy membership value. In order to obtain the membership value of each class, decisions of chromosomes that are contained in that class are combined by using a weighted sum. The weights of the chromosomes are the effectiveness values of chromosomes that are updated during the genetic operations according to the fitness of the chromosomes. The decision of each chromosome is also found by a weighted sum. To obtain a chromosome decision, decisions of genes in the chromosome are combined. This time, the weights are the CSF indices for the decision-giving class. Lastly, decision of each gene is calculated by using the similarity of the query object with the corresponding prototype object to which the feature belongs.

As can be seen from the explanations above, the calculation of the fuzzy membership value does not bring a significant processing overhead to the camera sensors as the calculations are mostly finding the weighted sums.

5.4 Algorithm Analysis

The steps that are followed before the actual classification algorithm is run are listed below. These are the steps to extract the features that we use in the classification of the object.

1. Detect moving object on the frame
2. Calculate the distance between moving object and camera sensor
3. Locate and calculate the speed of the object

We can assume that the features that we are using in the classification will be available as part of object detection and tracking in surveillance. Therefore, we focus on the analysis of the classification algorithm only in this section. We analyze the time, energy and storage complexity of the algorithm to see its applicability to WMSNs.

5.4.1 Time Complexity Analysis

The object classification process requires calculating the similarity between the query object and each of the prototype objects in the classifier model for each feature. Considering that preferred features are one-dimensional, performing an Euclidian-distance calculation in order to find the similarities requires a constant time, which is $O(1)$. Thus, assuming that m and n denote the feature count and the total number of prototypes respectively, the total classification process is performed in $O(mn)$ time.

Note that m is the feature count that cannot increase dramatically (e.g., it is 2 in our case). As a result, n is a more dominant value than m. Hence, we can simplify the complexity to $O(n)$ which is linearly dependent to n.

Table 5.1: Notation Used in Time Complexity Analysis

m	The feature count used in classification
n	The total number of prototypes used in classification

5.4.2 Energy Analysis

The energy consumption of classification algorithm depends on calculations made on each video frame for the classification process. In our algorithm, in order to calculate the similarity between the query object and each of the prototype objects in the classifier model, we perform Euclidian-distance calculation for each feature and it requires $O(mn)$ simple subtractions where m refers the feature count and n refers to number of prototypes. In order to calculate the total weight for the query object, we perform $O(mn)$ additions and multiplications. Then, for each class, we perform $O(1)$ division to find decision value for the prototype, $O(1)$ addition to find effectiveness for the prototype, and $O(1)$ multiplication and addition to find the total sum for the prototype. Lastly we perform $O(1)$ more division to find the decision of the class for the query object. Hence, if we have c classes, totally we perform $O(cmn)$ subtractions, $O(cmn + c)$ additions, $O(cmn + c)$ multiplications and $O(c)$ divisions. Since typically c is small value, total operation count is linearly dependent on mn. Therefore, total energy cost at each camera sensor for data processing would be the cost of performing $O(mn)$ multiplications, additions and subtractions.

Note that the classification can also be done at the sink. In that case, if we assume that we run the same algorithm, then the energy cost will be same but there will be increased delay for classification due to processing at the sink. On the other hand, if a different classification algorithm is to be run at the sink, then this requires the transmission of the detected objects as a whole (rather than the MBRs) which will introduce a significantly higher energy cost compared to our case. This is because, the size of the transmitted object/frame will be much higher than the MBRs which are in text. Meanwhile, the extraction of the objects will be as costly as our frame differencing.

Table 5.2: Notation Used in Energy Analysis for Object Classification

m	The feature count used in classification
n	The total number of prototypes used in classification
c	The number of classes that the objects will be classified

5.4.3 Storage Analysis

The classifier model contains two models: CSF model and the genetic algorithm model. CSF model includes the weights of each feature for each class. Thus, the complexity of the function for calculating the bytes needed for storage of CSF is $O(mc)$, where m and c denotes the number of features and classes respectively. Besides, genetic algorithm model holds features of each prototype and weights of the prototypes. As a result, the complexity of the storage required for the genetic algorithm model is $O(n(m + 1))$.

The overall complexity of the storage needed to store the classifier is $O(mc + n(m + 1))$ which is linearly dependent on the number of features, classes and prototypes. Since n is larger compared to c and m values (i.e., generally c and m are constants), we can simplify the complexity to $O(n)$. If we emphasize having limited storage on camera nodes, linear dependency is the best we can achieve for that purpose.

Table 5.3: Notation used in Storage Analysis for Object Classification

m	The feature count used in classification
n	The total number of prototypes used in classification
c	The number of classes that the objects will be classified

5.5 Experimental Evaluation

5.5.1 Experiment Setup and Performance Metrics

For the experiments of object classification, we assume a power plant surveillance application scenario. In this scenario, when an intrusion occurs in the area under surveillance, the detected objects are classified at the camera sensors. The classification is performed as a multi-class choice with 3 classes: *Human*, *Vehicle* and *Animal*. For the camera sensor experiment data, the Caltech 101 image dataset [Fei06] (for *Vehicle* and *Animal* classes) and search results from Google Image Search (for *Human* class) are used by formatting them into the CmuCam3 [Rowe07] output format. The CalTech101 dataset does not contain *Vehicle* and *Animal* classes. Therefore, images from several different classes in Caltech 101 dataset are re-grouped according to these classes. The dataset is divided into three sets: First-Training, Second-Training and Test. The number of images is determined as 10 for each class in each of the training sets and 20 for each class in the test set. Sample images from our constructed dataset are given in Figure 5.3.

The $Shape\_Ratio$ feature of each sample is extracted by simply dividing the width of the image by the height value. Besides, $Speed$ values are randomly generated, considering that it is not possible to have speed values in an image dataset. For the random generation, speed values between $[1, 10]$, $[5, 25]$ and $[10, 100]$ meter/sec are used for classes *Human*, *Animal*, *Vehicle*, respectively.

As mentioned in Chapter 5.3.2, the CSF mechanism [Yilmaz11-2] is applied in order to

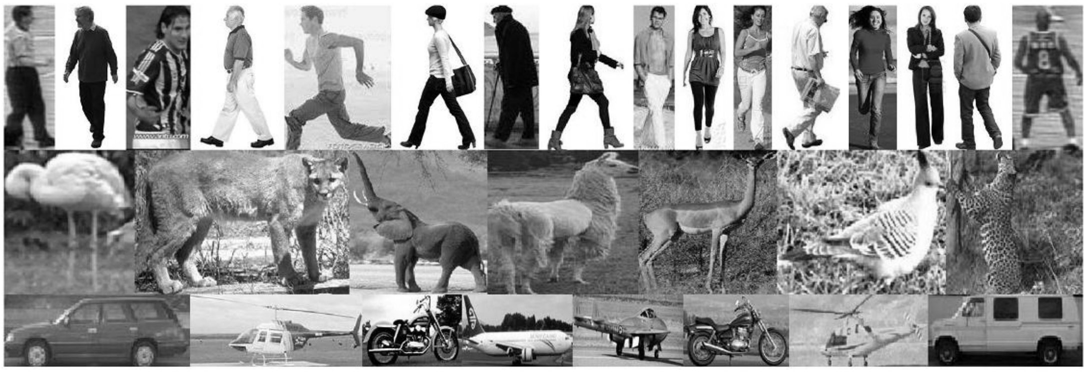


Figure 5.3: Sample images from test dataset

find representative and discriminative features for each object class. CSF mechanism gives weights of each feature for each class. Acquired weights are given in Table 5.4. According to these weights, it has been observed that *Speed* is the dominant feature for all classes. However, the effect of it is more for *Animal* class than the other two classes.

Table 5.4: CSF Weights

	$Shape\_Ratio$	$Speed$
$Human$	0.371051	0.628949
$Vehicle$	0.342217	0.657783
$Animal$	0.130904	0.869096

We have considered three metrics to assess the performance of the classification:

- *Classification Accuracy*: This metric shows the accuracy in estimating the class of the intruder. The bigger is the performance, the better is the quality of the approach.

- *Energy Overhead*: This metric constitutes total energy in processing and transmitting the frames (if needed). Our goal is to minimize this overhead in order to maximize the lifetime of the cameras.

- *Occupied Space*: This metric shows the required space for classifier model at the camera sensor. The occupied space should be minimum, considering that the sensor has a limited memory.

5.5.2 Performance Results

5.5.2.1 Classification Accuracy

Under given test setup, the classification results in Table 5.5 and 5.6 are obtained. As mentioned in Chapter 5.3, the classifier performs multiple labeling by providing fuzzy membership values in the range [0,1] for each class. Thus, in order to measure the precision values, the class with the highest membership value is taken as the classification result. The system performs a high accuracy ratio in which only a total of 1 instance is classified wrong among the 60 test instances. This 1 instance is a *Vehicle* instance and classified as an *Animal* due to its very low *Speed* value. As seen in Table 5.4, *Speed* is a more effective feature for *Animal* class. Thus, having *Speed* value very close to that of the *Animal* class caused such a classification.

Table 5.5: Confusion Matrix for GA-based Classification

		Prediction		
		Human	*Vehicle*	*Animal*
Actual	*Human*	20	0	0
	Vehicle	0	19	1
	Animal	0	0	20

Table 5.6: Class Accuracies for GA-based Classification

Class	**Accuracy**
Human	1.00
Vehicle	0.95
Animal	1.00
Total	**0.98**

In addition, we compared our classification results with a baseline approach which uses a fuzzy membership set defined by an expert user to do the classification. The results shown in Table 5.7 and 5.8 indicate that overall classification accuracy is increased to 0.98 by using genetic algorithm based classifier from 0.92, where an expert user prepares fuzzy membership functions to be used in classification process. In addition, the whole classification process is automated and human intervention is removed.

Finally, we repeated the same experiments by introducing a noise of 20% to the randomly-generated $Shape\_Ratio$ & $Speed$ values that may happen due to errors in localization or obstacles in the environment. We observe that the results do not change as the proposed approach can handle such errors perfectly.

Table 5.7: Confusion Matrix for Fuzzy-membership-based Classification

		Prediction		
		Human	*Vehicle*	*Animal*
Actual	*Human*	20	0	0
	Vehicle	0	19	1
	Animal	1	3	16

Table 5.8: Class Accuracies for Fuzzy-membership-based Classification

Class	**Accuracy**
Human	1.00
Vehicle	0.95
Animal	0.8
Total	**0.92**

5.5.2.2 Energy Overhead

We also performed experiments to assess the energy consumption of our approach at camera sensors. We used the AVR Simulation and Analysis Framework (AVRORA) to calculate energy costs [WEB:AVRORA] at camera sensors. We used a baseline approach which processes the frames at a sink and determines the location and classification of the objects at the sink. In that case, the frames travel through multiple hops (i.e., k) to reach the base station. This approach is referred to as '*Traditional Method*' in the graphs. Note that our approach performs the localization and classification on site and does not send any data to the sink. It only sends alarms whenever needed. The results are given in Table 5.9 and 5.10.

The results for varying k (*Hop Count*) values are depicted in Figure 5.4. As can be seen from this figure, energy overhead for our approach is constant and significantly less than the traditional method. We would like to note that in this experiment every moving object detection event is sent as an alarm to the sink. If we define some alarm criteria for the proposed

Table 5.9: Energy Costs for Different Tasks in Both Approaches

Task	**Cost in Joule**
C: One-time CPU cost to process the frame to extract and classify the moving object (our Approach)	0.0220
M: Transmission cost of the whole frame for 1 hop (traditional approach)	0.0700
T: Transmission cost of the alarm for 1 hop (our approach)	0.0007
Taking the video data	Same for both cases

Table 5.10: Total Energy Cost Comparison of Both Approaches (in Joules)

Process	**Traditional Method**	**Proposed Method**
For 1 Hop	$M = 0.0700$	$C + T = 0.0227$
For k Hops	$M * k = 0.0700 * k$	$C + T * k = 0.022 + 0.0007 * k$

method, the energy consumption would be further reduced (i.e., alarms are only sent when needed).

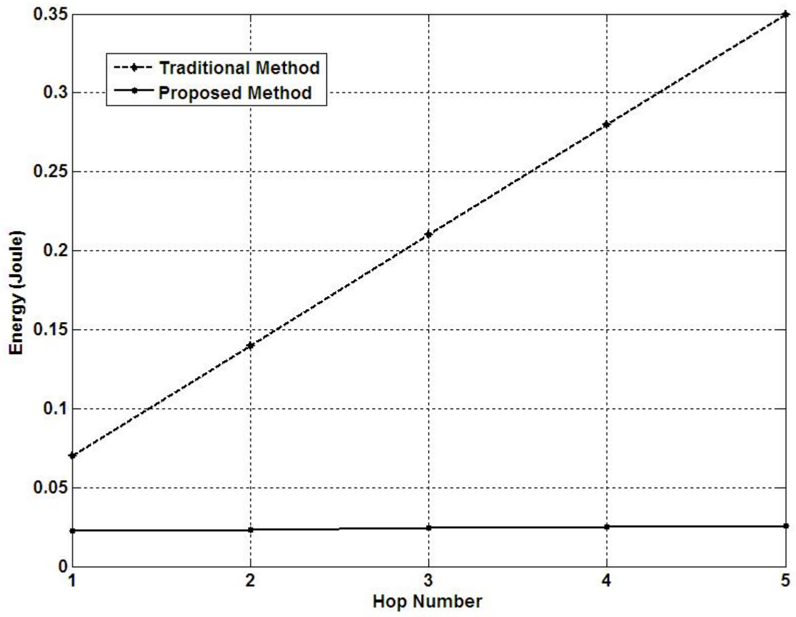


Figure 5.4: Energy costs of two different methods

5.5.2.3 Occupied Space on Camera Sensors

As mentioned in Chapter 5.3, the camera sensor holds two types of models for classifier. The actual required spaces for each of these in our experiments are given in Table 5.11. As can be seen in Table 5.11, the space requirements are not significant compared to the available space on CMUCam3 (e.g., 128MB).

Table 5.11: Space Occupied by Classifier Models on the Sensor

Model	**Occupied Space**
CSF Model	$24 Bytes$
GA Model	$360 Bytes$

5.6 Summary

In this chapter, we present a lightweight and accurate object classification approach which can work on-site at a camera sensor with limited information. The approach utilizes genetic algorithm to built classifiers on top of two simple but effective features, which are the $Shape\_Ratio$ & $Speed$ of the detected objects.

The experimental evaluation reveals that our approach can effectively classify typical objects, i.e., human, animal and vehicles, in a power-plant surveillance application with an error rate of at most 2% overall. We also assess the energy overhead of our approach on the individual camera sensors. The energy consumption is significantly reduced compared to the cases where the classification is performed at the sink due to reduction in the communication energy cost.

CHAPTER 6

EFFICIENT OBJECT TRACKING

In this chapter, we investigate the object tracking problem in WMSNs. We explain the algorithms that we propose in detail. At the end, we present the experiments and their results.

6.1 Object Tracking Issues in WMSNs

Object tracking is one of the major problems in surveillance applications using WMSNs. In this context, “the object tracking” term does not mean “tracking the objects in the video that the camera sensors take”; it means “tracking the object at the area under surveillance”. For instance, the detected object in the area under surveillance should be tracked efficiently to take precautions against it, i.e., for the purpose of catching it in an appropriate place. In addition, if there are multiple objects in the area, accurate tracking of each object is needed not to miss any events associated with them. For these reasons, in this chapter, we propose an efficient object tracking solution for WMSNs.

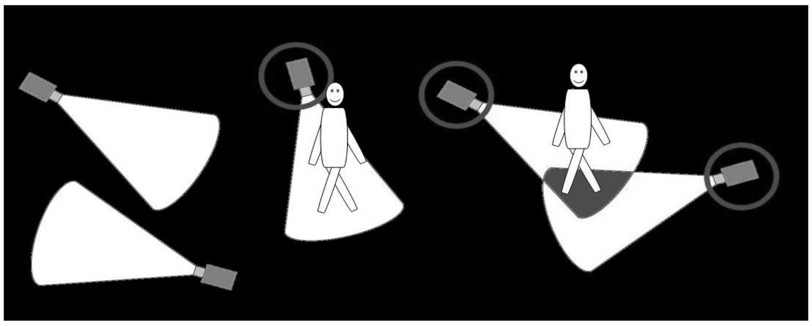
6.2 Problem Definition

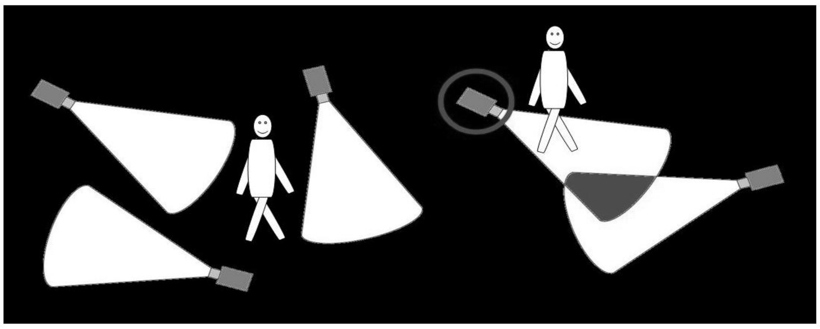
Multimedia delivery from highly resourceful sources, e.g., multimedia servers, to resource-constrained destinations, e.g., wireless handheld devices, has been studied extensively. However, in the context of wireless multimedia sensor networks, to store, process in real time, correlate and fuse multimedia data originated from different sources [Aksoy05] constitutes a major challenge. In a WMSN application with numerous blind and intersected areas, tracking multiple moving objects at the sink becomes a challenging process. We consider the following scenarios to better motivate and explain these challenges (The scenario is also demonstrated in Figure 6.1):

- Assume that at a specific time t_1, the sink receives the information of M different moving objects from different camera sensors, i.e. as shown in Figure 6.1(a) where $M : 3$. (The camera sensors that sense the moving objects are circled in Figure 6.1(a)). However, in fact, X different objects may exist in that area where $M \geq X$, i.e. as shown in Figure 6.1(a) where $X : 2$. This may happen, because more than one camera sensor may detect the same object due to the overlapping FoVs (intersected areas) and transmit different moving object information to the sink.
- At another time t_2, where $t_2 > t_1$, the sink receives N different moving objects, where $N < M$, i.e. as shown in Figure 6.1(b) where $N : 1$. This may happen because the some of the objects detected at t_1 may disappear by exiting from the FoVs and entering into the blind area.
- Finally, at another time t_3, where $t_3 > t_2$, the sink again receives M different moving objects. In this case, it is possible that the objects previously entered into the blind area may appear again in another camera sensor's FoVs or new objects different than previously sensed ones may enter the scene which means newly sensed objects at time t_3 may not be same objects with the sensed objects previously.

The challenges in this complex scenario necessitate the aid of an identification algorithm running at the sink to track the objects accurately. First of all, since the raw video data has a very large size and transmission of this data consumes lots of energy, data reduction techniques at the camera sensors and their consequences on identification and tracking at the sink should be investigated. Secondly, the moving objects and the corresponding events produced by different camera sensors must be processed spatially and temporally to match the received objects.

We should note that, because of intersected areas, M moving objects may be captured by K camera sensors where $M \leq K \leq N$. Therefore, data redundancy at the sink node is expected. To eliminate the data redundancy, the sink should be able to identify redundant gathered objects. However, limited data will be available at the sink, since camera sensors send only limited multimedia information due to resource constraints. Therefore, uncertainty is expected


(a) At time t_1


(b) At time t_2

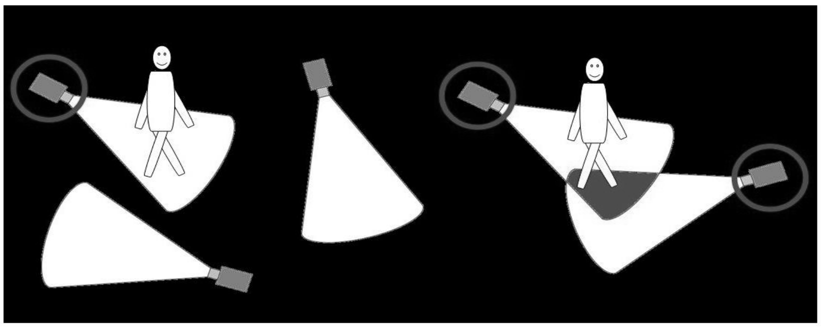

(c) At time t_3

Figure 6.1: An example tracking scenario where the camera sensor nodes that sense the intruders are shown with a circle around them

to be produced from multiple camera sensors sending similar data about the same objects.

Our problem for object tracking can be formally defined as follows: "Assume that we have N wireless camera sensors distributed randomly in a border surveillance application. At a specific time T, there exist M different moving objects (human intruders, vehicles or animals) in the area under surveillance. We would like to localize and track these M objects in real-time with minimized energy burden on camera sensors."

- The first stage is object localization and fuzzy classification in which moving objects (e.g., an intruder) are detected, localized and classified via fuzzy logic at the individual camera sensors. Instead of huge raw video data, only the location and classification information are relayed to the sink.

- The second stage is object identification/re-identification and tracking, in which the information collected from different camera sensors (via single or multi-hop communication), is processed to identify and track the real objects at the sink.

Our goal while solving the above problem is send only reduced and enough information from the camera sensors to the sink to reduce the transmission costs. As mentioned in the previous sections, energy is not the only constraint in camera sensor nodes. These nodes also have limited processing capabilities and therefore it is important to have efficient preprocessing while deciding on how to reduce the amount of data that needs to be transmitted.

6.3 Information Sent from Camera Sensors to the Sink

Once the location and classification of the object is computed, the MOE information given below is sent to the sink by the camera sensors. Note that, since the camera sensors do not know anything about the identities of the detected objects, no identity information is sent to the sink. It is the sink's responsibility to identify and track the detected objects by using the flowing MOEs from different camera sensors as explained below.

The following information is contained in the MOE:

- Sensor_ID of the sensor detecting the event
- Detection time
- Object
  - Location (X and Y coordinate)
  - Velocity of the moving object
  - Membership value of the object
    - for the class 'Human'
    - for the class 'Vehicle'

- ∗ for the class ‘Animal’

In Figure 6.2, we demonstrate the bandwidth demand as a basis for comparison if we were to send all camera data without any processing. When we consider multiple sensor nodes communicating such amounts of data, the problem is pronounced further within the network. Our approach, on the other hand, allows us to reduce the amount of data that will need to be supported. The communication overhead of our proposed architecture is demonstrated in Figure 6.3 in comparison to the case without any processing. In the figure, the node first processes the camera collected data to decide if a specific event is being observed. If that is the case, the event description, rather than multimedia description is sent to the sink node to minimize the communication overhead. In addition, particular snapshots are also sent to the sink node to assist query processing. The specifics of our approach are as described in the following sections.

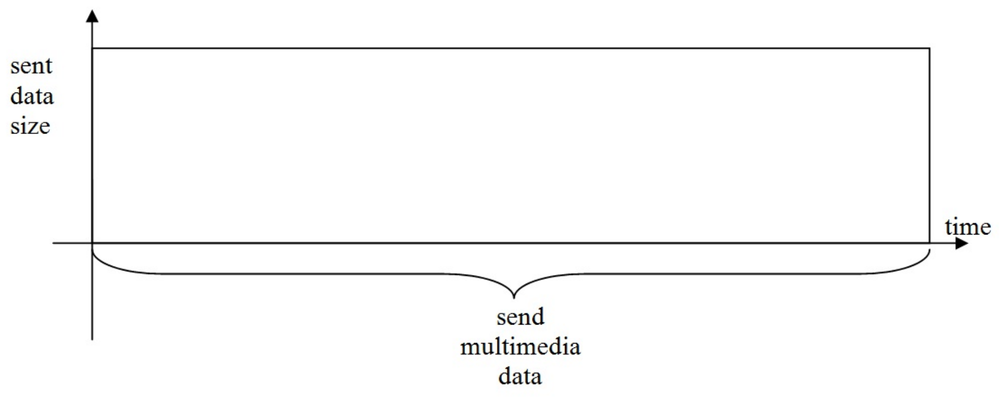


Figure 6.2: Communication overhead of the traditional wireless multimedia sensor networks

6.4 Efficient Object Tracking

Since running complex identification and tracking algorithms at the sensors is a problem in WMSNs due to our lack of energy and processing capability, we propose to run an identification algorithm at the sink to push the processing and the energy load from the camera sensors to the sink. One possible solution for identification in this case would be to be able to zoom in/out of these objects for better views once they are detected by the camera sensors and this information is received by the sink. While this type of solution would help in improving accu-

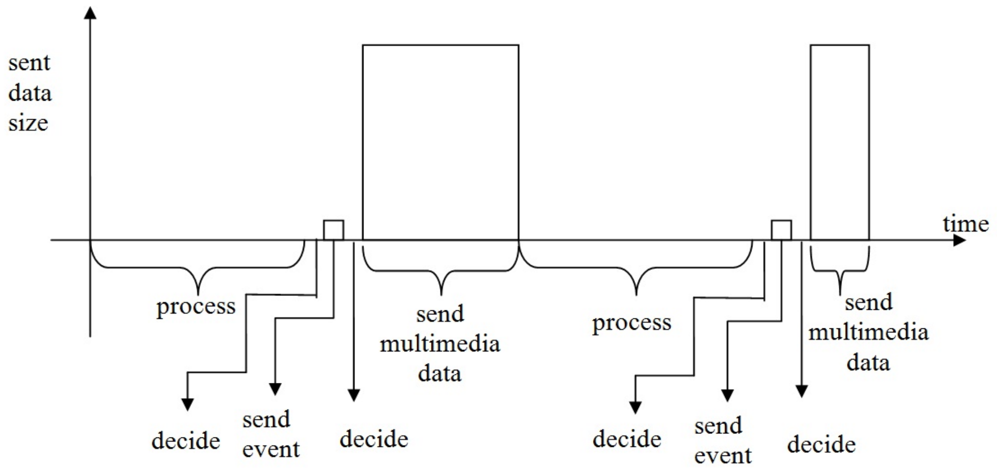


Figure 6.3: Communication overhead of the proposed architecture

racy, it is not feasible in our case assuming that the WMSNs employ simple cameras that do not have pan-tilt-zoom capabilities due to energy constraints. In addition, the choice of which frame/frames to send to the sink is not an easy process in such a case. Therefore, we opt for an identification solution based on the MOEs received from the cameras. Note that the sink does not perform object identification directly from the raw images. No information other than MOEs flows from the camera sensors to the sink. The sink performs object identification by just looking at the limited information carried in MOEs.

Basically, the MOEs coming from different camera sensors at different times are processed at the sink to identify the real objects in the area under surveillance. The identified objects are stored in a list at the sink. Since the similarity between two objects can not be expressed with crisp values, whenever a new MOE containing an object information is received from one of the camera sensors, a fuzzy similarity search is performed with the existing objects in the list to determine if this is a newly recognized object or not.

The total fuzzy similarity between two objects is calculated based on two *sub_similarities*, namely *the object class similarity* and *the velocity similarity*. While the former compares the membership values, the latter compares the speed of objects. The speed is a reasonable choice for comparison given that the speed of an object would not change abruptly between consecutive frames. We assign weights to each of these *sub_similarity* classes. The weight of *the object class similarity* on total similarity is taken as 0.75 since it includes three fuzzy variables (i.e., human, animal and vehicle) out of four including the velocity. The weight of *the velocity similarity* is 0.25. (In our application, these constants are chosen according to the features extracted in our framework. Of course, other constants may be chosen according to the features used. For instance, if another feature is used additionally or some of the features are not used at all, other constants may be set appropriately. The important point here is not the constants themselves but how they are used in the algorithm)

There are various methods for performing similarity analysis using membership value, such as the max-min, cosine amplitude, correlation coefficient and others, as described by [Ross95]. All methods give corresponding results, though numerical values would differ. We choose the max-min method because it is computationally simple yet effective.

Consider the similarity of two fuzzy membership vectors, V and Y having dimension n. The similarity value is defined as:

$$Similarity_{VY} = \frac{\sum_{i=1}^{n}(min(V_i, Y_i))}{\sum_{i=1}^{n}(max(V_i, Y_i))} \tag{6.1}$$

We now explain how Equation 6.1 is used at the sink. Consider two different objects, O_1 and O_2 received by the sink from different sensors S_1 and S_2 at different times T_1 and T_2 where $T_2 > T_1$. If *the object class similarity* between the objects O_1 and O_2 is above 0.66 which means that total similarity is above $0.66 * 0.75 = 0.5$ (our experiments shows that $0.5/0.75 = 0.66$ is a good choice), then there is a good chance that these two objects are the same 0.66 value is picked based on our observations from the experiments. Specifically, if a greater value is set as a threshold, a lot of objects were considered as different even if they were instances of the same object coming from different cameras. Similarly, if a smaller threshold (i.e., less than 0.66) is picked, this time different objects were considered as same even though they were not. Thus, 0.66 was the best choice in terms of balancing the accuracy. After calculating *the object class similarity*, if we achieve 0.5 value for *the total similarity*, thus we need to check *the velocity similarity*. Once *the velocity similarity* is computed, it is added to *the total similarity* and thus the cumulative object similarity is found.

Algorithm 1 implements this idea. If O_1 and O_2 are very different objects (i.e., one of them is a human being and the other one is an animal), then the $findObjectClassSimilarity$ function returns *the object class similarity* value as less than 0.66 (line 2). If O_1 and O_2 are similar kind of objects (i.e., two of them are human beings), then the $Total\_Similarity$ parameter becomes bigger than or equal to 0.5 and the *velocity similarity* is also considered, as shown in line 3. Specifically, the velocity of the object O_2 is computed by dividing the distance (i.e., the distance between the location of O_2 and O_1) by the time elapsed between the reporting times (i.e., $t_2 - t_1$). If this velocity is similar to the formerly computed velocity of O_1, then O_1 and O_2 are very likely the same object. In such a case, the $findObjectVelocitySimilarity$ function returns very close to 1.0 (line 5). Multiplied by its weight, 0.25, it is added to the $Total\_Similarity$ parameter. If they are actually the same object, this calculation makes the $Total\_Similarity$ parameter very close to 1.0 at the end.

Note that while the objects may be similar, their velocities may differ if O_2 decreases its velocity physically when traveling between the location of O_1 and O_2 (i.e., it may follow a zigzag movement in a blind area). In such a case, *the velocity similarity* is less than 1.0 making the $Total\_Similarity$ parameter between 0.5 and 1.0.

Algorithm 1 Object Similarity Algorithm at the Sink

```
Require: Two objects with their membership vectors and velocities exist.
Ensure: Total similarity value between two objects is found.
1: $Total\_Similarity \leftarrow 0$
2: $Total\_Similarity \leftarrow findObjectClassSimilarity(Object_1, Object_2) * 0.75$
3: if $Total\_Similarity \geq 0.5$ then
4:     $Total\_Similarity += findObjectVelocitySimilarity(Object_1, Object_2) * 0.25$
5: end if
```

The details of the algorithm at the sink are given in Algorithm 2. The sink stores the MOEs in an $Event\_List$ and identified objects in an $Object\_List$ as shown in Figure 6.4. The first received MOE is noted as a new object in lines 4 and 5. For the rest of the received MOEs, fuzzy similarity values of the object in MOE with the existing objects at the sink are compared by using Algorithm 3 in lines 7 and 8. For instance, the objects contained in MOEs coming from 3 different camera sensors in Figure 6.4 are compared one by one with the objects already detected and stored in the $Object\_List$ in FIFO manner. If the same object exists in the list (i.e., the fuzzy similarity value is greater than a threshold), the MOE is associated with that object definition stored at the sink (lines 10 to 14). Otherwise it is noted as a new detected object in line 16 and 17. When the moving objects are identified correctly, the current path of an object is simultaneously extracted in line 12 of the Algorithm 2 and stored as a part of the objects in the $Object\_List$ shown in Figure 6.4.

Algorithm 2 Object Tracking Algorithm at the Sink

```
Require: $Event\_List \neq \varnothing$ {Sink is getting MOEs from cameras and stores them in the
    $Event\_List$}
Ensure: $Object\_List \neq \varnothing$ {Objects are identified into the $Object\_List$}
 1: $Object\_List \leftarrow \varnothing$
 2: while $Event\_List \neq \varnothing$ do
 3:   $MOE \leftarrow$ Get a MOE from $Event\_List$ {FIFO}
 4:   if $Object\_List = \varnothing$ then
 5:     $Object\_List += MOE.Object$ {first object}
 6:   else {Objects are identified before}
 7:     for all Objects in $Object\_List$ do
 8:       Find object similarity values with $MOE.Object$ {use Algorithm 3}
 9:     end for
10:     if The maximum similarity value $\geq threshold$ then
11:       $CO \leftarrow$ chosen object having maximum similarity value
12:       $CO.Route += MOE.Object.Location$
13:       $CO.Last\_Seen\_By \leftarrow MOE.Sensor\_ID$
14:       $CO.Last\_Seen\_At \leftarrow MOE.Detection\_Time$
15:     else {The maximum similarity value $< threshold$}
16:       $Object\_List += MOE.Object$ {new object}
17:     end if
18:   end if
19: end while
```

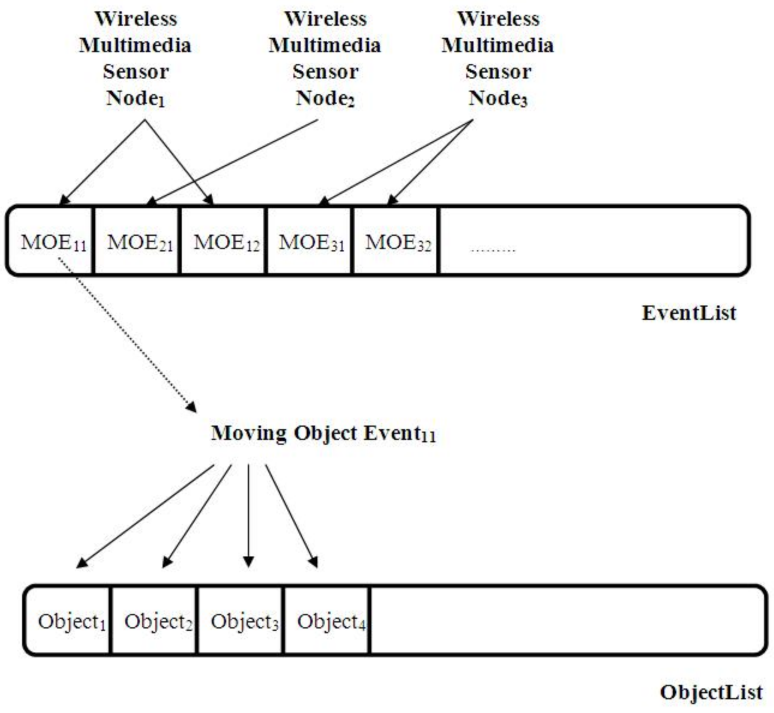


Figure 6.4: Progress of Object Tracking Algorithm. The sink stores the MOEs in the *Event_List* and identified objects in the *Object_List*. The objects contained in MOEs coming from 3 different camera sensors are compared one by one in FIFO manner with the objects already detected and stored in the *Object_List* (i.e., the first MOE of the first sensor is MOE_{11}). The object to have a maximum similarity value with the object contained within MOEs is associated with it.

We should note that we define a lifetime threshold for each object stored at the sink. This is because when an object has disappeared in a blind area, the sink should forget about it after its lifetime passes. If the time elapsed from the last detection time for an object exceeds this life time threshold (i.e., the object has disappeared for a long time), the object is removed from the list to prevent false matchings. If it re-appears, then it is considered as a new object. Hence, unnecessary storage of the detected moving objects is prevented and the negative effects of the blind areas are reduced.

The final object definition stored at the sink is given below:

- Route (List of X and Y coordinates in timeline)
- Last sensed by $Sensor\_ID$
- Last sensed at $Timestamp$

- Approximate velocity of the moving object
- Membership value of the object
  - for the class ‘Human’
  - for the class ‘Vehicle’
  - for the class ‘Animal’

6.5 Algorithm Analysis

For the tracking stage of our approach occurring at the sink, we should investigate the whole scenario. Assume that at time t, there exist M different objects and N different camera sensors in the area under surveillance. Let us also assume that there are m different objects, where $m < M$ is seen by n different camera sensors, where $n < N$. Hence, in the worst case, the MOE count that may be sent to the sink at time t is mn which is $O(MN)$. The sink compares each event with its own stored object definitions to find their similarities which has maximum number of $O(M)$. Hence at time t, the sink makes $O(M^2N)$ comparisons. If the whole scenario is run T time units, then the computational complexity for object identification and tracking at the sink is $O(M^2NT)$.

Table 6.1: Notation Used in Time Complexity Analysis for Object Tracking

t	Time instant when the inspected tracking occurs
T	Total time when the whole tracking occurs
M	Number of total objects exist at the area under surveillance
m	Number of objects sensed by the camera sensors
$M - m$	Number of object that cannot be sensed by the camera sensors i.e. in the blind area
N	Number of total camera sensors exist at the area under surveillance
n	Number of camera sensors that can sense any object at time t
$N - n$	Number of camera sensors that can sense none object at time t

6.6 Experimental Evaluation

6.6.1 Experiment Setup and Performance Metrics

To test the tracking performance, we have developed an application to simulate a border surveillance application. This simulator is explained in detail in Appendix A. An area of size 1000X600 m^2 is considered for surveillance purpose. For each group of experiments, different numbers of camera sensors are placed randomly with random orientations in order to achieve the desired coverage values as shown below. Coverage values are calculated in terms of total FoV with respect to the total area.

- 25% coverage: 29 camera sensors
- 50% coverage: 61 camera sensors
- 75% coverage: 102 camera sensors
- 100% coverage: 120 camera sensors

Note that these cases also cover the effect of any possible obstacles in the environment. In case of obstacles, depending on the number of such obstacles, the coverage values may decrease slightly. In addition to obstacles, in some cases, there may not be enough cameras to monitor the whole region. Hence, to account for such cases, we have considered coverage values from 25% to 100%.

Transmission ranges and DoFs are picked as 200m and 100m respectively. The sink node is assumed to be placed at (0, 0) location. Objects are generated randomly and their membership values and velocities are also assigned randomly. In each individual experiment, objects have special mobility pattern designed for these experiments. Each object has the purpose to enter the area and move towards the sink as if it was an intruder aiming to capture an important structure positioned at the sink. Membership values for a random object, (i.e., for a human intruder is a vector of (a, b, c)), are calculated as $a > 0.5$, $b + c = 1 - a$ where a, b and c are random numbers. For instance, for 6 different objects of 2 humans, 2 vehicles and 2 animals, the membership values produced with this formula are given in Table 6.2.

Two metrics are considered to assess the performance of the proposed algorithm.

Table 6.2: Random Membership Values for 6 Objects

Object Name	**Membership Value**
Human 1	(0.86; 0.04; 0.10)
Human 2	(0.93; 0.00; 0.07)
Vehicle 1	(0.14; 0.61; 0.25)
Vehicle 2	(0.03; 0.87; 0.10)
Animal 1	(0.12; 0.08; 0.80)
Animal 2	(0.13; 0.27; 0.60)

- *Identification Accuracy*: This metric indicates how successful the algorithm is when identifying the moving objects. It is defined as follows: At each time unit, the real object count under surveillance and the calculated object count by the sink are compared. If these values are not equal to each other, the ratio of the difference of these values to the real object count is noted as a *false ratio*. $(1 - falseratio)$ is defined as the identification accuracy. Note that the lifetime threshold defined before prevents the false ratio from being less than 0 or bigger than 1. Our goal is to maximize the identification accuracy.

- *Energy*: This metric is the energy consumption value spent in both communication and computation when tracking and classifying the moving objects at the camera sensors. Our goal is to minimize the energy consumption of battery-operated camera sensors.

We have conducted six groups of experiments:

- Accuracy evaluation in optimistic scenarios (i.e., no errors in the inputs and no packet loss in the network),

- Accuracy evaluation in pessimistic scenarios when the classification is faulty (i.e., a camera sensor classifies a human intruder as an animal or vehicle),

- Accuracy evaluation in pessimistic scenarios when the localization is faulty (i.e., a camera sensor calculates the location of an intruder with some error),

- Accuracy evaluation in pessimistic scenarios when the packet loss occurs in the network,

- Energy consumption evaluation of the network,

- Battery lifetime evaluation for each camera sensor.

6.6.2 Experiment Results

6.6.2.1 Identification Accuracy

In order to observe the identification accuracy, we have set up experiment groups that include 3, 6, 9 or 12 different moving objects. Each experiment group is performed under 4 different camera sensor coverage values given in the Experiment Setup subsection. The change of the average identification accuracy over time for 3 different moving objects for different coverage values is shown in Figure 6.5. Overall accuracies for all coverage values and for all object numbers are presented in Table 6.3. We note that the results included here are recorded between the time interval [40,160], since on average, a random moving object enters into one of the camera's FoV at time 40 and begins to disappear from the region at time 160.

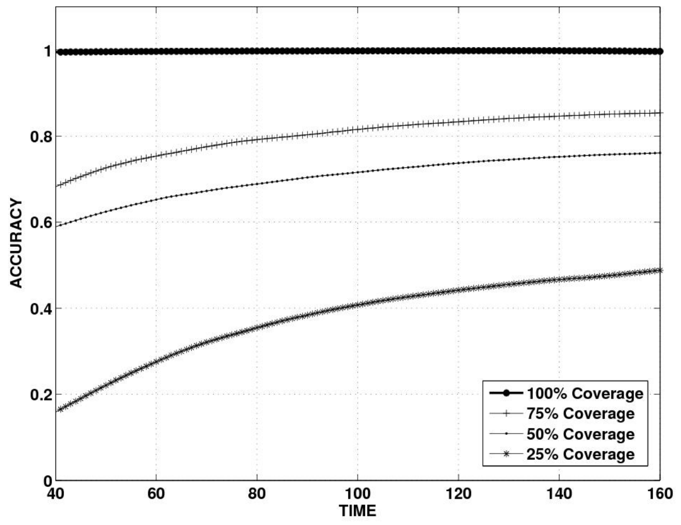


Figure 6.5: Change of identification accuracy values over time for experiments with 3 different objects on all camera coverage values.

In optimistic scenarios, we target the identification accuracy value at least as much as the total camera coverage percentage. For instance, for 25% camera coverage value, one should achieve at least 25% identification accuracy as the sink can monitor only 25% part of the whole area and 75% of the area is the blind area. The results of our experiments show that, under optimistic scenarios, the identification accuracies are higher than these base coverage

Table 6.3: Average Identification Accuracies

Object Count	**25% Cov.**	**50% Cov.**	**75% Cov.**	**100% Cov.**
3 Objects	0.38	0.71	0.81	1.0
6 Objects	0.38	0.74	0.81	1.0
9 Objects	0.39	0.73	0.83	1.0
12 Objects	0.39	0.75	0.83	1.0

values.

The results also indicate that when the coverage is 100%, our approach guarantees accurate tracking of any number of objects. With the decreasing coverage percentage value, the accuracy decreases. However, this decrease is not proportional to the decrease in coverage value, indicating the effectiveness of our approach. In all coverage cases, the identification accuracies are always higher than the coverage percentage values.

6.6.2.2 Effect of False Classification

The experiments on object identification accuracy given in the previous subsection were performed under the assumption that the camera sensors always produce similar results about moving object classification. For instance, a camera sensor always identifies an intruder as a human if a previous camera sensor has already identified it as a human. However, there may be some erroneous situations when camera sensors may malfunction. For instance, a camera sensor may see the intruder partially and identify it as being an animal instead of a human or an animal may be captured from a different angle and classified as a human.

To assess the effect of this kind of erroneous situation on the accuracy of identification, we have also conducted experiments in which we introduced an error of 5%, 10%, 15% and 20% in classifying the objects. The change of the average identification accuracy over time for the 5% error case with different coverage values is shown in Figure 6.6. All the results for erroneous situations in classification are given in Table 6.4. To compare the results with Table 6.3, we have used 3 different moving objects in these experiments.

Our experiments show that, with decreasing coverage, the accuracy increases over time as

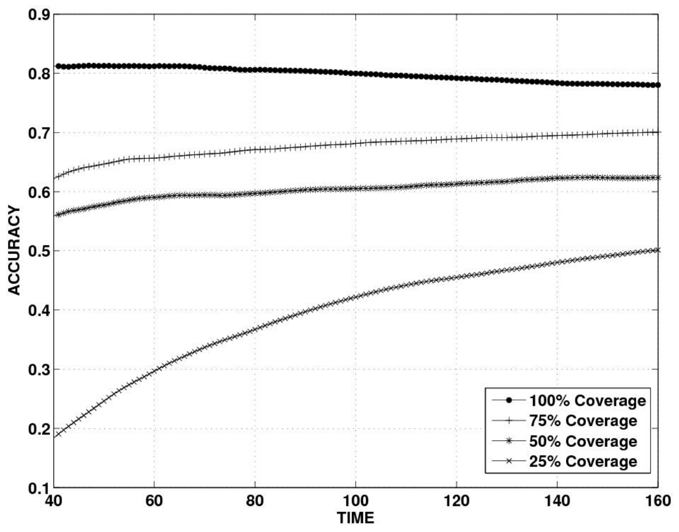


Figure 6.6: Change of identification accuracy values over time, if 5% error in classification exists at camera sensors for 3 moving object case

Table 6.4: Effect of False Classification over Average Identification Accuracies

Error Value Introduced	**25% Cov.**	**50% Cov.**	**75% Cov.**	**100% Cov.**
5% Error	0.37	0.60	0.67	0.79
10% Error	0.36	0.52	0.56	0.63
15% Error	0.33	0.43	0.48	0.50
20% Error	0.29	0.36	0.38	0.48

seen in Figure 6.6. This can be explained as follows: An erroneous case which may be recorded in a particular frame disappears while processing the rest of the video. For instance, an animal classified as a human in one frame is eventually classified as an animal on other frames since it is detected from the right angle to be classified as an animal. For increased coverage the accuracy is increased and it maintains a certain level even when the errors are increased. This can be attributed to that fact that more than one camera sensor may sense the same object at the same time in case of increased coverage. This increases the robustness of the algorithm. While one camera sensor may classify the sensed object into the wrong class, another sensor may classify the same object into the right class. As a result, if the error rate for one camera sensor is 10%, the error rate for two camera sensors to classify the same object into the wrong class becomes 1%. This rate decreases even further for three or more camera cases.

6.6.2.3 Effect of Localization Error

We have performed experiments to assess the effect of localization error. The results are given in Figure 6.7. The results indicate that for 20% localization error, the accuracy does not reduce more than 20% on average.

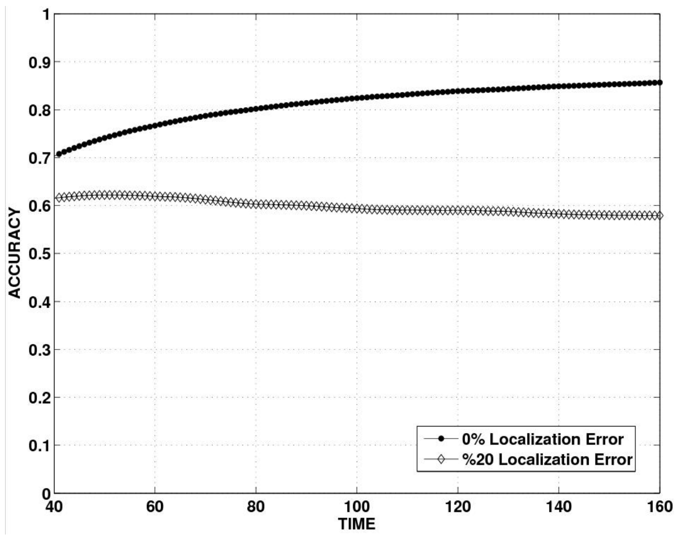


Figure 6.7: Change of identification accuracy values over time.

6.6.2.4 Effect of Packet Loss

There is always a chance that some packets may be lost in the network due to wireless environment and multi-hop transmissions. We have performed experiments considering different packet loss percentage values. The experiments have assumed 75% of coverage value for the network and 3 different moving objects existing in the area under surveillance. As explained in previous experiments, since our algorithm is robust against even low coverage values such as 50% or 25%, it can still identify and track moving objects even under high packet loss percentages. In case of packet loss, some of the information will not be available at the sink. However, given the nature of our algorithm which depends on different frames and information from different sources, it can maintain reasonable identification accuracy as seen in Table 6.5.

Table 6.5: Average Identification Accuracies when Packets Loss, 75% Coverage, 3 Objects

Packet Loss Percentage	**Average Identification Accuracy**
0%	0.81
5%	0.78
10%	0.75
15%	0.69

6.6.2.5 Energy Consumption at Sensors

In order to show the efficiency of this algorithm and its suitability for WMSNs, we have conducted experiments to test computational and communication energy costs. We have used the AVR Simulation and Analysis Framework (AVRORA), an emulator, to calculate energy costs of processing and communication [WEB:AVRORA]. We have considered two baseline approaches: 1) Transmission of the whole frame [Huiyu08][Qi07][Alaei09] referred to as *Traditional Method Sending Full Frame to Sink* in the graphs; and 2) Transmission of the frame part containing detected object referred to as *Traditional Method Sending Object Frame to Sink* in the graphs. The associated energy costs for these are given in Table 6.6. Based on this table, the total costs of the computation and transmission processes are calculated as in Table 6.7.

Table 6.6: Energy Costs for Different Tasks

Process	**Cost in Joule**
CPU cost to process the frame to extract the moving object	0.0180 = C
Transmission cost of the whole frame for 1 hop	0.0700 = M
Transmission cost of the part of the frame consisting the object for 1 hop (assuming at average the object covers 20% of the frame)	0.0140 = N
Transmission cost of the moving object alarm for 1 hop	0.0007 = T
Taking the video data	Same for both cases and ignored

Considering these energy costs, we have conducted experiments with varying hop-count (i.e.,

Table 6.7: Total Energy Costs in Joules

HOP #	Traditional Method (Full Frame Sent)	Traditional Method (Object Frame Sent)	Proposed Method (MOE Sent)
1	$M = 0.07$	$C + N = 0.032$	$C + T = 0.0187$
k	$Mk = 0.07k$	$C + Nk = 0.018 + 0.014k$	$C + Tk = 0.018 + 0.0007k$

network diameter), camera coverage and object count as seen in Figures 6.8, 6.9, 6.10 and Tables 6.8, 6.9, 6.10. The results clearly indicate how the proposed approach outperforms the baseline approaches in terms of energy consumption. The energy saving is significant, particularly when the network diameter, coverage or object count is increased.

Table 6.8: Energy Costs of Three Different Methods for Sending One Packet Under Increasing Number of Hops

HOP #	Traditional Method (Full Frame Sent)	Traditional Method (Object Frame Sent)	Proposed Method (MOE Sent)
1	$0.07J$	$0.032J$	$0.0187J$
2	$0.14J$	$0.046J$	$0.0194J$
3	$0.21J$	$0.060J$	$0.0201J$
4	$0.28J$	$0.074J$	$0.0208J$
5	$0.35J$	$0.088J$	$0.0215J$

Table 6.9: Energy Consumption of Whole Simulation with Varying Camera Coverage Values, in Which Each Coverage Value Contains Different Numbers of Deployed Camera Sensors

Camera Coverage Values & Numbers	Traditional Method (Full Frame Sent)	Traditional Method (Object Frame Sent)	Proposed Method (MOE Sent)
25% - 29	$8543.94J$	$1711.61J$	$88.26J$
50% - 61	$17331.43J$	$3472.21J$	$179.24J$
75% - 102	$27691.54J$	$5548.14J$	$286.75J$
100% - 120	$79128.22J$	$15852.81J$	$818.44J$

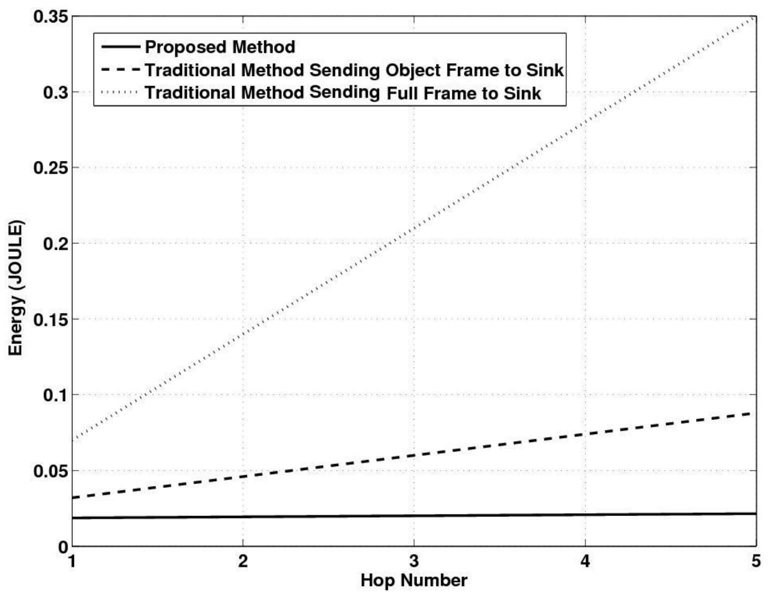


Figure 6.8: Energy costs graph of three different methods for sending one packet of information under increasing number of hops

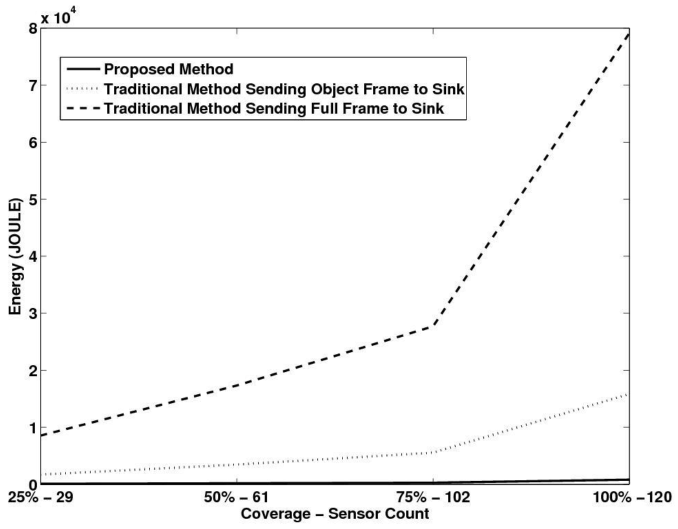


Figure 6.9: Energy consumption graph of whole simulation with varying camera coverage values

6.6.2.6 Battery Lifetime for Sensors

To check the individual battery life in a real-application, we performed experiments assuming that camera sensors may contain 4 AA batteries [Pinto10] for a total energy of 10KJ as given in [WEB:BATTERY]. We used the scenario of 75% coverage and totally 300 different intrusions were simulated. For this purpose, a total of 100 experiments each lasting 200 time

Table 6.10: Energy Consumption of Whole Simulation with Varying Object Counts

Object Counts	**Traditional Method (Full Frame Sent)**	**Traditional Method (Object Frame Sent)**	**Proposed Method (MOE Sent)**
3	27691.54J	5548.14J	286.75J
6	55195.13J	11058.43J	571.36J
9	84160.23J	16861.44J	870.99J
12	112274.43J	22494.16J	1162.01J

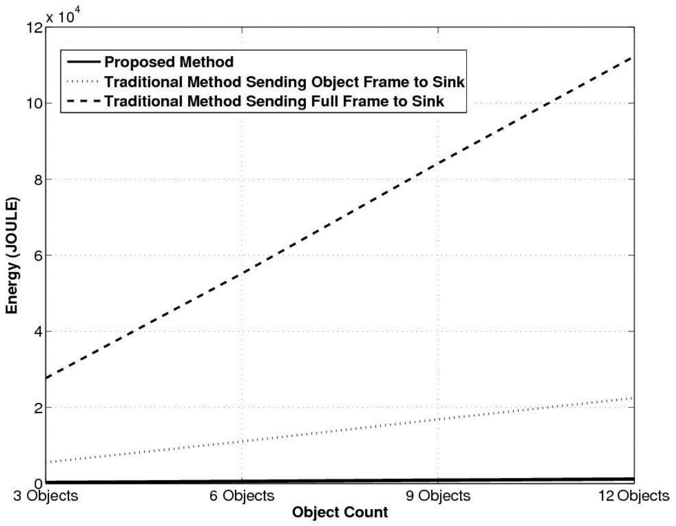


Figure 6.10: Energy consumption graph of whole simulation with varying object counts

units were performed. The remaining battery lives of each individual camera node after 20K (200X100) time units are shown in Figure 6.11. The results show that even after 300 different intrusions, none of the camera sensors had run out of energy. However, some of the sensor nodes consumed more energy than others since they were on the way of intruder objects and thus more likely to sense the intruders.

6.7 Summary

In this chapter, we have investigated a multiple object tracking problem in a typical border surveillance application of WMSN using minimal resources. Instead of raw frame or raw video data with huge size, the classification information along with the location of the object is sent to the sink as part of the MOE for tracking purposes. The sink gathers MOEs from

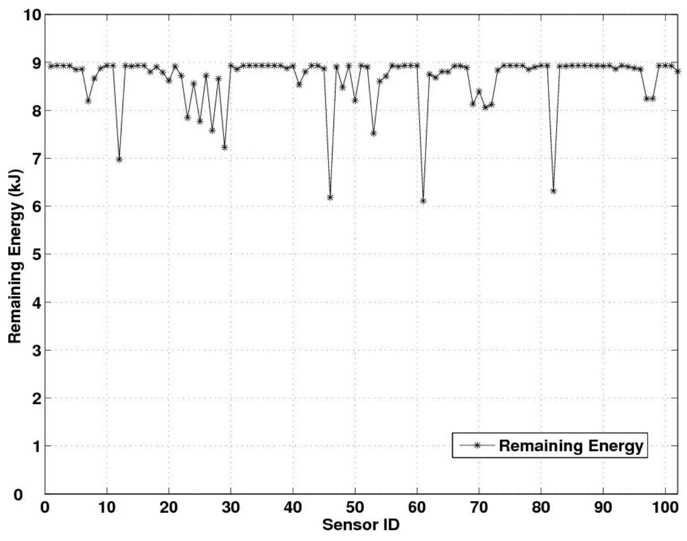


Figure 6.11: Remaining energy values of individual 102 randomly deployed camera sensors at the area after 300 different intrusions

different camera sensors and processes them using a fuzzy similarity and tracking algorithms. In this way, the sink is able to differentiate and identify distinct moving objects under mostly blinded areas and track them in real-time.

The algorithms proposed have been implemented and tested under variety of conditions. The experiments results have shown that the algorithm can accurately identify and track multiple moving objects even under low area coverage percentages. The results also indicate that this accuracy is slightly degraded with the introduction of errors (classification errors, localization errors or network packet losses). Nonetheless, in all cases, this process of tracking has been performed with significantly less energy consumption than are needed in the traditional approaches, which increases the battery lifetime of the camera sensors.

CHAPTER 7

ACTIVE RULE PROCESSING

In this chapter, we go into detail about active rule processing problem in WMSNs. For that purpose, we give the solution to the problem with a proof of concept. Then, we explain how we implement the active rules in a rule inference engine. Not only simple events but also complex events are explored.

7.1 Active Rule Processing Issues in WMSNs

Wireless Sensor Networks (WSNs) can collect a huge amount of data from the environment with inexpensive deployment and low labor costs [Akyildiz02]. After data collection, such data are typically processed by appropriate personnel, and necessary actions are taken through time. To reduce human intervention in processing the data and taking some limited actions without human intervention, Wireless Sensor and Actuator Networks (WSANs) have recently been proposed [Akyildiz04]. In such networks, typically the collected data are processed in an autonomous manner at the sink and the actions necessary to be taken are decided upon based on the received data. The actions can be taken either via actuators (actors) that are specialized based on the application (e.g., sprinklers, robots, etc.) or by informing the necessary personnel for human intervention. These types of networks are gaining popularity through cyber-physical systems as well [Noble09]. Depending on the application, such automated networks are mainly used to minimize human intervention and provide more effective actions. Examples of such use include fire fighting in forests, intrusion detection at borders and remote surveillance in outdoor contexts.

One of the main challenges in such sensor-actuator based systems is to implement a mechanism at the sink which can process the data and infer some actions. For this purpose, reactive solutions, which are based on asynchronous processes with the assumption that events may occur at any time, have been widely employed in the past [Zoumboulakis04][Korpeoglu06] [Ceri94][Ghanem93]. For instance, an intruder may enter an area at any time and an immediate reaction would be needed. In many of these works, Event-Condition-Action (ECA) rules are used to take necessary actions in response to these asynchronous events based on conditions. For instance, the rules in a forest fire monitoring application may be as follows: *On the event of excessive warming in the monitored area (event) and if the measured temperature is above 60 ^{o}C (condition), then send fire fighters to the area immediately (action).* This is an informal ECA rule written in natural language. Since a fire may occur at any time in the forest, the action should be performed immediately when it occurs.

While there has been a number of previous studies in WSNs that aim such a reactive framework using ECA rules [Korpeoglu06][Zoumboulakis04], this problem has not been studied for Wireless Multimedia Sensor Networks (WMSNs), which deploy wireless camera sensors that can collect image/video data [Akyildiz07][Rahimi05][Hengstler07][Soro09]. Such networks deploy a large number of camera (video) sensors with different capabilities and can collect/process multimedia data [Rahimi05][Hengstler07]. Typical applications of WMSNs include multimedia surveillance, target tracking, habitat monitoring, and intrusion detection [Akyildiz07][Akyildiz08]. In such applications, when a large number of cameras are deployed there is also a need for a large number of personnel in order to monitor the data. This is not feasible in large-scale applications like border monitoring [WEB:BorderReport05]. Therefore, a reactive system would be a convenient solution in reducing human involvement as well as in increasing the efficiency of these systems. In this chapter, an active rule processing solution is proposed for WMSNs.

Recent works on WMSNs focus on processing and compression of the raw video data sent to the sink in order to increase network lifetime, providing Quality of Service (QoS) for data communication and placement of cameras for desired coverage [Akyildiz07][Soro09] [Akkaya08][Akkaya09][Erdem06]. However, given limited energy reserves for camera sensors, instead of sending large-size raw multimedia data, limited data may be sent to the sink [Oztarak09]. For instance, when an intruder is detected in an event, instead of raw video of the object, information about the detected object, location of the object, class of the object and frame of the object, etc. may be sent directly to the sink. If the detected event is an event of interest for the sink, then the sink may decide to request the raw video data for better recognition of the event, otherwise the event may be ignored. Hence, this method reduces the energy consumption significantly, even though it presents new challenges regarding object and event identification, data processing and data fusion. These challenges are going to be explained in following paragraphs:

First of all, due to the goal of data reduction at the camera sensors, there will be limited/ uncertain/imprecise data at the sink regarding detected objects. For instance, in a typical surveillance application, individual low resolution camera sensors may not be able to decide whether the detected object is a human or an animal, due to their limiting video processing capabilities. Instead, they may send their guesses (i.e., by utilizing fuzzy logic) about the classification of the detected moving objects. Deciding which actions should be taken at the sink becomes more challenging with such imprecise data. Specifically, the reactive system should have the capability of reasoning when it has only uncertain and incomplete information.

Secondly, there needs to be a fusion mechanism at the sink before the collected data are processed. This is due to the redundancy of the received data. Typically, due to the availability of a large number of low-cost wireless camera sensors, an event may be captured by different camera sensors (i.e., the same event may be captured under the Field-of-Views (FoV) of several camera sensor nodes), creating redundancy at the sink. Hence before taking an action, the detected objects and their corresponding events should be analyzed and identified.

Finally, due to the complexity of events and the multi-modal nature of the data gathered at the sink, fuzziness in events needs to be supported. For instance, *"give an alarm, when there is a human intruder* ***running fast*** *towards to the left gate"*, contains a fuzzy term, ***running fast***, which is an uncertain concept. A reactive system used in such an application should process that type of information.

In this dissertation, in order to handle the challenges mentioned above, we propose to extend the ECA rules used in WSNs in order to take actions against asynchronous events, when providing autonomous operation in WMSNs. Data fusion is performed on the received objects before ECA is applied. This step is referred to as the *WHEN* part of the extended active rules. For instance, *"WHEN two different sensors sense the same human intruder at the same time (fusion), ON the event of human is running fast (event) and IF the human is close to the camera sensor (condition), THEN request the raw video data from close sensor immediately* *(action)"* is an informal extended ECA rule written in natural language. In this dissertation, we propose that, this syntax be employed as WECA rules. Based on this rule, raw video data is requested from the appropriate camera node only when some events and conditions are met. In this way, one can get rid of the burden of receiving huge raw video data at all times from the camera nodes.

In order to demonstrate the feasibility and effectiveness of the proposed *WECA* rule syntax, we assume a power-plant surveillance application, where the camera sensors transmit to the sink only detected object information rather than the raw video data. Such data are processed at the sink node using fuzzy logic. The goal is to be able to eliminate uncertainty, to identify detected objects along with their events and to take actions against the detected events. The rules are stored in a Data/RuleBase and retrieved in reference to existing events. An inference mechanism is then employed in order to decide on the actions to be taken. We have implemented the rules in a power-plant remote surveillance application and have developed a proof-of-concept using JESS, the rule engine for the JAVA platform [WEB:Jess], to show the feasibility of the reactive mechanism. We have also conducted experiments on the accuracy metric when performing object fusion with uncertain data. The results show that the rules can be processed with acceptable object identification accuracies.

7.2 Problem Definition

Our problem for active rule processing can be formally defined as follows: "Assume that we have N wireless camera sensors distributed randomly in a critical environment for a surveillance purpose. At a specific time T, let us assume that there exist $M > 0$ different moving objects in the area performing either simple events or complex events. Instead of raw video data, the camera sensors that detect these M moving objects send multimedia data containing information about detected objects to the sink. Our goal is to define a rule-based reactive model at the sink which will process the received uncertain and limited data regarding these M moving objects. Therefore, the defined model will decide the appropriate actions to be taken in an autonomous manner." Eventually, the ultimate goal is to minimize the need for human intervention on the WMSN applications and improve the accuracy of the event detection.

We should note that, because of intersected areas, M moving objects may be captured by K

camera sensors where $M \leq K \leq N$. Therefore, data redundancy at the sink node is expected. To eliminate the data redundancy, the sink should be able to identify redundant gathered objects. However, limited data will be available at the sink, since camera sensors send only limited multimedia information due to resource constraints. Therefore, uncertainty is expected to be produced from multiple camera sensors sending similar data about the same objects.

Our goal while solving the above problem is to minimize the need for human intervention by automating the reactive process against intruders at the sink.

7.3 Active Rule Processing

As mentioned in Chapter 1, because of energy considerations, camera sensors are expected to do some data elimination/reduction and send just limited information about the detected objects instead of sending huge amounts of raw video data. Obviously, whenever needed, the whole frame can be transmitted but our goal is to minimize/eliminate these types of transmissions. With such limited information about the objects at the sink, one needs to trade off the quality of object identification/classification etc. with energy savings, which in turn affects the quality of decisions to take actions. Therefore, our solution will utilize application-level information to improve the accuracy of data received at the sink.

The overall mechanism depends on the data produced at the camera sensors and the data consumed at the sink node. Specifically, for object localization and classification, we use the approaches proposed in [Oztarak09] and [Oztarak12]. These approaches first extract the detected objects from the video frames using frame differencing at individual camera sensors. Once the object is extracted, a simple fuzzy object classification algorithm is run based on the used application. Then the distance between the object and the camera sensor is calculated. The object location is then calculated by using the camera's location and direction, the object position on the frame and the distance between the object and the camera.

From the camera sensors, only the classification and location information of the sensed moving objects are relayed to the sink, instead of raw video data. Note that the size of this information, which is *Moving Object Event (MOE)* data, is very minimal compared to the size of the raw video frames that need to be sent to the sink in traditional methods. This enables us to save significant energy for the camera sensors and bandwidth for the network which improves the lifetime of the system allowing the system to function without any human intervention for longer times.

In this dissertation, we propose a data fusion and event processing mechanism at the sink after the camera sensors send their limited information about detected objects (MOEs) to the sink. The goal is to use this mechanism to enable a reactive behavior as in the case of ECA systems. For this purpose, fuzzy logic is utilized to capture and eliminate uncertainty regarding the events/objects. Given the limited data from the camera sensors and incomplete information about the events, we cannot come up with crisp event definitions and conditions at the sink node. Therefore, the sink performs a four-phase mechanism after receiving the MOEs from different camera sensors. This mechanism forms the basis for implementing an ECA-like system as will be detailed in the next section.

7.3.1 WECA Rules for WMSNs

In order to build a system that self reacts against the occurring events, we propose an extended ECA rule syntax at the sink as shown in Figure 7.1 which we call WECA. Different from the traditional systems, WECA rules perform data fusion first with their *WHEN* part, before the rest of the rule are processed. In the figure, *ft* refers to *fusion threshold*, and *et* refers to *event threshold*. These threshold values are used when fusing the data or producing the events. Each line of the rules refers to a phase that is performed at the sink.

```
WHEN fusion list  <ft> //Phase 1
ON event list     <et> //Phase 2
IF condition list      //Phase 3
THEN action/conclusion //Phase 4
```

Figure 7.1: Extended ECA rule: WECA

Brief descriptions of the phases performed are as follows:

1. Phase 1: *Fuzzy Object Fusion*: The first phase where MOEs coming from different camera sensors are processed and fused if applicable (i.e., deciding whether two different MOEs sent by two different camera sensors belong to the same intruder).

2. Phase 2: *Fuzzy Event Identification*: The semantic event performed temporally by moving objects in the area under surveillance is extracted based on predefined event definitions in the application (i.e., deciding which simple/complex event the object performs).

3. Phase 3: *Fuzzy Condition Matching*: The conditions are processed to check whether the detected object/event is in a particular state at the time when MOE is received (i.e., deciding whether the object/event has some predefined conditions).

4. Phase 4: *Fuzzy Inference and Combination*: The phase where actions are inferred when all preliminaries are suitable (i.e., deciding what kind of actions will be performed). In this last phase, the superimposition of all fuzzy inferences about an action may also be performed when there exist more than one inference. Again the actions should be predefined in the system.

The overall architecture for processing these phases is shown in Figure 7.2. The architecture's main components are explained below:

1. *The User Interface*: The user creates his/her rules by using the User Interface. The actions or results of the active rules are also returned to the user by the user interface.

2. *The Data/RuleBase*: Rules created by the user are stored in the Data/RuleBase.

3. *The Rule Controller*: When an event occurs in the area, the MOEs coming from different camera sensors are fused by the Rule Controller. The Rule Controller retrieves corresponding rules from the Data/RuleBase. The Rule Controller processes the received rules by using the Inference Mechanism having the four phases given before to produce the required actions.

We now explain each of these four phases in detail under following subsections:

7.3.2 Fuzzy Object Fusion

The goal of this phase is to identify individual objects from the redundant information which can come from multiple camera sensors. This phase will also serve the event identification phase (i.e., the next phase) to facilitate accurate event identification. With referral to data

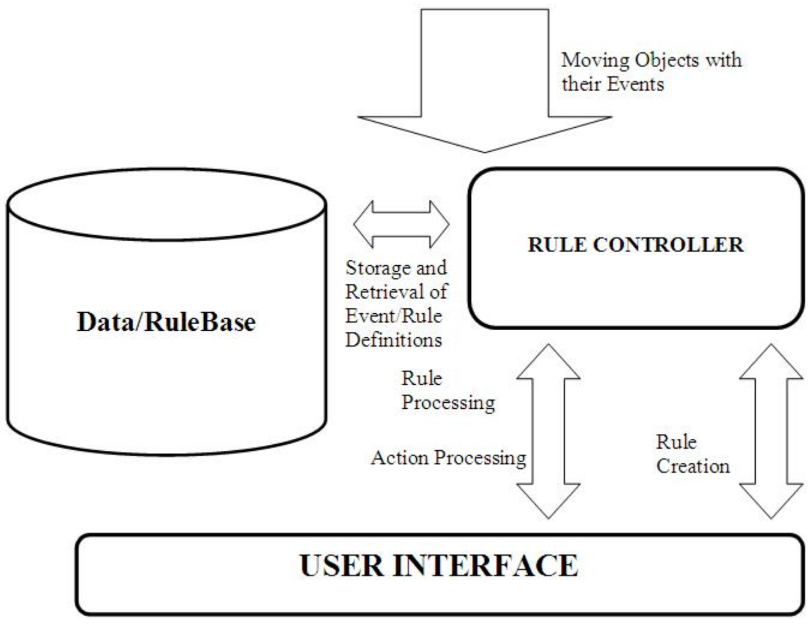


Figure 7.2: The architecture of the active framework

fusion, here we are aiming to fuse the collected information to gather the information of the objects that really exist in the region. In this phase, a list called the *Fusion List* is maintained. This list contains all possibly existing objects that arrived from camera the sensors at the sink along with their information. Data fusion is performed among the objects in this list. Fusion of two different objects depends on stored information in corresponding MOEs. For instance, a MOE may have the following information:

- ID of the camera sensor detecting the object
- Location of the camera sensor detecting the object
- Detection time
- Detected object
  - Location Estimate (X, Y, Z coordinates)
  - Velocity Estimate
  - Class/category of the object (based on a fuzzy function)

Consider the similarity of two fuzzy membership vectors, V and Y having dimension n. The similarity value is again defined as:

$$Similarity_{VY} = \frac{\sum_{i=1}^{n}(min(V_i, Y_i))}{\sum_{i=1}^{n}(max(V_i, Y_i))} \tag{7.1}$$

Similarly, we use the information stored in MOEs to create the n dimensions and compute the similarity value. We also define a fuzzy membership function based on the keywords given in the WECA rules. For instance, if the rule has the keyword VERY SIMILAR, then the similarity value calculated with the Equation 7.1 is compared with the membership value found using Figure 7.3.

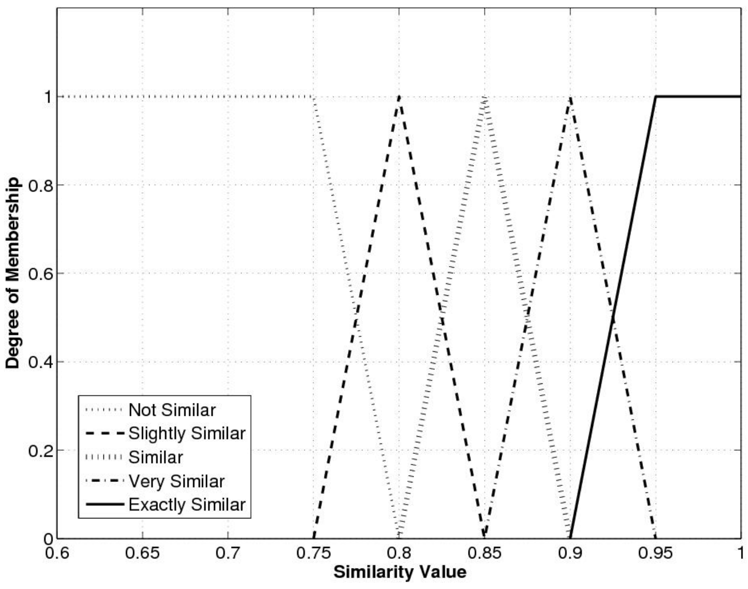


Figure 7.3: Example fuzzy membership set to process the fusion list

The details of the fusion algorithm and similarity comparison will be explained in the context of a sample application in Chapter 7.4.

7.3.3 Fuzzy Simple Event Identification

We propose identification of predefined events based on the temporal and spatial characteristics of the moving objects. The uncertainty due to incomplete information from the camera sensors is captured using fuzzy logic. The predefined events defined for the applied application will be used for matching with the existing temporal and spatial data. For instance, in a surveillance application, the events may be defined based on spatial features such as their velocity, direction or position. Each defined event will be checked with the existing event definitions from the database. The following events are sample events which can be defined using temporal and spatial states of the gathered objects at the sink:

- *Object getting closer to/further away from the camera sensor*: If the distance between the object's position and camera sensor's position is decreasing/increasing in two consecutive MOEs, we conclude that the object is getting closer to/further away from the camera sensor.
- *Human running*: If the velocity of the detected human intruder is above 5 km/h in at least three consecutive MOEs, we conclude that the human is running.
- *Object going east/west/south/north*: If the object is changing its position through east/ west/south/north at two consecutive MOEs, we conclude that the object is going east/ west/south/north. Determination of the direction can be based on a fuzzy function so that some flexibility can be introduced.
- *Object speeding up/down*: If the object is increasing/decreasing its speed in at least three consecutive MOEs, we conclude that the object is speeding up/down.
- *Object approaching a critical location (e.g., bridge/water supply)*: If the distance between object's position and critical location is decreasing between two consecutive MOEs, we conclude that the object is approaching the intended location.
- *Object climbing a structure*: If the Z coordinate of the object is changing while X and Y stay relatively similar in two consecutive MOEs, then we conclude that the object is climbing a structure. Note that this is not possible unless the location estimated is in 3-D.

7.3.4 Fuzzy Condition Matching

Before triggering an action, one needs to check the existing state of the objects involved in the events defined above. Note that conditions are defined different than events although they are part of the event occurring in the area. To check the special conditions, the system should investigate the properties of the object at the moment when such checking is performed. Therefore, no temporal investigation of the objects is needed. On the contrary, when identifying the events, we need to look at the temporal properties of the objects. For instance, an object's approaching to a critical structure is an event but this may not trigger an action until the object is very CLOSE to that structure, which is a condition about the position of the object.

At this phase, conditions are processed to check whether the network is in a particular state in which case the rule fires. These state conditions are directly related to object features and some of them can be expressed in fuzzy terms. For instance, a condition about the speed of an object may be given as being *slow* or *fast* or exactly *20 km/h*. Sample conditions for a typical surveillance application and how they can be checked on a single MOE may be listed as follows:

- *If the object is human/vehicle/animal*: This condition can be decided by checking the classification value of the gathered MOE.
- *If the object velocity is greater than v*: This condition can be decided by checking the velocity feature of the gathered MOE.
- *If the object is very close to a critical structure*: This condition can be decided by checking if the difference between the position of the intruder stored in gathered MOE and the critical structure is less than a predefined threshold (e.g., 10 meters).
- *If the object is close enough to the camera sensor*: This condition can be decided by checking if the distance between intruder and camera sensor is below $\frac{1}{4}$ of the camera's DoF.

7.3.5 Fuzzy Inference and Combination

The last phase concerns on inferring actions when all preliminaries are suitable. In this phase, the superimposition of all fuzzy inferences about an action is also performed when there exist more than one inference. Again all of these actions should be predefined in the system. Example actions for a typical surveillance application may be listed as:

- Sending an alarm to guards/personnel,
- Requesting the whole image from the next possible camera sensor on the way,

- Sending the received image of the detected object to the appropriate personnel's device,
- Activating all camera sensors around the object,
- Activating powerful PTZ cameras on the path of the object,
- Inserting an intruder record into the local database,
- Activating all emergency lights of the area to track the intruder better.

7.3.6 Complex Event Processing

7.3.6.1 Formal Simple Event Definition

The system proposed in this dissertation has camera sensors as the source of distributed messages. The camera sensors send extracted multimedia information to the sink. As explained in Chapter 7.3.3, simple events are semantic multimedia events that can be extracted from coming MOEs (messages coming from distributed sources, i.e. the camera sensors).

In this dissertation, we define the simple events formally as follows:

- SER: MOES $\rightarrow$ (SE, μ_S) is a **S**imple **E**vent **R**ule where:
  - MOES = $\{MOE_1$, MOE_2, ..., MOE_n $\}$ is a finite Set of Moving Object Events (MOE) coming from distributed camera sensors and each MOE_n has consecutive time labels with MOE_{n-1},
  - SE is a **S**imple **E**vent inferred by investigating the MOES depending on their semantic meaning,
  - μ_{SE} is the fuzzy membership value of inferred event SE and calculated through a fuzzy membership function defined appropriately.

For instance, a simple event of "a human intruder RUNNING FAST" can be defined with the proposed notation as follows:

HRR: $\{MOE_{t1}\}$ $\rightarrow$ (HR, μ_{run}) where MOE_{t1} has the velocity value for a human intruder at time $t_1$and μ_{run} is a membership value calculated through the fuzzy membership function

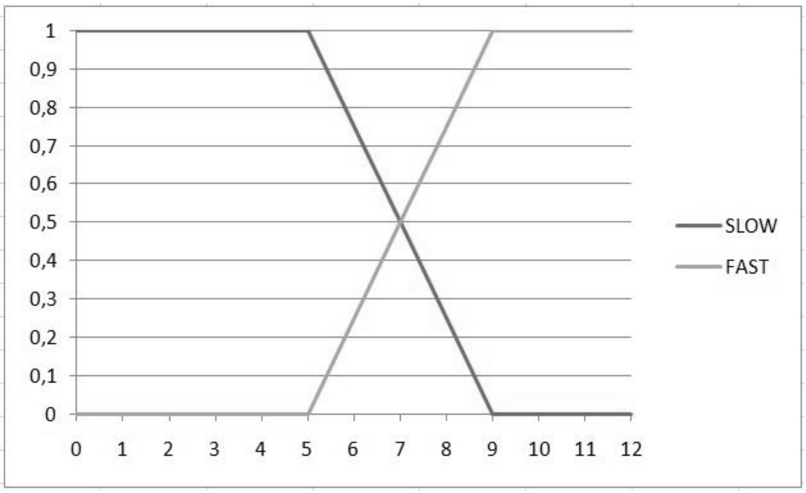


Figure 7.4: A sample fuzzy membership function for FAST and SLOW based on the velocity of human intruders

of being “FAST”. HR is the inferred “HUMAN RUNNING” event. For instance, a fuzzy membership functions for “FAST” and “SLOW” is shown in Figure 7.4.

Two another simple events named as “a vehicle ENTERING to a critical zone” and “a vehicle EXITING from a critical zone” which are based on spatial correlation between coming MOEs can be defined with the proposed notation as follows:

ENCZR: $\{MOE_{t1}, MOE_{t2}\} \rightarrow$ (ENCZ, μ_{encz}) where MOE_{t1} for a vehicle has the spatial coordinates out of a defined critical zone and MOE_{t2} for the same vehicle has the spatial coordinates in the defined critical zone. In most of the surveillance applications, the borders of the critical zones are defined precisely. So being out of it and being in it can be expressed with crisp values. Hence for this ENCZ kind of events μ_{run} is either 0 or 1.0. Similarly, EXCZR: $\{MOE_{t1}, MOE_{t2}\} \rightarrow$ (EXCZ, μ_{excz}) where MOE_{t1} for a vehicle has the spatial coordinates in the defined critical zone and MOE_{t2} for the same vehicle has the spatial coordinates out of a defined critical zone.

7.3.6.2 Formal Complex Event Definition

The complex events are built by correlating the simple events. In this dissertation, we define the complex events formally as follows:

- CER: SSE $\rightarrow$ (CE, μ_{CE}) is a **C**omplex **E**vent **R**ule where:

- SSE = {(SE_1, $F\mu_1$), (SE_2, $F\mu_2$), ..., (SE_n, $F\mu_n$)} is a finite **S**et of **S**imple **E**vents inferred previously by using simple event rules and each (SE_1, $F\mu_1$) pair is the simple event with its corresponding fuzzy membership value having the transition function such as "above some threshold" or "below some threshold" etc.

- CE is a **C**omplex **E**vent inferred by investigating the members of SSE depending on their semantic meanings.

- μ_{CE} is the fuzzy membership value of inferred event CE and calculated through a fuzzy membership function either based on 'Fmu_i's or independently.

For instance, a complex event of "ATTACK to the sink" can be defined with the proposed notation as follows:

ATHR: {(HR, $\mu_{run} > 0.5$), (ENCZ, 1.0)} $\rightarrow$ (ATH, μ_{ath}) where if there exists a human intruder RUNNING FAST and a vehicle ENTERS to a critical zone, then the rule concludes with the "ATTACK to the sink" having μ_{ath} as a fuzzy membership value calculated, i.e. based on the velocity of the vehicle.

Another complex event of "SMUGGLING from the critical zone" can be defined with the proposed notation as follows:

SMR: {(HR, $\mu_{run} < 0.5$), (EXCZ, 1.0)} $\rightarrow$ (SM, μ_{sm}) where if there exists a human intruder RUNNING SLOW and a vehicle EXITS from the critical zone which means they are taking out the goods secretly with the vehicle. Then the rule concludes with the "SMUGGLING from the critical zone" having μ_{sm} as a fuzzy membership value calculated, i.e. based on the velocity of the human beings.

7.4 Experimental Evaluation

In this section, we describe how such a system using WECA rules can be implemented in a power-plant surveillance application.

7.4.1 Power-Plant Surveillance Application with WECA Rules

A WMSN with a large number of simple cameras as well as sophisticated PTZ cameras can be very convenient to monitor the surroundings of a power-plant built in rural areas (see Figure 7.5 [WEB:PowerPlant]) against possible terrorist attacks, thefts or animal intrusions. We envision a continuous remote surveillance system which will last for several months if not years and trigger certain actions in an autonomous manner.

Figure 7.5: A sample power-plant surveillance area

In our sample power plant surveillance application, we assume that camera sensors may capture three different object types, namely *Human*, *Vehicle* and *Animal* which can possibly intrude in to different parts of the power-plant. Camera sensors also send the *Location* and *Velocity* of the detected objects. The definition of a MOE in our application is given as follows:

- ID of the camera sensor detecting the event
- Detection time
- Detected object
  - Location (X and Y coordinate)
  - Velocity

- Membership value of the object
  - for the class of 'Human'
  - for the class of 'Vehicle'
  - for the class of 'Animal'

Below we list some sample situations and discuss how they are expressed in WECA form.

7.4.1.1 Example-1: A Human Intruder is Detected by Two Different Camera Sensors at Similar Times

Consider an example active rule shown in Figure 7.6, expressed in natural language and defined for power plant surveillance:

```
A detected human intruder is approaching the transmission lines, and
if that human intruder is close enough to one of the camera sensors,
request original video frame from that camera to see the human int-
ruder better.
```

Figure 7.6: Active rule defined in natural language

Let us assume that this human intruder is detected by two different camera sensors, given that the number of cameras used is large. As a result, two different MOEs containing a human intruder object are sent to the sink by two camera sensors, namely $Camera\_Sensor\_1$ and $Camera\_Sensor\_2$. Let us assume that the objects detected are referred to as $H1$ or $H2$. Then, the WECA rule that needs to be defined at the sink is as shown in Figure 7.7. The WECA rule fires when a human intruder is detected (by at least two different camera sensors) (Phase-1) and approaches the transmission lines (Phase2). If the intruder is close enough to one of the camera sensors (Phase-3), the action to be taken is triggered, which is, in this case, requesting the original video frame from that camera sensor (Phase-4).

Now, let us look at each phase in detail:

In our work, fusion of two different MOEs sent by two different camera sensors is performed by using the similarity of MOEs based on their stored features, i.e., classification values and

```
WHEN (C1) detects human (H1)
     (C2) detects human (H2)
     at similar times and
     (H1) and (H2) are VERY SIMILAR
     and fused to (H)
ON   (H) is approaching one
     of the transmission lines
IF   (H) is close to a camera (C)
THEN Request video data from camera (C)
```

Figure 7.7: Corresponding WECA rule

their velocities. The total fuzzy similarity between the two MOEs is calculated based on two *sub_similarities*, namely *the object class similarity* and *the velocity similarity*. While the former compares the membership values, the latter compares the speed of objects. The speed is a reasonable choice for comparison given that the speed of an object would not change abruptly between consecutive frames. We assign weights to each of these *sub_similarity* classes. The weight of *the object class similarity* on total similarity is taken as 0.75 since it includes three fuzzy variables (i.e., human, animal and vehicle) out of four including the velocity. The weight of *the velocity similarity* is 0.25.

We now explain how Equation 7.1 and Figure 7.3 are used at the sink in Algorithm 3, *the Object Similarity Algorithm*. Assume that O_1 and O_2 are received by the sink from different sensors S_1 and S_2 at different times T_1 and T_2 where $T_2 > T_1$. Since we have three membership values and one velocity value, *the object class similarity* has the weight of 0.75 over *the total similarity*. If the rule has the keyword VERY SIMILAR, then the threshold value to check *the velocity similarity*, which we call W afterwards, becomes 0.675 ($W = 0.9 * 0.75 = 0.675$). If *the object class similarity* between the objects O_1 and O_2 is above 0.9 then there is a good chance that these two objects are VERY SIMILAR and at line 3, *the total similarity* becomes bigger than $W = 0.675$, then we need to check *the velocity similarity*. Once *the velocity similarity* is computed, it is multiplied by 0.25 and added to the total similarity. Thus, a final *total object similarity* is found at line 4. If the found *total object similarity* is above 0.9, then we may consider the gathered objects as VERY SIMILAR and fused to one object at the sink, meaning that two different cameras sense the same object at different times, T_1 and T_2.

Note that, while the objects may be very similar in their classes, their velocities may differ if

O_2 decreases its velocity physically when traveling between the location of O_1 and O_2 (i.e., it may follow a zigzag movement in a blind area). In such a case, *the velocity similarity* is less than 1.0 making the $Total\_Similarity$ parameter between 0.75 and 1.0. On the other hand if the same camera sensor sends two different MOEs containing different locations for the objects having the same classification value at the same time, the velocity is calculated as ∞ which makes *the velocity similarity* 0. Hence, even if *the object class similarity* is equal to 1.0, the system identifies two different objects, i.e., two different vehicles.

Hence, the MOEs sent by $Camera\_Sensor\_1$ and $Camera\_Sensor\_2$ in Figure 7.7 are fused with the help of Algorithm 3.

Algorithm 3 Object Similarity Algorithm at the Sink

```
Require: Two objects with their membership vectors and velocities exist.
Ensure: Total similarity value between two objects is found.
1: $Total\_Similarity \leftarrow 0$
2: $Total\_Similarity \leftarrow findObjectClassSimilarity(Object_1, Object_2) * 0.75$
3: if $Total\_Similarity \geq W$ then
4:     $Total\_Similarity += findObjectVelocitySimilarity(Object_1, Object_2) * 0.25$
5: end if
```

One final note is on the time of object detection. Obviously, the detection times of human intruders for $C1$ and $C2$ may not be exactly the same. We are interested in a more flexible "similar" time. To do this, the detection times in MOEs can be compared according to a specific application-dependent threshold, i.e., 2 *ms* in our case.

After the *WHEN* part of the WECA rule is processed, then event processing is performed. The event part (E) contains an *approaching* event which needs to check whether the distance between human's position and transmission line's location is decreasing between two consecutive MOEs. Note that transmission line's location and possible other important structures' locations should be supplied to the system in advance.

If the event is occurring, then the condition (C), which is *being close enough to a camera*, is checked. In order to check this, the location of the object and camera are needed to compute the distance. If that distance is below some predefined threshold, then the condition check is also positive.

If the condition is also met, then the action part (A) which is *requesting the video data from the camera* is handled. After requesting the raw video data, it is up to the application designers to use human intervention to see if there is something serious happening. Up to that point, the system reacts in an energy-efficient manner under uncertain/incomplete information.

7.4.1.2 Example-2: Two Human Intruders are Detected by Two Different Camera Sensors at Similar Times

Following the same event example above, let us assume that two MOEs detected by two camera sensors are not similar. Therefore, there are two different human intruders. Let us define two independent events for these intruders such that one of them approaches the transmission lines and the other one moves west which is the direction of the entrance gate. If these two events occur at similar times, we want to alarm the guards without checking any other conditions. As a result, the WECA rule will be defined as in Figure 7.8.

```
WHEN (C1) detects human (H1)
     (C2) detects human (H2)
     at similar times and
     (H1) and (H2) are NOT SIMILAR
ON   (H1) is approaching the
     transmission lines AND
     (H2) is moving west
THEN Send an alarm to the guards
```

Figure 7.8: WECA rule

7.4.1.3 Example-3: Two Human Intruders are Detected by a Single Camera Sensor

It may also be possible that these intruders are detected by the same camera sensor. Even if the same camera sensor senses these human intruders at similar times and sends two different MOEs containing two human objects, the rule defined in Figure 7.8 still works. $C1$ and $C2$ can be the same camera sensor which detects two objects on the same or consecutive frames. The camera sensor will send different locations for the two objects. Then the velocity is calculated as ∞ which makes *the velocity similarity* 0. Hence, even if *the object class similarity* is equal to 1.0 for them, the system identifies two different human beings for the same time instant.

7.4.2 The Rule Controller - JESS Implementation

In order to test the feasibility of WECA rules, we used JESS, the rule engine for the JAVA platform [WEB:Jess]. The reasons for using JESS instead of direct implementation in application code are similar to ones given for using ECA rules instead of application code in [Zoumboulakis04]. These reasons are listed as follows:

- JESS allows an application's reactive functionality to be specified and managed within a database/ knowledge base rather than being encoded in diverse programs, thus enhancing the modularity, maintainability and extensibility of applications.
- JESS has a high-level, declarative syntax and is thus amenable to analysis and optimization techniques which cannot be easily applied if the same functionality is expressed directly in application code.
- JESS has a mechanism that can abstract a wide variety of reactive behaviors, in contrast to an application code that is typically specialized to a particular kind of reactive scenario.

In our implementation, the rules created by the user are transformed into the objects and stored in a very simple object database. These rule-objects contain enumeration values for the predefined fusion parameters, event, condition and action types. When a rule is retrieved from the database, it is transformed into the JESS syntax and inserted into the JESS working memory.

In this part, a simple JESS example for a power plant surveillance application system will be given for the example WECA rule shown in Figure 7.7.

Each JESS rule engine holds a collection of knowledge nuggets called facts [WEB:Jess]. This collection is known as the working memory. Working memory is important because rules can only react to additions, deletions, and changes to working memory. Every fact has a template. The template has a name and a set of slots, and each fact gets these from its template. In Figure 7.9, Figure 7.10 and Figure 7.11, the template definitions for individual camera sensors, important structures and MOEs are given respectively.

```
(deftemplate SENSOR
  (slot name)
  (slot x_position)
  (slot y_position))
```

Figure 7.9: Template definition for sensors

```
(deftemplate IMPSTRCT
  (slot strct_name)
  (slot x_position)
  (slot y_position))
```

Figure 7.10: Template definition for important structure

```
(deftemplate MOE
  (slot object_name)
  (slot time)
  (slot sensor)
  (slot x)
  (slot y)
  (slot cls))
```

Figure 7.11: Template definition for MOEs

JESS rules have two parts, separated by the => symbol. The first part consists of the Left Hand Side pattern, the second part consists of the Right Hand Side action. We define the WHEN part (fusion part) of WECA rules as a sub-rule given in Figure 7.13. Notice that MOE template definition contains the part *object_name*. However, sensors do not know the identity information of the detected objects. Hence MOEs coming from camera sensors are fed into the JESS system after they are processed using Algorithm 3 and their identities are obtained in the architecture as shown in Figure 7.12.

A simple event for human beings is to approach important structures and is modified with the sub-rule given in Figure 7.14. In this simple event, if the same human object is sensed at

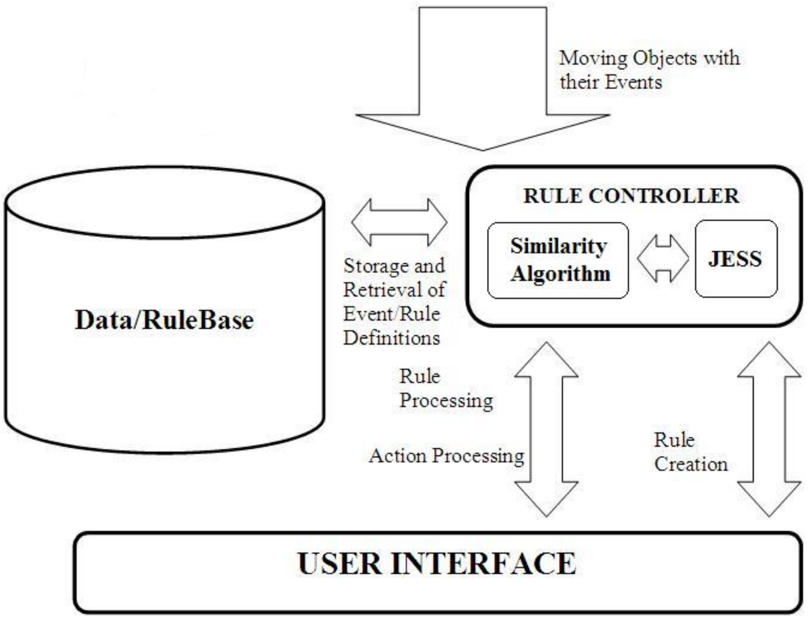


Figure 7.12: The architecture of the mechanism using JESS

```
(defrule TWO_SENSED
  (MOE (object_name ?O)
       (sensor_name ?x)
       (time ?t1))
  (MOE (object_name ?O)
       (sensor_name ?y&~?x)
       (time ?t1))
 =>
  (printout t "TWO SENSOR SENSE
     THE OBJECT " ?O " AT " ?t1 crlf)
  (assert (two_sensed ?O)))
```

Figure 7.13: Sub-Rule for fusion part

consecutive time intervals with decreasing distance to the important structure, such as transmission lines, then he/she is assumed to be approaching to that structure.

A simple condition of being close to a camera sensor is processed with the sub-rule given in Figure 7.15. In this condition, if the detected object and the camera show 5 meters difference between x and y positions, they are assumed to be close.

```
(defrule HUMAN_APPROACHING
  (IMPSTRCT (strct_name ?TL)
      (x_position ?x3)
      (y_position ?y3))
  (MOE (object_name ?O)
       (x ?x1)
       (y ?y1)
       (time ?t1)
       (cls human))
  (MOE (object_name ?O)
       (x ?x2&:(< (- ?x3 ?x2)
                  (- ?x3 ?x1)))
       (y ?y2&:(< (- ?y3 ?y2)
                  (- ?y3 ?y1)))
       (time ?t2&:(= ?t2 (+ ?t1 1)))
       (cls human))
  =>
  (printout t "HUMAN " ?O
           " APPROACHING " ?TL
           " AT " ?t2 crlf)
  (assert (human_approaching ?O ?TL ?t2)))
```

Figure 7.14: Sub-Rule for event part

```
(defrule NEAR_TO_SENSOR
  (MOE (object_name ?O)
       (x ?x1)
       (y ?y1)
       (time ?t1))
  (SENSOR (sensor_name ?s)
       (x_position ?x2&:(< (- ?x2 ?x1) 5))
       (y_position ?y2&:(< (- ?y2 ?y1) 5))
       (time ?t1))
  =>
  (printout t OBJECT ?O
           IS NEAR CAMERA SENSOR ?s
           " AT " ?t1 crlf)
  (assert (near_sensor ?O ?s ?t1)))
```

Figure 7.15: Sub-Rule for condition part

The final rule uses all of the sub rules' results and ensures that an object is detected by two different camera sensors and that the object is human and approaching the important structure. When that object is near one of the cameras, then the system requests original raw video data from that camera. Hence, instead of gathering raw video data from camera sensors at all times, the system requests it only when the intruder is performing a predefined event with the predefined conditions. The final rule is shown in Figure 7.16.

```
(defrule REQUEST_ORIGINAL_FRAME
        (two_sensed ?O)
        (human_approaching ?O ?TL ?t)
        (near_sensor ?O ?s ?t)
   =>
  (printout t "REQUEST ORIGINAL FRAME
       FROM " ?s " AT " ?t crlf))
   (request_original_frame ?s)
```

Figure 7.16: Final rule

After defining the templates and the rules for JESS, the JESS system is ready to accept the facts which are the real-word instances occurring in real time. Figure 7.17 shows some example facts for the rules given in previous Figures.

The result of JESS against the definitions and facts inserted into the working memory is shown in Figure 7.18. At time $T2$, the whole rule fires and the system requests the original video data from the camera sensor $S2$ in order to inspect the intruder appropriately.

7.4.3 Experimental Evaluation of Fusion

To test the object fusion performance of WECA rules (i.e., the *WHEN* part), we have used the simulator for the monitoring of our power-plant surveillance application. An area of 1000X600 m^2 is considered. For each group of experiments, different numbers of camera sensors are placed randomly with random orientations in order to achieve the desired coverage values as shown below. Coverage values are calculated in terms of total FoV with respect to the total area.

```
(assert (SENSOR (sensor_name S1)
    (x_position 40) (y_position 28)))
(assert (SENSOR (sensor_name S2)
    (x_position 19) (y_position 30)))
(assert (IMPSTRCT
    (strct_name Trnsm_Lines)
    (x_position 20) (y_position 24)))

(assert (MOE (object_name O1) (sensor S1)
     (time 1) (x 10) (y 20) (cls human)))
(assert (MOE (object_name O1) (sensor S1)
     (time 2) (x 16) (y 21) (cls human)))
(assert (MOE (object_name O1) (sensor S2)
     (time 2) (x 16) (y 21) (cls human)))
```

Figure 7.17: Facts

```
HUMAN O1 APPROACHING Trnsm_Lines AT 2
OBJECT O1 IS NEAR SENSOR S2 AT 2
TWO SENSOR SENSE THE OBJECT O1 AT 2
REQUEST ORIGINAL FRAME FROM S2 AT 2
```

Figure 7.18: Output

- 25% coverage: 29 camera sensors
- 50% coverage: 61 camera sensors
- 75% coverage: 102 camera sensors
- 100% coverage: 120 camera sensors

Note that these cases also cover the effect of any possible obstacles in the environment. In case of obstacles, depending on the number of such obstacles, the coverage values may decrease slightly. In addition to obstacles, in some cases, there may not be sufficient cameras to monitor the whole region. Hence, to account for such cases, we have considered coverage values from 25% to 100%.

Transmission ranges and DoFs are picked as 200m and 100m respectively. The sink node is assumed to be placed at (0, 0) location. Objects are generated randomly and their membership values and velocities are also assigned randomly. In each individual experiment, objects have special mobility pattern designed for these experiments. Each object has the purpose to enter the area and move towards the main building as if it was an intruder aiming to capture an important structure positioned at the building. Membership values for a random object, (i.e., for a human intruder is a vector of (a, b, c)), are calculated as $a > 0.5$, $b + c = 1 - a$ where a, b and c are random numbers. For instance, for 6 different objects of 2 humans, 2 vehicles and 2 animals, the membership values produced with this formula are as given in Table 7.1.

The metric that we have used in our experiment is Fusion Accuracy. This metric indicates how successful the algorithm is when identifying the moving objects. It is defined as follows: At each time unit, the real object count under surveillance and the calculated object count by the sink are compared. If these values are not equal to each other, the ratio of the difference of these values to the real object count is noted as a *false ratio*. $(1 - false\ ratio)$ is defined as the fusion accuracy. Note that the lifetime threshold defined before prevents the false ratio from being less than 0 or bigger than 1. Our goal is to maximize the accuracy.

Table 7.1: Random Membership Values for 6 Objects

Object Name	**Membership Value**
Human 1	(0.86; 0.04; 0.10)
Human 2	(0.93; 0.00; 0.07)
Vehicle 1	(0.14; 0.61; 0.25)
Vehicle 2	(0.03; 0.87; 0.10)
Animal 1	(0.12; 0.08; 0.80)
Animal 2	(0.13; 0.27; 0.60)

In order to observe the fusion accuracy, we have set up experiment groups that include 3, 6, 9 or 12 different moving objects. Each experiment group is performed under 4 different camera sensor coverage values given in the Experiment Setup subsection. The change of the average fusion accuracy over time for 3 different moving objects for different coverage values is shown in Figure 7.19. Overall accuracies for all coverage values and for all object numbers are presented in Table 7.2. We note that the results included here are recorded between the time interval [40,160], since, on average, a random moving object enters into one of the camera's FoV at time 40 and begins to disappear from the region at time 160.

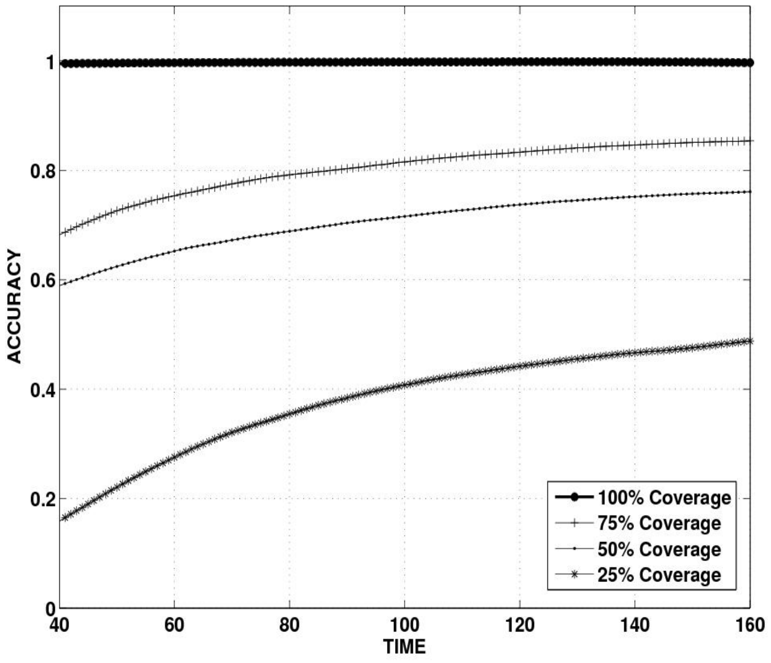


Figure 7.19: Change of fusion accuracy values over time for experiments with 3 different objects on all camera coverage values.

Table 7.2: Average Fusion Accuries

Object Count	**25% Cov.**	**50% Cov.**	**75% Cov.**	**100% Cov.**
3 Objects	0.38	0.71	0.81	1.0
6 Objects	0.38	0.74	0.81	1.0
9 Objects	0.39	0.73	0.83	1.0
12 Objects	0.39	0.75	0.83	1.0

In optimistic scenarios, we target the fusion accuracy value to be at least as much as the total camera coverage percentage. For instance, for 25% camera coverage value, one should achieve at least 25% fusion accuracy as the sink can monitor only 25% part of the whole area and 75% of the area is the blind area. The results of our experiments show that, under optimistic scenarios, the fusion accuracies are higher than these base coverage values.

The results also indicate that when the coverage is 100%, our approach guarantees accurate tracking of any number of objects. With the decreasing coverage percentage value, the accuracy decreases. However, this decrease is not proportional to the decrease in coverage value, indicating the effectiveness of our approach. In all coverage cases, the fusion accuracies are always higher than the coverage percentage values.

7.4.4 Experiments for Complex Event Processing

In this section, by using a typical border surveillance application we give a proof of concept about our complex event processing solution. Assume a border surveillance area as shown in Figure 7.20. Consider two example complex events shown in Figures 7.21 and 7.22, expressed in natural language and defined for this application.

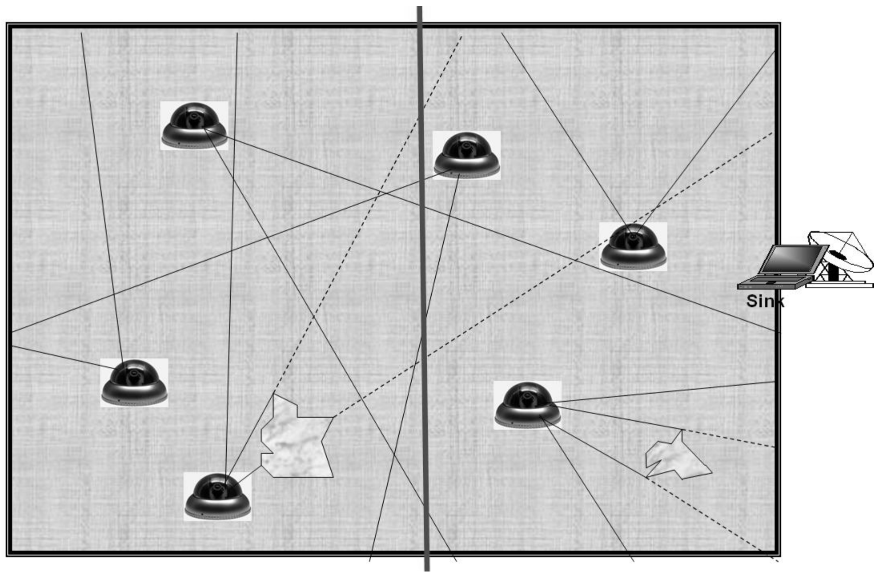


Figure 7.20: Typical border surveillance area where the left of the vertical line passes from the middle is a critical area for the sink

```
A human intruder is detected as RUNNING FAST and a vehicle is
detected as ENTERING TO THE CRITICAL ZONE then ALARM the guards
as ATTACK is happening
```

Figure 7.21: Complex event of ATTACK

Since these two complex events defined above are build on top of simple events, i.e., HUMAN RUNNING and ENTERING/EXITING to/from critical zone, the ON part of WECA rule is used for combining those simple events. WHEN part is useless for defining complex events

```
A human intruder is detected as RUNNING SLOW and a vehicle is
detected as EXITING FROM THE CRITICAL ZONE then ALARM the guards
as SMUGGLING is happening
```

Figure 7.22: Complex event of SMUGGLING

because there exist no need for fusion at this stage. The IF part is used to detect any feature defined in simple events and THEN part is used to infer actions when the rule fires. Two WECA rules defined for the complex events above are shown in Figures 7.23 and 7.24.

```
ON E1: HUMAN (H) is RUNNING
   and E2: VEHICLE (V) is ENTERING TO THE CRITICAL ZONE
IF  E1 is FAST
THEN GIVE ATTACK ALARM
```

Figure 7.23: WECA rule of ATTACK

```
ON E1: HUMAN (H) is RUNNING
   and E2: VEHICLE (V) is EXITING FROM THE CRITICAL ZONE
IF  E1 is SLOW
THEN GIVE SMUGGLING ALARM
```

Figure 7.24: WECA rule of SMUGGLING

In order to implement the two WECA rules given above, firstly template definitions for the FACTS should be defined in JESS working memory. Template definitions for MOE and Critical Zone are shown in Figures 7.25 and 7.26. The critical zone only needs X coordinate of the vertical line in this example scenario. Of course, more complex critical zones such as rectangle areas or circle areas in the area under surveillance can be defined according to the application.

Before complex events, the simple events are firstly extracted from the coming MOEs from the camera sensors. When they are extracted, they are also added to the JESS working memory to be used in complex event processing. For that purpose, the JESS needs template definitions

```
(deftemplate MOE
    (slot object_name)
    (slot time)
    (slot sensor)
    (slot x)
    (slot y)
    (slot v)
    (slot cls))
```

Figure 7.25: Template definition for MOE

```
(deftemplate Critical_Zone
    (slot x))
```

Figure 7.26: Template definition for critical zone

for the simple events. In Figures 7.27, 7.28 and 7.29, the template definitions for simple events of “Human Running”, “Entering to the Critical Zone” and “Exiting from the Critical Zone” are shown respectively.

```
(deftemplate Human_Running
    (slot object_name)
    (slot fast)
    (slot slow)
    (slot time))
```

Figure 7.27: Template definition for a simple event of “Human Running”

```
(deftemplate Enter_Critical_Zone
    (slot object_name)
    (slot time)
    (slot cls))
```

Figure 7.28: Template definition for a simple event of “Entering to the Critical Zone”

```
(deftemplate Exit_Critical_Zone
    (slot object_name)
    (slot time)
    (slot cls))
```

Figure 7.29: Template definition for a simple event of “Exiting from the Critical Zone”

The fuzzy membership function used for the RUNNING event that takes the velocity of the human beings as input and produces the fuzzy value of being FAST or SLOW is shown in Figure 7.30. This fuzzy membership graph has three regions where the first one only produces SLOW fuzzy membership value as 1.0, the second one (which is the middle region) produces values for both FAST and SLOW fuzzy membership values, and lastly the third one only produces FAST fuzzy membership value as 1.0. For that reason, three different rules that include the appropriate fuzzy membership value calculations for HUMAN RUNNING events are shown in Figures 7.31, 7.32 and 7.33.

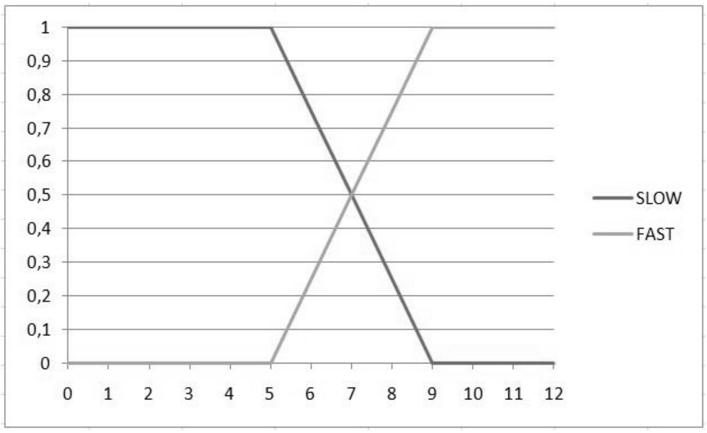


Figure 7.30: The fuzzy membership function for FAST and SLOW based on velocity

“Human Running FAST” event just calculates the fuzzy membership value for being FAST as shown in Figure 7.31. For that reason, there is no SLOW fuzziness value calculation in this rule.

The JESS rules that implements the simple events of “Entering to a Critical Zone” and “Exiting from a Critical Zone” are shown in Figures 7.34 and 7.35 respectively. These two events

```
(defrule FAST_RUN_HUMAN
 (MOE (object_name ?O)
    (v ?a&:(>= ?a 9))
    (time ?t)
    (cls human))
 =>
 (printout t ?O " is Running VERY FAST at t" ?t crlf)
 (assert (Human_Running (object_name ?O)
                   (fast 1.0)(slow 0.0)(time ?t)))
)
```

Figure 7.31: Rule definition for a simple event of “Human Running FAST”

```
(defrule SLOW_RUN_HUMAN
 (MOE (object_name ?O)
     (v ?a&:(<= ?a 5))
     (time ?t)
     (cls human))
 =>
 (printout t ?O " is Running VERY SLOW at t" ?t crlf)
 (assert (Human_Running (object_name ?O)
                   (fast 0.0)(slow 1.0)(time ?t)))
)
```

Figure 7.32: Rule definition for a simple event of “Human Running SLOW”

are produced by checking the object spatial features. When the object X coordinate at time $t1$ is out of critical zone and at time $t2$ in the critical zone (where $t2 > t1$) then this means that the object is entering to the critical zone. “Exiting from a Critical Zone” event definition is reverse of the “Entering to a Critical Zone” event definition.

The JESS implementations for the complex events of “ATTACK” and “SMUGGLING” are shown in Figures 7.36 and 7.37 respectively. In “ATTACK” event, being FAST is checked by having FAST fuzzy membership value above 0.5, because according the fuzzy membership graph given in Figure 7.30, the sum of fuzzy membership values of being FAST and SLOW equals to 1.0. If the FAST membership value is above 0.5 then this means that fuzzy membership value for SLOW is less than 0.5.

```
(defrule FAST_SLOW_RUN_HUMAN
 (MOE (object_name ?O)
     (v ?a&:(and (> ?a 5) (< ?a 9)))
     (time ?t)
     (cls human))
 =>
 (printout t "Object " ?O " is Running"
             ":SLOW " (+ (* ?a -0.25) 2.25)
             " - FAST " (+ (* ?a 0.25) -1.25)
             " at t" ?t crlf)
 (assert (Human_Running (object_name ?O)
           (fast (+ (* ?a 0.25) -1.25))
           (slow (+ (* ?a -0.25) 2.25))
           (time ?t)))
)
```

Figure 7.33: Rule definition for a simple event of “Human Running either SLOW or FAST”, fuzzy calculation

```
(defrule Enter_Critical_Zone
 (Critical_Zone (x ?a))
 (MOE (object_name ?O)
      (x ?b&:(> ?b ?a))
      (time ?t1)
      (cls ?c))
 (MOE (object_name ?O)
      (x ?C&:(< ?C ?a))
      (time ?t2&:(= ?t2 (+ ?t1 1)))
      (cls ?c))
 =>
 (printout t "Object " ?O "-" ?c " enters zone at t" ?t2 crlf)
 (assert (Enter_Critical_Zone (object_name ?O)
                 (cls ?c) (time ?t2)))
 )
```

Figure 7.34: Rule definition for a simple event of “Entering to a Critical Zone”

In JESS the FACTS are injected to the system and the rules fire according to these FACTS. In Figure 7.38, firstly the critical zone line coordinate is injected to the system. Then MOEs are injected to the system one by one. At each time interval, there exist two different objects exist at the area under surveillance. One of them is a human being, the other one is a vehicle.

```
(defrule Exit_Critical_Zone
 (Critical_Zone (x ?a))
 (MOE (object_name ?O)
      (x ?b&:(< ?b ?a))
      (time ?t1)
      (cls ?c))
 (MOE (object_name ?O)
      (x ?C&:(> ?C ?a))
      (time ?t2&:(= ?t2 (+ ?t1 1)))
      (cls ?c))
 =>
 (printout t "Object " ?O "-" ?c " exits zone at t" ?t2 crlf)
 (assert (Exit_Critical_Zone (object_name ?O)
                 (cls ?c)(time ?t2)))
)
```

Figure 7.35: Rule definition for a simple event of “Exiting from a Critical Zone”

```
(defrule ATTACK
 (Human_Running (object_name ?O1)
                (fast ?x&:(> ?x 0.5))
                (time ?t))
 (Enter_Critical_Zone
                (object_name ?O2)
                (cls vehicle)
                (time ?t))
 =>
 (printout t ?O1 " is running FAST and "
 ?O2  " enters Critical Zone: "
 "ATTACK ALARM AT t" ?t " " ?x crlf)
)
```

Figure 7.36: Rule definition for a complex event of “ATTACK”

“(run)” is a command for JESS to process the injected FACT by “(assert)” command. In each MOE, only X coordinates for the objects are given for simplicity, since the example rules defined above are only interested in X coordinates of the objects.

The outputs of JESS for the FACTS injected in Figure 7.38 are shown in Figure 7.39. At

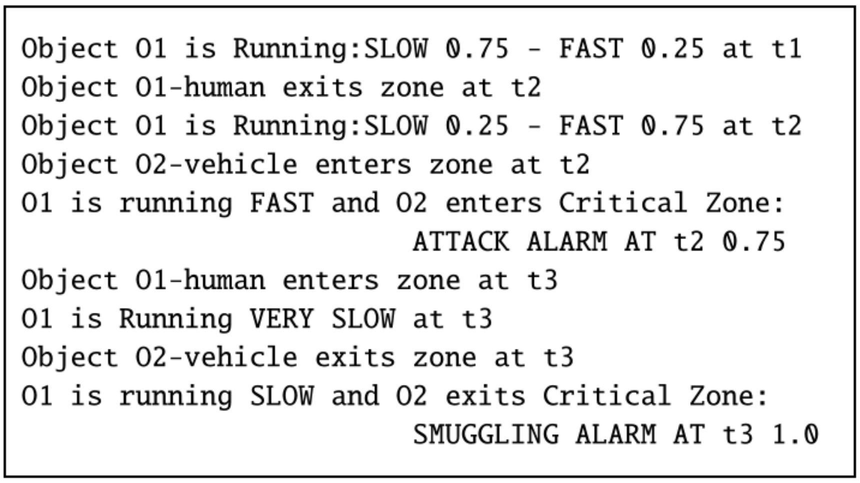
```
(defrule SMUGGLING
 (Human_Running (object_name ?O1)
                (slow ?x&:(> ?x 0.5))
                (time ?t))
 (Exit_Critical_Zone
                (object_name ?O2)
                (cls vehicle)
                (time ?t))
 =>
 (printout t ?O1 " is running SLOW and "
 ?O2  " exits Critical Zone: "
 "SMUGGLING ALARM AT t" ?t " " ?x crlf)
)
```

Figure 7.37: Rule definition for a complex event of “SMUGGLING”

```
(assert (Critical_Zone (x 78))) (run)
(assert (MOE (object_name O1) (x 73) (v 6)
             (time 1) (cls human))) (run)
(assert (MOE (object_name O2) (x 80) (v 30)
             (time 1) (cls vehicle))) (run)
(assert (MOE (object_name O1) (x 79) (v 8)
             (time 2) (cls human))) (run)
(assert (MOE (object_name O2) (x 50) (v 30)
             (time 2) (cls vehicle))) (run)
(assert (MOE (object_name O1) (x 75) (v 5)
             (time 3) (cls human))) (run)
(assert (MOE (object_name O2) (x 90) (v 40)
             (time 3) (cls vehicle))) (run)
```

Figure 7.38: Facts

time *t*1, human object O1 is detected as RUNNING SLOW with the probability of 0.75. At time *t*2, human object O1 is detected as RUNNING FAST with the probability of 0.75 and a vehicle ENTERS to the CRITICAL ZONE which means an ATTACK occurs at time *t*2. The probability of the ATTACK is found as 0.75. At time *t*3, human object O1 is detected as RUNNING VERY SLOW and a vehicle EXITS from the CRITICAL ZONE which means SMUGGLING occurs at time *t*3 with the probability of 1.0.



```
Object O1 is Running:SLOW 0.75 - FAST 0.25 at t1
Object O1-human exits zone at t2
Object O1 is Running:SLOW 0.25 - FAST 0.75 at t2
Object O2-vehicle enters zone at t2
O1 is running FAST and O2 enters Critical Zone:
                         ATTACK ALARM AT t2 0.75
Object O1-human enters zone at t3
O1 is Running VERY SLOW at t3
Object O2-vehicle exits zone at t3
O1 is running SLOW and O2 exits Critical Zone:
                         SMUGGLING ALARM AT t3 1.0
```

Figure 7.39: Outputs

7.5 Summary

In this chapter, we have proposed an active rule processing mechanism for a typical power-plant surveillance application that uses WMSNs. Instead of collecting the raw video data all the time, the sink gathers incomplete and imprecise moving object event information about the objects from different camera sensors only when the camera sensors detect an event. The sink then processes the received data using predefined active rules. For this purpose, we extend ECA rules according to the needs of WMSNs and propose WECA rules. The sink performs four phases for rule-based processing after getting multimedia data from different camera sensors: Fuzzy Object Fusion, Fuzzy Event Identification, Fuzzy Condition Matching and Fuzzy Inference. Data fusion is done via utilizing fuzzy logic and can thus capture uncertainty in the data.

The feasibility of the rule-based mechanism has been shown via a JESS implementation on the power-plant surveillance application. In addition, we have conducted experiments for assessing the accuracy of the proposed fusion approach. The experiment results show that acceptable accuracy values can be obtained via the proper processing of incomplete and uncertain information using fuzzy logic.

CHAPTER 8

CONCLUSION AND FUTURE WORK

A lot of new challenges emerge with the introduction of WMSNs in different applications. These are the applications where the frequency of incidents is too low to employ permanent staffing such as border surveillance or monitoring of critical infrastructures positioned desolate places, etc. Moreover, in such applications, large number of battery operated image/video camera sensors are deployed to capture the occurring events. These cameras are typically low-resolution and severely resource constrained in terms of CPU, memory and storage which makes them cheaper and convenient to be deployed. This dissertation proposes a remote surveillance reactive framework with three components: 1) Localization and tracking of objects; 2) Classification and identification of objects; and 3) Reactive behavior at the sink.

In this dissertation, we first present a novel object localization approach which can work on-site at a camera sensor without the need for collaboration with its neighbors. The localization approach proposed utilizes frame differencing in order to extract the object from the frame. Once the object is detected, its distance to the camera sensor is estimated using the size of the object on the frame and on the camera sensor. The distance is then used to determine the exact location of the object at the area under surveillance, by using the orientation of the camera sensor. The experimental evaluation has revealed that, our approach can effectively determine the location of an object with an error rate of 2% in the worst case. We also assess the energy overhead of our approach on individual camera sensors. The energy consumption is significantly reduced compared to the case, where the localization is performed at the sink due to the communication overhead.

Then, we present a lightweight and accurate object classification approach which can work on-site again at the camera sensor with limited information. The classification approach utilizes genetic algorithm to built classifiers. The experimental evaluation for classification reveals that our classification approach can effectively classify typical objects, i.e., human, animals and vehicles, in a power-plant surveillance application with an error rate of at most 2% overall. We also assess the energy and storage overhead of our approach on individual camera sensors. Like in localization problem, the energy consumption is also reduced compared to the traditional case. Additionally, storage overhead is not significant compared to the available space on a typical camera sensor.

Afterwards, we investigate a multiple object tracking problem in a typical border surveillance application of WMSNs using minimal resources. Instead of raw frame or raw video data with huge size, the classification information along with the location of the object is sent to the sink as part of the MOE for tracking purposes. The sink gathers MOEs from different camera sensors and processes them using a fuzzy object similarity and tracking algorithms. In this way, the sink is able to differentiate and identify distinct moving objects under mostly blinded areas and track them in real-time. The tracking algorithm proposed is implemented and tested under variety of conditions. The experiments results show that the algorithm can accurately identify and track multiple moving objects even under low area coverage percentages. The results also indicate that this accuracy is slightly degraded with the introduction of errors (classification errors, localization errors or network packet losses). Nonetheless, in all cases, this process of classification and tracking performs with significantly less energy consumption than is needed in the traditional approaches, and as a result, increases the battery lifetime of the camera sensors.

Lastly, we propose an active rule processing mechanism for a typical power-plant surveillance application that uses WMSNs. Instead of collecting the raw video data all the time, the sink gathers incomplete and imprecise moving object event information about the objects from different camera sensors only when the camera sensors detect an event. The sink then processes the received data using predefined active rules. For this purpose, we extend ECA rules according to the needs of WMSNs and propose WECA rules. The sink performs four phases for rule-based processing: Fuzzy Object Fusion, Fuzzy Event Identification, Fuzzy Condition Matching and Fuzzy Inference. Data fusion is done via utilizing fuzzy logic and can thus capture uncertainty in the data. Simple events and complex events are all extracted within the proposed framework. The feasibility of the rule-based mechanism is shown via a JESS implementation on the power-plant surveillance application. In addition, we conduct experiments for assessing the accuracy of the proposed fusion approach. The experiment results show that acceptable accuracy values can be obtained via the proper processing of incomplete and uncertain information using fuzzy logic.

Future works for this dissertation can be listed as follows:

1. The object detection approach used in this dissertation cannot deal with groups of animals/people when they stick very close to each other, as this is a different and challenging problem which has been also a research issue in computer vision. For instance, [Mckenna00] points out that robust tracking of individuals in a group of people requires person models that must cope with reasonable changes in illumination and varied clothing. Especially in outdoor surveillance, it is not reasonable to assume that people wear tightly fitting clothes or these clothes are uniformly colored. Hence, when they are very close to each other, it is not an easy task to differentiate individuals in the group. This is a more challenging task in our case, where inexpensive and simple cameras are used. For the case of animal groups, since animals do not wear any clothes or do not have any dominant color differences between each other, differentiating models based on color changes will not work appropriately. Introducing new lightweight detection algorithms dealing with those problems is a challenging research area for the camera sensors.

2. Without increasing the energy consumption much at the camera sensors, increase at the number of features used in classification to further improve the accuracy and be able to work with more object classes, is another research area. For that purpose, new features should be extracted with low processing from the raw video by using low energy usage. Additionally, the extraction process should be time efficient such that the classification information is used in real-time surveillance applications.

3. One may think that, it is easy for a target to compromise the classification system. For example, by changing body shape or moving speed suddenly, the object similarity algorithm would not work. A target/intruder may even compromise a system that has strong and powerful cameras (and thus with powerful image processing algorithms) by wearing animal costumes or changing body shape etc. However, in order to cheat the system, the intruder should know about algorithm details, such as fuzzy sets defined in the algorithm. In addition, the system can be equipped with some strong rules to trigger

action against such targets. Extension of the classification and active rule processing approaches to deal with cases of cheating by intruders (e.g., a human intruder trying to behave (walk) like animals) is another challenging future work for this dissertation.

4. The WMSN used in this dissertation is a single-tier flat and contains homogeneous camera sensors. For this reason, some problems such as object detection or classification have limitations with the usage of energy limited camera sensors. Every algorithm that is proposed should be energy efficient in order to work on these camera sensors. However, in the literature, WMSNs with heterogeneous sensors are also proposed in the reference architecture, [Akyildiz07]. Improving the proposed algorithms in this dissertation by using the information gathered from other types of sensor, e.g., scalar sensors or radar sensors, besides camera sensors is a research direction after this dissertation.

5. In this dissertation, we do not try to estimate the future concepts such as possible future path of the objects while tracking them or possible next events while extracting the current ones. These kind of predictions would stronger the active framework while taking precautions about the intruders. Making such future predictions is another research area after this dissertation.

REFERENCES

[Adali96] S. Adalı, K. S. Candan, S. Chen, K. Erol and V. S. Subrahmanian, “The Advanced Video Information System”, Data Structures and Query Processing, Multimedia Systems, vol.4, pp.172-186, 1996

[Aiken95] A. Aiken, J. Hellerstein and J.Widom, “Static Analysis Techniques for Predicting the Behavior of Active Database Rules”, ACM TODS, no.1, vol.20, pp.3-41, 1995

[Akkaya08] K. Akkaya, M. Demirbas and R. S. Aygun, “The Impact of Data Aggregation on the Performance of Wireless Sensor Networks”, Wireless Comm. & Mobile Computing Journal, vol.8, pp.171-193, 2008

[Akkaya09] K. Akkaya and A. Newell, “Self-Actuation of Camera Sensor for Redundant Data Elimination In Wireless Multimedia Sensor Networks”, IEEE International Conference on Communications (ICC’09), pp.133-137, 2009

[Aksoy05] D. Aksoy, “Information Source Selection for Resource Constrained Environments”, ACM SIGMOD Record, vol.34/4, 2005

[Akyildiz02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, “Wireless Sensor Networks: a Survey”, Computer Networks, vol.38, pp.393-422, 2002

[Akyildiz04] I. F. Akyildiz and I. H. Kasimoglu, “Wireless Sensor and Actor Networks: Research Challenges”, Elsevier Ad hoc Network Journal, vol.2, pp.351-367, 2004

[Akyildiz07] I. F. Akyildiz, T. Melodia and K. R. Chowdhury, “A Survey on Wireless Multimedia Sensor Networks”, Computer Networks Elsevier, no.4, vol.51, pp.921-960, 2007

[Akyildiz08] I. F. Akyildiz and K. R. Chowdhury, “Wireless Multimedia Sensor Networks: Applications and Testbeds”, In Proceedings of the IEEE, no.10, vol.96, pp.1588-1605, 2008

[Alaei09] M. Alaei and J. M. Barcelo-Ordinas, “A Cluster-based Scheduling for Object Detection in Wireless Multimedia Sensor Networks”, Proceedings of International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, pp.50-56, 2009

[Arth07] C. Arth, C. Leistner and H. Bishof, “Object Reacquisition and Tracking in Large-Scale Smart Camera Networks”, Proc. of 1st AC/IEEE Int. Conf. on Smart Distributed Cameras, 2007

[Balasubramanian07] S. Balasubramanian and D. Aksoy, “Adaptive Energy-Efficient Registration and Online Scheduling for Wireless Sensor Networks”, Elsevier Computer Network Journal, vol.51, pp.3427-3447, 2007

[Ballan11] L. Ballan, M. Bertini, A. D. Bimbo, L. Seidenari and G. Serra, “Event Detection and Recognition for Semantic Annotation of Video”, no.1, vol.51, pp.279-302, 2011

[Bay08] H. Bay, A. Ess, T. Tuytelaars and L. V. Gool, “Speeded-Up Robust Features (SURF)”, Comput. Vis. Image Underst., no.3, vol.110, pp.346-359, 2008

[Black04] J. Black, T. Ellis and D. Makris, “A Hierarchical Database for Visual Surveillance Applications”, IEEE ICME, pp.1571-1574, 2004

[Bober01] M. Bober, “MPEG-7 Visual Shape Descriptors”, IEEE Transactions on Circuits and Systems for Video Technology, no.6, vol.11, pp.716-719, 2001

[Bogomolov03] Y. Bogomolov, G. Dror, S. Lapchev, E. Rivlin, and M. Rudzsky, “Classification of Moving Targets Based on Motion and Appearance”, British Machine Vision Conference, pp.429-438, 2003

[Bovik00] A. Bovik, “Handbook of Image and Video Processing”, Publisher: Academic Press, 2000

[Bramberger06] M. Bramberger, A. Doblander, A. Maier, A. Rinner and H. Schwabach, “Distributed Embedded Smart Cameras for Surveillance Applications”, Computer Journal, no.2, vol.39, pp.68, 2006

[Brooks03] R. R. Brooks, P. Ramanathan and A. M. Sayeed, “Distributed Target Classification and Tracking in Sensor Networks”, Proceedings of the IEEE, pp.1163-1171, 2003

[Ceri94] S. Ceri and J. Widom, “Deriving Incremental Production Rules for Deductive Data, Information Systems”, Information Systems, vol.16, 1994

[Chaudhari08] N. S. Chaudhari and A. Ghosh, “Feature Extraction Using Fuzzy Rule Based System”, International Journal of Computer Science and Applications, no.3, vol.5, pp.1-8, 2008

[Chen11] J. Chen, S. Ravizza, J. A. D. Atkin and P. Stewart, “On the Utilisation of Fuzzy Rule-Based Systems for Taxi Time Estimations at Airport”, ATMOS-11, pp,134-135, 2011

[Clouqueur01] T. Clouqueur, P. Ramanathan, K. Saluja and K.C.Wang, “Value-fusion versus Decision-fusion for Fault-tolerance in Collaborative Target Detection in Sensor Networks”, 4th Int. Conf. Information Fusion, 2001

[Coifman98] B. Coifman, “Vehicle Reidentification and Travel Time Measurement in Real-time on Freeways Using the Existing Loop Detector Infrastructure”, Transportation Research Board, 1998

[Collins01] R. Collins, A. Lipton, H. Fujiyoshi and T.Kanade, “Algorithms for Cooperative Multi-sensor Surveillance”, Proceedings of the IEEE, no.10, vol.89, 2001

[Comaniciu00] D. Comaniciu, V. Ramesh and P. Meer, “Real-Time Tracking of Non-Rigid Objects using Mean Shift”, In Proceedings of IEEE CVPR, vol.2, pp.142-149, 2000

[Datta05] R. Datta, J. Li and J. Z. Wang, “Content-based Image Retrieval: Approaches and Trends of the New Age”, MIR ’05: ACM Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval, pp.253-262, 2005

[Dong06] L. Dong, D. Wang and H. Sheng, “Design of RFID Middleware Based on Complex Event Processing”, IEEE Conference on Cybernetics and Intelligent Systems, pp.1-6, 2006

[Duda01] R. O. Duda, P. E. Hart and D. G. Stork, “Pattern Classification”, Publisher: Wiley-Interscience, ISBN:0-471-05669-3, 2001

[Dunkel09] J. Dunkel, “On Complex Event Processing for Sensor Networks”, International Symposium on Autonomous Decentralized Systems, ISADS09, pp.1-6, 2009

[Erdem06] U. M. Erdem and S. Sclaroff, “Automated Camera Layout to Satisfy Task Specific and Floor Plan-specific Coverage Requirements”, Computer Vision and Image Understanding, pp.156-169, 2006

[Etzion10] O. Etzion and P. Niblett, “Event Processing in Action”, Manning Publications, 2010

[Fei06] L. Fei-Fei, R. Fergus and P. Perona, “One-shot Learning of Object Categories”, IEEE Trans. Pattern Anal. Mach. Intell., no.4, vol.28, p.594, 2006

[Fernandez12] F. Fernandez and F. Chavez, “Fuzzy Rule Based System Ensemble for Music Genre Classification”, EvoMUSART, pp.84-95, 2012

[Flatin07] J.P. Martin-Flatin, G. Jakobson and L. Lewis, “Event Correlation in Integrated Management: Lessons Learned and Outlook”, Journal of Network and Systems Management, no.4, vol.17, 2007

[Garofalakis02] M. Garofalakis, Y. Ioannidis, B. Özden and A. Silberschatz, “Competitive On-line Scheduling of Continuous-Media Streams”, Journal of Computer and Systems Sciences, vol.64/2, pp.219-248, 2002

[Gehrke04] J. Gehrke and S. Madden, “Query Processing in Sensor Networks”, IEEE Pervasive Computing, no.1, vol3, pp.46-55, 2004

[Ghanem93] N. Ghanem, D. DeMenthol, D. Doermann and L. Davis, “Representation and Recognition of Events in Surveillance Video Using Petri Nets”, Conference on Computer Vision and Pattern Recognition Workshop, vol.7, pp.23-39, 1993

[Giambiasi06] N. Giambiasi and J.C. Carmona, “Generalized Discrete Event Abstraction of Continuous Systems: GDEVS Formalism”, Simulation Modelling Practice and Theory, no.1, vol.14, pp.47-70, 2006

[Guo05] Y. Guo, S. C. Hsu, Y. Shan, H. S. Sawhney and R. Kumar, “Vehicle Fingerprinting for Reacquisition and Tracking in Videos”, CVPR: In Conference on Computer Vision and Pattern Recognition, vol.2, pp.761-768, 2005

[Hakeem05] A. Hakeem and M. Shah, “Multiple Agent Event Detection and Representation in Videos”, Proceedings of the 20th national conference on Artificial intelligence, vol.1, 2005

[Hamdoun08] O. Hamdoun, F. Moutarde, B. Stanciulescu and B. Steux, “Person Re-identification in Multi-camera System by Signature based on interest point descriptors collected on short video sequences”, ICDSC 2008: Second ACM/IEEE International Conference on Distributed Smart Cameras, pp.1-6, 2008

[Hampapur05] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl and S. Pankanti, "Smart Video Surveillance: Exploring the Concept of Multiscale Spatiotemporal Tracking", IEEE Signal Processing Magazine, no.2, vol.22, pp.38-51, 2005

[Hengstler07] S. Hengstler, D. Prashanth, Fong Sufen and H. Aghajan, "MeshEye: A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance", Proceedings of IPSN-SPOTS, 2007

[Hongeng04] S. Hongeng, R. Nevatia and F. Bremond, "Video-based Event Recognition: Activity Representation and Probabilistic Recognition Methods", Computer Vision and Image Understanding Journal, Special Issue on Event Detection in Video, no.2, vol96, 2004

[Huiyu08] Z. Huiyu, M. Taj and A. Cavallaro, "Target Detection and Tracking With Heterogeneous Sensors", IEEE Journal of Selected Topics in Signal Processing, no.4, vol.2, pp.503-513, 2008

[Jain00] A. K. Jain, R. P.W. Duin and J. Mao, "Statistical Pattern Recognition: A Review", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.22, pp.4-37, 2000

[Javed02] O. Javed and M. Shah, "Tracking and Object Classification for Automated Surveillance", Proc. of European Conference on Computer Vision, 2002

[Javed03] O. Javed, Z. Rasheed, O. Alatas and M. Shah, "KNIGHTM: A Real-time Surveillance System for Multiple Overlapping and Non-overlapping Cameras", Proc. of ICME03, pp.649-652, 2003

[Javed06] O. Javed and M. Shah, "Tracking and Object Classification for Automated Surveillance", Computer Vision - ECCV, vol.2352, pp.439-443, 2002

[Kassebaum08] J. Kassebaum, N. Bulusu and W. Feng, "Smart Camera Network Localization Using a 3D Target", Proceedings of the Workshop on Applications, Systems & Algorithms for Image Sensing, 2008

[Khatib99] W. Al-Khatib, Y.F. Day, A. Ghafoor and P.B. Berra, "Semantic Modeling and Knowledge Representation in Multimedia Databases", IEEE Transactions on Knowledge and Data Engineering, no.1, vol.11, pp.64-80, 1999

[Korpeoglu06] B. Bostan-Korpeoglu, A. Yazici, I. Korpeoglu and R. George, "A New Approach for Information Processing in Wireless Sensor Network Database Application", Proceedings of NETDB, pp.1, 2006

[Landabaso04] J. L. Landabaso, L. Q. Xu and M. Pardas, "Robust Tracking and Object Classification Towards Automated Video Surveillance", Lecture Notes in Computer Science, vol.3212, pp.463-470, 2004

[Lavee09] G. Lavee, E. Rivlin and M. Rudzsky, "Understanding Video Events: A Survey of Methods for Automatic Interpretation of Semantic Occurrences in Video", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol.39, pp.489-504, 2009

[Leonardi02] R. Leonardi and P. Migliorati, "Semantic Indexing of Multimedia Documents", IEEE Multimedia, no.2, vol.9, pp-921-960, 2002

[Li97] J.Z. Li, M.T. Özsu and D. Szafron, “Modeling of Moving Objects in a Video Database”, Proceedings of IEEE Int. Conf. on Multimedia Computing and Systems, pp.336-343, 1997

[Li07] M. Li, Y. Liu and L. Chen, “Non-Threshold Based Event Detection for 3D Environment Monitoring in Sensor Networks”, ICDCS ’07: Proceedings of the 27th International Conference on Distributed Computing Systems, p.9, 2007

[Li10] M. Li, M. Mani, E. A. Rundensteiner and Tao Lin, “Constraint-Aware Complex Event Pattern Detection over Streams”, Lecture Notes in Computer Science, vol.5982, pp.199-215, 2010

[Lin02] F. Lin and H. Ying, “Modeling and Control of Fuzzy Discrete Event Systems”, IEEE Transactions On Systems, Man, And Cybernetics - Part B: Cybernetics, no.4, vol.32, 2002

[Liu09] L. Liu, X. Zhang and H. Ma, “Dynamic Node Collaboration for Mobile Target Tracking in Wireless Camera Sensor Networks”, IEEE INFOCOM 2009, pp.1188-1196, 2009

[Lowe99] D. G. Lowe, “Object Recognition from Local Scale-Invariant Features”, Proceedings of the International Conference on Computer Vision, ICCV ’99, vol.2, p.1150, 1999

[Luckham98] D. Luckham and B. Frasca, “Complex Event Processing in Distributed Systems”, Stanford University, 1998

[Luckham08] D. Luckham, “The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems”, Lecture Notes in Computer Science, vol.5321, p.3, 2008

[Lyons00] D. Lyons, T. Brodsky, E. Cohen-Solal and A. Elgammal, “Video Content Analysis for Surveillance Applications”, Philips Digital Video Technologies Workshop, 2000

[Manjunath01] B.S. Manjunath, J.R. Ohm, V.V. Vasudevan and A. Yamada, “Color and Texture Descriptors”, IEEE Transactions on Circuits and Systems for Video Technology, no.6, vol.11, pp.703-715, 2001

[Mckenna00] S.J Mckenna, S. Jabri, Z. Duric, H. Wechsler and A. Rosenfeld, “Tracking Groups of People”, Computer Vision and Image Understanding, vol.80, pp.42-56, 2000

[Medioni01] G. Medioni and F. Bremond and R. Nevatia, “Event Detection and Analysis from Video Streams”, IEEE Transactions on Pattern Analysis and Machine Intelligence, no.8, vol.23, pp.873-889, 2001

[Michie94] D. Michie, D.J. Spiegelhalter and C.C. Taylor, “Machine Learning , Neural and Statistical Classification”, Technometrics Journal, no.4, vol.37, p.459, 1994

[Noble09] B. D. Noble and J. Flinn, “Wireless, Self-Organizing Cyber-Physical Systems”, NSF Workshop on Cyber-physical Systems, 2009

[Orten05] B. Orten, “Moving Object Identification and Event Recognition in Video Surveillance Systems”, Master’s Thesis, Electrical and Electronics Department, Middle East Technical University, Turkey, 2005

[Oztarak05] H. Öztarak, “Structural and Event Based Multimodal Video Data Modeling”, Master’s Thesis, Computer Engineering Department, Middle East Technical University, Turkey, 2005

[Oztarak06] H. Öztarak and A. Yazıcı, “Structural And Event Based Multimodal Video Data Modeling”, Proceedings of the Advances in Information Systems (ADVIS06), Lecture Notes in Computer Science (LNCS) 4243, pp.264-273, 2006

[Oztarak06-2] H. Öztarak and A. Yazıcı, “Flexible Querying Using Structural and Event Based Multimodal Video Data Model”, Proceedings of the 7th International Conference on Flexible Query Answering Systems (FQAS06), Lecture Notes in Computer Science (LNCS) 4027, pp.75-86, 2006

[Oztarak07] H. Öztarak, A. Yazıcı, D. Aksoy and R. George, “Multimedia Processing in Wireless Sensor Networks”, Proceedings of 4th International IEEE Conference on Innovations in Information Technology (Innovations’07), pp.78-82, 2007

[Oztarak09] H. Öztarak, K. Akkaya and A. Yazıcı, “Lightweight Object Localization with a Single Camera in Wireless Multimedia Sensor Networks”, Proceedings of the IEEE Global Communications Conference (IEEE GLOBECOM 2009), pp.12-13, 2009

[Oztarak11] H. Öztarak, K. Akkaya and A. Yazıcı, “Providing Automated Actions in Wireless Multimedia Sensor Networks via Active Rules”, The 26th International Symposium on Computer and Information Sciences (ISCIS 2011), Lecture Notes in Electrical Engineering (LNEE), pp.185-190, 2011

[Oztarak12] H. Öztarak, T. Yılmaz, K. Akkaya and A. Yazıcı, “Efficient and Accurate Object Classification in Wireless Multimedia Sensor Networks”, accepted from IEEE International Conference on Computer Communication Networks (ICCCN), 2012

[Oztarak12-2] H. Öztarak, K. Akkaya and A. Yazıcı, “Efficient Tracking of Multiple Objects in Wireless Multimedia Sensor Networks”, accepted from Ad Hoc & Sensor Wireless Networks Journal, 2012

[Park06] U. Park, A. K. Jain, I. Kitahara, K. Kogure and N. Hagita, “ViSE:Visual Search Engine Using Multiple Networked Cameras”, ICPR 06: Proceedings of the 18th International Conference on Pattern Recognition, vol.3, pp.1204-1207, 2006

[Pasula99] H. Pasula, S. J. Russell, M. Ostland and Y. Ritov, “Tracking Many Objects with many Sensors”, In Proceedings of IJCAI, vol.2, pp.1160-1171, 1999

[Patwari05] N. Patwari, J.N. Ash, S. Kyperountas, A.O Hero Lii, R.L. Moses and N.S. Correal, “Locating the Nodes: Cooperative Localization in Wireless Sensor Networks”, Signal Processing Magazine, IEEE, no.4, vol.22, pp.54-69, 2005

[Pinto10] A. Pintoa, Z.Zhang, X.Dong, S. Velipasalar, M.C.Vuran and C.M.Gursoy, “Energy Consumption and Latency Analysis for Wireless Multiemdia Sensor Networks”, IEEE Globecom Conference, pp.1-5, 2010

[Qi07] Q. Mei-bin, Z. Rui, J. Jian-guo and L. Xiang-tao, “Moving Object Localization with Single Camera Based on Height Model in Video Surveillance”, Proceedings of International Conference on Bioinformatics and Biomedical Engineering, (ICBBE’07), 2007

[Rahimi05] M. Rahimi, R. Baer, O. Iroezi, J. Garcia, J. Warrior and M. Srivastava, “Cyclops: in situ Image Sensing and Interpretation in Wireless Sensor Networks”, Proc. 3rd ACM Conf. Embedded Networked Sensor Systems (SenSys 2005), pp.193-204, 2005

[Rangaswami04] R. Rangaswami, Z. Dimitrijevic, K. Kakligian, E. Chang and Y.F. Wang, “The SfinX Video Surveillance System”, ICME, 2004

[Ravi05] N. Ravi, P. Shankar, A. Frankel, A. Elgammal and L. Iftode, “Indoor Localization Using Camera Phones”, Proceedings of 7th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA’06), 2005

[Rivlin02] E. Rivlin, M. Rudzsky, U. Bogomolov, S. Lapchev and R. Goldenberg, “A Real-Time System for Classification of Moving Objects”, ICPR’02, pp.688-691, 2002

[Ross95] T.J. Ross, “Fuzzy Logic with Engineering Applications”, McGraw-Hill Inc.ISBN 0-0711-3637-1, 1995

[Rowe07] A. Rowe, A. Goode, D. Goel and I. Nourbakhsh, “CMUcam3: an Open Programmable Embedded Vision Sensor”, Technical Report, RI-TR-07-13, Carnegie Mellon Robotics Institute, 2007

[Sharif09] A. Sharif, V. Potdar and E. Chang, “Wireless Multimedia Sensor Network Technology: A Survey”, Proceedings of 7th IEEE International Conference on Industrial Informatics, pp.606-613, 2009

[Shtovba00] S. Shtovba and G. Chernovolik, “Fuzzy Rule Based System for Decision Making Support of Pathology Anatomist”, Proc. of the ERUDIT- Fuzzy Diagnostic and Therapeutic Decision Support Workshop, pp.45-50, 2000

[Smeulders00] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta and R. Jain, “Content-based Image Retrieval at the end of the Early Years”, IEEE Trans. Pattern Anal. Mach. Intell., no.12, vol.22, pp.1349-1380, 2000

[Snoek05] C.G.M. Snoek, “Multimedia Event-based Video Indexing using Time Intervals”, IEEE Transactions on Multimedia, pp.638-647, 2005

[Solar10] G. Vargas-Solar, P. Bucciol and C. Collet, “On a Framework for Complex and Ad Hoc Event Management over Distributed Systems”, Polibits, no.41, pp.47-58, 2010

[Soro09] S. Soro and W. Heinzelman, “A Survey of Visual Sensor Networks”, Advances in Multimedia Hindawi Journal, 2009

[Spors01] S. Spors, R. Rabenstein and N. Strobel, “A Multi-Sensor Object Localization System”, Proceedings of the Vision Modeling and Visualization Conference, 2001

[Wang08] W. Wang, “Complex Event Processing in EPC Sensor Network Middleware for Both RFID and WSN”, 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pp.165-169, 2008

[Wang10] X. Wang, S. Wang, J. Ma and X. Sun, “Energy-aware Scheduling of Surveillance in Wireless Multimedia Sensor Networks”, Sensors Journal, no.4, vol.10, pp.3100-3125, 2010

[WEB:AVRORA] “AVRORA Website”, URL: http://compilers.cs.ucla.edu/avrora/, Last Accessed: April 2012

[WEB:BATTERY] "AllAboutBatteries.com", http://www.allaboutbatteries.com/Energy-tables.html, Last Accessed: April 2012

[WEB:BorderReport05] "Failure of the Administration to Deliver a Comprehensive Land Border Strategy Leaves Our Nation's Borders Vulnerable", URL: http://www.globalsecurity.org/security/library/report/2005/050524-border-final-version.pdf, Last Accessed: April 2012

[WEB:BWN] "BWN Research Lab Website", URL: http://www.ece.gatech.edu/research/labs/bwn/WMSN/testbed.html, Last Accessed: April 2012

[WEB:CmuCam] "CMUCAM3 Wiki", URL: http://www.cmucam.org/projects/cmucam3/wiki, Last Accessed: April 2012

[WEB:CYC] "Cyclops Website", URL: http://www.engineer.ucla.edu/news/2006/cyclops.html, Last Accessed: June 2007

[WEB:Jess] , "JESS Website", URL: http://herzberg.ca.sandia.gov/jess, Last Accessed: April 2012

[WEB:PowerPlant] "The Nesjavellir Geothermal Power Plant", URL: http://commons.wikimedia.org/wiki/File:NesjavellirPowerPlant_edit2.jpg, Last Accessed: April 2012

[Yilmaz11] T. Yilmaz, Y. Yildirim and A. Yazıcı, "A Genetic Algorithms Based Classifier for Object Classification in Images", ISCIS 2011, pp.519-525, 2011

[Yilmaz11-2] T. Yilmaz, Y. Yildirim and A. Yazıcı, "Exploiting Class-Specific Features in Multi-feature Dissimilarity Space for Efficient Querying of Images", FQAS 2011, pp.149-161, 2011

[Zaniolo93] C. Zaniolo, "On the Unification of Active and Deductive Databases", Advances in 11th British National Conference on Database, pp.23-29, 1993

[Zhang04] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks", Proceedings of the IEEE Transactions on Wireless Communications, no.5, vol.3, 1689-1701, 2004

[Zoumboulakis04] M. Zoumboulakis, G. Roussos and A. Poulovassilis, "Active Rules for Sensor Databases", DMSN 2004 International Workshop on Data Management for Sensor Networks, 2004

APPENDIX A

IMPLEMENTATION

In this section, our simulator and its features are explained in detail. The simulator has the capability of simulating any number of camera sensors distribution on an area of defined size. In Figure A.1, the simulator main screen is shown when there exists only the sink at the position of $(0,0)$. Transmission range for the sink is shown as a circle around the sink. The sensors who will directly communicate with the sink should be positioned in that area. Otherwise, they are out of range for the sink.

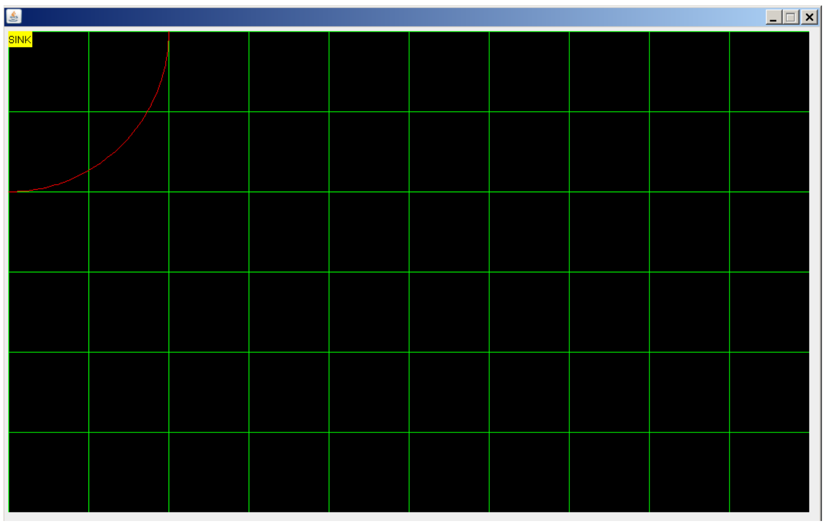


Figure A.1: The area under surveillance and sink position

The first camera sensor which is positioned on the area under surveillance is shown in Figure A.2. Similar to the sink, the camera sensor has also its own transmission range, drawn as a circle around it. Every camera sensor has its own id also, i.e. 1 in this case.

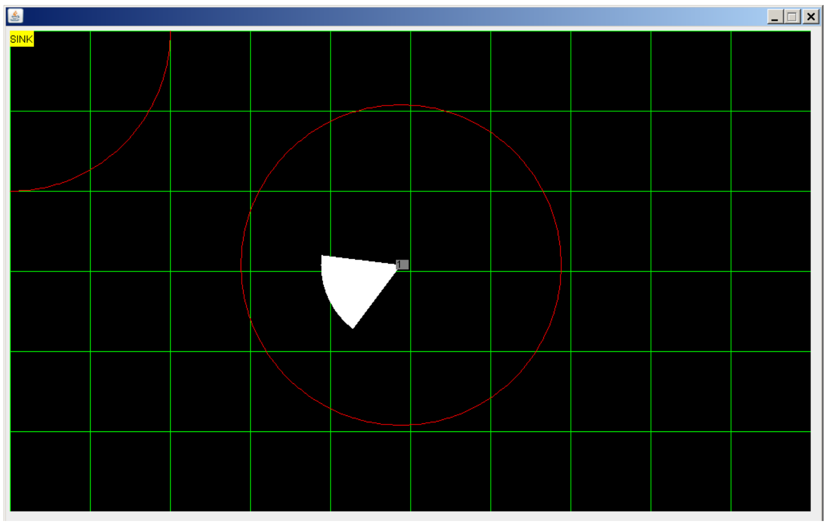


Figure A.2: A camera sensor on the area

Camera sensors can communicate with one another as long as they are within the transmission range of each other. In Figure A.3, 8 camera sensors are positioned on the area under surveillance, where each of the camera sensors has at least one neighbor in its transmission range. The simulator automatically calculates the neighboring graph and draws it with the lines between camera sensors or between camera sensors and the sink. This graph shows which sensor can communicate with which sensor. In this example, the graph is connected where each of the sensors has a path to the sink for communication.

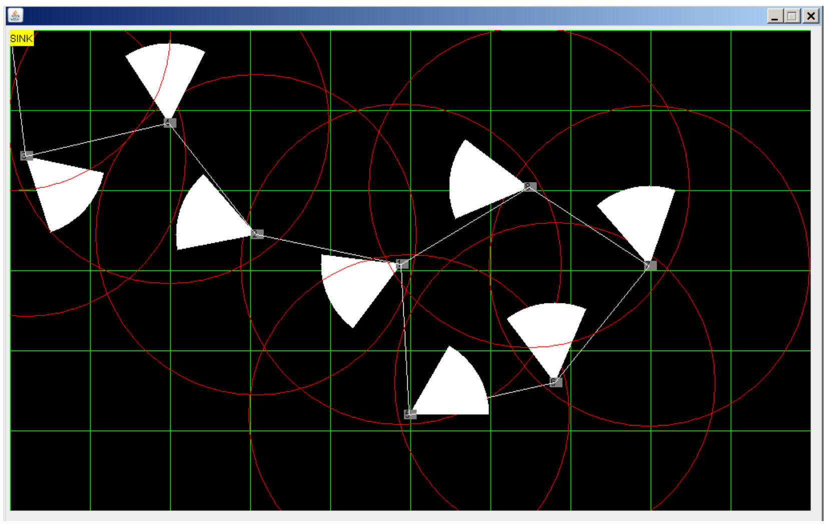


Figure A.3: Camera sensors and their neighboring graph

Camera sensors discover their neighbors automatically and use Dijkstra's shortest path algorithm to send their messages to the sink. Based on neighboring graph calculated in Figure A.3, the simulator calculates the shortest path of each sensor to communicate with the sink and draws it on the main screen as shown in Figure A.4.

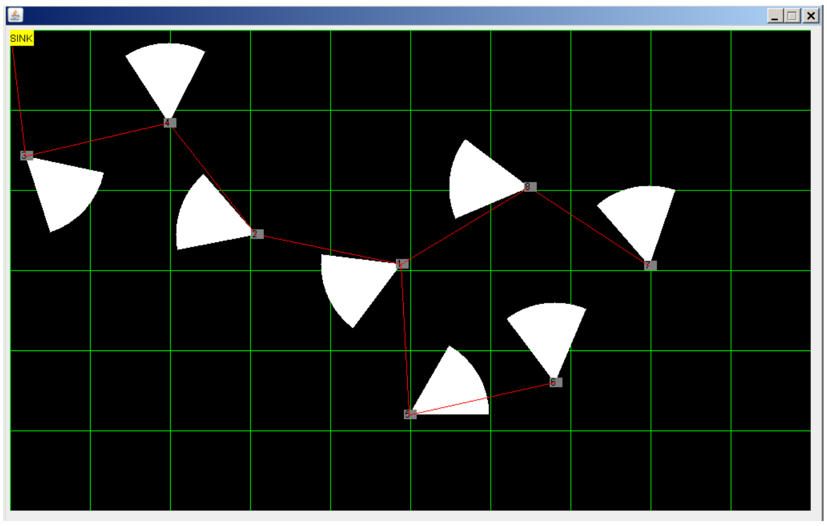
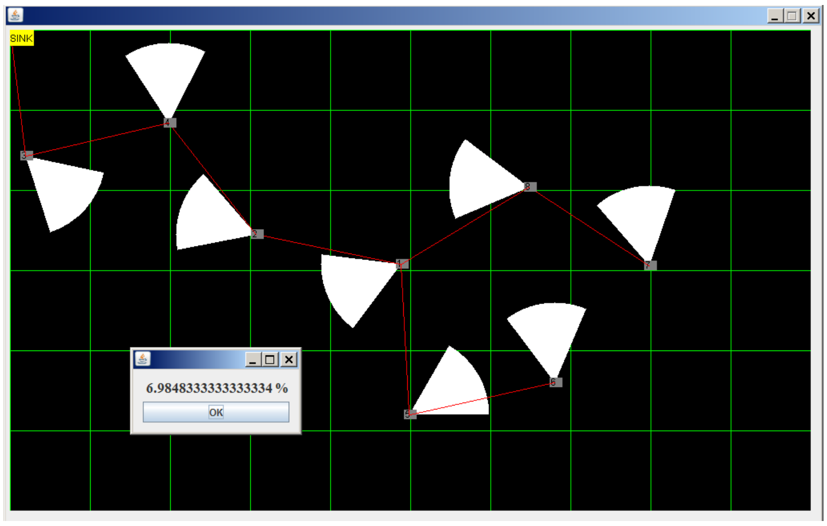


Figure A.4: Shortest path for each camera sensors to the sink

The simulator can also calculate the camera sensor's total coverage value for the area under surveillance. In Figure A.5, the coverage value for the distribution is calculated as 7% approximately.



6.9848333333333334 %

Figure A.5: Calculating camera sensors' coverage value

Since there may exists different kinds of sensors with different Depth of Views and Angle of Visions, the simulator has also the capability of simulating them. In Figure A.6, the properties of the camera sensors given in Figure A.5 are changed to increase the camera coverage value to 30% approximately.

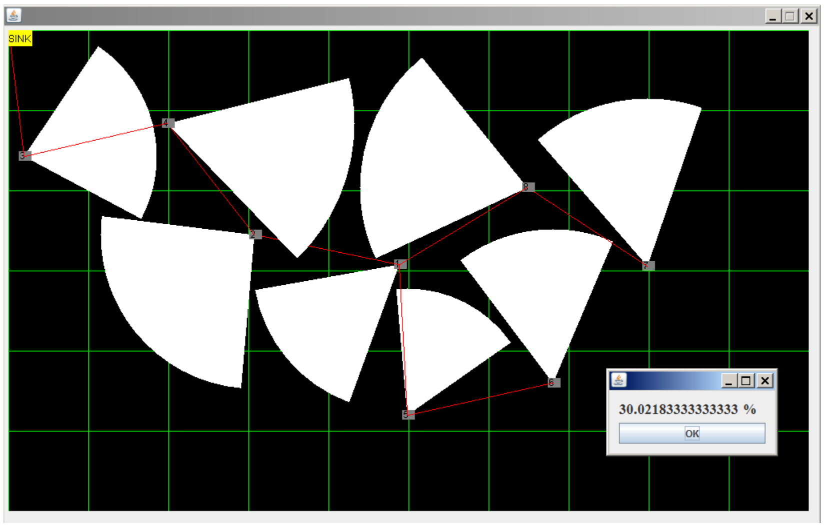


Figure A.6: Increased camera coverage value by changing the camera features

In Figure A.7, the camera sensors are distributed to the area to achieve 25% camera coverage value approximately.

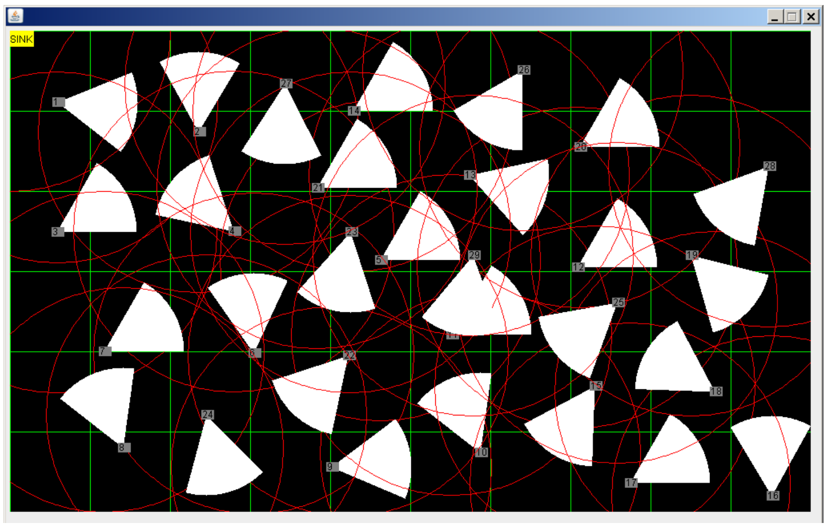


Figure A.7: Camera sensor distribution to achieve 25% camera coverage value

The experiments explained in Chapter 5 are performed under different camera coverage values. In Figure A.8, 75% camera coverage is achieved by 102 different camera sensors.

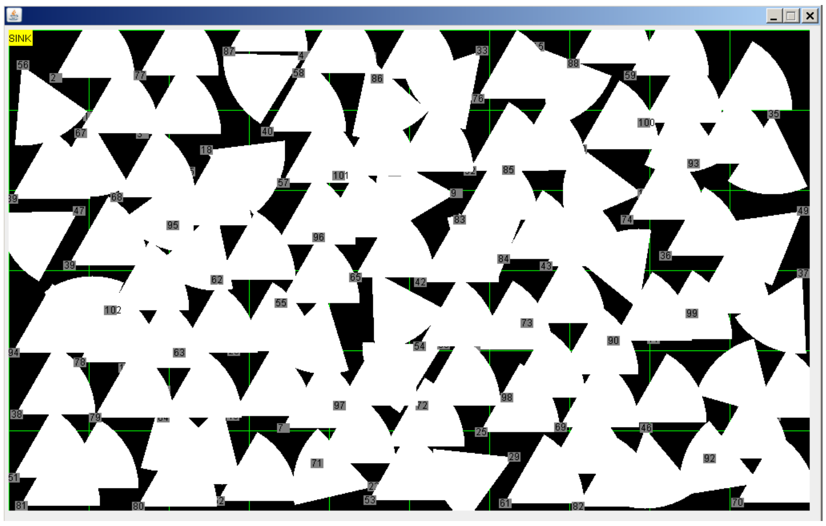


Figure A.8: 102 camera sensor to achieve 75% camera coverage value

The intruder objects (human, vehicle or animals) are injected to the area under surveillance with their appropriate membership values. In Figure A.9, a human intruder is injected to the system. The camera sensors that detect that human intruder and sends MOE information to the sink are the camera sensors 98 and 25.

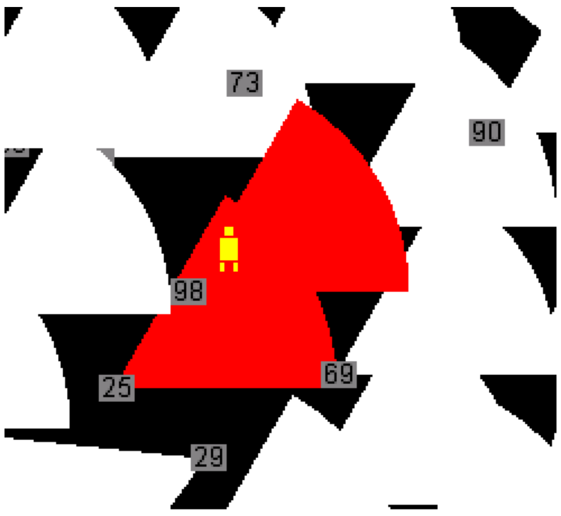


Figure A.9: A human intruder detected by two camera sensors

The MOEs sent by the camera sensors are processed at the sink to track the intruders. In Figure A.10, the path of the human intruder is drawn by the sensor according to the route information stored at the sink. This route information is calculated by the Object Tracking Algorithm at the Sink, i.e. Algorithm 2 given in Chapter 6.4.

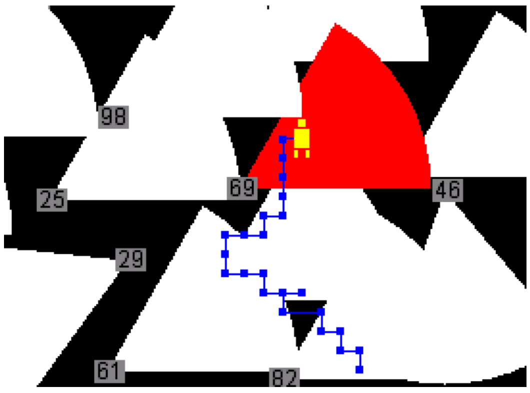


Figure A.10: Object tracking result

In Figure A.11, 12 different objects which 4 of them human, 4 of them vehicle and 4 of them animal are injected to the system. Each of the intruder objects has different random paths having the purpose of attacking to the sink. 13 camera sensors that detect those objects and send MOE information to the sink are shown in different color.

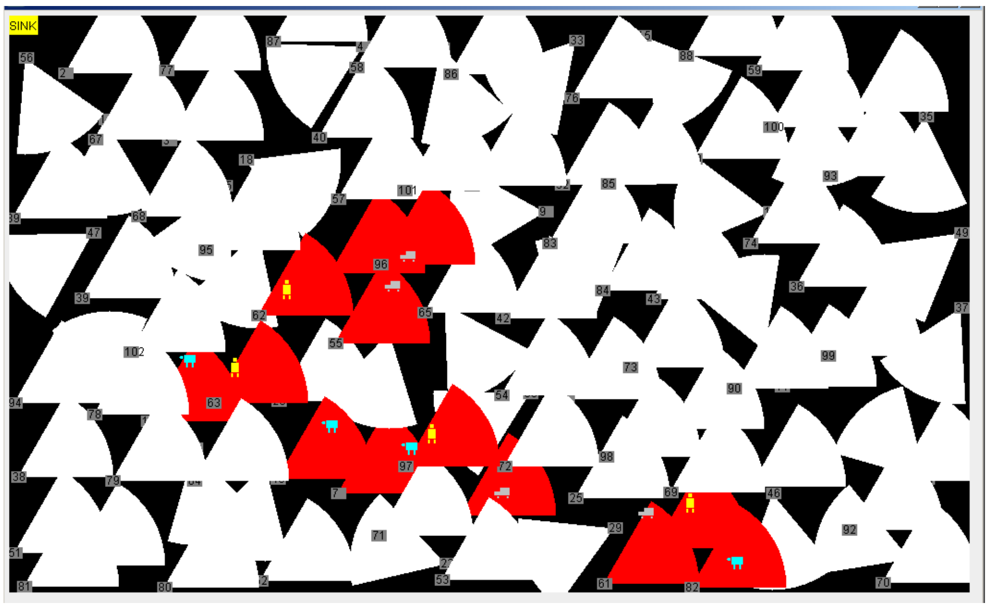
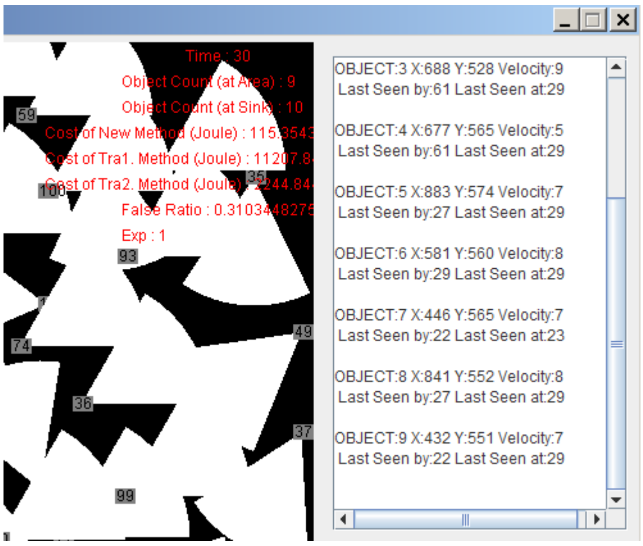


Figure A.11: Multiple random objects injected to the area under surveillance

The simulator has the information area in which the real-time information about the performing experiments is given. In Figure A.12, this area is shown. On the left side; time, energy measurements and object counts are given, on the right side; the information stored at the sink is given. Hence, the user is able to follow the experiments at real-time.



```
Time : 30
Object Count (at Area) : 9
Object Count (at Sink) : 10
Cost of New Method (Joule) : 115.3543
Cost of Tra1. Method (Joule) : 11207.8
Cost of Tra2. Method (Joule) : 2244.84
False Ratio : 0.3103448275
Exp : 1
```

```
OBJECT:3 X:688 Y:528 Velocity:9
 Last Seen by:61 Last Seen at:29

OBJECT:4 X:677 Y:565 Velocity:5
 Last Seen by:61 Last Seen at:29

OBJECT:5 X:883 Y:574 Velocity:7
 Last Seen by:27 Last Seen at:29

OBJECT:6 X:581 Y:560 Velocity:8
 Last Seen by:29 Last Seen at:29

OBJECT:7 X:446 Y:565 Velocity:7
 Last Seen by:22 Last Seen at:23

OBJECT:8 X:841 Y:552 Velocity:8
 Last Seen by:27 Last Seen at:29

OBJECT:9 X:432 Y:551 Velocity:7
 Last Seen by:22 Last Seen at:29
```

Figure A.12: Information area of the simulator

The simulator has the ability to detect moving objects in videos. In Figure A.14 moving object detection result of the simulator is shown for the sample frame given in Figure A.13.

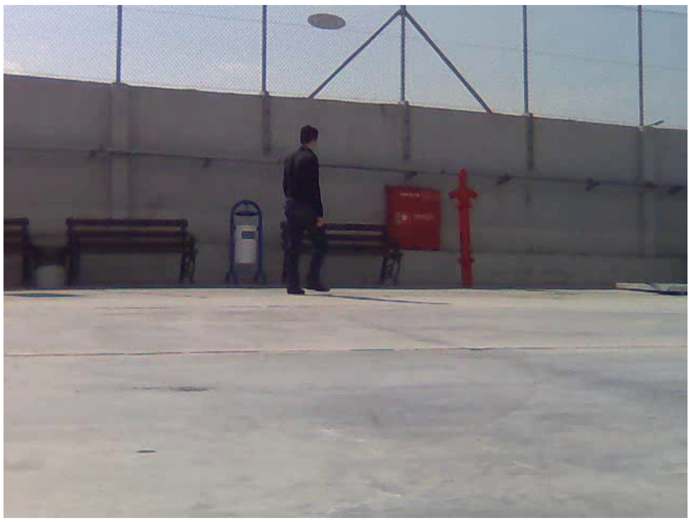


Figure A.13: Sample frame for moving object detection

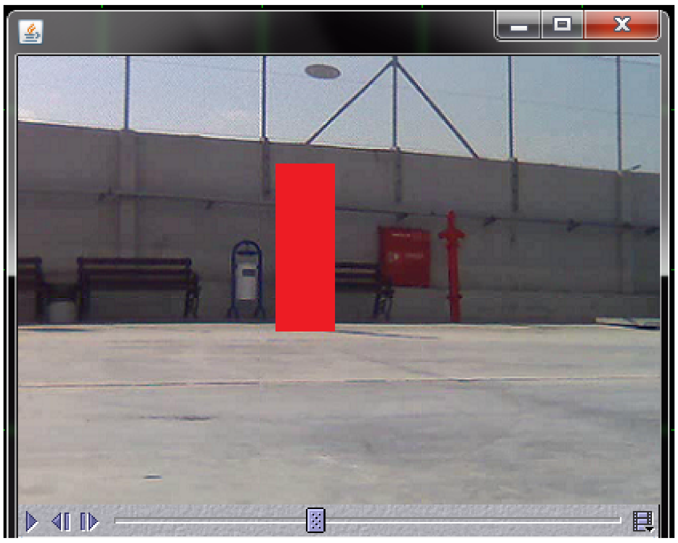


Figure A.14: Detected moving object is marked as red inside its MBR by the simulator

VITA

Hakan Öztarak was born in Denizli, Turkey in 1981. He received his B.Sc. degree from Computer Engineering Department of Middle East Technical University in 2003. He received his M.Sc. degree from the same department in 2005. Since 2003, he has been working at ASELSAN Inc., where he has participated in research and development (R&D) activities of Weapon Systems, Radar Systems and Command Control (C4ISR) Systems. His research interests include video databases, multimedia modeling, wireless multimedia (visual) sensor networks and active systems.

Publications

1. Hakan Öztarak, Kemal Akkaya and Adnan Yazıcı, "Efficient Tracking of Multiple Objects in Wireless Multimedia Sensor Networks", accepted from Ad Hoc & Sensor Wireless Networks Journal, 2012
2. Hakan Öztarak, Turgay Yılmaz, Kemal Akkaya and Adnan Yazıcı, "Efficient and Accurate Object Classification in Wireless Multimedia Sensor Networks", accepted from IEEE International Conference on Computer Communication Networks (ICCCN), 2012
3. Hakan Öztarak, Kemal Akkaya and Adnan Yazıcı, "Providing Automated Actions in Wireless Multimedia Sensor Networks via Active Rules", The 26th International Symposium on Computer and Information Sciences (ISCIS 2011), Lecture Notes in Electrical Engineering (LNEE), pp.185-190, 2011
4. Hakan Öztarak, Kemal Akkaya and Adnan Yazıcı, "Lightweight Object Localization with a Single Camera in Wireless Multimedia Sensor Networks", Proceedings of the IEEE Global Communications Conference (IEEE GLOBECOM 2009), pp.12-13, 2009
5. Hakan Öztarak, Adnan Yazıcı, Demet Aksoy and Roy George, "Multimedia Processing in Wireless Sensor Networks", Proceedings of 4th International IEEE Conference on Innovations in Information Technology (Innovations'07), pp.78-82, 2007

6. Hakan Öztarak and Adnan Yazıcı, “Flexible Querying Using Structural and Event Based Multimodal Video Data Model”, Proceedings of the 7th International Conference on Flexible Query Answering Systems (FQAS06), Lecture Notes in Computer Science (LNCS) 4027, pp.75-86, 2006
7. Hakan Öztarak and Adnan Yazıcı, “Structural And Event Based Multimodal Video Data Modeling”, Proceedings of the Advances in Information Systems (ADVIS06), Lecture Notes in Computer Science (LNCS) 4243, pp.264-273, 2006

Thesis

1. Hakan Öztarak, “Structural and Event Based Multimodal Video Data Modeling”, Master’s Thesis, Computer Engineering Department, Middle East Technical University, Turkey, 2005