FUZZY DECISION FUSION FOR SINGLE TARGET CLASSIFICATION IN WIRELESS SENSOR NETWORKS

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

BY

SERCAN GÖK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2009

Approval of the thesis:

FUZZY DECISION FUSION FOR SINGLE TARGET CLASSIFICATION IN WIRELESS SENSOR NETWORKS

submitted by **SERCAN GÖK** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

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

Prof. Dr. Müslim Bozyiğit
Head of Department, **Computer Engineering** _____

Prof. Dr. Adnan Yazıcı
Supervisor, **Computer Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Müslim Bozyiğit
Computer Engineering Dept., METU _____

Prof. Dr. Adnan Yazıcı
Computer Engineering Dept., METU _____

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

Assoc. Prof. Dr. Halit Oğuztüzün
Computer Engineering Dept., METU _____

Asst. Prof. Dr. İbrahim Körpeoğlu
Computer Engineering Dept., Bilkent University _____

Date: **11.12.2009**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: SERCAN GÖK

Signature :

ABSTRACT

FUZZY DECISION FUSION FOR SINGLE TARGET CLASSIFICATION IN WIRELESS SENSOR NETWORKS

Gök, Sercan
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. Adnan Yazıcı

December 2009, 60 pages

Nowadays, low-cost and tiny sensors are started to be commonly used due to developing technology. Wireless sensor networks become the solution for a variety of applications such as military applications. For military applications, classification of a target in a battlefield plays an important role. Target classification can be done effectively by using wireless sensor networks. A wireless sensor node has the ability to sense the raw signal data in battlefield, extract the feature vectors from sensed signal and produce a local classification result using a classifier. Although only one sensor is enough to produce a classification result, decision fusion of the local classification results for the sensor nodes improves classification accuracy and loads lower computational burden on the sensor nodes. Decision fusion performance can also be improved by picking optimum sensor nodes for target classification.

In this thesis, we propose fuzzy decision fusion methods for single target classification in wireless sensor networks. Our proposed fusion algorithms use fuzzy logic for selecting the appropriate sensor nodes to be used for classification. Our solutions provide better classification accuracy over some popular decision fusion algorithms. In addition to fusion algorithms, we present some techniques for feature vector size reduction on sensor nodes, and training set

formation for classifiers.

Keywords: Wireless Sensor Networks, Classification, Fusion, Fuzzy Logic

ÖZ

KABLOSUZ ALGILAYICI AĞLARDA TEK HEDEF SINIFLANDIRMASI İÇİN BULANIK KARAR BİRLEŞTİRMESİ

Gök, Sercan
Yüksek Lisans, Bilgisayar Mühendisliği
Tez Yöneticisi : Prof. Dr. Adnan Yazıcı

Aralık 2009, 60 sayfa

Günümüzde gelişen teknoloji ile düşük maliyetli ve küçük algılayıcılar yaygın olarak kullanılmaya başlandı. Kablosuz algılayıcı ağları askeri uygulamalar gibi birçok uygulamaya çözüm oldu. Savaş alanında bir hedefin sınıflandırılması, askeri uygulamalar için önemli bir rol oynamaktadır. Kablosuz algılayıcı ağları kullanarak etkili bir biçimde hedef sınıflandırması yapılabilir. Kablosuz bir algılayıcı, savaş alanındaki ham sinyal verisini algılama, algılanan sinyalden öznitelik vektörleri çıkarma, ve sınıflandırıcı kullanarak yerel sınıflandırma sonucu üretebilme yeteneklerine sahiptir. Sadece bir algılayıcının sınıflandırma sonucu üretebilmek için yeterli olmasına rağmen, algılayıcıların yerel sınıflandırma sonuçlarının karar birleştirmesi, sınıflandırma doğruluğunu geliştirir ve algılayıcılar üzerine daha az hesaplama külfeti yükler. Karar birleştirme performansı da hedef sınıflandırması için en ideal algılayıcıları seçerek geliştirilebilir.

Bu tez çalışmasında, kablosuz algılayıcı ağlarda tek hedef sınıflandırması için bulanık karar birleştirme yöntemleri önerilmektedir. Önerilen birleştirme algoritmaları sınıflandırma için kullanılacak uygun algılayıcıları seçmek için bulanık mantığı kullanmaktadır. Çözümümüz, bazı popüler karar birleştirme algoritmalarından daha iyi bir sınıflandırma doğruluğu sunmak-

tadır. Birleştirme algoritmalarına ek olarak, algılayıcılarda öznitelik vektörü boyut azaltma ve sınıflandırıcılar için öğrenim kümesi oluşturma için bazı teknikler sunulmaktadır.

Anahtar Kelimeler: Kablosuz Algılayıcı Ağlar, Sınıflandırma, Birleştirme, Bulanık Mantık

To my family...

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Adnan Yazıcı for his endless encouragement and support through this thesis. Without his expertise and patience, I would not complete this study.

I thank to jury members for reviewing and evaluating my thesis.

I also thank to TÜBİTAK UEKAE / İLTAREN and my colleagues who support me all the time.

I owe my deepest gratitude to my family for their endless love and morale support throughout my life.

I am also profoundly thankful to my friend Adnan Orduyılmaz for his continuous guidance and encouragement. His presence become an invaluable asset during this study.

Lastly, I want to express my thanks to my friends Kerim Nazlı and Mehmet Olduz for the help, support and joy they provide during this work.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVATIONS

WSN Wireless Sensor Network

FT Fourier Transform

DFT Discrete Fourier Transform

FFT Fast Fourier Transform

PSD Power Spectrum Density

DC Direct Current

PCA Principal Component Analysis

kNN k-Nearest Neighbor

ML Maximum Likelihood

SVM Support Vector Machine

SNR Signal to Noise Ratio

DSN Distributed Sensor Network

WDSN Wireless Distributed Sensor Network

AAV Assault Amphibian Vehicle

M1 Main Battle Tank

HMMWV High Mobility Multipurpose Wheeled Vehicle

DW Dragon Wagon

MAP Maximum A Posterior

BKS Behavior-Knowledge Space

MRI Multi-Resolution Integration

MADSN Mobile-agent-based DSN

SSWCC Spectral Statistics and Wavelet Coefficients Characterization

MVDF Majority Voting Decision Fusion

NNDF Nearest Neighbor Decision Fusion

DMDF d_{max} Decision Fusion

FDF Fuzzy Decision Fusion

FDFWT Fuzzy Decision Fusion with Threshold

UTM Universal Transverse Mercator

CFAR Constant False Alarm Rate

CHAPTER 1

INTRODUCTION

Due to recent improvements in hardware technology, the extensive use of wireless sensors has expanded into many application areas. Being small, low-cost and low-power devices, wireless sensors are considered to be an efficient option to deploy on real time environments. Sensor nodes can communicate with each other through wireless channels to share data. Data is gathered on a sink node (base station), and end-users examine the data subsequently. Such co-operation and co-ordination of sensor nodes construct a kind of network, which is called wireless sensor network (WSN). In WSN applications, sensor nodes are usually not deployed to pre-determined locations. Instead of that, random deployment of sensors is preferred.

Each sensor in WSN has to deal with its local computing process, signal processing and wireless communication; however, they have some challenges to face. Some of these challenges are listed [4] below:

- Sensor nodes are deployed in an ad hoc manner. That is, sensor nodes are spread out to the environment randomly, with no foreknown locations. Tough geographical conditions can complicate the survival of the sensor nodes.
- In most WSN, sensor nodes are once deployed and then they survive on their own. Human interaction is at minimum level. It is almost impossible to repair a sensor node once it is damaged in the event of deployment or as a result of unexpected environmental conditions.
- Sensor nodes have limited energy and limited computing power. Efficient algorithms and communication methods may minimize the complexity of computing and energy consumption.

- Sensor nodes have to be aware of the change in the environment. New sensor nodes added to the WSN, failing sensor nodes and other environmental changes can affect the traffic in WSN.

Besides these challenges, wireless sensor networks have some requirements to work more stable and robust. Considering these requirements in the design of network enables likelihood of the developing successful operations and applications. Main requirements of the wireless sensor networks can be listed as [30]:

- To make a full advantage of wireless sensor networks, sensor nodes should be deployed in large numbers. These large numbered sensor nodes should be controlled efficiently by clustering.
- Since sensor nodes have limited energy, minimizing the energy should be the major issue.
- Sensor nodes have small memories. Efficient use of memory is required.
- Each sensor produces data for its sensor network. Data aggregations should be done among individual sensor nodes in a cluster and among clusters in a network.
- Random deployment of the nodes makes self-organization ability of the wireless sensor nodes a requirement. Network should reconfigure itself to perform properly and to adapt to the changing conditions.
- Efficient signal processing inside of a single sensor node as well as data and decision fusion among all sensor nodes should be designed carefully for better performance.
- Wireless sensor nodes should have the ability to execute and return query operations on the data they sense.

Wireless sensor nodes have the ability to monitor several ambient conditions [12]. These ambient conditions can be counted as; humidity, temperature, vehicular movement, lightning condition, pressure, soil makeup, noise levels, and so on. Every different ambient condition has different characteristics and it is handled by a different kind of modality on each sensor node. The examples of modalities are seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic, and radar modality [18]. While some sensor nodes are designed to sense

using only one modality, some sensor nodes can use several modalities for sensing. Every single modality can be related to different application areas of WSN. However, some applications may require multi-modality sensor nodes to perform better.

Wireless sensors are extensively used in many various applications today. Most common application areas can be categorized as follows [2]:

- Wireless sensor networks can be used in military applications. Using the rapid deployed, self-organized and fault-tolerant properties of sensor nodes, battlefields are appropriate places for sensor nodes. Monitoring friendly forces, equipment and ammunition, reconnaissance of opposing forces and terrain, targeting, and battle damage assessment are some applications of wireless sensor networks for military.

- Environmental applications are also using wireless sensor networks. The environmental application examples can be counted as; tracking the movement of small animals, birds or insects, monitoring the conditions about agriculture, detecting forest fire, air or water pollution, flood and soil erosion.

- Some useful wireless sensor network applications have been employed in health area. Patients who need to be monitored constantly can take the advantage of sensor nodes. Tele-monitoring of human physiological data, tracking and monitoring doctors and patients inside a hospital, drug administration in hospitals are some common health applications of wireless sensor networks.

- Many home applications make use of wireless sensor networks. Some electronic devices such as micro-wave ovens or refrigerators can interact with each other to perform more economically, more efficiently and more safely. Moreover, these devices can be controlled remotely by wireless sensors.

- Other than above, wireless sensor networks application may include: Environmental control in office buildings, interactive museums, detecting and monitoring car thefts, managing inventory control and vehicle tracking and detection.

The scenarios, in which wireless sensor networks are applicable, are countless. More useful and practical applications can be developed in near future.

In military applications, monitoring battlefield (sensor field) is an important concept. Detection of a target, classification of the target and tracking of the target are the essential steps and extensive researches have been made on these three operations. When a target intrudes a battlefield, sensor nodes detect the target. After ensuring that there is a target in the battlefield, sensor nodes try to classify the target. Tracking of the target follows the classification process. By accomplishing these three steps, necessary information about target is retrieved and defense mechanism becomes ready to act.

Target classification in battlefields is the main issue discussed in this thesis. A target intruding the battlefield should be classified accurately so that the related precautions can be taken. Target classification is a complicated process, in which many design issues take place and play important role on classification accuracy. Collaborative signal processing algorithms, feature extractions, classifier design and choice, deployment and locations of the sensor nodes, the modality types of the sensor nodes, data and decision fusion are all parts of the classification process and many studies have been done on each of these topics.

Since data coming from just one sensor node is not reliable inside a wireless sensor network, some fusion mechanisms have been developed. By fusing among sensor nodes, not only the performance of the wireless sensor network increases, but also energy consumption can be lowered by only keeping the related sensor nodes busy. In [32], some advantages of using multiple sensors deployment over one sensor deployment are mentioned:

- Fusion is beneficial because when the entire sensor nodes sense the same features, they can provide redundant information. Fusion removes overall uncertainty and therefore increases accuracy.
- Having each sensor responsible for a subset of the network, more complementary information is gathered. In this way, the big picture of the wireless sensor network can be seen.
- If the fusion among sensor nodes is done in parallel, higher processing speed may be achieved.

Today many applications include some level of uncertainty. Criteria about the applications may not always be defined precisely. Fuzzy logic is a commonly used representation technique dealing with uncertainty, vagueness or impreciseness. In [34], fuzzy set is defined as

a class with a continuum of grades of membership. The difference of fuzzy set from crisp set is that fuzzy set members can have wider membership values than crisp set members. A member in a crisp set is either is a member of the set or not. However, members of the fuzzy set have some membership values showing the grade of membership. Fuzzy logic is applied to both inputs and outputs of the system and this enables the performance to be improved significantly.

In this thesis, we mainly focus on the decision fusion of the target classification. Acoustic feature vectors which are sensed by the wireless sensor nodes are used in local classification of the sensor nodes. Acoustic feature vectors can have high dimensions and reduction is needed to decrease the complexity on a sensor node. After a sensor node makes its local classification, it produces a result defining the target detected. For local classification, a classifier and a training set formed by previously obtained data are needed. Decision fusion is later done among the sensor nodes' results to produce the final result. In our study, we develop fuzzy fusion techniques for making efficient and accurate decision fusion among sensor nodes. Although various fuzzy approaches are studied in WSN, there exists no known fuzzy decision fusion method applied to target classification problem in WSN. We evaluate the performance of newly developed fuzzy decision fusion techniques over some popular decision fusion methods. Moreover, some optimizations in the feature vector size reduction and generation of training sets are done.

The remainder of the thesis is arranged as follows. In the next chapter, background information and related work about target classification problem and fuzzy approaches for wireless sensor networks are given. Later in Chapter 3, we mention about decision fusion for classification process in WSN including our fuzzy decision fusion techniques with running examples. Besides decision fusion techniques, optimizations in feature vector size reduction and training set generation take place again in Chapter 3. Next, the evaluation among decision fusion methods is made using real data in Chapter 4. Finally in Chapter 5, we conclude the thesis and discuss about some possible future works.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Background

In this section we give some background information about target classification process in wireless sensor networks and basics of fuzzy set theory.

In Figure 2.1 sample classification fusion process for WSN is shown [6]. The sensor nodes sense the signal and extract the relevant feature vectors. Then, classifiers on sensor nodes make local classification based on the feature vectors and training sets. Decision fusion is then done by a fusing algorithm using the decisions generated by all the sensor nodes where a final result is formed. Fusion center node can be any node randomly chosen to aggregate the data from other sensor nodes.

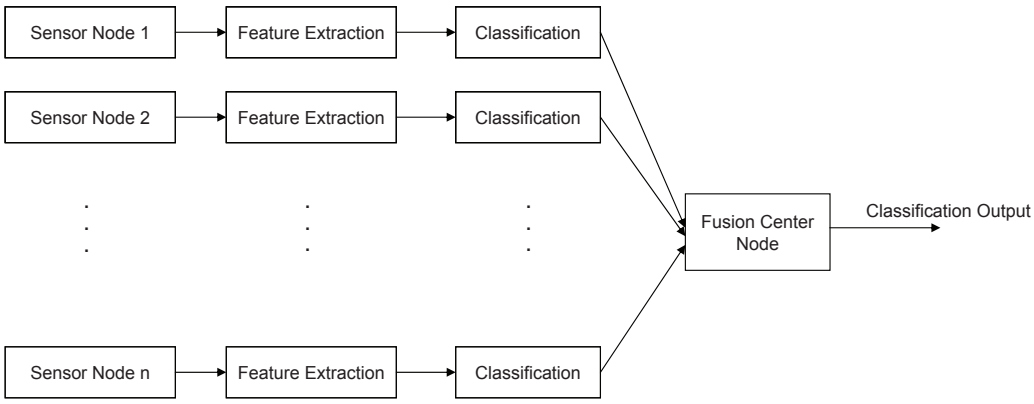


Figure 2.1: Classification fusion process for WSN

Feature extraction, local classification and fusion are the main parts in target classification.

2.1.1 Feature Extraction

Feature extraction is an important concept for target classification in WSN. The performance of the classifiers increases if the quality of the extracted feature vector is good enough to process [28]. Feature extraction depends on robust signal processing. Raw time-series signals (acoustic, seismic, thermal, etc.) are processed to form feature vectors. Sample feature extraction process is shown in Figure 2.2.

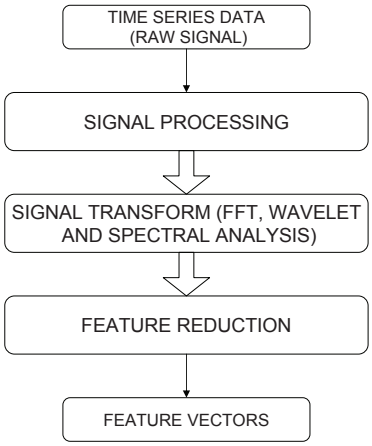


Figure 2.2: Block diagram of classical feature extraction

Feature extraction starts with the processing of the received raw time-series data signals. Signal processing phase is an optional phase to increase the signal quality. In [28], DC (Direct Current) component of the original signal is eliminated and noise from the original signal is removed during signal processing phase. After the signal processing phase, signal transform phase begins. Three most popular transforms for feature extraction are Fast Fourier Transform (FFT), Wavelet Transform and Spectral Analysis.

Fourier Transform (FT) is a calculation in which signal wave can be seen not only in time domain but also in frequency time domain. Generally, a signal wave is shown in a graph with time as the horizontal axis, and the amplitude as the vertical axis. However, by means of FT, signal wave can also be examined in frequency as the horizontal axis [21]. To analyze continuous waveform, data should be sampled in order to produce the time series of discrete samples. Such discrete samples are handled by Discrete Fourier Transform (DFT). In other words, FT requires continuous input functions whereas DFT requires discrete input functions. Finally,

FFT is an efficient method to compute DFT of a time series. By carrying out coefficients of the DFT iteratively, FFT saves considerable amount of time [8].

Like FT, wavelet transform is another mathematical tool for signal analysis [26]. Wavelet transform decomposes the input signal into different scales with different levels of resolution. Since wavelet transform is based on a square-integrable function and group theory representation, decomposition into scales is possible. Unlike FT, wavelet transform provides local representation of the signal in both time and frequency. When time-frequency resolution is needed, wavelet transform is suitable for analyzing a signal [26].

Spectral analysis reveals the frequency information of a target, and serves as a major feature [6]. Amplitude statistics, shape statistics and peak locations can be derived from using spectral analysis [28].

Power Spectrum Density (PSD) values can be generated after FFT or wavelet transforms. PSD values are the energy distributions of frequency features [6]. The feature vectors used by the classifiers can be chosen among the PSD values.

After signal transform phase, some reduction is made on feature vectors. Feature vectors may come large in size, and not every value they carry has the same importance. Some features are very close to each other among classification classes; therefore they have very little effect on the classification process. Removing unimportant feature vectors will speed up the classification computations. Principal Component Analysis (PCA) is a well known technique to reduce the feature vector size. PCA tries to reduce the feature vectors to a sufficient set. The most important information is kept untouched in the feature sets by a linear transformation matrix. The eigenvectors of the feature vector covariance matrix construct the linear transformation matrix [28]. However, since computation of the eigenvectors is difficult, PCA is not preferred in some applications [6]. In Chapter 3, we reduce the size of the feature vectors by comparing values of the feature vectors for the classification classes.

2.1.2 Local Classification

In the process of target classification in WSN, local classification takes place after feature vectors are extracted from sensor nodes. Wireless sensor nodes individually perform their local classifications before they go into fusion for the final result. After local classification

phase, every sensor node has a result for the target. In local classification phase, a good classifier has to be selected for proper classification. For classifiers to execute, they need a training set of feature vectors as an input besides feature vector to be classified. Training set includes features with right class labels attached to them. The classifier makes inferences using the samples in the training set. Right choice of the training set yields better classification accuracy. There are three types of classifiers used extensively in target classification in WSN. They are k-Nearest Neighbor (kNN), Maximum Likelihood (ML) and Support Vector Machine (SVM) classifiers. Brief information about the three classifiers is listed below [17]:

- kNN classifier is an easy yet a powerful classifier. When an input feature vector arrives to be tested, the Euclidean distance is calculated between feature vector and all the vectors in the training set. Then the nearest k feature vectors from the training set are determined. The class labels of the nearest k feature vectors are then combined by majority voting rule. That is, the class label with a maximum count among nearest k feature vectors is chosen as the result. If k is selected as 1, then kNN is called as nearest neighbor classifier. kNN produces very accurate results; however it may not be suitable if the training set size increases, as it requires too much memory space and power.

- In ML classifier, the distribution of feature vectors with the same class label within a training set is modeled as a mixture of Gaussian density functions. As the name implies, ML tries to find the training feature vector with the maximum likelihood to input feature vector. In other words, the principle can be stated as: Find an estimate for the input feature vector, which maximizes the likelihood of observing of those data which were actually observed (training set feature vectors) [11].

- The key idea in SVM is to map the current feature vector space to a higher vector space. For a feature vector space with N dimension, SVM maps this N dimension space to an M dimension space (M > N). By raising feature vector space, the training set feature vectors become more separable, therefore easy to classify. The disadvantage of SVM is that training phase can take long time. However, once the training set is completed, its calculation is rather easy. Generally, different SVM training is needed for each class.

2.1.3 Fusion

Fusion for target classification is the process of aggregating data from the sensor nodes and deciding on a final classification result. By efficient fusion, not only the performance of the overall system increases but also WSN does not need to depend on data coming from only one sensor node. Various fusion types exist in target classification in WSN. Three fusion methods can be counted as: Temporal fusion, multi-modality fusion and multi-sensor fusion [32].

- Temporal fusion is a time based method. Sensor nodes detect a target and start to sense and extract data from the target. The data gathered while the target is passing can be fused. Signals detected have different timestamps. Therefore, they can be treated as independent data and can be fused in a sensor node.

- Like temporal fusion, multi-modality fusion is done inside a sensor node (intra-node operation). If the sensor nodes have the ability to sense data using different modalities (acoustic, seismic, thermal, etc.), the results coming from these modalities can be fused. The data for fusion is considered as independent, since it comes from different channels.

- The final main type of fusion is multi-sensor fusion. This fusion is done among sensor nodes and it is commonly used in target classification process. To handle uncertainty and remove the faulty sensor nodes out, multi-sensor fusion is required.

Decision fusion is a type of multi-sensor fusion. In decision fusion, the sensor nodes make local decisions, and fusion is made among these decisions. The data fused must be independent. Although any fusion type in the classification process might be called as data fusion, Brooks et al. define data fusion as the fusion of correlated data [5]. Combination of the feature vectors from all sensor nodes can be considered as data fusion. The pros and cons of data fusion versus decision fusion can be listed as follows [5]:

- Decision fusion presents lower computational burden and fewer amounts of training data. It is preferable when training data is not very large.

- Data fusion gives better performance at the cost of computational and communication burden for correlated measurements.

- Data fusion is preferred to be used among modalities (seismic, acoustic, etc. inside a sensor node) whereas decision fusion is better to be used across sensor nodes.

- Some recent results [7] show that decision fusion performs better in a scenario where some sensor nodes are malfunctioning.

Decision fusion for target classification in WSN can be done by using variety of methods. Different parameters present in the sensor network environment may provide many techniques and combinations for decision fusion. Some possible fusion scenarios are as follows:

- Majority voting is the simplest but an efficient fusion technique. After all the sensor nodes make their local classifications and yield a decision, the decisions for each class label are counted. After this step, three different versions of majority voting behave differently [24]. In unanimous voting, all the sensor nodes should agree on the same decision. That is, the count of the decided class should be equal to the number of sensor nodes. Another version is simple majority, in which the majority of the sensor nodes should decide on the same class label to make that class label be chosen. Percentage of a class label should exceed %50 to be selected. In plurality voting, the class label with highest count is selected whether sum of the votes exceeds %50 or not. Generally the term majority voting implies the third version, plurality voting.

- Other fusion techniques apart from majority voting are mainly based on the parameters gathered from the WSN environment. These parameters can be distances (sensor nodes - target distance [10] or sensor nodes - sink node distance [27]), signal properties such as signal quality, signal energy and signal to noise ratio (SNR) [10], sensor energies [27], and so on. Combinations using these parameters are commonly used. The main goal is to select the optimum sensor node(s) for classification. Maximum A Posterior (MAP) Bayesian [10] and Dempster-Shafer [6] algorithms work on the principle of giving probability or belief values to sensor nodes. In d_{max} algorithm [10], only the sensor nodes within a given distance are considered. After the sensors are determined in the given distance, d_{max} algorithm works like majority voting algorithm.

- Fuzzy logic is again a good technique for better classification in WSN. The parameters above can be integrated into fuzzy fusion techniques. Two or more parameters are fuzzified and then based on some fuzzy rules; the defuzzified result may lead optimum

sensor node(s) selection.

2.1.4 Basics of Fuzzy Set Theory

Fuzzy sets are introduced by L. Zadeh in 1965. Fuzzy set theory is the main concept of fuzzy logic. In crisp set theory, an element is either belongs to the set or not. A definition of the crisp set can be expressed as in Equation 2.1 [23].

$$\mu_A(x) = \begin{cases} 1, & \text{if } x \text{ is an element of set A} \\ 0, & \text{if } x \text{ is not an element of set A} \end{cases} \tag{2.1}$$

In Equation 2.1, $\mu_A(x)$ represents the characteristic function for set A. According to definition, if x is a member of the set A, $\mu_A(x)$ has the value of 1. On the other hand, when x is not a member of the set A, $\mu_A(x)$ has the value of 0.

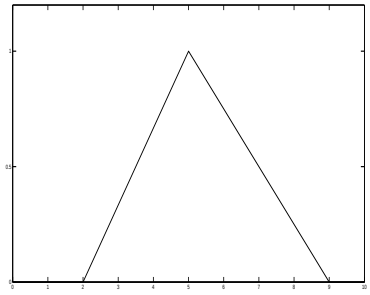
In fuzzy set theory, the situation mentioned about the crisp set theory is extended. The membership function for a fuzzy set may generate a real value in the interval [0,1] [34]. Fuzzy sets can be generalized as in Equation 2.2 [23].

$$A = \{x, \mu_A(x) \mid x \in X\} \tag{2.2}$$

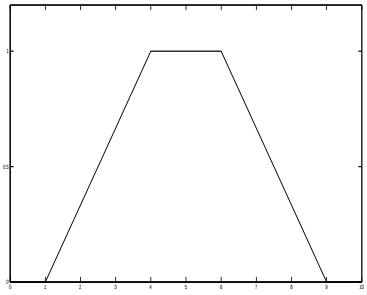
In Equation 2.2, X denotes the universe of discourse and x denotes elements of it. $\mu_A(x)$ is a membership function of x in A.

Designing robust membership functions is very important for fuzzy sets. Changing the parameters of the membership functions may cause different results in fuzzy systems [23]. Fuzzy membership functions must map each element of a set to a continuous membership value form 0 to 1. Several basic functions are used in fuzzy systems to generate fuzzy membership functions. Some examples are Triangular, Trapezoidal, Gaussian, Bell and Sigmodial membership functions [15]. Figure 2.3 shows the examples of these fuzzy membership functions.

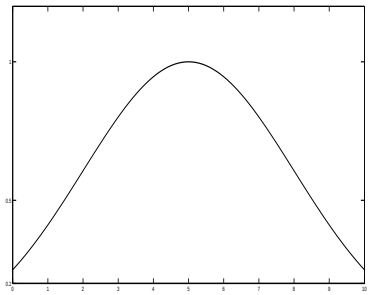
Fuzzy membership functions are chosen according to data to be used. Thus, the functions used may differ from application to application. The most commonly used fuzzy membership functions are triangular and trapezoidal fuzzy membership functions.



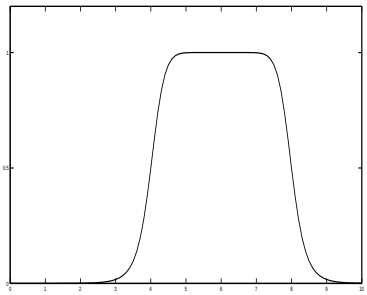
Triangular



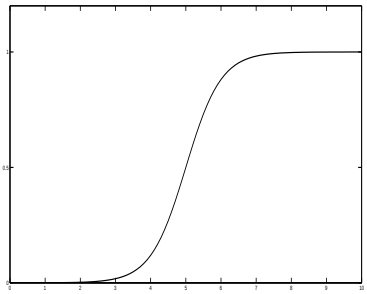
Trapezoidal



Gaussian



Bell



Sigmodial

Figure 2.3: Example sets of membership functions

Fuzzy inference is a framework for depending on the concepts; fuzzy set theory, fuzzy if-then rules and fuzzy reasoning [15]. Fuzzy inference system has three main conceptual components [15]:

- First component is *rule base*. It contains a selection of fuzzy if-then rules.
- Second component is *a database or dictionary*, which has the definitions of the membership functions used.
- Last component is named as *reasoning mechanism*, where inference procedure is realized based on the fuzzy rules to produce a conclusion.

Fuzzy inference systems actually map crisp input value(s) into a crisp output value [19]. Blocks of a simple fuzzy inference system is shown in Figure 2.4 [23].

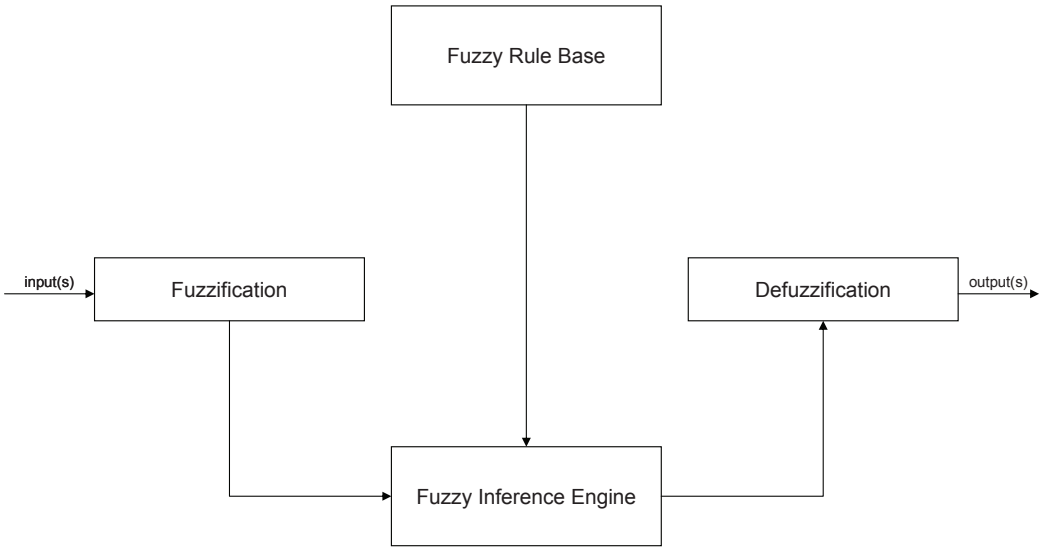


Figure 2.4: Basic fuzzy inference system

In the fuzzification phase, the input variables are fuzzified. The variables initially have crisp inputs and they have to be converted. Membership functions are created for all the input and output variables. The degree is calculated based on the input variables belonging to their appropriate fuzzy sets using membership functions [13]. Membership functions are labeled with some linguistic terms such as *high*, *low*, *small*, *big* etc.

Fuzzy rule base is comprised of fuzzy if-then rules. Every fuzzy inference system has set of fuzzy if-then rules with meaningful linguistic interpretations. These fuzzy if-then rules may be obtained from numerical data or some experts familiar with problem [19]. A sample if-then rule has the following format [15]:

$$\text{if } x \text{ is A then } y \text{ is B} \tag{2.3}$$

In 2.3, x is the input variable while y is the output variable. A denotes the linguistic variable for input variable and B denotes the linguistic variable for output variable. "x is A" is called antecedent and "y is B" is called consequence. These defined rules affect the fuzzy inference system. All the related fuzzy if-then rules are used when a defuzzified output value is generated for a given input value.

Fuzzy inference engine performs the inference operations using the fuzzy rules for a reasonable output [23]. There are three common types of fuzzy inference systems namely Mamdani fuzzy model, Sugeno fuzzy model and Tsukamoto fuzzy model [23]. The main difference among these inference systems are about their fuzzy rule consequents [33]. Different fuzzy rule consequents lead to different aggregation and defuzzification procedures [23]. Typical properties of these fuzzy inference systems are listed below:

- Mamdani proposed Mamdani fuzzy inference system to control a steam engine and boiler combination by control rules of some human operators [14]. In Mamdani fuzzy inference system, first the fuzzy inputs are fuzzified in fuzzification phase. After fuzzification phase, the rule evaluation is performed. Mamdani fuzzy rules have the form expressed in 2.4 with X_i and Y are the input and output linguistic variables, respectively, and with A_i and B being linguistic labels with fuzzy sets associated defining their meaning [9].

  $$\text{if } X_1 \text{ is } A_1 \text{ and ... and } X_n \text{ is } A_n \text{ then } Y \text{ is } B \tag{2.4}$$

  If a fuzzy rule has more than one antecedent, then logical operators OR (maximum) and AND (minimum) are used to estimate the result number. If OR operator is used, highest number is chosen, while if AND operator is used, lowest number is chosen. When the input values are zero, the output value is also zero. Aggregation of the rule

outputs is performed after rule evaluation process. This kind of aggregation forms a fuzzy set for Mamdani fuzzy inference system. In the final phase, the fuzzy set, which is the output of the rule aggregation phase, is defuzzified. The all fuzzy set is reduced to a single crisp value. This crisp value is considered as the result of the fuzzy logic inference process. Centroid method is generally used for defuzzification [13].

- Takagi, Sugeno, and Knag proposed Sugeno fuzzy inference system [14]. Typical fuzzy rule for Sugeno fuzzy inference system is given in 2.5 where A and B fuzzy sets in the antecedent and z is a crisp function for consequent. It can be observed that, the consequent part contains a function unlike Mamdani fuzzy inference system.

$$\text{if } x \text{ is A then } y \text{ is B then } z = \text{f}(x, y) \tag{2.5}$$

  f(x, y) function is a polynomial with the input variables x and y most of the times. However, it can also be any function that describes output of the system within the fuzzy region specified by the antecedent of the rule [15]. The construction of Sugeno fuzzy inference system is usually realized in two steps [1]. In the first step, determination of the fuzzy sets in the rule antecedents is done. Then, the parameters for the consequent functions are estimated. These consequent functions are chosen to be linear.

- Last fuzzy inference system to be mentioned is Tsukamoto fuzzy inference system. In Tsukamoto fuzzy inference system, the consequent of each fuzzy if-then rule is represented by a fuzzy set with a monotonical membership function [29]. For each rule, the output is defined as a crisp value and overall output is the weighted average of each rule's output [14]. Since each rule infers a crisp output and outputs of the rules are aggregated using the method of weighted average, Tsukamoto fuzzy inference system avoids the time-consuming process of defuzzification [15].

Last phase of a fuzzy inference system is defuzzification. By doing defuzzification, the fuzzy results of the outputs are transformed into crisp values [23]. The most commonly used defuzzification technique is centroid of area. Equation of the centroid of area technique is shown in Equation 2.6 [15]. In Equation 2.6, $\mu_{C'}$ is the aggregated output membership function.

$$z_{COA} = \frac{\int_z \mu_{C'}(z) z dz}{\int_z \mu_{C'}(z) dz} \tag{2.6}$$

2.2 Related Work

In this section, we mention about the related work done for the classification process and fuzzy logic approaches in WSN.

2.2.1 Classification Process in WSN

Many studies have been done about target classification in WSN. Even if the main focus is on a specific part of the classification process, almost every study includes whole classification cycle for performance evaluation.

Duarte et al. provide a baseline study in vehicle classification in distributed sensor networks (DSN) [10]. In this paper, whole classification process for sensor networks is conducted using real data set. Data is collected in a wireless distributed sensor networks (WDSN) experiment at Twenty-nine Palms, CA in November 2001. Seventy-five sensor nodes are used to classify four target vehicle classes: Assault Amphibian Vehicle (AAV), Main Battle Tank (M1), High Mobility Multipurpose Wheeled Vehicle (HMMWV) and Dragon Wagon (DW). Sensor nodes are capable of recording acoustic, seismic and infrared signal. In this thesis, we also use a subset of this dataset in order to evaluate our fuzzy decision fusion methods. Features are extracted using FFT. After feature extraction, local classification is done using kNN, ML and SVM classifiers. The performance of these classifiers is compared for seismic and acoustic modalities. Fusion algorithms such as majority voting, nearest neighbor, MAP Bayesian and d_{max} algorithms are applied and compared using classification results of the sensors. Since Duarte et al. provide a real dataset to research community; many classification studies use this dataset for evaluating target or vehicle classification algorithms.

Another classification process for sensor networks is studied by Wang et al. [32]. In this study, classification process starts with feature extraction as usual. Features are formed by Wavelet analysis coefficients and PSD values. As a classifier, kNN classifier is chosen. Afterwards, local classification time based temporal fusion is made. Up to this point, the process is done for both acoustic and seismic signal feature vectors. Multi-modality fusion is implemented to fuse results from acoustic and seismic channels. A method called Behavior-Knowledge Space (BKS) is used for multi-modality fusion. In this algorithm all the possible class label combinations are formed, and then the combinations are assigned to a class label based on

training sets. The majority voting is applied to assigned class labels. For example, consider a problem with two class labels w_1 and w_2 and two classifiers C_1 and C_2. The all possible class label combinations are w_1w_1, w_1w_2, w_2w_1 and w_2w_2 meaning first class label is from C_1 and second class label is from C_2. These four combinations are assigned to a class label w_1 or w_2 according to their occurrence in training sets. Suppose w_1w_1 has occurred 15 times in training sets, 10 times with true class label w_1 and 5 times with class label w_2. Then w_1 is assigned to this combination. When all four combinations find their corresponding class labels, most found class label is the result of multi-modality fusion [24]. When temporal fusion and multi-modality fusion is completed, Multi-Resolution Integration (MRI) algorithm is used for multi-sensor fusion. The basic idea is to form an overlap function on outputs of the sensors and resolve this function at various successively finer scales of resolution. Wang et al. also presented the architecture in this study called Mobile-agent-based DSN (MADSN). In MASDN, a mobile-agent-based collaborative sensor fusion is used. Overlap functions are calculated as in MRI, and then a mobile-agent carries these functions from one sensor node to a second sensor node. The overlap functions of two sensor nodes are combined and the result is carried to another sensor node. If the classification accuracy is achieved at the final sensor node, the process is terminated. Otherwise, the mobile-agent continues its migration. Classification accuracy using 1-sec segments, temporal-fusion, multi-modality fusion and fusion by MASDN architecture is evaluated in the end of the study.

Study of Sayeed et al. is about detection, classification and tracking in DSN [17]. In classification part, some information is given about the spectral features used for classification. Acoustic and seismic PSD of tracked and wheeled vehicles are shown in detail. The three classifiers, kNN, ML and SVM are used for classification. To compare these classifiers, available feature vector data is divided into three parts. When first part is tested, the other parts are used for training. Same procedure holds for second and third parts. This technique is called 3-way cross validation. The kNN with k = 1, ML and SVM classifiers are compared using low bandwidth seismic data and wide band acoustic data.

Classification fusion in WSN has also been studied by Chun-Ting et al. [6]. In this study, acoustic signals are used for classification. Wavelet transforms of the acoustic signals are put into process and also a comparison is made between FFT and wavelet transform. Classifier choice is a weighted kNN classifier. Weighted kNN classifier works like kNN classifier however the training set feature vectors have some weights according to their Euclidean distance

to input feature vector. Study includes comparison of weighted kNN classifier over kNN classifier. Dempster-Shafer theory is used for fusion. Sensor nodes get weights according to their weighted kNN classification. Therefore, in this application, behavior of Dempster-Shafer theory is similar to both data fusion and decision fusion. Outputs of weighted kNN classifier construct the belief functions to the sensor nodes, and then the desired sensor node(s) are selected based on the belief functions. The classification accuracies of majority voting fusion algorithm and Dempster-Shafer theory are compared.

Tian et al. study target classification on ground sensor systems [28]. A new feature extraction algorithm, Spectral Statistics and Wavelet Coefficients Characterization (SSWCC) is proposed for extracting more robust feature vectors. SSWCC has the statistical features coming from PSD, spectral and wavelet analysis. PCA is used for reducing the size of the feature vectors. The loss of energy in PCA calculation and k effect on the kNN classification are examined. Moreover, the performance of the different classifiers and training / test set selection is evaluated.

2.2.2 Fuzzy Logic Approaches in WSN

Having the power of dealing with uncertainty, fuzzy logic is used widely in WSN applications. In this part, we mention some remarkable studies about fuzzy logic in WSN.

Cluster-head selection in WSN is a popular and recent topic studied using fuzzy logic. Study of Gupta et al. is an example work of this area [13]. In this study, sensor nodes are separated as clusters and a cluster-head is selected for each cluster. The cluster-heads are responsible for the data aggregation within its cluster. The fuzzy variables include sensor node energy level, sensor node concentration and sensor node centrality. Based on these fuzzy variables and fuzzy rules, sensor node cluster-head election chance is calculated. It is obvious that sensor nodes having lower energies, lower concentrations and closer centralities possess less chance of cluster-head election than the sensor nodes having higher energies, higher concentrations and nearer centralities. At the last part of the study, the cluster-head election algorithms in different scenarios are compared with fuzzy approach. Another study on cluster-head election in WSN belongs to Kim et al. [16]. Similar to study of Gupta et al., in this study, sensor node energy and local distance of the sensor node (the sum of distances between the current sensor node and all the nodes within a distance r) are fuzzy variables. Cluster-head election

chance is calculated depending on these fuzzy variables and fuzzy rules. Proposed cluster-head election algorithm using fuzzy logic is compared with previous cluster-head election algorithms on network lifetime, standard deviation of the energy remaining when the first node is dead, cluster formation and number of clusters/alive nodes for every round.

Fuzzy fusion methods are studied by Samarasooriya et al. [25] and Su et al. [27]. In the study of Samarasooriya et al., local decisions, that are generated by sensor nodes, are considered with having various degree of accuracy. Each sensor node has its local decision result and an error probability. These error probabilities are modeled using fuzzy logic. Therefore, multi-sensor fusion is done among sensor nodes having fuzzy error probabilities. The global decision is made and then fuzzy result is defuzzified. Su et al. use fuzzy data fusion in cluster-based wireless sensor network. The goal is to make an efficient fusion algorithm to minimize the energy spent for sensing, processing, communication and aggregation of more useful and reliable data. The fuzzy variables are distance (distance between sensor node and sink node) and quality of received SNR at the sink node. The expected fuzzy output is named as state. State variable represents the magnitude of participation for an output. As most of the fuzzy systems, state variable has values depending on the fuzzy variables and fuzzy rules. The reported magnitude value based on the highest/weighted defuzzified value and the error between the initial and the estimated magnitude value based on the highest/weighted average defuzzified value are compared using Mamdani and Tsukamoto fuzzy inference methods.

Although many fuzzy fusion studies are done in research community, no fuzzy fusion method is known to be applied to target classification problem in WSN. In our thesis, we developed fuzzy decision fusion mechanism for single target classification in WSN. Our fuzzy decision techniques improves classification accuracy for the target classification problem in WSN.

CHAPTER 3

DECISION FUSION FOR CLASSIFICATION PROCESS

In this chapter, we propose fuzzy decision fusion methods for single target classification in WSN. In addition to our approach, we mention the details of three decision fusion algorithms which we use for comparison with our algorithm at Chapter 4. These three algorithms are majority voting, nearest neighbor and d_{max} decision fusion algorithms. For all algorithms mentioned, we also present running examples. Before getting into the details of the decision fusion algorithms, we first go over the feature reduction and training set formation approach we use throughout the classification process.

3.1 Feature Reduction

The feature extraction techniques, which transform time-series signal (raw data) into feature vectors, may result in high-dimensional feature vectors. Most of these high-dimensional feature vectors behave similarly and therefore become redundant. To save time for calculations on sensor nodes by reducing feature vector dimensions, an effective algorithm should be implemented. We have implemented a feature reduction algorithm based on the differences between feature vectors of different target classes. Training sets are used for different classes to calculate the difference. We pick dominant feature vectors which may help us through the target classification. The feature vectors which show similarity from class to class are eliminated. The algorithm used for a given test run we use is shown in Algorithm 3.1.

Algorithm 3.1 tries to find the f_{Count} feature vectors for a given test run, R_{test}. Firstly, the average values of feature vectors are calculated for each run using training set, T. The test run is held out from the training set to perform more realistic classification. In other words,

```
1: n ← Number of classes
2: f_Count ← Desired feature vector count
3: AVGMAT ← Matrix for average values of feature vectors for each class
4: SUMMAT ← Matrix for summation of feature vectors for each class
5: T ← Training set
6: R_test ← Test run
7: for all run R in T do
8:     if R_test = R then
9:         Continue for the next run in training set
10:    end if
11:    W ← Class in the run R
12:    SUMMAT[W] ← Column-wise summation of all feature vectors for R
13: end for
14: AVGMAT ← Column-wise average of all rows in SUMMAT
15: FTDIFFSUM ← Sum of differences for each feature vector
16: for all Feature vector column FC in AVGMAT columns do
17:    for i = 1 to n do
18:        sum ← 0
19:        for j = i + 1 to n do
20:            sum ← Difference between FC[i] and FC[j]
21:        end for
22:    end for
23:    FTDIFFSUM[FC] ← sum
24: end for
25: Sort FTDIFFSUM according to sum values of feature vector columns
26: Pick f_Count feature vector columns from sorted FTDIFFSUM
```

Algorithm 3.1: Feature Reduction Algorithm for a Given Test Run

the reduction of feature vectors for a given test run is not affected by the feature vectors in that test run. $SUMMAT$ structure has the column-wise summation of the feature vectors for each class and $AVGMAT$ structure has the averaged values of feature vectors for each class. After all classes have averaged feature vector values, dominant feature vectors are determined using the data in $AVGMAT$. To determine dominant feature vectors, differences of the feature vectors of classes are calculated. Sum of differences for each feature vector is kept in the structure $FTDIFFSUM$. The f_{Count} feature vectors having highest difference values are selected as the basis feature vectors. For example, consider a case with two target classes. The difference of feature vectors for these two classes is calculated for each feature vector, and f_{Count} dominant feature vectors are selected. When the target class count is three, then three difference calculations are done (These calculations are between first class and second class, between first class and third class, and lastly between second class and third class). These three difference values are later summed to get a single difference value. Then f_{Count} feature vectors are chosen. Obviously, this algorithm works efficiently when the target class count is small. As the order of the algorithm is polynomial, $n(n-1)/2$ difference calculations are required for n target classes.

In Figure 3.1, averaged acoustic feature vectors of AAV and DW vehicles are shown. Feature vectors for AAV and DW vehicles are provided by DARPA SensIT project [10]. Horizontal axis represents the 50 feature vectors and vertical axis represents value of these feature vectors. The acoustic feature vectors in this figure are averaged for each class to test an AAV run.

The differences for averaged feature values for two vehicles mentioned above are shown in Figure 3.2. The absolute values of the difference value between two classes are calculated for each feature vector separately. Horizontal axis again represents the 50 feature vectors and vertical axis represents difference value.

In Figure 3.2, 5 feature vectors having highest difference values are 5, 11, 4, 7, 10 respectively. If desired feature vector count is 5, then these feature vectors are used for target classification process.

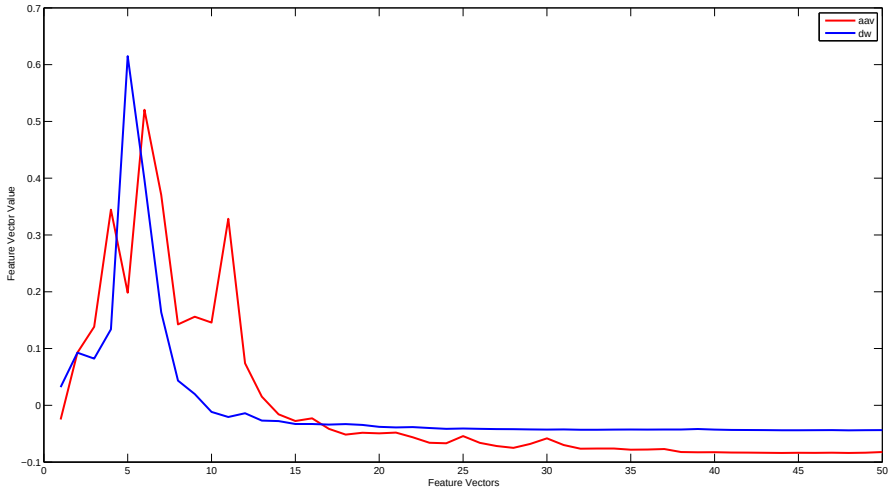


Figure 3.1: Averaged feature vectors for AAV and DW Vehicles

3.2 Training Set Formation

Training sets are mainly used by classifiers to classify a target. Classifiers take an input feature vector and they come to a conclusion by evaluating the relationship between input feature vector and the feature vectors in the training sets. Proper formation of the training sets yields better classification results. In this thesis, node-based training sets are used. For example, if the aim is to try forming a training set for a test run in the local classification process of Node 1, only the Node 1 feature vectors from all sample runs except test run are collected. The algorithm is shown in Algorithm 3.2.

In Algorithm 3.2, *Nodes* structure has the sensor nodes data and $NodesFT$ structure has feature vectors for each sensor node. First of all, the input test run, R_{test}, is excluded from the runs which form the training set similar to the algorithm for feature reduction. Training set is kept in the structure T. The feature vectors in the remaining runs are candidates for the training set. For each run, feature vectors belonging to a specific node is combined. $NodesFT$ structure used in this algorithm holds $< key >< value >$ pairs. Sensor nodes are the keys and feature vectors are the values. For example, all the feature vectors for Node 1 can be reached at *NodesFT[1]*, for node 41 at *NodesFT[41]* and so on. Therefore, every sensor node becomes the owner of its training set for a given test run.

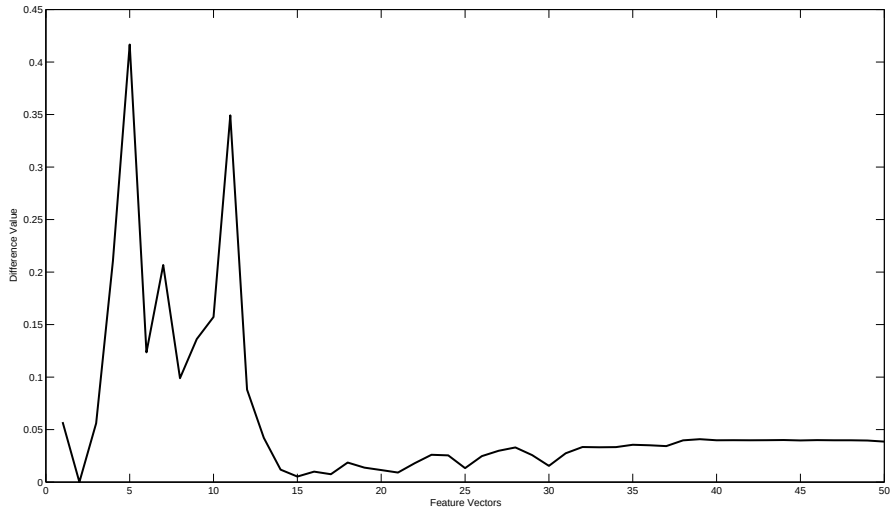


Figure 3.2: Difference values of feature vectors for AAV and DW Vehicles

Node-based training sets approach reduces the training set size considerably. Small training sets mean less computation on the sensor nodes in local classification process. By this way, execution becomes faster and sensor node consumes less energy.

3.3 Majority Voting Decision Fusion

In majority voting decision fusion (MVDF), every voter has one vote to use for selecting any candidate [31]. The candidate which collects highest number of votes is selected among other candidates. We refer to third version of majority voting, plurality voting, with the term majority voting. It is beneficial because its simplicity and low error count. The algorithm fails only when majority of the votes belongs to more than one target class. The algorithm we implement for MVDF is depicted in Algorithm 3.3.

In Algorithm 3.3, *Nodes* structure has the sensor nodes data and *NodesLC* structure has the local classification result for each sensor node.The algorithm begins with going over all the sensor nodes in the system. The local classification result for each sensor node is obtained. Counts of the class labels are stored in the structure *ClassCount*. After a pass through sensor nodes, *ClassCount* is filled with the counts of the class labels. Then, the *resultClass*, the class label with highest number of counts in *ClassCount*, is picked. A condition may prevent

```
1: T ← Training set
2: R_test ← Test run
3: Nodes ← Sensor nodes array
4: NodesFT ← Sensor nodes - feature vectors structure
5: for all Sensor Node S in Nodes do
6:     for all run R in T do
7:         if R_test = R then
8:             Continue for the next run in training set
9:         end if
10:        NodesFT[S] ← Feature Vectors of node S for run R
11:    end for
12: end for
```

Algorithm 3.2: Node-Based Training Set Generation for a Given Test Run

```
1: Nodes ← Sensor nodes array
2: NodesLC ← Sensor nodes - local classification result matrix
3: ClassCount ← Class count - Sensor nodes matrix
4: for all Sensor Node S in Nodes do
5:     class ← Find local classification result in NodesLC for sensor node S
6:     Increment ClassCount[class]
7: end for
8: resultClass ← Class label with highest count in ClassCount
9: if No other class with same count exist in ClassCount then
10:    Select resultClass as the final decision
11: else
12:    Reject the data sample
13: end if
```

Algorithm 3.3: Majority Voting Decision Fusion

resultClass from being the final decision. *resultClass* should be the only highest counted class label in *ClassCount*. If this circumstance is provided, *resultClass* can be announced as the final decision. Otherwise, if the circumstance fails, the sample data for the MVDF is rejected, no decision is found.

Although MVDF provides an easy implementation, it also contains some drawbacks. First of all, since no weight is given to any sensor node, all the sensor nodes are considered to be equal. However, some sensor nodes can be more reliable due to their locations or structures. These sensor nodes should be treated differently from other regular sensor nodes. Moreover, data can be rejected in MVDF. Having same number of highest counted class labels prevents system from producing a final decision.

3.3.1 Running example for MVDF

In this part, we introduce a running example to show how the MVDF algorithm works. In this example, we simply use five sensor nodes namely $s1$, $s2$, $s3$, $s4$ and $s5$ and also three class labels $c1$, $c2$ and $c3$. Table 3.1 presents the classification results for MVDF algorithm through five sample points.

Table 3.1: Sample classification flow for MVDF algorithm

SP	$s1$	$s2$	$s3$	$s4$	$s5$	*Result*
1	$c1$	$c1$	$c1$	$c2$	$c3$	$c1$
2	$c1$	$c1$	$c2$	$c2$	$c2$	$c2$
3	$c3$	$c3$	$c3$	$c3$	$c3$	$c3$
4	$c2$	$c1$	$c2$	$c2$	$c2$	$c2$
5	$c1$	$c1$	$c2$	$c2$	$c3$	*Reject*

In Table 3.1, rows show the sample points to be classified. First column is for sample points, the next five columns are for local classification results of sensor nodes and final column is for fused classification result. In each row, the classification fusion is done according to MVDF algorithm. At sample points from 1 to 4, the class labels of the sensor nodes are simply counted and the class label with highest count is selected as the final result. However, at sample point 5, the class labels $c1$ and $c2$ both appear twice. In this case, the sample point is rejected and no result is generated.

3.4 Nearest Neighbor Decision Fusion

Nearest neighbor decision fusion (NNDF) is another efficient decision fusion technique. Unlike MVDF, NNDF does not treat every sensor node in the same way. The distance between the target and a sensor node is calculated for each sensor node. After calculation of the distances, the sensor node closest to the target is chosen [10]. The local classification result for this sensor node is the final decision result. The implementation details of the NNDF algorithm is shown in Algorithm 3.4.

```
1: Nodes ← Sensor nodes array
2: NodesLC ← Sensor nodes - local classification result matrix
3: Distances ← Target distance - Sensor nodes matrix
4: for all Sensor Node S in Nodes do
5:     dist ← Calculate the distance between the target and sensor node S
6:     Distances[S] ← dist
7: end for
8: Sort Distances
9: closestS ← Select the sensor node with minimum distance in Distances
10: Select NodesLC[closestS] as the final decision
```

Algorithm 3.4: Nearest Neighbor Decision Fusion

Like in Algorithm 3.3, in Algorithm 3.4 *Nodes* structure has the sensor nodes data and *NodesLC* structure has local classification result for each sensor node. Another structure *Distances* exists for the data of the distance between target and sensor nodes. Firstly, the distance of the all sensors to the target is calculated and *Distances* is filled with this data. The distances are sorted and the sensor node having the minimum distance to the target is found out. *closestS* represents this sensor node. The local classification result of *closestS* is accepted as the final decision. If more than one sensor node has the same minimum distance to the target, the first sensor node evaluated is picked.

NNDF may produce good results using the advantage of being close to the target. However, relying on a single node can be hazardous. If the nearest sensor node somehow produces faulty results, then the system always obtains wrong data.

3.4.1 Running example for NNDF

A running example of NNDF algorithm is demonstrated in this section. Like at Section 3.3.1, we use five sensor nodes namely $s1$, $s2$, $s3$, $s4$ and $s5$ and also three class labels $c1$, $c2$ and $c3$. Table 3.2 shows the classification results for NNDF algorithm through five sample points.

Table 3.2: Sample classification flow for NNDF algorithm

SP	$s1 - Dist$	$s2 - Dist$	$s3 - Dist$	$s4 - Dist$	$s5 - Dist$	$Result$
1	$c1 - 100$	$c1 - 180$	$c1 - 250$	$c2 - 300$	$c3 - 300$	$c1$
2	$c1 - 75$	$c1 - 100$	$c2 - 200$	$c2 - 250$	$c2 - 275$	$c1$
3	$c3 - 100$	$c3 - 80$	$c3 - 100$	$c3 - 150$	$c3 - 200$	$c3$
4	$c2 - 150$	$c1 - 100$	$c2 - 50$	$c2 - 100$	$c2 - 150$	$c2$
5	$c1 - 200$	$c1 - 150$	$c2 - 100$	$c2 - 80$	$c3 - 80$	$c2$

In Table 3.2, rows and columns demonstrate same values as in Table 3.1 with an exception. In this table, the distance between the sensor node and target is given with the local classification result separated by a hyphen for each sample point. In NNDF algorithm, the sensor nodes with minimum distance to the target are preferred. Therefore, at sample points from 1 to 4, $c1$, $c1$, $c3$ and $c2$ are selected respectively for having minimum distances. At sample point 5, both $s4$ and $s5$ have same minimum distance to the target. In this case, the first sensor node evaluated, $s4$, is selected in order not to reject the sample point. Therefore $c2$, being the local classification result of $s4$, is the final result for sample point 5.

3.5 d_{max} Decision Fusion

Like NNDF, d_{max} Decision Fusion (DMDF) is also related to the distances between the sensor nodes and target. NNDF algorithm decides on the final decision result based on a single sensor node. However, in DMDF several sensor nodes may act in the classification process. A distance value, d, is chosen for DMDF algorithm. The sensor nodes ,of which distances to the target exceeds the value d, have no effect on decision fusion. The remaining sensor nodes are put into majority voting process to determine a final decision. Figure 3.3 illustrates the DMDF sensor selection method.

In Figure 3.3, the rectangle with T represents the target. The sensor nodes within the d

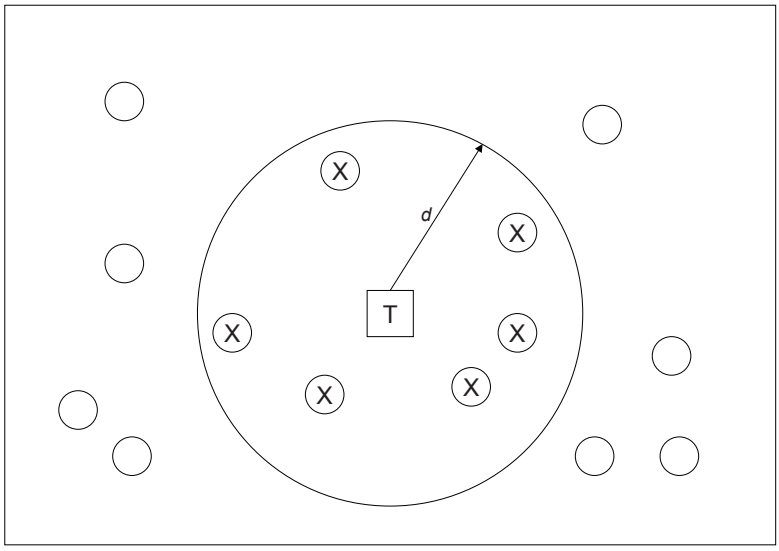


Figure 3.3: d_{max} decision fusion sensor node selection

distance are labeled with *X* sign. The decision fusion for DMDF algorithm is done using only these *X* signed sensor nodes.

A more formal definition of the DMDF algorithm is given in Equation 3.1 [10]. w_i denotes the weight of *i*th sensor node. The weight of a sensor node is either 0 or 1, based on its distance to target.

$$w_i = \begin{cases} 1 & d_i \leq d_{max} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

DMDF implementation details can be seen in Algorithm 3.5. *Nodes* denotes the sensor nodes data and *NodesLC* denotes local classification result for each sensor node in Algorithm 3.5. Like NNDF, the distance of the all sensors to the target is calculated and *Distances* is filled with this data in the first place. The *Distances* structure is then sorted according to distance values. The sensor nodes having the distance value less than or equal to *d* are determined. *ClassCount* structure is filled with number of class labels of these sensor nodes. All the other sensor nodes are eliminated. However, *TBSensor* is determined for future use in case it is needed. *TBSensor* is the sensor node which is closest to the target among the sensor

```
1: Nodes ← Sensor nodes array
2: NodesLC ← Sensor nodes - local classification result matrix
3: ClassCount ← Class label - count matrix
4: Distances ← Target distance - Sensor nodes matrix
5: d ← d distance value
6: TBSensor ← Tie break sensor node number
7: for all Sensor Node S in Nodes do
8:     dist ← Calculate the distance between the target and sensor node S
9:     Distances[S] ← dist
10: end for
11: Sort Distances according to distance values
12: for all Sensor Node - Distance Pair SD in sorted Distances do
13:     if Distance of SD less than or equal to d then
14:         Node ← Sensor node of SD
15:         Class ← Classification result from NodesLC for sensor Node
16:         Increment ClassCount[Class]
17:     else
18:         TBSensor ← Sensor node of SD
19:         Break for loop
20:     end if
21: end for
22: if ClassCount has 0 value for all classes or no TBSensor is found when ClassCount has
    more than one class having same highest count then
23:     closestS ← Sensor node having the closest distance
24:     Find the classification result from NodesLC for sensor closestS
25: else if ClassCount has more than one class having same highest count then
26:     Find the classification result from NodesLC for sensor TBSensor
27: else
28:     Find the class with highest count in ClassCount
29: end if
```

Algorithm 3.5: d_{max} Decision Fusion

nodes having distance to the target greater than d. Using *ClassCount*, a class label reported from maximum number of chosen sensor nodes is determined. This class label is the final classification result. If *ClassCount* has 0 value for all classes or no *TBSensor* is found when *ClassCount* has more than one class having same highest count, decision of the closest sensor node is taken into account like in the NNDF algorithm. If *ClassCount* has more than one class having same highest count, decision of the *TBSensor* is the final decision result.

DMDF algorithm has a place between MVDF and NNDF algorithms. If the sensor nodes are too far to the target and d value is small, then DMDF starts to act like NNDF algorithm. On the other hand, making d value big and deploying sensor nodes close to target may convert DMDF algorithm into MVDF algorithm. Therefore, selection of the d value plays an important role. The deployment of the sensor nodes should be studied enough before deciding the d value.

3.5.1 Running example for DMDF

We present a running example of DMDF algorithm in this section. Like at Section 3.3.1, we use five sensor nodes namely $s1$, $s2$, $s3$, $s4$ and $s5$ and also three class labels $c1$, $c2$ and $c3$. In Table 3.3, the classification results for NNDF algorithm through five sample points can be seen.

Table 3.3: Sample classification flow for DMDF algorithm

SP	$s1 - Dist$	$s2 - Dist$	$s3 - Dist$	$s4 - Dist$	$s5 - Dist$	$Result$
1	$c1 - 100$	$c1 - 275$	$c1 - 300$	$c2 - 450$	$c3 - 500$	$c1$
2	$c1 - 75$	$c1 - 100$	$c2 - 200$	$c2 - 300$	$c2 - 450$	$c1$
3	$c3 - 100$	$c3 - 80$	$c2 - 100$	$c1 - 150$	$c2 - 200$	$c3$
4	$c2 - 150$	$c1 - 100$	$c2 - 50$	$c1 - 100$	$c2 - 300$	$c2$
5	$c1 - 270$	$c1 - 300$	$c2 - 260$	$c2 - 280$	$c3 - 380$	$c2$

In Table 3.3, rows and columns illustrate same values as in Table 3.2 with distances at sensor node columns. In DMDF algorithm, several fusion rules are applied. We use 250 as d value in this example. At sample point 1, all the sensor nodes are eliminated except $s1$, because their distances to the target are greater than d value. The local classification result $c1$ of $s1$ is selected. If we look into sample point 2, we can see that sensor nodes $s1$, $s2$ and $s3$ have lower distances than d value. The label with highest count among these three sensor nodes is $c1$. None of the sensor nodes are eliminated due to d threshold at sample point 3. However,

there are two class labels with highest count, namely $c3$ and $c2$. Since no tie break sensor node exists, the sensor node with minimum distance, $s2$, is selected. The tie break sensor node $s5$ takes scene at sample point 4. Since more than one label has same highest count, $s5$ decides the final result. Unlike at sample point 3, all the sensor nodes exceed the threshold value at sample point 5. But the solution is same, the sensor node with minimum distance, $s3$, is selected with the result $c2$.

3.6 Fuzzy Decision Fusion

In this section, we present our fuzzy decision fusion (FDF) method for single target classification in WSN. In this approach, we use the power of fuzzy logic in the decision fusion phase of the classification process. When all the sensor nodes in the WSN decide their final decision on the target, they transmit these results to a sink node. The sink node realizes the fusion process. The result found after the fusion process is the final result.

We use two fuzzy input variables for our FDF algorithm. The first fuzzy input is the distance between the sensor node and the target in WSN. While the target is moving inside a WSN, the position of the target is recorded. Knowing the deployment positions of the sensor nodes, the distance can be calculated. Based on the data of DARPA SensIT project [10], a trapezoidal and triangular membership function is defined for fuzzy input variable *Distance*. The fuzzy set formed by this membership function and corresponding linguistic states is shown in Figure 3.4.

The crisp *Distance* input values are fuzzified using the membership functions in Figure 3.4. The linguistic variables for *Distance* are *near*, *medium* and far. For *near* and far linguistic variables, trapezoidal membership function is used. However, we prefer triangular membership function for the linguistic variable *medium*.

The second fuzzy input variable is the energy of the acoustic signal received by the sensor nodes. The energy information for sensor nodes while the target is passing through the WSN, is again received from DARPA SensIT project [10]. Like *Distance*, *Energy* is represented by trapezoidal and triangular membership functions and corresponding linguistic variables. Figure 3.5 shows the fuzzy set for *Energy*.

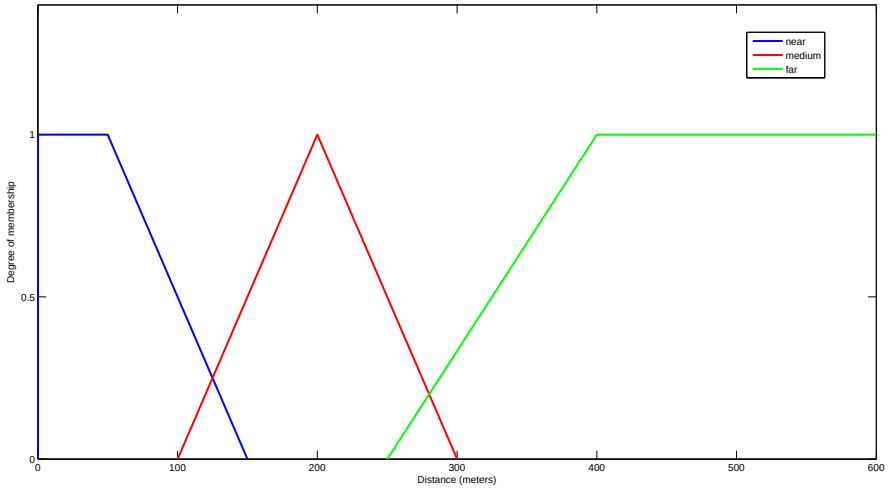
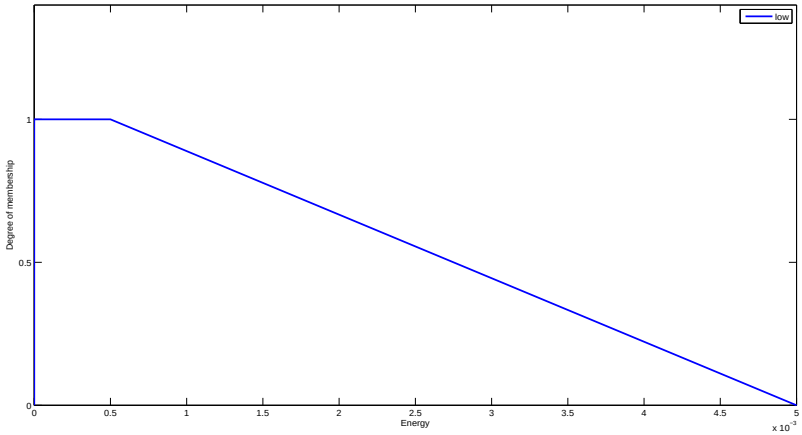


Figure 3.4: Fuzzy set for *Distance* variable

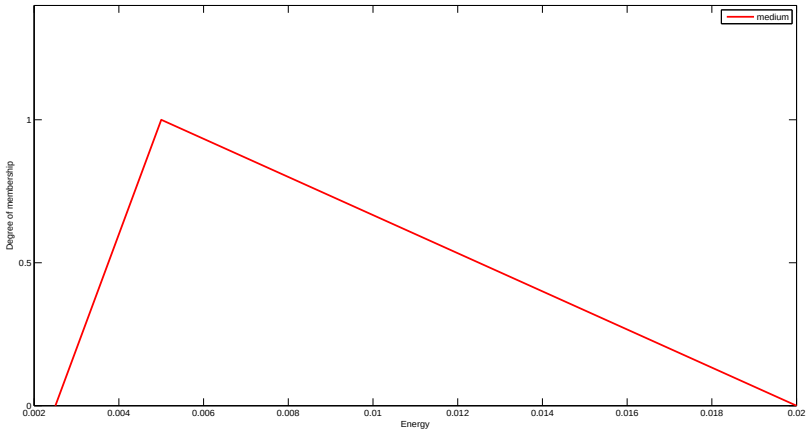
Three linguistic variables for *Energy* are *low*, *medium* and *high*. Since the energy values differ enormously from *low* linguistic variable to *high* linguistic variable, we do not prefer to show all the membership functions on a same graph. Instead of that, a separated graph is used for each membership function of the linguistic variables of *Energy*. Figure 3.5 (i) depicts the membership function of *low* linguistic variable while *medium* linguistic variable takes place in Figure 3.5 (ii). *low* linguistic variable has a trapezoidal membership function while *medium* linguistic variable has a triangular membership function. The last linguistic variable, *high*, is shown in Figure 3.5 (iii). Like *low* linguistic variable, *high* linguistic variable has a trapezoidal membership function. Actually *high* energy values can pass the value 1. For the clarity of the graph, we show *high* energy values up to 1. In our algorithms, the energy values greater than 1 are still considered belonging to *high* linguistic variable with a degree 1.

The fuzzy variable *Chance* represents the fuzzy output. Similar *Chance* output variable is used in [16]. *Chance* determines the election chance of the sensor nodes being used in decision fusion for target classification. *Chance* has nine linguistic variables and therefore nine membership functions. Fuzzy set for *Chance* is in Figure 3.6.

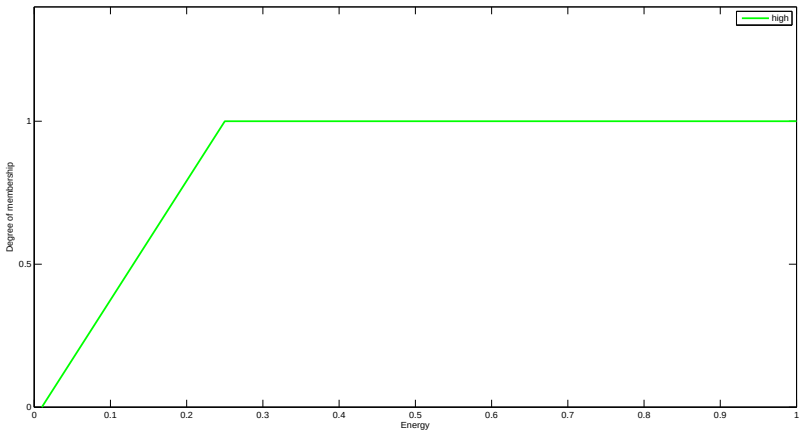
In Figure 3.6, nine linguistic variables are demonstrated. These are $LF(LowFar)$, $LM(LowMedium)$, $LN(LowNear)$, $MF(MediumFar)$, $MM(MediumMedium)$, $MN(MediumNear)$, $HF(HighFar)$, $HM(HighMedium)$ and $HN(HighNear)$. Only LF and HN membership functions are trape-



(i)



(ii)



(iii)

Figure 3.5: Fuzzy set for *Energy* variable

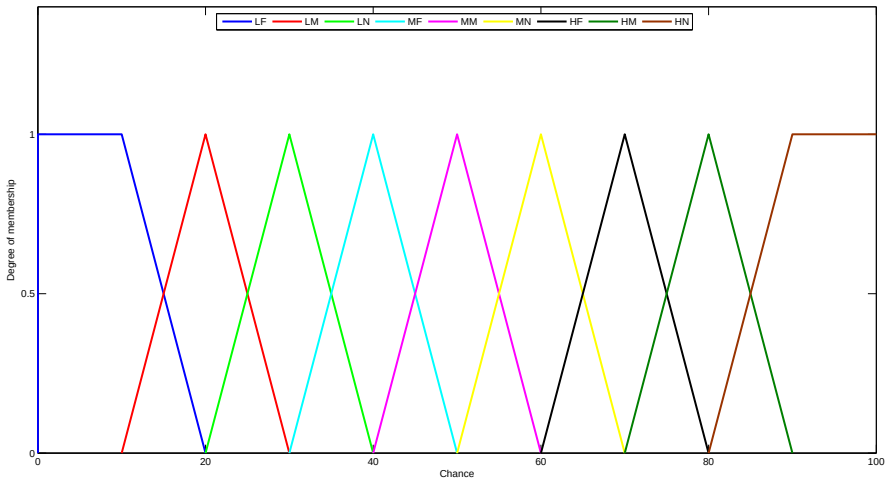


Figure 3.6: Fuzzy set for *Chance* variable

zoidal. All the other linguistic variables have triangular membership functions.

These three fuzzy sets are basically formed based on the data provided by DARPA SensIT project [10]. Many other fuzzy sets can be constructed by evaluating the data in the process.

As mentioned above, the fuzzified inputs are applied to the fuzzy rules. For our fuzzy decision algorithm, we have defined nine fuzzy rules. These nine fuzzy rules can be seen in Table 3.4 similar to the fuzzy rules in [16]. These fuzzy rules make use of AND operator as the conjunction of the antecedents. For example, Rule 1 states that "If *Distance* is far AND *Energy* is *low* then *Chance* is *LF*". One may think that a sensor node closest to the target should have the highest signal energy. Actually, this situation can be true at most of the cases. However, in real world environment, even if the sensor node is close to target, it may not get the highest signal energy due to some environmental factors and blocking. Random deployment may force sensor nodes to settle in holes or instable areas. On the other hand, some sensor nodes may have high signal energy values because of noise, wind and so on. These sensor nodes are eliminated if they are not close to the target. Moreover, our approach provides a clear distinction between sensor nodes having similar locations or signal energies. When one input variable is similar among the sensor nodes, the other input variable makes the difference.

In our approach, *Distance* and *Energy* fuzzy input variables do not have same weights on

Table 3.4: Fuzzy rules for FDF algorithm

Rule No	*Distance*	*Energy*	*Chance*
1	*far*	*low*	*LF*
2	*medium*	*low*	*LM*
3	*near*	*low*	*LN*
4	*far*	*medium*	*MF*
5	*medium*	*medium*	*MM*
6	*near*	*medium*	*MN*
7	*far*	*high*	*HF*
8	*medium*	*high*	*HM*
9	*near*	*high*	*HN*

the fuzzy output variable *Chance*. This situation can be observed by considering the rules in Table 3.4. Depending on the data we used, we prefer to give more weight on *Energy* fuzzy variable than *Distance* fuzzy variable. Therefore, in this approach, sensor nodes having more energy have more chance of being elected than the sensor nodes having close distances to the target.

We use Mamdani fuzzy inference system in our approach. We choose Mamdani inference system over Sugeno and Tsukamoto inference systems because it has some beneficial situations. The main reason we prefer Mamdani inference system is that it is intuitive and easy to implement. Moreover, it has widespread acceptance. Mamdani inference system is also well suited for human input [20].

Suppose two crisp input values, *Distance* = 120 and *Energy* = 0.01 are given. Firstly, the intersection points for the variables should be determined on the fuzzy sets of *Distance* and *Energy*. Figure 3.7 and Figure 3.8 depict the fuzzification of the crisp input values for the input variables *Distance* and *Energy* respectively.

In Figure 3.7, *Distance* fuzzy variable has the degrees of membership 0.3 *near*, 0.2 *medium* and 0 *far* for input x = 120. The degrees of membership for *Energy* variable are 0 *low*, 0.67 *medium* and 0 *high* for input x = 0.01. Only *medium* membership function for *Energy* variable is depicted in Figure 3.8 because *low* and *high* membership values are 0.

After crisp input values are fuzzified, the rule evaluation phase starts. In our approach, we have nine fuzzy if then rule listed in Table 3.4. In Table 3.5, nine rule evaluations for *Distance*

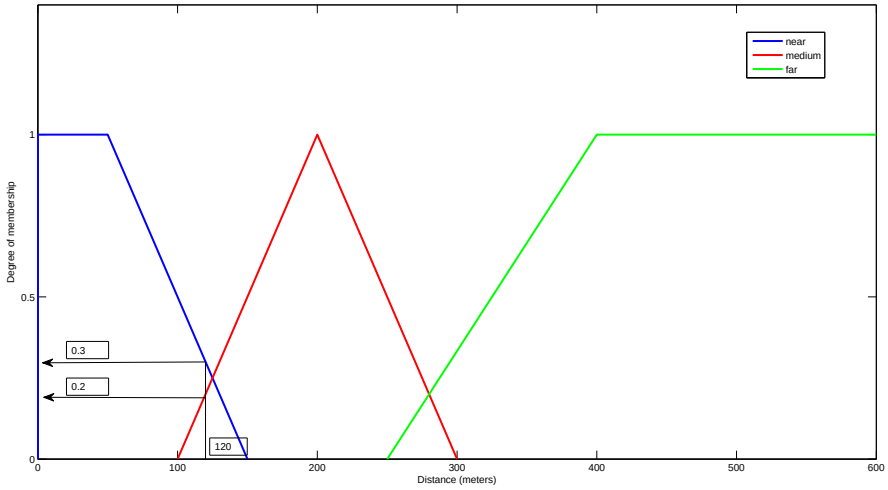


Figure 3.7: Fuzzification of crisp *Distance* input x = 120

and *Energy* input variables are shown.

Table 3.5: Fuzzy rule evaluation for *Distance* and *Energy* input variables

Rule No	*Distance*	*Energy*	*Chance*
1	$far = 0$	$low = 0$	$LF = 0$
2	$medium = 0.2$	$low = 0$	$LM = 0$
3	$near = 0.3$	$low = 0$	$LN = 0$
4	$far = 0$	$medium = 0.67$	$MF = 0$
5	$medium = 0.2$	$medium = 0.67$	$MM = 0.2$
6	$near = 0.3$	$medium = 0.67$	$MN = 0.3$
7	$far = 0$	$high = 0$	$HF = 0$
8	$medium = 0.2$	$high = 0$	$HM = 0$
9	$near = 0.3$	$high = 0$	$HN = 0$

In Table 3.5, nine fuzzy rules are evaluated and corresponding *Chance* values are calculated. Since the operator in the antecedent part is AND for all the fuzzy rules, minimum operator is used. In *Rule No* 5, *Distance* has the value of 0.2 and *Energy* has the value of 0.67. Applying minimum operator, 0.2 is obtained for MM linguistic variable at *Chance*.

After rules are evaluated, the aggregation of the rule outputs should be done. The resulting output area is shown in Figure 3.9. The shaded areas belong to the linguistic variables MM and MN for *Chance* fuzzy set. The other linguistic variables are not shaded in the figure because they have the degree of membership value 0 as the output of rule evaluation.

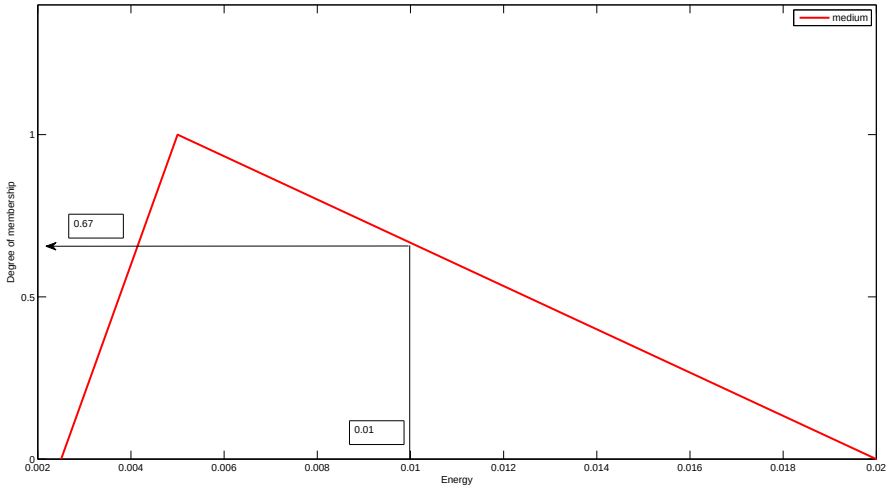


Figure 3.8: Fuzzification of crisp *Energy* input x = 0.01

The centroid of the shaded area in Figure 3.9 is then calculated. This crisp centroid value is the defuzzified chance value for the crisp inputs. In Equation 3.2, centroid of area is calculated as integral over the shaded area for *Chance* fuzzy set in Figure 3.9. The defuzzified chance value for crisp inputs *Distance* = 120 and *Energy* = 0.01 is approximately 56.07.

$$defuzzified\ chance\ value = \frac{\int_z \mu_{Chance}(z) z dz}{\int_z \mu_{Chance}(z) dz} \approx 56.07 \tag{3.2}$$

Our fuzzy decision fusion algorithm can be seen in Algorithm 3.6. For this algorithm, the sensor nodes have to complete their local classifications in the *NodesLC* structure. In *Nodes* structure, sensor nodes are kept. For all of the sensor nodes the distance between the sensor node and target is calculated. Like distance, we calculate the signal energy value for all the sensor nodes. The defuzzified chance value is then calculated based on the distance and signal energy fuzzy input variables as well as fuzzy rules in Table 3.4. The calculated defuzzified chance values for each sensor is kept in *NodesCH*. When the defuzzified chance value is calculated for all sensor nodes, we try to find the class which has the highest total chance by simply adding chance values from sensor nodes for each class separately. These values are stored in *WeightCount*. The class having the highest total chance in *WeightCount*, is the final result. However, if there is a situation where more than one class having the same highest total chance, then the algorithm continues. In this case, we pick the sensor node, *optSensorNode*, having highest defuzzified chance value. The class label of this sensor node is determined to

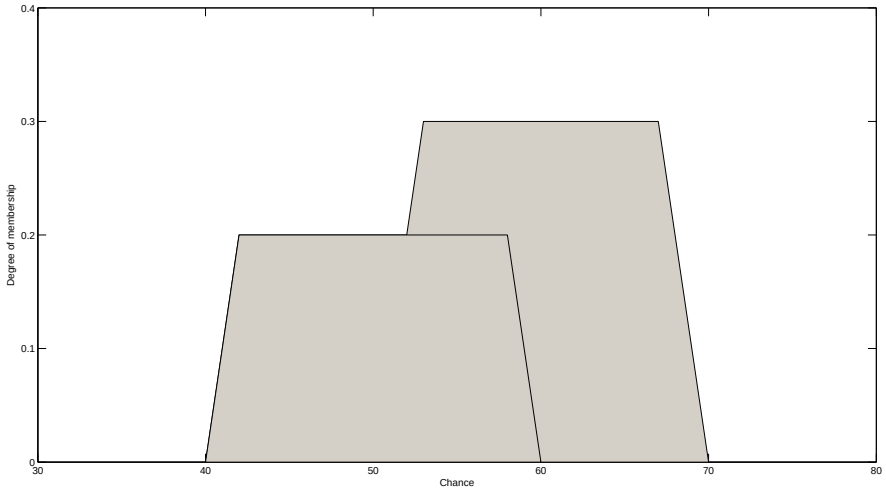


Figure 3.9: Output area after rule evaluation

be final classification result for the classification process. If the highest defuzzified chance value is owned by more than one different sensor nodes, the first sensor node evaluated is picked since no further separation can be made at this level.

3.6.1 Running example for FDF

In this section, a running example about FDF algorithm is told. Five sensor nodes namely $s1$, $s2$, $s3$, $s4$ and $s5$ and also three class labels $c1$, $c2$ and $c3$ are used as at Section 3.3.1. The classification results for FDF algorithm through five sample points are shown in Table 3.6.

Table 3.6: Sample classification flow for FDF algorithm

SP	$s1 - Chc$	$s2 - Chc$	$s3 - Chc$	$s4 - Chc$	$s5 - Chc$	$Result$
1	$c1 - 80$	$c1 - 80$	$c1 - 50$	$c2 - 60$	$c3 - 75$	$c1$
2	$c1 - 75$	$c1 - 60$	$c2 - 70$	$c2 - 90$	$c2 - 90$	$c2$
3	$c3 - 75$	$c3 - 80$	$c2 - 80$	$c1 - 75$	$c2 - 50$	$c3$
4	$c2 - 90$	$c1 - 80$	$c2 - 50$	$c1 - 75$	$c2 - 60$	$c2$
5	$c1 - 50$	$c1 - 90$	$c2 - 80$	$c2 - 60$	$c3 - 60$	$c1$

Like in Table 3.2, rows and columns show same values in Table 3.6. However, distance values coming with sensor nodes are replaced with defuzzified chance values. At sample points from 1 to 4, the defuzzified chance values are simply added by each class. The class label with

```
1: Nodes ← Sensor nodes array
2: NodesLC ← Sensor nodes - local classification result matrix
3: NodesCH ← Sensor nodes - chance matrix
4: WeightCount ← Class label - total chance matrix
5: for all Sensor Node S in Nodes do
6:     distance ← Distance between the target and sensor node S
7:     energy ← Signal energy for sensor node S
8:     chance ← Defuzzified chance value for distance and energy
9:     NodesCH[S] ← chance
10: end for
11: for all Sensor Node - Chance Pair SC in sorted NodesCH do
12:     Node ← Sensor node of SC
13:     Class ← Classification result from NodesLC for sensor Node
14:     Increment WeightCount[Class] with chance of SC
15: end for
16: if WeightCount has more than one class having same highest weight then
17:     optSensorNode ← Sensor node having the highest chance
18:     Find the classification result from NodesLC for sensor optSensorNode
19: else
20:     Find the class with highest weight in WeightCount
21: end if
```

Algorithm 3.6: Fuzzy Decision Fusion

highest total defuzzified chance value is selected. However, at sample point 5, $c1$ and $c2$ have same total defuzzified chance value 140. In this case, the sensor node with highest defuzzified chance value, $s2$, is chosen with the final result $c1$.

3.7 Fuzzy Decision Fusion with Threshold

This algorithm is highly similar to the FDF algorithm mentioned at Section 3.6. The fuzzy input or output variables and fuzzy rules for the FDF algorithm can also be applied to the fuzzy decision fusion with threshold (FDFWT) algorithm. The FDF algorithm uses all the sensor nodes in the environment to produce a classification result. All the sensor nodes contribute the final result with their defuzzified chance values. However, in some situations, some sensor nodes with low defuzzified chance values should be removed from the classification process. These sensor nodes may mislead the whole classification process.

In FDFWT algorithm, we define a threshold chance value for sensor nodes. The sensor nodes having defuzzified chance value lower than the threshold value are eliminated. Only the sensor nodes having the defuzzified chance value above the threshold are considered in fusion operation. The magnitude of the chance values from remaining sensor nodes are considered. The algorithm is shown in Algorithm 3.7.

In Algorithm 3.7, sensor nodes are kept in *Nodes* and local classification result for each sensor node is kept in *NodesLC*. FDFWT algorithm starts like FDF algorithm. When the defuzzified chance values are calculated for all sensor nodes in *NodesCH*, reduction of the sensor node begins using threshold value. The sensor nodes having lower defuzzified chance value than the T value are eliminated. *WeightCount* representing the total defuzzified chance values for each class is filled with the data from the remaining nodes. The class label with the highest total defuzzified chance value in *WeightCount* is selected as the final classification result. Unfortunately, some exceptional situations may occur in the algorithm. If no sensor node is higher than or equal to threshold value, that is *WeightCount* has 0 value for all classes, then we look into *optSensorNode*. Same procedure is applied when no *TBSensor* is found when *WeightCount* has more than one class having same highest count. *optSensorNode* is the node of which local classification result has the highest defuzzified chance value among all sensor nodes. Since the local classification result for *optSensorNode* is known beforehand, we

```
1: Nodes ← Sensor nodes array
2: NodesLC ← Sensor nodes - local classification result matrix
3: NodesCH ← Sensor nodes - chance matrix
4: WeightCount ← Class label - total chance matrix
5: T ← Threshold value
6: TBSensor ← Tie break sensor node number
7: for all Sensor Node S in Nodes do
8:     distance ← Distance between the target and sensor node S
9:     energy ← Signal energy for sensor node S
10:    chance ← Defuzzified chance value for distance and energy
11:    NodesCH[S] ← chance
12: end for
13: Sort NodesCH according to chance values
14: for all Sensor Node - Chance Pair SC in sorted NodesCH do
15:    if Chance of SC greater than or equal to T then
16:        Node ← Sensor node of SC
17:        Class ← Classification result from NodesLC for sensor Node
18:        Increment WeightCount[Class] with Chance of SC
19:    else
20:        TBSensor ← Sensor node of SC
21:        Break for loop
22:    end if
23: end for
24: if WeightCount has 0 value for all classes or no TBSensor is found when WeightCount
    has more than one class having same highest count then
25:    optSensorNode ← Sensor node having the highest chance
26:    Find the classification result from NodesLC for sensor optSensorNode
27: else if WeightCount has more than one class having same highest count then
28:    Find the classification result from NodesLC for sensor TBSensor
29: else
30:    Find the class with highest count in WeightCount
31: end if
```

Algorithm 3.7: Fuzzy Decision Fusion with Threshold

reach to the final classification result using *optSensorNode*. Moreover, if more than one class has the same highest defuzzified chance values among remaining sensor nodes, the problem still exists. In this case, the tie break sensor node, *TBSensor*, says the last word. The tie break sensor node is the sensor node having the highest defuzzified chance value below the threshold. No further action is done if there is more than one class having the same highest defuzzified chance value below the threshold in the last situation. When this problem occurs, the first tie break sensor node evaluated is chosen.

3.7.1 Running example for FDFWT

For FDFWT algorithm, we present a running example in this part. We use five sensor nodes namely $s1$, $s2$, $s3$, $s4$ and $s5$ and also three class labels $c1$, $c2$ and $c3$ as at Section 3.3.1. In Table 3.7, the classification results for FDFWT algorithm through five sample points are demonstrated.

Table 3.7: Sample classification flow for FDFWT algorithm

SP	$s1-Chc$	$s2-Chc$	$s3-Chc$	$s4-Chc$	$s5-Chc$	*Result*
1	$c1-20$	$c1-40$	$c1-25$	$c2-60$	$c3-50$	$c2$
2	$c1-75$	$c1-60$	$c2-90$	$c2-30$	$c2-45$	$c1$
3	$c3-90$	$c3-80$	$c2-85$	$c1-75$	$c2-85$	$c3$
4	$c2-60$	$c1-90$	$c2-85$	$c1-55$	$c2-30$	$c2$
5	$c1-20$	$c1-30$	$c2-25$	$c2-25$	$c3-40$	$c3$

In Table 3.7, rows and columns show same values in Table 3.2 with defuzzified chance value attached to local classification results of the sensor nodes. The samples in this example shows similarities with the DMDF running example. In this example, we use threshold value T as 50. At sample point 1, only sensor node $s4$ exceeds the threshold value. $c2$ is the final classification result. Sensor nodes $s1$, $s2$ and $s3$ have defuzzified chance values greater than threshold value at sample point 2. When the values are added according to class labels, $c1$ is selected. $c1$ has the total value 135 while $c2$ has 90. None of the sensor nodes are eliminated because of the threshold value at sample point 3. Since no tie break sensor node exists, the sensor node with the highest defuzzified chance value, $s1$, says the last word. The equality arised from the sum of the defuzzified chance values can be broken by tie break sensor node, $s5$, at sample point 4. Contrary to sample point 3, no sensor node is able to exceed the

threshold value at sample point 5. The sensor node with highest defuzzified chance value is selected again, which is $s5$ with result $c3$.

CHAPTER 4

EVALUATION

In this chapter, we will evaluate the performance of our fuzzy decision fusion algorithms with three accepted decision fusion algorithms; majority voting decision fusion, nearest neighbor decision fusion and d_{max} decision fusion algorithms. Before the analysis of the results, the testing environment for evaluation is described. Three different scenarios are presented according to sensor node deployments.

4.1 Testing Environment

In this work, we use the subset of the data provided by DARPA SensIT project [10] which is called SITEX02. 75 sensor nodes are deployed on a desert area for the experiment. The sensor nodes deployed along an east - west road, a south - north road and an intersection area. The target vehicles follow east to west road, west to north road and north to east road. The sensor nodes have acoustic (microphone), seismic (geophone) and infrared (polarized IR sensor) modalities.

The data set provided is composed of runs. A run is the drive of a vehicle through the testing field. We try to classify two kinds of military vehicles: AAV and DW. We choose only the east - west road for testing path for simplicity. We pick 17 sensors which collect data from the targets using this path. The runs through the east - west road are named with the name of the vehicle and times of three. For example, AAV3 run represents the first run of the AAV vehicle through the east - west road while AAV6 represents the second run of the AAV vehicle through same path. We use three AAV runs named AAV3, AAV6, AAV9 and four DW runs DW3, DW6, DW9 and DW12.

In our tests, we use acoustic features. Advantages of the acoustic sensors can be counted as having long sensing range and high-fidelity, no line-of-sight requirement, being co-operable and having passive nature. Moreover, acoustic sensors provide reasonable signal processing [3]. The data set includes the acoustic feature vectors for each sensor node and for each run. The extracted features are based on the frequency spectrum of acoustic signal. FFT of the signals is calculated for every 512 points yielding 512 FFT points. First 100 point is chosen. These points are averaged by pairs resulting 50-dimensional FFT-based feature vectors. We reduce this number using our feature reduction technique. We use [7x1] feature vector after reduction. Our classifier choice is kNN classifier due to its efficiency and simplicity. The k value in the tests for kNN classifier is 7.

The target positions for every 0.75 seconds for each run and positions of sensor nodes are provided. However, the target positions are in Universal Transverse Mercator (UTM) coordinates. The UTM distance between target and sensor nodes are calculated using Simpson's Rule provided in [22]. Moreover, DARPA data set has energy files for each run. Energy values are calculated in 0.75 seconds intervals like target positions. These values are determined by Constant False Alarm Rate (CFAR) detection algorithm. No information is provided about the time of feature extraction on sensor nodes. In other words, for a specific moment, the position of the target and the signal energy received by the sensor nodes are known. But, for that specific moment, it cannot be derived which feature vector set is extracted from a sensor node. Therefore, we assume all the used sensors record the entire motion of the target. When the feature vector set has more time points from target time points, we simply map feature vector time points to target time points.

Our metric for evaluating decision fusion algorithms is classification rate to measure classification accuracy. To compute the classification rate, we use Equation 4.1 [6]. R_c denotes the classification rate percentage while $n_{correct}$ denotes number of samples classified correctly and n_{total} denotes the number of total samples. To test the classification rate for a run, we use the data of all six other runs as training set. We use our node-based training set formation when obtaining data from training sets.

$$R_c = \frac{n_{correct}.100}{n_{total}} \tag{4.1}$$

Average of the classification rate results for seven runs determine the final percentage rate

for a decision fusion algorithm. Our comparison is based on this averaged classification rate value.

We compare fuzzy decision fusion algorithms with MVDF, NNDF and DMDF. In DMDF, we use the value 250 as the threshold distance value d.

Three different sensor node deployment scenarios are presented in the following sections. In the first scenario, sensor nodes are scattered through the target path while in the second scenario, sensor nodes are gathered. Last scenario contains all the sensors picked for east - west road.

4.2 Scenario 1: Scattered Sensor Node Deployment

In this scenario, we test decision fusion algorithms for scattered sensor node deployment. In Figure 4.1, a sample path for the AAV vehicle and positions of the sensors are illustrated. 9 sensor nodes are used for this scenario.

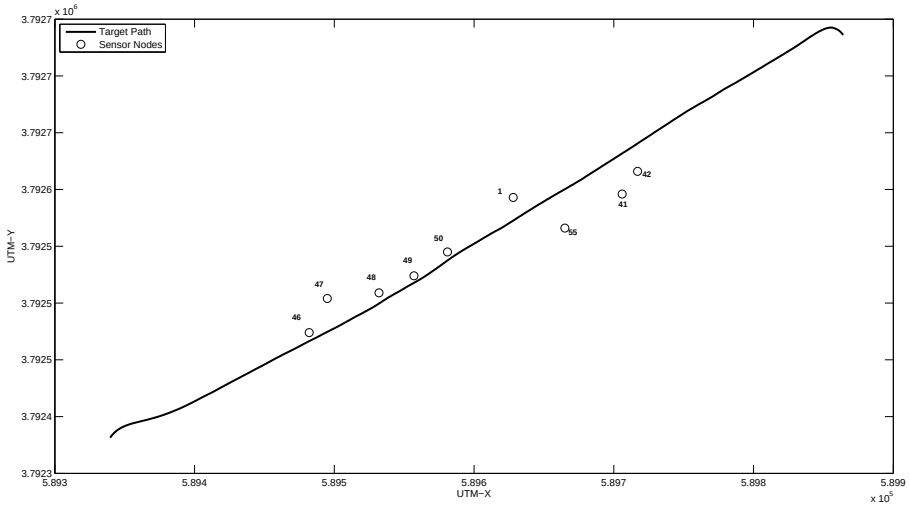


Figure 4.1: Scattered sensor nodes layout and sample AAV target path

First of all we try to find best threshold value for FDFWT algorithm. The classification rate values of seven runs for threshold value from 20 to 70 are calculated. Table 4.1 lists the produced results. Bold written classification rates show the maximum rate for a specific run.

Table 4.1: Classification rate of FDFWT using different threshold(T) values for Scenario 1

Run Name	$T = 20$	$T = 30$	$T = 40$	$T = 50$	$T = 60$	$T = 70$
AAV3(%)	13.44	21.31	22.62	24.59	27.21	**33.44**
AAV6(%)	**80.50**	73.58	74.21	72.33	74.84	74.21
AAV9(%)	50.63	59.49	58.23	59.49	60.76	**62.03**
DW3(%)	**90.91**	78.89	75.37	74.19	74.19	73.61
DW6(%)	**98.86**	94.25	93.10	90.80	89.08	88.51
DW9(%)	50.63	48.10	46.84	46.84	48.10	**55.70**
DW12(%)	**73.10**	66.20	64.14	64.83	63.44	60.00
AVG(%)	**65.44**	63.12	62.07	61.87	62.52	63.93

As seen in Table 4.1, FDFWT algorithm reaches the maximum classification rate when the threshold value is 20. We compare the FDFWT algorithm with other decision fusion algorithms with this threshold value.

After we find the best threshold value for FDFWT algorithm, we can compare five decision fusion algorithms for scattered sensor deployment. Table 4.2 shows the classification rates of MVDF, NNDF, DMDF, FDF and FDFWT algorithms. Highest rates for a run are again written in bold font.

Table 4.2: Classification rate of decision fusion algorithms for Scenario 1

Run Name	*MVDF*	*NNDF*	$DMDF(d = 250)$	*FDF*	$FDFWT(T = 20)$
AAV3(%)	2.29	13.11	4.26	8.85	**13.44**
AAV6(%)	79.25	52.83	76.73	79.24	**80.50**
AAV9(%)	29.11	20.25	29.11	49.36	**50.63**
DW3(%)	**99.12**	78.01	97.36	92.67	90.91
DW6(%)	**100.00**	96.55	**100.00**	**100.00**	98.86
DW9(%)	32.91	36.71	40.51	46.84	**50.63**
DW12(%)	86.90	60.00	**91.72**	76.55	73.10
AVG(%)	61.37	51.07	62.81	64.79	**65.44**

According to Table 4.2, FDFWT has the best classification rate among decision fusion algorithms for scattered sensor node deployment. FDF algorithm comes second after FDFWT algorithm. NNDF algorithm takes the last place.

4.3 Scenario 2: Gathered Sensor Node Deployment

Scenario 2 has 9 sensor nodes as in Scenario 1. However, sensor nodes are gathered in this time. Sensor nodes layout and a sample AAV vehicle target path are given in Figure 4.2 for gathered sensor node deployment.

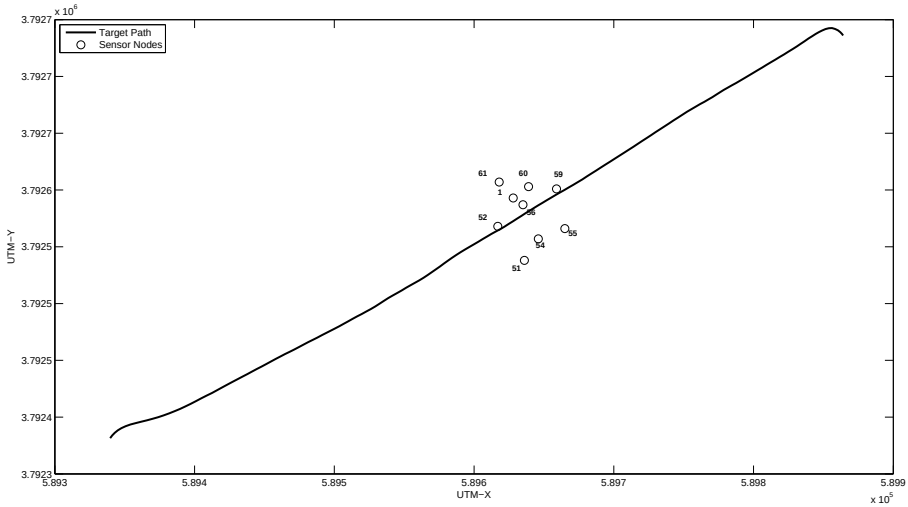


Figure 4.2: Gathered sensor nodes layout and sample AAV target path

We find the threshold value for FDFWT algorithm which provides the best classification rate for gathered sensor deployment. In Table 4.3, FFTWT algorithm classification rates are listed. Threshold value is ranging from 20 to 70.

Table 4.3: Classification rate of FDFWT using different threshold(T) values for Scenario 2

Run Name	$T = 20$	$T = 30$	$T = 40$	$T = 50$	$T = 60$	$T = 70$
AAV3(%)	14.60	22.87	24.52	28.37	28.66	**33.61**
AAV6(%)	**85.53**	78.95	78.95	80.26	82.24	82.90
AAV9(%)	**98.94**	**98.94**	**98.94**	**98.94**	**98.94**	**98.94**
DW3(%)	**94.72**	88.27	86.22	86.51	83.87	82.99
DW6(%)	**100.00**	99.43	98.85	98.85	98.85	98.85
DW9(%)	8.51	27.66	31.91	35.11	35.11	**42.55**
DW12(%)	42.31	43.85	46.15	46.15	46.15	**46.92**
AVG(%)	63.52	65.71	66.51	67.74	67.69	**69.54**

The best classification rate is reached with the threshold value 70 for FDFWT in gathered sensor deployment. This threshold value is used for comparing FDFWT with other decision

fusion algorithms.

For Scenario 2, Table 4.4 illustrates comparison of the five decision fusion algorithm. Threshold is 70 for FDFWT and distance is 250 for DMDF algorithms. Bold values are the highest values provided by an algorithm within a run.

Table 4.4: Classification rate of decision fusion algorithms for Scenario 2

Run Name	*MVDF*	*NNDF*	*DMDF*($d = 250$)	*FDF*	*FDFWT*($T = 70$)
AAV3(%)	2.48	31.40	5.51	9.92	**33.61**
AAV6(%)	90.13	67.76	81.58	**90.79**	82.90
AAV9(%)	96.80	82.98	94.68	**98.94**	**98.94**
DW3(%)	**100.00**	98.53	99.71	99.12	82.99
DW6(%)	**100.00**	**100.00**	**100.00**	**100.00**	98.85
DW9(%)	2.13	6.38	2.13	5.32	**42.55**
DW12(%)	36.92	23.85	36.92	41.54	**46.92**
AVG(%)	61.21	58.70	60.08	63.66	**69.54**

Like in Scenario 1, in Scenario 2 FDFWT algorithm has the best classification rate. FDF has the second place. However, the classification rate difference is bigger this time between fuzzy approaches and the others. NNDF algorithm has the worst percentage.

4.4 Scenario 3: All Sensor Node Deployment

Last scenario includes all 17 sensors for testing. Layout of this 17 sensors with sample AAV target path is depicted in Figure 4.3.

As in the previous two scenarios, we start with choosing the best threshold value for FDFWT algorithm. Table 4.5 lists the classification rates for FDFWT using all sensor nodes. Threshold values are again in the interval of 20 and 70.

Based on the results in Table 4.5, threshold value chosen is 30. FDFWT is compared with other decision fusion algorithms with this value as it happens in the previous scenarios.

Table 4.6 shows the classification rates for five decision fusion algorithms in all sensor node deployment. Threshold for FDFWT is 30 and distance is again 250 for DMDF.

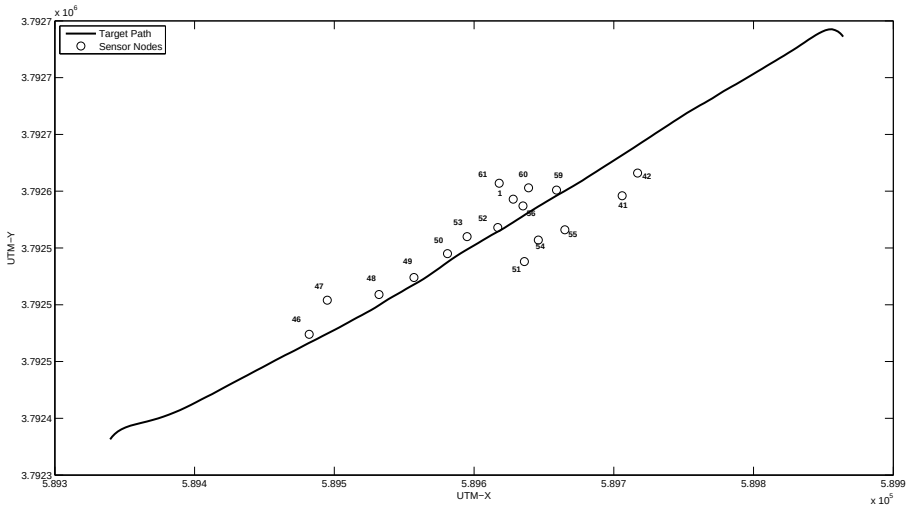


Figure 4.3: All sensor nodes layout and sample AAV target path

Table 4.5: Classification rate of FDFWT using different threshold(T) values for Scenario 3

Run Name	$T = 20$	$T = 30$	$T = 40$	$T = 50$	$T = 60$	$T = 70$
AAV3(%)	6.89	16.53	17.63	22.59	26.45	**30.03**
AAV6(%)	**84.28**	78.62	78.62	78.62	77.99	75.47
AAV9(%)	89.36	**92.55**	90.43	82.98	76.60	76.60
DW3(%)	**98.24**	90.32	86.22	84.75	82.11	79.77
DW6(%)	**98.28**	91.38	88.51	86.78	85.63	82.19
DW9(%)	7.45	24.47	26.60	26.60	29.79	**37.23**
DW12(%)	**79.31**	75.17	72.41	69.66	68.97	66.21
AVG(%)	66.26	**67,01**	65.77	64.57	63.93	63.93

In this case FDFWT algorithm has the best classification rate. FDF is slightly worse than FDFWT. MVDF takes the third place in scenario with deployment of all the sensors. NNDF is again in the last place.

4.5 Overall Evaluation

Based on the data presented in the previous three scenarios, Figure 4.4 can be shown.

In Figure 4.4, averaged overall evaluations are depicted. According to results, FDFWT has the best classification results among all the decision fusion algorithms. FDF is second and MVDF is the third. Like in all the scenarios, NNDF is in the last place.

Table 4.6: Classification rate of decision fusion algorithms for Scenario 3

Run Name	*MVDF*	*NNDF*	*DMDF*($d = 250$)	*FDF*	*FDFWT*($T = 30$)
AAV3(%)	2.20	**17.91**	2.48	4.96	16.53
AAV6(%)	**92.45**	52.83	85.83	89.94	78.62
AAV9(%)	74.47	22.34	63.83	90.43	**92.55**
DW3(%)	**100.00**	87.39	99.41	**100.00**	90.32
DW6(%)	**100.00**	93.10	96.55	**100.00**	91.38
DW9(%)	4.26	**27.66**	4.26	3.19	24.47
DW12(%)	83.45	66.90	**87.59**	80.00	75.17
AVG(%)	65.26	52.59	62.81	66.93	**67.01**

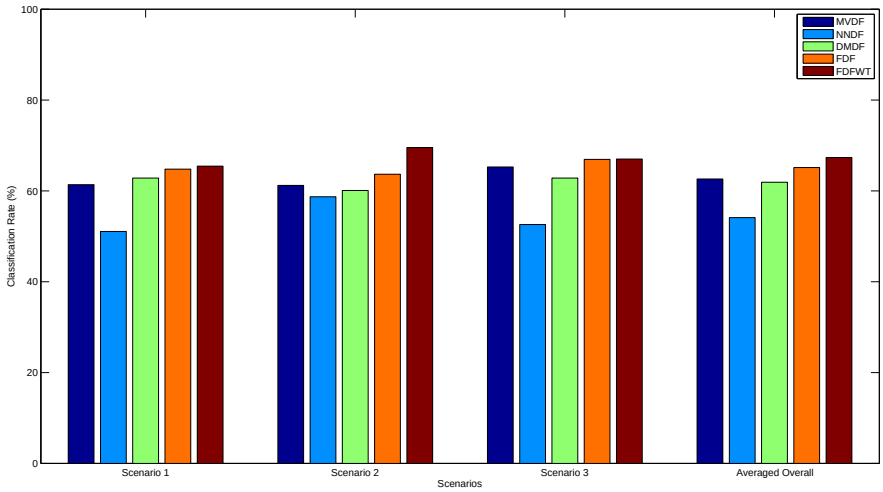


Figure 4.4: Overall evaluation of decision fusion algorithms

The fuzzy approaches perform the best approach when the sensor nodes are gathered. When the distance factor is almost same for all sensor nodes in this scenario, the distance based decision fusion algorithms experience more failure. Since sensors are close to each other, they all may have a wrong classification result together. In scattered deployment, sensor nodes can have different classification results and a wrong result can be eliminated due to sensor nodes being in very different locations.

According to [6], AAV vehicle moves at low speed in the run AAV3 and DW vehicle has a relatively high speed in the run DW9. This explains why the classification rates are seriously wrong in these runs. However, fuzzy approaches performed much better than the other decision fusion algorithms for this runs. Signal energy variable makes this difference. The distance based decision fusion algorithms are good at vehicles at normal speed. Since the

fuzzy approaches both include the distance and signal energy variables, they perform better overall.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, two fuzzy decision fusion methods for target classification in WSN have been developed. We applied fuzzy logic in order to improve the accuracy of the classification process. Additionally, feature reduction and training set formation techniques are provided as a part of this work.

Our feature reduction technique is based on the difference values of the class labels. The dominant feature vectors having the highest difference values among classes are determined and these feature vectors are used for target classification. By reducing features, the time spent for local classification on a sensor node is diminished which in turn considerably increases the efficiency of the algorithm.

Our training set formation approach again focuses on reducing the time spent on intra-node computations for local classifications. We propose node-based training set formation. To classify an input feature vector for a specific node, only the training set feature vectors for that specific sensor node are used. The other feature vectors are ignored for this sensor node as they do not have observable contribution.

The fuzzy decision fusion algorithm is our first approach on decision fusion. We use fuzzy input variables *Energy* and *Distance*. The only fuzzy output variable is *Chance*. Based on the distance and energy values of the sensor nodes and fuzzy rules, a defuzzified chance value is calculated for each sensor node. Having higher energy and nearer distance leads a sensor node to have a higher defuzzified chance value. Defuzzified chance values are added class-

wise and the class having the highest total defuzzified chance value is selected as the final result.

Fuzzy decision fusion with threshold algorithm has the same preliminaries with fuzzy decision fusion. The defuzzified chance value is also calculated for each sensor node in this algorithm. However, the sensor nodes having lower defuzzified chance value than the threshold value are eliminated. After elimination of the sensor nodes, algorithm behaves like fuzzy decision fusion algorithm.

We evaluate our two decision fusion algorithms with three popular decision fusion algorithms: Majority voting, nearest neighbor and d_{max} algorithms. We use real data set for classification process. Both of the fuzzy based algorithms proposed in this thesis perform better classification accuracy compared to these three known decision fusion algorithms in these evaluations.

To conclude, fuzzy decision algorithms have a high potential to improve classification accuracy for target classification in WSN.

5.2 Future Work

In this thesis, we evaluate decision fusion algorithms according to their classification accuracies. Some further research may be done for evaluating other parameters. For instance, the sensor energy level can be considered in classification process. The decision algorithm consuming least energy can be found. The fusion algorithms may be changed or improved to consume less energy. In the same manner, the time spent to execute decision fusion algorithms may be compared. The compatibility of the decision fusion algorithms may change based on these evaluation criteria. The ultimate decision fusion algorithm which yields best results for each parameter can be designed.

Feature extraction can also be studied alone. The affect of the feature extraction technique on classification accuracy may be examined. The observation is made between the classification accuracy based on random features and the classification accuracy based on robust and healthy features.

Different modalities may have different results. The optimum modality for specific target classification problem can be found. Data sensed for all the available modalities can be compared

for the best classification result.

Classifiers also have some effect on classification accuracy. Several classifiers can be tested for fusion algorithms. The behavior of the fusion algorithms with local classification results formed from different classifiers may be observed. The best classifier- fusion algorithm pair may be discovered.

Since the classification algorithm involves many parameters during process, the optimization of these parameters seems to be endless. A positive change in a parameter can have negative result on other parameter. Nevertheless, a stable model with optimum parameters (robust feature vectors, suitable modalities, the most appropriate classifier and decent training set, best fusion algorithms) can be developed for further study.

REFERENCES

[1] J. Abonyi, R. Babuska, and F. Szeifert. Modified gath-geva fuzzy clustering for identification of takagi-sugeno fuzzy models. *IEEE Systems, Man and Cybernetics, Part B*, pages 612–621, 2002.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Comput. Netw.*, 38(4):393–422, 2002.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks Journal*, October 2004.

[4] Archana Bharathidasan and Vijay Anand Sai Ponduru. Sensor networks: An overview. *UC Davis Technical Report*, 2001.

[5] R. Brooks, P. Ramanathan, and A. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, November 2003.

[6] L. Chun-Ting, H. Hong, F. Tao, and S. Xiao. Classification fusion in wireless sensor networks. *Acta Automatica Sinica*, 2006.

[7] 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.

[8] W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Jr. Maling, D. E. Nelson, C. M. Rader, and P. D. Welch. What is the fast fourier transform? *Proc. IEEE*, 55:1664–1674, October 1967.

[9] Oscar Cordon, Francisco Herrera, Frank Hoffmann, and Luis Magdalena. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases (Advances in Fuzzy Systems - Applications & Theory)*. World Scientific Publishing Company (February 15, 2002).

[10] M. F. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, July 2004.

[11] Scott R. Eliason. *Maximum Likelihood Estimation: Logic and Practice*. Sage Publications, Inc; illustrated edition (29 Sep 1993).

[12] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. *ACM MobiCom'99*, pages 263–270, 1999.

[13] I. Gupta, D. Riordan, and S. Sampalli. Cluster-head election using fuzzy logic for wireless sensor networks. *Proceedings of the 3rd Annual Communication Networks and Services Research Conference(CNSR'05)*, pages 255–260, 2005.

[14] P.S. Hari, P.R. Kumar, D.T. Rao, and K.R. Rajeswari. Fuzzy modelling for discrimination and merit factor of radar signals for range resolution. *Proceedings of the 7th WSEAS International Conference on Multimedia Systems & Signal Processing (MUSP'07)*, pages 146–151, 2007.

[15] J.S.R. Jang and C.T. Sun. Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, 83:378–406, 1995.

[16] J. Kim, S. Park, Y. Han, and T. Chung. Chef: Cluster head election mechanism using fuzzy logic in wireless sensor networks. *Advanced Communication Technology ICACT 2008*, pages 654–659, February 2008.

[17] D. Li, K. Wong, Y. Hu, and A. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19, March 2002.

[18] R. Lin, Z. Wang, and Y. Sun. Wireless sensor networks solutions for real time monitoring of nuclear power plant. *World Congress on Intelligent Control and Automation*, June 2004.

[19] J.M. Mendel and G.C. Mouzouris. Designing fuzzy logic systems. *IEEE Trans. Circuit and Systems-11: Analog and Digital Signal Processing*, 44:885–895, 1997.

[20] S. Mohagheghi, R. G. Harley, and G. K. Vanayagamoorthy. Modified takagi-sugeno fuzzy logic based controllers for a static compensator in a multimachine power system. *Proc. IEEE IAS*, pages 2637–2642, October 2004.

[21] Kiyoshi Morita. *Applied Fourier Transform*. Ios Pr Inc (January 1, 1995).

[22] Morten Nielsen. SharpGIS, GIS from a .NET and JavaScript developer's perspective. http://www.sharpgis.net/post/2007/10/14/Accurate-distance-calculations-in-UTM-projections.aspx, last visited on 6 September 2009.

[23] Haci Mustafa Palancioglu. *Extracting Movement Patterns Using Fuzzy And Neuro-Fuzzy Approaches*. PhD thesis, The University of Maine, May 2003.

[24] R. Polikar. Ensemble based systems in decision making. *IEEE Circuit Syst. Magazine*, 6:21–45, 2006.

[25] V. N. S. Samarasooriya and P. K. Varshney. A fuzzy modeling approach to decision fusion under uncertainty. *Proc. 1996 IEEE Int. Conf. on Multisensor Fusion and Integration for Intell. Syst.*, December 1996.

[26] S. Santoso, E. J. Powers, W. M. Grady, and P. Hofmann. Power quality assessment via wavelet transform analysis. *IEEE Trans. on Power Delivery*, 11(2):924–930, April 1996.

[27] Weilian Su and Theodores C. Bougiouklis. Data fusion algorithms in cluster-based wireless sensor networks using fuzzy logic theory. *Proc. of* 11^{th} *WSEAS Intl. Conf. on Comm.*, pages 291–299, July 2007.

[28] Y. Tian and H. Qi. Target detection and classification using seismic signal processing in unattended ground sensor systems. *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '02)*, 4:4172, May 2002.

[29] Y. Tsukamoto. An approach to fuzzy reasoning method. *Advances in Fuzzy Set Theory and Applications*, pages 137–149, 1979.

[30] M. Tubaishat and S. Madria. Sensor networks: An overview. *IEEE Potentials*, 22(2):20–23, May 2003.

[31] Merijn van Erp, Louis Vuurpijl, and Lambert Schomaker. An overview and comparison of voting methods for pattern recognition. *Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 195–200, August 2002.

[32] X. Wang, H. Qi, and S. S. Iyengar. Collaborative multi-modality target classification in distributed sensor networks. *Proceedings of the Fifth International Conference on Information Fusion*, 2:285–290, July 2002.

[33] H. Ying. Theory and application of a novel fuzzy pid controller using a simplified takagi-sugeno rule scheme. *Information Science*, 123:281–293, 2000.

[34] L. A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.