

DOCTORAL DISSERTATION

Distributively Observed Events in Wireless Sensor Networks

Dissertation zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.) am Fachbereich
Mathematik und Informatik der Freien Universität Berlin

vorgelegt von
Dipl.-Inform. Norman Dziengel

Eingereicht: 18. November 2015
Disputation: 21. März 2016
Gutachter: Prof. Dr.-Ing. Jochen H. Schiller
Prof. Dr. rer. nat. Thiemo Voigt

Acknowledgements

First and foremost I want to thank my advisor Prof. Dr.-Ing. Jochen H. Schiller for his constant support with valuable feedback and words of encouragement whenever needed. His generosity allowed me to research without restrictions while having a fantastic backbone of knowledge, wisdom, and passion in form of his experiences in research and his fantastic and always fully supportive working group *Systems and Telematics* behind me. I am also thankful for the excellent example he has provided as a successful computer scientist and professor while always following his motivating principle “*acta non verba*”.

My deepest gratitude also goes to Prof. rer. nat. Thiemo Voigt at *SICS* for being available as a reviewer and supervisor for my thesis. His well-known passion for wireless sensor networks has always been a refreshing inspiration for me during the past years.

The hardware used in this dissertation would not have been possible without the discerning questions and realizations of Dr.-Ing. Achim Liers; his support during the years that I spent in our working group is so valuable that I am always smiling when thinking back to that wonderful time.

It is a real treat to give the next thank-you note to Dr. Georg Wittenburg. He is the main reason for my interest in embedded systems; he inspired me from my first day of work within our working group to follow this very satisfying path of research. I will miss the wonderful, very relaxed, and productive time working with him.

During the projects AVS-Extrem and VIVE, which I am grateful for being given the opportunity to oversee, I had inspiring, skillful, and very passionate colleagues who paved the way for my research by supporting the projects with their knowledge, passion, and significant contributions to the system development and experiments. Martin Seiffert’s perfectionism supported both our projects and my research in a fundamental manner. His motivating and kind words always accompanied my work in our projects and supported my research in an overwhelming way. I want to thank him for his constant support, his shared passion about distributed problems in embedded systems, and our more than fertile discussions during our time together in the institute. Marco Ziegert always has interesting hardware projects running on the side; his passion and will to create new things supported our projects and my research optimally. I want to thank him for the opportunity to be a part of his inspiring world of embedded hardware whenever I wanted. Furthermore I want to acknowledge my colleague Stephan Adler for his contributions to the projects and his kind words whenever needed; his way of dealing with problems and challenges during our development was outstanding and more than once gave my research a new perspective and direction. Stefan Pfeiffer’s support during the last years comprises countless in-depth discussions as well as solutions for practical and theoretical questions that leveraged our projects and therefore my research to a higher level of professionalism. I want to acknowledge his unstoppable will to help immediately when his qualifications were needed. Zakaria Kasmi’s reliability and ability to focus on solutions for hard

problems helped finish our projects with an unexpected wealth of features. I want to acknowledge his support and contributions, not to mention his wonderful tea that brought freedom to our all hearts whenever needed. I also thank Jakob Pfender for his outstanding support throughout the whole VIVE project and far beyond with implementations and support during experiments and his more than helpful advice for writing English texts. The hard work of Stefan Bochow pushed our presentations and demonstrations to a new level during the projects; I also very much enjoyed my discussions with him about state-of-the-art technology.

I have appreciated the camaraderie and expertise of my colleagues and I am very thankful for the contribution of the invention of the embedded operating system FireKernel created and supported mainly by Dr. Heiko Will, Dr. Thomas Hillebrandt and Kaspar Schleiser, as it represents the fundamental basis of this thesis.

I very much enjoyed the cooperation and enduring support of researchers Dr. Enrico Köppe, Detlef Hofmann, Viktoriya Tkachenko, and Michael Fischer of the Federal Institute for Materials Research & Testing - Fabeckstraße Berlin (BAM). These wonderful people helped answer questions in the area of structural health monitoring and they made a wooden research bridge available for my project and supported us with room and outstanding administrative flexibility.

I am thankful for the wonderful time and cooperation with our colleagues Prof. Frank Neitzel and Sven Weisbrich from the Institute of Geodesy and Geoinformation Science of the Technische Universität Berlin. They helped significantly with validating the sensor nodes of the Freie Universität Berlin and with understanding fundamental structural health monitoring principles.

The following specialists helped me acquire research funds, generate new ideas based on existing research results, and push the current research into new directions and a higher level. A huge thank you to the high professionalism of administrative and consulting support to the whole team of Profund Innovation, mainly Aneta Bärwolf and Fabian Feldhaus, as well as Dr. Claudia Keil-Dieckmann and Andrea Hübner from the patent and licensing service of our institute. Our external consultant Dr. Christoph Schaffranek did always the best and fastest job, I will never forget his clearly written words. The Charité supported our project ideas with wonderful and inspiring colleagues, namely Johannes Tebbe, and Marcus Luther.

My time at the Freie Universität was made enjoyable in large part due to the many friends and colleagues that gave me advice or inspiration and always had time to give feedback on specific research questions, to proof-read publications, or to come up with completely new insights due to questions that had immeasurable impact on my research direction or methods. Thank you very much Sven Schaust, Dr. Bastian Blywis, Oliver Hahm, Hans Peter Heitzmann, Dr. Matthias Wählich, Prof. Marcel Kyas, Prof. Katinka Wolter, Prof. Mesut Günes, Dr. Kirsten Terfloth, Alexandra Danilkina, Tomasz Naumowicz, Dr. Fabian Stehn, Dr. Marc Scherfenberg, Nico Wanner, Prof. Marco Block-Berlitz, Hauke Petersen, Markus Hoffmann, Dr. Emmanuel Baccelli, Simon Schmitt and Dr. Freddy Lopez Villafuerte.

Stefanie Bahe supported me during the projects to master several steps to start and finish my projects in terms of financial and organizational regulations. Angelika Pasanec helped me to solve multiple administrative barriers to finally complete the dissertation and disputation in elegant and very supporting way. Thank you both for your all time kindness and additional information.

Our precision engineering team of Detlef Müller helped me with passion and joy to design and create multiple versions of the 3D casings that protect our sensor nodes from damage while having a realistic applicability. Special thanks go to Melanie and Florian Klockenberg, Dirk Hund and Alexander Gold.

In the following I want thank all the students that had the confidence to join our projects or to choose me as their supervisor, it was a huge pleasure to work with all of you and I am very happy that I could share this time with you. The countless contributions and lessons you all gave me helped me stay where I am right now; thank you with Christian Wartenburger leading the way, Benjamin Aschenbrenner, Stefan Bochow, Dr. Frederik Hermans, Stefan Kolano, Thomas Krusczyk, Simon Stapelfeld, Nicolai Schmittberger, Fabian Schneider and Ron Wenzel.

I want thank my friends for supporting me with motivation to finish my thesis, Thomas Gröne, Sven Henf, Ramona Karow, Titi Choulamany, Dr. Tassilo Tiemann, Claudia Frost, Matthias Kühn, Verena Bärmann, Alex Jäger, Andrea Otto, Fabian Otto, Dr. Maria Knobelsdorf, Wiete Hagel, Patricia Luniak, Karsten Fiedler and the whole Cosmic Ventures crew.

With all my heart, I want to thank my parents Christine and Alexander, my lovely sister Silvia, and my parents-in-law Ingrid and Jürgen Kühn for supporting me with love, motivation and a wonderful backbone to easily come through all challenges.

Last but far from least I want to thank from the bottom of my heart my own beloved and wonderful family. You three have such an abundance of patience but have never been sparing with love, cake, spaceships, and infectious laughs. Thank you for always smiling and always showing me the bright side of life. Thank you to let me be a growing quarter of something really special. Thank you to my magical wife Moja (aka Marion or Marianne), my splendid son Frido and my shiny son Emil.

Berlin, March 2016
Norman Dziengel

Contents

Abstract	1
1 Introduction	3
1.1 Wireless Sensor Networks	3
1.2 Motivation	4
1.3 General Idea	6
1.4 Contribution	8
1.5 Thesis Structure	9
2 Current State of Research	11
2.1 Event Detection Architectures	11
2.1.1 Local Event Detection	12
2.1.2 Centralized Event Detection	14
2.1.3 Decentralized Event Detection	16
2.1.4 Distributed Event Detection	18
2.1.5 Local & Centralized Detection (two-tier)	19
2.1.6 Local & Decentralized Event Detection (two-tier)	20
2.1.7 Decentralized & Centralized Event Detection (two-tier)	23
2.2 Distributed Event Detection Requirements	23
2.2.1 # of Events	23
2.2.2 Information Fusion	24
2.2.3 WSN Scale	24
2.2.4 Realization	25
2.2.5 Applicability	25
2.2.6 Detection Algorithms	26
2.3 Conclusions about the Current State of Related Work	29
3 Pattern Recognition To WSN Transition	33
3.1 Sensing & Segmentation	34
3.2 Feature Extraction	35
3.3 Classification	35
3.3.1 Evaluation Metrics	36
3.3.2 Distance Measures	36
3.3.3 Classifier	37
3.3.3.1 Bayesian Classifier	37
3.3.3.2 KNN	38
3.3.3.3 Prototype Classifier	39

3.4	Report	39
3.5	Supervised Training	40
3.5.1	Feature Selection	40
3.5.2	Classification Model Generation	41
4	Distributed Event Detection System	43
4.1	Distributed Event Detection Concept	43
4.2	Evaluation Framework	46
4.2.1	Calibration	46
4.2.2	Training	47
4.2.2.1	Sampling & Segmentation	47
4.2.3	Modeling	47
4.2.3.1	Preprocessing - Signal Creation	48
4.2.3.2	Topology	50
4.2.3.3	Feature Extraction	51
4.2.3.4	Features	52
4.2.3.5	Feature Fusion	54
4.2.3.6	Feature Selection	55
4.2.3.7	Weighted Feature Selection	58
4.2.3.8	Model Creation	59
4.3	Distributed Event Detection Framework	60
4.3.1	Local Event Recognition	60
4.3.1.1	Sampling & Segmentation	60
4.3.1.2	Preprocessing	61
4.3.1.3	Central Node Extraction	61
4.3.1.4	Feature Extraction	61
4.3.1.5	Feature Distribution	62
4.3.2	Distributed Event Recognition	62
4.3.2.1	Central Node Filter	62
4.3.2.2	Feature Fusion	63
4.3.2.3	Feature Normalization	63
4.3.2.4	Classification	63
4.3.2.5	Quality Extraction	64
4.3.3	Report	65
4.3.3.1	Quality & Application Filter	65
4.3.3.2	Event Handling	66
5	Theoretic System Investigation	67
5.1	Distance Measure Criteria	67
5.1.1	Euclidean Distance	69
5.1.2	Mahalanobis Distance	69
5.1.3	Distance Measure Comparison	70
5.2	Information Fusion Approaches	72
5.2.1	Information Fusion Preliminaries	72
5.2.2	Raw Data Fusion	74

5.2.3	Feature Fusion	75
5.2.4	Classification Fusion	75
5.2.5	Multiple In-Network Fusions	76
5.2.6	Distributed Event Detection	76
5.2.7	Classification Accuracy Assumptions	77
5.3	Payload in Large Scaled WSNs	77
5.3.1	Payload Calculation	77
5.3.2	Payload Evaluation	80
5.4	Lifetime Extrapolation Concept	83
5.4.1	Energy Consumption	84
5.4.2	Lifetime Calculation Preparation	86
5.5	Lifetime Results & Evaluation	89
5.5.1	Lifetime Evaluation - Events per Hour	90
5.5.1.1	Best Case Setup	91
5.5.1.2	Standard Case Setup	92
5.5.1.3	Worst Case Setup	93
5.5.2	Lifetime Evaluation - # of affected SNs	94
5.5.2.1	Best Case Setup	94
5.5.2.2	Standard Case Setup	95
5.5.2.3	Worst Case Setup	97
5.5.3	Lifetime Evaluation - All affected Sensor Nodes pass the Central Node Filter	98
5.5.3.1	Best Case Setup	99
5.5.3.2	Standard Case Setup	100
5.5.3.3	Worst Case Setup	101
5.5.4	Lifetime Evaluation - Probability of Critical Events	102
5.5.4.1	Best Case Setup	102
5.5.4.2	Standard Case Setup	103
5.5.4.3	Worst Case Setup	104
5.5.5	Lifetime Evaluation - Probability of passing the CN-Filter	105
5.5.5.1	Best Case Setup	105
5.5.5.2	Standard Case Setup	106
5.5.5.3	Worst Case Setup	108
5.5.6	Lifetime Conclusions and Applicability	108
5.5.6.1	Reference Parameter Setups	109
5.6	Classifier Selection	111
5.6.1	Event Description	112
5.6.2	Event Categories	113
5.6.2.1	General Areal Categories	113
5.6.2.2	Construction Site Categories	114
5.6.3	Classifier Evaluation	114
5.7	Feature Vector Evaluation	116
5.7.1	Feature Fusion Inspection	117
5.7.2	Reference Vector Inspection	118
5.7.3	Missing Features	119
5.7.3.1	Re-Balance Method	120

5.7.3.2	Classification Performance with Inoperative Sensor Nodes	121
6	Distributed Event Detection Platform	125
6.1	Platform Architecture	125
6.1.1	Hardware Layer	125
6.1.2	Application Layer	126
6.2	Sensor Nodes	127
6.2.1	AVS-Extrem Sensor Node	127
6.2.2	F4VI2 Sensor Node	128
6.3	Casing	130
6.3.1	AVS Casing	130
6.3.2	F4VI2 Casing	131
6.4	Hardware Evaluation	132
6.4.1	Calculation Performance	133
6.4.2	Latency and PDR	133
7	Applications and Applicability	137
7.1	Fence Surveillance	139
7.1.1	Experimental Setup	139
7.1.2	Training	140
7.1.3	Results & Evaluation	141
7.1.3.1	Events	141
7.1.3.2	Categories	143
7.1.4	Fence System Comparison	145
7.1.4.1	System (a) - Neighborhood Confirmation	146
7.1.4.2	System (b) - Multiple In-Network Fusions	146
7.1.4.3	System (c) - Theoretical Distributed Event Detection	147
7.1.4.4	System (d) - Theoretical Distributed Event Detection with Un- trained Event Locations	147
7.1.4.5	System (e) - Distributed Event Detection System	147
7.1.4.6	System (f) - Distributed Event Detection System classifying Gen- eral Areal Categories	148
7.1.4.7	System (g) - Distributed Event Detection System classifying Con- struction Site Categories	148
7.1.4.8	ROC Analysis	149
7.1.5	Lessons Learned	150
7.2	Sport Device	152
7.2.1	Experimental Setup	153
7.2.2	Results	155
7.2.3	Evaluation	157
7.2.4	Lessons Learned	158
7.3	Therapeutic Exercises	159
7.3.1	Experimental Setup	159
7.3.1.1	The Bowman	159
7.3.1.2	Classical Frog Arm Pattern	160

7.3.1.3	Dumbbell Curl	161
7.3.1.4	Faulty Exercises	162
7.3.2	Results	163
7.3.3	Evaluation	164
7.3.4	Lessons Learned	166
7.4	Bridge Surveillance	167
7.4.1	Detection System to SHM Adaption	167
7.4.2	Preliminary System Validation	168
7.4.3	Experimental Setup	170
7.4.3.1	Event Experiments	170
7.4.4	Results	173
7.4.5	Evaluation	174
7.4.6	Lessons Learned	175
8	Conclusions and Future Work	177
8.1	System Requirements	178
8.2	System Realization	179
8.3	Theoretic System Analysis	180
8.4	Deployment Analysis	181
8.5	Future Work	182
8.5.1	Refinements	183
8.5.2	Extensions	183
A	Abbreviations	193
B	Zusammenfassung	195
C	Erklärung	197

Abstract

Wireless Sensor Network (**WSN**)s [1, 2] comprise multiple battery powered and wireless networking and sensor based minicomputers called Sensor Node (**SN**)s. **SN**s are typically equipped with at least one sensor to autonomously observe the environment by acquiring sensor data that is processed in the embedded hardware of the **SN**s in order to potentially communicate data to other wirelessly connected **SN**s of the network.

Event Detection is an observing and assessing process of real incidents or phenomena. **WSN**s have the potential to observe and detect environmental events in order to offer support in safety matters such as Structural Health Monitoring (**SHM**) [3] to detect e.g. age-related bridge damage, areal overviews to support firefighting operations [4], or fence monitoring systems to detect intruders [5].

Event detection with **WSN**s is challenging because events typically cause different measurements at **SN**s at different locations, but we want a single comprehensive meaning or interpretation as a result. A simple strategy based on redundant data collection makes sense in case of threshold based events such as fire detection, whereas pattern based events like intrusion detection at fences benefit from multiple perspectives on the event.

The requirements for event-observing **WSN**s are in contradiction to their properties, especially if they need to sustain their functionality over a long period and need to deliver accurate event detection while using small and ubiquitous sensor devices with limited energy and limited computational power.

This thesis presents a Distributed Event Detection system that shifts the evaluation process of the event data from the Base Station (**BS**) into the network. This distributed evaluation concept reduces the communication load to one single event notification which leads to an increased lifetime of the most charged **SN**s as well as the whole network.

We introduce two frameworks that help to realize the concept of the Distributed Event Detection, while having the goal to preserve the global event knowledge. The frameworks make this possible by preserving the diverse event perspectives of the **SN**s despite a necessary data reduction. The Evaluation Framework comprises the automated creation of an event model based on the data of a supervised training to support the transferability of the system on most different applications.

The Distributed Event Detection Framework enables real world deployments by implementing the detection system on the **SN**s that uses the trained model within the wireless network. The **SN**s affected by the event autonomously exchange event information within the network to cooperatively classify the event. In the end the final decision is made on whether the detected event is worth notifying the **BS** or any other responsible system for.

We present the technical limits for our system by investigating the trade-off between the communication savings to the **BS** and the increased in-network communication. Compared with other classical information fusion approaches, we can clearly attest an outperforming energy efficiency

due to the conceptionally reduced communication. All compared classical information fusion approaches are conceptually reused and modularly extended within the Distributed Event Detection system in order to ensure that our proposed system runs under a non-recommended setup as it can fall back step-wise to the underlying information fusion concepts to support the needs of the application.

The proposed Distributed Event Detection System reaches a high event detection sensitivity of more than 80 % up to 100 % depending on the application. With four real world deployments, we show the functionality, event detection performance, and the application specific lessons learned, which are evaluated in relation to the Distributed Event Detection's applicability.

CHAPTER 1

Introduction

1.1 Wireless Sensor Networks

Today, more than four million sensors are produced per day [6]. Such sensors present a chance for humans to extend our own limited sensory abilities to observe, understand, and reflect our environment in a much more diverse way. The huge and growing demand for these sensors shows that we are already making extensive use of this compelling chance.

Definition 1.1.1: Sensor Node

An embedded system called wireless SN that comprises at least one of several possible sensors that are soldered on a small sensor board which further consists of memory to store data, a sufficiently powerful Micro Controller Unit (MCU) to process data, a transceiver to transmit data, and a source of energy (battery, accumulator or energy harvester). SN tend to be very small in order to be integrated into our environment; hence, the miniaturized node size causes a constrained hardware performance and energy supply that limits the lifetime of the SN.

Definition 1.1.2: Wireless Sensor Network

WSNs can consist of only one SN (see Definition 1.1.1) that sends data to a BS with a gateway SN up to several hundreds of SNs depending on the application. WSNs combine the comfort of wireless communication with the extension of our cognitive abilities through sensors. Multiple SNs of a WSN build networks to cover areas with sensors or to span a path to a BS through the network and to exchange data between the SNs to organize themselves through the use of their networking abilities. The limited energy and computational power in combination with a high energy demand for communication with other SNs leads to the general problem of maximizing the lifetime of a WSN.

Such WSNs increase the ability of a single sensor entity to observe the environment in which they are deployed with the goal to observe events much more comprehensively and holistically. By using multiple, rather cheap SNs rather than one very expensive and less holistic observing station, the capabilities can be further increased.

The majority of the area of WSN research is working on solutions to especially expand the lifetime of WSNs by optimizing, minimizing or balancing the communication load [7,8]. The reason for this research concentration is that wireless data transmission is the most energy demanding task that is typically performed on a SN. Hence, it is necessary to reduce transmissions to a

minimum in order to lengthen the network's lifetime – which is the most desired characteristic for the applicability of **WSNs**. In addition – and especially for event detection tasks – it is necessary for **WSNs** to observe and assess the sensors' environment with the highest possible accuracy without significantly disturbing the environment or the user.

The main task of a typical **SN** is to collect information about the surrounding environment in order to use radio transmission to send these data to a base station or other sensor entities. Radio based transmissions are very energy demanding. Unfortunately, the embedded or mobile devices' value strongly depend on their lifetime, which simply means they should provide an assigned task for the longest possible time or they should at least be available for the longest possible time. Hence, **SN** components should be chosen with respect to their energy consumption – but more importantly, a technology is needed to organize the observation task within the network in an energy efficient and highly scalable way according to the number of participating **SNs**.

As a consequence, it is recommended to pre-process the sampled data in order to trim it down to the essential information. The aforementioned **SN** components' activity should be reduced to a minimum. In addition, most of the **WSN** applications tend to have requirements that force the **SNs** to be a very small device in order to provide ubiquitous applications, to hide, or to not disturb the main application with additional physical load.

1.2 Motivation

WSNs inspire researchers to the idea of intelligent and autonomous networks due to the fact that wirelessly communicating **SNs** and the concept of small and cooperative **SNs** comprise high potentials for ubiquitous real world applications. **WSNs** have the potential to observe environmental events e.g. in order to support us, especially in safety matters such as **SHM**, providing an areal overview to detect critical events during an firefighting operation, or providing detailed information about time-wise unpredictable events such as tsunamis or earthquakes. **WSNs** are typically used as data loggers, but rarely as a system with the goal to assess environmental events collaboratively.

We derive the most important properties in the following:

WSN properties:

- Wireless communication allows flexible deployments even in hard-to-access, infrastructureless areas.
- Lightweight and small **SNs** imply an ambient incorporation into the environment.
- Ad-hoc capabilities lead to easily scalable applications.
- Multiple multi-modal **SNs** enable holistic environmental observation.
- In-network communication enables collaborative decision-making with symbiotic effects.

The requirements of event observing **WSNs** are in contradiction to their properties, especially if they need to sustain their functionality over a long period in combination with the need to deliver an accurate event detection while using small and ubiquitous sensor devices with limited energy supply and computational power.

The well-known, agreed-upon assumption that wireless data transmission is the most energy demanding task that is typically performed on a **SN** leads to the necessity of limiting transmissions to a minimum in order to lengthen the networks' lifetime.

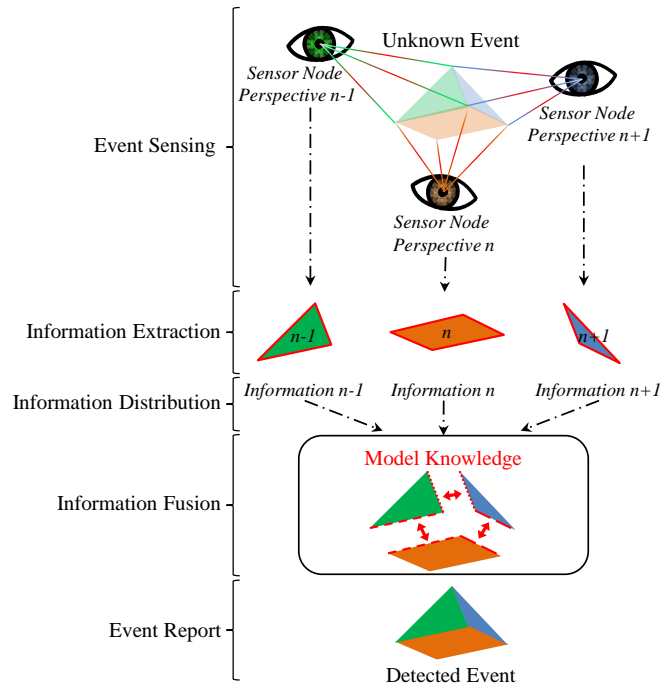


Figure 1.1.: General Distributed Event Detection Idea: Multiple sensor nodes observe an arising event from different perspectives to fuse the acquired data based on an a priori trained model.

Real World Requirements for WSNs:

- Hardware design needs to be miniaturized to fulfill state-of-the-art requirement of ubiquitously integrated and wireless communicating SNs.
- As SNs are often miniaturized, the energy supply needs to be a small battery or a constantly available energy-harvesting power source.
- Huge amounts of measured data have to be processed without limiting the WSN's lifetime, reactivity or data evaluation precision.
- The lifetime of all SNs has to be as long as possible.

WSNs that intend to observe the events mentioned above mostly collect as much event related sensor data as possible and transmit it to the BS or a server node where it is evaluated. Besides the lack of existing real world applications and deployments, most of these event-observing WSNs produce a stream of data to a base station or server SN which increases with an increasing number of affected SNs. This data stream is in stark contrast to the limited resources a WSN offers and is not recommended. Other typical approaches omit this data stream but do not maximize the computational potential of the WSN as they make use of threshold detection approaches that limit the accuracy of the event detection.

In addition, it is a goal of WSNs to observe the sensors' environment with the highest possible precision without disturbing the environment or the user unnecessarily. This means that typical miniaturized SNs are consequently provided with very limited energy and computational power.

A classical centralized information fusion at the BS causes high network load while especially the network load of all affected SNs has to be handled by the last Relay Node (RN) that has to forward

data to the BS. Typically, the RN's battery depletes first under these conditions which leads to a rapidly degrading network connectivity with the consequence that the BS has no connection with the remaining SNs.

The distributed computational power of the SNs is still not used to assess the incoming data cooperatively to detect events. Exactly this distributed computational power offers a high potential to facilitate WSNs to autonomously assess events within the network. We provide an event detection system that supports the distributed computational power to reduce the network load to the BS in the first place.

Our goal is to provide a fully applicable system that allows taking advantage of the distributed computational power and the wireless communication technology in practical deployments. We want to determine which applications and setups outperform classical information fusion approaches, although we need to increase the in-network communication to provide an in-network cooperation in general.

1.3 General Idea

The proposed idea uses the viable approach to reduce the communication with the base station by shifting the evaluation capabilities from the BS into the network. To solve the introduced contradiction between high detection accuracy and a long lifetime while applying miniaturized embedded hardware, we reduce the amount of data transmissions to the BS by aggregating collected environmental measurements such that the WSN is able to evaluate arising events observed by multiple SNs within the network by the use of a classifier.

Every SN provides event information as pieces in a puzzle as depicted in Figure 1.1, which allows the proposed system to observe events from different perspectives. The SNs need to know a dedicated classification model – which is comparable to the clue to a puzzle – to fuse the given pieces of information properly. The SNs collect these information pieces during the event sensing, extract descriptive information, and distribute this information amongst each other. An information fusion is performed by classifying the arising event collaboratively with a pattern recognition approach based on an a previously trained classification model.

The model knowledge of the classification model comprises global knowledge of the whole event in a holistic way. The SNs' extracted information is multidimensional and bears an event relation to the other affected SNs as depicted in Figure 1.1, [9]. The concept is able to detect and report events to a BS or directly to an event-driven process chain within or outside of the network.

An event-detecting WSN that uses a Centralized Event Detection (Figure 1.2 (left)) at the BS causes high network load. Conceptionally, it is a problem if multiple SNs start a communication over a multi-hop route to a BS as all involved relaying SNs suffer from early energy depletion if the communication is not well balanced. Even if the communication is balanced, communication initiated from multiple SNs to a BS lowers the overall network lifetime. In contrast to this effect, the lifetime of WSNs should always be extended as much as possible as the value of a WSN is directly linked to its lifetime.

We suggest applying the mentioned *Distributed Event Detection* approach (Figure 1.2 (right)) that reduces the initiated communications and the network load to the BS by shifting the evaluation into the network. The suggested solution is only possible at the cost of increased in-network communication, see Figure 1.2. The in-network communication is necessary to exchange event

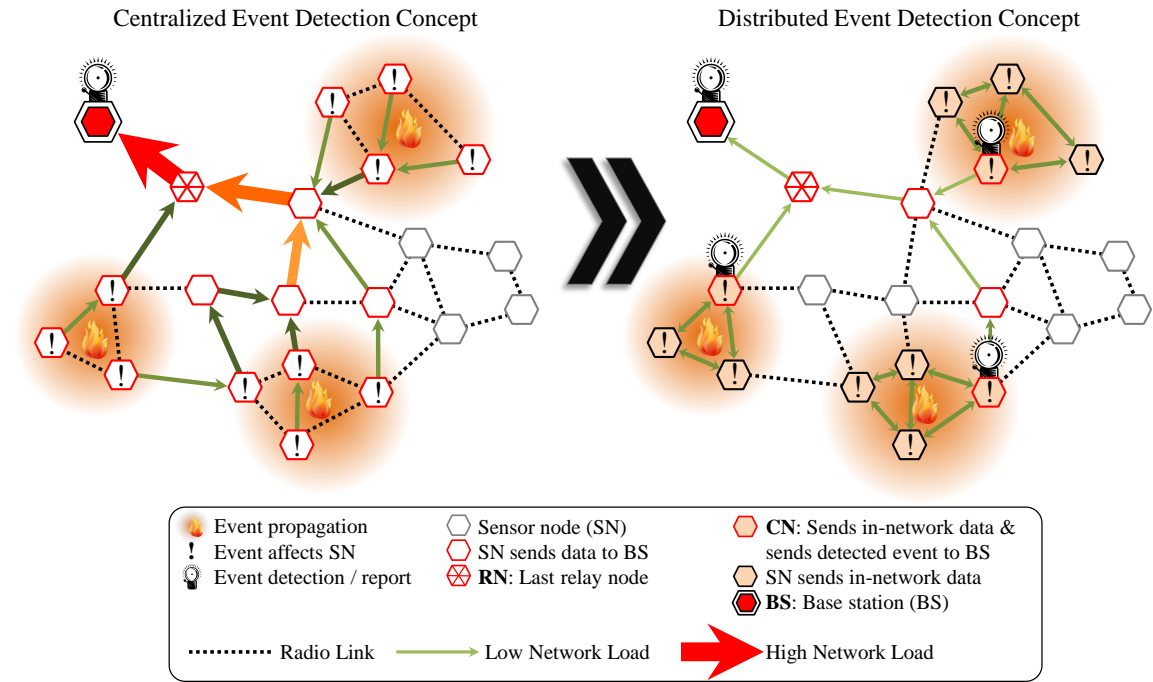


Figure 1.2.: Centralized Event Detection at the base station causes high network load (left). The Distributed Event Detection (right) reduces the network load to the base station at the cost of increased in-network communication.

information between the affected **SNs** and to filter evaluation criteria to find an appropriate evaluation node Central Node (**CN**).

We derive the following two main limitations for our solution idea:

- **Communication Minimization Limit:** While the communication to the **BS** is reduced, the in-network communication is increased.
- **Application Limitations:** If we reduce the sensed raw data to lossy event information characteristics, these characteristics always trade in some event details that reduce the event detection applicability in real world applications.

In order to show the potential and limits of our solution, we investigate the lifetime of dedicated **SNs** with an in-depth analysis. We are interested in the **SNs** with the highest computational and communication demands as these **SNs** are responsible for a first function failure as soon as the battery is depleted. We investigate the last **RN** that has to forward all data to the **BS** and the **CN** that is responsible for the final in-network event evaluation. Both of these **SNs** are highlighted in Figure 1.2.

1.4 Contribution

We want to define a hypothesis for this thesis in order to allow the reader to follow the subsequently listed contributions on a more context-oriented basis:

Definition 1.4.1: Hypothesis

“An autonomous resource-constrained WSN can be developed and deployed for a wide range of applications under real world conditions to observe and detect events cooperatively with an in-network communication-based event detection approach that clearly enhances network lifetime expectations and offers a reasonable detection accuracy.”

This thesis performs the validation of the stated hypothesis by presenting the elaboration of a four-fold contribution:

- **System Requirements:** A generic list of requirements is derived from the investigated related work to provide comprehensive recommendations for designing a cooperative Distributed Event Detection System for WSNs.
- **System Realization:** A Distributed Event Detection system for WSNs is presented which comprises two frameworks. The Evaluation Framework shows that the system is fully adaptable through a supervised training and classification modeling process. The Distributed Event Detection Framework shows that the system uses a cooperative in-network event detection system that fuses distributed event characteristics extracted according to the trained model.
- **Theoretic System Analysis:** A detailed theoretic analysis of the event detection system for multiple relevant parameters is presented that shows the system’s low energy demands, that the system is effective even in large scale networks and that it provides high classification results even in case of inoperative SNs. The system’s impact towards single SN detection system is shown. The decision-making ability of the proposed system can autonomously activate systems inside or outside the WSN.
- **Deployment Analysis:** The presented real world deployments span an area of four applications (in different environmental structures and different ways SNs are affected by events), which proves the system’s functionality under real world conditions. The application-dependent detection performance and the lessons learned identify the detection system’s aspect of feasible detection accuracy and wide transferability and applicability.

As mentioned, this thesis introduces a novel and completely autonomous Distributed Event Detection system for WSN events. The presented system extends the known information fusion approaches of raw data, feature, and classification fusion ([10–12]) by integrating them on the basis of the known input-output scheme of Dasarathy et al. [13]. Our system benefits from its own new qualities, but if applications do not conform to our recommended reference setup, the proposed system can apply the underlying information fusion approaches instead.

1.5 Thesis Structure

The remainder of this thesis is structured as follows:

2 Current State of Research: In Chapter 2, we give a broad overview of the current state of research by providing a taxonomy with seven different event detection architectures. From the related work, we derive a criteria catalog of requirements to build a profound Distributed Event Detection system that functions as a guideline for the current thesis. The concluding tables of the related work give a compact overview of these criteria and their application in the current research.

3 Pattern Recognition To WSN Transition: Chapter 3 gives a compact overview of the area of pattern recognition and how the introduced specialized needs and restrictions of WSNs have to be addressed within the classification approach. We address the typical process operation components of a classification problem comprising sensing and segmentation, normalization, feature extraction, an in-depth consideration of the used event detection metrics, and classifier variants. We conclude the chapter with a short introduction to the expected needs and challenges of a classifier training for WSNs.

4 Distributed Event Detection System: In Chapter 4 we introduce the main concept for the Distributed Event Detection system by giving an easy to understand overview of all components. The whole system comprises two main components, a training component called the Evaluation Framework and the deployable event detection component called the Distributed Event Detection Framework. Both frameworks are introduced in detail within this chapter while the leading Figure 4.2 helps to understand the whole process as a big picture.

5 Theoretic System Investigation: We investigate the introduced Distributed Event Detection System theoretically in Chapter 5 in order to validate how our concept will perform under various different parameters and their variations. For this purpose we investigate the advantages and disadvantages of three different distance measures in order to pick one for our evaluation basis. We introduce four different information fusion approaches that are to be compared with our own approach and derive their maximum classification accuracy potential. For all approaches, we evaluate the resulting payload within large-scale WSNs. Furthermore, we use real world energy consumption measurements of our system processes in order to extrapolate the lifetime of all information fusion scenarios by providing a best, standard, and worst case configuration with five system-influencing parameters. In addition, we use Fence Surveillance application data to evaluate three different classifiers with the introduced Evaluation Framework. To investigate the robustness and impact of the Distributed Event Detection we present an in-depth evaluation of the theoretic classification performance of our system under the influence of inoperative SNs.

6 Distributed Event Detection Platform: In Chapter 6 the platform architecture comprising the hardware layer and application layer is introduced. We introduce two different SNs, AVS-Extrem and F4VI2 along with their casing, as those have been used in subsequently introduced deployments. This chapter ends with a performance comparison of both SNs and a latency process investigation and Packet Delivery Ratio (PDR) investigation of the used transceiver CC1101 under multiple real world conditions.

7 Applications and Applicability: In Chapter 7 we introduce the universal applicability of the proposed Distributed Event Detection with the deployment of four applications which differ in their environmental structure and the way SNs are affected by the corresponding events. For all applications, we introduce the experimental setup, present the results along with an evaluation and present our lessons learned during the real world deployment. Depending on the application, we go into some more detail to enlighten specific historical development results of the Fence Surveillance Application, the casing setup of the Sport Device, application related requirements for faulty events of Therapeutic Exercises deployment, and preliminary system experiments for relevant event identification at the Bridge Surveillance deployment.

8 Conclusion and Future Work: We conclude the thesis by outlining the results of the theoretic investigations and the real world deployments in order to summarize the systems' applicability, energy efficiency and accuracy in Chapter 8. We finalize the thesis with an outlook of possible refinements and extensions of the introduced Distributed Event Detection system.

CHAPTER 2

Current State of Research

Data processing and event detection play a key role in [WSNs](#) and several different approaches have been developed and deployed to process and to transport data and events within a [WSN](#). The common goal is to use a [WSN](#) for event detection while the secondary goal is often to do this efficiently and with accurate results. Event detection has been a hot topic in [WSN](#) research in recent years as it reflects one of the most compelling use cases demonstrating the need for [WSNs](#). In the following section we give a broad overview of the existing research work in the area of event detection in [WSNs](#).

2.1 Event Detection Architectures

We mainly structure the related work by assigning them to one of seven *Event Detection Architectures*. More taxonomy details can be seen in the summary table in Figure 2.2. We want to introduce seven different types of *Event Detection Architectures* in the following sections based on their taxonomic relationship as depicted in Figure 2.1. The idea of this taxonomy is to reflect at which network point event decision are made and data or event fusions are processed and whether local or global knowledge is existing or lost. The taxonomy figure is split into two section, a global knowledge and a local knowledge section. For the purpose to clarify these terms, we introduce two definitions:

Definition 2.1.1: The *Global Event Knowledge* is a description of a relationship between data gathered by multiple [SNs](#), typically described by e.g. a classification model, a rule set or by the raw data of the [SN](#) as these contain all available knowledge of the events. The resulting event detection can bring results of multiple [SNs](#) into a meaningful relationship if data of that relationship is available that exceeds a local classification result, e.g. features, raw data, or other multidimensional information.

Definition 2.1.2: If a [SN](#) creates *Local Event Knowledge*, the event is classified with the goal to assess the whole event but from a constrained local perspective. Additional event information for or of other affected [SNs](#) have not been considered. The resulting classification or decision is one-dimensional and therefore the most reduced description of a global event. The resulting event detection lacks the possibility of bringing results from multiple [SNs](#) into a meaningful relationship other than e.g. a majority vote.

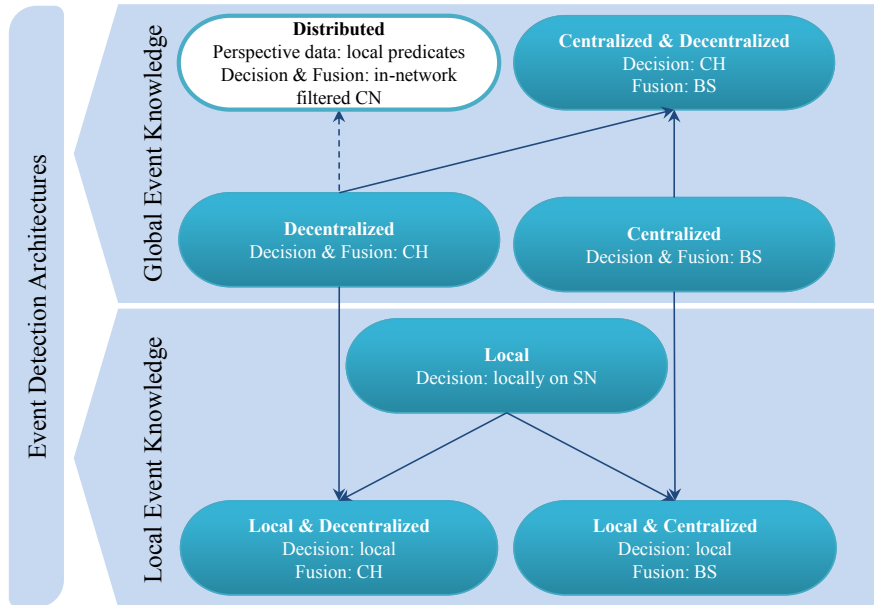


Figure 2.1.: Seven event detection architectures and their taxonomic relation in the context of global and local event knowledge.

We can summarize that if a classical centralized approach (data fusion at the **BS**) or a classical decentralized approach (data fusion on a dedicated Cluster Head (**CH**)) are combined with a preceding local event detection that creates classes/decisions based only on local data, the global knowledge is so far reduced that these architectures can only create decisions based on local event knowledge.

Representatives of the global event knowledge classical decentralized approach (data fusion on a dedicated **CH**) try to preserve some global knowledge. In this thesis, at least some global knowledge is available when data is fused on the **BS** for centralized approaches or when data is fused on the **CH** for the decentralized approaches. The combination of both architectures preserves the global information at least for the decision creation on the **CH** which still allows the preservation of the global data for a subsequent decision fusion of multiple **CHs**.

The distributed approach is technically close to the decentralized approach but differentiates too much for an official derivation from the decentralized approach. The distributed approach creates local predicates to define a local event perspective. These perspectives are fused and used for the decision creation on an in-network filtered **CN**.

We give a comprehensive overview of available related work for each event detection architecture in the subsequent section.

2.1.1 Local Event Detection

The *Local Event Detection* is completely and directly performed on the single **SN** with data from this single **SN** only. For the local event detection, only local event knowledge is available as defined in Definition 2.1.2.

If an event is distributed or propagated over an area, the event parts observable from every **SN** have global event knowledge as defined in Definition 2.1.1. This knowledge might be represented by a more detailed information such as size, duration, angle, intensity and many more details than

a pure local classification result of an arising event. If a classification is performed by preserving only local knowledge, this detailed information of the global knowledge can hardly be brought into relation with observations of other SNs. An example is a runner wearing multiple SNs. If each SN classifies its own data as “running”, the global knowledge of a possibly unbalanced running style is lost.

Of course, the local knowledge creation saves a lot of energy as it avoids transmitting raw data or any data other than the pure classification from a single SN. Only classification results need to be communicated to a BS. Nevertheless, this approach incurs a high communication overhead as multiple SNs need to communicate with a base station in case of an alarm.

The Local Event Detection is mostly applicable to fairly simple types of events without high complexity or the necessity to evaluate data from multiple sensor types. Nevertheless, the given papers have to deal with complex events like fence intrusion [14, 15] or earthquake detection [16] and fire detection [17]. A fire detection application could make sense as all SN can look for the same pattern of heat generation and changes in relative humidity. For the most part, the size of these networks don't exceed one or two SNs. Only one paper uses a feature fusion approach [18]. The *Local Event Detection* is mostly implemented in laboratory tests [16, 19] and real world tests [14, 15], while the used detection algorithms are widely distributed over the existing bandwidth of technology. Most of the papers promise a high applicability within their application domain.

Example implementations of Local Event Detection approaches are discussed as follows.

- In [14], Kim et al. investigated a typical threshold but comprehensive fence monitoring that covers a number of real world problems. It implements a multi-modal approach whose strength lies in the diversity of parallel sensing sensors: Ground and fence SNs aim to detect intruders, a network camera is used and supported by an unmanned ground and air vehicle to extend the communication and interaction of the system. After a node detects objects, it sends all sensing data to the BS and the camera is used to identify the intruder. As the local decision is responsible for any further action, we assess this approach as a local event detection. Their system is limited to events with a longer duration than 20s because of a duty cycle of 30s comprising 10s for sensing and 20s for an energy saving mode.
- In [15], Yousefi et al. evaluate an event detection system with a single SN that is attached to the middle of one fence element. They evaluate two events (rattling and climbing) that have been exposed to one fence element with a detection rate of 90%. The involved classifier is a Bayesian classifier at the BS with an underlying state machine. Resonance frequency based features are extracted and the frequency bands are matched to one of the events. The detection accuracy was above 90% for the data recorded from three different fences in a two-day real time test.
- A local event detection scheme based on fuzzy logic decision making is presented by Krasimira et al. in [17]. They investigated the applicability of fuzzy logic for fire detection for burning mattresses or chairs. The system detects fire events with fuzzy methods with a comparable accuracy as a Bayesian approach. The detection approach focuses on a single SN detection as the communication with the neighborhood slows down the detection, while the detection delay on a single SN varies between 97 seconds for burning mattresses to 236 seconds for burning chairs.
- Redhwan et al. introduce a low cost early earthquake warning system [16] that focuses on the

detection of the onset of an earthquake. In order to detect low frequency ground motions, data from a Microelectromechanical systems (MEMS) 3D-accelerometer and a piezofilm element is fused and analyzed over a looping 2s sliding time window locally on the SN. The event detection is based on a Markov process and evaluated with a Neymann-Pearson test using emulated magnitudes with vibrators and shake tables. The applicability of the rather simple algorithm on cheap and low-power SNs and a fair level of accuracy and efficiency could pave the way for domestic use.

- Zöttler et al. introduce and evaluate a prototypical event detection system for WSNs to observe conditions of containers on trucks in [19]. They observe shake and tilt events with an equipped 2D acceleration sensor that notifies the user during shock sensitive transports. The so-called decision maker (user) creates a priori score sheets on a smartphone by defining multiple thresholds for the sensors available on the SNs mounted in the truck. The score sheets are uploaded to the SNs. If one or multiple threshold are exceeded, a warning is transmitted to the user's smartphone for further manual decisions. The system was evaluated by investigating packet loss and Received Signal Strength Indicator (RSSI) values.
- Hein et al. introduce a Leaky Bucket Counter (LBC) [18] that is comparable to a classical hysteresis approach to detect events. They compare this local detection approach to a classical threshold approach and a running average approach by simulating the algorithms. Because of the slightly advanced threshold approach, we decided to classify the algorithmic approach as a model as it allows making use of a more complex model to describe events than threshold only approaches are able to. They define every modeled threshold as a feature which leads to multi-feature events based on multi-modal SNs. They provide a theoretical evaluation for the LBC by comparing it to a moving average window and a basic threshold mechanism.

2.1.2 Centralized Event Detection

In a *Centralized Event Detection*, the event assessment is performed exclusively at the BS. Characteristic event descriptions are extracted and global event knowledge is preserved because of detailed information transmitted in form of the raw data from every SN affected by the event as defined in Definition 2.1.1.

SNs of the WSN need to communicate over one to multiple hops with a BS in order to benefit from the BS's higher computing power and energy resources. Another reason for using this architecture is the comfortable way to program all SNs to communicate the gathered information to the same SN. The event detection is performed at that BS or any external hardware that is not comparable with the hardware of the SNs. The SNs just send raw data (compressed or optimized to reduce amount of data to send), pre-processed data chunks. A huge advantage for this architecture is that high precision data are available at one single point of evaluation, the BS. Bigger network scales in particular have the downside of energy problems that arise while streaming raw data through a multi-hop network. Especially the last RN close to the BS suffers from relaying all raw data from the WSN. As a disadvantage, this architecture is prone to the failure of the RN with the consequent possibility of an unavailable WSN for the BS.

The Centralized Event Detection has to deal with a huge amount of communication especially as it tends to be used by bigger network sizes. Only one paper in this section uses a classification

approach and not a raw data fusion [20] on the BS. It is worth mentioning that nearly half of the papers of this section propose either concepts, simulations, or off-line algorithms for their approach [20–24], while the other half propose real world implementations to verify the applicability [25–27]. Nearly all systems performed raw data fusion by transmitting the sensor data to the BS [21–27]. The implemented detection algorithms are clearly dominated by light-weight threshold detection algorithms but we can also see some model based approaches [21,27] as well as one pattern recognition based approach [20]. Nearly all papers are restricted to one specific application and about 50% of the proposed concepts are restricted to detect only one single event type.

Example implementations of Centralized Event Detection approaches are discussed in the following.

- In [20], Gupchup et al. proposed a statistical approach using the Principal Component Analysis (PCA) to build a compact model of the observed phenomena that captures daily and seasonal trends in the collected measurements. The goal is to predict the onset of rain events using the temperature sensors of the wireless SN. They sampled once every minute and aggregated the data over ten minutes of soil and air temperature measurements over a period of a year with ten SNs. They project the SN's daily measurements on this subspace in order to identify events by detecting deviations in this subspace on an external centralized system.
- Khelil et al. propose a distributed event detection technique [21] based on MWM (Map-based World Model), in which each SN creates a map of its environment and sends this model to a sink for detection of environmental events. In addition, the model can be added by topological information of the WSN. The authors show that the proposed approach is able to detect events in a timely manner.
- In [22], Li et al. provide a threshold based design for a reactive WSN to detect wildfire. The authors' approach is to stream raw data through a network to a mobile base station based on different spatiotemporal queries. By in-network processing, the region and spreading of the fire is approximated. The mobile BS is introduced to guarantee real-time data. They evaluated reactivity, reliability, robustness and network lifetime with a simulator.
- In [27], Lauterjung et al. present German Indonesian Tsunami Early Warning System (GITEWS) that has the aim to detect hazardous tsunamis within 5 to 10 minutes in order to provide a comprehensive threat prediction for the population. GITEWS is actually a running system and was installed by the German Centre for Geosciences (GFZ). It uses monitored data from more than 300 sensors where mainly Global Positioning System (GPS) positions help to reflect significant displacements of the Earth's surface. In combination with seismological data they can assess earthquake fractures within 5 minutes. Furthermore, pressure sensors, radar and floaters combined with additional GPS sensors help to detect displacements of the surface. All data are pre-processed and verified on-line and transmitted over satellite to the Decision Support System (DSS). The centrally aggregated data are compared to pre-calculated modeling to reflect the current situation. This may lead to a region-dependent warning dissemination. In contrast to the paper, the project website currently states that they stopped the initial buoy system as it is not able to deliver information in real time about earthquake location and magnitude. Instead, the GPS based solution is still running.

- Bo et al. introduce an abstract and high level event detection system for coal mines [23] using WSNs with a finite automation algorithm. They introduce an event monitoring system that transmits all information of methane and oxygen concentration from underground ZigBee based wireless SNs to wired cluster heads which we classify as BS due to the wired connection. Further event handling is conducted within an off-line Service-Oriented Architecture (SOA) using the Business Process Execution Language (BPEL). They implemented the concept but do not share any performance results.
- Zan et al. introduce a leakage detection in pipelines for urban water supply [25] by using an anomaly detection approach using a joint time-frequency analysis. They analyzed the detection approach in a test-bed with 35 SNs deployed in the FPCW water distribution zones in Singapore. Every 30 seconds, they transfer compressed data (pressure measurements acquired at 250 Hz) to a base station, where the burst detection algorithm is applied.
- Viani et al. deployed a system for prevention wildlife-vehicle collision in [26]. The network consists of SNs, a gateway node for connecting the control unit (sink) and communicates over multiple anchor nodes to support redundant communications paths. If a movement event is registered and triggered by the sensors (ultrasound sensor, infrared, Doppler radar), the SNs stream the raw data to the BS for sensor data fusion to calculate the target's direction, velocity, and size. If a certain level of danger is found, the actuator node is activated and a light signal is given in real-time. They tested the functionality with four SNs in multiple deer detection scenarios. The false-positive rate is very low (2%), which means that nearly every wildlife presence is detected.
- Yang et al. introduce an event capturing scheme for structural health monitoring [24] that addresses short term events mainly to detect earthquakes of bridges. Their goal is to collect data of all SNs of the bridge on a BS. To achieve this, they address the problem of waking up all nodes as quickly as possible. All nodes triggered by vibration broadcast a wake up message to all other nodes to let all nodes collect data as early as possible. Based on the fused data, the BS decides whether the SNs need to continue collecting data and sends an appropriate notification if needed. They state that a synchronization message can delay the data capturing start. In order to minimize missing data, they suggest synchronizing the data after finishing data collection. They simulated the entire algorithm.

2.1.3 Decentralized Event Detection

Decentralized Event Detection assesses the gathered raw data or just classification of one or multiple SNs on a CH. Characteristic event descriptions are extracted and the global knowledge (according to Definition 2.1.1) is preserved by representatives of this approach at every SN affected by the event.

A typical CH selection increases the energy demands as CH selection approaches are still energy demanding due to their concepts and come with additional energy demanding tasks of cluster formation, creation, and distribution as summarized in [28]. A proactively determined CH suffers from the repetitive task of handling each event fusion and detection within a cluster and therefore behaves similar to a RN as this node relays a majority of data to a fixed central communication node.

The majority of the event detecting research approaches in WSNs available at this moment pursues the strategy to perform the event detection on a special designated and predetermined CH or SN within the WSN. This node classifies the event or makes a decision about the importance and communicates with the BS if necessary. For further in-depth literature on clustering algorithm please refer to [29]. Further, cluster activity can be organized based on events, which means that all clusters not affected by an event can remain in an energy-efficient standby mode. A disadvantage of the architecture is the potential communication of the raw data from the SN to the designated CH which causes a heavy energy demand for the SN as communication is still a very power consuming task. The majority of the decentralized evaluating papers detect between two and ten events [30]. Nearly all papers using the decentralized architecture focus on a decision fusion on the cluster heads within network sizes from mostly four to seven [30, 31], but up to 30 [32] or even 50 SNs [33]. None of the papers perform real world tests, but some of them implemented laboratory tests [32–34] while all others use offline algorithms [30, 31] or simulations [32, 35–37] to verify the applicability and to evaluate the approaches. Decentralized event detection is performed with pattern recognition [30, 34] as often as with model based algorithms [31, 32, 36].

Example implementations of Decentralized Event Detection approaches are discussed in the following.

- In [33], Waelchli et al. implemented a Decentralized Event Localization and Tracking Algorithm (DELTA) to detect flashlight events in an office environment for intruder detection. This approach sends event data to a cluster head to estimate the event position and to classify the incident based on a Simplex Downhill algorithm. In this particular simple event type – the appearance of a light – the global knowledge of the flashlight motion is preserved by maintaining the spatial distribution of the SNs and by evaluating the position in relation to each affected SN. In contrast to our work, the author detects changes in the time series of the measured data with an anomaly detection approach.
- Wang et al. [34] uses a decentralized event detection architecture and describes a habitat monitoring system that is capable of recognizing and localizing animals based on acoustics, e.g. frog calls. They observe acoustic signals and classify different animals by the maximum cross-correlation coefficient between both the reference spectrogram and the unknown spectrogram. They employ a CH with additional processing capabilities that detects crying animals locally with spectrogram pattern matching and requests compressed raw data from other nodes for the localization of the detected animal. The subsequent localization is based on beamforming using Time Difference of Arrival (TDOA).
- A decentralized event detection system is also proposed by Martincic and Schwiebert [36]. SNs are grouped into grid-like cells based on their location. All nodes in a cell transmit their raw data samples to a CH which averages the results and retrieves the averages from adjacent cells. The raw data preserve the global knowledge. Event detection is performed on the CHs by arranging the in-cell collected averages in the form of a matrix and comparing it to a second predefined matrix that describes the event. An event is detected if the calculated matrix matches the predefined matrix.
- In [37], Thuc et al. investigated a fuzzy logic based approach that clusters the simulated sensor network. Each cluster is assessed based on historical detection values (true positive rate and false positive rate). Their credibility value represents the probability that a certain

cluster result is trustworthy or not. The global knowledge is preserved by using fuzzy logic and taking the individual credibility into account when the final decision is made at the fusion center.

- In [30], Ghasemzadeh et al. introduce the concept of motion transcripts and proposes an event detection approach with pattern recognition techniques to detect specific human motion patterns. The transcripts define parts of motions to construct the whole motion from inertial sensors and identify human motions by using a classification fusion at a dedicated and predefined server station or CH. The authors collected the local classification results with seven SNs, performed the classification, and evaluated the resulting detection accuracy with MATLAB®. In contrast to our approach, each SN is assigned a specific role on the body and the applicability of this approach is restricted to the human body.
- Probabilistic model-based techniques as proposed by Jie et al. in [32] utilize probability theory to model two events and to detect them if their occurrence is probable. STED (Spatio-Temporal Event Detection) detects events using a belief propagation technique. 30 SNs are used to collect light strength data every minute, while the event detection evaluation is performed on an external computer. Additionally, sensor synthetic data is created and the event detection is simulated with 500 SN to investigate the large scale capability of the approach.
- Jovan Rada et al. evaluate in [31] an ice-warning system for roads to prevent drivers from inadvertently driving into slippery areas. They used Dempster-Shafer's model of belief function in order to detect three different events: freeze, slip, and safe. SNs with temperature sensors send their raw data to a stationary and solar-powered CH near the roadside. This CH exchanges the mass vector with the confidences of all subsets of the frame of discernment with the bypassing car. The mass function of the car is also taken into consideration for the event detection. They evaluated the system by emulating the algorithm based on the sensor data of three SNs and one vehicle.

2.1.4 Distributed Event Detection

The *Distributed Event Detection* is the introduced approach used in this thesis. It performs the event detection on a SN that is located according to the event's center and uses compressed descriptive data of all neighboring SNs affected by the event.

The Distributed Event Detection approach proposed and evaluated in this thesis creates a unique perspective towards the event on each SN by creating local multidimensional descriptive local data and distinguishes itself from the Decentralized Evaluation in multiple aspects:

Without the need for a dedicated CH or BS, the SNs of the distributed event detection communicate with each other to autonomously classify the event together. Part of this process is a filter process to e.g. find a CN in a context-based fashion.

The whole event detection is pre-processed, calculated and decided within the network without the need for any external BS. Characteristic event descriptions need to be extracted at every SN affected by the event. These descriptions are in relation to the global knowledge (Definition 2.1.1) and represent pieces of the complete event that are fused within the network based on the global knowledge (e.g. a classifier model) to provide a complete description of the events. This means

that every single SN is capable of performing the final classification as all SNs run the exact same code.

As a result, only a single packet transmitting the event description may be sent to the control center or BS for further actions. This approach does not need to communicate with a BS as the classification result is available for all subsequent processes based on the autonomous in-network detection. The in-network classification can immensely reduce communication with the BS if additional knowledge about the importance of certain events is available (additional context knowledge). For example, during working hours it is not necessary to send an alarm if someone opens the door of a fence on a construction-site, while such events during all other times should raise an alarm.

Only one paper featured a real world deployment using a distributed approach [38]. However, it is unrealistic to transfer the concept to other applications as the physical displacement and destruction of the SNs are part of the event. All other papers use rather simple threshold detection algorithms and all are designed to detect only one event type, which means that all approaches are bound to a very specific application.

Example implementations of nearly Distributed Event Detection approaches are listed below.

- Li et al. [38] propose a highly specialized event detection system for coal mine surveillance to localize collapse events. The Structure-Aware Self-Adaptive principle (SASA) of the 3D environment surveillance is used to detect falling nodes or changing positions of SNs by finding anomalies in acceleration data, RSSI data and acoustic measurements within a coal mine. In case of an event, the involved nodes are mapped and transformed into a signature file that is transmitted to the control center where all data are evaluated centrally. The authors conducted a large-scale simulation to evaluate the system's scalability and reliability.
- In [39], Li et al. propose a system to detect a fire event as a composite event which is constructed out of a high temperature event, a dazzling light event, and a dense smoke event. An event is detected if one or multiple sub-events occurred, while simultaneous and sequential occurrences are supported. Every SN provides a local and threshold based event decision which are sent to a gateway that checks if the fire event can be composed. The approach allows evaluating the distributed fire event on every SN within the network by composing threshold-based and locally detected events from multiple SNs. The authors evaluated the approach in a simulation process.
- Vargas-Fallas et al. introduce a distributed debris flow detection system for rivers [40]. The SNs are assumed to be deployed along the observed river and its riverside. During an arising debris flow, these nodes are transported by the flow down the river. The authors want to detect topology changes by counting newly available links to previously unconnected SNs and counting link losses to previously connected SNs. If the number of new links or lost links exceeds a certain threshold and messages of multiple SNs confirm this within the network, a warning is sent. The system was evaluated by simulations and seems to be a rather theoretical approach as real world issues have not been discussed at all.

2.1.5 Local & Centralized Detection (two-tier)

The two-tier architecture *Local & Centralized Detection* combines the introduced architectures of the local SN event assessment and of the centralized BS fusion of several of these local assessment

results.

The combination of local and centralized detection saves communication as most of the gathered data are mapped to a single decision locally on a *SN*. This architecture inherits the disadvantage of the lost global event knowledge due to the local event detection according to Definition 2.1.2. In contrast to the combination with the decentralized event detection, the centralized detection saves the additional implementation for a cluster head communication but incurs more communication to reach the *BS* if network sizes are increasing. The final centralized data fusion is realized in most of the cases based on a majority vote as no event context is preserved after finding an event decision locally on the *SN*. A typical application that can be found in recent research for this architecture is to gather data locally on *SNs* and pre-process them on *SNs* to finally evaluate the event in an external simulated environment. For this detection method, comparatively few research papers exist, none of which use a real world deployment for the detection. Instead, all of the papers use a pattern recognition approach and support the detection of multiple events. The only trainable approach [41] is evaluated by simulation.

Example implementations of Local & Centralized Event Detection approaches are listed below.

- The system proposed by Yang et al. [42] is aimed at recognizing human activities. It is a Body Area Network (BAN) consisting of eight *SNs* attached to the body of a person who may perform one of twelve detectable actions. Features are extracted from an accelerometer and gyroscope and classified on each node. As this local pre-evaluation does not consider the event context, we classify this approach as producing local knowledge. If a local classification is promising, the data of all nodes is transmitted to the base station and classified once again in a centralized way. The classification process identifies an action by matching the linear representation of the extracted feature vector to one of several subspaces, each of which corresponds to one type of action.
- In [41], Zoumboulakis et al. define complex events as “sets of data points that constitute a pattern” and detect events using a distance metric for a string matching approach, called symbol aggregate approximation (SAX) transformation. The whole system is separated into two phases: a learning phase and a detection phase. They transform sensor data into characters in order to calculate the shortest distance between previously trained events and unknown incoming events. The approach classifies the extracted event string of the *SN* locally for autonomous operation. The authors stated that the techniques are implemented in MATLAB® and the simulations are based on real sensor data from 54 *SNs*.
- Doolin et al. described a threshold based approach [43] to detect fire incidents. Temperature, relative humidity, barometric pressure, and *GPS* data is gathered with onboard sensors. When a fire occurs, the sensors measure temperatures and humidity values which do not occur under normal environmental conditions. The *SNs* can then assume that a critical event has occurred and send an alert to the base station. They deployed 10 *SNs* into a field and created a real fire wall to detect fire causing every *SN* to eventually fail to report data at some time before, during, or after the passage of the flame front.

2.1.6 Local & Decentralized Event Detection (two-tier)

The combination of the two-tier *Local and Decentralized Event Detection* introduced above uses a preliminary classification results performed locally on the *SNs* with local event knowledge only

(see Definition 2.1.2) with the aim to fuse these classification results on a CH.

The decentralized fusion is able to evaluate event detection results within the network at a CH but still has to deal with additional communication demanded by the CH as introduced in Section 2.1.3. Typically, a classification is performed on the SNs. This causes information loss as all data is assigned to one single class without considering the context of the neighboring SNs. The cluster head then has no access to these measured data anymore and needs to work with the lossy data set. Mostly, this widely used concept is used to perform a majority vote at the CH with decisions created on SN of the neighborhood.

Although this approach seems to be a straight forward solution for broad WSN applications, it is rarely found in real world deployments [44] or even laboratory tests [45]. In one case an off-line algorithm is implemented [46] and all other approaches use simulations [47–50] or a concept [51] to evaluate the approach while none of these approaches used data from real SNs.

Example implementations of Local & Decentralized Event Detection approaches are listed below.

- Shukui Zhang et al. provide a fuzzy-decision approach (FL-CED) [49] that can determine whether a composite fire event is ongoing or not on a cluster head. The cluster head sends the detected fire event to the base station, which then derives the threatened area from all received events. They use three types of cooperating SNs: temperature, smoke, and light sensing nodes. In contrast to classical fuzzy logic approaches, the approach can be customized to the event type by adjusting the membership matrix and weight matrix, which considerably reduces the limiting size of the fuzzy rule base for simple events. The constructed membership functions for every sensor are based on predefined thresholds (e.g. temperature). They evaluated the approach by simulating it with MATLAB® and Java™ to evaluate the detection accuracy, fault detection capability and error rate.
- A threshold-based event detection concept for heterogeneous sensor networks is proposed in [51] by Kumar et al. The authors motivate locally evaluating threshold-based events and incrementing so-called event counters. A CH collects the counters of the child-nodes and matches them with a pre-configured event matrix. If e.g. an explosion happens, the child-nodes indicate sub-events such as light, heat, and noise that will be assessed at the CH to cause an alarm if these sub-events coincide. With a proposed event tree, a publish/subscribe communication-like paradigm for in-network collaboration is to be achieved.
- Duarte and Hu [44] use pattern recognition algorithms in a vehicle tracking deployment. After gathering the acoustic and seismic data, each SN classifies the events by using extracted features. The features are extracted from the frequency spectrum after performing a Fast Fourier Transform (FFT). The detection comprises three classification algorithms: k-nearest neighbor, machine learning, and support vector machines. The classification result is sent to a fixed cluster head for detection reasons and is combined with reports received from other nodes for tracking a vehicle. The level of classification fusion inherits information loss already at the SNs.
- In [48], Malazi et al. present a fuzzy based FED as an event detection approach that groups SNs into clusters and detects events locally on the node and maps them to fuzzy values on each SN. The fuzzy values are estimated with a membership function to classify the arising events. This process is comparable to a classification progress with the difference of the uncertain meaning of the fuzzy values. The event detection comprises three main steps: simple event

detection, composite event detection, and event stream maintenance. The fuzzy values are constantly analyzed at the cluster head in order to estimate the frequency occurrence, the correlation of the collected events, and the relevance.

- In [47], Moradi et al. present a fusion-based event detection method based on Bayesian approach, in which a Kalman estimator is used to replace missing data with approximated values. Events are then detected by a Bayesian detection scheme locally on the **SNs**. The global decision fusion fuses all local decisions at the fusion center of neighborhood **SNs** within the network. The proposed system is simulated with 400 **SNs** and it is assumed that 100 **SNs** are involved in one single event.
- Gu et al. implemented a ferrous object tracking with **SNs** in [45]. They introduce a hierarchical classification architecture consisting of four tiers: sensor-level, node-level, cluster-level with static cluster heads, and sink-level. Each level fully classifies the arising events and communicates the result to the higher level, while the sensor-level is the lowest level. Classification results are represented as confidence vectors in order to reduce the communication overhead calculated locally on the **SNs**. Based on a preliminary majority vote, the **BS** calculates the direction and velocity of the detected targets on predefined thresholds.
- Tavakoli et al. [46] consider a scenario in which targets are tracked using an undersea acoustic sensor network. Similar to the approaches [45] and [48], the **SNs** report their local classification result to a cluster head, which in turn performs an evaluation of the data and may report the outcome to a **BS**. In addition to the number of incoming reports, the **CH** also considers the accuracy of past reports. They introduce a hierarchical classification architecture consisting of tree tiers: sensor-level, node-level, cluster-level, and sink-level. The sink finalizes the results based on a confidence value derived from historical data.
- Hong et al. introduced an event detection scheme [50] for **WSNs** that collects local event decision of **SNs** on a **CH**. **SNs** with a closer proximity to the event and to each other are rated higher to reduce the false alarm rate. The authors assume a constant communication range and use the transmission range as indicator of physical neighborhood. They simulated the threshold-based multi-tier event detection approach with 320 **SNs** in a 4x4 grid containing 20 **SNs** each.
- Schuldhaus et al. suggest a majority decision fusion based system to classify seven different human actions with four sensors mounted at wrist, hip, chest and ankle in [52]. For each **SN**, they used six time domains-based features to classify the human actions locally at every node. Classifications on the nodes are performed for a five second window while using sliding windows with a 50% overlap. With two different classifiers, they gain an average classification rate of 93.9%.
- In [35], Ko et al. simulated a threshold-based Decentralized event detection scheme to detect events locally within a **WSN** that needs to be divided into square grids. The local event detection at the **SNs** provides a 1 if an event is detected. Each grid is defined by a **CH** that counts arising events of assigned **SN**. The **CH** shares the collected information with neighboring **CH** to give support to the threshold-seeking main algorithm.

2.1.7 Decentralized & Centralized Event Detection (two-tier)

The combination of decentralized and centralized event detection shifts the decision level from the **SNs** to the **CH** and the results of that detection are sent to the **BS** for the next fusion step. This causes a very heavy load on communication within the cluster, as all raw data need to be transferred to the specific **CH**, but it preserves all information gathered by providing the global knowledge according to Definition 2.1.1 within a cluster to the advantage of a high precise cluster based event evaluation. If events are detected by multiple clusters, the next fusion step can be performed at the **BS**. The **BS** uses the reduced data to fuse multiple decisions for a final result. This approach is seldom represented in the area of research as it seems to be very much tailored to wildfire where the lifetime of the **SN** has to be limited until the event occurs. Especially the **SNs** affected by the event are exposed to high traffic load within the cluster. This is acceptable as long as they can fulfill the task at least once. It is very likely that after such an event the affected **SNs** are defective due to the fire, hence a battery or a node replacement will be necessary for these nodes anyway.

One example implementation of *Decentralized & Centralized Event Detection* approach is listed below.

- Hao et al. propose a fusion approach [53] driven by the Mandani fuzzy logic method Cluster-based Fuzzy Decision Fusion Algorithm (CFDFA) in **WSNs**. With K-means clustering, the optimal cluster centroids are calculated. Mobile **CH SNs** will move to these centroids to reduce the intra-cluster communication for the **CH** fuzzy fusion based on the received raw data of all cluster **SNs**. They do not discuss any real world issues that could arise if a **CH** needs to physically move to a certain position in order to reach an advantageous position for data transmission. The **CH** decisions are fused at a **BS**. They evaluate the detection of fire events based on temperature, light intensity and Carbon Monoxide density measurement by simulating 200 static and 15 mobile **SNs**.

2.2 Distributed Event Detection Requirements

In order to provide future developments for Distributed Event Detection systems for **WSNs**, we derive fundamental requirements that are essential for an applicable system.

We derive the requirements from the investigations of the prior related work and our experiences while developing our own system. We postulate that all the following features should be considered in order to guarantee high quality in event detection for a **WSN** and as a consequence, we define these requirements as goals to be fulfilled.

In general, an event detection system needs to guarantee a high detection accuracy while minimizing communication. Nevertheless, a certain amount of inter-node communication needs to be considered in order to provide a holistic event detection by providing a collaboration of all **SNs** affected by the event.

2.2.1 # of Events

53% of the considered papers are capable to detect more than one event. A Distributed Event Detection system should be able to distinguish multiple events and 53% of the related work perform their detections based on a single event.

Approaches that are able to detect only one event or very specific events cannot provide a broad event diversity which is essential for a transferability of the system to multiple applications. Most event detection scenarios in the real world are very diverse and it is unlikely that only one specific event needs to be analyzed. Even widespread fire detection systems typically need to distinguish between fire, heavy smoke, low smoke, and no fire. In addition, it may be reasonable to add chimney events and multiple meteorological events to provide a low false alarm rate.

It should at least always be possible to distinguish critical event categories from uncritical event categories as suggested in Section 5.6.2.1.

2.2.2 Information Fusion

Raw Data Fusion is widely used (26%) and provides the optimal event assessment but causes high communication load as raw data have to be communicated, the principle of raw data information fusion is introduced in detail in Section 5.2.2.

Decision Fusion based systems are widespread with 50% in the related work. Decisions are made based on thresholds, anomaly detection, model, or fuzzy based approaches. The final communication load to the BS is conceptually identical to the Classification Fusion as only classes, or in this case decisions, have to be transferred. Typically, collected decisions run a majority vote to assess events on the basis of multiple classification results if no global knowledge is available.

26% of the related work described above uses a Classification Fusion that results mostly in a majority vote to assess events on the basis of multiple classification results if no global knowledge is available. Some Classification Fusion approaches use multiple classifiers to assess a single event.

As a guideline we recommend using more of a compromise approach that preserves as much information as possible while reducing communication to a minimum.

The Feature Fusion is used by only 8% of the related work. It is our preferred variant of fusion approaches as it describes events with multiple but limited descriptive features while limiting the communication to a reasonable amount of data. Compared with the Decision or Classification fusion an increased amount of data will be transmitted but the resulting event detection has a better data basis to elaborate on than just a classification or decision.

2.2.3 WSN Scale

Nearly half of the related work considered (45%) developed their approaches without considering a true WSNs scale. In these cases, artificial (synthetic) sensor data, data from theoretical considerations, or data derived from one SN is used. This can lead the researchers to unrealistic assumptions about their expected real world problems.

A Distributed Event Detection system should be applicable in larger WSNs and based on realistic WSN scales to provide future applications with increasing numbers of SNs.

In order to adapt an event detection system to multiple applications, it should be considered that applications using WSNs tend to utilize the main advantages of WSNs, which are scalability and flexibility. The integration of multiple SNs in our environment (or industries) is on the rise not least because of the further reduced size of SNs which will lead to larger WSNs with more measuring entities integrated in close range applications as well as huge area applications.

To reveal true problems that are caused by the event detection deployment with SNs we recommend to apply larger networks of more than 10 SNs (defined as *mid-scale WSNs*, see Figure 2.2) in order to understand event detection problems. 72% of the related work did not follow this scale

suggestions. To cover the multi-hop communication effects and influences on the lifetime of the network during application, at least mid-size networks of 11-39 SNs should be involved, which is already done by 26% of the related work considered. 18.8% of the research considered already uses large-scale WSNs (>39 SNs), which is of course the optimal case for an evaluation.

2.2.4 Realization

66% of the papers considered realized the introduced approaches solely by concept descriptions, simulations, or offline algorithms. Concepts are understood as purely theoretical and not yet implemented ideas. Simulations are performed in a simulation environment or by using mathematical calculation and evaluation. An offline algorithm is an implemented solution for the WSN problem but running on a single computer or server. A Distributed Event Detection should be tested and evaluated in a real world application to understand and uncover problems that would not be addressed in a purely theoretic investigation. A real world deployment helps to find utilizable solutions for complex real world sensor network applications. To verify high applicability, it is essential to perform real world deployments, which is done by 21% of the considered related work.

Performing experiments with real hardware and wireless communication during the evaluation process helps to understand effects of hardware failure, environmental effects, network and communication problems, and computational complexity of the scientific problem and to decide whether these effects are part of the problem.

A laboratory test is performed by 21% of the related work and helps to verify a concept by an implementation on the SNs. It represents an implementation of the detection algorithm on the SNs and adds the necessary communication between the SNs, which provides a better understanding of the problem and verifies whether an approach solves an event detection problem. As the word *laboratory* indicates, the experimental conditions are optimized for the experiments and do not reflect real world conditions.

2.2.5 Applicability

An event system should ideally be adaptable to multiple detection applications. The majority (74%) of the related work is restricted to one dedicated application or a dedicated application domain: fire detection [22, 43, 49, 53], detection of burning mattress and chairs [17], explosion detection [39, 51], onset of rain detection [20], tsunami detection [27], coal mine collapse detection [38], toxic gas detection in coal mines [23], earthquake detection at bridges [24], domestic earthquake detection and early warning [16], burglars using flashlights detection [33], icy road detection [31], debris flow detection [40], vehicle tracking [44], tracking ferrous objects [45], submarine detection [46], wildlife detection to prevent collision with traffic [26], frog sound detection [34], human motion detection [30], human action detection [42, 52], fence intrusion detection [14, 15], shake and tilt events during transport [19].

18% of the related work considered does not mention any kind of specific applications. In addition, all of these works introduce conceptual ideas or purely theoretical ideas without providing any practical experiments, laboratory tests, or simulations.

Only 5% of the related work considered can be trained for multiple applications as shown in [32] and [41]. This trainability is a very good way to leverage event detection systems to high levels of applicability toward different applications and is therefore recommended.

2.2.6 Detection Algorithms

To provide accurate event detection accuracy, it is important to choose an event detection algorithm that is adaptable and able to reflect the whole complexity of distributed events that take multiple parts of events with different sensors and measuring points into account. The following five algorithm types reflect the typical implementations found in the related work considered.

Threshold Based Algorithms This algorithm is used by 32% of the related work considered. Threshold values are suitable for a lot of applications, e.g. fire detection, detection of flooding, or other applications in which a sensor can detect a critical boundary of the measured value. Approaches with a threshold-based detection algorithm are lacking in accuracy as only a rather simple parameter is used to indicate an event, which typically leads to a high rate of false positives or false alarms and a high rate of false negative if application parameters change slightly (temperature, duration, intensity). Threshold-based algorithms are efficient and robust for simple detection problems, but do not adequately solve detection problems of events that are composed of multiple sub-events such as movements, human motions, environmental disasters, or structural destruction detection.

As an additional possible downside, every node that detects an event because a threshold value was exceeded causes a communication with other SNs. This may cause an early energy depletion of multiple SNs in the WSN if the system often raises false alarms because of the imprecise threshold approach.

Anomaly Detection This algorithm is used by 5% of the related work considered. Anomaly Detection is a term used for approaches that focus on the specific case of detecting whether a particularly unusual event has occurred within a time-series that inherits healthy status if nearly unchanged or constant values are expected. This is achieved by learning typical system behavior over time and then classifying specific events as either normal or anomalous. Approaches with this goal expand upon the principles of the two previously described approaches and incorporate techniques from the field of intrusion detection and even bio-inspired artificial immune systems. For example, Waelchli [33] designed a Distributed Event Localization and Tracking Algorithm (DELTA) that is based on a small short term memory to temporarily buffer normal state. The algorithm was deployed as part of a light sensor office surveillance with the goal of detecting and tracking persons carrying flashlights. DELTA provides the leader node with the information needed to localize and classify an event based on a Simplex Downhill algorithm.

Model-based This algorithm is used by 24% of the related work considered. Model-based event detection systems try to model the reality. With mathematical definitions, map-based approaches or rule collections are typically provided for specific applications. As such solutions are mostly very application specific, a good transferability to other applications or problems is not provided. Nevertheless, a model is capable of handling complex events if experts in the event domain add knowledge to the model. Alternatively, an automated training system – as introduced with this theses – that allows training the introduced model or maps to different applications can support a high transferability of the underlying concept.

As models tend to describe events in great depth in order to be precise, they are often complex and demand a lot of computational power from the detecting system. As the wireless SNs' computational power is restricted, SNs are a challenging choice for model based event detection systems.

Reducing the complexity of the model is an appropriate method to bring models into embedded systems, with the downside being reduced detection accuracy.

For specific event detection problems, such model base algorithms may be a good solution, but typically, their applicability is restricted to the chosen application's problem.

Fuzzy Logic This algorithm type is used by 13% of the related work considered. Fuzzy logic-based approaches tolerate unreliable and imprecise sensor readings and map them with membership functions into states like *warm*, *cold* or *hot*. As an advantage, these states are more relatable for humans and come with a rather small effort as they tend to be easier to implement.

When using fuzzy logic, it is known that the rule-base grows exponentially with increasing numbers of input variables. Reducing the number of rules has a direct influence on the applicability as a reduced ruleset decreases the probability of getting useful detection results for a given input. As SNs have limited memory available, storing larger rulesets may cause problems in real world deployment scenarios. In particular, additional spatial and temporal definitions that are included in the detection system increase the number of rules. A large rule set reduces the reactivity of a detection system as it has to constantly pass through the whole ruleset, which has a negative impact on the applicability [17].

Pattern Recognition 21% of the investigated research approach applied a pattern recognition approach. In WSNs, it is especially important to combine information from multiple SNs in order to describe and reflect complex events while having a low communication load. Pattern recognition approaches typically use multiple features (characteristic descriptions) of the events. In addition, it is possible to train a classifier to create reference data sets that represent a class or event type. Pattern Recognition allows reducing the communication to a minimum by using these features and offers a good compromise between accuracy and data compression with scalable and individually suitable features. The number and mathematical complexity of these features has to be determined for each event detection scenario, but in contrast to simple threshold and anomaly based algorithms, pattern recognition is more precise; in contrast to model based and fuzzy logic approaches it is much less computationally and memory demanding.

Typically, pattern recognition is performed in multiple subsequent steps that can be carried out by independent parts of the whole detection system. During the *sampling* process, raw data is gathered, optionally preprocessed, and handed over to the *segmentation process*, which detects the start and the end of the samples that belong to an event. These event-based chunks of data are then sent to a feature creation component in order to create the most descriptive features from the gathered data. All created features form a feature vector for that data chunk which is labeled after the data gathering device or region. Features can vary from descriptive attributes of the data such as spectrum features (FFT), orientation features (magnetometer based information), energy features (intensity), or statistical features (histogram, minimum, the maximum). In the final step, the *classification* decides which class has to be assigned to an unknown event by statistically analyzing or comparing the event features to either previously trained or fixed features such as threshold values. In most cases, prior training is necessary to deliver a sufficient set of training data that initializes the classifier. The following systems presented in Section 2.1 of this chapter make use of this algorithm type:[15,20,30,34,41,42,44,52]. Only [41] provide a training component for their system that is simulated and tested in a real sensor network.

All pattern recognition based systems mentioned create a decision based on the classification

but none of these systems uses features from multiple affected nodes in order to fuse them within a common feature vector to benefit from the distributed WSN knowledge of the arising event as proposed in our work.

Hierarchical Pattern Recognition A Hierarchical Pattern Recognition is performed on multiple tiers consecutively, e.g. sensor-level, node-level, cluster-level, and sink-level by 5% of the investigated papers.

The related work [45, 46] that implements this event detection algorithm either uses a final majority vote at the BS – which reduces the credibility of the results if multiple complex events are to be detected – or implements an off-line algorithm[46] for the evaluation. As introduced, cluster based approaches typically add a significant amount of communication to the existing communication load, hence these approaches have to be investigated in detail as to whether they make sense for specific approaches. A general framework approach that helps to cover multiple applications seems not to be feasible with a multi-tier fusion algorithm.

		Event Requirements																	References										
		# of Events		Information Fusion			WSN Scale				Realization				Applicability			Algorithm											
		Single Event	Multiple	Raw Data	Decision/State	Classification	Feature	None	One Sensor Node	Small (2-10)	Mid (11-39)	Large (>39)	Concept	Simulation	Offline Algorithm	Laboratory Test	Real World Test	Theoretic		Specific Application	Domain	Trainable	Threshold Based	Anomaly Detection	Model Based	Fuzzy Based	Pattern Recognition	Hierarchical PR	
Event Detection Architectures	Local	x			x						35						x		x								[14]		
			2			x			x									x		x							x	[15]	
		x			x					x						x				x					x			[17]	
		x			x				x									x							x			[16]	
			x		x						2						x				x		x						[19]
	Centralized		x				x	x				10				x			x								x	[20]	
			x	x				x							x				x						x			[21]	
			x		x										x				x				x					[22]	
				x	x							300				x				x					x				[27]
			x		x				x				35			x				x				x					[23]
				x	x									35					x						x				[25]
				x	x							4								x				x					[26]
	Decentralized		x										50					x										[24]	
			x			x						5		50				x										[33]	
				x	x	x							5					x										[34]	
				x									7						x										[36]
				10									7																[37]
				2		x							30						x										[30]
	Distributed											4																[32]	
												4																[31]	
			x					x					27					x										[38]	
	Local + Centralized		x										200					x										[39]	
			x											200					x									[40]	
																												[42]	
	Local + Decentralized		x	12			x					8							x									[43]	
				x								10																	[41]
				x									52																[44]
			x											75															[35]
				2										200															[44]
													6																[45]
			x										6															[46]	
			x											6														[47]	
			3																									[48]	
			x																									[49]	
Decentralized + Centralized												4																[52]	
		x																									[50]		
																												[51]	
																												[49]	
																												[52]	
																												[50]	
																												[53]	

Figure 2.2.: Related work organized by event detection architecture and requirements.

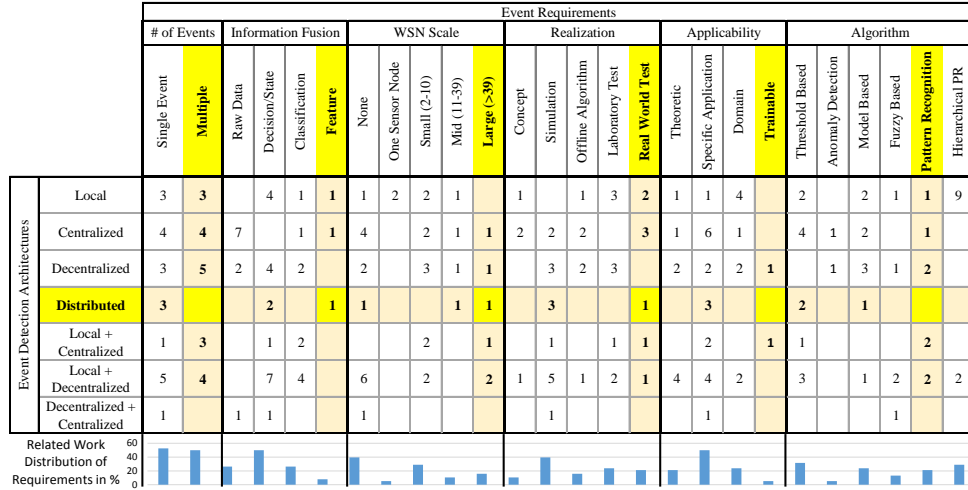


Figure 2.3.: Compressed visualization of all related work papers investigated by summing up the requirements covered within the corresponding architecture type.

2.3 Conclusions about the Current State of Related Work

By analyzing the given related work using the taxonomic classification of table in Figure 2.2, we can resume a good overview of the existing state of the art including the deficits in the existing research for event detection in WSNs.

In general we can conclude that all papers agree with the following statement. The strength of WSNs is the opportunity to make use of hundreds of SNs in order to cover e.g. huge areas to observe. The weakness of WSNs is that every packet which needs to be sent back to the base station costs energy at every single hop and depletes the energy of SNs and important RNs. This weakness increases with growing network sizes as the number of necessary hops also increases.

We derive the following general statements from our discussion of the WSN research domain:

- Theoretical considerations and simulations are mainly used in research but provide abstract knowledge.
- Deployment driven research is required for a sound understanding of the WSNs in their applications.
- A new distributed approach for in-network evaluation and observation of events should be accurate and energy efficient.
- The challenge is to create an approach with a minimized communication concept that performs an event detection which preserves the global event knowledge for the information fusion.

In particular, a complete implementation for an autonomous in-network event detection system including the (i) embedded software, (ii) embedded hardware, and (iii) casing has not been developed, deployed and studied for real world scenarios so far by any of the research considered.

Ideally, an event detection system for WSNs should benefit from and support the main advantages of WSNs:

- Distributed Computational Power

- Wide-Ranging Applicability
- Easy Deployment

In the related work, we see only a very small minority of implementations [38–40] that make use of a truly distributed approach as defined in section 2.1.4. The advantage is that a distributed approach uses highly descriptive information from multiple SNs to evaluate them within the network. It allows any given SN to evaluate the events without a predefined cluster-head or the need for any additional infrastructure like a WSN. None of the systems mentioned above uses features from multiple affected nodes in order to fuse them within a common feature vector to benefit from the distributed WSN knowledge of the arising event as proposed in our work.

In [54], Wang et al. give an overview of current energy aware WSNs with the goal of collecting data and to lengthen lifetime in parallel. They confirm that the typical centralized approach for data collection does not fit the needs of WSNs. Typical duty-cycle-based approaches collect data in rounds or try to outsource communication into or over the network.

We postulate that the need for collecting and evaluating a huge amount of data can be reduced for many applications by using the in-networking collaboration that allows performing a distributed event detection. If we prevent the system from sending raw data through the whole network to a sink, the WSN can be leveraged to multiple new applications without running into early energy balancing problems.

We conclude that a true *Distributed Event Detection Architecture* as defined in section 2.1.4 is necessary to create an effective event detection system in order to benefit from the distributed knowledge of a WSN in its full extension.

In addition to the architectural recommendation, we conclude that to the best of our knowledge, none of the related work mentioned above includes all of the subsequently described manifestations of the event detection system requirements introduced above (please refer to 2.2) that are essential to create and validate an event detection system in relation to its applicability, reliability, and accuracy. The compressed visualization of the related work is depicted in Figure 2.3 and shows the number of research works for each architecture type and highlights with yellow bars that no area of architecture types covers all recommended event detection system requirements as follows:

- # Of Events: The event detection system should be proven to be able to distinguish multiple events as an event detection for only one single event type cannot cover a scalable approach that addresses our research goal to cover multiple and diverse applications.
- Fusion Type: The related work investigated rarely applies feature fusion. Nevertheless, feature fusion is highly recommended because it is very effective in WSNs, as it compresses raw data to highly descriptive representatives that can easily be distributed within few packets in a distributed architecture [55].
- WSN Scale: Communication, longevity, and routing algorithms should be investigated in real WSNs of a size of more than 5 to 10 real SNs to reveal problems within and caused by the network communication as well as environmental influences.
- Realization: Fully performed field tests under real world conditions are necessary to prove the concept, functionality, lifetime, and reliability of the event detection [56,57]. The deployments can benefit from simulations and laboratory tests.

- **Applicability:** To allow an event detection system to be adapted to different applications, it should be trainable or able to use reference or historical data as a training set or setup input.
- **Detection Algorithm:** Pattern Recognition is a highly recommended event detection algorithm for **WSNs**. Through the distributed computational power of **WSNs** used for calculating local descriptions, the communication costs can be greatly reduced, while the event detection precision is leveraged by the fusion of these locally calculated features and subsequent classification based on the distributively calculated features.

CHAPTER 3

Pattern Recognition To WSN Transition

“Pattern recognition—the act of taking in raw data and making an action based on the category of the pattern—has been crucial for our survival, and over the past tens of millions of years we have evolved highly sophisticated neural and cognitive systems for such tasks” [10].

Pattern recognition is obviously a very important human skill which we try to adapt in technical solutions in order to benefit from automated and additionally utilizable extensions to our senses. Pattern recognition in the context of computer science is to be described as the aggregation of characteristics in order to detect similarities to known patterns within noisy and unstructured raw data while the current investigated pattern is unknown and needs to be classified. Often, we use the word *event* as a substitute for class in this thesis. Events are incidents in our environment and may range from a specific movement of a leg that indicates that you are running, through fire or intrusion events, up to earthquakes or flood events. Events investigated by WSN researchers in numerous projects are listed within Chapter 2.

In order to detect an event, a WSN has to identify which application-specific incident has occurred based on the raw data gathered by individual SNs. In this context, an event may be anything from a malfunction of monitored machinery to an intrusion into a restricted area. The goal is to provide high-accuracy event detection at minimal energy cost in order to maximize network lifetime.

“Given some examples of complex signals and the correct decisions for them, make decisions automatically for a stream of future examples” [58].

A classification problem can be defined as a non-empty set $P_{p \times f}$ composed of p patterns with f features for each pattern and a set of classes C_c with c classes. It is worth mentioning that not every pattern needs to be described by the same features or number of features. A classified pattern can be defined as a tuple (x, y) , $x \in P_{p \times f}$, and $y \in C_c$. A representative training set T with known classes for each pattern is used to assign an unknown pattern to a class $y \in C_c$. The classification algorithm $f : P \rightarrow C$ uses the *classification model* which is based on T in order to assign the patterns to a class [59].

We want to give a brief overview to *Pattern Recognition* [10] and how the classical pattern recognition terminology is mapped to our research domain of WSNs. In addition, we want to give essential and initial reason for major design decisions we made in order to make WSN reasonable applicable for pattern recognition problems, based on WSN specific constraints like limited memory or energy.

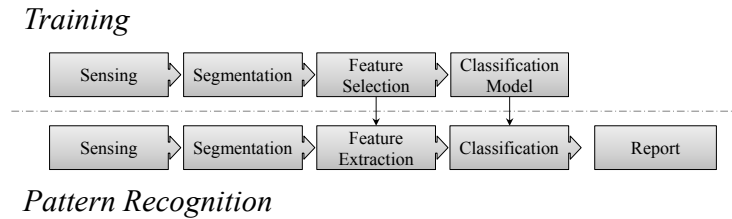


Figure 3.1.: Standard training and pattern recognition scheme adapted from [10]

First and foremost we want to assess distributively observed events within our environment, which means for applications that we want to detect specific events by using a [WSN](#). From the related work in Chapter 2 we derived that pattern recognition approaches are suitable in order to detect and classify patterns of events with a [WSN](#) because we can train certain events, which means that we can use prior knowledge of event characteristics that describe events in order to specify the pattern to be detected.

Pattern recognition mainly comprises sensing, segmentation, feature extraction, and classification as depicted in Figure 3.1.

3.1 Sensing & Segmentation

Sensing allows the transfer of real world metrics into a digitally processable time series. Sensors vary from cameras, microphones, Passive Infrared ([PIR](#)) sensors, and inertial sensors to pressure, temperature, and gas recognition sensors. Typical digital sensors have an integrated Analog-to-Digital Converter ([ADC](#)) in order to map the analogously recognized information to a digital representation. Analog sensors can be found for photocells or temperature sensors. These types of sensors need an additional [ADC](#) on the used [SN](#) in order to map the sensor readings, which are represented by a specific amount of current or ampere, into a defined range of digital values – so-called discrete values.

In this thesis we use the term *sampling* as an equivalent for sensing, as we combine with each sensor the ability to gather a distinct number of samples within a certain amount of time, represented by the *sample rate* f which is expressed by the *sampling frequency* Hz.

The sensor limits the applicability of the sensing system to the specification and configuration of the sensor itself. This means that the settings have to meet the requirements of the application and the given circumstances of the real world. Measurements depend on temperature, pressure, distance to the event, and the necessary accuracy, precision, resolution, and sensitivity that is supported and configured by the sensor itself.

A pattern needs to be separated from other patterns and from background noise. This isolation or separation process is called *segmentation*. The segmentation process itself depends on the application and sensors used. Some solutions use a predefined segmentation setup where the number of segments is static, while other approaches use a more dynamic segmentation approach that is based on the the dynamic of the pattern process [60].

3.2 Feature Extraction

Features are very important for the classification process. As stated in [10], a perfect feature reduces the importance of a classifier, while a perfect classifier is quite independent from features.

Typically, features are used to characterize the events using their relevant properties. A feature in this thesis is a numeric representation that describes a pattern using one type of significant information that can be calculated by an algorithm. Features are typically extracted from signals, while signals are representations of raw data. A so-called feature pool represents all available features that can be used for a classification problem. Typically, a feature pool can be easily extended with new features. We go into more detail in Section 4.2.3.3 but want to highlight that good features are very descriptive, which means that they ideally discriminate clearly between different classes. In addition, features are representations of complex event coherences that are reduced to mostly one simple number. The small amount of memory that is needed to store a feature combined with the given expressiveness can support WSNs to become an efficient event detection system if used in a cooperative way. As the energy consumption depends on the quantity of data (more precisely: on the number of packets) to be transmitted, a small description of an event can be very helpful. The calculation of a feature should be verified to be feasible on the chosen embedded hardware in order to not overcharge the given hardware.

3.3 Classification

The classification (see Section 3.5.2) is the core process that finally decides which class is with the highest probability the correct representation for the unknown event. The classification needs to be initialized with a classification model which is mostly based on features.

In general we can differentiate between two types of classification approaches, the *statistical classification* and the *syntactical classification* [10].

While the statistical pattern recognition is very common, the syntactical approach is very barely used and not very popular, as stated in [11]. Statistical pattern recognition does not try to represent a pattern or event with a feature vector but by formal grammar. Nevertheless, in certain applications such as in the assessment of human motions and creating transcripts of a motion sequence it makes sense to create a syntax [30, 60]. Syntactical pattern recognition is able to match a continuous stream of characters against a defined string. As mathematical operations are not applicable on a formal grammar, syntactical pattern recognition has the drawback that the classification results are not very easy to post-process as soon as a syntax is extracted. A typical distant measure like the *Levenshtein Distance* [61] is commonly used; other distance measures for syntactical classification can be found in [62].

Statistical pattern recognition tries to find the most probable class for a given pattern with a numeric solution. Typical and very common groups of approaches to reach that goal are *Frequentist procedures* [10], *Bayesian procedures* [63], as well as the most commonly used approaches from the huge group of *Linear classifiers* [64]. All of the procedures mentioned above use a reference (or prototype) feature vector for every learned class which is typically generated by calculating the mean of each feature type of a feature vector that is assigned to one distinct class.

If an unknown pattern has to be classified, the distance to each reference feature vector is calculated in order to find the shortest distance to the class that fits best (w.g. Euclidean Distance (ED) or Mahalanobis Distance (MD) (see Section 3.3.2)). This thesis follows the statistical pattern

recognition approach with a mathematically applicable distance measure that returns numeric distances and leaves the door open for future mathematical post-processing approaches as the quality estimation of features or raw data as suggested in [65].

3.3.1 Evaluation Metrics

The following evaluation metrics are used to evaluate the system's classification reliability while exposing the Distributed Event Detection system to different classes of events in multiple applications. We reach a good comparability between all scenarios, applications and experiments by constantly using these evaluations metrics.

We evaluate the classification results with the well known binary classification types *true positive*, *true negative*, *false positive*, and *false negative* as introduced below:

- TP = A *true positive* describes a correctly identified event. E.g. the event *climb over the fence* was performed and detected.
- TN = A *true negative* describes a correctly rejected event. E.g. the event *climb over the fence* was not performed and it was detected that this event did not happen.
- FP = A *false positive* describes an incorrectly identified event. E.g. the event *kick the fence* arose but the climb over the fence event has been detected.
- FN = A *false negative* describes an incorrectly rejected event. E.g. the event *climb over the fence* was performed but classified as not to be that event.

We use the generally known and established statistical measures (sensitivity, specificity, positive prediction value, negative prediction value) of the performance of a binary classification test as our main significant evaluation metric:

- Sensitivity = $\frac{\#TP}{\#TP+\#FN}$, also called *recall*, corresponds to the proportion of correctly detected events.
- Specificity = $\frac{\#TN}{\#TN+\#FP}$, corresponds to the proportion of correctly ignored events.
- Positive Predictive Value (PPV) = $\frac{\#TP}{\#TP+\#FP}$, also referred to as *precision*, corresponds to the probability that detecting an event reflects the fact that the system was exposed to that matching event.
- Negative Predictive Value (NPV) = $\frac{\#TN}{\#TN+\#FN}$, corresponds to the probability that ignoring or rejecting an event reflects the fact that the system was not exposed to that matching event.
- Accuracy = $\frac{\#TP+\#TN}{\#TP+\#TN+\#FP+\#FN}$, corresponds to the proportion of true results in the population, i.e., the sum of all correctly detected and all correctly ignored events.

3.3.2 Distance Measures

Each feature we use to describe a classification problem is mapped to its according numerical dimension which increases the dimensionality of the class to be classified. These dimensions are represented by \mathbb{R}^n in which for each class and a feature vector \vec{v}_f comprising multiple features f_i is defined, see Equation (3.3.2).

We can simply imagine the position of a television chair within a 3D area of a living room as such a vector. The position of that chair is represented by three features, the x, y and z coordinate within this specific room. The pattern recognition now postulates that similar problems (in the case of the position of the television chair with our room) are represented by the features with more or less the same numeric values.

We can transfer the example of the chair position to a mathematical expression called *spatial proximity* measured by *distance measure*. If the chair is not exactly placed at the *initial position* - class we can now measure the distance of the new position within the defined and by this evaluate how dissimilar the position of chair is compared to its initial position.

$$\vec{v}_f = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \in \mathbb{R}^n \quad (3.1)$$

The distant measure is an important part of the pattern recognition process as it allows to quantify how dissimilar a newly detected class is in comparison to the trained classes. “*The choice of a particular metric depends on the application. Computational considerations aside, for feature selection and extraction purposes you would choose the metric that gives the best performance (perhaps in terms of classification error on a validation set)*” [12].

The **ED** and **MD** are two of the most important and most used distance measures. These metrics suit our needs; as they are implemented for various classification problems and are often cited to be surprisingly well performing we use them for our preliminary investigation. In general, the proposed event detection system should be evaluated with a distance measure that suits the general needs of a **WSN** and is accepted as a well performing distance measure for most classification problems.

Nevertheless, these distance measures are different in their main purpose: Euclidean is used when the correlation of patterns is low, while Mahalanobis is preferred for patterns with medium or high correlation. We can assume the worst case for our real world applications, which means that our patterns will not correlate well as they are performed by humans, who tend to repeat motions in a highly correlated way only if they are used to them through intense training. However, the correct decision on the selection of the distance measure depends on the available data and the statistical distribution of the features [66]. While having these theoretical statements in mind, we investigate the distance measure in order to decide which to prefer in Section 3.3.2.

3.3.3 Classifier

We concisely introduce three of the most known classifiers and discuss whether they can be used for **WSN** classification problems.

3.3.3.1 Bayesian Classifier

The Naive Bayes classifier is a very successful classifier and competes with very sophisticated approaches [67]. The Bayesian classifier assigns a class with the maximum probability – instead of the closest physical distance – to a given event described by its feature vector. This classifier is very effective in many practical applications, such as text classification, medical diagnosis, and systems performance management, because of a common simplification called *conditional independence*

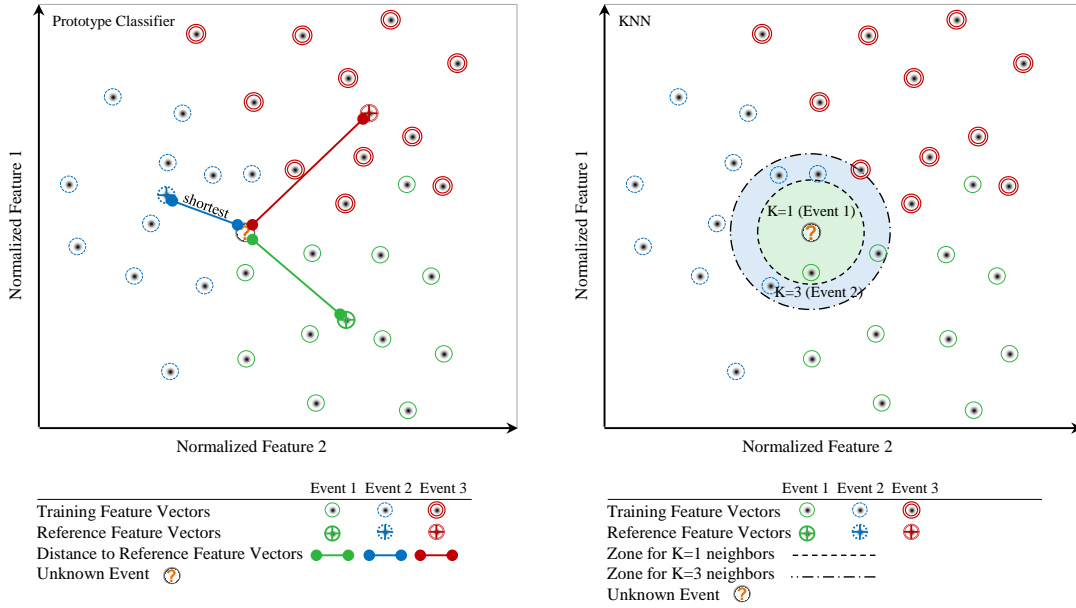


Figure 3.2.: Example depiction of the nearest prototype classifier and the K nearest neighbor classifier (KNN).

assumption that assumes that features are independent of each other given the class, that is:

$$P(X|C) = \prod_{i=1}^n P(X_i|C) \quad (3.2)$$

where $X = (X_1, \dots, X_n)$ is a feature vector and C is a class [67].

For our classification problems, the a priori distribution is unknown, which leads to the simple solution of either using historical data to construct an a priori distribution, or assuming that all of the measured values have equal probability [68]. In our scenarios, it makes more sense to assume that events are all equally probable, which can be easily seen if we think about a training session where different motions have to be performed. Because of its simplicity and lightweight implementation, the Bayesian classifier is a good candidate for our distributed event detection system evaluation to compare different applications.

3.3.3.2 KNN

The K Nearest Neighbor (**KNN**) introduced in [63, 69] uses a simple but very effective approach to assign an unknown event to its most comparable training data (neighbor) if K equals 1. In any other cases where K is increased, the **KNN** works as a majority vote, where K is the amount of comparable training data (neighbors) of one class that have to be more comparable towards the unknown event in contrast to all other training data in the neighborhood. More comparable training data means the training data is positioned in a closer neighborhood compared to other training data. Depending on the distance between the unknown event and the training data, it might be more effective to weight the final decision with these distances rather than counting the numbers of effectively the closest neighbors. Through this, every neighbor is able to contribute to the vote by the fact of its own proximity in an averaging way.

The original algorithm starts by finding the closest neighbors to the unknown event and classi-

ifying the unknown event as the class for which the first K closest neighbors can be found; see a sample depiction of the **KNN** approach in Figure 3.2 (right hand).

It is important to have a large K in order to minimize the probability of a non-Bayes decision for the unknown event. In contrast, K should be small compared to the amount of training data. It is important that the training data are close enough to the unknown event to guarantee an accurate estimate of the posterior probabilities of correct classification. Hence, the 1-NN approach is an option especially for the smaller training data sets we are using as suggested in [69].

From a technical point of view, the **KNN** has a slight disadvantage for embedded systems as it needs to store all training data on the node to calculate the K nearest neighbors.

3.3.3.3 Prototype Classifier

Classification algorithms offer a very broad functionality and applicability, but the majority of applications make use of decision trees, neural networks, support vector machines, and K nearest neighbor approaches [10].

The Distributed Event Detection system allows exchanging the classifier in order to support a vast majority of applications, but for evaluation purposes we constantly deployed different applications with one classifier with the aim to compare the results with each other. As wireless **SNs** are limited in resources like memory energy and processing power, we decided in [55,70] to use the classification process employing the prototype modeler [71] within our distributed event detection system as further discussed in Section 5.6.3.

For a better understanding of the ongoing work, we want to introduce the classifier used with a simple description early in the thesis. Figure 3.2 (left hand) depicts an example of the prototype classification based on three trained classes based on two features which represent a simple two-dimensional feature space.

Three so called *Prototype Vectors* represent the trained classes. They have been calculated by averaging the selected feature types of all training data corresponding to their specific class. A new and unknown event represented by a feature vector is classified by calculating all distances to all prototype vectors. As previously mentioned, the **ED** is used, but other distance measures can be applied, such as Manhattan distance or **MD**. The shortest distance of all calculated distances indicates that the features of the unknown event are mostly comparable to the class with the shortest distance. All other classes are assessed as a non relevant option.

3.4 Report

Reporting an event is an essential task for **WSNs** as this task plays a key role in energy efficiency. Especially if the report can be minimized by filtering routines a **WSN** benefits of the saved communication load. In the current thesis, we focus on reducing the communication with the **BS** and ideally to omit unnecessary reports if possible. A solution to reducing reports is to introduce e.g. event categories with varying relevance. The report is then only sent in case of a relevant event. Further, time dependency or other dependencies can be added to reduce the reports. A key solution comprehensively discussed in Chapter 4 is the collaborative in-network evaluation of multiple **SNs'** features, as this enables a single classification result that reduces the communication with the **BS** to a minimum. The increased in-network communication and the trade-off between both communication types is discussed in Chapter 5.

3.5 Supervised Training

Supervised training is a very common technique to assign training data to a specific class after each training run in order to add the knowledge of the trained classes to the classifier by the supervisor.

The training allows us to collect relevant and representative data representation of events we want to detect or do not want to detect in the near future by a pattern recognition system as well as by our distributed event detection system. The training goal is to learn from numerous event examples the currently available characteristics in order to preserve them and to load them as model knowledge into the later recognition system. Therefore, the execution of the training has a huge influence on the classification model as any variance or effects that occur during training because of a training situation are stored within the reference data.

As a consequence, the supervisor of the training has to observe the training progress very carefully and has to annotate every training case as to whether certain training samples are faulty, useful, have obvious issues, or lack intensity which may be caused by the training-related bodily fatigue if a sound physical effort is necessary to perform the training. Event detection may suffer from the uniformity of the training data and can benefit from a more exhaustive training. We can see that the effort of a training stays in a clear relationship to its effort, but in general it can be said that a more comprehensive training leads to a better classification system. This goal can be reached by collecting a preferably large set of training instances for each class in order to filter outliers more successfully and to form a better approximation of the true event characteristics.

The support for event-specific training can not be recommended for all kinds of event detection applications. It may in fact be preferable to classify events based on expert knowledge created from historical data rather than a generic classifier using training data from example events. This is especially true for scenarios in which training data for all types of events is very hard to generate, such as earthquake detection, flood detection, volcanic eruption detection, or tsunami early warning systems. In summary, the goal of a learning algorithm should be a concept or a general description of a class of objects [55,68].

For WSNs, we have to store raw training data on the SNs and need to add a simple communication packet that tags training data with a class or deletes the training data directly on the SNs. Another option is to send all raw data to the sink after each training run and to tag the training at the BS. Which approach is preferable depends on the number of SNs.

3.5.1 Feature Selection

Feature extraction is the initial step to create meaningful characteristics of a pattern and to be able to compare them to a priori learned details about all known pattern. Proper feature types can be extracted from various signals and can create a huge feature pool ranging from Fourier based features through intensity, histogram, energy, or peaks, which is just a small fraction all possible features.

As the number of features can increase significantly, it is important to select those that are most descriptive and to remove the irrelevant or even redundant or correlated features with the goal to reduce the necessary feature space without reducing the detail of a class description [72]. For WSNs too many features would increase the need for memory and the calculation time to an extent that would reduce the operability. Hence, a highly descriptive and compressed feature vector that fits into a single communicated data packet is the main goal for the feature selection.

A method that is typically used to find a good feature subset is called *filter method* [73, 74], which considers the features independently from the classifier and calculates general parameters of the training set to select features that perform best within this abstract statistical approach. The first approach that implemented a filter method for the Distributed Event Detection is made in the context of our Fence Surveillance project as well, see [75].

To compress the resulting feature vectors, a well-established method to perform a promising feature selection is the so-called *wrapper method* [74], which is used and introduced in detail in this dissertation in Section 4.2.3.6. The first approach that implemented a wrapper method for the Distributed Event Detection is made in the context of our Fence Surveillance project as well, see [76].

In general, the wrapper method uses a predictive model based on the classifier itself (also called *induction algorithm* [74]) which is used within the deployed detection system to assess features and combinations of the features in order to find the best feature subset. With every feature combination, the classifier is trained and tested against a specific training data set. The number of correctly and wrongly classified training sets is used to assess the current combination and to decide how to continue to optimize this specific combination. As the wrapper method iteratively trains the classifier with each feature combination, this approach has a very high computational effort and requires a high performance computer. Nevertheless, wrapper methods are superior to the filter methods as they deliver the best results if they operate with reference feature vectors [74] just like the proposed Distributed Event Detection system.

Although the computational effort of the filter method is very low compared to the wrapper method, the resulting feature set is not tailored to the classifier used in the investigated approach. Typically, this leads to a higher quality output of the wrapper method. Even for embedded systems, the wrapper method is a viable approach, as the training is performed in a processing step that takes place before the real world application with the application of wireless SNs for the purpose of event detection.

3.5.2 Classification Model Generation

As mentioned in Chapter 3, the classification algorithm $f : P \rightarrow C$ uses the *classification model* which is based on the training set T in order to assign the patterns to a class. In order to create such a classification model, an abstract description of classes is needed. Multiple approaches to create such an abstract description can be found in [77], where they are called knowledge representation. Duda et. al [10] uses the word *model*. A classification model is generally created using training data, as mentioned above features can represent an abstract version of these data.

For our case of WSNs, a matrix for the relative SN neighborhood or physical topology is necessary, the number of physical dimensions used for the SN positions, the maximum number of affected SNs, and additional information like CN-Features to find the node closest to the event center. A more in-depth discussion and the complete list of model parameters is introduced in Section 4.2.3.8.

This classification model is then used in the recognition process. If the complete training data set is used to support the classification model, it represents the most precise data basis. As a downside, the complete training data set causes storage and computational bottle necks on each SN. Hence, a reduced and optimized model that relies on features and additional network specific parameters as described above is highly recommended.

CHAPTER 4

Distributed Event Detection System

4.1 Distributed Event Detection Concept

Our main goal is to detect events within the network while simultaneously reducing the amount of data that needs to be transferred to a control center.

As stated in [78], WSNs are able to reduce the communication by implementing the following main concepts:

- In-Network Processing: data is evaluated within the network
- Data Compression: data is compressed and evaluated at a dedicated station
- Data Prediction: predicts arising events based on data and models

A feasible data prediction is not suitable for the proposed system as such an approach is suitable for long-term forecast events types like meteorological or seismic events. In contrast, we want to distinguish instantaneous critical events from uncritical events in a fence or bridge surveillance as well as a body motion-based scenario. Our goal is to solve the classification problem of distinguishing trained events within a WSN. We want to solve the given problem by combining in-network processing and data compression. The in-network processing solution uses a feature distributing classification system that compresses data in form of a characteristic data representation – the features – that reduce the necessary event notification to one single event classification that can be as small as one single byte.

As introduced in [5], our proposed in-network approach is able to involve observations of multiple SNs to obtain an extended and holistic view of an event. In principle, it observes events from different perspectives with SNs located at different positions at e.g. a fence, a bridge or a body to complement one another and to get an extensive view of the observed event. We can summarize that the introduced Distributed Event Detection System benefits either from events that spread spacially or from events consisting of multiple distinct information depending on the SN's location or perspective.

An integrated classification model is used to define a proper combination of the trained and diverse event descriptions. To develop a proper event detection classification algorithm [10] for WSNs, it is important to take the required simplicity and the accumulating protocol overhead into account. The design and evaluation of application dependent *feature types* is a key component of

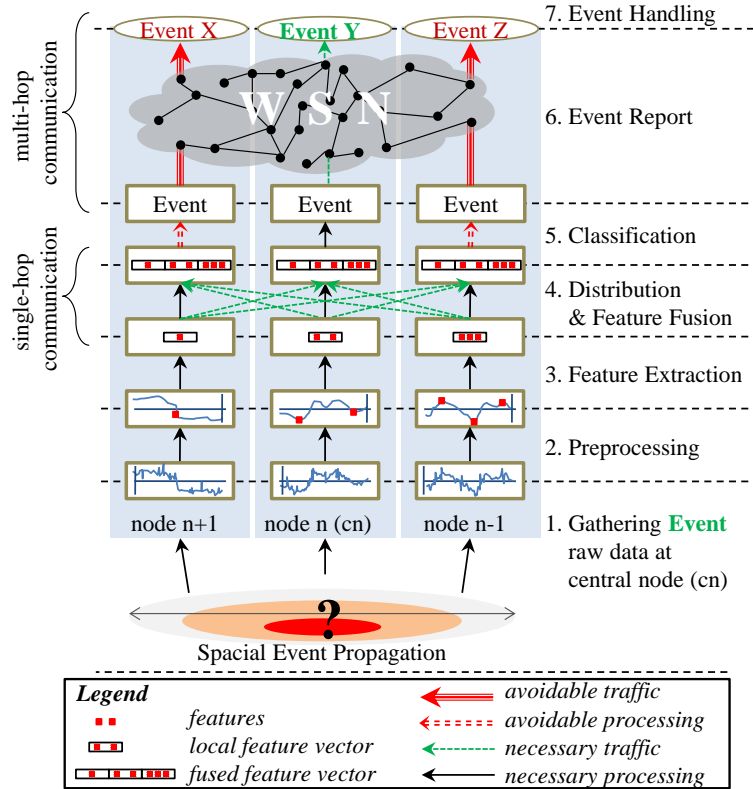


Figure 4.1.: Distributed event detection concept reduces traffic in multi-hop networks during event reporting [5].

our distributed event detection system. It allows us to transform the huge amount of arising raw data into very tiny and highly descriptive representations of specific event properties. In addition, they differentiate distinct event classes (high inter-class distance) while events of the same event class show similar characteristics concerning the corresponding feature values (short intra-class distance) [12]. A comprehensive introduction to the topic of classification and pattern recognition and data mining is given in Witten et al. [64]; additionally, an in-depth introduction to the subject of pattern recognition is available in [10, 11, 63].

As depicted in Figure 4.1, our Distributed Event Detection concept starts the event detection by gathering event related raw data from multiple SNs. It provides *preprocessing*, *feature extraction* of local event features, and *feature distribution* between the SNs, and finally *fuses* the features of the feature vectors on the SNs into one *Fused Feature Vector* to *classify* the unknown event with a classifier. In the last step, the events are reported to a control center. The unique selling point here is that we only report events through the multi-hop network from one single SN that performed the feature fusion beforehand. To provide minimal communication, only critical events are sent by filtering out all uncritical events within the network.

We have to be aware of the multi-hop communication as this communication influences the network lifetime and thus its applicability in the first place. Figure 4.1 shows that it is important to avoid to the processing task of the classification task at every affected SN. We recommend classifying the event only on a CN that is the only SN to send the classification result through the WSN. The avoidable communication and processing steps of all other SNs are highlighted with red arrows. Black and the green arrows indicate our proposed communication and processing

concept. The event handling has to deal with only one classification result, which helps to decide how to react. In contrast, multiple different events cause more confusion than a clear emergency for example.

Before deploying our system, it is important to pre-evaluate the effectiveness of the selected features. Descriptive features from several feature types such as intensity or histogram are used to distinguish different events from each other. Therefore, our main concept comprises two concrete frameworks which are described in detail in the following sections and which were introduced in their first iterations in [5, 76].

We introduce an *Evaluation Framework* for the supervised event training that performs the pre-evaluation of the training data and allows to extend the involved quantity and types of features of the feature pool to the application needs. The *Evaluation Framework* is responsible for the final classification model that includes the reference vectors. The creation of the classification model is based on iterative evaluations of the classification accuracy in theory within the *Evaluation Framework*.

Our deployment itself is driven by our *Distributed Event Detection Framework* that reflects the main architecture of the *Evaluation Framework* but is implemented on embedded SNs for deployment purposes. Compared to the *Evaluation Framework*, the *Distributive Event Detection Framework* is extended by a communication layer, a comprehensive event filtering process, and a report process. The distribution and feature fusion process fuses the available environmental data within the network to achieve an increased event detection accuracy. In contrast, systems with a local evaluation architecture lack the ability to view events in their entirety, systems using a decentralized architecture need to transmit all data to a dedicated CH which may cause huge traffic loads, and centralized architectures cause the most extensive traffic load as all data need to be transmitted to a BS.

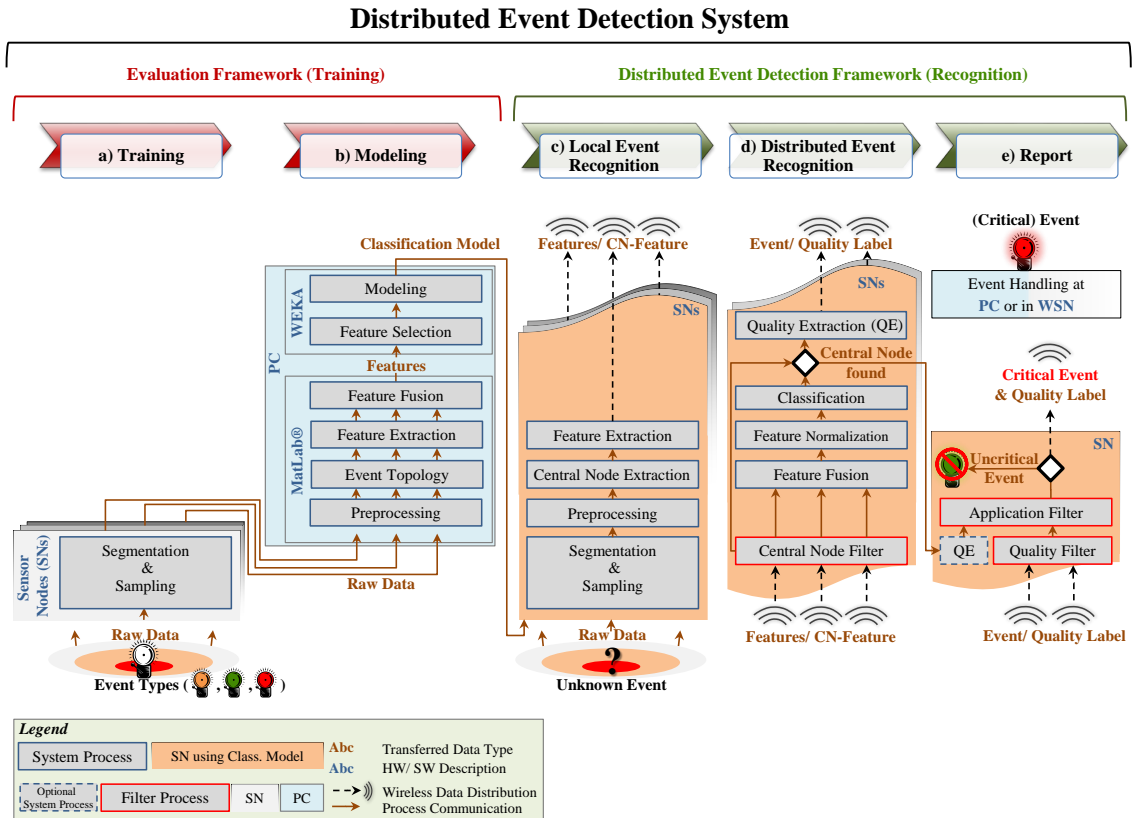


Figure 4.2.: Distributed Event Detection System, composed of the Evaluation and the Distributed Event Detection Framework, adapted from [5].

4.2 Evaluation Framework

As introduced in [5], the overall goal of the *Evaluation Framework* is to analyze the application dependent classification problem and to build up the classification model which specifies the classification problem precisely. A classification model can be understood as a compilation of abstract event descriptions that will be stored on the SNs for later comparison in case of an event occurrence. The framework allows to evaluate multiple classification algorithms with respect to the resulting classification accuracy. The *Evaluation Framework* comprises two initial process blocks, (a) *Training* and (b) *Modeling* (cf. Fig. 4.2) which are used to calculate a Classification Model including an optimal feature set from the given training events to set up the subsequent *Distributed Event Detection Framework*.

4.2.1 Calibration

A preceding calibration as introduced in [55] adjusts each sensor to the local environment and allows to auto-discover background noise within the observed environment. For example, a microphone sensor will discover noise while no acoustic event arises in order to create a threshold value (for threshold logic details please refer to Section 6.2) that feeds the event detection in order to segment the signal into event-specific time series of raw data chunks. An automated self-calibration is automatically initiated at any point in time during the deployment by the SN if necessary, e.g., after a temperature or magnetic field change which could cause a continuous drift in sampled values or an

increase in background noise while the automatic calibration routine takes interruptions during the calibration into account. However, sophisticated sensors often integrate auto-calibration methods which improve current application-dependent calibration processes.

4.2.2 Training

During the training process block (Fig. 4.2(a)), all events that shall be detected in future applications need to be trained multiple times with the SNs attached at the correct positions. It is necessary to gather the knowledge of the correct assignment of the training data to their corresponding event class for an optimal assessment of the features extracted later. Hence, the training is performed under supervision, which means that each trained event is known and labeled by the system using a training setup file. During the training it is not necessary to have an established wireless communication available, as all training data can be stored directly on the SNs for the subsequent event modeling. Nevertheless, a wireless communication helps to tag faulty training runs or to tag training runs with the supervising class names.

In order to cover the application's need, it is necessary to train all relevant events and all possible and known variants that can be expected. The training is a key process that has a huge influence on the resulting classification model quality. The evaluation Framework can create a highly reliable classification model only if all types of events are covered during the training, distinguishing the trained events from unknown events, and to assign arising variations of events to known classes most appropriately.

We save the training data on the SD card of the SN. For this purpose, a simple training ID including the class to be trained is sent to each SN. The subsequently stored event data are tagged with the previously stored identifier. In case of a malfunction or a faulty performed training data, we can delete this training data. The corresponding and incrementally maintained identifier can easily be identified by its absence in the later evaluation. As a result, subsequent feature selection is performed off-line. A more general overview into the supervised training and possible realizations is introduced in Section 3.5.

4.2.2.1 Sampling & Segmentation

All raw data are sampled during the *Sampling Process* and segmented into distinct events directly on the SNs using segmentation approaches depending on the application.

Typically, segments are created when physical effects indicate an event. The segmentation distinguishes whether sensor data is part of an event depending on the segmentation parameters discovered during the *calibration*. The segmentation can be performed with thresholds or with knowledge of defined changes in event-related environmental data. In combination with a hysteresis as introduced in [79], we can adjust the tolerances to detect event start and stop occurrences.

4.2.3 Modeling

The modeling process block (Fig. 4.2(b)) evaluates the classification problem by creating and evaluating different features within the evaluation framework. The feature evaluation is supported by the in-framework classification algorithm which supports short feature development cycles.

Acceleration Data					Frequency F(s)							
Raw Data					Normalized N(s)							
SIGNALS s	X	Y	Z	\vec{v}	N(X)	N(Y)	N(Z)	$N(\vec{v})$	N(F(X))	N(F(Y))	N(F(Z))	$N(F(\vec{v}))$

Figure 4.3.: Different signal types are derived from the raw data by applying a normalization, creating a magnitude signal and a frequency transformation, adapted from [5, 76]

4.2.3.1 Preprocessing - Signal Creation

The collected event segments are processed in MATLAB® on a PC (control center) to obtain feature data. The *Preprocessing Process* uses the generated raw data of a training event and preprocesses all sampled data points in order to remove noise from the data stream and to perform data transformation in order to calculate additional signals. The signal creation can lead to feature types combined from multiple different sensors built into the SN (multi-modal SNs).

Further, we need the opportunity to let the *Feature Selection* pick features that are invariant and therefore still meaningful in case of slight changes of events that are not relevant for the application. Example events that should be detected as the same event despite some variation within their performance could be two different intruders, one climbing over a fence slowly and the other quickly; obviously, both should be detected as a fence intrusion event regardless of the time taken to overcome the fence. For this purpose we need e.g. time independent features based on frequency signals. In the following we introduce all used signals we derive from our actual acceleration sensor raw data on the SNs. The extracted signals depend on the sensor type and can easily be expanded for future applications.

We can derive 12 different signals s from the raw data by applying normalization or Fourier transformation, by using the raw data streams as they are, or by combining multiple raw streams to a vector. Every feature that can be derived from these signals then reflects a specific event characteristic in various domains, defined by the signal used.

Raw Data s: We mainly use the acceleration sensor data for the evaluations within this thesis; *Acceleration Data* is given as the main input. This sensor type can be exchanged against any other sensor. We derive the main data types from the acceleration sensor: *Raw Data* which creates three initial signals s : X, Y, and Z, one signal for each axis of the 3D-acceleration sensor. Additional signals allow us to pick features out of multiple and diverse highly descriptive information representation.

A sensor fusion based on all axis allows us to derive the magnitude signal which is independent from the orientation of the acceleration and reflects the magnitude of all acceleration axes combined in one vector. As introduced in [5], every sampled triple of acceleration values (x, y, z) is represented in three-dimensional space as an acceleration vector \vec{v} . This enables the abstraction of every triple to one orientation independent acceleration value by extracting the magnitude of the vector $|\vec{v}|$. Features calculated from this signal are for example totally invariant to changes of the SN's alignment.

Normalized N(s): With the help of time and value normalization, we can create signals that are less dependent on time or value or both; for example, normalized axis values $N(X)$, $N(Y)$, $N(Z)$ or a normalized magnitude based features $N(|\vec{v}|)$ as represented in Figure 4.3. Every normalization process adds distortion as the original signal changes and artificial values maybe added to a signal if it needs to be normalized to a longer time series or a higher maximum value. The positive effect

is that a normalized signal can help to deliver an approximation of equalized event conditions.

The included feature-dependent normalization process provides optional linear scaling data normalization functions as described in [80] to obtain normalized signals concerning time and intensity, as well as the frequency domain representation of the original signal.

As a side effect, normalization changes the signal in a specific way, which can cause information loss. For example, null points will be lost, as will the relation of the three sensor axis of an acceleration sensor. For features that need these relations, such as a *null point feature* or the *orientation feature*, we preserve the raw data for non-normalized signals.

Typically both time and value normalization are performed; exceptions can be made depending on the needed feature characteristics. While time normalization scales every signal linearly to the same amount of samples, value normalization scales every signal linearly to a common maximum value range. As a result of the classification model, the *maximum signal values* are needed to repeat this normalization on signals that arise during the Distributed Event Detection in the application. For the final classification model, a defined maximum amplitude value to which the signal has to be normalized is used as a reference to calculate a linearly extrapolated value normalization. Additionally, the maximum event duration of all training data is used as a reference value for the time normalization. In the time domain, our events are normalized to 1,024 samples or 10s. These maximum values are used during the initial signal creation as well as preserved for the model creation in Section 4.2.3.8. It is worth to mention that the maximum amplitude value should not be the maximum data-type value, as these values need to be processed during the MD or ED calculation which would quickly exceed the maximum values. For our applications the defined maximum amplitude value is set to 1,000 which provides enough buffer for future developments in a 32-bit architecture.

Frequency F(s): In order to cover the mentioned frequency domain we use the FFT F(s) introduced by Cooley and Tukey [81] to transform the sampled data from the time domain to the frequency domain. The resulting signal comprises the ordered frequency distribution of the original signal with the corresponding amplitudes. This signal is invariant to variations of the event in the time domain. This means varying event lengths or time displacement of event characteristics do not affect features created from the frequency signal. For the FFT realization on the SN AVS-Extrem (introduced in Section 6.2.1) the implementation of Douglas L. Jones [82] is used as introduced in [83]. For the realization of the FFT on the SN F4VI2 (introduced in Section 6.2.2), the implementation of ARM Cortex Microcontroller Software Interface Standard (CMSIS) [84] is introduced in [85].

The signal normalization in the frequency domain is slightly different. The value normalization affects the amplitude of the frequency spectrum, and the time normalization is of course transformed into a frequency normalization, which always represents half of the sampling rate used in our experiments (100 Hz) during the event detection; hence, this frequency normalization results to a maximum of 50 Hz. The sampling rate needs to be constant for this calculations. This mapping ensures the comparability of the frequency domain representation of signals of different events. Thus, we gain three frequency signals based on the individual axis $N(F(X))$, $N(F(Y))$ and $N(F(Z))$. By combining all three axis into a vector we can also create a frequency based signal on this vector \vec{v} which is then defined as $N(F(|\vec{v}|))$.

4.2.3.2 Topology

We need to know the real world topology of the **SN** and the topology of the event distribution for the autonomously mapped position of the event into the real world network location by using a **CN-Feature**. We introduce both topology types, the **CN-Feature** and some subsequent application limitations.

Node Topology In order to take the **SN** placement into account with regards to the event detection, the physical node topology needs to be added to the classification model to support the automated calculation of the relative location of the **SNs** within the **WSN**. This topology can be represented by a list or matrix of neighborhood types and relations and needs to be added by the physical network topology called *node topology matrix*. Depending on the complexity of the network, we use multi-dimensional data types like matrices or linked lists to map the physical topology of the **SNs** into an abstract representation of relative node positions or neighbourhood relations to one another.

Every physical topology is supported as long as the topology does not change over time, which is necessary to derive the relative positions of **SNs** during the event detection within the **WSN** according to the supervised training.

The exact setup, order, and topology of the **SNs** of the training needs to be used as a mandatory deployment guideline for the placement of the **SNs** during the distributed event detection deployment. In the case of the example fence surveillance with circular and closed fence comprising 49 fence elements, we use a simple circular linked list that stores the 49 **SN** IDs in the same order as they are deployed in the fence.

Event Topology The additionally required *event topology* is known at the end of the training because of the supervised training that determines the training location which represents the **CN**. The **CN** represents the entry point for the relative position of neighboring nodes, which is for example the node closest to the event center. Our evaluated application examples use a linear (physical) network topology. The **SN** that is closest to an intruder climbing over a fence can be seen as a **CN** as the event oscillations spread out physically from this node to both sides into the fence.

The relative location n of the **CN** is used to derive all other node locations towards the **CN**'s location. Depending on the event spreading and distribution into the environment, the maximum number of nodes has to be figured out during the training. In particular, it is important to know how many **SN** will be affected in which direction or dimension.

In the case of the fence surveillance system, the application topology comprises two dimensions representing the spacial event propagation. The example *topology matrix* used in our later fence surveillance deployment comprises two affected **SNs** to the left side of the **CN** and three affected **SNs** affected on the right of the **CN**. This leads to the following example relative **SN** IDs: $cn - 2, cn - 1, cn, cn + 1, cn + 2, cn + 3$ as depicted in Figure 4.4, called *event topology matrix*. Other event detection applications may have more dimensions and a different number of **SNs** in each orientation. A representation of the **CN** and its relationship to other **SNs** is depicted in Figure 4.1.

CN-Feature As events occur at unpredictable locations and points in time, the location of the **CN** and the relative neighbor locations have to be determined dynamically and autonomously from the **WSN**. Only if the **CN** is successfully determined during the subsequent detection phase can the

features be fused in the same way as in the event training. For different events and applications, the *CN-Feature* has to be determined in order to reduce the probability of multiple *SNs* sending their classification results to a *BS*; hence, the *CN-Feature* helps to filter the dedicated *CN* with the best physical position for arising and unknown events to use the training information to fuse the calculated data of the neighboring *SNs* corresponding to the event topology matrix. This simply means that if events occur at different locations, the *CN* has to be selected autonomously according to the *CN-Feature*.

The *CN* is detectable by a dedicated but application dependent *CN-Feature*. For our example fence surveillance scenario, such a feature is represented by determining the *intensity* of the first bin (partition) of a fence event which locates the *SN*'s position relative to the excitation location of the occurring event. The *CN-Feature* is taken from the x-axis for both the fence and the bridge surveillance.

Application Limitations As a consequence, our approach's classification model needs to be updated if *SNs* are changing their relative position to each other, which may happen more often in Mobile Ad Hoc Network (*MANETs*) or situations of misplaced *SNs* because of a faulty setup. One solution could constantly use the same *CN* wherever possible. Some setup faults like upside-down placement or temperature changes can be corrected if an orientation sensor is available or a temperature compensation algorithm is implemented. However, the proposed system does not address the area of mobile ad hoc networks as the thesis is out of the scope of the area of research of *MANETs*.

We want to emphasize that e.g. the *SN* mounting needs to be consistent for all *SNs*. It should be omitted that *SN* change their standard mounting position. In addition, the *SN* mounting should be the same during the training and the deployment, as changes to the mounting may have a huge impact on the way events are recognized. Ideally, it should be ensured that all *SNs* can recognize events exactly in the same way as they did during training.

4.2.3.3 Feature Extraction

During the *feature extraction* process, all available features are extracted from a combination of the corresponding feature types and signals.

For our Distributed Event Detection System, we generate features for n *partitions* with k samples of the event to describe the event in progress. A partition generation fragments an event into a series of sorted event partitions in order to create features for different parts of the event such as the beginning, the middle section, and the end. Hence, each partition represents a chronological part of the event which is defined by a *binID*, where the partition size is freely configurable.

We decided to use 3 partitions for our deployments as this number turns out to deliver very stable and repeatable results. With a higher number of partitions, we would expect an increased accuracy of the reflected chronological event partition. This is true in theory and for a perfect event input signal. Perfect means that the event has the same duration and amplitude all the time. During the deployments we found that the ability to build partitions of slightly varying events is reduced if the number of bins exceeds 3. The reason for this lies in the fact that the bins are created at fixed sampling numbers. These concepts fail for a very detailed evaluation in case of input signals that are shifted slightly in the time domain or are performed at a different speed because of the performance variation. A new approach that falls out of the scope of this thesis is capable of solving this problem: instead of using a fixed sample number as a partition indicator,

a partition should be created using the signal characteristics, e.g. speed, orientations, brightness, noise, angles, or other characteristics. Such a partitioning approach is independent from time and amplitude variations in the event data as introduced for an example motion application in [60, 86].

The sampled values of sample i are combined as vector \vec{v}_i in the Euclidean space \mathbf{R}^3 using all available axes, namely (x, y, z) . As we use a 3D accelerometer in our experiments, this definition makes sense here. To calculate the features, all \vec{v}_i of a certain partition are summed to the partition vector \vec{v}_{part} , see Equation ((4.4)) and [87].

The signals mentioned above allow us to extract a broad number of features in the *Feature Extraction Process* in order to characterize the events more precisely. The available feature space already allows a wide variety of applications but is easy to extend in order to allow an even broader variety of features in the future. We implement several quantitative (numerical) features [88], which can be differentiated into continuous-valued features such as *signal energy*, *intensity*, *orientation*, *magnitude* [87] and discrete features like *number of peaks*, *number of null points*, *histogram*, *Peak-to-Peak*, *Zero-to-Zero*, *natural frequency*.

Every feature is extracted for every partition during the training. For a more fine-grained signal investigation, the number of partitions can be increased while increasing the corresponding number of features with every partition.

Typically, the features are only combined with specific signals if this combination makes sense in the context of the event detection scenario. The natural frequency feature for example is typically not feasible to characterize a rehabilitation training for therapeutic exercises; in contrast, the orientation feature makes more sense as the different movements are very orientation dependent. The feature types *null point*, *Zero-to-Zero* and *orientation* are only combined with the raw data signal. The supervisor of the application training has to pick appropriate features from the feature pool in order to provide a meaningful feature selection depending on the application in general.

The remaining meaningful feature types can be combined with the normalization based signals to take advantage of the given comparability of the features in time, frequency, SN alignment, and signal intensity, (see Figure 4.3).

The additional feature normalization process, also known as feature scaling, allows the evaluation towards all other used features and the fusion of these features of different metrics in feature fusion. The feature normalization neutralizes the effect of different metrics across all features. The normalization process transforms all features into one common metric, which is ranged from 0 to 1. For this process, the min and max values of the original, non-normalized features are used as relative boundaries, while the min value of a feature reflects the lower boundary at 0 and the max value represents the upper boundary at 1. The classification model includes, among others, the parameters of the min and max values of each selected feature called *min/max feature boundaries* in order to be able to perform the above mentioned normalization process during the event detection.

4.2.3.4 Features

In the following paragraphs the individual features used are introduced.

Intensity: As introduced in [55, 76] the intensity I_{Feature} reflects how intense the signal is for a signal period s_p , which means that the smallest value within s_p is taken and compared to the highest value within the signal segment. This rather simple feature is defined in Equation ((4.1)) and uses only time and value normalized signals. For this reason, this feature needs to be used with more than one partition, otherwise all intensity features would always be equal.

$$I_{\text{Feature}} = \text{Max}(s_p) - \text{Min}(s_p) \quad (4.1)$$

Histogram: Histograms are already successfully applied in our prior work and are therefore introduced in multiple works [55, 70, 76]. A typical histogram segments the value range of a signal s into n equally sized bins. The number of values that fall into each bin are counted for every bin. As a result, an abstract representation of the distribution of the existing values is given, while the order of the values is lost. Exactly this distribution of each bin is used as a feature.

Energy: The general idea to use the signal energy as a feature is taken from [15]; the authors introduced the energy feature to assess fence intrusion events as shown in Equation (4.2).

$$E_{\text{feature}} = \sum_{i=-\infty}^{\infty} |s(x_i)|^2 \quad (4.2)$$

The energy feature reflects the introduced energy into the signal s by creating the absolute values of all samples and dividing them by the number of considered samples $a - b$ (where a is the first and b the last of s) in order to guarantee that the resulting energy value $E_{\text{featureSimple}}$ will not exceed the intensity feature I_{Feature} introduced above, which is the maximum amplitude of the signal. This optimization helps to have a predictable memory usage. This simplification results in Equation (4.3) introduced by [76].

$$E_{\text{featureSimple}} = \frac{\sum_{i=a}^b |s(x_i)|}{b - a} \quad (4.3)$$

Null Point: As introduced in [76], the Null Point feature simply counts the number of null points within a signal s and divides this number by the number of samples in order to gain a comparable value even for different event lengths. It is important not to perform a value normalization as the signal would lose all null points due this process. A time normalization does not help improve the signal comparability of this discrete feature, hence this feature uses non-normalized signals.

Peak: As introduced in [76], the number of peaks is counted for value normalized signals or frequency signals. Depending on a specific peak threshold, the peak is only counted if the signal increases by at least the predetermined threshold or more and then decreases the signal by the exact same value. It is important to mention that both the increase and the decrease are not allowed to be interrupted by an opposing increase or decrease. We use a threshold of 100 mg in this thesis.

Zero-To-Zero: As introduced in [76], the number of samples between two consecutive Null Points is counted. This process is performed for all pairs of consecutive Null Points. The highest number of samples between a pair of consecutive Null Points is represented by the Zero-To-Zero feature. This feature describes the longest event course section without an intense change of the signal into the opposite direction within the signal course. It is important to neither perform a value nor a time normalized signal as the signal would lose all relevant original information necessary to count the number of samples in between two Null Points.

Peak-To-Peak: As introduced in [76], the number of samples between two consecutive Peaks is counted. This process is performed for all pairs of consecutive Peaks. The highest number of samples between a pair of consecutive Peaks is represented by the Peak-To-Peak feature. This feature describes the longest event course section without an intense change within the signal course.

Orientation: As introduced in [87], both features, Orientation and Magnitude, are technically bound to each other as they need the same preliminary calculation base which uses the acceleration values of sample i combined as vector \vec{v}_i in the Euclidean space \mathbf{R}^3 using all axes, namely (x, y, z) . To calculate the features, all \vec{v}_i of a certain partition are summed to the interval vector \vec{v}_{int} , see Equation ((4.4)) and [87].

The orientation feature is defined by an orientation vector \vec{o}_{int} . This vector points in a direction based on the input of the three axes x , y , and z . This direction already describes the orientation. As a cleanup process, the length of the vector (see next feature Magnitude) is eliminated from that vector by performing a typical vector normalization, see Equation ((4.5)).

$$\vec{v}_{\text{int}} = \vec{v}_1 + \vec{v}_2 + \dots + \vec{v}_k \quad (4.4)$$

$$\vec{o}_{\text{int}} = \frac{1}{\|\vec{v}_{\text{int}}\|} * \vec{v}_{\text{int}} \quad (4.5)$$

It is important to neither use an amplitude nor a time normalized signal as the signal would lose all relevant original information necessary to create the orientation.

Magnitude: The Magnitude feature is defined by the length of the \vec{v}_{int} , which means it represents exactly the value of the previous vector we eliminated through the vector normalization. In order to gather this value we have to separate it during the normalization of Equation ((4.5)). The Magnitude feature is calculated by the square root of the scalar product over all degrees of freedom in Equation ((4.6)).

$$\|\vec{v}_{\text{int}}\| = \sqrt{\langle \vec{v}_{\text{int}}, \vec{v}_{\text{int}} \rangle} = \sqrt{x^2 + y^2 + z^2} \quad (4.6)$$

It is important to neither use an amplitude nor a time normalized signal as the signal would lose all relevant original information necessary to create the magnitude.

Natural Frequencies: The natural Frequencies are calculated based on the fast Fourier transformed signal by applying a peak selection for the first natural frequency.

4.2.3.5 Feature Fusion

In the *feature fusion* process the features for the subsequent feature selection are stored into one large distributed feature space, we call it source set S . All features of all SNs are stored in here.

With the subsequent feature selection, the best subset of features needs to be selected for every SN that is affected by the events to gain the highest classification accuracy. The goal is to reduce the size of these huge feature vector which exists for each trained event class. The reduction of the huge dimensionality is necessary in order to address the need for a reduced in-network communication that exchanges these features. By using features of multiple SNs which have

observed one distributed event from different perspectives we are able to describe a distributed event within one single and concatenated feature vector. Multiple event types differ in their feature vector characteristics which allows a classifier to assign the appropriate class to an arising event. For the subsequent *Feature Selection* we use this fused vector.

The resulting feature space F_{space} is defined in Eq. (4.7). In short, every feature is created for every SN- observing the event from a different perspective. All these features are created for all distinct events that have to be trained. In more detail, the number of SNs N that could possibly be affected by a single event are multiplied with the number of meaningful signal s and feature type T combinations called function $M(F(< T, s >))$. A tuple is called F . This means that depending on the application some possible F s are discarded because of contextual reasons, as mentioned in Section 4.2.3.3. Furthermore, we multiply the number of partitions P , which scales the fine-grained event characterization.

$$F_{space} = M(F(< T, s >)) * N * P \quad (4.7)$$

In the case of our example fence application, the real numbers of created features are calculated as follows: The given feature space F_{space} is calculated from 60 meaningful features F and these *available features* (see Figure 5.30) are based on the meaningful combinations of signal and feature type tuples ($< T, s >$). These features are extracted for all 6 affected SNs and for each of the 3 (see Section 4.2.3.3) event partitions P . For each event class the resulting features space of 1,080 features is available for the feature selection. 19 features are eventually selected for the classification model.

4.2.3.6 Feature Selection

Features are represented in different metrics. For example, a feature that represents null points in a function uses the *quantity* metric, while an intensity feature for acceleration values uses the m/s^2 metric. In order to use multiple features in a common feature vector – which we aim to do – all features have to use the same metric, which can be reached throughout the normalization process, or by re-scaling the range of features [80]. The different metrics across the features are normalized into a common metric ranging from 0 to 1. For this process, the minimum and maximum values gathered during the training of the respective and non-normalized features are used as a relative reference, while the minimum value of a feature reflects 0 and the maximum value represents 1. The subsequent feature selection refers on normalized features.

The goal is to select only appropriate feature type subsets of the complete feature space. These subsets represent the events using so-called *reference vectors*. Hence, the SNs do not have to calculate all other unused features and address the well known WSN restrictions by reducing processor and memory usage and omit data to be transmitted by this.

The *Feature Selection* process is powered by the well known data mining tool WEKA [64] which allows us to investigate the classification problem with an easy to extend and community-driven platform. In order to pick the best-performing features from the given feature space a procedure is needed that is able to assess the quality of the features and selects the best features that optimizes solving the given classification problem.

In [12] and [89] the authors introduce several optimal feature selection algorithms (e.g. branch and bound, british museum or procedure of dynamic programming) with the goal to find the best feature set for a given classification problem without the need to investigate every single feature

subset. Nevertheless, these approaches are not computationally feasible as the number of features still causes an exponential growth of feature subsets [12]. In order to benefit from the broader feature coverage of high-dimensional feature spaces we have to pick a sub-optimal but minimized feature set. A very effective and efficient approach is the Sequential Floating Search Method (SFSM) comprising of a linear forward selection and a linear backward elimination algorithm as introduced in [90,91] and [92] and deployed in [64]. Hence, we suggest not to use optimal but approximating algorithm in favor to computational time.

In order to optimize the feature selection, a ranking method can be applied for each feature by assessing in beforehand the sole property to solve the classification problem. In [93] it is suggested to combine the SFSM with such a ranking algorithm that helps to ensure the investigation of highly descriptive features at first. Although each feature is assigned a rank – *it has been recognized that the combinations of individually good features do not necessarily lead to good classification performance* – which means – *the m best features are not the best m features* [91]. Hence, the ranking method enhances the feature selection process. The SFSM is used to test these features in different combinations in order to finalize the validation of their capabilities to be used in the resulting reference vectors.

The SFSM uses two sets, the empty target collection T of selected features and the set S containing all available features. With every iteration of SFSM a feature f_S is moved from S to T if it increases the classification quality; this is called forward selection. When new features are added to T , other previously added features' classification impact may be reduced or become redundant. Hence, the subsequent backward elimination deletes every feature f_T that increases the classification quality if it is removed from T . In addition, and depending on the event detection problem, we can add specific weights to the features' ranking in order to tailor the feature selection to specific needs. Some example feature selection weights are introduced in Section 4.2.3.7 based on the patent [94].

It has been shown that the SFSM delivers a nearly optimal subset of features within an acceptable run time by [95]. In addition, in [64] SFSM has been proven to deliver feature sets that do not run into the over-fitting problem if applied according to the Wrapper-Method [73,91,96]. The wrapper method uses the integrated classification algorithm to assess different feature sets, which results in a much better classification result of the chosen classification algorithm [12]. This combination of feature assessment and feature selection delivers a nearly optimal subset of all possible combinations of the features.

The principle of our feature assessment is based on a simple check routine. In order to assess the features according to the so-called Wrapper Method, we create a classification model for the chosen classification algorithm. The next step classifies the trained classes to check whether the classification model can successfully assess all events to its according class. As the according class is known by the preceding supervised training, classification errors for all classes are now easy to trace.

If we use the same training data set for the creation of the classification model as well as for the check routine, the resulting statement expresses only how well the original training data set can be classified. We need an estimate of how well the feature set performs under real conditions with “new” or thus far unknown events. Therefore, we combine the wrapper method with the *Cross Validation* to most effectively perform the mentioned check routine. The cross-validation allows us to evaluate the classification problem with a pseudo-unknown data set. By this we create a better estimation of the performance of the classifier under more real world conditions. To reach

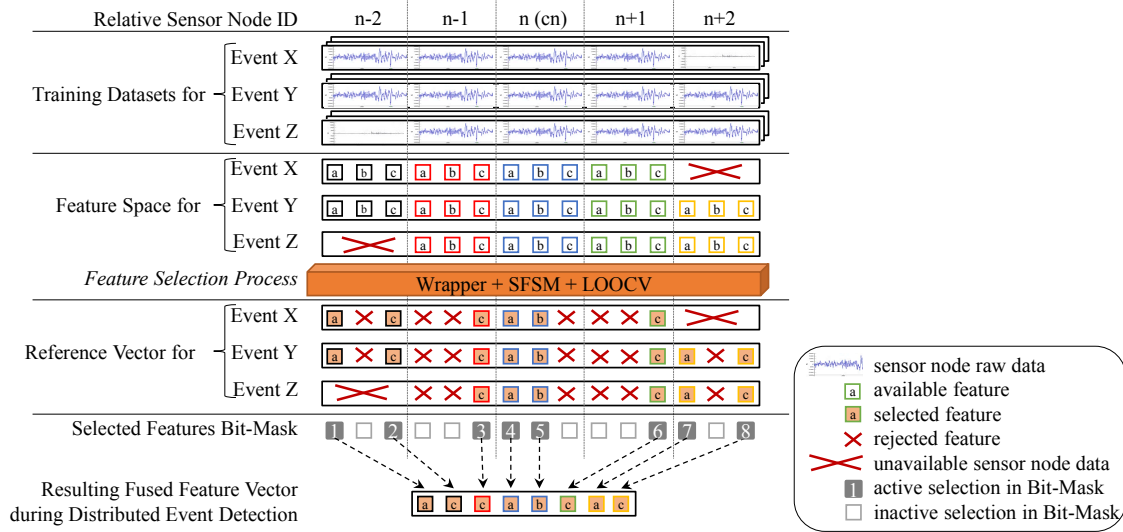


Figure 4.4.: Process of *reference vector* and *selected features bit-mask* creation. By applying the *selected features bit-mask* the Distributed Event Detection is able to recompile the *fused feature vector*, adapted from [55].

this goal, the cross-validation splits the training set T into two subsets A and B . Set A is used for the classification model and set B to check it as described in [64]. A common cross-validation type is the k -fold cross-validation that divides into k equal partitions and k repetitions of training and evaluation are performed. All sets of the $k - 1$ folds are iteratively used for each training step while the remaining fold is used for the classification evaluation. The most meaningful results can be reached by the deterministic Leave-one-out Cross-Validation (LOOCV) algorithm [97] invented by Quenouille [98] where k is equal to the amount of training sets n in the data set. To test the feature set's ability to classify the investigated events correctly, LOOCV simply repeats the check routine for all n training sets while in every pass only one distinct training set (B) is removed from the data collection. With these $n - 1$ training sets (A) the classification model is created and the removed single data set B is then used to determine the according classification error.

This whole check routine is repeated for every feature to be evaluated concerning its operational capability. As stated above, SFSM adds or removes a feature, and for every modification of the feature set the LOOCV validates the overall classification error for all possible $n - 1$ subsets of A . If a feature increases or does not change the classification error, it is not selected, otherwise it is added to or kept in the resulting feature vector.

The selected feature types are used in the event detection to classify the event, see Fig. 4.4. In order to have a prototype for each event type, we need the reference vectors introduced above for each class within the later applied prototype classifier. We calculate the reference vectors for each event class by averaging all extracted features of a single feature type of all training sets of a specific class. The resulting averaged features are collected in the reference vector for each class in a fixed and overall known order.

As depicted in Figure 4.4, the example available feature space as well as the trained data sets for each event class are used as input for the feature selection process described above. In case of events that have less propagation, some SNs are not affected and hence do not and will not deliver appropriate event data. This is reflected in a further reduced feature space for the according events, depicted with a red cross for unavailable data. The feature selection process applies the introduced

wrapper method and extracts for each event class an according and best performing reference vector. Small red crosses depict features rejected by the wrapper method as these features do not improve the classification performance. In order to extract a common *Fused Feature Vector* that can be extracted from future unknown event data that have to be classified, the *selected features bit-mask* is created from the given reference vectors.

The bit-mask masks all used relative *SN* positions with their features. The selected features represent a common definition that allows to discriminate between all investigated event types.

The common bit-mask is stored on every *SN* as a guideline to which features are needed and how they should be fused during the Distributed Event Detection. The bit-mask additionally saves the relative node-ID and thus reflects the global knowledge of the event types by saving the relationship of the affected *SNs*' perspectives to one another.

A problem will arise during the distributed event recognition for features of classes that do not support those features, as they are not available because of unaffected *SNs*. Figure 4.4 depicts this problem in event *X* at node $n - 2$ and in event *Z* at node $n + 2$. Both events do not even exceed a wakeup threshold at the mentioned *SNs* (for threshold logic details please refer to Section 6.2). Nevertheless, the majority of the performed events do wake up these *SNs* and raw data is created, hence features are selected. The question is how to deal with the situation of the so-called *missing feature problem* during the later event detection. We discuss this problem in Section 5.7.3.

4.2.3.7 Weighted Feature Selection

The ranking method of the feature selection can be customized and modified by additional topological information regarding the position of the *SNs* which is assigned either by e.g. the help of a *GPS-Module*, a dynamic model or a predefined relative position of the individual *SNs* in the *WSN*. As defined and introduced in detail in [94,99,100], based on the topological information, the feature selection takes into account the topological information regarding the position of the individual *SNs*. The weighting methods from [94,99,100] described in the following can be independently added to the ranking method.

- **SN Weight:** A weighting can be applied to the feature selection with the effect that features which originate from a *SN* that has already contributed other attributes to the selection are weighted more strongly.
- **Feature Count Weight:** A second weighting can be applied with the effect that features in which corresponding features are already identified by other *SNs* are weighted more strongly.
- **Event Location Weight:** An third weighting can be applied by taking into account how many features a *SN*, depending on its specific location relative to the location of the event, has already contributed to the recognition.

These weights reduce the number of involved *SN* and features to address performance issues in case of heavily increased network size and density. In addition, the number of selected and to be distributed features can be reduced with the introduced weights in case of a huge feature space selected by the *SFSM* that has a negative effect in future applications features as the number of features may extend the packet size.

The events investigated in this thesis do not affect more than six *SNs* at once. As investigated in Section 5.5.6, our proposed approach can theoretically handle up to 64 affected *SN* without running the risk to find itself outperformed by one of the later introduced information fusion approaches.

As a result we did not use these weights in our investigations as the reduction of **SN** for our current applications would reduce the beneficial influence to observe events from multiple perspectives. In contrast, future deployments with more affected **SNs** and an immensely increased feature pool would benefit from this weighting.

4.2.3.8 Model Creation

In the following we introduce all parts of the classification model and highlight all model parameters in a bold font.

The main parameter of a classification model is the **classifier** that is being used, hence the classification algorithm used by the *Feature Selection* and the *Modeling Process* is exchangeable. Typical applicable classifiers are introduced in Section 3.3.3. It is possible to evaluate different classification algorithms off-line to find the algorithm which matches the application dependent requirements as we did for example in Section 5.6.3. Nevertheless, the chosen classification algorithm should be assessed according to the needs and restrictions of **WSNs** and **SNs**. The evaluation framework gives relevant evidence whether a carefully chosen classification algorithm with the according feature set will perform well or not.

The classification model created during the model creation consists of a list of used **signals** which we introduced in Section 4.2.3.1.

In order to have event representatives available during the online classification, we need **reference vectors** introduced in Section 4.2.3.5 if the Euclidean distance-based classifier is chosen; in any other case, the according reference data need to be stored on the **SNs**.

According to the reference vector, it is important for the **SNs** to know all features that need to be extracted for a comprehensive event detection; the selected features (see Section 4.2.3.6) are represented by the **selected features bit-mask** as introduced in Section 4.2.3.6 and depicted in Figure 4.4.

The signal normalization processes has to have the individual **maximum signal values** available for the time and value normalization of the raw data as introduced in Section 4.2.3.1.

The feature normalization processes need **min/max feature boundaries** for each feature introduced in Section 4.2.3.3.

The filter package's main goal in the *Distributed Event Detection Framework* is to filter the **CN**. The **CN-Feature** introduced in Section 4.2.3.2 identifies the specific **SN** that performs the feature fusion depending on the event location and transmits the final event report. Our example fence monitoring system, for example, uses the introduced intensity feature of the first x-axis bin of a fence event to locate the node's closest position relative to the location of the occurring event. The application of the filter package is described in detail in Section 4.3.

In real world applications, it is important to distinguish e.g. between neutral, intentional, and critical events. We can simply map the trained events in a simple **event categories matrix** introduced in Section 5.6.1 – as part of the classification model – to such or other appropriate event categories by deriving some categories that group events in a common context that is relevant to the application. In addition, one or more **report flags** introduced in Section 5.6.1 can be added to the classification model to define which event types should be reported to the control center. Additional date and time related dependencies can be added to the report flags in order to support special applications that need to detect events dependent on e.g working hours or specific dates.

The **node topology matrix** introduced in Section 4.2.3.2 reflects the real world and physical

SN deployment structure and helps to know which SN has a neighborhood to another SN.

The **event topology matrix** introduced in Section 4.2.3.2 identifies the relative location for affected SNs during an event to each other.

Once the training and the included data mining process are complete, the created and evaluated classification model parameters are transmitted to each SN which enables the subsequent distributed event detection process.

4.3 Distributed Event Detection Framework

As introduced in [5], the overall goal of the *Distributed Event Detection Framework* is to classify or detect events based on the preceding training and to finally filter relevant events that should be reported to an event handler that is observed by an e.g. another SN. The event handler can be located within the WSN and activates further processes automatically, such as an example automatic evacuation procedure or an additional camera surveillance system. The WSN can alternatively advise a human control center that could e.g. send maintenance workers to a bridge or a security guard to a fence for further actions.

The *Distributed Event Detection Framework* comprises three process blocks, (c) *Local Event Recognition*, (d) *Distributed Event Recognition* and (e) *Report* (cf. Fig. 4.2) which are used to calculate locally observed parts of an event, the event fusion with distributed information and the final report of critical or relevant events. The focus is to send only one single event packet to the BS in order to support multi-hop sensor networks with a very low energy consumption through data transmission.

In contrast to the preceding *Evaluation Framework*, the process blocks of the *Distributed Event Detection Framework* are connected by wireless transmitted data packets in order to provide real world conditions during the evaluation of the WSN.

4.3.1 Local Event Recognition

The first process block is the (c) *Local Event Recognition* and represents the idea that a single SN is exposed to a distributed event but that node can only recognize a part of the event out of a specific view. Multiple SN gather data locally in *Local Event Recognition* and extract exactly the features selected in the event modeling process.

The local event recognition describes the event part out the different perspectives of the SNs. This means that every single SN delivers a piece in order to help complete the whole puzzle.

Process block (c) ends with the transmission of all extracted features to the one-hop neighborhood which covers the affected SNs in our investigated applications. If events have a larger spatial distribution than a typical 1-hop communication does, a preliminary event announcement phase is needed to ensure that all nodes know which nodes are affected by events. This is necessary in order to determine the CN based on the physical event center. None of our investigated applications exceeded the 1-hop communication for the additional in-network communication.

4.3.1.1 Sampling & Segmentation

In process block (c), every SN affected by the event gathers raw data with the same sampling rate as in the (a) *Training* process block. An event threshold allows the system to use a power down mode until the threshold is exceeded by events.

The segmentation is performed during the event detection according to the created segmentation parameters of the training in Section 4.2.2.1. Again, every SN performs the segmentation independently and locally on the SN as indicated in Figure 4.1 at process block (c). In contrast to the (a) *Training*, the segmentation directly continues to preprocess the segmented raw data on the SNs.

4.3.1.2 Preprocessing

Depending on the classification model, parameters of all signal types are now created on every SN affected by an event in order to feed the subsequent feature extraction with data. All signals are derived from the event segment and stored on the SNs to allow the feature extraction to create all necessary features.

4.3.1.3 Central Node Extraction

The procedure of the CN extraction depends heavily on the application and its characteristic as introduced in Section 4.2.3.2. In general, the CN should be detectable for every event during the event detection, which means it should ideally represent the center of an event or it should be as close as possible to the center of the event. In case of fire events, the SN closest to the fire source should be assigned to the role of the CN for example. In some scenarios it is not necessary to find the CN autonomously because the number of SN is too small and the events always arise at the same positions relative to the SNs. A good example is the use of Wireless Body Area Networks (WBAN) to detect different movements. Movements like running, sitting, or jumping are constantly performed in the same position towards the SNs. In particular, the training is performed at exactly the same SN configuration, while a fence surveillance system can not guarantee that an intruder uses the same fence as the trainer. Hence, the fence has to detect which fence the intruder is climbing by itself.

For our example of a fence surveillance system, we used the acceleration intensity feature of the first x-axis bin of a fence event to locate the SN's position relative to the location of the occurring event. For this purpose, we extracted the intensity value of the SN called CN and located to the left and closest to the training. Please refer to Figure 7.3 for an example CN location. Every SN extracts the CN feature which describes the SN's potential to be the CN. As soon as the SNs have collected the SN features from all affected SNs, each node will know and can filter which SN is the CN for the current event as described in Section 4.3.2.1.

4.3.1.4 Feature Extraction

As mentioned, the classification model contains the selected features bit-mask for the *Feature Extraction* which indicates all the features that have to be extracted. Even feature types that are exclusively used by the CN are necessary to be extracted by the other nodes as they could be the CN.

Let us examine a simple example where the CN is defined as n : Feature type a is needed for node $n-2$, n , and $n+2$. Feature type c is needed for node $n-2$ and $n-1$. As the n needs to know all the a and c feature from the neighborhood, every node has to extract and transmit this feature within the network to the neighborhood. Feature type b is needed only for node n .

4.3.1.5 Feature Distribution

According to the main concept depicted in Figure 4.1, the *Feature Distribution* connects in Figure 4.2 process block c) and d).

After the feature extraction process in Figure 4.2 (c), all event-triggered SNs broadcast the calculated features and their CN parameter in order to distribute the features within their n-hop neighborhood.

Usually, n is set to 1 for our scenario; under Line of Sight (LOS) conditions, the radio range exceeds the 200 m of our SNs. The radio range is significantly larger than the typical spacial expansion of any event of our four investigated applications. For example, in our fence scenario, a maximum spacial event expansion of approximately 22 m, which reflects the length of six fence elements, is caused by the events in our fence application. The event expansion subsequently affects six SNs that can easily be reached within a single hop by a radio transmission as the 22 m event range is clearly smaller than the aforementioned 200 m communication range of our transmitter. Nevertheless, we have implemented and successfully evaluated multi-hop communication scenarios as presented in [101].

The principle of the Distributed Event Detection means that every SN that recognized an event assumes that it is the CN until the CN-Filter detects the true CN. In order to classify this event, the CN has to combine all features of all affected SNs into one feature vector according to the bit-mask. Feature types that are exclusively used by the CN do not have to be transmitted to the neighborhood in the WSN, as the data fusion is performed on the CN itself. In Figure 4.4 we depict an example where the selected feature bit-mask forces only the CN to extract feature b . All other feature types, including those used by multiple nodes, have to be extracted from every node. This can further reduce the exchanged features.

4.3.2 Distributed Event Recognition

As depicted in Figure 4.2, the *(d) Distributed Event Recognition* represents the idea that multiple SNs exposed to an event can recognize events better with more than one of these SNs. All these SNs are physically distributed so that they cover the spacial event propagation at least with the whole WSN. The definition of the *local event inspection* states that we can use the local event recognition of every single SN involved in an event. By combining the features of the local event inspection they turn into a *piece in the puzzle*. As each piece in the puzzle needs to be assembled in the correct way, the SNs need to know the classification model of the events as introduced in Section 4.2.3.8. The selected features bit-mask of that classification model can now be used by every SN to decide how to assemble the collected puzzle pieces in the correct way. In order to classify the events successfully, only the CN has to combine all features of all affected SNs into one feature vector according to the selected features bit-mask.

These two important concepts, “local event inspection” and “distributed event recognition” allow us to assemble the distributed knowledge on the basis of the well-known divide-and-conquer principle.

4.3.2.1 Central Node Filter

Every SN collects distributed features by receiving the packet that delivered the CN feature. With the help of the CN feature type the SNs decide whether they are the CN or not. The SN with the

highest value feature or equal to the CN feature with the highest value passes the CN filter which initiates the subsequent feature fusion.

In the case of a SN that was determined to be one of multiple CNs, this node needs to communicate later on with all potential CN SN to find the one with the highest reliability. Multiple CNs indicate that some errors arose during feature extraction. Errors can arise for several reasons. One can be that the perspectives of the SN towards the event are too similar which means that the data gathered is very similar. Other problems can be a misplaced SN, dampening effects, a depleted battery, unexpected weather conditions, and many more. But these errors can only be investigated after classification within the *Quality Extraction* process described in Section 4.3.2.5.

All SNs determined not to be the CN by the CN-Filter do not need to continue with the process of event detection, event fusion, and report [5].

4.3.2.2 Feature Fusion

Features of the CN's own observation are extended by features of the neighborhood depending on the event topology. Hence, every node that passes the CN's filter builds its own unique view of the event using features sent from the neighboring nodes. The CN as well as all other SNs use the known event topology in order to assign the SN-ID of an incoming packet to a position relative to the CN.

As soon as all relative positions of the neighbors are assigned, the relative positions are then looked up within the selected features bit-mask in order to arrange all the features in the fused feature vector according to the selected features bit-mask entries and event topology, which is introduced in Section 4.2.3.6. This created feature vector is now composed of the features of multiple SNs out of the incoming packets.

As depicted in Figure 4.4, the CN has to create exactly the depicted event vector with due regards to the order of the features.

4.3.2.3 Feature Normalization

The different metrics (e.g. m/s, mg, # of peaks) given by the different features are normalized into the common metric ranged from 0 to 1. For this process, the classification model delivers the min/max feature boundaries of the original trained and non-normalized features.

All extracted features are feature normalized according to the min/max values of the classification model introduced in Section 4.2.3.3.

The feature normalization neutralizes the effect of different metrics across all the features. The normalization process transforms all different metrics of the features into one common metric which is ranged from 0 to 1. For this process the min/max feature boundaries values of the respective and not normalized features are used as a relative reference, while the min value of a feature reflects 0 and the max value represents 1. The classification model includes among other parameters the min and max values of each selected feature.

4.3.2.4 Classification

Finally, the classification process is performed on the CN. As introduced in Section 3.3.3.3, we use the prototype classifier for our event detection and classification; this means that during the classification process, the distance between all trained reference vectors and the unknown vector is calculated. The classifier assigns the unknown event to one of the known events.

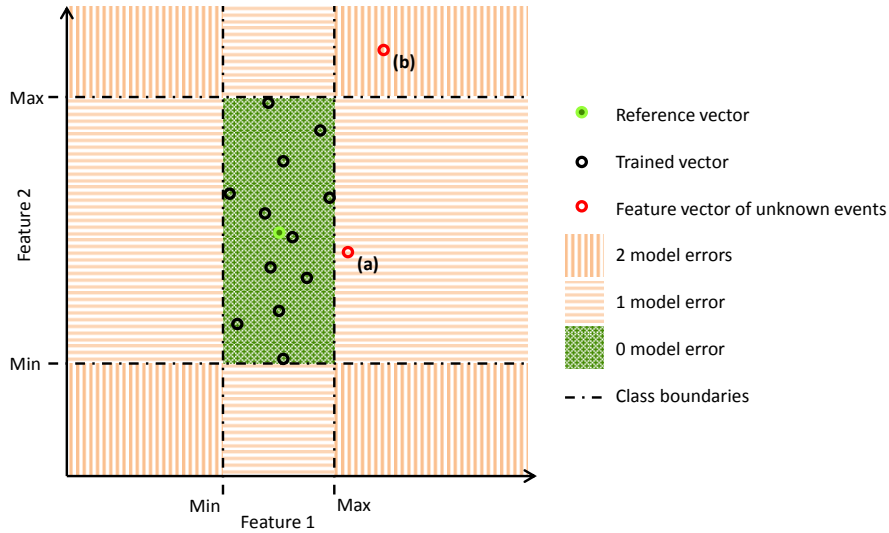


Figure 4.5.: Model error examination during the quality extraction process of an unknown event, adapted from [76].

If the unknown event is not one of the a priori trained events, the setup during the training needs to be updated; however, the subsequent *quality extraction* is able to provide some feedback on the reliability of the classified event if the application requires it.

4.3.2.5 Quality Extraction

The *Quality Extraction* creates a quality label for the resulting classification. This is only necessary if multiple SNs pass the CN filter but can be performed to an extra level of confidence for the classification result and its reliability. These SNs exchange the quality label amongst each other to determine which node is the most reliable *cn* by reviewing the classification results against the classification model to obtain a quality label.

In any other case, the *Quality Extraction* is skipped by using the *Application Filter* directly without any additional necessary communication.

In order to verify the individual classification quality we have to compare the alignment of the feature towards the class boundaries evaluated during the training. These min/max feature boundaries are already used for the feature normalization. If a feature extracted during the event detection does not lie within these boundaries, it is not a good representative of the trained data; a clue that the current observed event was not trained the way it is recognized. Every feature that does not lie within the class boundaries is counted as a model error E_{Model} as depicted in Figure 4.5.

An exception has been defined for missing features, which are introduced in Section 5.7.3. Missing features can not be considered within the quality extraction procedure as they do not indicate a deterministic model error.

The classification quality Q introduced in [76] and formalized in Equation (4.8) is calculated from the fraction of the number of features used for the classification M to the number of found model errors E_{Model} .

$$Q = 1 - \frac{1}{M} * E_{\text{Model}} \quad (4.8)$$

The example in Figure 4.5 shows the reference vector as well as the trained data. The trained data define the class boundaries for one class and within the given example only for two features for illustration purposes. The red circles represent the feature vectors of unknown events; depending on the region of their location, they cause 0 to 2 model errors. Red circle (a) passes the class boundary of feature 1 and causes 1 model error, while red circle (b) passes the class boundary of both features and thus causes 2 model errors. For this particular example, we sum the errors up to 3 resulting model errors E_{Model} . Let us use the 19 features that are necessary for our fence surveillance application as an example for parameter M . By calculating an example quality label Q with Equation (4.8), the value Q of 84.2% is calculated. This quality label Q is subsequently distributed as the necessary filter parameter to assess the classification result of a specific SN .

The resulting transmission of only one resulting classification packet instead of transmitting packets from multiple SNs leads to an increased classification unambiguity and a communication reduction. In order to resolve this classification unambiguity, one additional in-network packet for all SNs passing the CN-Filter has to be transmitted in order to evaluate the classification based on the above introduced quality label.

4.3.3 Report

During the *Report Process Block* the final classification transmission to the BS or an event handler that could be located within the network or directly integrated in an alarm system is prepared and executed within the subsequent filtering and handling processes.

4.3.3.1 Quality & Application Filter

The quality filter is performed on every SN that passes the CN-Filter . The distributed Quality label of every SN that passed the CN is now known at each of these SNs . This means that all nodes that passed the CN can now compare their value (Q_{CN}) to all other received $Q_{\text{N}\pm x}$, while $\text{N}\pm x$ is the relative location to the evaluating SN . If Q_{CN} turns out to have a higher value than any other node it reports the event, see Equation (4.9). If Q_{CN} turns out to have a lower value than any other node, it does not report, see Equation (4.10). If the Q_{CN} cannot outperform any other SN but equals at least one SN , the SN with the smallest SN-ID is allowed to report the event, as no SN can be preferred, see Equation (4.11).

$$Q_{\text{CN}} > Q_{\text{N}\pm x} \forall x \mid Q_{\text{CN}} \text{ reports event} \quad (4.9)$$

$$Q_{\text{CN}} < Q_{\text{N}\pm x} \forall x \mid Q_{\text{CN}} \text{ reports nothing} \quad (4.10)$$

$$\exists_{\text{N}\pm x} \text{ for } Q_{\text{CN}} == Q_{\text{N}\pm x} \wedge \nexists_{\text{N}\pm x} \text{ for } Q_{\text{CN}} < Q_{\text{N}\pm x} \mid \min(\text{SN-ID}) \text{ reports event} \quad (4.11)$$

This means that the quality filter eliminates leftover classifications of SNs which contain an inappropriate event perspective or, as stated above, a lower classification quality, by stopping any further transmission directly at the node.

In addition, the *Application Filter* checks whether the resulting event classification falls into a predefined relevant event category based on the event category matrix that is necessary to be transmitted. This categorization is important for the responsible personnel or user to be informed about relevant events – but only when necessary. Hence, only the SN with the highest quality label is allowed to relay the labeled classification to the control center if this event falls into a

relevant event category, where a relevant event category means that a report flag indicates that this category needs to be reported.

If the quality filter did not need to be passed, we can still optionally extract the quality on the resulting CN, see Figure 4.1 (e). As soon as the quality label is available – either due to the filter needs or due to the optional extraction – the resulting event report provides an event notification including an additional event assessment called *quality label* available at the BS. Thus, the quality label can help decide how to handle specific events as varying event detection qualities may cause different reaction directives. For example, a fence with a low quality shake may cause an alarm or additional camera systems activation as a low quality label indicates that the classification did not perform optimally and additional systems are necessary to verify the result more precisely.

For the purpose of energy efficiency, it is important to mention that most of the arising events are not relevant for typical scenarios like fence surveillance, fire detection, or disaster prevention. Depending on the introduced report flag, and additional possible time or weather dependencies it is defined which events need to be reported. In addition, it should be mentioned that even a good monitoring system always has a chance of causing a false negative. Nevertheless, if the network capacities and lifetime requirements of the application allow additional transmissions of detected events, the transmissions should be allowed if they increase safety for particular security applications.

4.3.3.2 Event Handling

The final process is not covered by our Distributed Event Detection system directly. First and foremost, the event handling is performed by humans who receive the event notification from the WSN. Nevertheless, it is very likely that future manifestations of the Distributed Event Detection system will add automatically triggered systems that can be triggered directly – if they provide a compatible transceiver – without the need to contact the BS first. An example could be a fire-triggered self-activation of a sprinkler, or an automatic boom gate regulation, or an additional and automated camera system, or an automated horn signal. More complex services will be possible and activated in a cascading way where such subsystems are activated by the previously event-detecting Distributed Event Detection system. A fully autonomous and e.g. cascading scenario could be activated by the proposed system by detecting a fence intrusion with our system whose event notification activates spotlights and cameras with a subsequent door regulation.

CHAPTER 5

Theoretic System Investigation

In the following sections we want to investigate our proposed system towards the theoretical applicability and its performance with a more generic analysis in order to clearly point out the overall impact towards the questions of how lifetime, communication effort, and classification performance evolve in a broad scale of applications. We explain why we need to pick an appropriate distance measure and we compare our new approach with four other information fusion approaches by investigating the amount of necessary packet transmissions over a scaling number of communication hops and the resulting lifetime depending on five different parameters. In addition, we investigate the classification performance under ideal hardware conditions and introduce how varying numbers of inoperable sensor nodes affect the proposed system. We compare multiple classifiers and select the most suitable for our application of fence surveillance. By introducing event categories for the event classes we show how real world requirements may influence the classification results if the detection tasks adapt to it.

5.1 Distance Measure Criteria

As mentioned in Section 3.3.2, we want to investigate the **ED** and the **MD** measures with respect to our application of **WSNs** in order to pick the most suitable distance measure. The following eight criteria have been chosen to qualify a distance measure to be applicable for our Distributed Event Detection system, adapted and extended from [59]:

(i) Metric, (ii) Feature Compatibility, (iii) Reduced Communication, (iv) Fast Evaluation, (v) Memory Consumption, (vi) Missing Feature Tolerance, (vii) Small Training Data Set Tolerance, and (viii) Decision Boundaries.

Metric: For our **WSN** applications it is important to choose an appropriate qualified distant metric that suits our purposes. A distant measure should meet the requirements of a metric in order to guarantee that the distance measured expresses directly the quality of its membership to a class. If d_{rs} is the dissimilarity of object s from r , then

$$d_{rs} \geq 0 \quad \forall r, s \tag{5.1}$$

$$d_{rr} = 0 \quad \forall r \tag{5.2}$$

$$d_{rs} = d_{sr} \quad \forall r, s \quad (5.3)$$

Only if additionally to the three conditions above, the following triangle inequality is true for the dissimilarity measure, it is called *metric* where the *distance* is appropriate to be used [12].

$$d_{rt} + d_{ts} \geq d_{rs} \quad \forall r, s, t \quad (5.4)$$

Feature Compatibility: Our Distributed Event Detection system uses features to describe classes. Hence, a feature vector compatibility should be given for the selected distance measure that allows processing the incoming event representation.

Reduced Communication: Our main purpose is to reduce the amount of data that has to be exchanged between SNs or that has to be sent back to the BS. The reason is that all radio activity is heavily energy demanding and should be avoided whenever possible [102]. Hence, the distance measure should not increase the communication to more than a linear increase in packets if the number of hops increases.

Fast Evaluation: The Distributed Event Detection system's report or feedback should be given immediately after the event arises in order to provide the system with reactivity. Long processing times strain the lifetime of the battery and reduce usability due to reduced reactivity. A suitable distance measure should provide the system to classify the incoming event description in a very short period. For this, the metric needs to provide a non-quadratic runtime in order to provide a good scalability in terms of evaluation speed even if the problem size increases.

Small Training Data Set Tolerance: The high number of training sets is important in order to holistically cover the classification problem. If only a limited number of training sets is available, it is likely that relevant characteristics of the event or classes may be overlooked for the later application. Nevertheless, it is important to see that the number of training sets increases tremendously with every class that has to be investigated, especially in the case of physically applied events where the subject group has to climb over a fence or has to perform exhaustive sport exercises [10].

Decision Boundaries Support: Decision boundaries [12] help discriminate between certain classes. A perfectly fitting decision boundary reflects mostly the training data but is less tolerant to real world circumstances; this issue is called over-fitting [10]. A linear decision boundary may be too rough under the first investigation but can offer an appropriate solution under the restrictive requirements of a WSN.

Memory Consumption: Memory consumption is still an important topic in WSNs if we consider that some classification problems may involve hundreds or even thousands of classes or training data sets. This means that such problems may increase the necessary amount of related information within the classification model to an extent that exceeds the available space on such limited hardware as a sensor node. An appropriate distance measure has to restrict itself to a small, ideally linear amount of data compared to the problem size.

Missing Feature Tolerance: In the area of WSNs, successful radio transmissions can not be guaranteed during the event detection, as packets may not reach their destinations even after a

reasonable amount of retransmission. In addition, the event definitions do not necessarily comprise the same number of features. This leads to a missing feature constellation during every classification process. A chosen distance measure should tolerate these circumstances as good as possible. We go into more detail concerning missing features in Section 5.7.3 and Section 5.7.3.2.

5.1.1 Euclidean Distance

The L2-Norm or **ED** is a widely used distance in the area of mathematics and statistics as it is simple to implement and shows very good results. **ED** mainly calculates the linear distance between two vectors based on the Pythagorean theorem. The **ED** is calculated by the square root of the sum of all squared differences between two points in the Euclidean space $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ as represented in (5.5), taken from [12].

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^d (x_i - y_i)^2} \quad (5.5)$$

A normalization (rescaling) process has to be performed for the features in order to guarantee that all features are in a comparable scale, otherwise some features have an unfair influence because of their inherent value range.

$$c = f * (\text{sizeof}(\text{type})) \quad (5.6)$$

ED represents a metric according to (5.1), (5.2), (5.3), (5.4) and supports the usage of features. As raw data are reduced to these features only and no further information has to be transmitted to use the **ED**, it effectively supports minimized communication. The algorithm is simple and has a runtime of $\mathcal{O}(f^2)$, which is reasonable and allows a fast evaluation for a distant measure in **WSNs** even if the problem size increases. The memory consumption c is very low as only the features f according to the used data **type** have to be stored and finally transmitted as shown in (5.6). The offered decision boundaries for the **ED** are only linear boundaries which reduces the expressiveness to discriminate between classes.

5.1.2 Mahalanobis Distance

The **MD** is defined as follows: For a data distribution that is defined as a mean \mathbf{m} (which is the center of the mass of a multivariate normal distributed training set) and a variance-covariance matrix Σ , the **MD** from a point $\mathbf{n} \in \mathbb{R}^d$ to the mean \mathbf{m} is :

$$d(\mathbf{n}, \mathbf{m}) = \sqrt{(\mathbf{n} - \mathbf{m})^T \Sigma^{-1} (\mathbf{n} - \mathbf{m})} \quad (5.7)$$

The variance-covariance matrix defines the relation between the features of a class and attention has to be payed on the fact that the variance-covariance matrix is inverted for the **MD** which restricts the application to non-singular variance-covariance matrices. In one dimension, this can simply be interpreted as the distance from the mean as measured in units of standard deviation. The **MD** represents the **ED** if the covariance matrix is the identity matrix. The variance-covariance matrix maintains the correlation information of every available feature and thus of every dimension [104, 105]. This directly explains the advantage of the **MD** as this information may enhance the classification accuracy based on the distant measure.

$$c = f^2 * (\text{sizeof}(\text{type})) \quad (5.8)$$

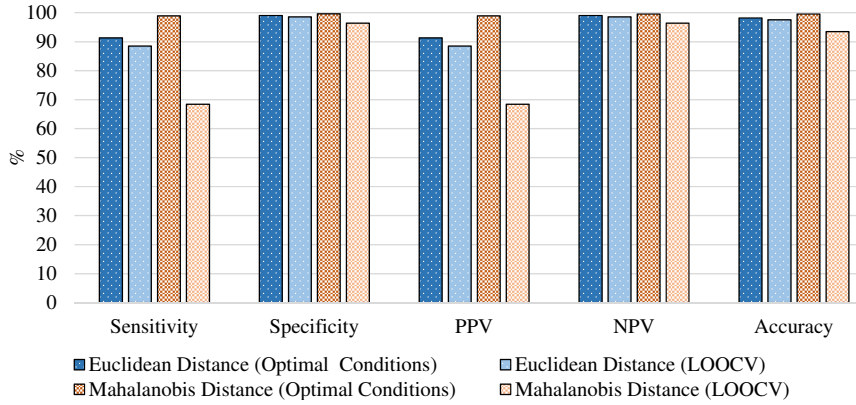


Figure 5.1.: Performance comparison of the Euclidean distance with the Mahalanobis distance under optimal conditions and under real world conditions simulated with the LOOCV, adapted from [59].

MD represents a metric according to (5.1) and supports the usage of features very well. This advantage directly costs memory c as the inverse variance-covariance matrix has to be held in memory for every class, see (5.8). This supplemental information increases the amount of data that has to be computed and transmitted compared to the ED. The algorithm is easily implemented and has a run time of $\mathcal{O}(2f^2)$, which adds just a constant to the quadratic runtime. The offered decision boundaries for the MD are the linear and curved boundaries which allow completely individual adjustment of every dimension.

In order to be able to generate well-estimated parameters for the variance-covariance matrix for the MD, the data set size of the training data has to be significantly higher than the classifier dimensionality (number of used features). Otherwise, the variance-covariance matrix is ill-conditioned, which leads to an increased estimation error when calculating the inverse matrix [106]. If the feature number has to be small, as is the case for our area of research (WSN), the features' precision needs to be very high for every single value in order to sustain accuracy of the variance-covariance matrix [103]. The given statements lead to the assumption that the MD could be negatively influenced by our event detection parameter: low number of feature, low number of training data sets, and furthermore, a feature set with varying numbers of features and therefore missing features per class. Missing features are common in our applications because different event classes are represented by differently sized feature vectors; further details on this topic are given in Section 5.7.3. In addition, packets may not necessarily be delivered successfully in a WSN due to the unreliable nature of radio communications [107]. Typically, retransmissions help increase the PDR, but if the Acknowledgement Packet (ACK) is not received by the initial sending sensor node, it is still uncertain whether a packet has reached its target node or not. Missing features negatively influence the complete inverted variance-covariance matrix [108–110]. To investigate the performance on our applications directly, we evaluate the classification performance of both distance measures with a training data set of our fence surveillance system.

5.1.3 Distance Measure Comparison

In order to compare both introduced metrics, we investigate in the following how they perform with ascertained training data from our fence monitoring example (cf., the experiment setup in Section 7.1.1) introduced in Section 7.1.2 that provides missing features as introduced in Sec-

Criteria	Euclidean Distance	Mahalanobis Distance
Metric	+	+
Feature Compatibility	0	+
Reduced Communication	+	+
Fast Evaluation	+	+
Memory Consumption	+	0
Missing Feature Tolerance	+	-
Small Training Set Tolerance	+	-
Decision Boundaries	0	+

Table 5.1.: Comparison of the applicability of the Euclidean and Mahalanobis distance within an event detection system for Wireless Sensor Networks.

tion 5.7.3. First, we investigate the classification results of both the [ED](#) and the [MD](#) by training the classifier with all available training data that reflect the *optimal conditions*. The conditions are optimal because we reuse the training data for the detection process too. Hence, the results show the theoretic maximum possible detection rate with this training data, as all events are already trained and known by the system.

Second, we investigate the classification results of both distance measures by training the classifier with the classic [LOOCV](#) (introduced in Section 4.2.3.6) which reflects the approximated real world conditions. This approximation is reasonable as for every training data that we check with the [LOOCV](#) principle, the classifier needs to be trained with all other training data. The selected training data was not allowed to be part of the training for the reason that this training data has to be new to the classifier during the detection process in order to reflect the real world situation where all events are not known in their concrete realization.

It can be seen in Figure 5.1 that theoretically, the [MD](#) has a higher potential to detect events of our fence application correctly (Optimal Conditions). Nevertheless, as soon as we start to investigate our application with a more realistic (but still theoretic) approach by applying [LOOCV](#), all metrics of the [MD](#) compared with the [ED](#) are clearly negatively influenced. This means that the [MD](#) is, under the circumstances of our [WSN](#), much more sensitive as our used data are not in the ideal form the [MD](#) needs them to be in order to tap the full potential. Based on the theory, [MD](#) can be recommended, but under the more realistic circumstances of the given specific training data we can not recommend the [MD](#) for our system.

By summing up the benefits and drawbacks collected above within Table 5.1, we can see that the [MD](#) has some good points to offer. We have to take into account the circumstance that we use an energy restricted, unreliable [WSN](#) that will have noticeably fewer than 100 training sets per class with different feature vector lengths. As an advantage, the [MD](#) takes the feature's relations into account, but we could not verify that this impact could exceed the negative influence of the missing feature and small training set intolerance. Both distance measures have to distribute the same amount of feature data and a [CN-Feature](#). Theoretically, a better decision boundary is offered by the [MD](#), but as shown in Figure 5.1, the application specific data are well separated by the [ED](#) as well. The [ED](#) performs better and more stable and is therefore recommended for such approaches, although there might be some specific, probably more laboratory-oriented Distributed Event Detection applications that would benefit from the [MD](#).

5.2 Information Fusion Approaches

Our proposed Distributed Event Detection system is based on four fundamental hierarchical structured data processing and potentially fusion approaches. We investigate each of these information fusion approaches to show which fusion approach has the most impact on the SNs' lifetime under varying setup conditions. If the Distributed Event Detection is outperformed towards energy efficiency it is recommended to switch to the best-performing fusion approach.

In the following sections we introduce the information fusion approaches with which we compare our approach. The current work of research Chapter 2 shows four information fusion approaches, Raw Data Fusion, Feature Fusion, Decision/State Fusion, and Classification Fusion. We reuse all of these approaches but combine the Decision/State Fusion and the Classification Fusion within the Classification Fusion due to the identical amount of data that has to be transferred in our evaluation assumptions. We assume that a single packet for each Classification or Decision/State that results from the event detection process needs to be transferred to the BS.

We investigate two additional information fusion approaches: our former information fusion approach – now called *Multiple In-network Fusions* (introduced in [55]) – and our own proposed approach, called *Distributed Event Detection*.

List of investigated information fusion approaches:

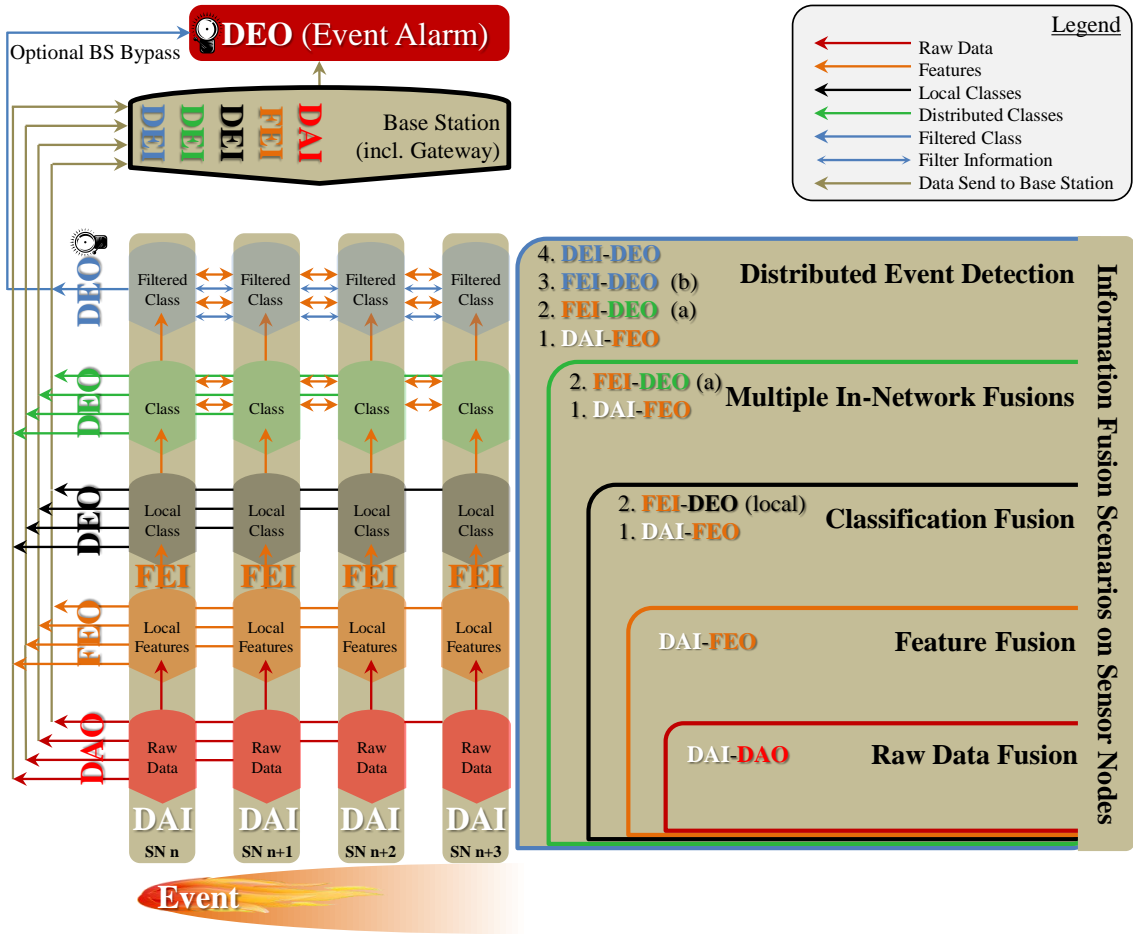
1. (Section 5.2.2) Raw Data Fusion
2. (Section 5.2.3) Feature Fusion
3. (Section 5.2.4) Classification Fusion (representing Decision/State and Classification fusion)
4. (Section 5.2.5) Multiple In-Network Fusion (former approach)
5. (Section 5.2.6) Distributed Event Detection

5.2.1 Information Fusion Preliminaries

Data can be compressed by exploiting spatial correlation between sensor nodes if at least two sensor nodes gather correlated sensor data or create correlated features [111].

As we use the pattern recognition-based event detection approach in order to investigate events from multiple different perspectives, our goal is to select uncorrelated features. During the training we already investigate and minimize possible redundancies within the exchanged data, which means that we need to extract only uncorrelated features in order to create a descriptive and compressed feature vector as introduced in Section 3.5.1.

In our case, every SN transmits the same features, not knowing which are its own relevant features. The transmitted features can theoretically be further reduced based on the aforementioned spatial correlation of the sensor nodes. For this, the sensor nodes have to know their relative position to the event – which allows figuring out any spatial correlated features – before they need to distribute the features. As the relative positions of the SN to each other can only be calculated with event related data that has to be exchanged first, this improvement can only be effective if a spatial correlation is known a priori. Indeed, there exists one case where we can prevent the distribution of data based on *constant spacial correlation knowledge*: If an event arises, the affected SNs still do not know the CN directly, which means that every affected SN has to extract the features for the CN as one unknown node from among all affected sensor nodes will take the role of the CN.



Input-output scheme of Dasarathy et al. applied on the investigated information fusion scenarios.

The Distributed Event Detection inherits the hierarchical structure of the investigated information fusion scenarios.

Figure 5.2.: Input-output scheme of Dasarathy et al. [13] applied to our hierarchical structured information fusion approaches.

The role of the CN consists of fusing all received features from the neighbouring SNs, which means that only those features of the neighbouring SNs need to be exchanged within the network. This reduces the number of redundant features to be sent to the CN by means of spacial correlation of the central node (cf., Section 5.7).

Duda [10], Niemann [11], and Web [12] postulate three main layers of information fusion: (i) data fusion, (ii) feature fusion, and (iii) classification fusion. Each of these layers handles the trade-off between the amount of data to manage and the potential information loss by reducing the amount of data to support a lower communication load. We want to investigate these layers by using the input-output scheme of Dasarathy et al. [13]. The authors introduced the Data-Feature-Decision (DFD) model by defining the principle of calling incoming data the *input* and outgoing data the *output*, with the input being fused to generate a condensed output. We will use these fusion schemes as they can be adapted to define different fusion approaches within a WSN, while others do not fit [5]. With help of the DFD model, multiple information fusion approaches can be generated and defined with the following simple definition rules [112].

- Data In > Data Out (DAI-DAO): In this class, information fusion relies on raw data and the result is therefore raw data, possibly more accurate or reliable.

- Data In > Feature Out (DAL-FEO): Information fusion uses raw data to extract features or attributes that describe an event.
- Feature In > Feature Out (FEI-FEO): FEI-FEO fusion works on a set of features to improve/refine a feature, or extract new ones.
- Feature In > Decision Out (FEI-DEO): In this class, information fusion takes a set of features of an event to generate a classification.
- Decision In > Decision Out (DEI-DEO): Decisions can be fused to obtain new decisions or to emphasize previous ones.

As we can see, these rules are hierarchically connected to one another. This means that every higher level information fusion is able to support all lower level information fusions; this is the concept we are following as well. As a simple example we want to use our own proposed distributed event detection system. To decide whether an event is worth notifying the BS about, our system touches upon raw data, features classification (first decision), and the final assessment of the necessity to send the notification to the BS or not. By having all these data available, we are therefore able to send these data to the BS on demand if necessary. This is true for all subsequently introduced approaches and adds significant value to the energy and lifetime evaluation. Although we are evaluating each approach as if it is in competition to all other approach, the resulting knowledge of each information fusion approach is available as an additional option for each information fusion approach that aligns itself on a higher level hierarchically.

We depict the subsequent information fusion approach and its linked capability derived from the inherent data fusion layers of the DFD model in Figure 5.2. We generate four classical information fusion approaches from the given input-output schemes in order to compare them with our own system which represent conceptionally the highest level of information fusion.

We do not use the FEI-FEO fusion in our considerations as it does not add a new level of data reduction to our concept but rather changes the data within its static level of data amount and can add or exchange new features.

Classical compression techniques, such as the Ziv-Lempel and Huffman families [Nelson and Gailly 1995], are not information fusion methods, as they consider only the coding strategy used to represent data regardless of their semantics [112]. We do not investigate these techniques directly here as they can be applied to any type of payload communication with standard routines. A broad overview of data compression approaches in WSNs in general can be found in [113].

5.2.2 Raw Data Fusion

The first and lowest fusion layer is the raw data layer where the originally gathered and preprocessed raw data have to be transmitted to a BS in order to be fused. The detail level of this information is the highest possible but the communication effort is obviously the highest possible. The measured, filtered, and segmented raw data are processed at a control center. We can expect that this approach will not have an advantage in energy expenditure performance or communication load towards other approaches. Nevertheless, a large number of WSN scenarios use this approach and will use it in the future. Most applications using this fusion type want to collect all available data for a later and most precise evaluation rather than having a system that uses a lossy compression and reduces the amount of data by eliminating information permanently, especially redundant

information. For applications where the complete raw data set has to be preserved, we can list research reasons like an a posteriori study, documentation, or diagnostic reasons in order to cover the data completeness as a possible requirement to use the Raw Data Fusion approach.

In the Raw Data Fusion approach, the **WSN** does not perform any data aggregation within the network. Instead, the control center has to perform the complete pattern recognition algorithm off-line after receiving all relevant raw data packets from the **WSN** by performing the **DAI-FEO** combined with the **FEI-DEO** fusion.

5.2.3 Feature Fusion

The second fusion layer represents the mid-layer with the Feature Fusion approach where features have to be transmitted to a **BS** in order to be fused. Depending on the features, the level of information detail may vary within a broad range, but with features as a general fusion base, we have a huge potential to characterize events on a high quality level. *The function of any representation scheme is to capture the essential features of a problem domain and make that information accessible to a problem-solving procedure [68].*

The Feature Fusion approach increases the abstraction of the Raw Data Fusion approach to highly descriptive features, as introduced in Section 3.2 and many pattern recognition references [10–12]. Highly descriptive feature extraction **DAI-FEO** reduces the data traffic to the control center without any in-network aggregation. An appropriate feature fusion and classification has to be performed on the control center in order to evaluate the events offline with a **FEI-DEO** fusion. The Feature Fusion approach saves the in-network communication in contrast to the later introduced Classification Fusion approach as well as the Distributed Event Detection approach. However, each **SN** that is exposed to the event has to send the features to the **BS** through the multi-hop network because of the Feature In (**FEI**) approach. This approach will show a significant amount of data traffic savings in contrast to the Raw Data Fusion approach. However, this approach will not scale with an increasing problem size as the data traffic increases more than linear.

5.2.4 Classification Fusion

The last layer is the classification layer that leverages the approach to send the classification results only of the individual sensor nodes to a **BS**, which will further reduce the amount of communication.

FEI-DEO (local) is performed by a typical local classification based on the created features of one sensor node. This means that this **FEI-DEO** (local) uses only locally extracted features and represents a classification result of the local view of the **SN**. We mention this because of the subsequently re-used information fusion concept of fusing features **FEI-DEO**. **DEI-DEO** is performed on the **BS** with the individual incoming classifications of the sensor nodes, e.g. with a majority vote.

The classification fusion layer abstracts the level of information detail of all information fusion approaches most strongly, as any detail information is already condensed to one specific class for each sensor node involved. The classification accuracy for a Classification Fusion approach can not outperform a Feature Fusion approach, as the feature fusion will preserve all feature data until they are used for the final classification, which allows assessing the events in its context. The Classification Fusion approach performs a lossy compression which loses the relation of the features to neighboring sensor nodes.

5.2.5 Multiple In-Network Fusions

As well as the previously mentioned Feature Fusion and Classification Fusion approach, the Multiple In-Network Fusions approach extracts representative features from incoming raw data **DAI-FEO** after an event has occurred. The Multiple In-Network Fusions approach performs the data fusion within the network, albeit at every single sensor node after distributing the features via broadcast from every affected sensor node to the 1-hop neighborhood, which follows the concept of **FEI-DEO** (a). This approach transmits each classification to the control center as the optimal classification can not be detected within the network with this approach. The in-network event detection can only be performed if events are always triggered at exactly the same location. For the event distribution, this means that every affected node is responsible for the same sub-event as during the training. For such applications the fusing **SN** can be assumed to be known as it does not change. For any other case, the **BS** needs to filter plausible results, which might be done with the aid of network cameras that record fence activity when triggered (**DEI-DEO**) in order to map the event location to the best performing sensor node, as described in [55]. The additional use of a boundary distance event rejection can reduce the amount of sensor nodes communicating with the **BS** to an average of 65 % [75]. This means we assume that only 3.5 **SN** on average will send event results to the **BS**.

5.2.6 Distributed Event Detection

We apply the already introduced Distributed Event Detection System on the input-output scheme of Dasarathy et al.; to do this we recap the Distributed Event Detection in a compact description in order to make it easier to follow the evaluation:

In the presented approach, data fusion within the network is performed by extracting representative features after an event has occurred (**DAI-FEO**). The gathered features and further model parameters are transmitted between all triggered nodes in a 1-hop neighborhood. The classification result is calculated (**FEI-DEO** (a)) on all **SNs** that passed the **CN-Filter**. Under exceptional circumstances it is possible that multiple **SNs** pass the central node filter. In this case, these nodes exchange the additional quality feature value to decide which of these nodes classifies with the smallest amount of model errors (**FEI-DEO** (b)), see 4.3.2. Based on our application observations during the fence surveillance experiments, we found that in at most 10% of the events, only two **SNs** may pass the **CN-Filter**. The **SN** that finally passes the quality filter sends a packet with the classification result to the **BS** only if the application filter (Fig. 4.2) confirms the necessity. The internal decision creation is done by mapping the classification result to a certain type of actions like *send alarm* in case of a *critical event* like an intrusion, or *do not react* in case of a neutral event like wind shaking the fence. These events may be evaluated independent of parameters like date, daytime, and current weather situation. This information fusion can be referred to a **DEI-DEO** based information fusion and, in case of a critical event, causes an alarm packet transmission through the multi-hop network to the **BS**. Otherwise, it is not mandatory to brief the **BS** about the event and in contrast to all other approaches this packet will not be sent. In addition, it is possible to bypass **BS** in order to send the alarm directly to the alarm-giving instance or to an autonomous instance within the **WSN**. The option to contact an autonomous instance within the **WSN** enables a new level of energy efficient self-deciding systems.

5.2.7 Classification Accuracy Assumptions

Table 5.2.: Classification Capabilities derived from the Input Detail Level for Fusion

Approach	Input Detail Level for Fusion	Classification Capabilities
Raw Data Fusion	Data In (DAI)	++++
Distributed Event Detection	FEI	+++
Feature Fusion	FEI	+++
Multiple In-Network Fusions	Decision In (DEI)	++
Classification Fusion	DEI	+

We assume the classification capabilities for each of the data fusion approaches mentioned above because a profound investigation of multiple classification algorithms is beyond the scope of this thesis. We derive the maximum possible classification performance from the available information detail that is given from the data input type which ranges from DAI over FEI to DEI. We compare the approaches introduced above in Table 5.2 by representing the relative maximum classification ability. We can see that raw data (DAI) represents the highest amount of information detail and the abstract decision input of classifications DEI represents the lowest amount of information as all details are discretized into one statement. The Classification Fusion creates classification results at every sensor node that have to be fused at the BS e.g. with a majority vote. The classification capabilities of the Multiple in-Network Fusions approach depend very much on the knowledge of the position of the CN. If the CN can be determined at the BS, the accuracy of the fusion result is comparable to our approach, if not, it is more comparable to a performed majority vote at the BS of the Classification Fusion and is therefore rated in between both approaches. The Distributed Event Detection approach is able perform the fusion only at one sensor node and delivers only one classification result from the network by discovering the CN automatically. Both the Features Fusion approach and the Distributed Event Detection approach use features of multiple sensor nodes to build a comprehensive view of the arising event and are to be equally assessed concerning the classification capabilities. The most promising classification capabilities are given by the Raw Data Fusion approach as we do not lose any detail information based on any prior signal discretization.

5.3 Payload in Large Scaled WSNs

For each of the approaches introduced above in Section 5.2.2 to Section 5.2.6, we analyze the payload that needs to be transmitted within the network to notify a classified event, as introduced in [5]. Depending on the hop, we show how many packets need to be transmitted in the network on average to classify the arising event and additionally to report the classified event to the control center.

5.3.1 Payload Calculation

For all information fusion approaches introduced above, we have to define the same specific setup for data sampling at 100 Hz, the event duration of 10 s and a three axis sensor that gathers information

with a 16 bit data type for each acceleration sensor axis x , y and z . A time stamp is taken into account with 4 byte. Hence, the data created for one standard event needs 6004 byte of memory.

We use our example fence surveillance system to compare the information fusion approaches and we use the following numbers for our subsequent theoretic considerations of the network scale effect of the payload. Events of different event classes do not necessarily trigger the same number of nodes. In our example fence surveillance experiments, from a maximum amount of 6 affected sensor nodes, on average 5.4 SNs are affected by an event. The reason is that all events that open the fence as a door trigger only five sensor nodes and all events that close the fence trigger four sensor nodes. The reduced spatial event propagation arises because of a physical disruption by breaking the chain of fence segments into two separate physical entities during the opening or closing of the fence, see Section 5.7.2.

If a packet is not filled to the maximum capacity, we fill that packet with dummy-data to a size of 44 byte for Unicast communication to the BS and 48 byte for Broadcast communication within the network. This is necessary to assure constant receive and transmit times for the energy-aware Wake-On-Radio communication, as introduced in [102] in detail. This technique has no negative effect for our system.

The packets themselves are, in general, composed of a 4 byte time-stamp and the event data. A CN-Feature (2 byte) is only created for the in-network communication Distributed Event Detection. As each feature needs 2 byte, the aforementioned SFSM could fill up a packet for the distributed Event Detection approach with up to a maximum of 19 to 20, depending on the fusion approach features for broadcast based feature distribution.

The Distributed Event Detection optimizes the necessary amount of features to be exchanged, hence the SNs send only features that do not characterize a hypothetical CN-position of the SN. Hence, for this example investigation the 9 non-CN-features are distributed and then fused with the CN-features on the filtered CN. As a result, SFSM delivers us prototypes for the event classes with exactly 9 features (see Section 5.7.1) that are to be transmitted during feature distribution by each node. The CN fuses these features with its own features – 10 for our example fence surveillance. The Multiple In-Network Fusions did not support this efficient feature distribution during the time of publishing [55], however we assume that this technique is available for purposes of fairness.

Raw Data Fusion: For the Raw Data Fusion approach we assume the introduced 6004 byte of data that need to be transmitted to the control center for one single event and each affected node. With a packet size of 44 bytes we have to transmit 137 packets in total from every single affected sensor node. The simultaneously arising data traffic will cause a significant event-to-response-time increase, as the BS can receive and process data packets consecutively over a single communication channel. This means that the general expectations are that this approach will heavily lack in scalability of supported multi-hop network sizes.

In order to reduce the amount of data to be transmitted, we can use standard lossless data compression algorithms like Lempel–Ziv–Welch Codec (LZWC) or MinDiff which are able to compress time series data of about 58 % and above [114, 115]. By using the compression ratio of 58 %, the amount of data to be transmitted is reduced to 2522 byte or to 58 Unicast packets in total for each SN and event to be sent to the BS. We use these numbers as a basis for further calculations concerning the Raw Data Fusion approach in order to leverage the Raw Data Fusion approach to a more deployable approach.

To calculate the total network payload for the Raw Data Fusion approach, a total payload of

2522 byte or 58 Unicast packets in total for each SN is assumed. For every hop this payload is accumulated in order to reflect the amount of data the whole network has to deal with.

Feature Fusion: For the Feature Fusion we assume that every affected SN (5.4 on average) creates for each event the number of features (19 features * 2 byte) that our own approach would produce. Additionally we add to each packet a 4 byte timestamp which results in a 42 byte payload or 1 Unicast packet to be sent to the BS for each event and affected SN. This can be summed up to a total event payload of 226.8 byte or 5.4 Unicast packets on average to be sent to the BS for each event.

To calculate the total network payload for the Feature Fusion approach, it uses a total event payload of 226.8 byte or 5.4 Unicast packets on average to be sent to the BS for each event, which is again accumulated for each hop.

Classification Fusion: For the Classification Fusion approach we assume that every affected SN (5.4 on average) creates for each event 1 single classification * 1 byte. We add a 4 byte time-stamp which means that a single affected SN creates 5 byte of data or 1 single Unicast packet for each event and affected SN to be sent to the BS. This can be summed up to a total event payload of 27 byte or 5.4 Unicast packets on average to be sent to the BS for each event. To calculate the total network payload for the Classification Fusion approach it uses a total event payload of 27 byte or 5.4 Unicast packets on average to be sent to the BS for each event.

Multiple In-Network Fusions: For the Multiple In-Network Fusions approach we assume that every affected SN (5.4 on average) creates for each event the number of features (19 @ 2 byte) while only 9 of them plus a 4 byte time-stamp have to be distributed within the network. This results in a 22 byte payload or 1 Unicast packet to be broadcast in the 1-hop area for each event. This can be summed up to a total in-network payload of 118.8 byte or 5.4 Broadcast packets on average to be broadcast.

As mentioned, we can assume that 3.5 SN on average will send event results to the BS, hence, each of these nodes will create a packet comprising the classification result of 1 byte plus the 4 byte time-stamp which results in a payload of 5 byte or 1 single Unicast packet for each event and filtered SN to be sent to the BS. This can be summed up to a total event payload of 17.6 byte or 3.6 Unicast packets on average to be sent to the BS for each event.

To calculate the total network payload for the Multiple In-Network Fusions the in-network communication of the features is just caused once with 118.8 byte or 5.4 Broadcast packets on average. For every hop we add to this base-load the data that strains the network and that has to be sent to the BS. A total event payload of 17.6 byte or 3.6 Unicast packets on average is therefore added to the base-load for each event.

Distributed Event Detection: For the Distributed Event Detection approach we assume that every affected SN (5.4 on average) creates for each event the numbers of features (19 features @ 2 byte) while only 9 of them plus a 4 byte time-stamp and an additional CN-Feature of 2 byte have to be distributed within the network. This results in a 24 byte payload or 1 Unicast packet to be broadcasted in the 1-hop area for each event. This can be summed up to a total in-network payload of 129.6 byte or 5.4 Broadcast packets on average to be broadcast.

The Distributed Event Detection sometimes needs to initiate a *second in-network communication* if the **SN** have a very similar perspective on the event, which means that the **CN-Filter** is identical in its expression for multiple **SN**. Hence, the **CN** needs to be filtered based on the introduced Quality-Filter data (see Section 4.3.3.1) in order to finally filter the **CN** within the network. This process needs to exchange additional packets within the network. For this purpose, 2 or up to 6 of the affected **SNs** have to communicate with each other. For each **SN**, the packet comprises a 4 byte time-stamp and a 2 byte quality feature which results in a single packet of 6 byte. Depending on the number of **SN** with the same **CN-Filter** value, the sum of the second in-network communication increases from 12 byte for the minimum of 2 concerned **SNs** up to 36 byte for all 6 **SNs**.

The resulting **CN** creates the classification and sends a single notification packet to the **BS** comprising the classification result of 1 byte plus the 4 byte time-stamp and plus 2 byte for the additional quality value for future and additional evaluation purposes at the **BS**, which results in a payload of 7 byte or 1 single Unicast packet for each event to be sent to the **BS**. As mentioned, we added an application filter (see Section 4.3.3.1) that only sends data if a critical event arises. Having this in mind it becomes clear that depending on the application requirements this concept can be very beneficial.

To calculate the total network payload for the Distributed Event Detection approach the first in-network communication of the features is just caused once with 129.6 byte or 5.4 packets on average. The second in-network communication of the additional Quality-Filter data as well as the event notification depend on two scenarios which we introduce in the following, the standard and worst case scenario.

- For the worst case setup we assume that all 5.4 **SN** cause the second in-network communication with a probability of 100 %. The second in-network communication therefore causes 1.2 byte and 0.2 packets on average for the standard scenario which results in a constant base-load of 162 byte or 10.8 packets on average for the in-network communication as the base-load. The final classification notification is performed in the worst case setup with a probability of 100 % of an event being critical. The notification packet payload causes an additional payload of 7 byte or 1 packets that we add to the base-load at every hop.
- For the standard case setup we assume that only 2 **SN** cause the second in-network communication with a probability of 10 %. The second in-network communication therefore causes 1.2 byte and 0.2 packets on average for the standard scenario which results in a constant base-load of 130.8 byte or 5.6 packets on average for the in-network communication as the base-load. For every hop, we add to this base-load the event notification that has to be sent to the **BS**. The final classification notification is performed in the standard case setup with a probability of 50 % of an event being critical. The notification packet payload comprises 7 byte or is 1 single Unicast packet for each second event to be sent to the **BS**. This causes an additional payload of 3.5 byte or 0.5 packets that we add to the base-load at every hop.

5.3.2 Payload Evaluation

In Figure 5.3 we present the standard case setup and in Figure 5.4 we present the standard case setup for all introduced fusion approaches and how it turns out to affect volume of data (byte) that need to be transmitted for increasing hop distances.

In Figure 5.5 we present the standard case setup and in Figure 5.6 we present the standard case

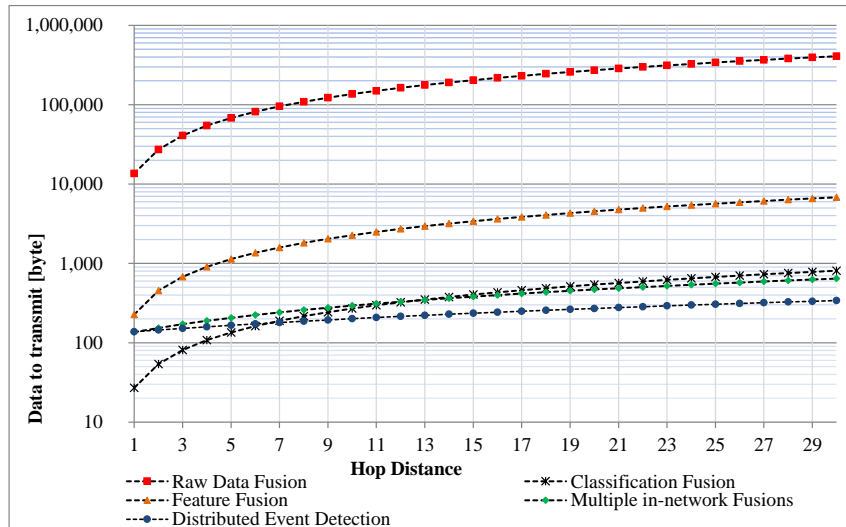


Figure 5.3.: [Standard Case] Comparison of payload in [byte] expected for varying hop distances. Critical events arise with probability of 50 %.

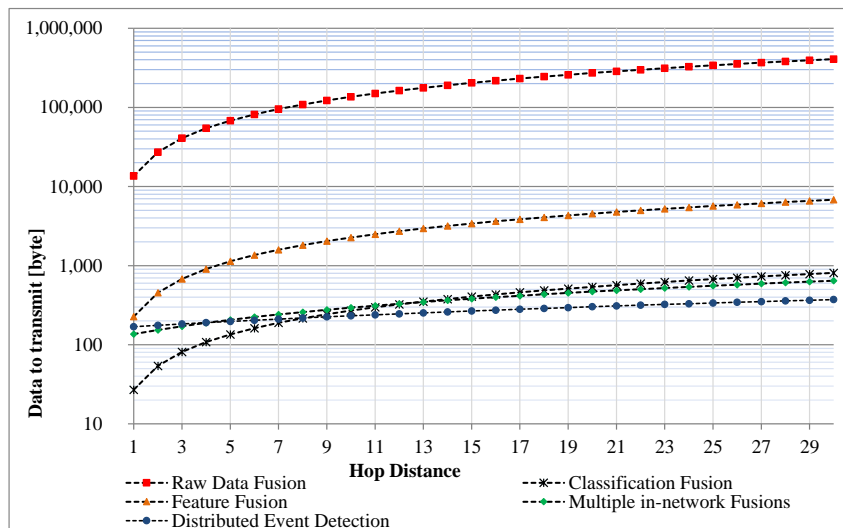


Figure 5.4.: [Worst Case] Comparison of payload in [byte] expected for varying hop distances. Critical events arise with probability of 100 %, adapted from [5].

setup for all introduced fusion approaches and how it turns out to affect the number of packets that need to be transmitted for increasing hop distances.

The handled volume of data in Figure 5.3 is outperformed (undercut) by the Distributed Event Detection approach from a hop distance of 7 and above.

If we investigate the packets that have to be transmitted, the Distributed Event Detection clearly outperforms (undercuts) every other introduced approach from a hop distance of two and above if the packets to be transmitted are compared in Figure 5.5.

The reason for the late performance in the worst case setup is the initial one-hop base-load of 11.8 packets on average in the Distributed Event Detection approach. That base load is higher than the payload of the Feature Fusion approach and the Classification Fusion approach but increases only linearly by one packet per hop.

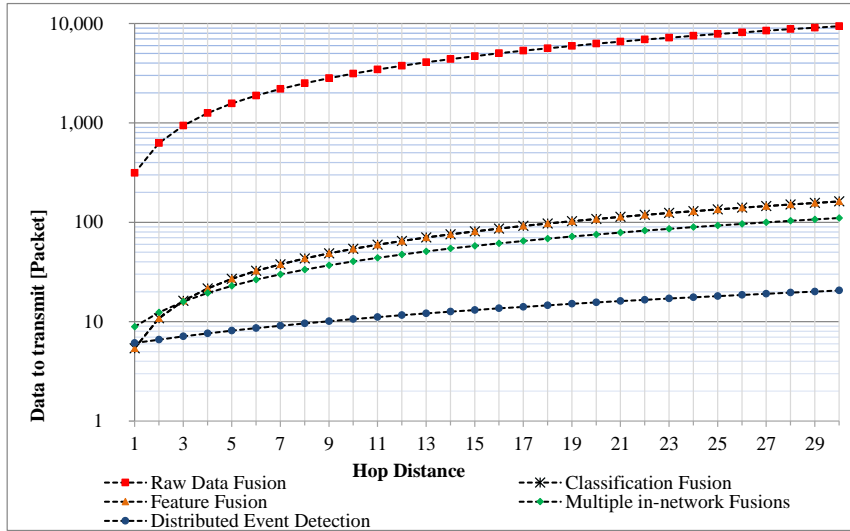


Figure 5.5.: [Standard Case] Comparison of payload in [packets] expected for varying hop distances. Critical events arise with probability of 50 %.

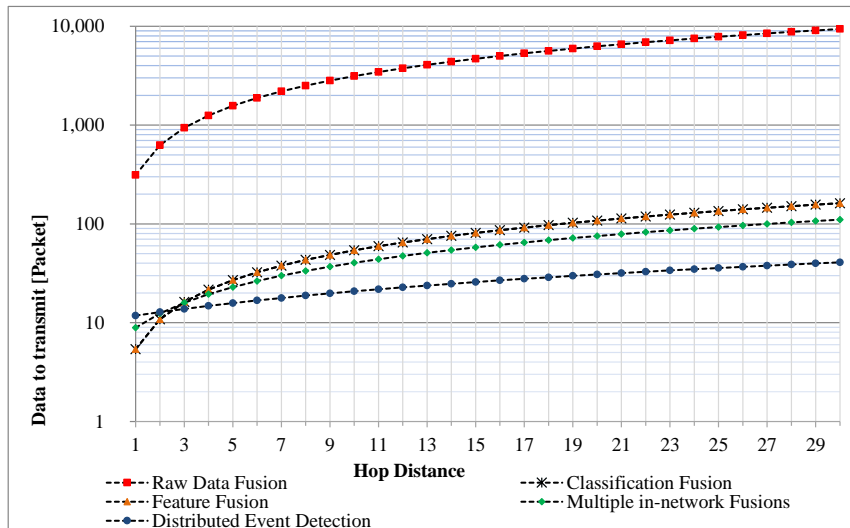


Figure 5.6.: [Worst Case] Comparison of payload in [packets] expected for varying hop distances. Critical events arise with probability of 100 %, adapted from [5].

As expected, the Raw Data Fusion approach puts the largest strain on the network as it sends raw data chunks to the control center. Feature Fusion approach and Classification Fusion approach send fewer packets compared to the Distributed Event Detection approach in one-hop environments. Hence, the applicability of feature extraction or local sensor node classification and subsequent fusion on a BS is more reasonable for one-hop networks.

The Feature Fusion approach is clearly a better choice than applying the Classification Fusion approach regarding network load and can be recommended as a simple, efficient and lightweight approach for small networks as well. Nevertheless, the Feature Fusion approach is outperformed by the Multiple In-Network Fusions approach by WSNs with four or more hop distance based WSN.

With increasing hop distance, the Distributed Event Detection approach benefits more and more from the reduced communication with the BS and thus heavily reduces the network load.

For simple, small and lightweight networks, the Feature or Classification Fusion approach should be preferred.

Even under these extreme conditions of the worst case setup, the Distributed Event Detection approach is able to outperform the compared approaches for network sizes of 3 hops and above, see Figure 5.6. Although the overall created number of bytes to be transmitted is lower for the Classification Fusion approach up to 8 hops as well as for the Multiple In-Network Fusions approach up to 4 hops (see Fig. 5.4), the number of packets to be transmitted is an important key factor in WSN communications as every communication costs additional energy.

The results show quite clearly that from 2 hop distance (see Fig. 5.6) upwards for the standard case setup and from 3 hop distance upwards for the worst case (see Fig. 5.4), the network load is reduced by our approach compared with all information fusion approaches introduced above. Smaller networks with 1 hop distances will only benefit from the Classification and Feature Fusion approaches as well.

5.4 Lifetime Extrapolation Concept

In the following Section, we compare the energy consumption and resulting lifetime of the CN and the RN (see Figure 1.2) of the introduced approaches from Section 5.2.

The RN reflects a SN in the WSN which is responsible for transferring packets wirelessly with the last hop to the gateway node at the BS. The RN eventually forwards all packets that have to be transferred to the BS. If more RNs are available, the network load is shared between these SNs. In order to cover the worst case, we assume that the WSN uses one common RN for this service. If the RN becomes inoperative or overloaded, the WSN has to find a new route with a possibly increased number of hops to the gateway. If no alternative route is available the WSN can not communicate with the gateway anymore. Hence, the RN represents a natural bottleneck in typical concast oriented WSNs [5, 116].

In order to represent the network load and its consequences for the whole network, the RN is investigated as a representative for an extensively stressed sensor node that becomes inoperative at first. No other node within the WSN will be stressed more than the RN as the network load is either shared between multiple nodes or the whole traffic caused by the event detection is routed through this node in the worst case. The first node that becomes inoperative is very important as an indicator for the lifetime and thus efficiency of the event detection approach. As soon as the first sensor nodes becomes inoperable the network is not able to communicate with the BS by using the shortest route. This indicates the point of time from which on the WSN starts to suffer from reduced connectivity, less efficient communication routes or a reduced number of measuring points or both.

In order to represent the network load and its consequences for the detecting SNs, the CN is investigated as a representative for an extensively stressed SN that may become inoperative early on. The CN is used as the SN that gathers data and sends the most data to the BS in all information fusion approaches. As all event observing SNs in the fusion approaches are doing the same, except for the Distributed Event Detection, it is important to clarify that the name CN is only relevant for the Distributed Event Detection. The role of the CN additionally sends a notify packet to BS in case of an alarm while all other observing SN do not send a notify packet in the Distributed Event Detection. In contrast, the CN for all other investigated information fusion approaches can be every observing SN without any exceptions as all gather and send their data

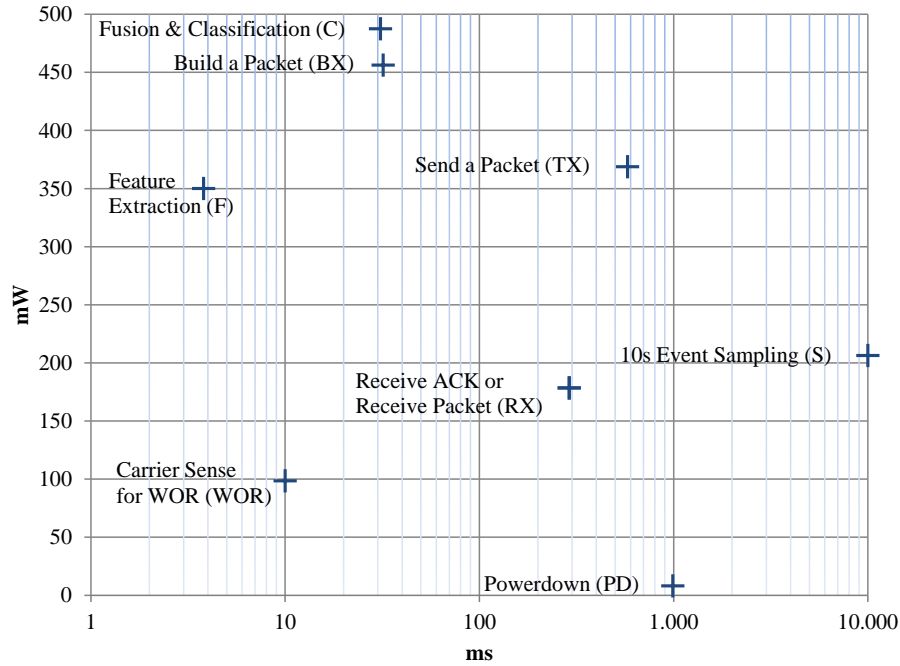


Figure 5.7.: Energy consumption of essential event processing tasks during the event detection as a basis for our theoretical considerations.

(raw data, features, classes) to the BS.

The arising question now is how the different information fusion approaches introduced in Section 5.2 influence the lifetime of the CN and the RN depending on various influencing parameters to be investigated. We answer this question in the following sections.

5.4.1 Energy Consumption

Our lifetime evaluation investigates various settings for varying numbers of events per hour that could arise at the CN. We assume that every event arises exactly at the same CN in order to present a very repeatable situation. If one single SN is assigned to the CN for all events, this is a very unfavorable situation with a high energy consumption. Nevertheless it is advantageous for the evaluation as this demonstrates the worst case a Distributed Event Detection for WSNs could be exposed to. Depending on the real world application, event locations should vary, but it is not known in which way. We can say that the situation in the real world will not be worse. Some event detection approaches as body area networks for rehabilitation exercises will have a fixed CN while a fence or a bridge will have an event location dependent CN. If events arise at distinct locations, the CNs lifetime will increase because of the distribution of the in-network communication and event detection task over the whole network.

In [102], it is assumed that 5 events per hour arise at the fence in one position. Additionally, a keep-alive-packet is sent every 20 minutes. We extend this investigation from 5 to 500 events per hour. Applications with such an intense amount of events could be applications for border control with several hundreds of kilometers of fence. Second, we do not investigate the number of keep-alive-packets as the network load depends on the network size, which will not be examined in detail within this thesis.

For all theoretic evaluations the packet loss rate is based on the individual radio communication

Table 5.3.: Detailed list of measured and derived energy consumption values of event processing tasks during the event detection as a basis for our theoretical considerations.

Event Processing Tasks (abbr.)	[mV]	[mA]	[ms]	[mW]	[mWs]
10s Event Sampling (S)	412.5	41.3	10000.0	206.3	2062.5
Feature Extraction (F)	700.0	70.0	3.8	350.0	1.3
Powerdown (PD)	16.1	1.6	990.0	8.1	8.0
Build a Packet (BX)	912.5	91.3	32.0	456.3	14.6
Send a Packet (TX)	737.5	73.8	580.0	368.8	213.9
Receive ACK or Receive Packet (RX)	357.0	35.7	271.0	178.5	51.8
Fusion & Classification (C)	975.0	97.5	31.0	487.5	15.1
Carrier Sense for WOR (WOR)	196.9	19.7	10.0	98.5	1.0

which ideally can transmit data under **LOS** conditions. Depending on the environment of the scenario the **WSN no LOS** is given – which is generally the case – and radio communication suffers from possible packet collisions and a decreased connectivity due to multiple interfering influences like attenuation, scattering, diffraction, reflection, or refraction, see [107]. We leave it to the reader to factor in the relevant packet loss rate for their scenario.

Every **SN** in the fence is assumed to be supplied by an individual 90,000 mWh battery pack (18,000 mAh at 5 V) and limits itself to a max lifetime of 418.8 days lifetime if no event arises because of a basic energy consumption of at least 8.95 mWs. This lower boundary of energy consumption is defined by the demand for Wake-On-Radio (**WOR**) duty-cycle, MCU-power down mode, monitoring sensors as well as the step down converter efficiency. Event transmissions and calculations lead to an application dependent maximum of events per hour, which is highlighted in the following paragraphs. The resulting energy measurements can be seen in Table 5.3 in detail.

For simplicity reasons of the subsequent lifetime calculation we assume that only software **ACKs** are available. This means that our **WOR** related transmission can not be interrupted by hardware **ACKs** as those are not available. In order to guarantee that the sender *A* catches the wake up period of the receiver *B* the sender *A* has to repeat the packet transmission to surpass the sleep period of the receiver *B*. The sleeping period of the **WOR** takes 542 ms in our exemplary setup and results in a 580 ms repetitive **WOR** related transmission for the sender *A*. The receiver *B* has to await the uninterruptible re-transmissions of the sender *A*. As soon as the receiver *B* recognizes via Carrier Sense Multiple Access with Collision Avoidance (**CSMA/CA**) that the sender *A* stopped sending, the receiver *B* can send the software **ACK** to the initial sender *A*. We assume that the receiver *B* will awake at half the sleeping period of **WOR** on average. This results in an average receive time of 271, ms for software **ACKs** or any other received packets.

All calculations are based on individual event processing task measurements of an example sensor board AVS-Extrem, introduced in [102], see Figure 6.2 in Section 6.2.1. To measure the energy consumption of the whole board as accurately as possible, we soldered a 10 Ω shunt resistor into the supply line which is powered by a reference voltage of 5 V. To measure the voltage of the shunt resistor, a Digital Sampling Oscilloscope (**DSO**) is attached. As the resistor and the voltage are known, we can calculate the value of the current and use it to calculate the electric power used by the sensor node over the time of one **DSO** sample. By integrating the electric power over the time of one system state, like Transmit (**TX**) or Receive (**RX**), we can exactly measure the energy used for each event processing task and can use this information to approximate the

energy consumption of the whole detection system in different information fusion approaches. The resulting energy consumption for every energy-demanding process is shown in Figure 5.7; cf. Table 5.3 for more detail. It shows that the energy-consumption is still dominated by communication tasks, but the duration of sampling can easily outperform the energy demand for communication depending on the application. During the event detection phase, the sensor nodes use the **MCU** power down mode, which also shuts down all internal peripherals. The wireless transceiver uses the **WOR** mode [102] and is able to process incoming data.

5.4.2 Lifetime Calculation Preparation

To calculate the energy consumption and lifetime for the introduced information fusion approaches, we measured the actual energy consumption for all operations to get data with the highest possible reliability. We use these measurements introduced above in order to extrapolate these measurements in theory to cover a higher scaled evaluation, see [116]. In our experiments, we use Micro Mesh Routing (**MMR**) [4, 102, 117] to utilize a reactive routing protocol. We assume that all network routes are already configured and known at the beginning of the experiments as we want to investigate the effect of the Distributed Event Detection approach on the lifetime. As **SN** for our Distributed Event Detection approach do not change their relative position to each other, we can assume to have a relative stable route table with only a small amount of maintenance necessary, which is ignored for our considerations as a route request would affect all information fusion approaches in the same way.

If we want to pick a theoretically perfect routing protocol for the Distributed Event Detection system this should be done depending on the application where two main concepts have to be addressed. The first concept is the communication in the neighborhood of the **SN** affected by the events. This neighborhood can be a logical neighborhood or a real physical neighborhood. An example suggestion for an efficient communication for such neighborhood situations is published by Mottola et al. in [118] where a **SN** defines the own logical neighborhood by processing static attributes such as the node type, or dynamic attributes such as sensor measurements with a rule-based expression. The second communication concept needs to handle a communication transfer back to the **BS** through the whole network. Multiple example routing protocols are available for transfer of data to the **BS** like the **RSSI**-based Power-Efficient Gathering in Sensor Information Systems (**PEGASIS**) [119]. More dedicated approaches like IPv6 Routing Protocol for Low power and Lossy Networks (**RPL**) [120] focus on transferring data from one or more **SNs** to exactly one destination like the **BS**. Maximize Unsafe Path Routing Protocol (**MPU**) [121] extends **RPL** under critical situations like node failure because of natural disasters.

The goal of the subsequent lifetime evaluation is to estimate whether the proposed concept of the Distributed Event Detection can hold its promising impact towards lifetime if the application parameters are changed to a wider range than our real world experiments will cover. In order to realize the extrapolation from measurements to a higher scaled evaluation, we introduce two types of equations, the **RN**-equations and the **CN**-equations to describe the energy consumption of both types of sensor nodes. We reuse the abbreviations of Figure 5.7 and Table 5.3 for our equation arguments to indicate which specific value we use for our ongoing evaluation.

The processes denoted by a capital \mathcal{E} -preamble are using the metric J or Ws as the energy descriptor while the \mathcal{P} -preamble denotes the power consumption using the metric W . Uni-cast packets which are sent over a **RN** to a **BS** are acknowledged by an **ACK** (E_{ACK}). In-Network

communication is not acknowledged as we use Broadcasts for this purpose.

$$E_{\text{Baseload}} = \mathcal{P}_{\text{PD}} * t_{\text{PD}} + (t_x - t_c) * \mathcal{P}_{\text{WOR}} + (\mathcal{P}_{\text{PD}} + \mathcal{P}_{\text{WOR}}) * t_{\text{LRest}} \quad (5.9)$$

All event detection processes are subject to the system's base load E_{Baseload} (see see Eq. (5.9)) which are hardware dependent and of a more constant nature. During power down mode, the **SNs** still consume power \mathcal{P}_{PD} for collecting acceleration sensor data to trigger a potential event and to wake up the **MCU** for about t_{PD} seconds. The time needed for communication t_c is subtracted from the whole processing time t_x in order to get the time period while **WOR** can be active - during communications **WOR** is typically inactive. To be able to react to incoming packets and to support **WOR** we have to perform carrier sensing every 542 ms [102] while no communication is performed \mathcal{P}_{WOR} . The left battery lifetime t_{LRest} is spend between the arising events while the system stays in the power down mode \mathcal{P}_{PD} and **WOR** mode \mathcal{P}_{WOR} .

The energy consumption for relay nodes is defined in the general equation E_{RN_g} (Eq. (5.10)) that allows us to replace the parameter M_{RN_x} with the individual modifier equations for the dedicated information fusion approaches (see Eq. (5.11) to Eq. (5.15)) in order to analyze parameters as varying numbers of events $\#_{\text{EV}}$ per hour. The information fusion dependent modifier is multiplied by the energy to receive data by a **RX**-packet (E_{RX}), forward a **TX**-packet (E_{TX}) to a **BS** and to receive an **ACK** (E_{ACK}) which again has to re-build E_{BX} and send to the **CN** E_{TX} . It is important to mention that for every packet that needs to be sent E_{TX} , the packet has to be built first, which costs energy E_{BX} . Even if the packet has to be forwarded, we have to re-build the packet (E_{BX}) as a new Destination of a Packet (**DST**) as well as a new Source of a Packet (**SRC**) has to be changed within the packet header for every new hop, hence we always have to add a build packet action for a **TX**-packet. This energy is multiplied by the number of events $\#_{\text{EV}}$ that are expected to arise per hour.

$$E_{\text{RN}_g} = M_{\text{RN}_x} * (E_{\text{RX}} + E_{\text{BX}} + E_{\text{TX}} + E_{\text{ACK}} + E_{\text{BX}} + E_{\text{TX}}) * \#_{\text{EV}} + E_{\text{Baseload}} \quad (5.10)$$

$$M_{\text{RN}_x} = M_{\text{RN}_{\text{raw}}} = 58 \quad (5.11)$$

$$M_{\text{RN}_x} = M_{\text{RN}_{\text{feature}}} = \#_{\text{SN}_{\text{EV}}} \quad (5.12)$$

$$M_{\text{RN}_x} = M_{\text{RN}_{\text{class}}} = \#_{\text{SN}_{\text{EV}}} \quad (5.13)$$

$$M_{\text{RN}_x} = M_{\text{RN}_{\text{multi}}} = 0.65 * \#_{\text{SN}_{\text{EV}}} \quad (5.14)$$

$$M_{\text{RN}_x} = M_{\text{RN}_{\text{distrib}}} = P_{\text{EVC}} \quad (5.15)$$

Eq. (5.11): As introduced in Section 5.2, our example fence experiments need to send 2522 byte for one single event and affected node to the control station for the Raw Data Fusion approach. This leads to exactly 58 packets that have to be sent by every measuring **SN** over the **RN** to the **BS**. Both the Feature Fusion (Eq. (5.12)) and Classification Fusion approach (Eq. (5.13)) send one packet for each event per hour to the **BS**.

The selected features can theoretically extend the maximum payload of 38 byte, however, this can be simply limited during the **SFSM** if the number of selected features exceed the limited memory capacities of the used **SNs**. Nevertheless, in our experiments we never exhausted this limit; for further details see Section 5.3.2.

The Multiple In-Network Fusions approach (Eq. (5.14)) is able to reduce the number of involved

SNs to 65% thanks to an event rejection process. The Distributed Event Detection approach (Eq. (5.15)) reduces the necessary communication by a certain probability that reduces the communication to events classified as *critical* only.

The energy consumption for the CN is defined in the general equation CN_g that allows us to replace the parameter M_{CN_x} with the individual modifier equations for the dedicated information fusion approaches (Eq. (5.20) to Eq. (5.24)) to analyze the above mentioned parameters e.g. varying number of events $\#_E$ per hour. First of all we add to every information fusion dependent equation the energy consumption of a sampling time of 10 seconds E_S . The event sampling E_S is performed by every information fusion approach.

The event sampling E_S plus the information fusion dependent modifier is multiplied by the number of arising events per hour $\#_{EV}$. The system's base load $E_{BaseLoad}$ is hardware dependent and calculated as shown for the M_{RN} -equations.

The feature extraction E_F is skipped only by the Raw Data Fusion approach (see Eq. (5.20)). The classification process E_C is performed by the Classification Fusion (see Eq. (5.22)), the Multiple In-Network Fusions approach (see Eq. (5.23)) and the Distributed Event Detection approach (see Eq. (5.24)). The Feature Fusion approach (see Eq. (5.21)) simply communicates all features from all affected sensor nodes to the BS over the relay node.

$$E_{CN_{RXTX1}} = (\#_{SNEV} - 1) * E_{RX} + E_{BX} + E_{TX} + E_{PD} \quad (5.16)$$

The number of SNs affected by the event $\#_{SNEV}$ influences the in-network communication $E_{CN_{RXTX1}}$ (see Eq. (5.16)) which is necessary to distribute the features and the event intensity between the SNs. Only the CN of the Multiple In-Network Fusions approach (see Eq. (5.23)) and the Distributed Event Detection approach (see Eq. (5.24)) need to send and receive information to and from SNs affected by the event.

$$E_{CN_{RXTX2}} = (\#_{CNF} - 1) * E_{RX} + E_{BX} + E_{TX} + E_{PD} \quad (5.17)$$

A second in-network communication $E_{CN_{RXTX2}}$ (see Eq. (5.17)) with the CN may be necessary for the Distributed Event Detection approach if the CN-Filter is passed by more than one SN $\#_{CNF}$. This communication distributes the classification quality label between the involved SNs with a certain probability P_{CNF} for this circumstance. This can increase the communication compared to all other approaches. Every in-network communication comes with a certain time period where we have to wait for the neighboring SN to distribute their packets. During this period the system is set to the power down mode E_{PD} in order to be reactive and energy efficient.

$$E_{RN_{TX}} = E_{BX} + E_{TX} + E_{ACK} \quad (5.18)$$

The CN can be a true CN or for approaches without a CN-Filter just any other node involved in the detection approach that sends data to the BS. As all sending nodes perform the same actions, we use the CN-description as a good representation for one of these nodes. This sending node (CN) transmits a data packet with raw data, features or classifications to the BS and receives an ACK which is assumed to be sent over the relay node RN which is expressed by $E_{RN_{TX}}$ (see Eq. (5.18)).

$$E_{CN_g} = (E_S + M_{CN_x}) * \#_{EV} + E_{BaseLoad} \quad (5.19)$$

$$M_{CN_x} = M_{CN_{raw}} = 58 * E_{RN_{TX}} \quad (5.20)$$

$$M_{CN_x} = M_{CN_{feature}} = E_F + E_{RN_{TX}} \quad (5.21)$$

$$M_{CN_x} = M_{CN_{class}} = E_F + E_C + E_{RN_{TX}} \quad (5.22)$$

$$M_{CN_x} = M_{CN_{multi}} = E_F + E_C + E_{CN_{RXTX1}} + E_{RN_{TX}} \quad (5.23)$$

$$M_{CN_x} = M_{CN_{distrib}} = E_F + E_C + E_{CN_{RXTX1}} + E_{CN_{RXTX2}} * P_{CNF} + E_{RN_{TX}} * P_{EVC} \quad (5.24)$$

Every affected **SN** in the Raw Data Fusion approach needs to send the sampled raw data via 58 packets to the **BSs** to the **RN** $E_{RN_{TX}}$ for each event $\#_{EV}$. The Distributed Event Detection approach sends only one packet with one classification from the **CN** to the **BS** if the event classification is assessed to be a critical event. The mapping of the classification to the status *critical* happens with a specific probability P_{EVC} and has the potential to reduce the communication $E_{RN_{TX}}$ compared to other approaches.

5.5 Lifetime Results & Evaluation

The following results and evaluations are based on the assumptions, measurements and equations previously introduced in Section 5.4.

By investigating the previously resulting **RN** and **CN** equations and comparing them to the most demanding tasks presented in Figure 5.7, the factors that affect the energy demand the most are radio transmissions via **TX** or **RX**. Event sampling, which may cost more energy depending on the sensing duration, only increases linearly with increasing network size or changing communication protocols. Hence we will investigate influencing evaluation parameters that have an effect on the communication in the first place.

We are investigating three types of setups in order to compare our Distributed Event Detection approach with all other information fusion approaches: (i) The *Worst Case Setup* describes a setup that stresses the Distributed Event Detection approach with a parameter setting that causes the highest possible energy consumption, (ii) the *Standard Case Setup* reflects a more realistic setup we may have to deal with in a real world situation, and (iii) the *Best Case Setup* represents more optimal conditions for the Distributed Event Detection.

We will describe the different cases individually in the appropriate section.

It is worth to mention that the Feature Fusion approach and the Classification Fusion approach perform nearly equally in all further investigations. Nevertheless, we show both approaches in all figures as we want to emphasize that not the data compression ratio (which is higher for the Classification Fusion approach compared with the Feature Fusion approach) but the individual communication concept affects the overall network load as a key factor.

As mentioned, our investigations are based on the assumption that each event arises at the same location within the **WSN**. Nevertheless, our approach benefits from the situation where events do not arise at the same location over and over again. As we want to know the applicability of our approach we have to take this situation into consideration, but we want to emphasize that under real world conditions, events will in most cases arise at more distributed locations within the network. This distribution does not reduce the overall energy consumption but spreads the consumption of the **CN** over all event locations which is very beneficial for the lifetime of the **CN**. Nevertheless, our investigation shows how the role of the **CN** or a detecting **SN** will affect the

energy demand by different information fusion approaches. The same situation arises for the RN, as it is not possible to guarantee that each packet has to be relayed by one single RN. In real world applications this network load can be more distributed over multiple RNs if multiple base nodes are integrated in the WSN concept, as introduced e.g. in [117].

We employ the introduced equations from Section 5.4.2 in order to cover the energy demand of all five introduced information fusion approaches and we use the energy consumption measurement results from Table 5.3 in Section 5.4.1 to construct the subsequent results.

It is more important to investigate the percentage differences in energy demands of the subsequently investigated information fusion approaches than to investigate the lifetime difference in e.g. days depending on a specific battery capacity. Hence, the ordinate (y-axis) is always given in % and shows the relative lifetime towards all other approaches. 100% lifetime always reflects the relative maximum possible lifetime in respect of the chosen setup. The ordinate (x-axis) is given in different metrics and shows the varied parameters of interest. The varied parameters change in the following discussion depending on the focus of our investigation and as a result of the creation of the abstract equations, we can use the following five parameters (a) to (e) in order to investigate the lifetime of the CN and RN in broader detail. Every information fusion approach is evaluated by two curves: one curve for the lifetime of the RN by using one geometric marker (e.g. one blue circle for the Distributed Event Detection) and one curve for the lifetime of the CN by using two geometric markers (e.g. two blue circles for the Distributed Event Detection). We varied the line style between solid and dotted in order to investigate RNs and CNs more visibly. Only the RN of the classification fusion violates this rule as it wouldn't be visible otherwise, hence, we picked another dashed plus dotted style for that purpose.

- a) Number of arising events per hour ($\#_E$)
- b) Number of SNs affected by an event ($\#_{SN_{EV}}$)
- c) Probability of a critical event (P_{EV_C})
- d) Probability of passing the CN-Filter (P_{CNF})
- e) Number of SNs passing the CN-Filter ($\#_{CNF}$)

For the results of the parameter $\#_{SN_{EV}}$ and $\#_{CNF}$ we use points to represent each SN at the abscissa without a connecting line between the SNs' numbers as we do not want to interpolate between two natural numbers of SNs. For all other parameters we prefer the solid line as a graph depiction.

If the lifetime function of an information fusion approach does not continue until the lifetime of the represented SN reaches 0%, the processing time is consumed and no further actions can be performed by that SN. E.g., if the number of events per hour increases, the processing time to handle this increasing number of events limits the maximal possible number of manageable events per hour.

5.5.1 Lifetime Evaluation - Events per Hour

In order to investigate the lifetime of the CN and RN within the information fusion approaches, we increase the number of arising events per hour by varying parameter $\#_E$ from 0 to 500 events per hour on the abscissa. We pick a neutral setup of $\#_{SN_{EV}} = 5.4$ SNs that are affected on

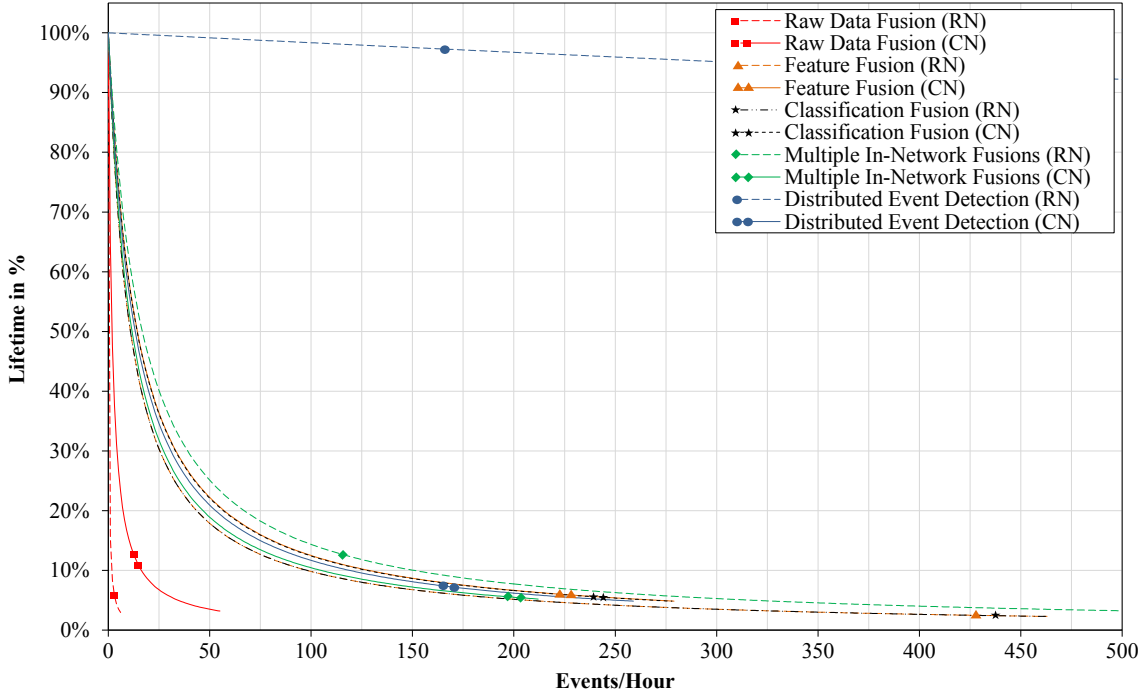


Figure 5.8.: [Best Case] Lifetime investigation with varying parameter $\#_{\mathbb{E}}$ to investigate the influence of the number of arising events per hour, adapted from [5].

average by events as we could observe this parameter during our experiments with an example fence surveillance system deployment that uses our Distributed Event Detection system.

Applications with a sparsely arising events will cause a lower energy demand compared with applications having a more regular occurrence of events because of the increased communication initiated by each event. Increasing numbers of events per hour may limit the applicability for the different approaches; we discuss these limits in the following.

5.5.1.1 Best Case Setup

The best case setup reflects our assumptions of optimal conditions for the Distributed Event Detection. With a probability of $P_{EV_C} = 1\%$ the events are assumed to be critical and worth reporting to the base station. This setting causes a very reduced communication with the base station for the Distributed Event Detection. With a probability of $P_{CNF} = 0\%$ more than one SN passes the CN-Filter, hence, no second in-network communication within the Distributed Event Detection approach is caused. This reduces the in-network communication to the feature distribution to the first in-network communication only.

The results of the best case setup are depicted in Figure 5.8. An increased number of events per hour increases the energy demand for all information fusion approaches.

The Raw Data Fusion is stressed most by increasing the number of events per hour and reduced in its functionality to handle only a maximum of six events per hour as the RN runs out of processing time due to the sheer amount of packets to be handled. For future investigations where the parameter $\#_{\mathbb{E}}$ should be constant in order to gain comprehensible results, we use $\#_{\mathbb{E}} = 6$ in order to allow the Raw Data Fusion to be part of the comparison. It is clear to see that the lifetime of the Raw Data Fusion is very limited, which is caused by the huge raw data communication load.

The lifetime of the Feature Fusion equals the lifetime of the Classification Fusion and the CNs

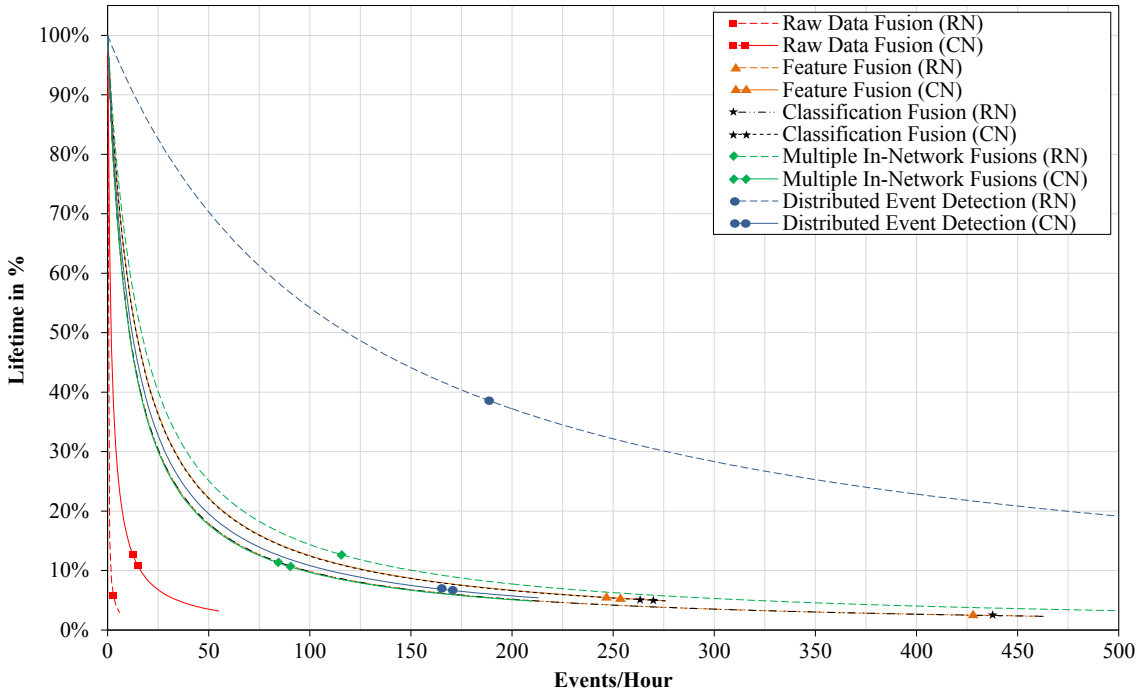


Figure 5.9.: [Standard Case] Lifetime investigation with varying parameter $\#_E$ to investigate the influence of the number of arising events per hour.

outperform the CNs of all other approaches as no in-network communication and only one single packet (with features or classification results) per event and affected SN has to be transferred. Nevertheless, as every affected SN has to transmit a packet to the base station which needs to be relayed by the RN, its lifetime is clearly outperformed by the Distributed Event Detection approach. This leads to the clear recommendation to prefer the Distributed Event Detection as the network whose SNs deplete first is under an ongoing network degeneration and will with high probability in a short amount of time change to an inoperative state.

While the Feature and Classification Fusion approaches are able to handle scenarios of up to 279 events per hour limited by the CN, the Multiple In-Network Fusions approach is limited by the CN to 213 events per hour and the Distributed Event Detection approach is limited by the CN to 278 events per hour. The reason for the extended capability to handle fewer events per hour is the additional operation time that is needed for in-network communication.

The RN of the Distributed Event Detection approach with its huge lifetime outperforms all other approaches and shows a significantly reduced overall network communication in this example. The CN lifetime of the Distributed Event Detection approach outperforms the CN lifetime of the Multiple In-Network Fusions approach because of omitted communication with the BS in case of non-critical events.

5.5.1.2 Standard Case Setup

With a probability of $P_{EVC} = 50\%$ the events are assumed to be critical and worth reporting to the BS. With a probability of $P_{CNF} = 10\%$ more than one SN passes the CN-Filter and causes a second in-network communication within the Distributed Event Detection approach. If the CN-Filter is passed by more than one SN, we assume $\#_{CNF} = 2$ SN have to initiate a second in-network communication.

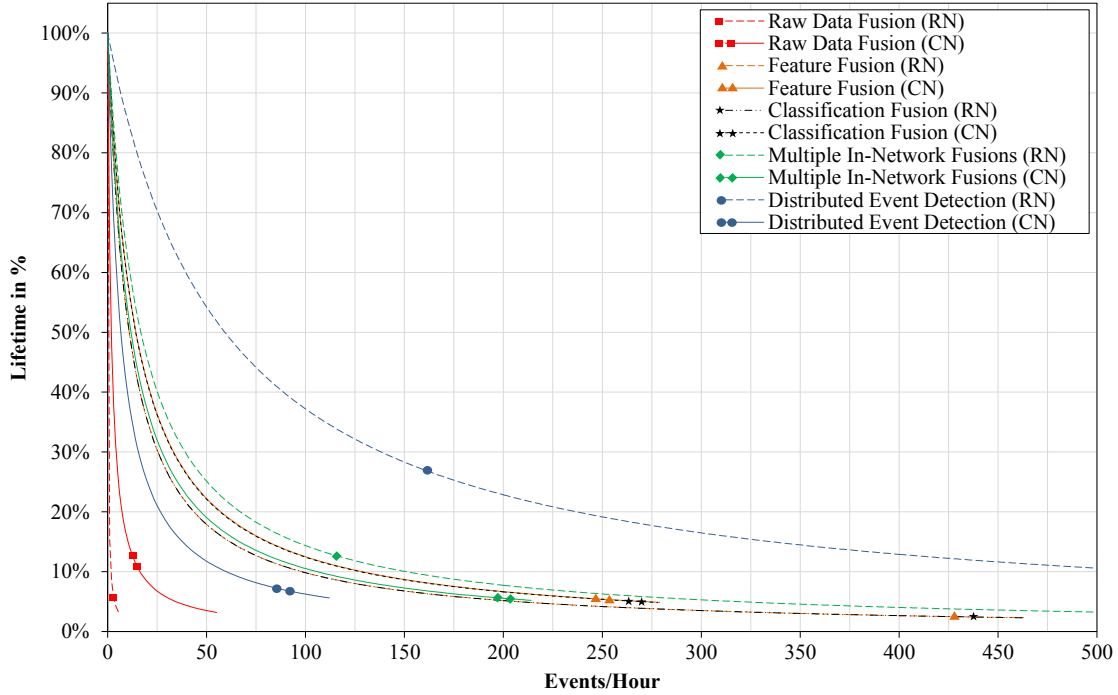


Figure 5.10.: [Worst Case] lifetime investigation with varying parameter $\#_E$ to investigate the influence of the number of arising events per hour.

The results of the standard case setup are depicted in Figure 5.9. The standard case differs only slightly from the best case which can be seen in a reduced but still clearly dominating lifetime of the RN in the Distributed Event Detection approach. This lifetime decrease is caused by an increased amount of critical events (10 %) which causes a higher communication load on the CN with the base station which has to be relayed by the RN. The processing time increases mainly for the Distributed Event Detection approach as the second in-network communication arises within the standard scenario. This influences the ability to handle scenarios limited to 213 events per hour by the according CN.

5.5.1.3 Worst Case Setup

With a probability of $P_{EVC} = 100\%$ the events are assumed to be critical and worth reporting to the BS, which causes the maximum amount of communication with the BS as every event has to be reported by the Distributed Event Detection. With a probability of $P_{CNF} = 100\%$ more than one SN passes the CN-Filter and causes a second in-network communication within the Distributed Event Detection approach. If the CN-Filter is passed by more than one SN, we assume all $\#_{CNF} = 5.4$ SNs have to initiate a second in-network communication. This causes the maximum second in-network communication as every affected SN has to communicate a second packet within the network.

The results of the worst case setup are depicted in Figure 5.10. The worst case setup stresses the Distributed Event Detection with maximized in-network communication because of the assumption that all SNs affected by an event will pass the CN-Filter which causes a second in-network communication with the CN and all affected SNs. The resulting network load reduces the lifetime of a potential CN compared to other approaches clearly and limits the manageable events per hour to 112. Although the lifetime of the RN is reduced as every event is treated as a critical event in

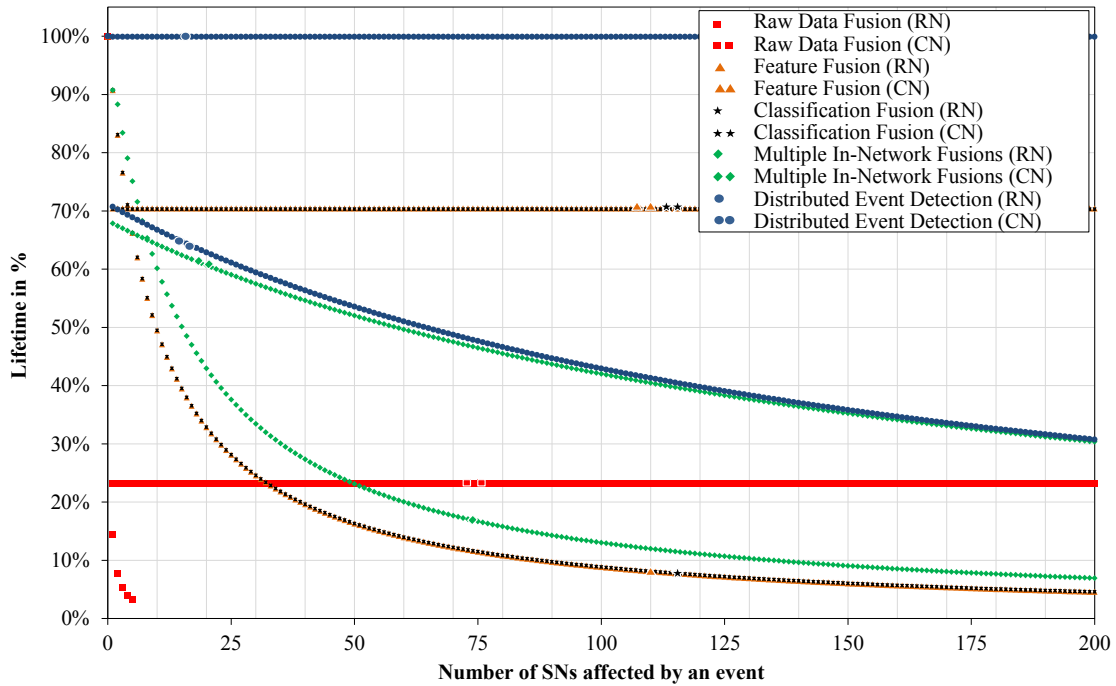


Figure 5.11.: [Best Case] Lifetime investigation with varying parameter $\#_{\text{SN}_{\text{EV}}}$ to investigate the influence of the number of SN affected by an event.

the worst case setup, it clearly outperforms the lifetime of other approaches as only one packet per event has to be transmitted. This advantage affects the whole WSN as the traffic through the network is reduced if no communication is performed in the case of uncritical events. On the flip-side we have to deal with the drawback of a slight increase in communication of the in-network communication between the event detecting nodes.

5.5.2 Lifetime Evaluation - # of affected SNs

In order to investigate the lifetime of the CN and RN within the information fusion approaches, we investigate the increased number of affected SNs by varying parameter $\#_{\text{SN}_{\text{EV}}}$ from 0 to 200 SNs on the abscissa. We pick a neutral setup of events per hour $\#_{\text{E}} = 6$. The number of affected SNs is highly application dependent but we can imagine that more affected SNs inherit a high potential to cover large scale event detection scenarios where natural disaster detection plays a major role – paralleled by an increased communication load. This trade-off will be discussed in the following sections.

5.5.2.1 Best Case Setup

With a probability of $P_{\text{EVC}} = 1\%$ the events are assumed to be critical and worth reporting to the base station. This setting causes a highly reduced communication with the BS for the Distributed Event Detection. With a probability of $P_{\text{CNF}} = 0\%$ more than one SN passes the CN-Filter, hence, no second in-network communication within the Distributed Event Detection approach is caused. This reduces the in-network communication to the feature distribution during the first in-network communication and shows the lowest energy demand for the introduced setup.

The results of the best case setup are depicted in Figure 5.11.

An increased number of affected **SNs** increases the energy demand for the **CNs** of the Distributed Event Detection and the Multiple in-Network fusions approach as well as all **RNs** except for the **RN** of the Distributed Event Detection approach. Again, the **RN** of the Distributed Event Detection approach outperforms with its huge lifetime all other approaches and shows a very much reduced overall network communication in this example. All other **RNs** are clearly suffering under the constantly increasing load of packets that have to be relayed to the **BS** caused by an increased amount of affected **SNs** at the event location. The **CN**'s lifetime of the Raw Data approach stays for all numbers of affected **SNs** at 23 % while the Feature and Classification Fusions lifetime is constantly at 70 % compared to the lifetime of the Distributed Event Detection **RN** which means these approaches are not affected by an increase of affected **SNs** in addition. The reason for these unaffected **SNs** is the non-existing in-network communication for these approaches. Especially in smaller and short term **WSNs** these approaches have a good efficiency and the simple realization makes it possible to deploy such approaches spontaneously.

The lifetime of the Distributed Event Detection **CN** outperforms the Multiple In-Network fusions approach, with an improvement of about 2.9 percentage points at one affected **SN**. This difference is reduced by an increasing number of affected **SNs** but remains for all values. This remaining and advantageous difference for the Distributed Event Detection approach can be explained by the reduced communication of uncritical events compared to a concept that reports all detected events despite their relevance.

The Raw Data Fusion is reduced in its functionality to handle only a maximum of 5.7 affected **SNs** on average as the **RN** runs out of processing time due to the sheer amount of packets to handle.

While the Feature and Classification Fusion approach are able to handle scenarios up to 417 affected **SNs** limited by the processing time of the **RN**, the Multiple In-Network Fusions approach is limited by the **RN** to 641 affected **SNs**. The Distributed Event Detection approach is limited by the processing time of the **CN** and reaches a maximum of theoretically 2006 manageable **SNs** affected by one event which would reduce the lifetime of the **CN** to 5 %.

By talking about the general applicability of the given approaches we can conclude that especially the **SNs** with the shortest lifetime dictate the usability of the network. As soon as one of the investigated **SNs**' energy source is depleted, we can say that the **WSN** starts to degenerate and changes with a higher probability (compared with other approaches) into an inoperative state in a short amount of time. In the given setup the Distributed Event Detection mostly outperforms any other approach if the number of affected **SN** is big enough. The Feature and Classification Fusion approaches are more efficient at up to 5 affected **SNs**, which means that the Distributed Event Detection has a lower lifetime for less than 6 affected **SNs** than Feature and Classification Fusion approaches, while from 5 affected **SNs** and upwards, the **RN**'s lifetime of the Feature and Classification Fusion is outperformed by the Distributed Event Detection.

Nevertheless, all other constellations with more than 5 affected **SNs** are very advantageous for the Distributed Event Detection which leads to the recommendation to apply the Distributed Event Detection under the given setup.

5.5.2.2 Standard Case Setup

With a probability of $P_{EVC} = 50\%$ the events are assumed to be critical and worth reporting to the base station. This setting still causes a reduced communication with the base station for the Distributed Event Detection. With a probability of $P_{CNF} = 10\%$ more than one **SN** will

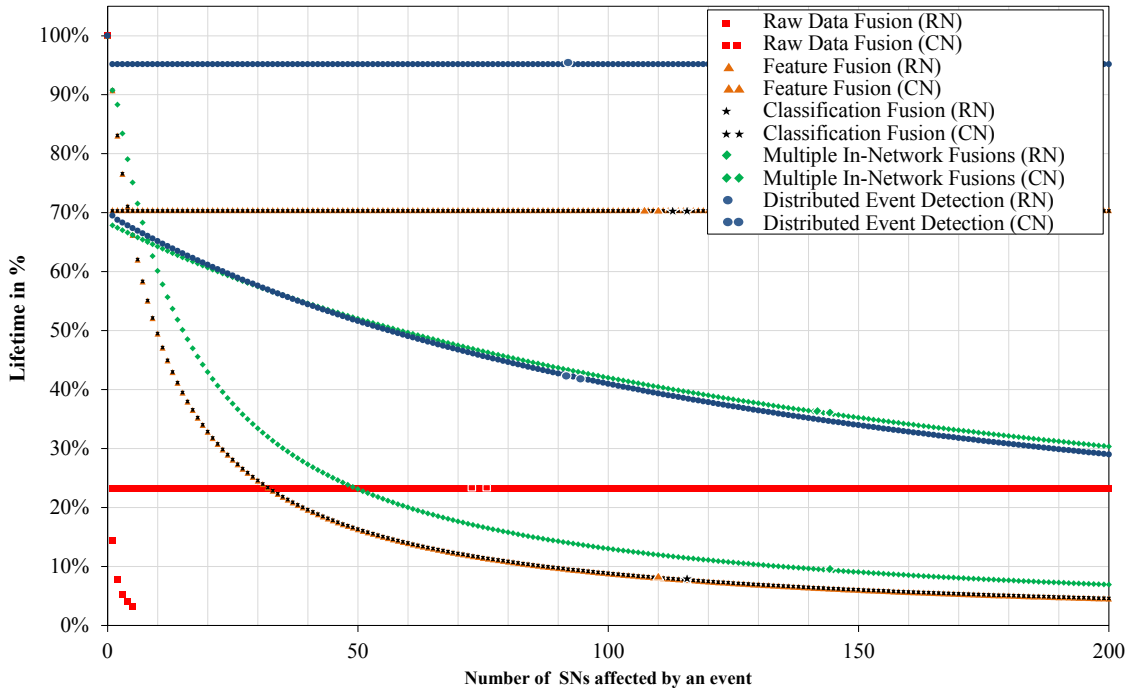


Figure 5.12.: [Standard Case] Lifetime investigation with varying parameter $\#SN_{EV}$ to investigate the influence of the number of SNs affected by an event.

pass the CN-Filter which causes a second in-network communication within the Distributed Event Detection approach. If the CN-Filter is passed by more than one SN, we assume $\#CNF = 2$ SNs have to initiate a second in-network communication, but this logically depends on the parameter $\#SN_{EV}$ because only as many SNs as SNs involved in the event recognition can pass the CN-Filter ($\#SN_{EV} \geq \#CNF$).

The results of the standard case setup are depicted in Figure 5.12. The standard case differs only slightly from the best case which can be seen in a different representation of the CN for the Distributed Event Detection.

The lifetime of the Distributed Event Detection CN outperforms the Multiple in-Network fusions approach while at one affected SN the improvement is about 1.6 percentage points. This difference turns into a negative value at 34 affected SNs. From this point on, the Multiple In-Network Fusions approach outperforms the CN of the Distributed Event Detection with a maximum of 1.3 percentage points at 219 affected SNs. This remaining and advantageous difference for the Multiple In-Network Fusions approach can be explained by the absence of the second in-network communication, which is outsourced to a communication to the base station. The resulting and much lower lifetime of the corresponding RN compared with the RN of the Distributed Event Detection reflects this relation. This increasing and huge lifetime difference between the RNs emphasizes that the above advantage of the Multiple In-Network Fusions comes with a very huge drawback for the whole network. The Lifetime of the RN of the standard case is lowered by 5 percentage points compared to the best case introduced above which is reasonable because of an increased amount of critical events that have to be relayed to the base station.

By talking about the general applicability of the given approaches we can conclude that especially the SNs with the shortest lifetime dictate the usability of the network in the long run. In the given setup the Distributed Event Detection mostly outperforms any other approach if the number of

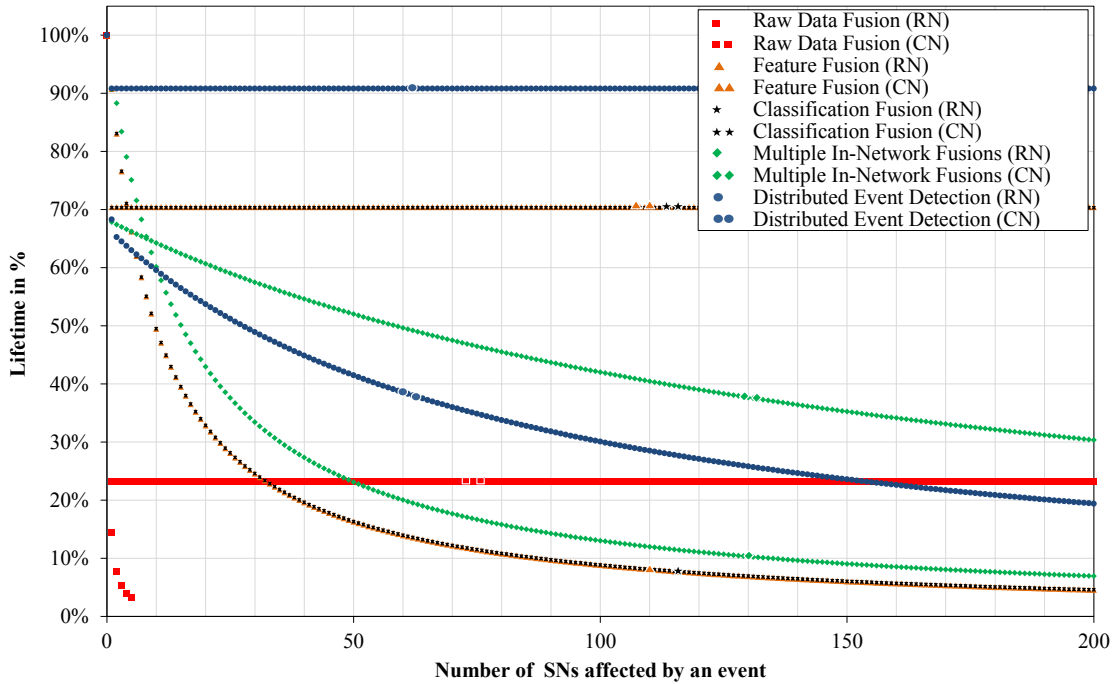


Figure 5.13.: [Worst Case] Lifetime investigation with varying parameter $\#_{\text{SN}_{\text{EV}}}$ to investigate the influence of the number of SNs affected by an event.

affected SN is big enough. Only the Feature and Classification Fusion approaches are more efficient at up to 4 affected SN, which means that the Distributed Event Detection has a lower lifetime for less than 5 affected SN than Feature and Classification Fusion approaches, while from 5 affected SNs and upwards, the RN's lifetime of the Feature and Classification Fusion is outperformed by the Distributed Event Detection.

Nevertheless, all other constellations are very advantageous for the Distributed Event Detection as for all compared approaches either the CN or the RN depletes before this happens for the Distributed Event Detection. We can recommend making use of the Distributed Event Detection in applications with more than 5 affected SNs under the given setup.

5.5.2.3 Worst Case Setup

With a probability of $P_{\text{EV}_c} = 100\%$ the events are assumed to be critical and worth reporting to the BS which causes the maximum communication with the BS as every event has to be reported by the Distributed Event Detection. With a probability of $P_{\text{CN}_F} = 100\%$ more than one SN passes the CN-Filter and causes a second in-network communication within the Distributed Event Detection approach for every event. If the CN-Filter is passed by more than one SN, we assume $\#_{\text{CN}_F} = 2$ SNs have to initiate a second in-network communication, but this logically depends on the parameter $\#_{\text{SN}_{\text{EV}}}$ because only as many SNs as SNs involved in the event recognition can pass the CN-Filter ($\#_{\text{SN}_{\text{EV}}} \geq \#_{\text{CN}_F}$).

The results of the worst case setup are depicted in Figure 5.13. The worst case setup stresses the Distributed Event Detection with maximized in-network communication, based on the setup conditions that all SNs affected by an event will pass the CN-Filter which causes a second in-network communication with the CN and all affected SNs. The resulting in-network load obviously reduces the lifetime of the CN of the Distributed Event Detection. The CN of the Multiple In-

Network Fusions outperforms the Distributed Event Detection with a maximum of 12 percentage points at 106 affected **SNs**. Even the Raw Data Fusion would outperform the Distributed Event Detection at 154 affected **SNs** with increasing lifetime growth, but the Raw Data Fusion is limited to a maximum of 5.4 affected **SNs** on average as the **RN** runs out of processing time because of the sheer amount of packets to be handled which means the advantage of the Raw Data Fusion is obsolete. The lifetime of the **RN** of the worst case is lowered by 9 percentage points compared to the best case setup introduced above, which is reasonable because of an increased amount of critical events that have to be relayed to the base station.

Although the Distributed Event Detection approach suffers heavily from the worst case scenario, it still makes sense to use this approach as all other approaches are much earlier depleted in lifetime by either the **RN** or the **CN** compared to the Distributed Event Detection. Let us investigate a short example at 100 affected **SNs**. Even if we used the Multiple In-Network Fusions, the **RN**'s lifetime is reduced to 13% compared to the outperforming 30% of the **CN** lifetime of the Distributed Event Detection. As soon as one of the investigated **SNs**' energy source is depleted, we can state that the **WSN** starts to degenerate and turns with high probability into an inoperative network or at least into a network with a reduced radius of operation in a short amount of time.

By deriving the general applicability of the given approaches we can conclude that especially the **SNs** with the shortest lifetime dictate the usability of the network in the long run. In the given setup, the Distributed Event Detection is superior to other approaches if the number of affected **SN** is big enough. The Feature and Classification Fusion approaches are more efficient at up to 5 affected **SNs**, which means the Distributed Event Detection has a lower lifetime with less than 6 affected **SN** than the Feature and Classification Fusion approach. The Multiple In-Network Fusions approach is more efficient at up to 10 affected **SNs**, which means that the **CN** of the Distributed Event Detection has a lower lifetime for approaches with less than 11 affected **SN** than the Multiple In-Network Fusions approach. As we are investigating the worst case setup it is not surprising to see the limits of the Distributed Event Detection while other approaches cover the area of applications from this point on. Nevertheless, there is still a huge application area left especially for applications with more than 11 affected **SNs** where the Distributed Event Detection starts to outperform the energy demands of all other information fusion approaches.

5.5.3 Lifetime Evaluation - All affected Sensor Nodes pass the Central Node Filter

In order to investigate the lifetime of the **CN** and **RN** within the information fusion approaches, we investigate the increased the number of affected **SNs** by varying parameter $\#_{\text{SN}_{\text{EV}}}$ as well as $\#_{\text{CNF}}$ from 0 to 200 on the abscissa in the same way. This means both parameters are equally varied and are identically represented by the abscissa. Technically this means that if a certain number of **SNs** is affected by an event, exactly this number passes the **CN**-Filter. This is a very stressing but unlikely constellation for the Distributed Event Detection. Nevertheless, we want to investigate this setup in order to explore the limits of our approach in more detail. The number of affected **SNs** is highly application dependent, but we can imagine that more affected **SNs** inherit a high potential to cover large scale event detection scenarios where natural disaster detection plays a major role – paralleled by an increased communication load. This trade-off will be discussed in the following sections. We pick a neutral setup of events per hour $\#_{\text{E}} = 6$.

All curves besides the **CN** representation of Distributed Event Detection behave exactly as described in the previous Section 5.5.2. As a difference to the previous $\#$ of affected **SNs** setup we

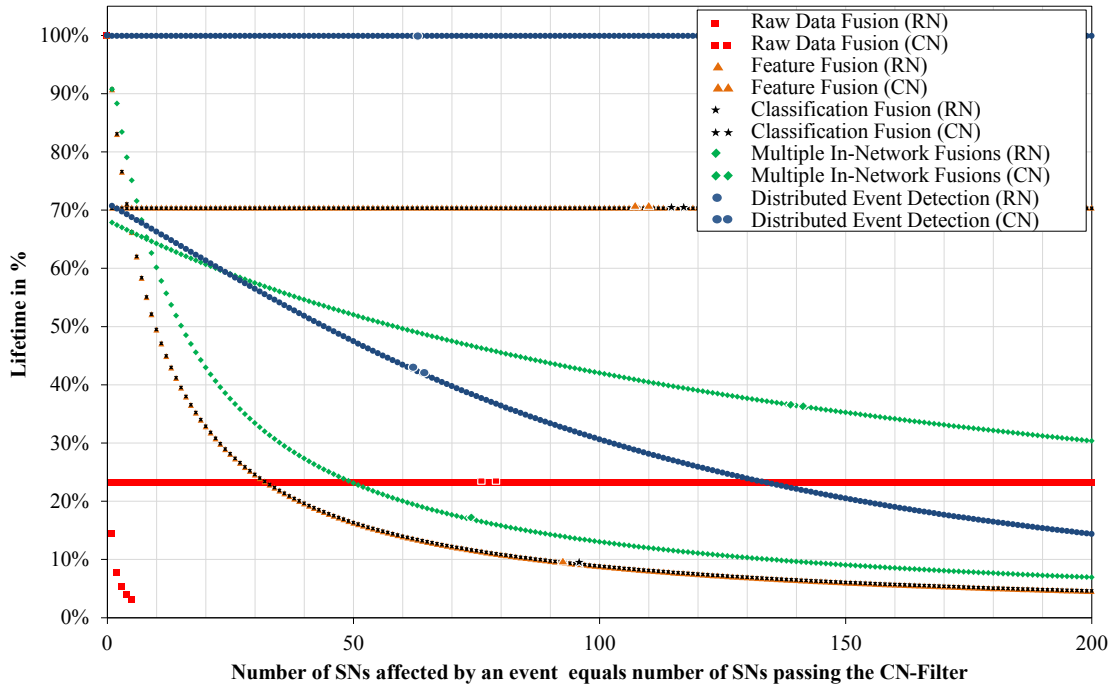


Figure 5.14.: [Best Case] Lifetime investigation with equally varying parameters $\#_{\text{SN}_{\text{EV}}}$ and $\#_{\text{CNF}}$ to investigate the influence of the number of affected SNs and the equal number of SNs passing the CN-Filter by an event

now only vary the parameter $\#_{\text{CNF}}$. This means that we will focus on the evaluation of the CN representation of the Distributed Event Detection approach in the following setup evaluation.

5.5.3.1 Best Case Setup

With a probability of $P_{\text{EV}_C} = 1\%$ the events are assumed to be critical and worth reporting to the BS. This setting causes a very reduced communication with the BS for the Distributed Event Detection approach. With a probability of $P_{\text{CNF}} = 1\%$ all affected SNs pass the CN-Filter, hence, in one of hundred events a second in-network communication within the Distributed Event Detection approach is caused. This strongly reduces the in-network communication to the feature distribution during the first in-network communication but still allows a second in-network communication.

The results of the best case setup are depicted in Figure 5.14. The behavior of the depicted course of the CN-lifetime function of the Distributed Event detection indicates clearly an increased energy demand with increased number of affected SNs while having a second in-network communication for all these SNs. But still, and because of the optimal conditions of our setup ($P_{\text{EV}_C} = 1\%$, $P_{\text{CNF}} = 1\%$), the Distributed Event Detection CN is able to outperform the Multiple In-Network Fusions approach with a maximum of 2.9 percentage points of additional lifetime. This advantage is reduced to 0 until the curve reaches 25 affected SNs as all of these SNs will pass the CN-Filter. It is worth mentioning that 25 SNs are already a rather large number of affected SNs and can cover a huge number of applications. Applications where more than 25 SNs are involved in a detection process are rare but possible and, as mentioned, more likely in the area of natural disaster prevention and detection. In addition, we want to mention that to the best of our knowledge, there is no existing real world deployment in the area WSNs with event detection scenarios investigating more than 20 concurrently affected SNs at the moment. Thus, our curves describe more the future

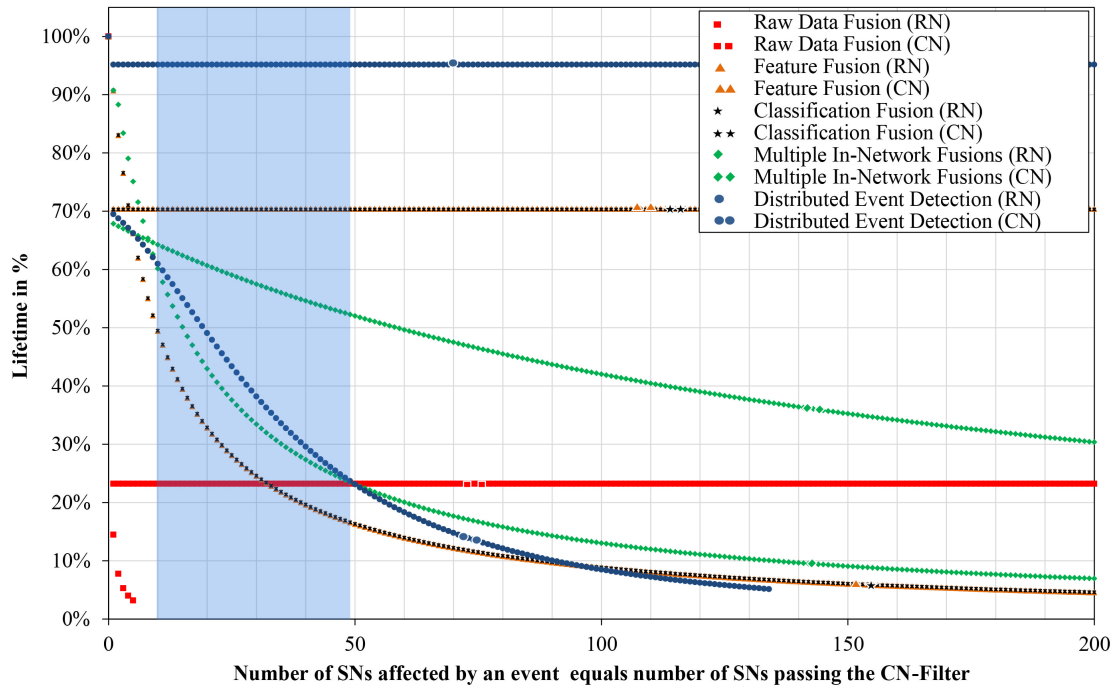


Figure 5.15.: [Standard Case] Lifetime investigation with equally varying parameters $\#_{SN_{EV}}$ and $\#_{CNF}$ to investigate the influence of the number of affected SNs and the equal number of SNs passing the CN-Filter by an event.

potential of the proposed approach.

By investigating the more general view in this setup it is important to take into account the first SN that depletes its energy storage. For the Distributed Event Detection approach the CN always depletes first compared to the according RN. If we compare the lifetime of this CN with the lifetime of the first depleted SN of the other approaches, the following conclusion can be made: The RN of the Feature and Classification Fusion approach is outperformed by the Distributed Event Detection at 5 and more affected SNs. The Multiple In-Network Fusions approach is always outperformed by the Distributed Event Detection as either the CN or the RN of the Multiple In-Network Fusions approach has a lower lifetime than the CN of the Distributed Event Detection approach has. As a general statement we can say that the Distributed Event Detection is less energy demanding with 5 or more affected SNs under the special circumstance that all affected SNs are passing the CN-Filter in addition.

5.5.3.2 Standard Case Setup

With a probability of $P_{EV_C} = 50\%$ the events are assumed to be critical and worth reporting to the BS. This setting still causes a reduced communication with the BS for the Distributed Event Detection. With a probability of $P_{CNF} = 10\%$ all affected SNs will pass the CN-Filter, which causes a second in-network communication within the Distributed Event Detection approach.

The results of the standard case setup are depicted in Figure 5.15. The behavior of the depicted course of the CN-lifetime function of the Distributed Event Detection indicates a heavily increased energy demand with increased number of affected SNs while having a second in-network communication for all these SNs.

The CN of the Distributed Event Detection can outperform the CN of Multiple In-Network Fu-

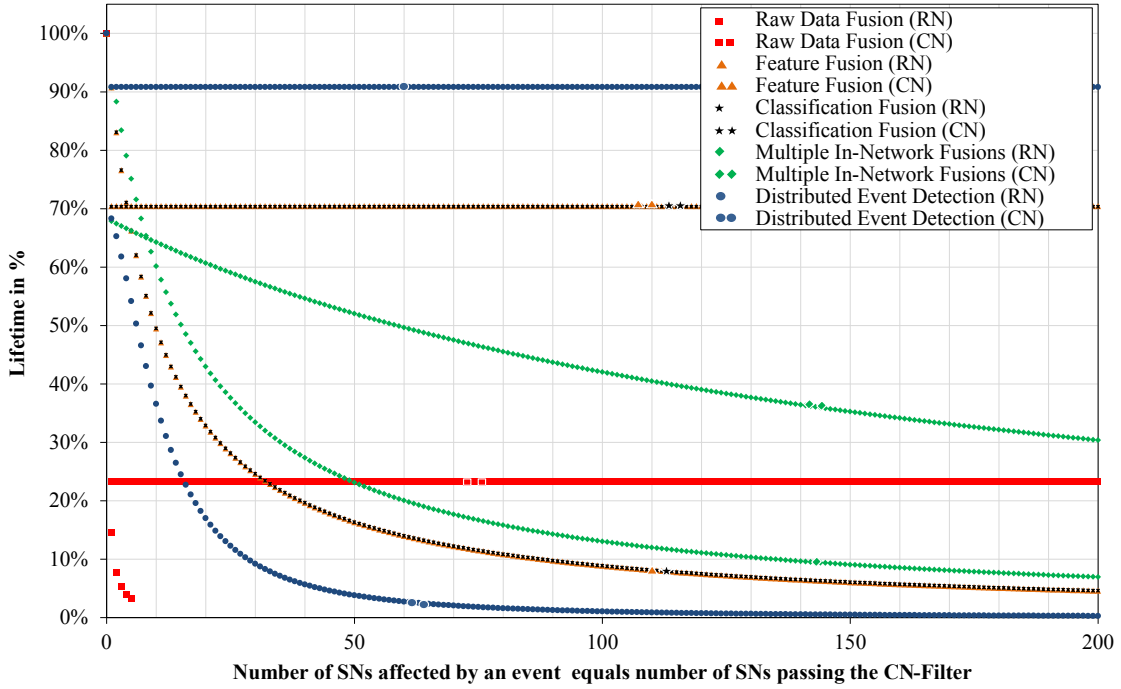


Figure 5.16.: [Worst Case] Lifetime investigation with equally varying parameters $\#SN_{EV}$ and $\#CNF$ to investigate the influence of the equal number of affected SNs and the number of SNs passing the CN-Filter by an event.

sions for up to 4 affected sensor nodes. From 6 to 9 affected SNs, the Distributed Event Detection approach is outperformed by the Multiple In-Network Fusions approach as the CN of the Distributed Event Detection approach has a lower lifetime than both the CN and RN of the Multiple In-Network Fusions approach. The Distributed Event Detection dominates the compared field of approaches from 10 affected SNs to 49 affected SNs with a maximum lifetime advantage of 6 percentage points towards the Multiple In-Network Fusions and with a maximum of 16.1 percentage points towards the Feature and Classification Fusion approaches, see the blue transparent area in Figure 5.15.

In addition, the Feature and Classification Fusion approaches outperform the Distributed Event Detection up to 6 affected SNs. From more than 49 affected SNs the Feature and Classification Fusion approaches outperform the Distributed Event Detection again. The increased network load reduces the maximum lifetime of the Distributed Event Detection if more than 49 SNs get affected. The maximum lifetime of a potential CN compared to other approaches is clearly reduced and limits the manageable events per hour to 134 because of 50% critical events and a 10% probability to have SNs passing the CN.

With the given setup, we are reaching the limits of the applicability of the Distributed Event Detection. Nevertheless, we have a minority area of applications left (10 affected SNs to 49 affected SNs) that are more efficient to be used with the Distributed Event Detection, see the blue transparent area in Figure 5.15.

5.5.3.3 Worst Case Setup

With a probability of $P_{EV_C} = 100\%$ the events are assumed to be critical and worth reporting to the base station, which causes the maximum communication with the base station as every event has to

be reported by the Distributed Event Detection. With a probability of $P_{\text{CNF}} = 100\%$ all affected **SN** pass the CN-Filter and cause a second in-network communication within the Distributed Event detection approach for every event. As mentioned, this is a very unlikely situation but shows our worst setup which is not recommended for our approach.

The results of the worst case setup are depicted in Figure 5.16. With this setup the applicability of the Distributed Event Detection is marked. It is obvious to see that under the given setup the Distributed Event Detection is outperformed by all approaches except the Raw Data Fusion. Although the **RN** has still a very good energy level, the **CN** will be depleted in the given setup if all events occur at the same **CN**. It is a clear recommendation not to apply the Distributed Event Detection if every event would cause all affected **SNs** to initiate two rounds of in-network communication. In particular, the **CN** energy demand increases due to the increased in-network communication which can not be compensated by the Distributed Event Detection approach.

5.5.4 Lifetime Evaluation - Probability of Critical Events

In order to investigate the lifetime of the **CN** and **RN** within the information fusion approaches, we investigate different probabilities of an arising critical event as introduced in 5.2.6. For this purpose the parameter P_{EVC} is varied from 100 % down to 4 % on the abscissa. We want to investigate this setup in order to explore the benefits and limits of our approach. It is expected that in general, more critical events cause more traffic for the Distributed Event Detection. We pick a neutral setup of events per hour $\#_{\text{E}} = 6$. In addition, we pick a neutral setup of $\#_{\text{SNEV}} = 5.4$ **SNs** that are affected on average by events, because this was the average number of affected **SNs** during our fence surveillance system experiments.

5.5.4.1 Best Case Setup

With a probability of $P_{\text{CNF}} = 0\%$ more than one **SN** passes the CN-Filter, hence, no second in-network communication within the Distributed Event Detection approach is caused. This reduces the in-network communication to the feature distribution during the first in-network communication.

The results of the best case setup are depicted in Figure 5.17. As depicted, a decreased amount of events that are flagged as critical increases the lifetime of the **CN** as well as the **RN**. In the case of 100% probability of having critical events, the lifetime of the **CN** Distributed Event Detection and the Multiple In-Network Fusions approaches are equal. The Distributed Event Detection starts outperforming the Multiple In-Network Fusions from that point down to 12 % where it reaches a lifetime increase of 2.5 %. We can see that the increased in-network communication directly affects the lifetime if the probability of arising critical events increases towards $P_{\text{EVC}} = 100\%$ and thus the applicability of the Distributed Event Detection. It is therefore very important to reduce these cases and to find a reliable **CN-Filter** function.

Although the **CN** lifetime of the Feature and Classification Fusion approaches outperforms the Distributed Event Detection in the direct comparison, the lower lifetime of the Feature and Classification Fusion **RN** limits its applicability in contrast to the Distributed Event Detection. Both **SNs** of the Raw Data Fusion are below 30 % of the lifetime that is reached by the **RN** of the Distributed Event Detection at 100 % probability for critical events.

The expectation that more critical events cause more traffic for the Distributed Event Detection can be confirmed. This leads to the recommendation for the best case setup that applications with

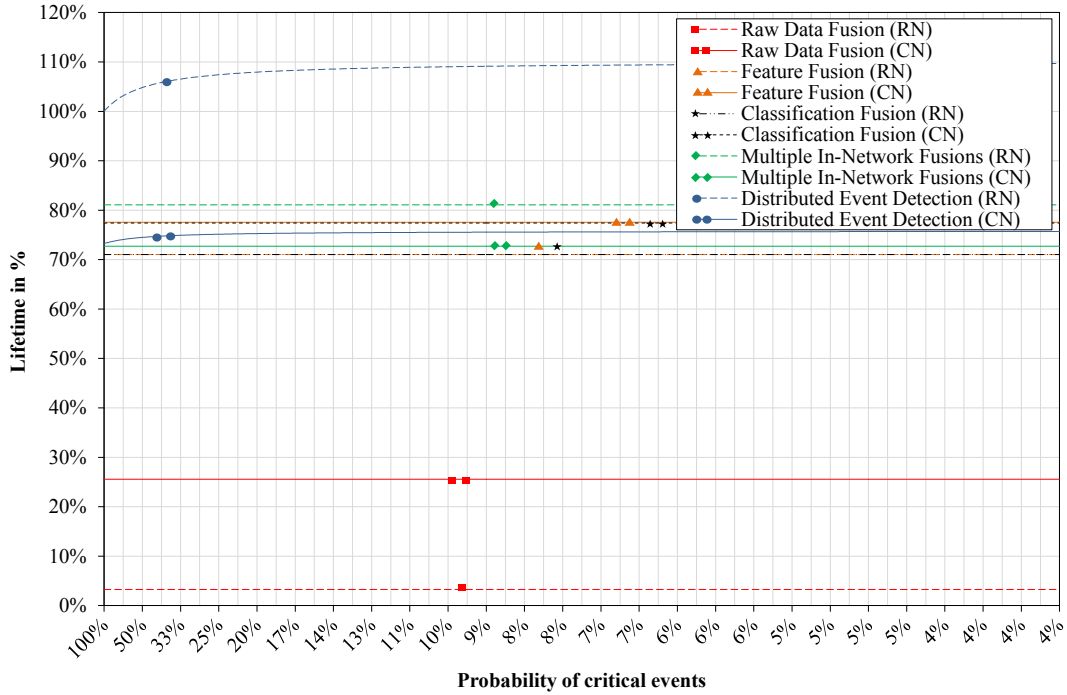


Figure 5.17.: [Best Case] Lifetime investigation with varying parameter P_{EVC} to investigate the influence of different probabilities of arising critical events.

a critical as well as uncritical events to be detected can very much benefit from the Distributed Event Detection as this approach saves energy by not sending non-relevant events to the base station.

5.5.4.2 Standard Case Setup

With a probability of $P_{CNF} = 10\%$ more than one **SN** will pass the **CN-Filter** which causes a second in-network communication within the Distributed Event Detection approach. If the **CN-Filter** is passed by more than one **SN**, we assume $\#_{CNF} = 2$ **SNs** have to initiate a second in-network communication.

The results of the standard case setup are depicted in Figure 5.18. The Distributed Event Detection starts to outperform the Multiple In-Network Fusions approach already with a probability for critical events of 100%. The Distributed Event Detection starts to outperform the Multiple In-Network Fusions from that point down to 2% where it reaches a lifetime increase of 2%. We can see that the increased in-network communication directly affects the lifetime if the probability of arising critical events increases towards $P_{EVC} = 100\%$ and thus the applicability of the Distributed Event Detection. It is therefore very important to reduce these cases and to find a reliable **CN-Filter** function.

Although the **CN** lifetime of the Feature and Classification Fusion approaches outperforms the Distributed Event Detection in the direct comparison, the reduced lifetime of the Feature and Classification Fusion's **RN** limits its applicability compared with the Distributed Event Detection in general. Both **SNs** of the Raw Data Fusion are below 30% of the lifetime that is reached by the **RN** of the Distributed Event Detection at 100% probability for critical events.

The expectation that more critical events cause more traffic for the Distributed Event Detection

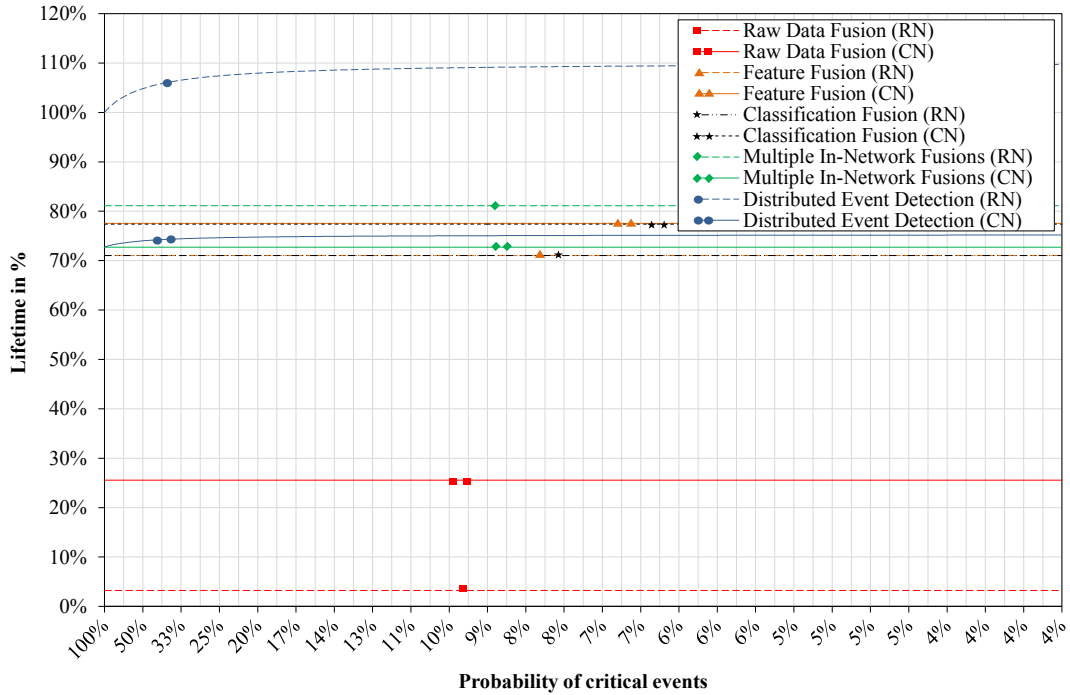


Figure 5.18.: [Standard Case] Lifetime investigation with varying parameter P_{EVC} to investigate the influence of different probabilities of arising critical events.

can be confirmed again. This leads to the same recommendation for the standard case setup that applications with a critical as well as uncritical events to be detected can very much benefit from the Distributed Event Detection as this approach saves energy by not sending non-relevant events to the BS.

5.5.4.3 Worst Case Setup

With a probability of $P_{CNF} = 100\%$ more than one SN passes the CN-Filter and causes a second in-network communication within the Distributed Event Detection approach for every event. If the CN-Filter is passed by more than one SN, we assume for this worst case setup that all SNs have to initiate a second in-network communication which is based on our neutral setup for this investigation are $\#_{CNF} = 5.4$ SN. This means that for every arising event always all 5.4 affected SNs will initiate a second in-network communication. As mentioned, this is a very unlikely situation, but it represents the worst setup which is not recommended for the Distributed Event Detection.

The results of the worst case setup are depicted in Figure 5.19. Compared with the best case and standard case setup it is clear to see that the maximum increase of in-network communication consumes all advantages of the in-network communication in terms of lifetime. The overall lifetime of the CN of the Distributed Event Detection settles at 59% of the lifetime that is reached by the RN of the Distributed Event Detection at 100% probability for critical events. This means that all other approaches except the Raw Data Fusion approach outperform the Distributed Event Detection for this worst case setup in general.

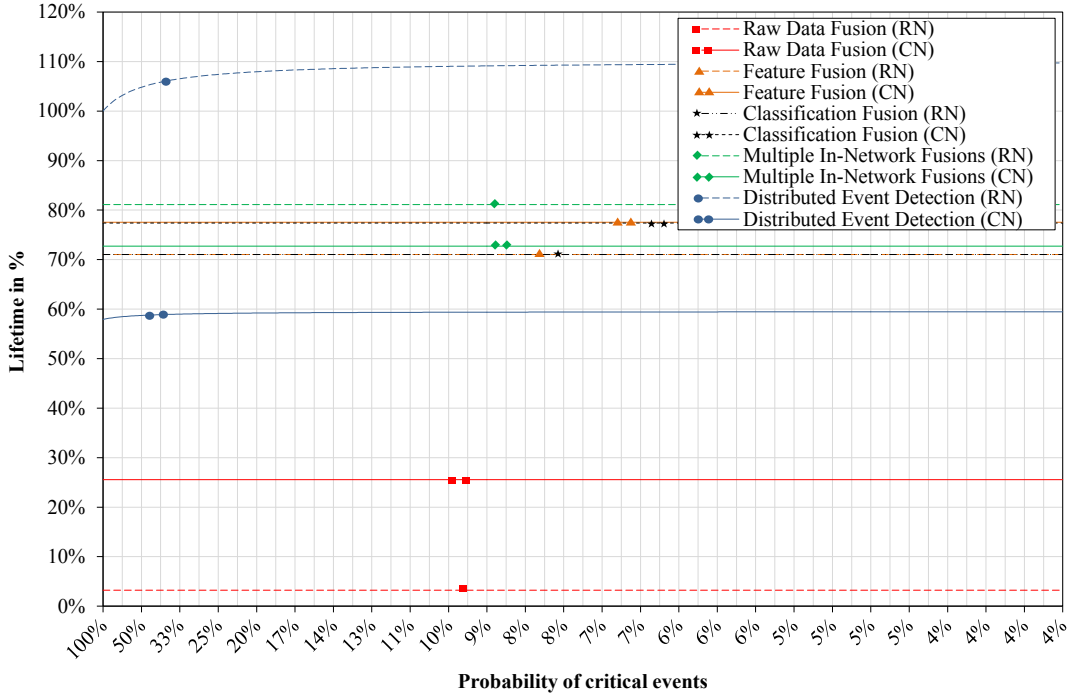


Figure 5.19.: [Worst Case] Lifetime investigation with varying parameter P_{EV_C} to investigate the influence of different probabilities of arising critical events.

5.5.5 Lifetime Evaluation - Probability of passing the CN-Filter

In order to investigate the lifetime of the **CN** and **RN** within the information fusion approaches, we investigate different probabilities of **SNs** of passing the **CN-Filter** as introduced in 5.2.6.

For this purpose the parameter P_{CNF} is varied from 100% down to 4% on the abscissa. We want to investigate this setup in order to explore the benefits and limits of our approach. It is expected that in general, the energy demand increases if the probability of passing the **CN-Filter** is increased. A lower probability should reduce the second in-network communication and should support the advantages of the Distributed Event Detection compared to other information fusion approaches.

We pick a neutral setting for the events per hour $\#_E = 6$ and for on average affected **SNs** by events $\#_{SN_{EV}} = 5.4$ as we could derive this parameter during our experiments with an example fence surveillance system deployment that uses our Distributed Event Detection system.

5.5.5.1 Best Case Setup

With a probability of $P_{EV_C} = 1\%$ the events are assumed to be critical and worth reporting to the base station. This setting causes a very reduced communication with the **BS** for the Distributed Event Detection. If more than one **SN** passes the **CN-Filter**, two **SNs** pass the $\#_{CNF} = 2$ in the best case setup.

The results of the best case setup are depicted in Figure 5.20. The results show that a probability of passing the **CN-Filter** of above 77% causes the traffic for the second in-network detection for two **SNs** to reduce the lifetime of the **SN** of the Distributed Event Detection below approaches with no second in-network communication as the Multiple-In-Network Fusions approach. Instead, for probabilities of passing the **CN-Filter** lower than or equal to 77% the Distributed Event Detection

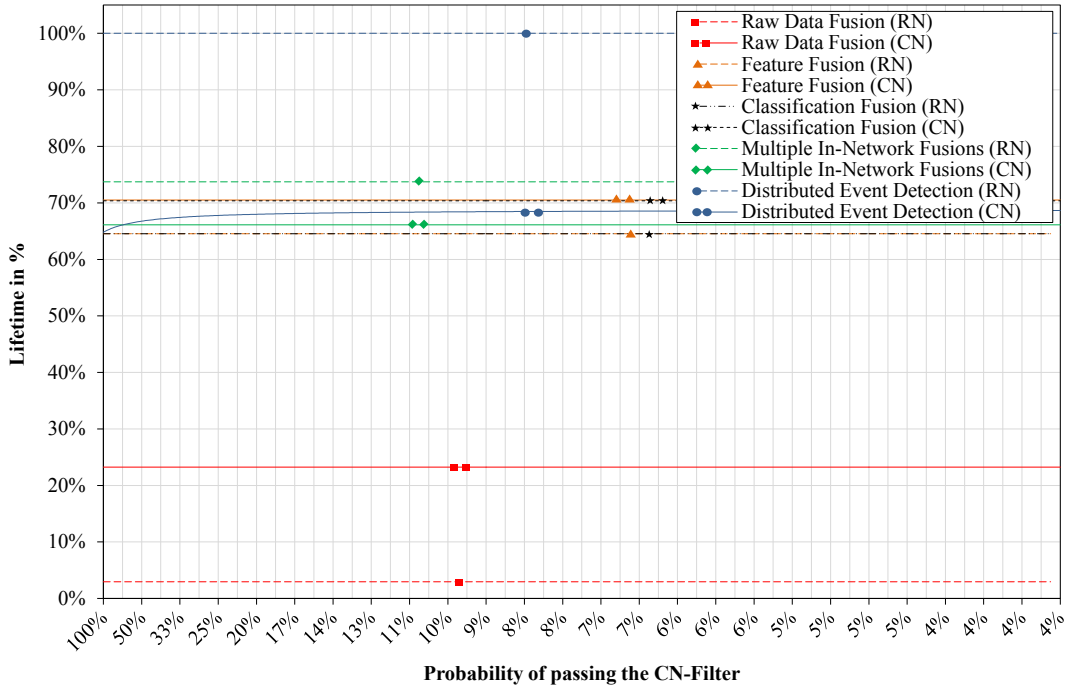


Figure 5.20.: [Best Case] Lifetime investigation with varying parameter P_{CNF} to investigate the influence of different probabilities of passing the CN-Filter.

outperforms the lifetime of compared approaches, which leads to a better general usability.

Although the CN lifetime of the Feature and Classification Fusion approaches outperforms the Distributed Event Detection in direct comparison, the lower RN lifetime of the Feature and Classification Fusion limits its general applicability compared to the Distributed Event Detection.

The expectation that the energy demand is increased if the probability of passing the CN-Filter is increased in general can be confirmed. The lower probability limits the second in-network communication and supports the advantages of the Distributed Event Detection compared to other information fusion approaches.

5.5.5.2 Standard Case Setup

The results of the standard case setup are depicted in Figure 5.21. With a probability of $P_{EVC} = 50\%$ the events are assumed to be critical and worth reporting to the base station. This setting still causes a reduced communication with the base station for the Distributed Event Detection. If more than one SN passes the CN-Filter, two SNs pass the $\#_{CNF} = 2$ for the standard case setup. The results in comparison to the best case setup show two effects. First, the increased in-network communication causes a lower CN lifetime for the Distributed Event Detection. This is reflected by the reduced lifetime of that SN compared to other approaches with a slight advantage towards the other approaches from a probability of passing the CN-Filter of about 53% and lower. Second, the increased probability for critical events $P_{EVC} = 50\%$ causes the RN of the Distributed Event Detection to have more packets to be relayed. This leads to a relative increase of the RN lifetime of all other approaches of about 0.4 percentage points for the Raw Data Fusion approach, about 5 percentage points for the Feature and Classification Fusion approaches and about 5.2 percentage points for the Multiple In-Network Fusions approach.

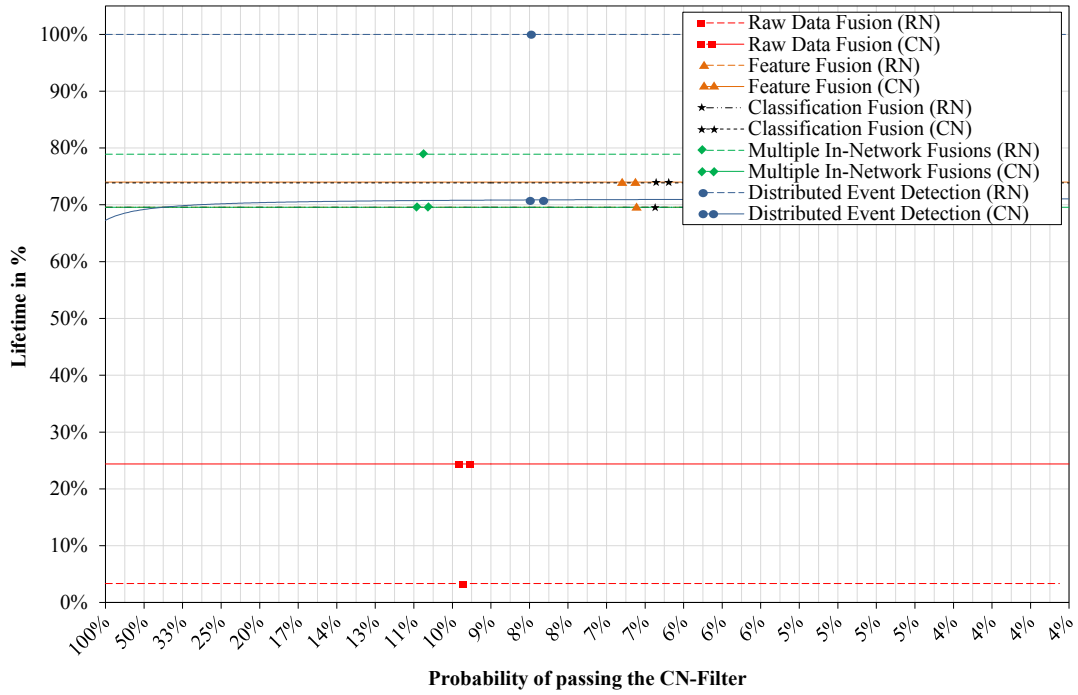


Figure 5.21.: [Standard Case] Lifetime investigation with varying parameter P_{CNF} to investigate the influence of different probabilities of passing the CN-Filter.

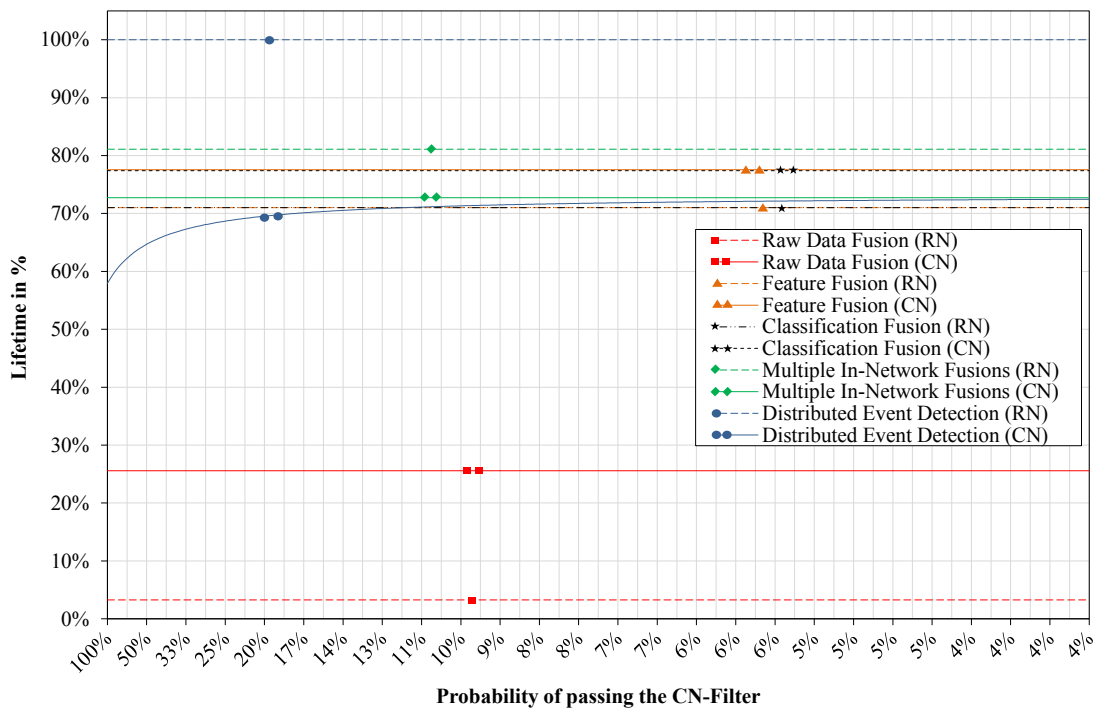


Figure 5.22.: [Worst Case] Lifetime investigation with varying parameter P_{CNF} to investigate the influence of different probabilities of passing the CN-Filter.

5.5.5.3 Worst Case Setup

The results of the worst case setup are depicted in Figure 5.22. With a probability of $P_{EVC} = 100\%$ the events are assumed to be critical and worth reporting to the BS. This setting causes the highest possible communication with the BS for the Distributed Event Detection as each event is reported. In addition, all affected SNs pass the CN-Filter, which are in this case $\#_{CNF} = 5.4$ SNs in the worst case setup.

This worst case setup finally shows the limits for the Distributed Event Detection as the RN of the Feature Classification Fusion are slightly outperformed in case of a lower probability of passing the CN than 12%. The Multiple In-Network Fusions approach can be outperformed from 4% and below. Of course, the Raw Data Fusion approach is always outperformed. Hence, we can conclude that CN-Filter plays a key role as this filter regulates additional occurring communications and thus regulates the lifetime. Applications that tend to produce very similar sensor data over multiple SNs are not recommended to be conducted with the Distributed Event Detection, which is thus feasible if we consider that in general the proposed concept benefits most from diverse sensor readings from multiple and different perspectives.

If an application produces very similar sensor data over multiple SNs it may cause multiple SNs to pass the CN-Filter. This is an indicator for redundant deployment of SN that should be reconsidered with the goal to optimize the deployment in favor of the efficiency and impact of each SN's data acquisition.

5.5.6 Lifetime Conclusions and Applicability

As discussed in Section 5.3.2, a multi-hop WSN will suffer with increasing hop length under heavy traffic beyond network sizes of two hops for Feature and Classification Fusion as well as for the Multiple In-Network Fusions.

With the given analysis, a large variety of applications will be able to find an appropriate event detection system with our proposed approach. In general it can be stated that the Distributed Event Detection is very energy efficient thanks to the conceptionally reduced communication and outperforms the introduced approaches in a huge variety of different setups. Nevertheless, if the in-network communication increases, e.g. because too many SNs are affected by the event, and if too many of these SNs cause a second in-network communication, the Distributed Event Detection can be outperformed by other approaches as the advantage of the compelling in-network communication and evaluation turns into a too high energy demand. Typically, this is the case if SN are deployed inefficiently and can not deliver a diverse data set as their perspectives on the event are too similar.

Even though, the RN of our proposed system surpasses all other approaches in term of lifetime. The outstanding lifetime of the RN affects the whole network positively as the lifetime of the RN reflects the low energy demands for all other SNs involved in a communication route to transmit data to that RN. If one SN depletes first, the remainder of the system starts to degenerate more and more from that point on, hence a depleted SN is a very good indicator to judge a system as outperformed. In the particular case of the Distributed Event Detection RN's lifetime, this is an indicator of a significantly reduced network load.

In combination with the investigation of Section 5.3.2, it is possible to say that the Distributed Event Detection outperforms all other approaches for networks with a hop-count of three or more. The Feature Fusion approach outperforms the other approaches for networks with a small hop-count (1-hop and 2-hops).

Section 5.2.7 helps us decide that for the best classification result, the Raw Data Fusion is recommended, although this approach only makes sense for very short term deployments compared to the other information fusion approaches as shown in the evaluation. For research, in depth investigation, or our previous training process in the Evaluation Framework (see Figure 4.2), this approach has its specific area of application. The Distributed Event Detection catches up with the Feature Fusion approach as both apply with FEI the same input detail level for the fusion process. In contrast to the Feature Fusion, the Distributed Event Detection is able to perform the classification within the network which enables to perform further automated in-network process decisions. It is shown in the previous Sections that the Distributed Event Detection outperforms the Feature Fusion approach in a huge variety of application setups as well as the Multiple In-Network Fusions.

In some rare worst case setups the Multiple In-Network Fusions and the Feature and classification Fusion approaches outperform together with the Distributed Event Detection (see Figure 5.10, 5.16 and 5.19). The classification fusion has the same energy demand as the Feature Fusion but has lower classification capabilities because of a lower input detail level for the fusion process (see Table 5.2); hence, the Feature Fusion approach should be preferred instead of the Classification Fusion approach if local sensor classification results are not necessary within the network. The Multiple In-Network Fusions has a lower input detail level for the fusion process as decisions have to be fused based on a majority vote if no additional location based sensor data (like camera or GPS based data) are available in order to properly assign the classification results to the real world at the base station. Thus, for the mentioned rare worst cases the Feature Fusion should be preferred instead of the Classification fusion and Multiple In-Network Fusions.

The Distributed Event Detection is most efficient for WSNs that need to create routes with more than two hops on average. The Distributed Event Detection delivers a very high lifetime for the CN and an outstanding high lifetime for the RN if compared to the introduced information fusion approaches. This positive conclusion means that especially the RN's lifetime will influence the whole network positively. In addition, the Distributed Event Detection delivers all necessary information as event position and event category type within the network system which leads to high energy savings for all SNs for most of the investigated setups.

A drawback of the Distributed Event Detection is mainly the possible second in-network communication that could arise if multiple SN pass the CN-Filter. As mentioned, the second in-network communication is an indicator for redundant sensor data over multiple SNs which should motivate a revision of the deployment setup for the affected applications. Instead, the goal of the Distributed Event Detection is in its core to benefit most from diverse sensor readings from multiple and different perspectives, and this will influence the classification results as well as the energy demands positively if applied correctly.

5.5.6.1 Reference Parameter Setups

In order to benefit most from a WSN with the Distributed Event Detection, we provide as a rule of thumb an upper and lower bound reference setup recommendations based on the previous evaluations that pushes the parameter boundaries to the limits while always outperforming all other approaches. In contrast to the previous figures, the subsequent evaluation shows the CN of the Distributed Event Detection curve in two different setup variation within one figure. It is important to mention that the RN of the Raw Data Fusion can handle the proposed setup only

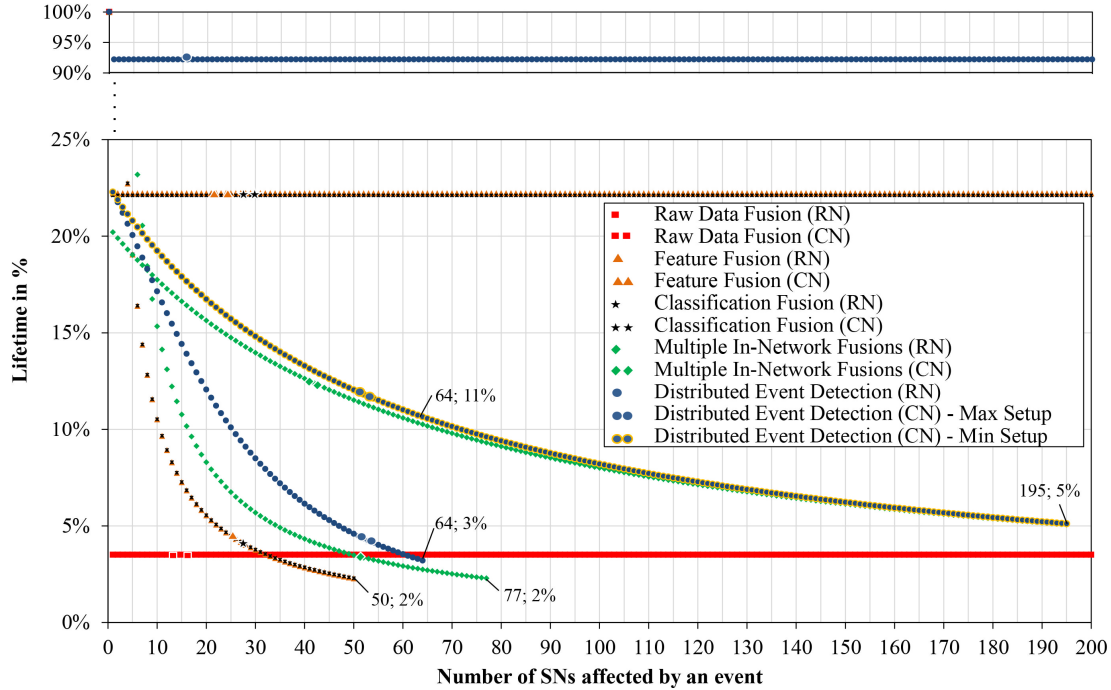


Figure 5.23.: Two reference setups with an assumed minimized and a maximized second in-network communication. The lifetime calculation is performed with increasing parameter $\#_{SN_{EV}}$ to investigate the number of affected SNs .

with a maximum of 5.4 affected SNs on average and a setup of a maximum of 6 events per hour and is not depicted for that reason.

Max-Setup: The Max-Setup maximizes the second in-network communication of the Distributed Event Detection that is in this setup often necessary for the CN-Filter process. We reflect this situation by maximizing the probability of passing the CN-Filter (P_{CNF}) 7% and the number of SNs passing the CN-Filter ($\#_{CNF}$) to 64. The number of SNs passing the CN-Filter increases equally with the number of SNs affected by an event. The parameter maximization of these two parameters and all others is always outperforming at least one of the SNs of all competitive approaches.

- a) Number of arising events per hour ($\#_E \leq 50$)
- b) Number of SNs affected by an event ($\#_{SN_{EV}} \leq 64$)
- c) Probability of a critical event ($P_{EVC} \leq 10\%$)
- d) Probability of passing the CN-Filter ($P_{CNF} \leq 7\%$)
- e) Number of SNs passing the CN-Filter ($\#_{CNF} \leq 64$)

Min-Setup: The Min-Setup minimizes the second in-network communication of the Distributed Event Detection that is in this setup rarely necessary for the CN-Filter process. We reflect this situation by minimizing the probability of passing the CN-Filter (P_{CNF}) to 0% and the number of SNs passing the CN-Filter ($\#_{CNF}$) to 0.

- a) Number of arising events per hour ($\#_E \leq 50$)
- b) Number of SNs affected by an event ($\#_{SN_{EV}} \leq 64$)
- c) Probability of a critical event ($P_{EVC} \leq 10\%$)
- d) Probability of passing the CN-Filter ($P_{CNF} = 0\%$)
- e) Number of SNs passing the CN-Filter ($\#_{CNF} = 0$)

Recommendations: The reference parameters can be lowered while maintaining the advantage of the energy demand. As depicted in Figure 5.23 the in-network maximized setup outperforms all other approaches under rather using increasing numbers of the affected SNs as well the SNs passing the CN-Filter and thus offers a huge amount of applications within this setting. If we reduce the in-network communication to a reasonable minimum the resulting CN-curve (Min-Setup) always outperforms the Multiple In-Network Fusions approach.

Figure 5.23 shows that the increasing number of involved SNs within the reference setup leads to a limitation for the applicable number of affected SNs as the processing time is consumed as soon as the curve ends. The Feature and Classification Fusion approaches are limited to 50 affected SNs. The Multiple In-Network Fusions approach is limited to 77 affected SNs, although the CN is able to handle up to 195 SNs as the lower lifetime boundary dictates the system's lifetime. The Distributed Event Detection is able to handle 64 affected SNs for the Max-Setup and 195 for the Min-Setup. We can see that the lifetime increase reached by Min-Setup towards the Max-Setup reaches a maximum of 8% at 64 SNs for this setup, but if we decrease the number of events per hour ($\#_E$) to only one event per hour this impact increases to 24%. The presented range of possible application that outperform all other information fusion scenarios is huge, but as a matter of fact an optimal setup recommendation has to be calculated based on the applications requirements individually.

In addition to these setup recommendations we stated initially in Section 5.2.1 that every higher level information fusion is able to support all lower level information fusions. As the Distributed Event Detection reflect the highest information fusion level we have access to all other fusion levels. In case of an application that does run within the recommended setups or tends to switch to a less efficient setup we can simply switch back to one of the better performing information fusion approaches integrated within our approach.

The maximum values for above introduced setups are so high that they seem to be out of focus for current applications and we recommend to use application with these or lower values to guarantee a better performing system. Nevertheless, the expected future miniaturization of the SNs' size and the currently heightened ongoing interest and focus in the direction of Mobile Edge Computing (MEC) is a good indication for applications in a not too distant future that will make full use of the maximum values of the reference parameters (a)-(e), showed above. In addition, we want to emphasize that these are the maximum recommended values, which clearly indicates that lower values are to be preferred and obviously give a better energy profile in general.

5.6 Classifier Selection

The *classification model* of the introduced Distributed Event Detection System supports various classification algorithms as introduced in Section 4.2.3.8.

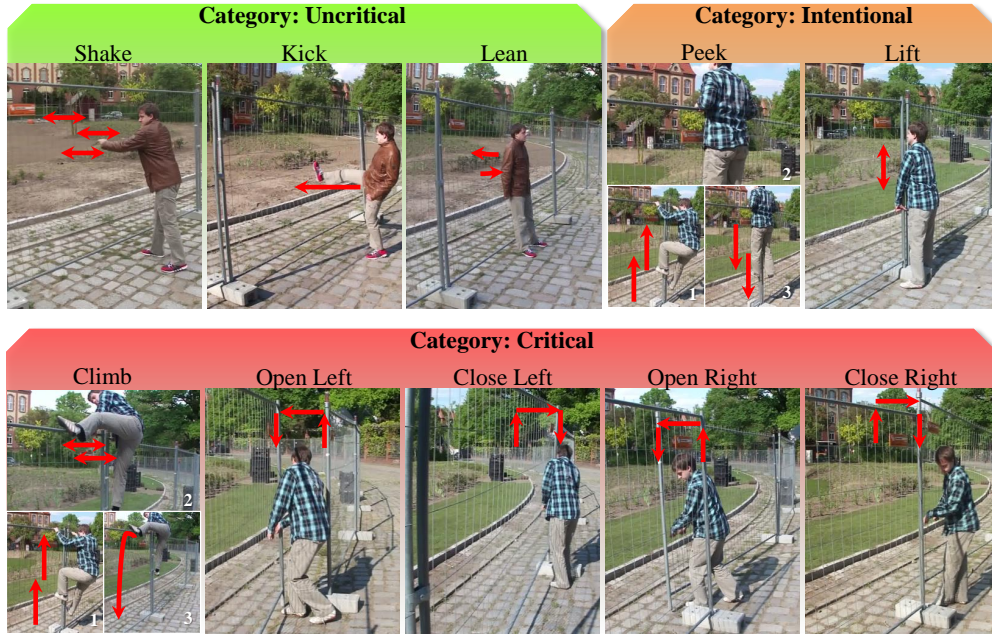


Figure 5.24.: [General Areal Categories] 10 fence events grouped in 3 event categories: Uncritical, Intentional and Critical, applied from [5].

To choose an appropriate classifier for our experiments we apply a real world training set of an example fence surveillance system to our evaluation framework to evaluate the potential classifiers, as introduced in [5]. The training set-up comprises a training data set of 300 events, created by 3 different test persons performing a training of 10 runs for each of the 10 event classes; the training setup is described in detail in Section 7.1.2. In the following sections we introduce our investigated events and derived categories and setup our system with the subsequently introduced events and categories in order to select an appropriate classifier for an example application.

5.6.1 Event Description

The investigated events in Figure 5.24 are derived from urban activities at and near fences during construction work, passer-by activities next to the fence, and typical intrusion activities. The *shake event* represents incidents such as heavy tremors or physical shocks caused by construction vehicles and machinery or by passers-by shaking the fence by hand. The *kick event* is an abstraction of all objects that accidentally fall into or are deliberately thrown at the fence. *Lean events* describe softer interaction with the fence including gusts of wind and passers-by or workers leaning against the fence. The *peek event* covers all actions where people try to intentionally peek over the fence by taking one step onto the fence and staying there for a while. The peek event could also be interpreted as the first part of an intrusion event and it is possible to use it as an early warning sub-event if necessary.

Lift events are performed by construction site employees as a preceding action if they want to open the fence. Lift events can also be caused by intruders trying to move the fence or checking the fence's weight and flexibility. The *climb event* is a representation of an intruder trying to climb over the fence. The climb event is performed by climbing up to the top of the fence and jumping down to the ground. All kinds of *opening* and *closing events* can be performed by construction site employees during their working period as well as by intruders trying to overcome the fence.

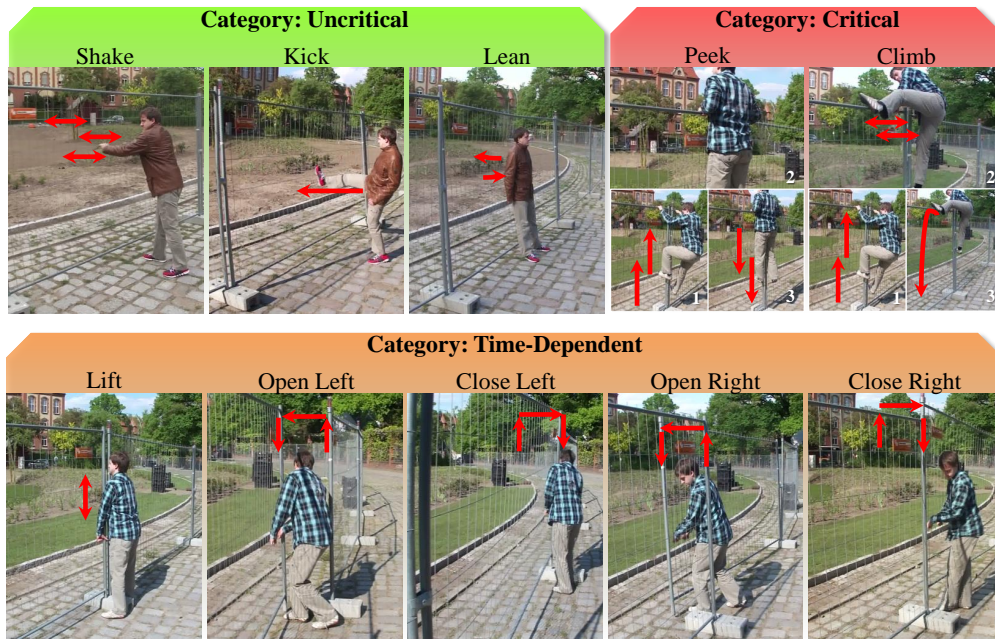


Figure 5.25.: [Construction Site Categories] 10 fence events grouped in 3 event categories: Uncritical, Critical and Time-Dependent, applied from [5].

The indication of the direction of an open or close event is necessary due to the orientation of the event towards the CN. If we pick the open left event the fence element on the left-hand side of the CN is opened.

5.6.2 Event Categories

In real world applications, it is important to distinguish between different categories of events. The event description is not always the most relevant information. Categories that represent, for example, critical events that have to be sent to a BS or uncritical events that do not need to be sent to a BS can leverage the WSN to a self-deciding entity. The event categorization is performed on-line within our *Application Filter* introduced in Section 4.3.3 in order to notify the user only in case of a relevant event. For this purpose, the given events are categorized into two different setups with three categories each, introduced in the following sections.

5.6.2.1 General Areal Categories

The *General Areal Categories* setup comprises the categories *Uncritical*, *Intentional* and *Critical*. This setup can be applied e.g. to a general area surveillance application. The relationship between the General Areal Categories and event class is depicted in Figure 5.24.

Some events need attention but do not indicate a critical event; we call them intentional events. All classes are assigned once to one of the categories. The category *uncritical* describes the common events shaking, kicking and leaning at the fence. The category *intentional* contains the events of lifting and peeking over the fence as it is not clear whether someone wants to break in or not. The *critical* category contains all events which point at a violation of the fence's integrity or a clear intrusion: opening or closing the fence from the left or right side as well as climbing over the fence.

5.6.2.2 Construction Site Categories

The *Construction Site Categories* setup comprises the categories *Uncritical*, *Critical* and *Time-Dependent*. The relationship between the General Areal Categories and event class is depicted in Figure 5.25. This setup can be applied e.g. to a construction site that has two time periods of interest: First, the work period where workers are all over the place with regular construction activities. Second, during the time period after work, some activities have a different meaning because of the special time context.

All classes are assigned once to one of the categories. The category *uncritical* describes the common events shaking, kicking and leaning at the fence – the event are uncritical despite the active time-zone. The category *critical* contains the events that are always critical despite the active time-zone. In detail, if someone peeks over the fence or climbs over the fence this, is always worth raising an alarm.

The time-dependent category contains all events which point at a violation of the fence’s integrity during the after-work time period, whereas they are uncritical if they arise during the work period: opening or closing the fence from the left or right side as well as lifting the fence are normal actions during the work period. In contrast, after work the fence should never be opened, closed or lifted.

5.6.3 Classifier Evaluation

We create and assess classification models for the *prototype classifier* [71], the “naïve” *Bayes (NB) classifier* [10] and the *KNN classifier* [63] with parameter $K = 1$ and $K = 3$, and apply the Evaluation Framework described in Section 4.2 and Figure 4.2 with the training data set introduced in Section 5.6.

Implementations for **NB** and **KNN** are already supported by Waikato Environment for Knowledge Analysis (**WEKA**). We added our prototype classifier for this theoretical evaluation to the **WEKA** implementation. We evaluated the classifiers by using the metrics *sensitivity*, *specificity*, *Positive Predictive Value (PPV)*, *Negative Predictive Value (NPV)*, and *accuracy* as introduced in Section 3.3.1 and as defined in [55]. We investigated two methods for evaluating the classifiers. The first method evaluates the classifier by distinguishing each of the 10 classes from one another (see left side of Fig. 5.26). The second evaluation method distinguishes two types of different application oriented categorizations with three categories each as introduced above. The results of the different categorizations are depicted by the mentioned metrics (see middle and right side of Fig. 5.26). With an additional confusion matrix it is possible to get a better insight into the details for the aforementioned averaged metric; to provide this insight we show the confusion matrix for all four classifiers distinguished for the 10 events in Figure 5.27, the general areal categories in Figure 5.28 and the construction site categories in Figure 5.29.

The evaluation result from **WEKA** (see Fig. 5.26 left-hand) shows that all algorithms perform well, as all metrics are at least above 90%. The evaluation result from **WEKA** of 10 different event classes shows in the left-hand of Figure 5.26 that **KNN** classifier has slightly more problems to successfully classify the given events in general. Nevertheless, all classifiers share roughly the same classes with decreased classification success. **KNN** classifier has slightly more problems to distinguish between open and close events and sometimes detects lift, lean, or climb events instead. The misclassifications are more spread within the field of classes for the **KNN** classifiers where the **NB** and prototype classifier have more specific misclassifications. The prototype classifier classifies some open left events as close left events and vice versa. The same problem arises for the open

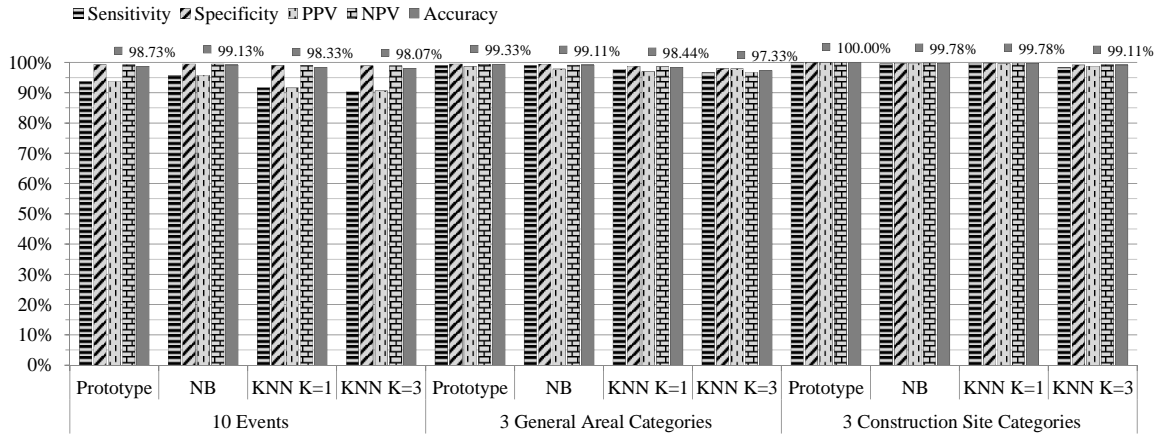


Figure 5.26.: Evaluation of four classifiers with the evaluation framework. Three setups with the averaged metrics are compared: 10 events, three General Areal Categories and three Construction Site Categories, applied from [5].

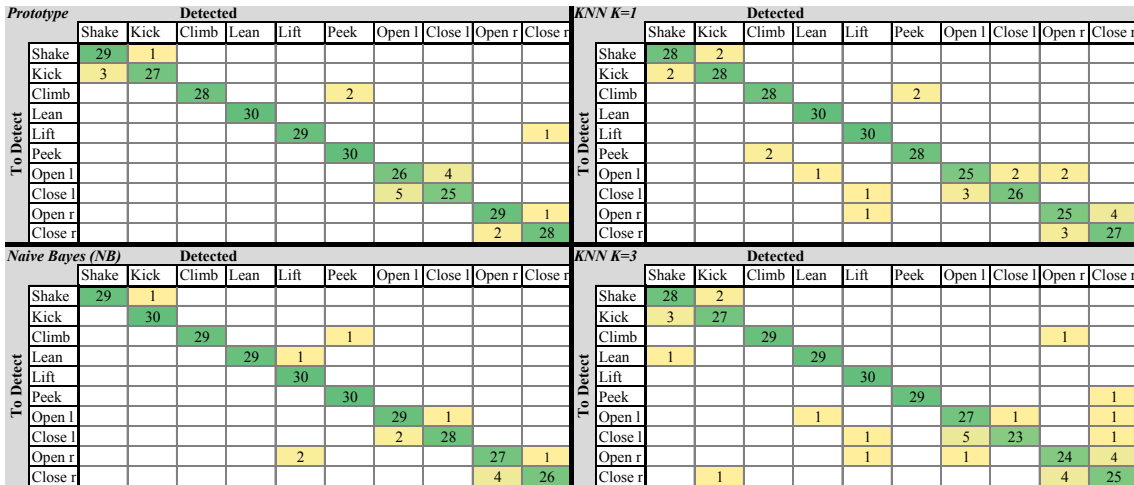


Figure 5.27.: Confusion matrix for the four classifiers using all 10 events

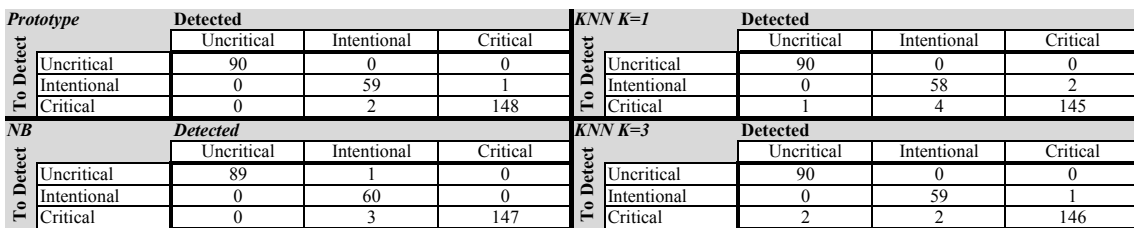


Figure 5.28.: Confusion matrix for the four classifiers using the three General Areal Categories, applied from [5]

right event that is misclassified in some case as a close right event and vice versa. This happens in a similar intensity for the NB. All classifiers seem to have slight problems with detecting shake events and kick events correctly as some misclassifications are observed for all classifiers, see Figure 5.28.

NB achieves slightly better results than the prototype classifier for the differentiation of 10 event classes. As our hardware and software support multi modal sensors, it is possible to improve open and close event detection for all classifiers by introducing a gyroscope.

Prototype		Detected			KNN K=1		Detected		
		Uncritical	Time-Dependent	Critical			Uncritical	Time-Dependent	Critical
To Detect	Uncritical	90	0	0	To Detect	Uncritical	90	0	0
	Time-Dependent	0	150	0		Time-Dependent	1	149	0
	Critical	0	0	60		Critical	0	0	60
NB		Detected			KNN K=3		Detected		
		Uncritical	Time-Dependent	Critical			Uncritical	Time-Dependent	Critical
To Detect	Uncritical	89	1	0	To Detect	Uncritical	90	0	0
	Time-Dependent	0	150	0		Time-Dependent	2	148	0
	Critical	0	0	60		Critical	0	2	58

Figure 5.29.: Confusion matrix for the four classifiers using the three Construction Site Categories.

For the second evaluation method all events are consolidated into three categories while we present two categorizations types (see middle and right-hand of Fig. 5.26). If we want to know whether an event is critical or not, all algorithms are able to answer this question with the given metrics with a high average accuracy of more than 90%. Further, the construction site categories perform better in general compared to the general areal categories with a slight advantage of 1% to 2% accuracy.

Some critical events have been classified as uncritical instead, which lowers the accuracy of **KNN** by 2% below the accuracy of the prototype and the **NB** classifier. In general, the prototype classifier performs as well as the **NB**, while investigating both confusion matrices in Figure 5.28 and 5.29 for the category evaluation, we see that only minor deviations occur. For the general areal categories in Figure 5.28 the prototype classifier classifies only three events incorrectly, the **NB** reaches four misclassifications, **KNN** $K = 1$ reaches seven misclassifications, and **KNN** $K = 3$ reaches five misclassifications. For the construction site categories in Figure 5.29 the prototype classifier classifies all events correctly, the **NB** causes one misclassification, **KNN** $K = 1$ causes one misclassification, and **KNN** $K = 3$ reaches four misclassifications.

From a technical point of view, the **KNN** has a slight disadvantage for embedded systems as it needs to store all training data on the node to calculate the K nearest neighbors. Additionally, we have to attest the **KNN** the lowest classification accuracy compared to other classifiers and the necessity to hold all training data on each sensor node; we do not recommend **KNN** for the given purpose. The **NB** and prototype classifier are both a good choice, while for our specific application the prototype classifier performs slightly better.

Hence, the prototype classifier performs best by distinguishing between events that should cause no alarm (uncritical events) and those that should cause an alarm (critical events). Intentional or time-dependent event categories can give the whole system a more diverse handling depending on the application. In general the application should be able to assist the responsible personnel to protect the area. One option could be to notify this personnel with clear decisions. The prototype-classifier deployment is able to support this approach best and distinguishes fence events that should cause an alarm from those that should not cause an alarm.

5.7 Feature Vector Evaluation

The feature vectors are an essential part of the Distributed Event Detection System as they describe the known and trained classes with the *Reference Vectors*. In addition, unknown events are described with the *Fused Feature Vector* introduced in Section 4.2.3.6. It is important to handle the *missing feature problem* which arises because of reference vectors that are differently affected in their characteristic in case of different events and it is interesting which features have to be

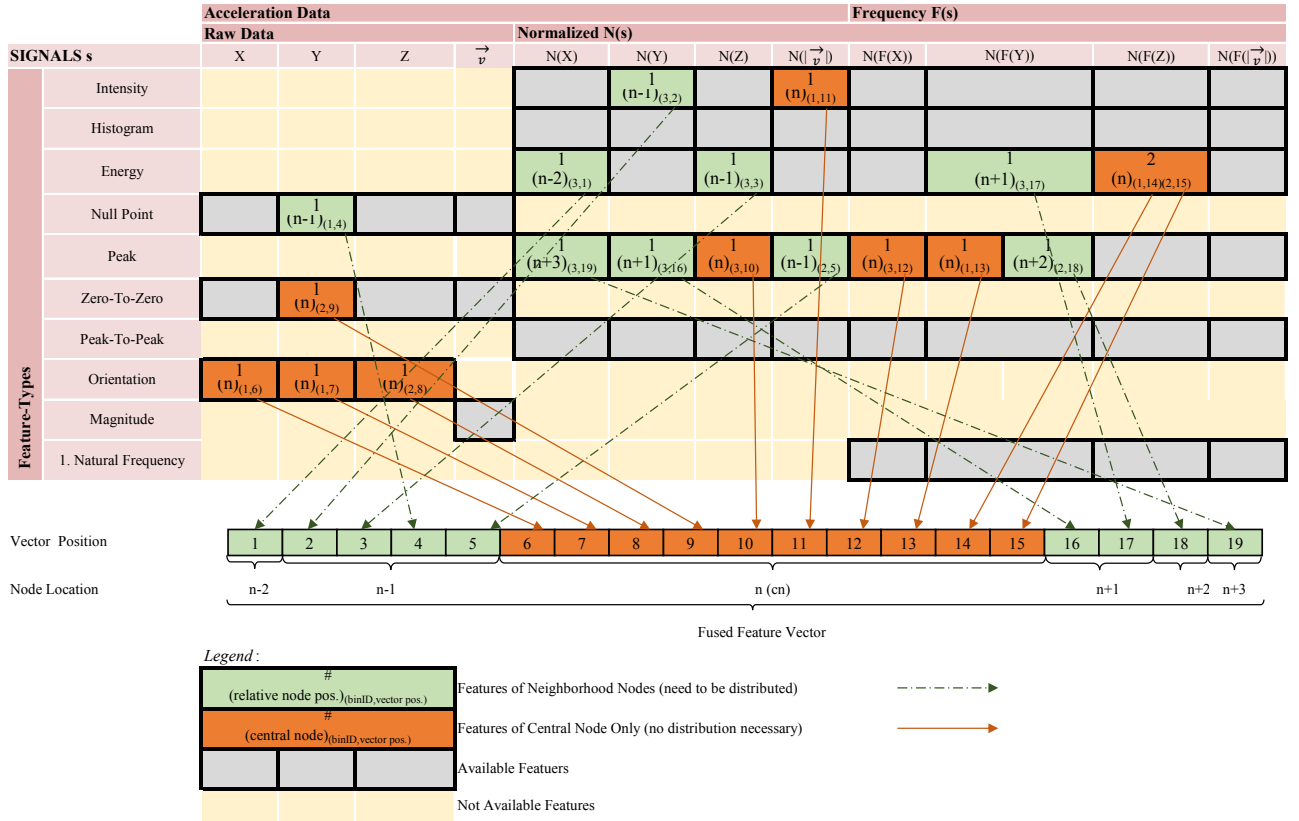


Figure 5.30.: Selected features for the fused and distributed feature vector from the Evaluation Framework for the trained classes, applied from [5]

distributed in order to be able to fuse an arising event correctly and efficiently.

5.7.1 Feature Fusion Inspection

The in-network information fusion of the Distributed Event Detection is first and foremost possible due to the distributed feature extraction of multiple SNs affected by the event. With the example of our fence surveillance system, we want to show which SN has to create which feature and which data have to be transmitted during this process.

We want to show the selected features in form of a feature vector from the Evaluation Framework in Figure 5.30 which has to be created during the Distributed Event Detection within the network at the CN. The according features are extracted during the final detection process on each SN.

All 9 green colored features except the 10 orange ones from node n – which represents the perspective of the CN – have to be sent into the neighborhood by all affected SNs. The orange colored features have to be calculated but not sent into the neighborhood as these features represent the CN. As we support multiple partitions of an event, features can thus be extracted and used for multiple partitions. In case of the fence surveillance systems we picked three partitions (as motivated in Section 4.2.3.3 in order to cover an event beginning, middle section and end section. The partitions are identified with so called binIDs in Figure 5.30. The position of a selected feature within the fusion vector is defined relative to the central node. For example, one energy feature for normalized x-values $N(X)$ is selected for SN $n - 2$, which means for the second SN to the left side of the CN. The feature is used within the third partition and will be positioned at

Relative SN Location	n-2	n-1				n (cn)										n+1	n+2	n+3	# of involved	
Fused Vector Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	SNs
Reference Vectors for all Class	shake	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	6
	kick	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	6
	climb	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	6
	lean	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	6
	lift	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	6
	peek	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	6
	open right	-	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	5
	close right	-	-	-	-	-	x	x	x	x	x	x	x	x	x	x	x	x	x	4
	open left	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	5
	close left	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	4
	<u>5.4 (average)</u>																			

x	Feature and SN used for this class
x	Feature and SN not used for this class
-	Feature and SN not used for this class

Figure 5.31.: Reference vectors extracted during the feature selection process for each class, showing involved and missing features because of structural feature absence and SNs.

the first position in the fusion vector.

We can see that the feature selection did not select any natural Frequency feature; no Peak-To-Peak feature and no Histogram feature was selected. In addition, no feature was derived from the normalized and frequency based magnitude signal $|F(N(\vec{v}))|$. In contrast, mainly peak, energy, orientation, and some intensity features have been selected by the system’s feature selection introduced in Section 4.2.3.6. The selected features directly describe the nature of the fence events. Fence movement Peaks, energy and the orientation of the fence play the main role in the event description, while frequencies are totally ignored. This makes sense as it is unimportant whether a fence oscillates at a specific frequency, while it is more important to detect steps (peak and energy) while someone is climbing on the fence or to detect whether someone is opening the fence (orientation and intensity).

As soon as the CN-Filter has detected the CN, exactly this CN builds the fused feature vector according to the relative positions to the neighboring SN according to the order of the features as determined during training with the bit-mask. For the resulting fused feature vector of the unknown event the ED to all trained reference vectors (cf. Section 5.7.2) is calculated and compared in order to find the shortest distance (Section 5.7.2). The resulting EDs provide a representation of the unknown event. Typically the shortest ED is used to select the according trained class as the classification result for the unknown event.

5.7.2 Reference Vector Inspection

In Section 4.2.3.6 we introduced how the reference vectors are created and that the vectors represent the different event classes. With the example fence surveillance application, we show how these reference vectors are represented depending on the physical event propagation. Figure 5.31 depicts that the feature selection for most of the event classes has the same result. In contrast, the open and close events – where a fence element is opened or closed – differ slightly in their representation. 4 event classes have a reduced reference vector that reduces the involved number of SNs to 4 to 5 SNs, while both open fence events affect five SNs and both close fence events affect four SNs. On average, this means that 5.4 SNs are affected by an event if we assume that every event is equally probable.

This behavior can be explained by the reduced event propagation that results from a slightly

reduced event intensity that is induced into the fence while opening or closing a fence element. If we open a fence on the left-side of the CN, the fence elements that are typically affected by events at the outermost right side are not affected anymore as the fence opening and closing events cause a reduced fence motion, which leads to a reduced spacial event propagation at the fence. The same effect happens for the typically affected outermost left fence elements during open and close right events.

The open events cause a slightly more intense event propagation as the initial opening procedure affects the whole fence, while the closing event is the most calm event of all events and affects SNs of the untouched fence side only during the very last and final part of the closing event procedure.

These events cause the subsequently introduced *structural feature absence* (see Section 5.7.3) during the distributed event recognition introduced in Section 4.3.2.

5.7.3 Missing Features

As introduced in [5], missing features in the event-describing feature vector are an important problem that need to be discussed, as they have a huge influence on classification. The first question is how missing features may occur.

- Transmission Failure:

The packet transmission during the *Feature Distribution* as introduced in Section 4.3.1.5 between two SNs may fail even with implemented re-transmissions, which leads to missing features within the fused feature vector of the unknown event. A transmission failure that happened during the training process block (a) can be precluded by the supervisor who may simply re-initiate a specific training run. Another reason for a transmission failure can be a depleted battery or a defective SN.

- Feature Creation Failure:

Some features can fail to be created from a given signal if the signal does not deliver all necessary information needed to create the feature, which may be the amount of data, a high enough amplitude, or other parameters that are feature relevant. We handle such missing features in the same way we handle transmission failures; as the creation failure has the same effect on our proposed system, the feature is sometimes not available during the Distributed Event Detection process. An example for a feature creation failure could arise during the calculation of the natural frequency which depends very much on the data length and amplitude.

- Structural Feature Absence:

Events may not all affect the same number of SNs as this depends on the event characteristic itself. For example a fence opening event affects fewer SNs than a fence shaking event, as the event distribution is physically disrupted by breaking the chain of fence segments. As a result, we have different feature vector dimensions available. The structural absence of features is further discussed in [122] and a discussion within the context pattern recognition is given in [123].

5.7.3.1 Re-Balance Method

As classifiers like the NB classifier can handle missing features better than our preferred distance based classifier as remarked in [64] we need to improve our nearest prototype classifier.

In the following paragraphs we discuss a solution for the previously introduced missing feature problem. For the solution we need to support and address the distance measure based classifiers as these components are necessary for our detection system.

The obvious solution – to estimate a missing feature – is only appropriate if the reason for a missing feature is known. If we could determine whether a feature is missing because the event characteristic does not affect a certain sensor node (Structural Feature Absence) or because a transmission error arose (Transmission Failure), we could estimate the feature value based on previous measurements for the latter case.

Another method to handle missing features is *Case Deletion*. It ignores the whole feature vector if at least one feature is missing. This approach restricts the applicability of the Distributed Event Detection as only those event types can be handled that always affect the same number of SNs. In case of a security application we need to evaluate each arising pattern, even if a SN is unable to operate.

Zero-Imputation simply replaces all distance calculations of missing features with zero and assesses these features with a perfect feature match as the distance calculation for two features is only zero if their value matches 100% exactly. As this case is very unlikely we do not want to use this approach.

The core of the classification process for all distance based classifiers is the distance calculation between the feature vector of the unknown event and the reference vectors. To solve the missing feature problem we derive two cases to be distinguished [5]:

- Problem: Varying Unknown Feature Vector Dimensions

One or more features within the unknown fused feature vector is missing based on a transmission failure or feature creation failure. All distance calculations depend on the fused feature vector as all calculations use the same number of features. The problem is how to match the unknown correct dimension of the feature vector within this distance calculation.

Solution: Feature Deletion

A straightforward solution for this problem is the *Feature Deletion* approach. The Feature Deletion skips only the distance calculation for the feature value pairs in which the value is missing in the reference vector or in the unknown event vector or in both. We calculate the distance from the remaining features only.

- Problem: Varying Reference Vector Dimensions

One or more features within one or more reference vectors are missing. This case reduces the amount of features involved in the distance calculation to a fused feature vector. The problem is how to match the different dimensions of the reference vectors within this distance calculation.

Solution: Re-Balancing The *Re-Balancing* [5] approach elegantly covers both aspects, missing features because of the varying reference vector dimensions and the varying unknown feature vector dimensions. The Re-Balancing method affects the comparability of the resulting distances, as the reference vectors with smaller dimensions have an advantage

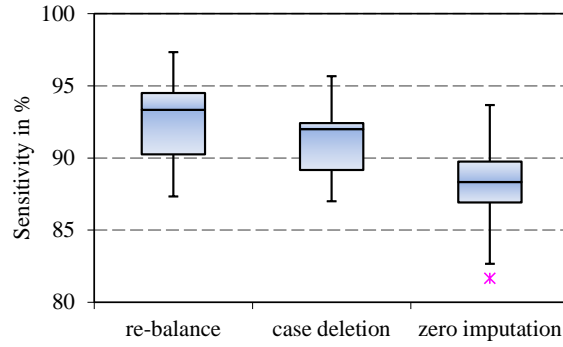


Figure 5.32.: Missing feature compensation method comparison with prototype classifier, adapted from [5].

in the calculation of the Euclidean distance as discussed for the *Zero-Imputation* approach. This imbalance can be resolved by calculating the vector distances relative to the amount of features considered during the calculation as shown in Equation (5.25). First we have to calculate the distance D based on the previously introduced feature deletion approach in order to calculate the distance for those pairs of features that are available. F_{all} is the number of all expected features while F_{missing} reflects the number of missing pairs of features resulting from a comparison with a reference vector and the unknown event. Hence, the number of missing feature pairs F_{missing} depends on the investigated reference vector compared to the feature vector of the unknown event that represents one of the trained classes. The improved distance D_{balanced} is calculated using the previously calculated distance D and an additional distance approximated based on the distance D as shown in Equation (5.25):

$$D_{\text{balanced}} = D + \frac{D}{F_{\text{all}} - F_{\text{missing}}} * F_{\text{missing}} \quad (5.25)$$

We compared the *re-balancing* method with the mentioned *case deletion* method as well as with the *zero-imputation* method as can be seen in Figure 5.32.

All experiments are based on the prototype classifier utilizing the Euclidean distance as a distance function. We performed eight experiment runs with every experiment being based on the cross-validation check discussed in Section 4.2.3.6 with differing feature sets. Independent from the feature set, the results showed that each feature set performed differently, but that in all cases the re-balance method performed best, case deletion performed second best, and zero imputation came last.

The box-plot in Figure 5.32 shows that the re-balance approach performs best with a positive impact of 1% to 5% sensitivity improvement if we compare the median values. We can conclude that the new re-balance method can be highly recommended for our Distributed Event Detection system, although case deletion also performs adequately. Nevertheless, case deletion does not solve the problem but rather ignores it, which is not the aim of our approach.

5.7.3.2 Classification Performance with Inoperative Sensor Nodes

Packet loss or inoperative **SNs** during the Distributed Event recognition may cause not all feature data to be accessible at the **CN** in order to classify an unknown event. We used the proposed evaluation framework to analyze the influence of inoperative **SNs** on the resulting classification

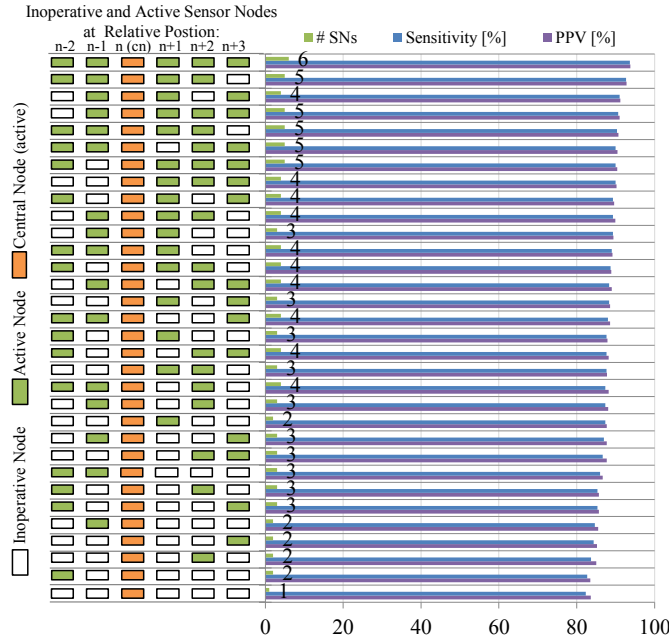


Figure 5.33.: Sensitivity performance comparison with 32 combinations of inoperative wireless sensor nodes, applied from [5].

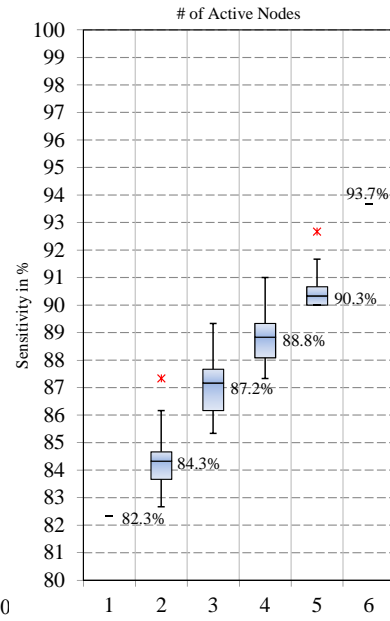


Figure 5.34.: Box plot evaluation of the six different numbers of active SNs, applied from [5].

sensitivity. Therefore, we process all 32 combinations in which the feature data of one or more neighboring nodes are not available. If a packet is missed, this case is handled by the re-balance method introduced in Section 5.7.3.1. It is assumed for all cases that the correctly detected CN is active and operating. For this evaluation we average the resulting sensitivity for the 10 event classes described in Section 5.6.1. The corresponding classification results are depicted in Figure 5.33 and Figure 5.34. The results are derived from evaluations that are based on the fused feature vector introduced in Section 5.7.1 in Figure 5.30.

It is important to show how the Distributed Event Detection performs and whether it is still able to detect events properly if relevant SNs are inoperative. On the other hand, it is important to show that adding SNs to a single SN adds a benefit to the proposed detection approach.

Let us investigate the features: Most features for the classification are contributed from the CN with 10 out of 19 used features, see Figure 5.30. We investigate how effectively this single SN can classify events. If the CN becomes inoperative, another node automatically takes the role of the CN as described in Section 4.3.2.1, hence, it is investigated how possibly inoperative SN next to the CN influence the classification results.

Figure 5.33 depicts the different settings with active and inoperative SNs, while the CN is depicted as orange, active SNs are depicted as green and inactive SNs are depicted as white. The corresponding location of the SN ranges from n for the CN to $n - 2$ for a maximum of two left-oriented SNs and to $n + 3$ for a maximum of three right-oriented SNs. Additionally, the sensitivity is depicted in a blue bar chart as well as the number of active SNs are depicted in a green bar chart.

Figure 5.34 depicts the box-plot for the number of active SNs and depicts the median value for one to six active SNs.

First of all, based on the already good sensitivity of 82.3% of only one active SN in Figure 5.34, we

have to state that the **CN** alone is very effective in detecting events. Nevertheless, the sensitivity can be increased by about 11.4 percentage points by adding **SNs**, up to 93.7%. This is very important for the Distributed Event Detection in general as it shows that there is a true benefit for the idea of additional **SNs** sharing the task to detect an event together. This is obvious for rather complex movement events of a human body, as every part of the body is doing something different. In the given example we use acceleration data of a fence and each fence element is oscillating very much like its neighbor. A difference between the fence elements was to be expected, but a clear improvement of 11.4 percentage points is much more concrete and confirms this expectation. We can state this even under hard conditions where the impact of the Distributed Event Detection is provable.

4 to 5 active **SNs** reach a sensitivity between 87.3% and 92.7%. 2 to 3 **SNs** reach a sensitivity between 82.7% and 89.3%. If only the **CN** is active, the performance is worst with 82.3%. We can conclude that the Distributed Event Detection sensitivity increases if a higher quantity of **SNs** is involved in the event detection process. The robustness of the system depends on the strength of the remaining features. We clearly see that even if all affected neighboring **SNs** become inoperative, the **CN** is still able to operate with a decreased detection sensitivity.

To conclude, we state that the Distributed Event Detection system offers a good sensitivity even with one **SN**. In order to increase this sensitivity by about 11.4 percentage points the costs of 5 additional **SNs**, including the in-network communication between the nodes, have to be raised.

Together with the energy evaluation it is clear that there are applications that will heavily benefit from the Distributed Event Detection as it increases the lifetime and the detection sensitivity and offers new features such as an automated in-networking subsequent processing of tasks that are linked to the detected events.

CHAPTER 6

Distributed Event Detection Platform

6.1 Platform Architecture

The aim of the platform architecture (first versions mentioned in [5,101,117,124,125]) is to provide a large amount of applications. Figure 6.1 shows that the platform architecture is composed of two layers: the hardware layer and the application layer.

6.1.1 Hardware Layer

We start with the lower layer, the hardware layer, which contains first and foremost the **SN** itself. The **SN** is exchangeable and we give references to two supported **SNs**, the AVS-Extrem **SN** [102] which is based on the MSB-A2 **SN** design [126], and the F4VI **SN** series introduced in [127].

Each **SN** offers the functionality of a transceiver to transmit and receive data packets, multiple sensors to gather data from the environment, and a permanent memory to store the classification model. The whole system is optimized for an ARM-architecture based **MCU** and a CC11xx series based transceiver and implements a FAT32 system for SD-Card management. Drivers for multiple acceleration, inertia sensors, humidity, and temperature sensors are available.

We use the FireKernel, which relies on the low energy concepts of Real-Time Operating System (**RTOS**) introduced in [4] and supports threading and priority-based preemptive multitasking which enables continuous data sampling while communication with other **SNs**. For communication purposes, the FireKernel already includes the **MMR** routing protocol introduced in [4,117].

MMR reuses the aspects of the Ad hoc On-Demand Distance Vector (**AODV**) and Dynamic Source Routing (**DSR**) protocols. In order to take advantage of hop-by-hop routing during data transmission, the **AODV** routing principle is used, while the principle of **DSR** enables **MMR** to update its own routing tables by investigating and collecting all route information from relayed packets. With the help of sequenced packets, unintended packet loops are avoided and the freshness of a route is maintained.

MMR has been further developed in order to support multiple base nodes as introduced in [102, 117]. The routing protocol communicates with a so-called virtual base station that eventually relays the packets to the real **BS**. These virtual **BS** are connected to the **BS** wirelessly or wired to all base nodes in order to ensure a fault tolerant connectivity within the **WSN**. However, we do not address this feature in more detail in the current thesis.

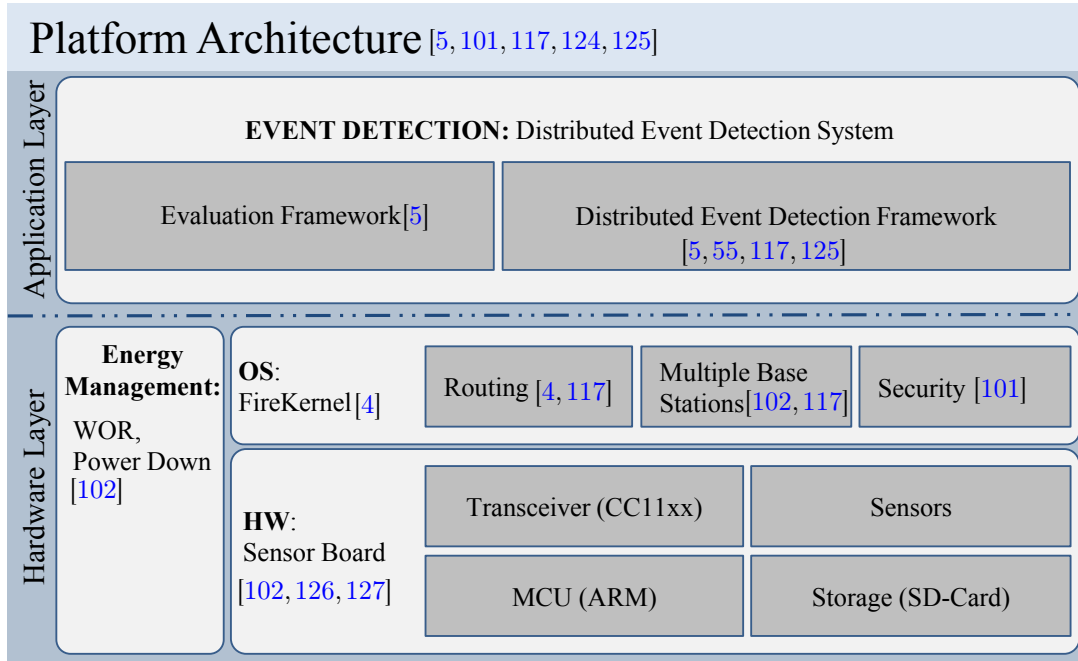


Figure 6.1.: Platform Architecture depiction composed of the Hardware and Application Layer with references to in-depth publications for further details.

In order to provide encrypted communication, a security layer is provided that supports various applications with all security relevant requirements: confidentiality, authenticity, integrity, freshness, semantic security, availability, and access control of transmitted data by using a symmetric cipher-algorithm in Cipher Block Chaining (CBC)-mode. For further details, refer to [101].

In order to maintain the SN's lifetime, an energy management system supports the application with WOR to reduce the transceiver power consumption and numerous MCU-specific energy saving techniques like power-down and idle mode. The SNs should be equipped with energy efficient sensors to wake-up the MCU when an event threshold value is exceeded, otherwise the MCU remains in the power-down mode. Both of our example SNs support the platform with appropriate sensors as the 3D-accelerometer SMB380 [128] from Bosch or MPU-9150 [129] from Invensense.

6.1.2 Application Layer

The second layer of the platform architecture of Figure 6.1 comprises the software content of the present thesis and is called *Application Layer*. The application layer contains the event detection application introduced above as the Distributed Event Detection System which was introduced in [5] and introduced with extended detail in Section 4. The first part of the Distributed Event Detection System is the Evaluation Framework introduced in [5] and introduced in further detail in Section 4.3. The second part of Distributed Event Detection System is the Distributed Event Detection Framework introduced in [5, 55, 117, 125] and introduced in further detail in Section 4.2. The description, problem inspection and solution, and the performance and applicability evaluation of the Distributed Event Detection System is the main topic of this thesis, therefore we refer to the previous chapters for further introductions as well as to the subsequent chapters for the evaluation of the suggested system.

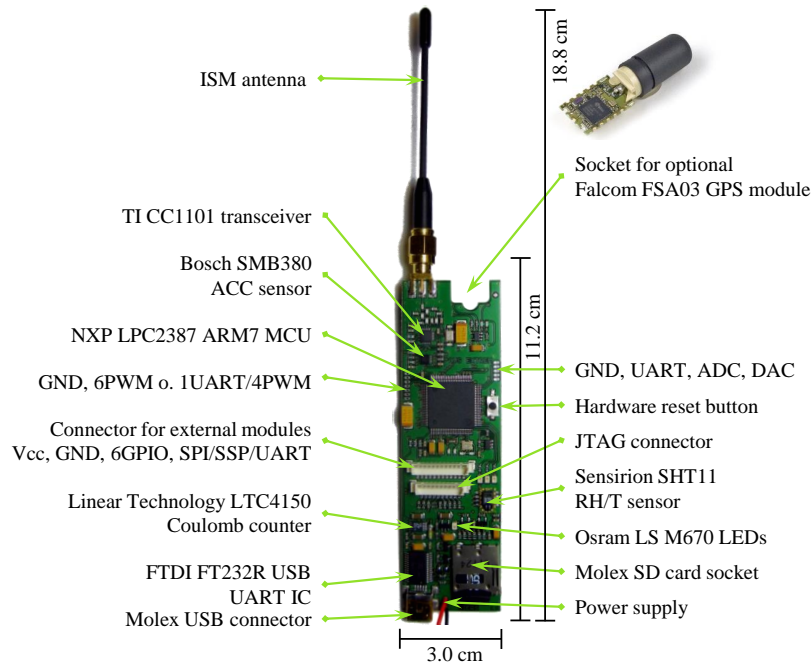


Figure 6.2.: AVS-Extrem hardware sensor node [102]

6.2 Sensor Nodes

Most of our experiments use the *Autonome vernetzte Sensorsysteme in Extrembedingungen (mst-AVS)* (*AVS-Extrem*) SN described in the following, as this SN was designed in the earlier stages of the presented thesis. The new F4VI SN series was created based on the experiences and the lessons learned about our *AVS-Extrem* SN during the experiments and development. Nevertheless, the latest development of the F4VI SN series is in use by multiple projects currently in preparation, e.g. [60, 127].

6.2.1 AVS-Extrem Sensor Node

We designed the *AVS-Extrem* SN [102] with the goal to provide a tailored hardware including a 3-axis-accelerometer for our implementation of the fence surveillance system while also supporting other applications like bridge surveillance or sport devices, see Figure 6.2. As a highly restrictive requirement we had to deal with the SN's shape as it has to fit into a construction site fence rod in the first place while being mounted within a casing including the battery supply. The resulting physical dimensions of the SN's new Printed Circuit Board (PCB) are: length: 11.2 cm (including the antenna 18.8 cm), width: 3.0 cm, thickness: 0.6 cm.

Inspired by the MSB-A2 SN [126] developed at the *Freie Universität Berlin*, the *AVS-Extrem* SN comprises an NXP LPC2387 32 Bit as part of the ARM7TM MCU family operating at 72 MHz [130]. The memory of the MCU is divided into 512 g on-chip flash memory and 96 kb Random Access Memory (RAM). The SN's weight is 17 g without and 22 g with attached antenna.

For communication purposes we use the low power CC1101 transceiver [131] from Texas Instruments in order to support a broad range of communication frequencies (315, 433, 868, 915 MHz) for the Short Range Devices (SRD) and of the Industrial, Scientific and Medical (ISM) radio bands, with our experiments always being set to 868 MHz for SRD. The CC1101 is driven by a 26 MHz

clock.

The 3-axis-accelerometer SMB380 [128] offers a comparable high sampling rate of up to 1.500 Hz and a sensitivity ranges of ± 2 , ± 4 and ± 8 g for our motion-centric applications. The sensitivity up to 8 g was helpful to detect for example high acceleration shocks during kick and climb events which turns out to be our general setting for experiments with this SN.

In addition, an integrated logic detects the event preambles with a predefined threshold value register. In general, the ARM7 MCU is set to Power Down (PD) mode that switches the SN to the operating mode by an interrupt for further processing if the difference between two successive moving averages exceeds the threshold. We calculate the threshold during the system's start up sequence and after each event in order to avoid false alarms caused by environmental noise provoked by weather conditions such as wind or changes in fence positions. Additionally, this allows us to handle manufacturing tolerances of internal converters in the sensors. During experiments, the node-specific and dynamically gathered thresholds reached a maximum of 0.2 g, which guarantees a highly responsive system in case of an event [124].

In addition to the core components explained above, the PCB comprises the temperature and humidity sensor SHT11 from Sensirion [132], as well as a Molex microSD Card connector [133] to the store prototype vectors. The LTC4150 coulomb counter [134] can be used to monitor battery states. Furthermore, peripheral Universal Asynchronous Receiver/Transmitter (UART) ports and connectors for external modules depicted in Fig. 6.2 are used as development interfaces and to extend the applicability in stages. In order to provide outdoor localization of the position of the SN, an optional GPS module from Falcom is attachable (FSA03 [135]). The Molex Universal Serial Bus (USB) 2.0-Mini-Interface supports the charging process, energy supply, flashing process and provides the serial interface over the FTDI-Chip [136].

6.2.2 F4VI2 Sensor Node

With the experiences of our predecessor SN AVS-Extrem, we designed a new SN F4VI2 [127] driven by the needs of the application domain for distributed motion detection in WBANs, see Figure 6.3. As a main requirement we had to tailor the SN's shape to a minimum in order to make it as comfortable as possible to wear these SN on the body or underneath clothes. Additionally, the new version of our SN had to fulfill the same needs and more as the AVS-Extrem SN. We discarded all unnecessary external connectors as well as the modularity for external modules in order to minimize the SN's shape. Of course, we heavily benefit from the updated and new MCU-Technology as well as System-in-a-Package (SIP)-components available since 2014 for our future research. The resulting physical dimensions of the SN's new PCB are: length: 3.6 cm, width: 2.4 cm, thickness: 0.8 cm. A U.FL socket is available for the flexible antenna in order not to enlarge the dimensions of the SN.

According to the intended applications, more available resources compared to the AVS-Extrem SN are needed in order to cache longer chunks of data and process the data within short time slots. We approach this on the F4VI2 by utilizing the STM32F415RGT 32-Bit as part of the ARM®32-Bit Cortex®-M4 Central Processing Unit (CPU) family operating at a maximum of 168 MHz. The memory of the MCU is divided into 1 Mbyte on-chip flash memory and 192 kb Static Random Access Memory (SRAM) and supports our applications with an integrated Floating Point Unit (FPU) and Digital Signal Processing (DSP)-instructions support [137]. The SN's weight is 6.1 g including the lightweight antenna and 6.7 g with additional PAN1740 Bluetooth® module(BLE).

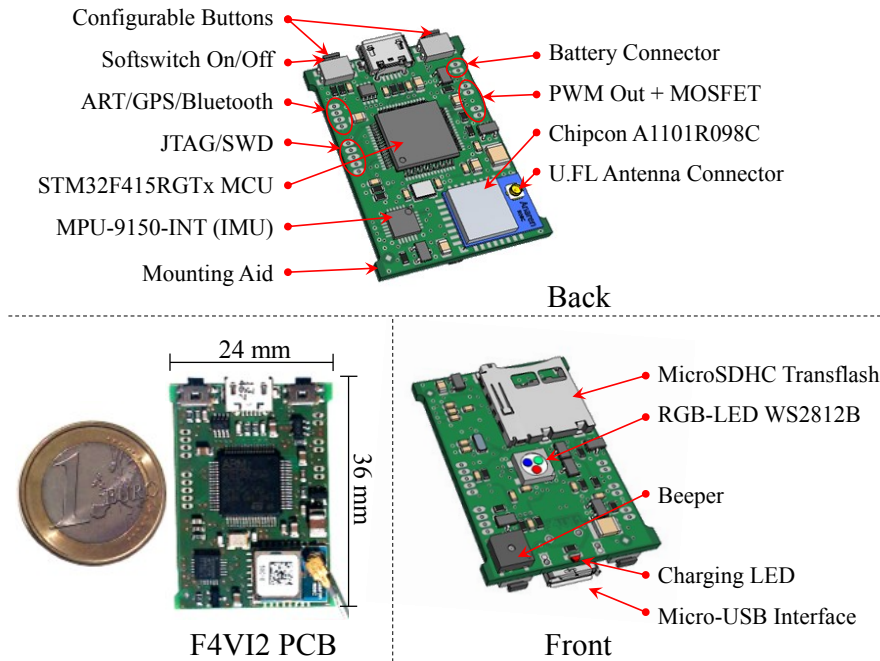


Figure 6.3.: F4VI2 [SN](#) and casing with component description in concept and assembled development stage, applied from [\[127\]](#).

For wireless communication between [SNs](#) we chose a compact 868 MHz transceiver TI CC1101 driven by a 26 MHz clock and integrated into the A1101R08C Anaren [SIP](#)[\[138\]](#), whose output power level is configurable to fit short-range environments such as [WBANs](#) with several [SNs](#) on one extremity. As a theoretic downside, the frequency range of this chip is limited from 868 MHz to 870 MHz, but since our experiments are always set to 868 MHz for [SRD](#), this posed no limitation. Although the frequency limitation exists in the datasheet, we could not confirm a true limitation of available frequencies during our internal tests with the Anaren [SIP](#), hence we conclude that currently, the A1101R08C is universally usable.

To sense linear and angular motion, we use the MPU-9150TM [\[129\]](#) integrated nine degrees of freedom motion tracking device. The integrated digital processing unit features offloading of sensor calibration and sensor fusion algorithm from the host processor. The [MPU](#) integrates a 3-axis [MEMS](#) gyroscope, a 3-axis [MEMS](#) accelerometer, a 3-axis [MEMS](#) magnetometer and a Digital Motion ProcessorTM hardware accelerator engine. The sensitivity of the accelerometer has ranges of ± 2 , ± 4 , ± 8 and ± 16 g.

The gyroscope offers a comparable high sampling rate of up to 8,000 Hz and the full-scale range of the gyro sensor may be digitally programmed to ± 250 , ± 500 , ± 1000 , and ± 2000 Degrees per Second ([dps](#)).

An optional Molex microSDHC Card connector [\[133\]](#) is used to store network configurations, firmware images and reference data assigned to distributed event detection and evaluation. The user interface of the F4VI2 consists of two buttons, one bright RGB Surface-Mount Device ([SMD](#))-Light-Emitting Diode ([LED](#)) WS2812B [\[139\]](#) for visual feedback and a beeper for acoustic feedback. Two Pulse-Width Modulation ([PWM](#)) outputs are available for e.g. optional motors with the intention to provide haptic feedback in future projects. Via external connectors, it is possible to support alternatively [GPS](#), Bluetooth[®] serial bridge, or [UART](#). The charging process can be performed via the Micro-[USB](#) interface. The [USB](#) interface can be programmed as a [USB-Slave](#)

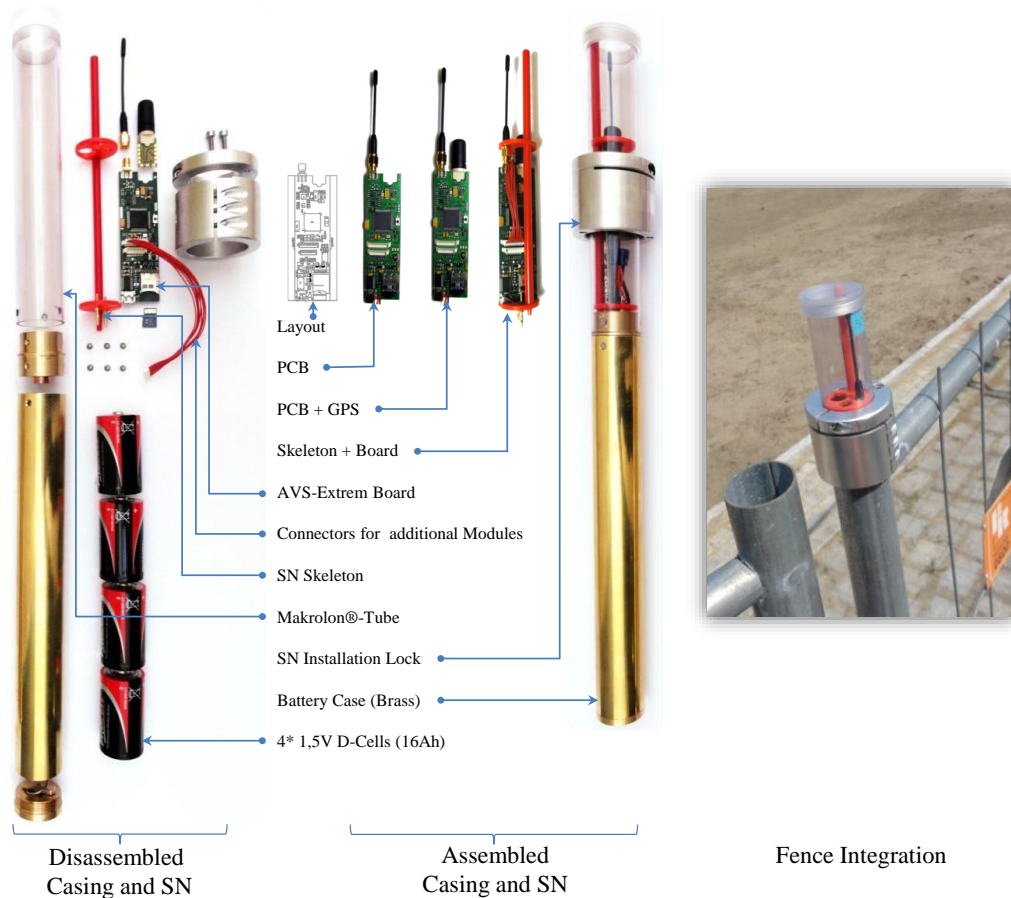


Figure 6.4.: AVS Extrem sensor node and casing: Disassembled, assembled and fence integration visualization

or [USB-Host](#) device or can be used to flash the [SN](#) with an application.

6.3 Casing

The importance and advantage of a robust casing has to be seen in the enhanced deployment and handling of mostly fragile embedded hardware. A second advantage of a tailored casing is the repeatability of deployments which allows us to avoid different mountings or an accidentally deviating setup. A casing allows us to ensure a setup with a minimum of varying influencing factors that have an impact on the experiments, such as temperature and humidity. Whereas inconsistent mounting, orientation as well as an overall simple installation and handling of the [SNs](#) helps to develop meaningful scientific results. In addition, alternative energy technologies can simply be exchanged with the batteries if standard sizes are employed within the casing.

6.3.1 AVS Casing

To deploy the [SNs](#) along the fence (Fig. 6.4), we have to solve several engineering problems that are typically ignored in simulations because of the difficulty in simulating the exact behavior. These problems are long-time battery supply and robustness against weather conditions as well as a rugged casing that meets the requirements of a construction site installation. Further we need a

lock to enable 24h – 7d deployments in real world environments.

As introduced in [102, 117] it is important to integrate as much of the SN as possible into the vertical rod of a zinced steel fence, as the fence offers a high grade of robustness by itself, see Fig. 6.4. In contrast to this benefit, the antenna has to stay outside the fence as the vertical rod affects wireless communication like a Faraday cage.

The usage of standard D-Cells enables a flexible design and delivers a stable capacity even during low temperatures. The battery case of our casing – as depicted in detail in Figure 6.4 – is a brass hull which provides solidity and conductivity for the ground connector. A PVC isolated copper core connects the positive pole within the brass coupling connector. The SN itself is held in the correct position by a red Makrolon® skeleton which connects a connecting plug to the SN. The skeleton with the SN can be plugged into and unplugged from the casing as a whole. A Makrolon® tube surrounds the SN in order to support undisturbed radio communication and to guarantee a high resistance against a wide range of weather conditions. The advantages of Makrolon® are its resistance to water, multiple mineral acids, oils, and fats; it is a good electrical insulator and has heat-resistant and flame-retardant properties, as well as a high impact resistance, rigidity, sturdiness, and toughness.

The battery case is shuttered with a screw cap which is actuated by a spring. In rare cases, a vertical shock may compress the spring, which disconnects the battery. This problem is solved by an additional capacitor to bypass a temporary interrupt to the power supply. To attach the SNs to the fence, a two part installation lock is used to fix the node at the lock and the lock at the fence for security reasons and to prevent easy thievery. Problems with dented and cement polluted fence elements confirmed in [5] that the environment demands a robust casing. We had to bulge and clean some fence elements with self-made steel wedges to be able to integrate the SNs properly.

6.3.2 F4VI2 Casing

We created a setup to integrate the F4VI2 SN into a compact mm * 52 mm * 17 mm casing [127] which makes use of a 120 mAh battery to provide long-term applications and optimized the visibility of the LED beam angle with a light diffuser integrated in the cap. The 3D-printed casing can be worn on the body by using an easy-to-attach Velcro® tape that can be edged through the casing as can be seen in the assembled row of Figure 6.5. In addition, the casing has some free space to allow the integration of a small Bluetooth® (BLE) compatible module to also communicate with mobile devices, as can be seen in the exploded view Figure 6.5. The 3D-printed casing ensures a very lightweight material. The casing itself is comfortable to wear and clearly shows the direction of any future developments that will take place in the area of WBAN. Such devices will be much smaller in the near future and will provide integrated solutions such as the suggested Distributed Event Detection as a SIP solution including a software library and predefined classification models.

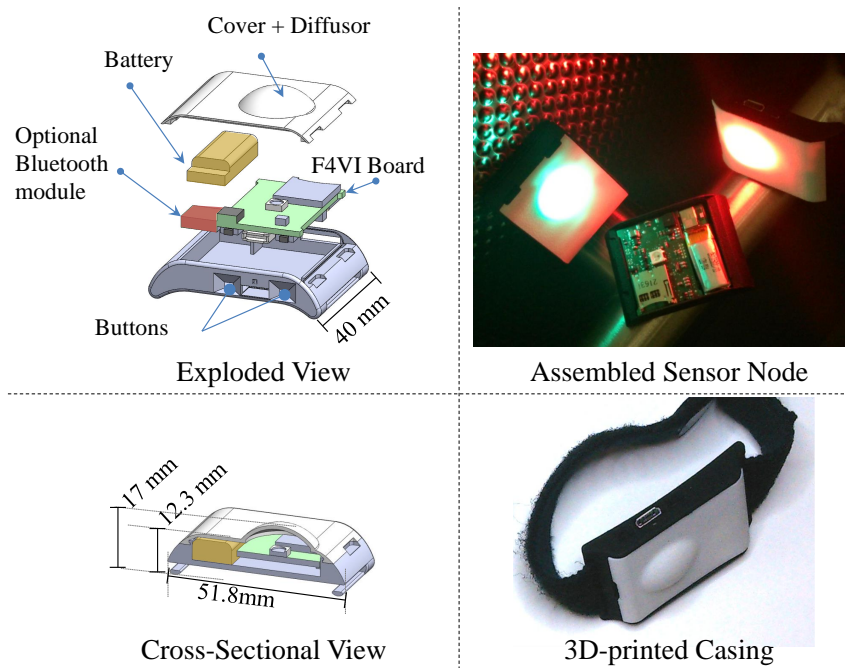


Figure 6.5.: F4VI2 SN and casing with component description in concept and assembled development stage, applied from [127].

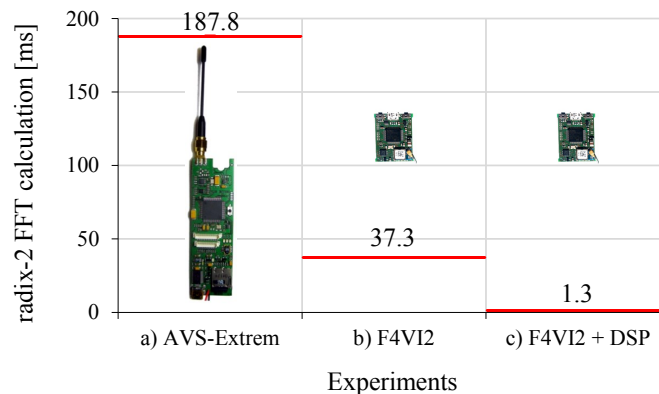


Figure 6.6.: Performance comparison of both used sensor nodes, adapted from [127].

6.4 Hardware Evaluation

We want to compare our two SN in order to give a short performance overview. It is worth mentioning that these SNs are from different generations: The AVS-Extrem SN was produced in early 2011 in the German Federal Ministry of Education and Research (BMBF) Project AVS-Extrem. In contrast, the F4VI2 SN was produced in late 2015 in the BMBF VIP-Project Validierung des Innovationspotentials verteilter Ereigniserkennung in drahtlosen Sensornetzen (Project-ID: 03V0139) (VIVE) and uses up-to date hardware for the time of creation. Hence, the differences shown should not be seen as a competitive comparison but more as evolutionary progress.

6.4.1 Calculation Performance

As first introduced and presented in [127] we compare the calculation performance of the two SNs [AVS-Extrem](#) and F4VI2 introduced in Section 6.2.1 and Section 6.2.2 in order to give a short prov of functionality and performance. As a representative calculation example we use the time needed to perform a typical natural frequency peak detection using the well known radix-2 Cooley–Tukey [FFT](#) algorithm [81]. We use this calculation for our subsequently described application for bridge surveillance in Section 7.4 in order to calculate the first and second natural frequency features.

We perform three experiments: The first is performed on the [AVS-Extrem SN](#) (a), the second on the [F4VI2 SN](#) (b), and the third experiment (c) uses a [DSP](#) optimized implementation of the [FFT](#) algorithm to support the [DSP](#)-instruction set of the [F4VI2 SN](#). All SN are exposed 10,000 times to a continuous 50 Hz vibration actuated by a physical oscillator (VEB ROBOTRON Otto Schön) and should detect exactly this frequency.

In Figure 6.6, we show the different performances possible with our introduced SNs. The [AVS-Extrem ARM7TM](#) based SN needs 187.8 ms for [FFT](#) calculation, the [Cortex[®]-M4](#) based [F4VI2 SN](#) reduces this calculation time to 37.3 ms (speedup factor 5). By applying a [DSP](#) optimized code for the [F4VI2 SN](#) the calculation time can be reduced to 1.2 ms which is a huge step to a new generation of performance. For future applications we highly recommend using the [F4VI2](#) if the shape format fits the needs. In the case of the subsequently introduced Fence Surveillance system (Section 7.1), Bridge Surveillance system (Section 7.4), and Sport Device (Section 7.2) we did not have the current technology of the [F4VI2 SN](#) available.

In addition, Ziegert et al. compared the energy consumption of both SNs in [127] and stated that the [F4VI2](#) reaches a much better footprint with 3.5 mW base load towards 8.1 mW in the stop mode (*Stop Mode: all clocks are stopped, regulator running, regulator in low power mode, optionally the integrated Flash can be powered down in this mode at the cost of a longer wake-up cycle*[127]). During the most energy-demanding process of sending a packet, the [F4VI2](#) needs 241 mW at 168 MHz while the [AVS-Extrem SN](#) needs 456 mW while working at 72 MHz instead. As we can see the [F4VI2](#) clearly needs less energy, with its processing power easily outperforming the [AVS-Extrem SN](#).

We deployed all Distributed Event Detection system experiments except for the therapeutic exercises using the [AVS-Extrem SN](#), which shows that these problems can be solved by using the [AVS-Extrem SN](#). By applying the new [F4VI2 SN](#) instead, the increased processing power will reduce the energy demands firstly by optimizing the energy footprint of the SNs in general and secondly by reduced calculation times.

6.4.2 Latency and PDR

Both introduced SNs use the same Chipcon transceiver for communication tasks. In order to validate the communication process quality under harsh conditions, we performed an experiment with two SNs integrated in a fighting stick (sport device), introduced in Section 7.3.1. The peculiarity of this setup is that both SNs are aligned in the horizontal direction to each other with an opposing orientation. For communication, this is not an optimal situation as the omni-directional antennas' power radiated is very much reduced above and below the antenna. The most power radiation is to be expected around the antenna in form of a flat curl. This orientation is the worst antenna orientation we have in our experiments, as on top of this, the sport device itself is in between the antennas as an additional obstacle; [LOS](#) between antennas is recommended to avoid packet

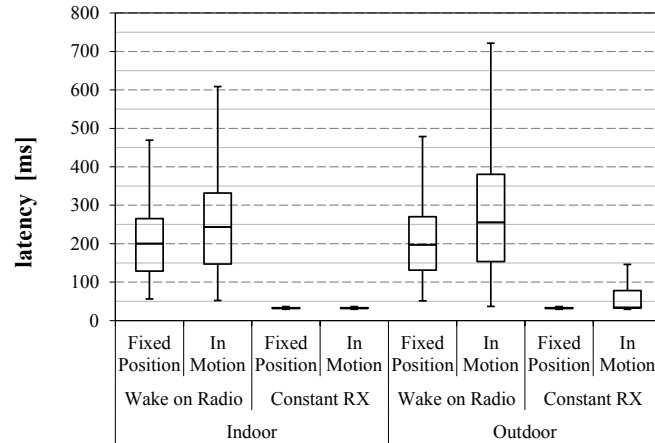


Figure 6.7.: Latency experiment results of processing time created by five differently combined parameters: location, radio mode, activity, adapted from [87].

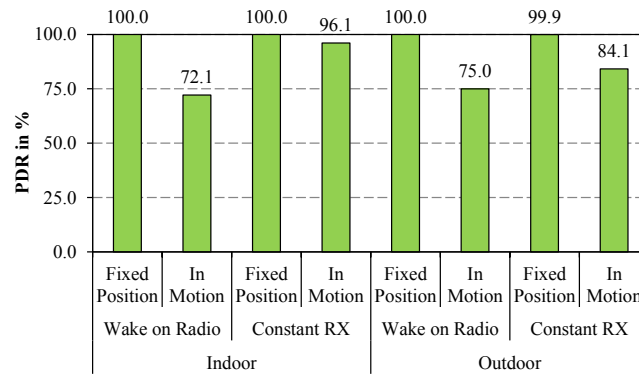


Figure 6.8.: PDR calculation from experiment results created by five differently combined parameters: location, radio mode and activity, adapted from [87].

delays or losses [107]. We investigated this situation in more depth in [87] and the results are recapitulated in the following.

We evaluate the communication by measuring the latency of the packet send process as well as the PDR using five differently combined parameters: location, radio mode, and activity. We differentiate experiments in indoor and outdoor locations while performing these experiments with the energy saving radio mode WOR (introduced in [140]) active or the latency reducing but energy demanding Constant RX active instead. In addition, all experiments are performed in combination with two activities: Holding the sport device in a fixed position (horizontal line to the floor) without moving (see Figure 7.16(e) 1) and spinning the device with a turn rate of roughly 360° per second (see Figure 7.16(e) 2).

To calculate the latency, we measured the Round-Trip Time (RTT) of 800 packets, with one packet transmitted per second. This allows us to ignore the clock drift between the SNs. We measured the time before activation of the send()-Routine and after reception of the packet at the sending station. We divided the resulting RTT by two in order to gain the estimated latency for a packet transmission. Thus, we have measured the processing time needed to send and receive packet within our system while having the acc-sensor active. After each packet transmission, we saved the timestamps on the SD-Card and waited two seconds before continuing with the next transmission. During the communication the test person is constantly spinning the sport device

for the *In Motion* experiment. For the *Fixed Position* experiment, the test person is holding the sport device in a fixed position. A primitive Transmission Control Protocol (TCP) derivative supports the communication with a simple re-transmission function in case of a missing ACK packet to minimize the packet loss.

The latency depicted in Figure 6.7 clearly shows that the WOR has an increased latency compared to the constant RX communication of about 165 ms to 230 ms on average. This is reasonable as the WOR activates the transceiver only every 542 ms as introduced in [102]. We can assume that the communication window should only add at most half of the WOR window size on average, which in our case means an expected increase of approximately 270 ms.

The second obvious influence can be seen if we compare the latency of packets transmitted while holding the device in a fixed position. This causes a generally better latency than with a moving device of about approximately 44 ms to 59 ms in WOR mode and movements have a slight effect on constant RX mode, resulting in a median of 89 ms for experiments performed outdoors.

The reason for an increased latency and reduced PDR during motion is to be explained by the Doppler effect [141] that arises when we move the sport device with the SNs while sending. The motion changes the wavelength of the sent packet and the bit-length of the transmitted data in addition. This known effect is not recognized by the receiver. Hence, packets are not always successfully transmitted in case of a changed transmission characteristic. In addition, the orientation of the omni-directional antennas has a negative impact to the communication. It should be noted that for future development of SN devices that have to detect events during a motion, an antenna is recommended that is robust against changes in antenna polarization, like fractal antennas in chip design as typically used in cell-phones.

The performance of indoor and outdoor experiments has a slight negative influence on the results if the device is in motion. As mentioned, the antenna orientation influences this scenario negatively. The indoor scenario may benefit from packet reflections from walls, which are not present during the outdoor scenario to the extent and density found in buildings.

Packet loss could not be prevented when the device was in motion as presented in Figure 6.8, where indoor experiments with activated WOR have the lowest PDR with only 72.1%, followed by the outdoor experiments with activated WOR showing a PDR of 75.0%. Constant RX communication again shows a better performance but also has some deficits: Indoor experiments with Constant RX reach a good PDR of 96.1%, while the corresponding outdoor experiment reaches only 84.1%. All other experiments reach very good results close to or exactly at 100%.

As a sport device or health application user does not want to wait for extended periods of time, we defined all packets with a latency of more than 15 seconds as lost, which make up 1% of sent packets. We recommend applying Constant RX in case of SNs in intense motion in order to optimize PDR and latency. As higher latency does not influence the process of data collection and distribution, the resulting data aggregation proceeds successfully in our subsequent evaluation of applications. Nevertheless, it is important to take the users' needs into account and to find a good trade-off between high reactivity of the application, a low energy profile, and reliable communication especially in case of security related application. Hence, we recommend that future bridge and fence surveillance applications use WOR in order to benefit from the low energy profile.

CHAPTER 7

Applications and Applicability

		Environmental Structure	
		Flexible	Rigid
Events affect SNs	Similar	Fence Surveillance Section 7.1	Bridge Surveillance Section 7.4
	Diverse	Therapeutic Exercises Section 7.3	Sport Device Section 7.2

Figure 7.1.: Taxonomy of investigated applications based on the environmental structure and manner in which events affect SNs.

To investigate the applicability of the Distributed Event Detection system we deployed several different applications, which are introduced in the following sections. The deployment and realization forced us to solve the Distributed Event Detection problem up to a very high level of applicability; beyond that, we learned lessons that should be solved in the next iteration of a future Distributed Event Detection system.

In order to evaluate the Distributed Event Detection System, we decided to investigate the distributed aspect of two different types of events and structures. Events can have very *similar or uniform* effects on the environment, while the environment can be *flexible or rigid* if we consider the observed environmental structure, as depicted in Figure 7.1. We investigate all four derivable possibilities. We start to introduce the similarly affected environmental structures: Such environments may be flexible structures such as fences with connected but flexible fence elements. Alternatively, such environments may be rigid structures such as bridges, which typically consist of less flexible substructures with more or less the same motion or frequency in the whole structure.

We can distinguish these similar types of events from event types that cause very diverse effects to the structures equipped with SNs deployed in the environment. These event types can be e.g. therapeutic exercises performed by humans with SNs flexibly worn on different body parts. Diverse event effects can also be caused by sport devices that integrate SNs in rigidly structured devices such as golf clubs, tennis rackets, or a fighting stick for martial arts as different parts of these devices are moved in different ways.

To cover the range of applications that we introduced above, we decided to investigate four applications, each of which can be taxonomically assigned to one category of the matrix in Figure 7.1. We shortly introduce the applications and why they fit into their categories:

Similarly affected SNs in a flexible structure: Fence Surveillance We investigate the application of a fence surveillance system to secure an enclosed area from intruders and show the applicability of this approach with a 49 SN setup. This application is an example for similar event effects in a flexible connected structure as the fence elements are connected by articulated connections. As our previously introduced theoretic evaluation is based on training data of this fence surveillance application, we focus on this approach with the most in-depth investigations.

Similarly affected SNs in a rigid structure: Bridge Surveillance We investigate the application of a bridge surveillance system to detect critical changes that cause damage or indicate a critical situation for a bridge that is not limited to one compact bridge segment with a 4 SN setup. This application is an example for similarly affected SN in a rigid structure.

Diversely affected SNs in a rigid structure: Sport Device We investigate the application of a sport device by introducing a fighting stick that detects multiple techniques. The applicability of this approach is shown with a 2 SN setup. This application is an example for diverse event effects in a rigid structure as the stick and therefore the SNs are coupled and not articulated to each other.

Diversely affected SNs in a flexible structure: Therapeutic Exercises We investigate the application of therapeutic exercises that can be distinguished and show the applicability of this approach with a 3 SN setup. This application is an example for diverse event effects in a flexible structure as the SNs are attached to different body parts which are connected by versatile joints.

In the subsequent section all applications are introduced in detail along with their setup and evaluated by analyzing the event detection results and by sharing our learned lessons.

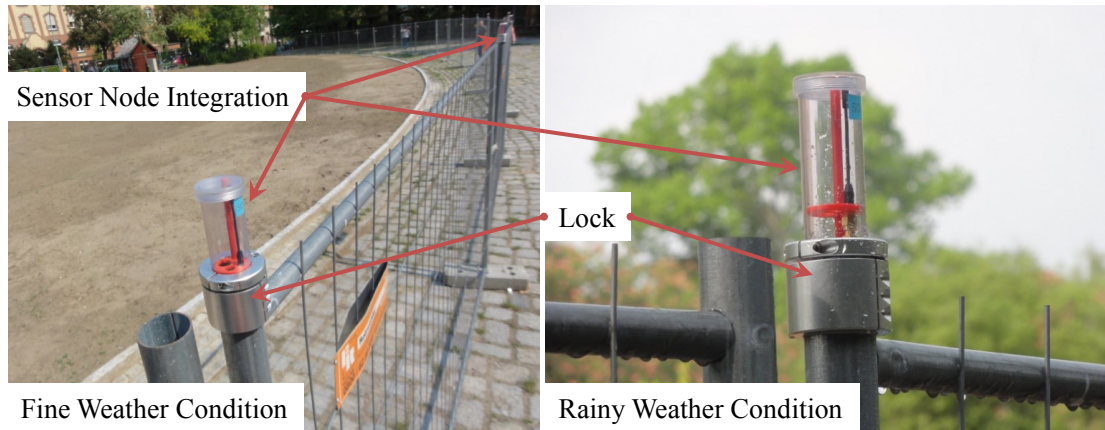


Figure 7.2.: *SN* integration into a construction site fence during sunshine and rain weather, applied from [5].

7.1 Fence Surveillance

The initial idea for a fence surveillance project for *WSNs* in general is described by Wittenburg et al. in [142], in which the authors designed a security relevant fence monitoring for *WSNs* with an early rule-based and threshold dependent detection algorithm. Our latest Distributed Event Detection System has been published in [5] and covers the introduced framework-based evaluation step and deployment procedure.

Fence surveillance systems are of interest for various applications, such as protection of open areas like construction sites, airports, concerts, or festivals, but also for border protection and security fence constructions for prisons or embassies. The main goal of a fence is of course to protect a specific area from unauthorized access. A wireless fence surveillance system can leverage this physical protection to a higher flexibility for guards or security personnel and to a higher security level for the people in authority. A physical fence plays the role of a barrier that can potentially be surpassed. In contrast, a wireless fence surveillance system should be able to detect attempts to surpass the fence when they happen while at the same time being immune against typical wire cutting attacks. As a drawback, wireless systems have to deal with limited energy resources, as introduced in Section 1.2 and Section 1.1. As long as energy harvesting for such systems is not available as a standard energy supply, they should only be implemented as an addition to existing systems in order to increase that system's security.[5, 117, 125]

7.1.1 Experimental Setup

Our entire discussion up to this point now leads into the real world deployment of the distributed event detection system on physical *SNs* and the introduced platform. The platform as introduced in Section 6.1 was created in the BMBF-Project *AVS-Extrem* [9, 143], where multiple *SNs* were integrated into a fence. In detail each fence module was equipped with exactly one *SN* (see Figure 7.2) within a casing that allows us to perform weather independent deployments. In order to guarantee a long term and realistic real world deployment, which entails performing experiments under reproducible conditions, the sensor platform was deployed in a rugged and weatherproof casing introduced in [102] and Section 6.3.1. Our setup for the training and the final evaluation was based on 49 *SNs* deployed and integrated into a construction site with an according number of

49 fence elements near our institute. The construction site fence had a perimeter of about 175 m. Owing to the size of the fence elements, the SNs were deployed roughly 3.57 m apart from each other and mounted on top of the elements at a height of 2 m in the left rod of a fence if seen from the outside.

We used the integrated 3D-accelerometer of the AVS-Extrem SN in order to gather data on the fence motions, but the system is not limited to one type of sensor. The initially introduced feature based approach that is well known in pattern recognition systems is able to handle multi-modal SNs as well. The motion of the fence is used to describe the events which may be initiated by an intruder trying to climb over the fence which the system has to classify, as well as several other events. This classification allows differentiating between multiple events.

7.1.2 Training

In order to feed the introduced Evaluation Framework of the presented system, the fence was exposed to multiple training runs for each class. The resulting reference vectors for each class embody a prototype of the average of all performed training runs of one event type composed of the different perspectives of each SN affected by the event. The subsequent event evaluation uses the stored reference vector to classify unknown events as described in full detail in Chapter 4.

For the supervised training, the basic data processing, and the pattern recognition, we used the AVS-Extrem SN, re-utilized from Dziengel et al. [102] and introduced in Section 6.2.1.

In order to perform a supervised training as well as the event detection, we developed a special SN – called AVS-Extrem node – introduced in Dziengel et al. [102] and with this thesis in Section 6.2.1. The goal was to design a special shape for a SN that could be integrated into a fence rod as depicted in Figure 7.2.

The training was performed on four consecutive days at the fence at the SN-ID 31 (see Figure 7.3), and we removed the nodes at the end of each day in order to download the gathered and stored training data from the SD-Card of the SNs. After we successfully finished the training sessions, construction site employees changed the structure and positions of the individual fence elements. In addition, the employees changed the order, angle, and fixtures of several fence elements. This was necessary in order to cover the enclosure of the new construction site area, owing to the ongoing construction.

The fence position changed by at least 1 m from its original position and event training positions; as a result, we could not ensure that our training position and configuration was the same during the evaluation and during training. This change in the fence structure forced us to test the Distributed Event Detection under realistic conditions: a slightly changed environment compared to the training. We therefore decided to pick a different but similar structured fence location for the evaluation as depicted in Figure 7.3. Figure 7.3 also shows that the SNs of the fence elements are always integrated in the left fence rod of the fence element if seen from the outside.

The particular changing layout of the fence owing to the ongoing construction during our research allowed us to perform a training that covered realistic area conditions of an evaluation experiment that approximates an untrained event location. In Section 5.6 we introduced ten different events which we trained with three different test persons in Figure 5.24 and Figure 5.25. The three test persons performed ten repetitions of each of the ten event classes, which resulted in a training data set of 300 events. We used different testers in order to ensure different performances of the events and thus provide diversity in the training set, which is to be expected in increased intensity in

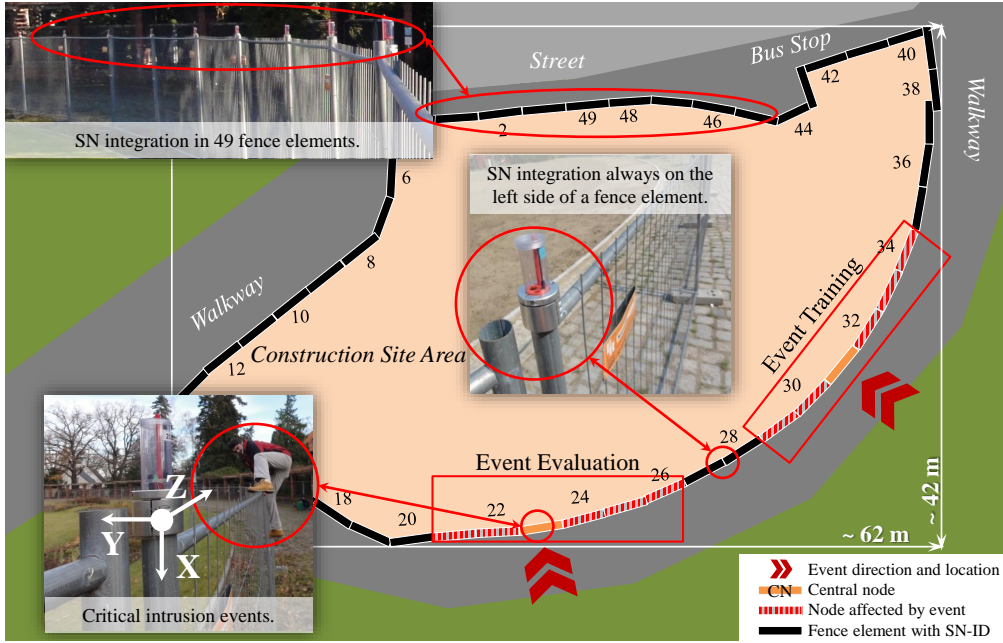


Figure 7.3.: Physical sensor network structure is driven by the fence structure. Event distribution, training and evaluation location depiction, applied from [5].

the future application. A future event detection needs to detect a much larger variety of persons than we could ever train, hence it is important to increase the diversity of the training as much as possible.

Although we tried to the best of our abilities to ensure a static and reproducible SN deployment, we want to emphasize that the results of the detection process are influenced by a fragile experimental setup, which is driven by a construction site fence with not uniformly connected fence elements that are occasionally bent, damaged, or deformed. Furthermore, the experiments are influenced by varying event intensity and varying practical event conduction. The reason for these inconsistencies lies in the complexity of event movements, which can hardly be performed by humans in exactly the same way each time. Therefore, movement repetitions are subject to a natural diversification.

7.1.3 Results & Evaluation

During our evaluation experiments, we performed ten events for each of the classes performed by two test persons, so 200 events altogether. In the subsequent sections we compare the results of the theoretical event detection as introduced in Section 5.6.3 with the results of the deployed event detection. The details of the distributed feature vector for this application are introduced in Section 5.7.1 and depicted in Figure 5.30.

7.1.3.1 Events

We evaluate the detection performance of the Distributed Event Detection approach by analyzing it with the metrics introduced in Section 3.3.1. All classification results of the previously introduced fence events (Section 5.6) are shown in Figure 7.4 in two different evaluations. The upper part of the depiction shows the theoretically evaluated results from the Evaluation Framework for each event.

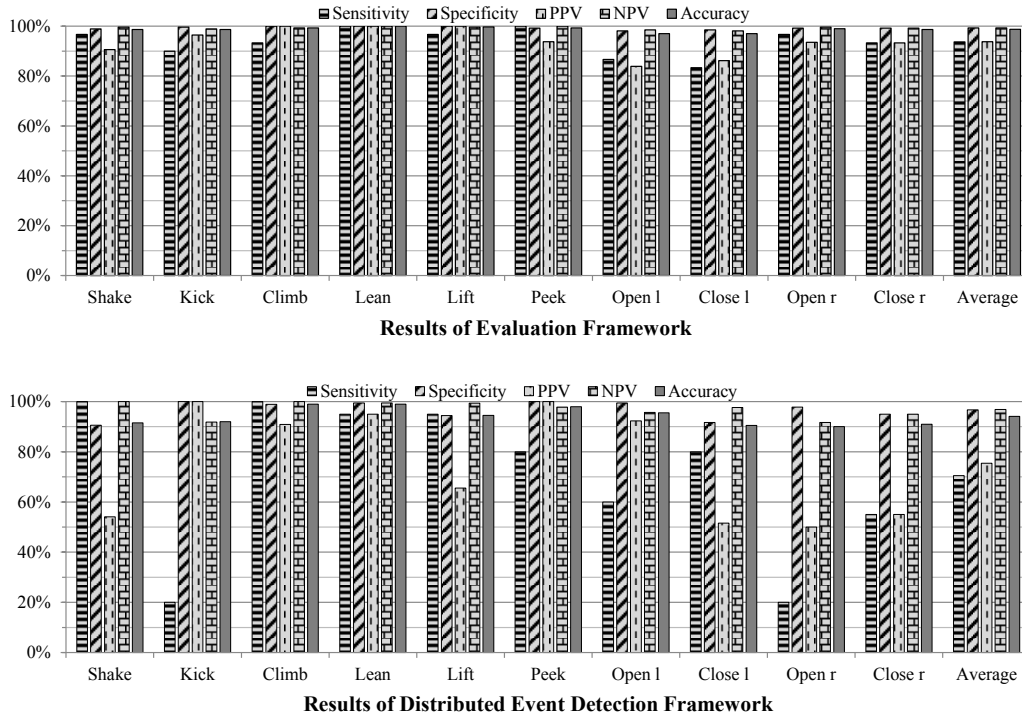


Figure 7.4.: Comparison of the classification results of all 10 events provided by the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower), adapted from [5].

Evaluation Framework		Detected									
		Shake	Kick	Climb	Lean	Lift	Peek	Open l	Close l	Open r	Close r
To Detect	Shake	29	1								
	Kick	3	27								
	Climb			28			2				
	Lean				30						
	Lift					29					1
	Peek						30				
	Open l							26	4		
	Close l							5	25		
	Open r									29	1
	Close r									2	28

Distributed Event Detection Framework		Detected									
		Shake	Kick	Climb	Lean	Lift	Peek	Open l	Close l	Open r	Close r
To Detect	Shake	20									
	Kick	10	8						1		1
	Climb			19					1		
	Lean		1		19						
	Lift				1	19					
	Peek					3	11			6	
	Open l							12	8		
	Close l					4			16		
	Open r					4			3	6	7
	Close r					1			1	7	11

Figure 7.5.: All 10 fence events compared in confusion matrices. Left: Theoretic results of the Evaluation Framework. Right: Deployment results of the Distributed Event Detection Framework.

In contrast, the bottom depiction shows the classification results of our final deployment using the reference feature vectors created by the Evaluation Framework. The results of the deployment are self-generated by the resulting detection of the Distributed Event Detection Framework while performing the event at the fence.

The events shaking, climbing, leaning, and peeking over the fence can be detected with a very high accuracy and high general performance in both evaluation types, both theoretical and during the deployment. The lowest sensitivity and accuracy is shown by the kick (20.0%), open left (60.0%), close left (80.0%), open right (20.0%), and close right (55.0%) event during the deployment. All these events show a comparable tendency if compared to the Evaluation Framework results, although the results of the Evaluation Framework clearly outperform the deployment results. We can recognize a tendency in the Evaluation Framework for open and close event to have a lower classification performance, which is confirmed by the real world experiments. The shake and kick event shows some performance loss, which is also confirmed by the real world experiment.

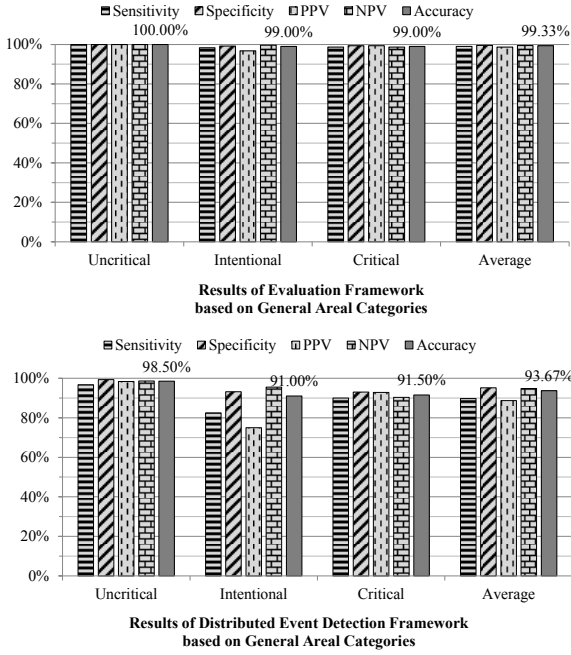


Figure 7.6.: Comparison of the classification results of the 3 derived General Areal Categories in the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower).

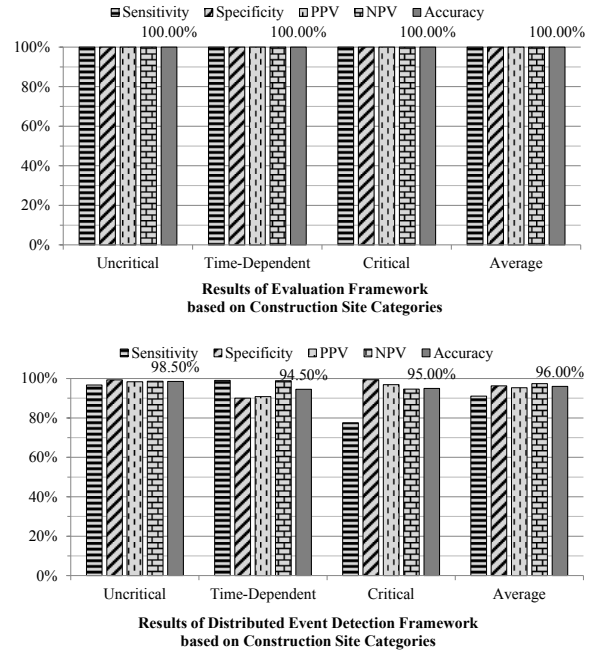


Figure 7.7.: Comparison of the classification results of the 3 derived Construction Site Categories in the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower).

The kicking event interferes heavily with the shake event during the deployment as 10 of 20 kick events were detected as shake events, which is reflected by the very low kick sensitivity and reduced shake specificity (90.6%) as well as PPV (54.1%). Nevertheless, all shake events were detected correctly, which is reflected by the corresponding sensitivity value of 100%.

The reasons for the confusion between kick and shake events can be explained by their similar event characteristics. The events are hard to distinguish and mostly depend on the intensity, which turns out to be a weak feature in case of a soft kick. The same problem arises with the lift event, which is classified as open right, close left and peek events. Lifting the fence is part of the events open and close: in case of a very gentle or short open or close event, the event may be classified as a lift event, which is reflected by PPV (65.5%) for lift.

The confusion matrices in Figure 7.5 show the reasons mentioned above in more detail. It has to be mentioned that of course, the resulting numbers can't be directly compared, as the Evaluation Framework has a training base of 300 data sets and the real world experiments (Distributed Event Detection Framework) have a base of 200 data sets. Nevertheless, we refrained from introducing relative values as it makes more sense to study the concrete numbers of events performed.

7.1.3.2 Categories

In case of an emergency scenario or in situations where a decision needs to be made, it is important to deliver an alarm or a concrete event description; furthermore, it is often not necessary to deliver detailed information about whether a fence is opened or closed. The responsible personnel should ideally only be informed about an incident in case of a critical or time-dependent critical event,

General Areal Categories				
Evaluation Framework		Detected		
		Uncritical	Intentional	Critical
To Detect	Uncritical	90	0	0
	Intentional	0	59	1
	Critical	0	2	148
Distributed Event Detection				
Evaluation Framework		Detected		
		Uncritical	Intentional	Critical
To Detect	Uncritical	58	1	1
	Intentional	1	33	6
	Critical	0	10	90

Figure 7.8.: Comparison of the confusion matrices of the 3 derived General Areal Categories in the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower).

Construction Site Categories				
Evaluation Framework		Detected		
		Uncritical	Time-Dependent	Critical
To Detect	Uncritical	90	0	0
	Time-Dependent	0	150	0
	Critical	0	0	60
Distributed Event Detection				
Evaluation Framework		Detected		
		Uncritical	Time-Dependent	Critical
To Detect	Uncritical	58	1	1
	Time-Dependent	1	99	0
	Critical	0	9	31

Figure 7.9.: Comparison of the confusion matrices of the 3 derived Construction Site Categories in the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower).

which leads to the necessity of a meaningful categorization of events. In order to evaluate the system using a categorized event configuration, we use the two types of categories introduced in Section 5.6 and show the results of the first categorization *General Areal Categories* [5] in Figure 7.8 and the second categorization *Construction Site Categories* in Figure 7.9. The overall goal of saving energy reached by minimizing communication between the SNs can be supported by transmitting only critical events and not transferring neutral or intentional events. For intrusion detection, it would be sufficient to know whether arising events are critical. We consolidated the event classes into three event categories for both category types as introduced in detail in Section 5.6.2.

The results in Figure 7.8 (top) show that all event categories for the general application approach are reliably detected during the Evaluation Framework evaluations as all metrics are above 96%, while the accuracy is above 99%. If we compare these results with the deployment results from Figure 7.8 (bottom), we have to recognize a reduction in classification accuracy for uncritical events to 98.5%, for intentional events to 91.0%, and for critical events to 91.5%. Both the theoretical and deployment results show good performance when taking into account that deployment results are typically less accurate compared to theoretic considerations. Nevertheless, tendencies of high and low detection accuracy expectations can be read from the graph by using the theoretic evaluation only.

The intentional events in 7.8 (bottom) reach slightly lower metrics than critical events do. In detail, the sensitivity of 82.5% is comparatively low, which means that in this case, six intentional events have been detected as critical. The reduced PPV to 75.0% in combination with a slight reduction of the Specificity to 93.1% indicate that a meaningful number of other events has mistakenly been classified as intentional; in this case, 10 critical events have been classified as intentional. Compared to the 90 correctly classified critical events, 10 critical events could not be detected, while 7 events caused a false alert.

If we compare the results of the general categories in Figure 7.8 with the results of the construction site categories in Figure 7.9, it is obvious that the results of the construction site categories perform much better in both evaluations. The theoretic investigation reach perfect results of 100% overall, as well as the deployment results with all metrics above 90% except for the specificity of the critical events with 77.5% as 9 of 40 critical events were classified as time-dependent critical, see Figure 7.9 (bottom). This means that on average, 9 critical events will not be detected and announced if the work period is in effect. During the after work period, these events will be an-

nounced as critical again, which reduces the false rejections to about 74% if we assume a 9 hour work period. The Distributed Event Detection for the General Areal Categories classifies 10 critical events as intentional and 6 intentional events as critical. Both event types can be understood as worth alerting the personnel for, while uncritical events won't. Hence, the results are good and will support the intrusion detection application, but they aren't 100% reliable. We can state that because of the good classification results and the overall low energy footprint of our proposed approach as evaluated in Section 5.5.6, the fence surveillance application can be valuable if used in combination with security systems to support e.g. a camera-based system with seismic sensors (e.g. geophones) integrated in the ground in order to detect footsteps.

Although the results have a certain degree of potential for improvements, they are the best results compared to our past deployments, which are presented in the following Section 7.1.4. Nevertheless, both presented categorization results outperform the classification results of all our previous results.

7.1.4 Fence System Comparison

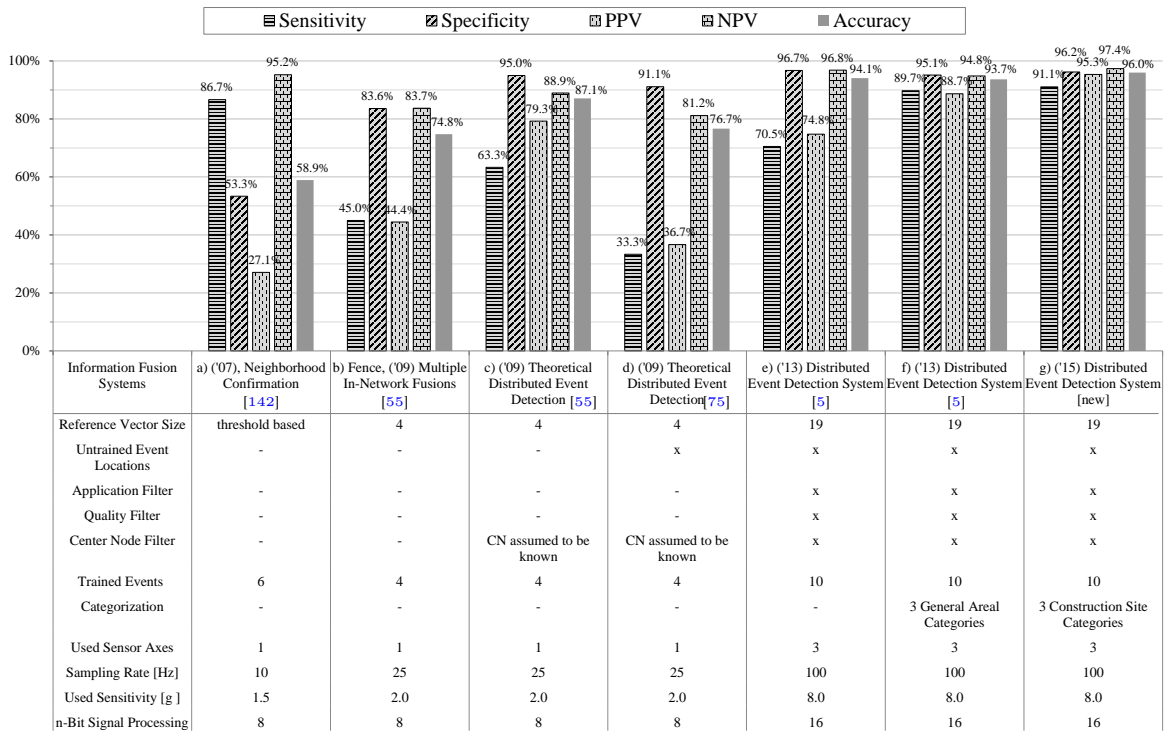


Figure 7.10.: Classification capabilities comparison of seven deployed different fence event detection systems with different setup parameters, adapted from [5].

In the following section we want to compare the current averaged results of the classification performance with the historical past of our research in its entirety, which has been introduced in parts in [5]. Each system is marked with small letters (a) to (g) which represent the historical creation and evaluation order, whereas the publication order may differ. The numbering is consistent for better understanding in both Figures 7.10 and 7.11.

7.1.4.1 System (a) - Neighborhood Confirmation

The idea to observe events at a fence was proposed by Wittenburg et al. in [142] with a neighborhood confirmation based concept, see Figure 7.10 (a).

All systems except system (a) depicted in Figure 7.10 implement the same core idea of detecting events on the basis of an a priori training with a subsequent event detection step that extracts features, distributes features to the neighborhood, and finally tries to generate a reliable classification result. System (a) was the first approach and not based on distributed information fusion. This system also can be derived from the used input-output scheme fusion from Section 5.2 as it follows a DEI-DEO scheme. It uses a custom-built heuristic classifier implemented in FACTS middleware [144]. The event description was derived from a visual inspection of the raw data. Although the fence monitoring system (a) does not directly fit into the information fusion concepts of the other depicted systems, we want to emphasize that it is interesting for historical reasons to compare this approach with the actual research results.

The general setup for this approach was based on 6 events (kick, lean, short shake, long shake, peek, climb). The signal processing was performed at 8 – Bit with the MMA7260Q 3-axis accelerometer soldered on a Modular Sensor Board (MSB) [145]. The sampling rate is dynamically set from 1 to 10 Hz if an event arises while the used sensitivity of the acceleration sensor is set to 1.5 g. Although the approach is able to handle data of all axes, the experimental results are based on only one axis.

This first approach had a good sensitivity of 86.7%, but due to the low specificity of about 53.3%, the event detection caused a huge amount of false alarms, which is reflected by the low PPV of 27.1%. In addition, the approach paved the way for the later research with the very first and fundamental lessons learned.

7.1.4.2 System (b) - Multiple In-Network Fusions

System (b) represents the Multiple In-Network Fusions with an additional distance rejection algorithm with the goal to reduce the events that are sent to the control center. Otherwise, all affected SNs would have to send their classification results based on the multiple in-network fusion to the base station. The distance rejection algorithm was implemented with the intention to identify the CN, but the system's performance suffers from the fact that at least 65% of the SNs triggered by an event will still send the classified events to the control center as the distance rejection could not successfully reject these classifications. Hence, multiple results have to be fused at the BS. A classification fusion at the BS was introduced in [55], which led to results for sensitivity and PPV of below 50% owing to about 50% false positives and false negatives.

The general setup used differs from system (a) by an increased sensitivity to 2% and an increased sampling rate to 25 Hz. Furthermore, the number of events has been reduced to four (shake, kick, lean, climb).

The results show a low sensitivity of 45.0%, but an acceptable specificity of 83.6%, which clearly indicates a low detection rate. The PPV stays below 50% and indicates a rather impracticable applicability. This can be explained by multiple in-network fusions being used to find a majority with this detection results. The majority of the distributively detected events do not reflect the trained events as they are evaluating the events from untrained locations, which indicates that such a majority-based approach does not support the distributed event detection.

7.1.4.3 System (c) - Theoretical Distributed Event Detection

The fence evaluation (b) results were not convincing, hence we wanted to find out whether the distribution scheme would perform in case of an assumed exiting solution for the mention rejection problem.

With a theoretical Distributed Event Detection (c) where the CN was assumed to be known a priori, as in [55, 117, 125], the same scenario introduced for system (b) has been investigated. The sensitivity reached 63.3%, which, together with the specificity of 95.0%, proved the fundamental functionality of the distribution scheme as depicted in Figure 7.10 (c). During application, this system rejects too many relevant events, but shows an initial good tendency for the distributed event detection.

The general setup matches the setup of system (b).

The results are good and, with an improved sensitivity of 63.37% and an excellent specificity of 95.0%, clearly indicate that the general idea of distributed event detection is a viable concept.

7.1.4.4 System (d) - Theoretical Distributed Event Detection with Untrained Event Locations

With an additional evaluation, we wanted to investigate how the distributed event detection from (c) at that time would perform at untrained event locations. The theoretical Distributed Event Detection of [75] assumed the CN to be known a priori. In addition, the system was exposed to events at untrained event locations at the fence. The results are depicted in Figure 7.10 (d).

The general setup matches the setup of system (b).

The good results of the theoretic system evaluation (c) could not be confirmed under more realistic conditions when performing events at untrained locations. The results clearly indicate that this realization is not robust against real world conditions as the lowest sensitivity with 33.3% indicates that the results are worse than random. Improvements of the realization concept in general can be reached with additional knowledge about the event location, but that knowledge was not available for the detection system.

7.1.4.5 System (e) - Distributed Event Detection System

The introduction of the Distributed Event Detection Framework introduced in [5] led to the system deployments introduced in this thesis. We add to the comparison the following results of the proposed Distributed Event Detection, while all subsequent systems (e) to (g) evaluate the events with a fully applicable system evaluated at untrained locations that implements a filter package including a CN, Quality, and Application Filter to improve the overall detection performance. Figure 7.10 (e) shows the performance of the classification of 10 different fence events with the configuration mentioned above.

We added the introduced Evaluation Framework with a significantly higher feature space to support a more reliable classification model. In addition, we were able to filter the CN by utilizing the characteristics of signal propagation within the network. In rare cases, two or more nodes can pass the CN-Filter. As introduced, we add the quality filter which selects the most reliable classification result. The filter package reduced the communication of unnecessary classification results packets to a maximum of one classification result per event. Further, the application filter reduces the communication to the BS depending on the relevance of the event as introduced in Section 4.3.3.1.

The general setup for this approach was based on 10 events (shake, kick, climb, lean, lift, peek, open left, close left, open right, close right). The signal processing was performed at 16 – Bit with the Bosch SMB-380 3-axis accelerometer soldered onto the AVS-Extrem sensor board introduced in Section 6.2.1. The sampling rate is set to 100 Hz while the used sensitivity of the acceleration sensor was set to 8 g.

With a tailored hardware platform for our application as well as a completely new concept to realize the distributed event detection for the first time as a fully applicable approach, our real world deployment reached very good results that outperformed all previous implementations. The currently best, but still improvable, sensitivity of 70.5% paired with a very good specificity of 96.7% showed an event detection system with a significantly minimized false alarm rate.

7.1.4.6 System (f) - Distributed Event Detection System classifying General Areal Categories

Figure 7.10 (f) shows the results of the classification of three General Areal Categories as introduced in Section 5.6.2, in order to cover the real world requirement that only critical events should be transmitted. The 10 events are mapped to the three introduced General Areal Categories. The generally high results of all metrics show a high classification reliability. For real world purposes, this shows that the system would especially support general fence monitoring systems.

The general setup for system (f) matches the setup of (e).

The results of the categorized realizations show the good metric value of 88.7% for the PPV and very good metric values for the sensitivity 89.7%, specificity 95.1% and 94.8%. These values confirm that the proposed system is a robust and highly homogeneous detection system with very similar behavior in terms of detection rate for relevant and irrelevant events, which is important as this behavior allows defining a common reliability for the user without sacrificing too many of the qualities the system can offer. Based on the given results, we can state that the application is able to detect events with 88.7% without overestimating any parameter.

7.1.4.7 System (g) - Distributed Event Detection System classifying Construction Site Categories

Figure 7.10 (g) uses three different categories compared to (f). The Construction Site Categories with a special time dependent category for critical events help cover the work and after work periods in an according and dedicated construction site application. The results of this system outperform all previous systems.

The results show an improved detection for our recent approaches (d), (e) and (f) towards the past results. The general setup for system (g) matches the setup of (e).

The best result can be presented with a new type of categorization that outperforms all previously discussed system. The increased value for all metrics indicate a further improvement. The sensitivity reaches 91.1%, the specificity reaches 96.2%, the PPV reaches 95.3% and PPV reaches 97.4%. These results clearly show a very stable and robust system with a reliability of 91.1%. Based on the classification results, we can recommend this system for fence surveillance applications at construction sites. In addition, the prior energy demand investigation in Section 5.5 already proved the advantages over typical information fusion approaches, hence we can recommend this system for WSNs as the considered lifetime investigations highlight a promising applicability if compared to classical information fusion approaches.

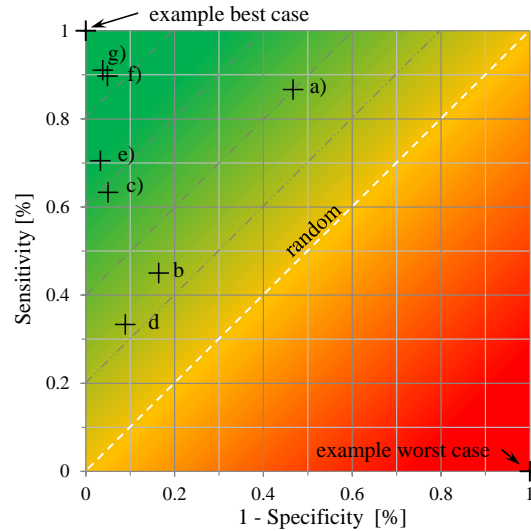


Figure 7.11.: Classification capabilities comparison within a Receiver Operating Characteristic space graph of seven deployed different fence event detection systems, adapted from [5].

7.1.4.8 ROC Analysis

Figure 7.11 summarizes the relation between sensitivity and specificity of our deployments within a Receiver Operating Characteristic (ROC) space graph. The previously introduced systems can now be compared to one another, although we see that system (b) and (d) are close to random results and can not be recommended for any application. To summarize, system (b) plays the role of the “lessons learned“ system, while (d) clearly shows the limits of Distributed Event Detection at that point in time.

System (a) shows a better performance than system (b) and (d) do but still has a huge problem with a very high false alarm rate, which would not satisfy serious applications. As (a) was never intended to play the role of an application ready system, it fulfills its goal to prove the general question of whether events can be detected at a fence with a WSN.

System (c) performed well and represents a more basic evaluation with incomplete functionality and a less efficient energy profile. The system was very helpful in developing the next big step towards the existing Distributed Event Detection System, as previously lessons learned could directly influence our development.

System (e) performed with good results and evaluated ten classes at untrained fence locations and clearly outperformed prior approaches. System (e) supports a fully functional system and could be improved with an additional gyroscope as introduced with the F4VI2 SN, as this can add an information layer based on precise rotational motion data. A practical realization is not recommended unless such application dependent optimizations have been implemented. Systems (f) and (g) show the performance with the application specific additional filters, and both systems show great performance and can be recommended for their investigated applications. The new Evaluation Framework of the Distributed Event Detection System of Systems (e), (f), and (g) created reference vectors that are much more resistant to variations in the events than system (d). The experiments with our proposed Distributed Event Detection System have been performed at untrained locations with very good resultant detection rate.



Figure 7.12.: Fence buildup challenges comprise a vast variety of over time changing parameters that can theoretically be covered by an increased training effort, adapted from [55].

7.1.5 Lessons Learned

The fence surveillance system is a very challenging application, as the nature of the fence allows events to only affect the SNs in similar rather than diverse ways. A fence can mostly oscillate around one axis and, depending on material and construction, this can be strongly reduced. Especially for huge construction sites, the fence may imply a huge number of changing parameters, hence the assertion that a fence surveillance system applied as the construction site variant is the most challenging application due to its ever-changing nature. The main parameters are depicted in Figure 7.12: a general poor buildup, dampening obstacles can interfere the fence's oscillation, building machines pose a danger for the hardware as well as for the PDR, fence elements can be exchanged against untrained elements, the fence connections can be inhomogeneous with e.g. a chain, while fence elements can additionally be disconnected and the fence structure may change over time.

As introduced in [55], the problem of structural diversity could theoretically be solved with an increased training effort that covers these situations. In reality, this seems unfeasible. However, some general structural situations such as a fence corner and widespread situations such as fence doors or damping due to raised sand should be trained. A more practical advice is to apply such a system for fences with a carefully constructed structure as this seems most beneficial for the proposed application and leads to very reliable detection results as shown above.

In contrast, typical border fences or more longterm fence deployments offer a much more suitable structure and behavioral parameters. A very suitable fence that still fits our investigated fence types for construction sites with a high quality construction, clean and fully connected fence elements as well as a homogeneous setup is used at the former Berlin-Tempelhof Airport (IATA: THF). Such



Figure 7.13.: Optimal conditions for a fence surveillance application at the example location of the former Berlin-Tempelhof Airport (IATA: THF).

fence buildups indicate the area of applications we would suggest as very appropriate for fence surveillance with the Distributed Event Detection System, see Figure 7.13

The proposed system can not guarantee a 100% reliable detection, but it offers a new physical security layer with a highly redundant sensor-based observation and without the single point of failure that a cable based system would imply for existing security systems. A system based on a cameras or virtual fences [146] benefits from the additional and independent detection system which adds multiple independently observing SNs. Even a single node reaches high detection rates of approximately 82%, while in cooperation, the SNs outperform all of our previously investigated systems with up to 97.4%.

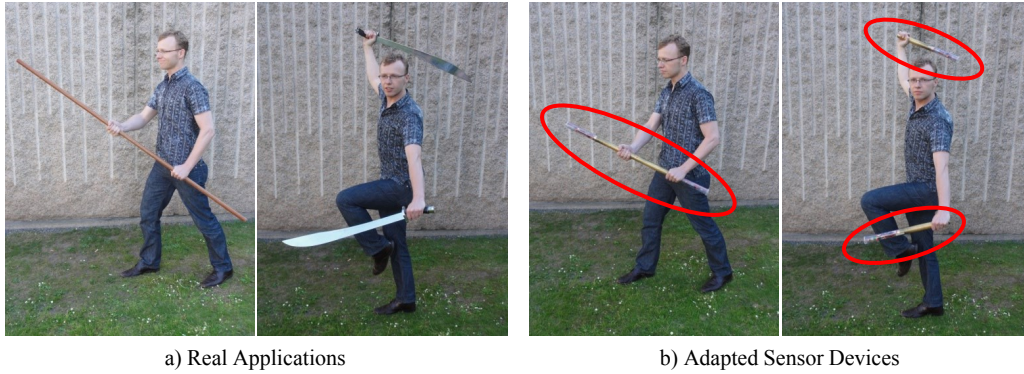


Figure 7.14.: a) shows the real world applications while (b) adapts the technical realization with the Wireless Motion-Based Training with a coupled as well as an uncoupled device. Our experiments are based in the coupled stick variant, applied from [87].

7.2 Sport Device

Athletes improve their skills in large parts owing to the trainer’s feedback, which might not be available during every training session, such as in the case of the trainer’s absence. Supervising training devices with integrated digital feedback are a good solution to improve the training efficiency for such cases.

Feedback is an important requirement for improving skills and is defined as any kind of sensory information related to a response or movement in the area of motor learning. Intrinsic feedback is given as a direct result of a performed movement and recognized via muscles, joints, and balance as well as vision, proprioception, and hearing [147]. The feedback we provide – the external feedback – is based on the knowledge of the technique. This feedback may be given verbally by a teacher during the training, or it can be a signal given in another fashion.

Martial art techniques in particular are very hard to master and need a constant supervision by the trainer in order not to internalize wrong motions. In order to train several techniques in martial arts, it is widely accepted as best practice to perform multiple techniques in a series called a *form*. The order of the techniques defines the form. The correct order and performance of each technique is part of a traditional training.

In [87] we introduced to the best of our knowledge for the first time a new ubiquitous computing device – called a Wireless Motion-Based Training device – for in-network event detection in order to support fighting stick training. The Wireless Motion-Based Training device enables the user to get a direct feedback from the device itself while performing training exercises during the absence of the trainer. With the possibility of increasing the number of SNs, the reliability of such devices can be improved by adding multiple perspectives to the information fusion. The fighting stick application is an example application in the area of sports, where humans have to learn and repeat dedicated movements in a correct order and manner according to specific guidelines.

Compared to other approaches, the proposed realization of the Distributed Event Detection technology gives an immediate visual feedback to the athlete through LED attached to the SNs.

Furthermore, we do not need a stationary BS to process the sensed motion data, as is the case with e.g. the Nintendo Wii [148], as all data are evaluated within the network and in a cooperative way. Heinz et. al [149] analyze motions of Wing Chun techniques with wired industrial SNs in order to extract features for a rough detection scheme that differentiates amateurs from experts

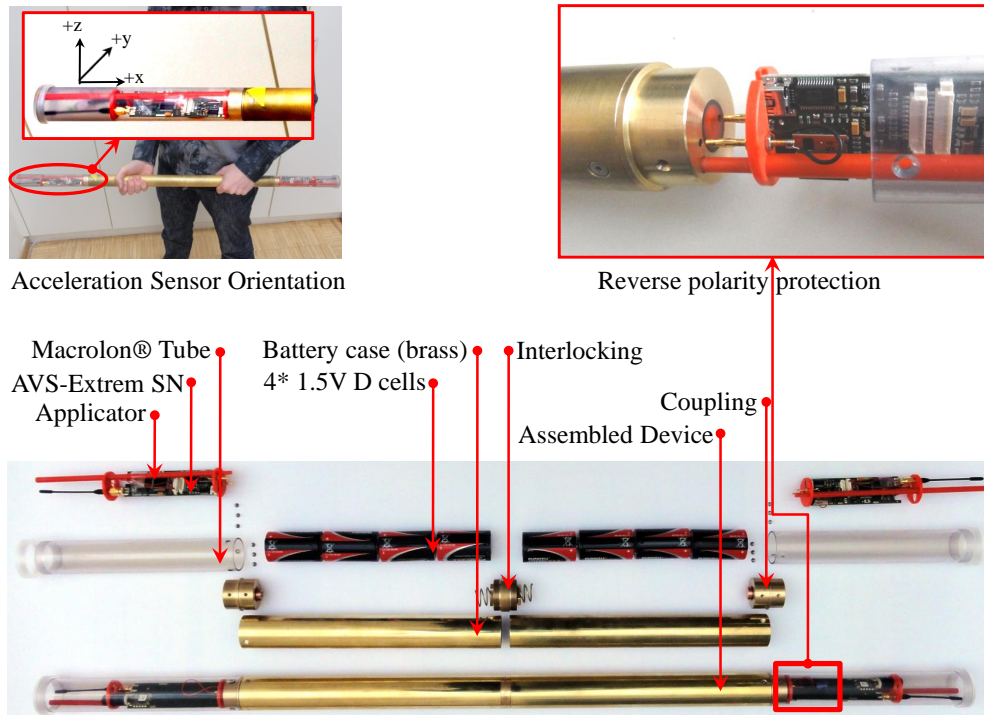


Figure 7.15.: The enhanced casing to create a training device based on the fence casing depicted in Figure 6.4.

when performing martial arts techniques. As cables (as used in [149]) are still a single point of failure, especially if movements bend and fold the cables, and because such cables may pose limits to agility, their use should be avoided.

The goal is to verify whether it is possible to train multiple martial arts techniques with the coupled version of the Wireless Motion-Based Training device; for this purpose, the training device should give a proper feedback that indicates whether the techniques were correctly performed.

7.2.1 Experimental Setup

We combined two of our previously developed AVS-Extrem SNs and cases from the fence surveillance application as depicted in Figure 7.15. The casing was designed to be robust and splash-proof in order to withstand construction site incidents such as dirt, rain, or shocks and is therefore well suitable for a contactless martial arts application. We combined two identical brass cases and connected their lower ends. The power supply (6 V) is placed inside the brass housing of the stick, using four standard D-cells. Two wireless SNs are plugged into the opposing sides and sheltered with Makrolon® tubes for solidity. To ensure the proper handling during the training and evaluation experiments, the athletes can check the correct orientation of the stick by markings on the stick for the correct handhold.

Following our previous investigations, we reuse the prototype classifier for the current problem. In contrast to the fence application, we do not need the CN-Filter, as both nodes are always in the same relative position to the user, which means that the fence problem is a stronger problem concerning the Distributed Event Detection System. The training events for the Evaluation Framework are performed by a teacher with six different stick-fighting techniques, with each technique being performed 20 times to gain an overall number of 120 training data sets. Furthermore, WOR

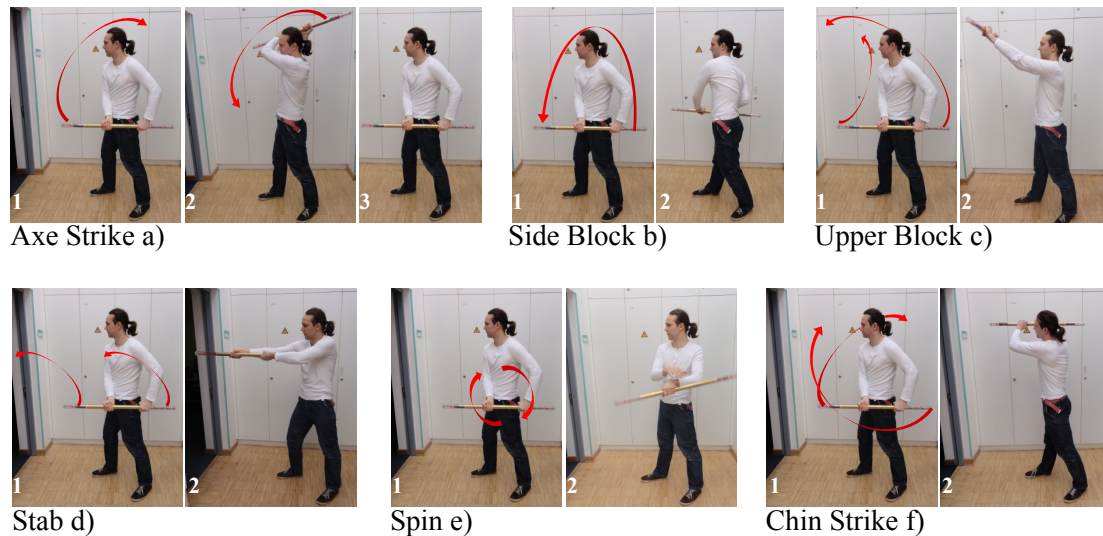


Figure 7.16.: Six different fighting techniques trained 20 times and evaluated 50 times, applied from [87].

is activated to evaluate a real world scenario and to extend the lifetime. To acquire the wide range of possible acceleration forces, we set the sensitivity of the accelerometer to 8 g and the sampling frequency to 100 Hz.

As depicted in Figure 7.14, we train the stick fight with one long stick. Each of the 6 techniques is trained with our Evaluation Framework in order to acquire the corresponding class descriptions necessary for the classification model.

As depicted in Figure 7.16, we trained and evaluated 6 techniques: two strikes, two blocks, one stab, and one spin technique. All techniques start at the same position, but end in different ones, see Fig. 7.16. As soon as the stick stops for 1.5 seconds, the technique is recognized as being finished. In detail, the axe strike (a) is a very typical strike for several weapons disciplines in martial arts and is performed with power and a strong finishing motion. The side block (b) is an impulse block that can also be performed in other directions but mainly blocks attacks in form of stabs. The upper block (c) blocks frontal stabs or axe counter-strikes with a stable stance and strong impulse above the fighter's head. The stab technique (d) is a quick and piercing technique that concentrates the power in one point towards the opponent. Spin (e) is a technique used to train the stick handling as well as to attack and block more rapidly from the active state of spinning. This is the most complex technique and is performed with three 360° turns, which requires some training in order to be mastered. The chin strike (f) is an optional two-step strike where the front part of the stick may hit the opponent first. The second hit of the back-part of the stick is the main attack with more power and energy added to the movement. Both hits aim at the opponent's chin.

During event detection, each technique is classified by the Distributed Event Detection System. The evaluation is performed by the Wireless Motion-Based Training device after a technique is fully performed and finished. Three kinds of LEDs notify the user about the evaluation result: the green LED lights up if the technique was correctly performed and the red LED lights up if the technique was not correctly performed or a wrong technique was performed. A blue LED is used to notify the user about a successfully completed series of techniques in the correct order.

		Acceleration Data			
		Raw Data			
SIGNALS s		X	Y	Z	\vec{v}
Feature-Types	Orientation	4 (n) _{(4,1)(5,2)} (n+1) _{(2,11)(4,12)}	2 (n+1) _{(1,13)(5,14)}	4 (n) _{(1,3)(2,4)(5,5)} (n+1) _(5,15)	
	Magnitude				10 (n) _{(1,6)(2,7)(3,8)(4,9)(5,10)} (n+1) _{(1,16)(2,17)(3,18)(4,19)(5,20)}

Legend:

# (relative node pos.) _(binID,vector pos.)	Features of Neighborhood Nodes (need to be distributed)
# (central node) _(binID,vector pos.)	Features of Central Node Only (no distribution necessary)
	Not Available Features

Figure 7.17.: Selected features for the Wireless Motion-Based Training device, only relevant features and signals are presented.

7.2.2 Results

During the modeling phase within the Evaluation Framework, only the magnitude and orientation features have been selected by the Feature Selection process. These features describe the motions for the subsequent classification process most precisely. With the existing acceleration sensor, it makes sense to pick the orientation feature as well as the strength feature for high varied motions. An improvement for the given system would be an inertia system featuring a gyroscope. Because of the complexity and length of the events, we increased the number of bins from 3 to 5 compared to the fence surveillance. By this, we are able to cover more details of a certain motion sequence.

20 features have been selected, as depicted in Figure 7.17. For node 1 (right hand), two orientation features for the X-Axis and three for the Z-Axis are selected. For node 2 (left hand), two orientation features for the X-Axis, two for the Y-Axis and one for the Z-Axis are selected. Five magnitude features are selected, while a magnitude feature is selected for all 5 bins.

For node 1 (right hand), two orientation features for the X-Axis and three for the Z-Axis are selected. For node 2 (left hand), two orientation features for the X-Axis, two for the Y-Axis and one for the Z-Axis are selected.

As both SN extract 10 features, it is irrelevant which node is assigned to the role of the CN, as the number of transmitted features is 10 in any case.

The evaluation was performed by two test persons that were not involved in the previous training process in order to guarantee an uninfluenced evaluation. The *advanced test person* did not train but had seen the techniques beforehand. The teacher leads the way with his own fighting-stick and shows the techniques. The learner tries to imitate the techniques with 50 repetitions each. The advanced test person was well-trained and advanced in martial arts. The *beginner test person* performed the evaluation in the same way; he was also well trained, but a beginner in martial arts and therefore less skillful in adapting techniques and performing them with the same accuracy as an advanced test person.

We show the training results by applying the introduced metrics of specificity, sensitivity, PPV, and NPV for the advanced test person in Figure 7.18, including the confusion matrix depicted in Figure 7.20, while the beginner's results are shown in Figure 7.19 and the confusion matrix depicted in Figure 7.21.

In general, sensitivity is below the specificity, which means in the current application that the test person did not successfully perform all techniques. Nevertheless, all techniques were detected

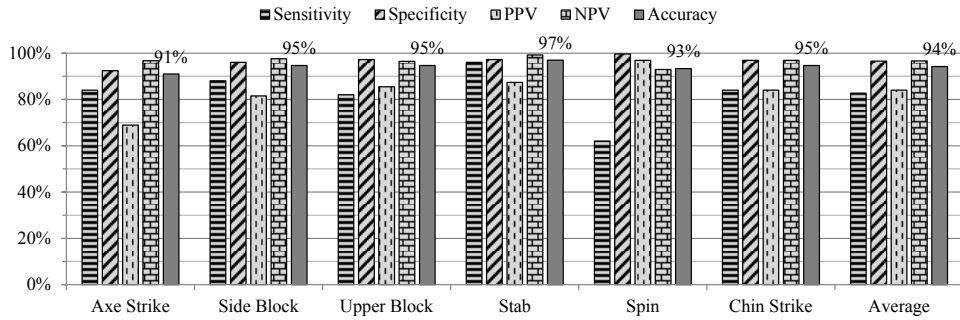


Figure 7.18.: Classification results of 6 stick fight techniques evaluated with 50 technique for each technique by an advanced test person, applied from [87].

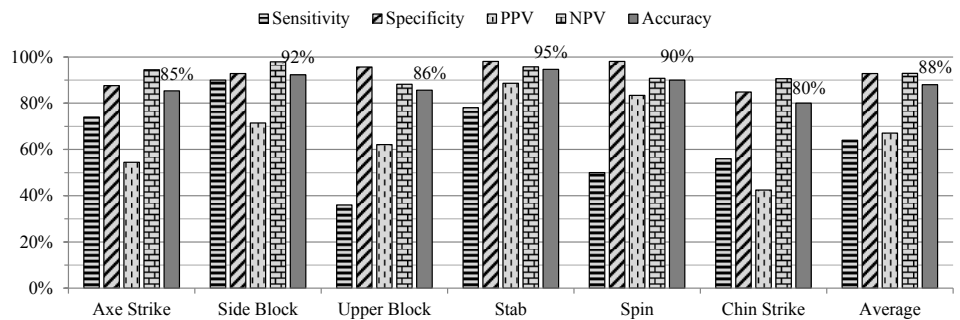


Figure 7.19.: Classification results of 6 stick fight techniques evaluated with 50 technique for each technique by a beginner test person.

		Distributed Event Detection Framework					
		Detected					
To Detect	Axe Strike	42	1	1	1	1	4
	Side Block		44	1	5		
	Upper Block		4	41	1		4
	Stab			2	48		
	Spin	19				31	
	Chin Strike		5	3			42

Figure 7.20.: Confusions matrix for classification problem of 6 stick fight techniques evaluated with 50 technique for each technique by an advanced test person, applied from [87].

		Distributed Event Detection Framework					
		Detected					
To Detect	Axe Strike	37	9		2	2	
	Side Block	3	45		1	1	
	Upper Block	2		18	2		28
	Stab				39	1	10
	Spin	15	9	1		25	
	Chin Strike	11		10		1	28

Figure 7.21.: Confusions matrix for classification problem of 6 stick fight techniques evaluated with 50 technique for each technique by a beginner test person.

with an accuracy of at least 90 % for the advanced test person while the beginner test person only reached an accuracy of at least 80 %. The difference of 10 percentage points reflects the difference between the skills of the test persons.

The confusion matrix clearly indicates a nearly perfectly-marked principal diagonal for the advanced test person. The advanced test person had a major problem with the spin technique as this

technique is defined by several rotations to be repeated which have to be performed within one constant orientation axis and without any tumbling. The achieved sensitivity was only 62%. The fighting stick should not tumble during the rotation, but this was exactly what happened for the advanced user in cases where the vent could not be detected correctly. The axe strike performed very well, but was classified once as each of the other classes, except for the chin strike, as which it was classified four times. Axe strike and spin interfere with each other as the axe strike was misclassified 19 times in the case of a spin, which is reflected in a lower PPV of 62% and sensitivity of 84%.

The confusion matrix clearly indicates a well-marked principal diagonal for the advanced test person but also highlights problems in general and especially with the techniques upper block, spin and chin strike within the principal diagonal.

The beginner had major problems repeating the more complex techniques correctly, especially the upper block, the spin, and the chin strike. In contrast to the advanced test person, the beginner user performed the spin techniques incorrectly by spinning only one or two times. In addition, he had problems with the tumbling while spinning the stick, which led to a sensitivity of only 50%. The axe strike was performed well, but was classified as the side block 9 times and 2 times as stab and spin. The axe strike was detected quite often for other classes, especially for the spin and chin strikes, which leads to a lowered PPV of 54%. The upper block is worth discussing as it was classified as the chin strike 28 times, which is more than 50% of the upper block events; furthermore, the chin strike was also classified as an upper block ten times. The reasons for this could be identified during the evaluation: The beginner test person was not able to repeatedly coordinate both fighting stick ends in parallel into the block position and performed an additional slight winding-up prior to the movement, which causes the classification system to detect the chin strike in many cases. This led to a very low sensitivity of 36% as well as a lowered PPV of 62%. The chin strike has similar problems as 10 events were classified as the upper block and 11 as the axe strike. In addition, 38 events were falsely classified as the chin strike. This leads to a very low sensitivity of 56% and the lowest specificity of 85% in the field.

7.2.3 Evaluation

The purpose of the Wireless Motion-Based Training device was achieved. Differently skilled test persons were able to learn from the device. The device classification quality of the advanced test person should ideally reach 100%, but some mistakes by the test person could not be avoided due to the complexity of the techniques. Hence, the evaluation depends on the test person, but it still has potential for improvements as the motion feedback is still very simple. Future work should focus on giving more concrete feedback in order to indicate meaningful correction advice. Aside from the technical successes, the test person felt motivated by the feedback throughout the training; this had not been a stated goal, but further justifies the development of such devices.

The spin technique was too demanding of both test persons. Both test persons were able to learn from the feedback given by the training device and to improve their own technical skills during training. The training motivation was increased by the direct feedback through the LED. The absence of a teacher can be partly compensated by the feedback, as rough mistakes can be easily detected by the training device. A detailed feedback for the technique with a qualified description of the motion faults requires a different technology that is able to investigate the path of motions and which includes a feedback for the path as investigated in Seiffert et al. [60] and Dziengel et

al. [86], but this technology is out of the scope of the current thesis.

Nevertheless, with the investigated training device, correct and incorrect orders of the technique can be detected. For future applications, it is feasible to implement a slightly improved feedback that uses the given knowledge of the detected and expected class. In case of a detected class that does not match the expected class, a feedback system could simply use integrated loudspeakers or a display within the device to remind the user which technique was expected. This imitates the behavior of a human teacher.

7.2.4 Lessons Learned

The application of a fighting stick performed very well and of course depends of the user's proficiency, their ability to imitate the techniques and the way the device is used. The application of a fighting stick able to detect techniques will work great as a training guide in order repeat learned techniques in a prescribed order. It will help to verify whether a user performs the technique a supervising system – e.g. a mobile phone app or any other interactive system – is asking for. Typical wearables are more or less part of the past generation, while cooperative detecting systems integrated into one or multiple sport devices that evaluate more than one measuring point represent the beginning of the future of *cooperative wearables*. Cooperative wearables show a high potential for addressing the users' needs for a holistic evaluation of the sport, rehabilitation, or everyday motion and interaction with all kinds of devices.

Lifetime advantages can outperform traditional approaches thanks to reduced communication load if multiple SNs and multi-hop routes through larger WSNs are part of the application as introduced in Section 5.3.2. Currently only two SNs are involved in the proposed application Distributed Event Detection, hence, the advantages are not fully exploited. Our proposed application shows the realizability of the technology, while a larger scale network with smaller SNs will benefit much more in comparison to classical information fusion approaches. The ongoing size reduction of SNs motivates the increase in network sizes that will leverage the applicability of the introduced technology. The ubiquitous integration of conceivably hundreds of SNs and devices into our environment and clothing paves the way for all kinds of conceivable collaborative systems.

7.3 Therapeutic Exercises

In contrast to the previously introduced rigidly coupled version of the Wireless Motion-based Training device which was affected in a diverse way, we now investigate our system in a flexible environment structure with three SNs. We want to investigate movements from the domain of rehabilitation and health care that diversely affect the three SNs. The idea of a rehabilitation based application for Distributed Event Detection was first introduced in [143].

Therapeutic exercises help support our health care and are useful in broad applications of rehabilitation. The correct execution of these exercises is essential to make decent progress during rehabilitation or as a prophylactic action. As therapeutic exercises require moving different parts of the human body in a diverse fashion, every SN observes a different motion, which should be beneficial for our event detection system. It is important to rate the quality of an exercise in order to give a concise extrinsic feedback about the improvement or the deterioration of the exercise performance.

We investigate three different exercises in order to show, for example, the direction of possible application types as well as limitations given by the introduced Distributed Event Detection System. First, we want to address and investigate the capabilities of our Distributed Event Detection System to distinguish different exercises. Second, we want to address for this application the special case of event rejection by evaluating the rejection capabilities in case of incorrectly performed exercises.

7.3.1 Experimental Setup

In order to reflect and map all possible motions of the hand, forearm and upper arm (forward/backward, up/down, left/right translation pitch, yaw, and roll rotation), a SN has to be attached between each joint at a minimum: between the wrist and fingers, between the wrist and elbow, and finally between the elbow and shoulder. This enables the creation of a holistic view of the whole arm motion.

For classification of the exercises performed by the user, one F4VI2 SN as introduced in Section 6.2.2 ($n+1$) is placed on the back of the hand by using a simple bicycle glove, a second SN (n) is placed on the forearm and a third SN ($n-1$) is placed on the upper arm. Placement positions are depicted in Figure 7.24.

Each arm exercise mentioned in the following was repeated 100 times by 3 test persons. The Dumbbell Curl was performed by lifting a 2.5 kg weight as depicted in Figure 7.24. All test persons were male and right-handed and performed the exercises with the right arm exclusively, which means that the three SNs were attached to the right arm.

7.3.1.1 The Bowman

The goal of the bowman exercise is to enhance the muscle skills of the shoulder, elbow, and wrist in order to learn how to move the joints with an improved dexterity by using the principle of performing contra motions. Contra motions are difficult to learn if one is not accustomed to them, but they offer a high potential to improve one's capabilities in coordinating motions. They further improve the knuckle musculature with the goal of stabilizing motions by using the according knuckles under high weight load without running into the risk to cause an injury. We

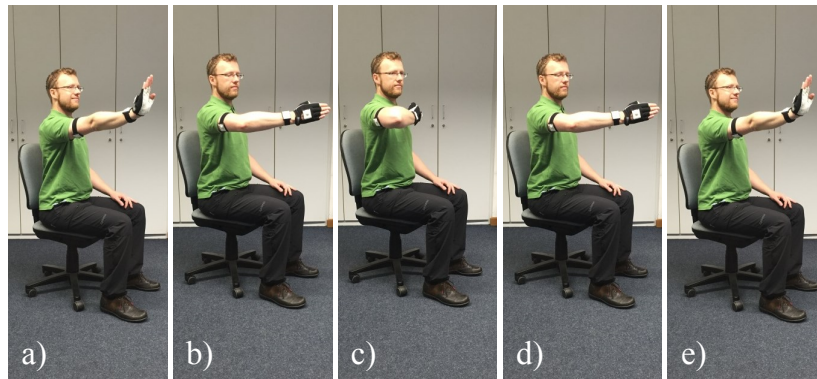


Figure 7.22.: The Bowman start position (a): Extension at the elbow, joint(end position (c): Flexion at the elbow joint.

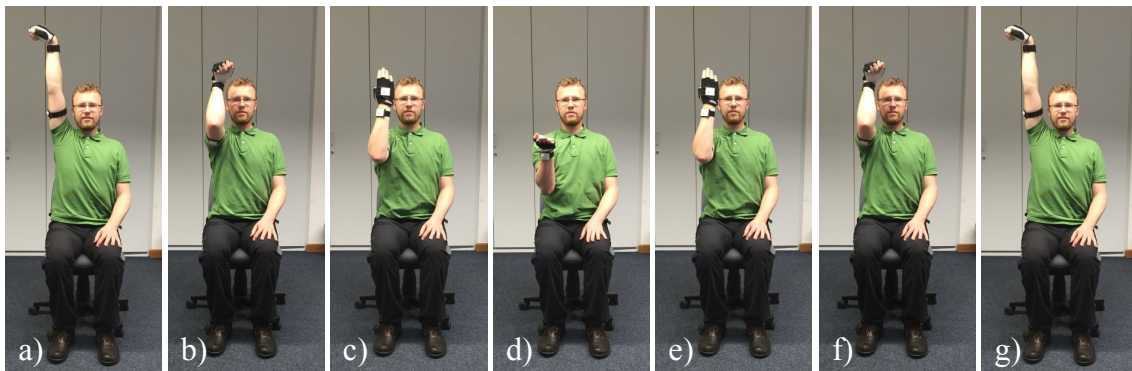


Figure 7.23.: Armpattern Classic Frog: Practicing the primary movement of the right arm [150].

cite the German introduction of the bowman exercise from Spirgi-Gantert et al. [150] as closely as possible:

“*Exercise directions: Sit on a chair and lift your right arm forward until the hand and the elbow are on an equal level with the shoulder. The thumb points to the top. Imagine that your hand holds the end of an arrow between the index finger and the middle finger, see Figure 7.22 (a). This arrow is then pulled back (see Figure 7.22 (b)) in a straight line to the shoulder, while the elbow moves outwards, see Figure 7.22 (c). The elbow stays on the level of the shoulder. At the same time, the shoulder moves slightly towards the hand. Let loose the arrow and the hand moves forwards in a straight line, see Figure 7.22 (d). Hereby, the elbow tip turns in the direction of the ground, the palm of the hand is directed forwards and the finger tip aims upwards. Pretend that you want to push an object forwards, see Figure 7.22 (e). At the same time the shoulder moves slightly backwards. Stay in this position for a short moment and then let the tension fall off.*” [150]. Finish the exercise by moving the arm back to the initial position in reverse order.

7.3.1.2 Classical Frog Arm Pattern

The classical frog, also abbreviated as *the Frog*, is an exercise to activate the function of the muscles of the abdomen in order to narrow the upper abdomen and to abbreviate the lower abdomen. For this purpose, both arms and legs should be moved simultaneously following a specific pattern. The weight of both arms and legs are used to strain the abdominal muscles with each stroke. As the coordination of both patterns is very complex and both sub-patterns can be trained separately to

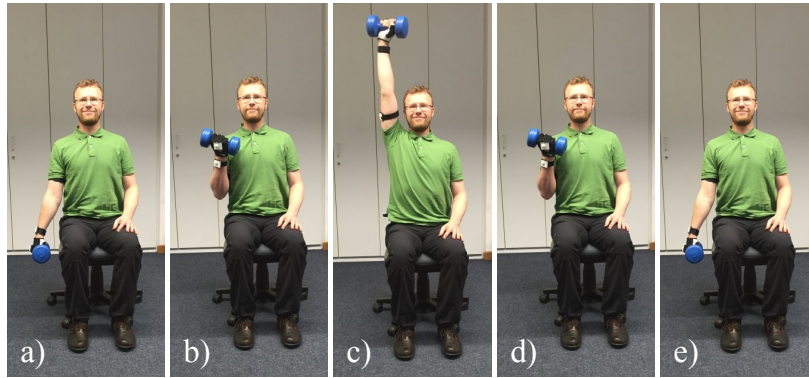


Figure 7.24.: Dumbbell curl exercise: Movement of the arm, adapted from [60, 86].

learn the resulting movement, we concentrate on the arm pattern in this thesis.

We cite the German introduction of the bowman exercise from Spirgi-Gantert et al. [150] as closely as possible: *Sit in on a chair and hold your arms up next to your head. While breathing in, the elbows are stretched and the hands form a fist pointing outwards, see Figure 7.23 (a). This is the starting position. While breathing out, the elbows move downwards in a straight line towards the middle (umbilicus), see Figure 7.23 (b). The hands open up and become fans, see Figure 7.23 (c); the lower arm rotates so that the palms face upwards, see Figure 7.23 (d). The arms are moved back, see Figure 7.23 (e,f) to the starting position next to the head and the hands form a fist again, see Figure 7.23 (g) [150].*

7.3.1.3 Dumbbell Curl

The combination of a dumbbell curl and arnold press helps to strengthen the muscles of the forearm, the upper arm and the shoulder [86, 151], while the intensity of the exercise is regulated by the weight of the dumbbell curl.

“Sit on a chair and hold your arms stretched out to the bottom, with your hands forming a fist to hold the dumbbell tight and safe, see Figure 7.24 (a). The motion sequence comprises four segments. In the first segment, see Figure 7.24 (b), the dumbbell is lifted to the biceps and turned clockwise by about 90° . The elbow does not move. In the second segment, see Figure 7.24 (c), the dumbbell is lifted above the head until the arm is straight. At the same time, the dumbbell is turned counter-clockwise by about 180° . The movement shall be carried out smoothly and without gaining momentum. The third segment, see Figure 7.24 (d), is the reverse movement to the movement in segment (b). The dumbbell is returned to the position in segment (b), with the dumbbell moving downwards and turning 180° clockwise. The movement shall be carried out with the same velocity and as smoothly as in segment (b). The fourth segment (e) is the counterpart segment to (a). The dumbbell is returned to a neutral position and, by doing so, turned 90° counter-clockwise. The elbow does not move. Section (e) shows the neutral position in which the dumbbell is held tightly by the user with the arm hanging down and being stretched. In order not to harm the joints and ligaments, it is preferable that these motions be carried out correctly, especially in the case of increased weight of the dumbbell” [60, 86].

SIGNALS s		Acceleration Data		
		Raw Data		
		X	Y	Z
Feature Types	Orientation	2 (n-1) _{(1,1)(3,2)}	2 (n-1) _{(1,3)(3,4)}	2 (n-1) _{(1,5)(3,6)}
		3 (n) _{(1,7)(2,8)(3,9)}	2 (n) _{(3,10)(1,11)}	2 (n) _{(1,12)(3,13)}
		3 (n+1) _{(1,14)(2,15)(3,16)}	2 (n+1) _{(3,17)(1,18)}	2 (n+1) _{(1,19)(3,20)}

Legend:

# (relative node pos.) _(binID,vector pos.)	Features of Neighborhood Nodes (need to be distributed)
# (central node) _(binID,vector pos.)	Features of Central Node Only (no distribution necessary)

Figure 7.25.: Selected features for human body motions, only relevant features and signals are presented.

7.3.1.4 Faulty Exercises

In case of a faulty execution of one of the exercises introduced above, the rehabilitation and health care concept wants to inform and ideally correct the test person, whereas a correctly performed exercise should trigger positive feedback to motivate and encourage the test person. As an improvement compared to the previously introduced fighting stick, we defined the following list of four faults for every exercise, derived and changed from [59]. The faults are introduced into the according exercises in an alternating fashion in order to verify how the Distributed Event Detection could support the test person in such cases.

All exercises were performed by a person who did not contribute to the training set in order to guarantee a true evaluation of the effectiveness of the detection system.

We investigate the Euclidean distance of the resulting classification towards the reference vectors of the correct event class in order to compare the distances caused by faulty exercises and correctly performed exercises. In case of a clear difference between the euclidean distance of incorrectly performed exercise and a correctly performed exercise, we can apply this logic to notify the test person about faulty exercises.

Faulty performance of *The Bowman*

- Starting the exercise with the thumb pointing to the floor during the whole exercise
- Starting the exercise with the palm held parallel to the floor but moved correctly for the rest of the exercise
- Exercise rotation of the hand overacted so that the back of the hand tries to point to the floor at the position shown in Figure 7.22 (c)
- Elbow lowered to the hip at the position shown in Figure 7.22 (c)

Faulty performance of *Classical Frog*

- The exercise starts with correctly stretched arms but the fist points inwards instead of outwards
- The exercise starts with slightly bent instead of stretched elbows

- While the elbows move down in a straight line to the middle (umbilicus), the palm faces towards the body instead of upwards
- The exercise is performed as described but the fist or palm always face the floor, which means that the lower arm rotation is not performed.

Faulty performance of *Dumbbell Curl*

- No rotation of the shoulder joint, which leads to a rigid dumbbell orientation
- The rotation direction of the hand is inverted, which means that the rotation of the hand exercise is performed clockwise from Figure 7.24 (a) to (b) and counterclockwise from Figure 7.24 (b) to (c). The reverse exercise from Figure 7.24 (c) to (d) is performed accordingly.
- The exercise is performed only from Figure 7.24 (a) to (b).
- The exercise is performed but the elbow is moved with a vital momentum between Figure 7.24 (a) to (b) to support the weight lifting.

We distinguish in the subsequent evaluation three types of classification results:

- Exercises classified as *correct* are correctly performed and classified exercises of the specific exercise, e.g. bowman. We are interested in the Euclidean distance from the classified vector to the according reference vector.
- Exercises classified as *faulty* are performed as described above with alternating faults but correctly classified. We are interested in the Euclidean distance from the classified vector to the according and reference vector.
- *Misclassified* exercises are other correctly performed exercises that have been classified as the exercise trained at this moment. This means the user is doing something different but the detected class is the one we are looking for.

7.3.2 Results

The evaluation was performed by executing 30 repetitions of each exercise by a fourth person who did not take part in the training. The selected 20 features only comprise orientation features from the given acceleration signals, while mostly the first and the last of three bins were used as depicted in Figure 7.25. All 3 signal dimensions x , y , z are used nearly equally. The number of features extracted by SN $n+1$ and n is 7 in both cases while $n-1$ has to extract only 6 features. Therefore, we can choose between the SNs $n+1$ and n to assign the role of the CN to one of these nodes to minimize the transmitted features.

The results in Figure 7.26 show 100 % for all used metrics and show that all events could be performed perfectly separated from each other, which confirms the perfect separability of the different classes. In contrast, the incorrectly performed exercises did not reach metrics as high as the correctly performed exercises. The dumbbell exercise has the highest metrics with at least 93 %, while both the bowman and the frog exercise reach comparably low metrics with an accuracy of 77.8 %. On average, the classification performs less accurately, reaching an accuracy of 84.4 %. The confusion matrix of the faulty exercises shows that almost all events were mixed up between the bowman and the frog.

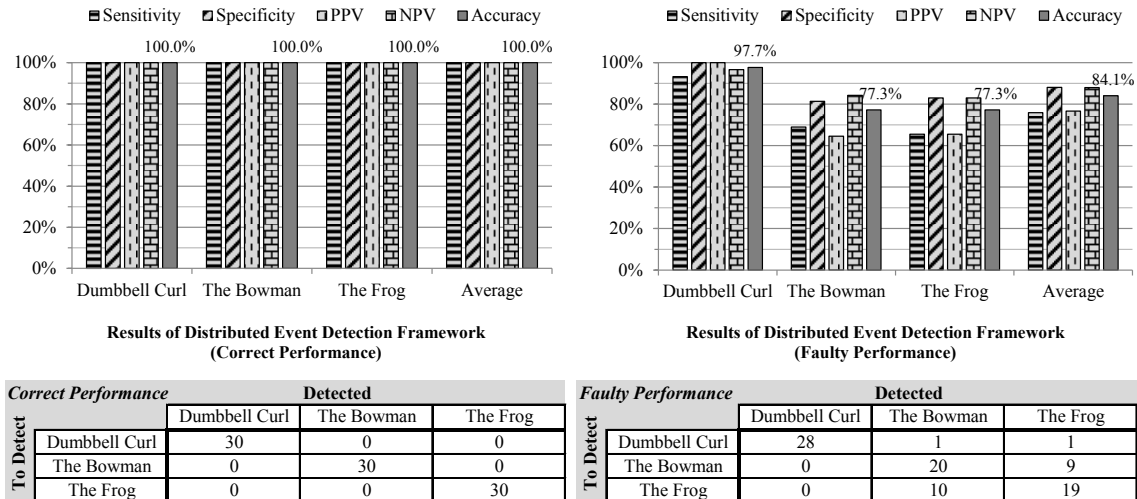


Figure 7.26.: Results of correctly (top left) and incorrectly performed exercises (top right) including the corresponding confusion matrix view (bottom) for both experiments.

The results in Figure 7.27 show the calculated Euclidean distances for the 30 correctly performed exercises as a green box-plot while the Euclidean distances of the 30 intentionally incorrect exercises are shown as orange box-plots. The white box-plots represent the movements misclassified as this particular exercise while performing other exercises.

By using the maximum distance value of the correct bowman exercise as a rejection boundary, 60% of the incorrectly performed motions are correctly recognized as incorrectly performed exercises. In addition, a second boundary can be used to distinguish faulty exercises from misclassified exercises with 100%.

By using the maximum Euclidean distance of the correctly performed classical frog exercises to their corresponding reference vector, it is possible to declare 58.3% of all misclassified instances as one of the other exercises.

By using the maximum distance value of the correct dumbbell exercises as a rejection boundary, 85.7% of the incorrectly performed exercises are correctly recognized as a dumbbell exercise with a faulty part. As no misclassification arose for this exercise, an according rejection boundary is not necessary.

7.3.3 Evaluation

The orientation feature type is selected in combination with the x, y, and z signal only, which shows that the events are easy to distinguish by their physical direct motion characteristic. A visual inspection of the exercises shows that the events have a very diverse effect on the SNs, which simply means that each SN is affected in a completely different way by the exercises.

Obviously, the feature selection could successfully separate all three classes from one another. The used features represent all partitions of the event at least once. This means the feature selection takes all parts of the motion into account in order to build pattern references from the training data.

In terms of a holistic assessment of the performance of the introduced exercises, the proposed approach lacks the completeness of investigating all parts of the event on all axes as the middle partition of both the roll and pitch axes are not covered by features as depicted in Figure 7.25.

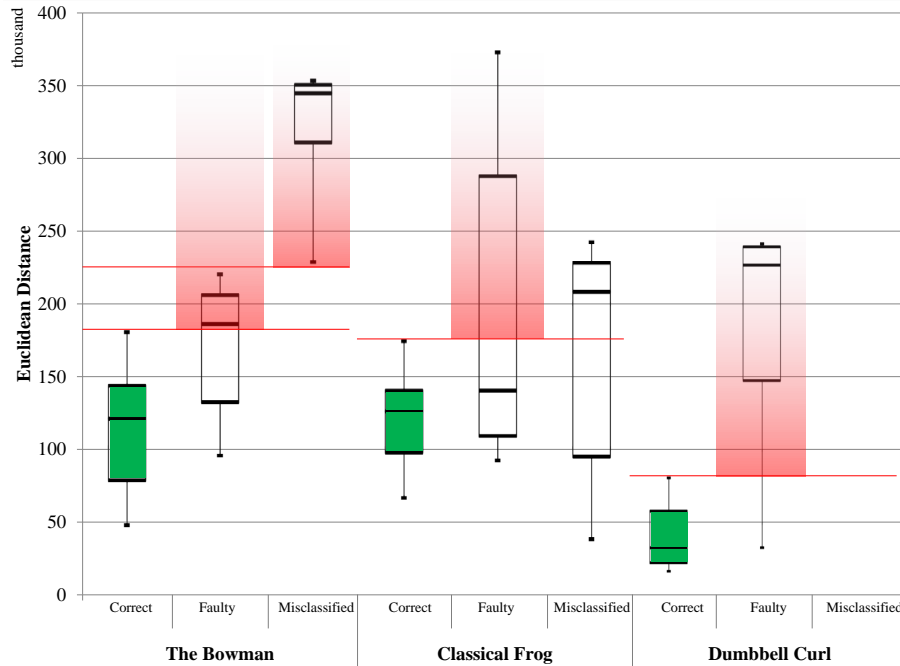


Figure 7.27.: Euclidean distance result investigation for rejection boundary definition (red line). Comparison for all exercises for the correctly and incorrectly performed exercises as well as misclassified exercises, adapted from [59].

A simple solution could force the feature selection algorithm to simply pick at least one feature for every partition and axis. Although this idea stands in contrast to the goal of minimizing the communication load, this application could benefit from a feature vector that represents all axes in all bins and thus all partitions of the exercise. If an increased communication load is acceptable, a more event-dependent assessment of the path of the exercise should create an increased number with according features to represent these partitions. The partitions should further depend on the natural motion segments which could help to support the event characteristic, as introduced in [60, 86].

The Distributed Event Detection System works perfectly for this application, but as mentioned, it should be improved by an enhanced feedback system to give the user a more helpful extrinsic feedback. We can say that this application works best in comparison to our other applications.

Even if we add intentional faults to the exercises as introduced in Section 7.3.1.4, the classification still works for a good majority of the exercises. The reason of course has to be owed to the limited number of three classes that are available for a classification in this particular application. Nevertheless, if we investigate the distances that are calculated for the according classes, we can see that we could use the maximum distances as safe rejection boundaries in this application without sacrificing true positives, see Figure 7.27.

We wanted to know whether it will be possible to use a Euclidean distance rejection approach to notify the user in case of an incorrectly performed exercise. As depicted in Figure 7.27 and indicated by the red horizontal line for each exercise, a strict rejection boundary helps reduce finding faulty exercises as well as misclassified exercises without sacrificing the event classification of correctly performed exercises. The rejection boundary is not able to find all faults, and therefore should not be used for health-critical systems, as a fair amount of faults will not be detected by that approach. If an increased time exposure for a training of more exercises is possible, a faulty

performance of a specific exercise can be trained and added to the classification model in order to be able to detect it like all other classes. Although this is a work-intensive approach, it is very effective.

The user will benefit from the rejection boundary if notifications can obviously tell that the wrong exercise was performed. The bowman and dumbbell exercise perform well with the introduced rejection boundary, while the frog exercise makes it hard to distinguish incorrectly performed from misclassified exercises as both types have very similar ranges of ED to the reference vector of the classical frog.

The classification of the Distributed Event Detection system works perfectly for the introduced exercises. A rejection boundary can improve the notification quality towards the user in contrast to our system without a rejection boundary. It should be noted that further improvements for the usability of the dedicated application might be necessary.

7.3.4 Lessons Learned

We can state that therapeutic exercises seem to fit perfectly to the Distributed Event Detection if we consider the classification results. With three SNs in use, the network aspect is rather easy to manage and the role of the CN can be assigned to each SN. Each SN observes a distinct part of the event and adds a unique perspective to the event detection which fully supports the qualities of the Distributed Event Detection System. The classification results confirm this assumption.

The communication is very limited as only three SNs are involved in the current communication and a single-hop environment is not particularly beneficial for the introduced system in terms of energy demands as evaluated in Section 5.3.2. Further, it is to be expected that SNs' size will decrease in order to fit into a training suit. Such suits with multiple, very tiny, integrated SNs will communicate wirelessly in order to provide comfort. In addition, the number of used SNs and sensors is huge in the early adopting community of *Quantified Self* [152] and will increase with the decreased size of such platforms. Existing Quantified Self tools will benefit from this approach as the fully integrated evaluation of arising events allows using the system without the connection to an infrastructure such as the Internet, a cloud service, or a dedicated server - even a mobile phone is not needed as the pure classification results can be indicated with very simple audio, visual, or haptic feedback techniques. This will add a new level of comfort to Quantified Self. The approach will be very suitable if the SNs have to cooperatively assess a human movement or detect a specific and dangerous movement in advance in order to warn the user. Such critical movements could arise for people affected by hemiparesis, rescue troops, or people with orthotics or prosthetics.

We can state that in particular, multiple differently moved and sensorily affected sections are better to distinguish from one another than uniformly moved sections are. Industrial facilities with multiple measuring points which are distributed over a huge area that cause multi-hop routes can offer a proper self-deciding application problem of a business nature that should very much fit into the qualities of the proposed system.

We can state that the proposed system has the potential to be trained for more events and more similar manifestations of these events like specific faulty events as the chosen exercises did not exhaust the Distributed Event Detection system.

7.4 Bridge Surveillance

In the research area of **SHM**, the current research employs **WSNs** to solve the complex task of finding a viable trade-off between the cost of additional surveillance infrastructure, optimal safety, and necessity. The information base of buildings is still under research and it is often the goal of the research to understand the buildings by investigating the structure and to compare the real world to a model as done in [153, 154]. Typically, it is important to collect these data in long term projects and to offer a solution that enables a raw data based **SHM**. As evaluated, the **WSN** communications of raw data of events causes a high communication load – examples of such approaches can be found for lab deployments in [155, 156] as well as real world deployments [157–159].

Most existing approaches with real-world applications follow the basic ideas laid down by Doebling et al. [160] in order to understand the structure, detect critical events and evaluate new measurement approaches. These solutions typically use model driven approaches [155, 160–162].

For classic damage identification methods, Doebling et al. [160] identify the following categories: visual/localized, magnetic field methods, radiography, eddy-current methods and thermal field methods. All of these methods share the drawback that they are applied post-fact; i.e. they require the knowledge that the structure is damaged in order to be applied. In contrast to this, the authors motivate the exploration of global quantitative detection methods (which examine the changes in the vibration characteristics of a structure) which can be adapted to multiple structures and can recognize damage without a priori knowledge. Most of the quantitative global damage detection methods which can be applied to complex structures examine the changes in the vibration characteristics of a structure. This idea is in general the motivation for the research discussed in this thesis. Only a few of the quantitative approaches use wireless technology, and even fewer calculate the results in a distributed manner (see [154, 159]) or are trainable (see [157]).

An additional problem arises because of the raw data communication which costs more time than the communication of a compressed event packet due to the sheer amount of data that has to be transferred. Additional time has to be reserved for the computational off-line modeling that has to be afforded at the data fusion and evaluation center or **BS**. A swift reaction in case of an emergency situation is not possible with such solutions. The detection of damage or a critical incident (event) has to be as done as soon as it occurs, which means that (mathematical) model based simulations, e.g. Finite Element Model (**FEM**), that need all acquired data on one processing unit are unfeasible.

7.4.1 Detection System to SHM Adaption

Recent advances in **WSN** technologies in combination with our distributed event detection can removed this hurdle of model based off-line evaluation and facilitate cheap, swift and reusable deployment of **SHM** systems even after building a structure. Our system relies on distributed measurements carried out by independent **SNs** cooperating to detect damage events without the need to rely on a base station for processing. The adaptation of our system for **SHM** tasks implies a long-term surveillance of structures with an in-network event evaluation that informs skilled personnel e.g. to investigate the structure when the need arises as mentioned first in [143].

Thanks to its wireless nature, our system can be deployed after completion of a structure and without significant construction effort, thereby protecting heritage sites or buildings whose techni-

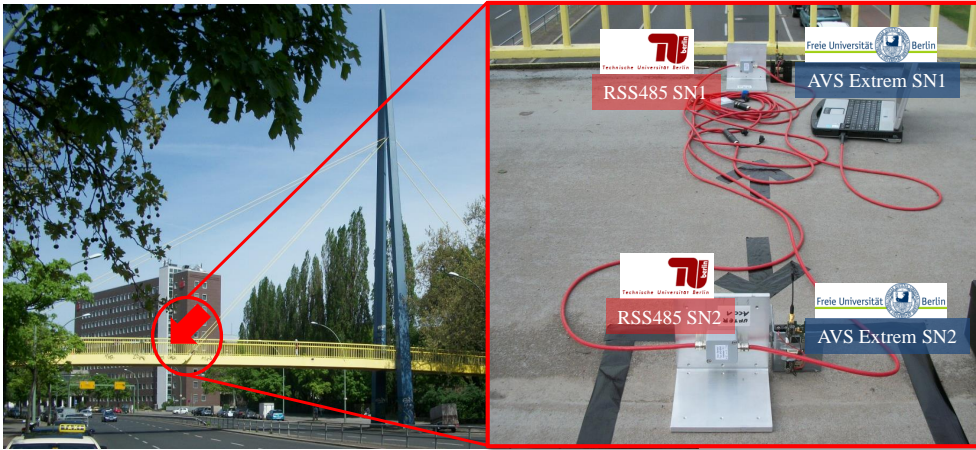


Figure 7.28.: Setup of the bridge experiment at the Volksparksteg in Berlin Figure 7.28 (left hand) to validate our *AVS-Extrem* hardware concerning the capabilities to extract proper natural frequencies by comparing the system to a reference system, adapted from [83].

cal shortcomings are only recognized later in their lifespans. Because of the cooperation between *SNs*, structural changes may not only be evaluated upon scheduled inspection, but also put into context in real-time, including appropriate reactions such as raising an alarm during the report process block as introduced in Section 4.3.3. As introduced, a sensitive system with in-network evaluation capabilities (see the application filter Section 4.3.3.1) will be able to decide on its own whether to notify the arising event, thus preventing transmission overhead.

Our approach fuses the properties *trainable*, *distributed event detection architecture*, *pattern recognition detection algorithm* and *real world deployment* into a system that is adaptive to all kinds of structures, trainable to detect numerous kinds of structural damage as long as reference data exists. Hence, we suggest using the distributed event detection system as a statistical approach that uses the physical and local characteristics of the bridge in order to classify the damage.

As introduced, our approach is based on continuously gathering sensor data for relevant events. The data can then be used to extract descriptive features algorithmically by determining representatives for each class of event. This results in a concise “description” of each type of event as well as the undamaged state which can be easily stored on each wireless node. This in turn facilitates a distributed event detection scheme in which nodes exchange identified features in order to create a unified view of the structure. As a result, the system is capable of deciding whether an event has occurred and subsequently classifying this event.

7.4.2 Preliminary System Validation

For the subsequently introduced application evaluation we have to validate whether our hardware is capable of calculating at least the first two natural frequencies as classical and global assessing parameters for a bridge, see [83]. In order to guarantee that our system and hardware is able to calculate natural frequencies precisely enough, we evaluated our measurement using a wired and highly sensitive reference system (RS485) in a field test (see Figure 7.28) with 2 *SNs* in cooperation with the Institute of and Geoinformation Science of the Technische Universität Berlin (TU Berlin) [163]. The reference system has a known measuring error of 10^{-5} Hz. We compare the resulting

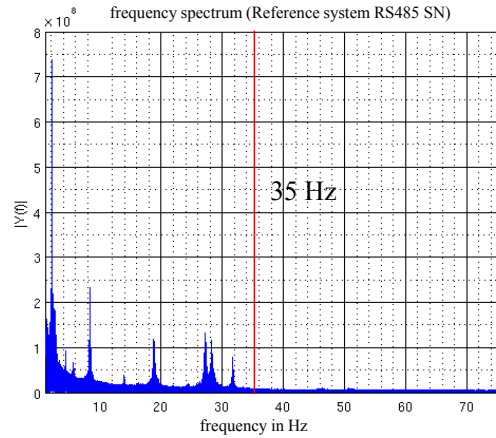


Figure 7.29.: Results of our preliminary bridge measurements show a maximum relevant frequency of 35 Hz in the frequency spectrum of the Volksparksteg, adapted from [83].

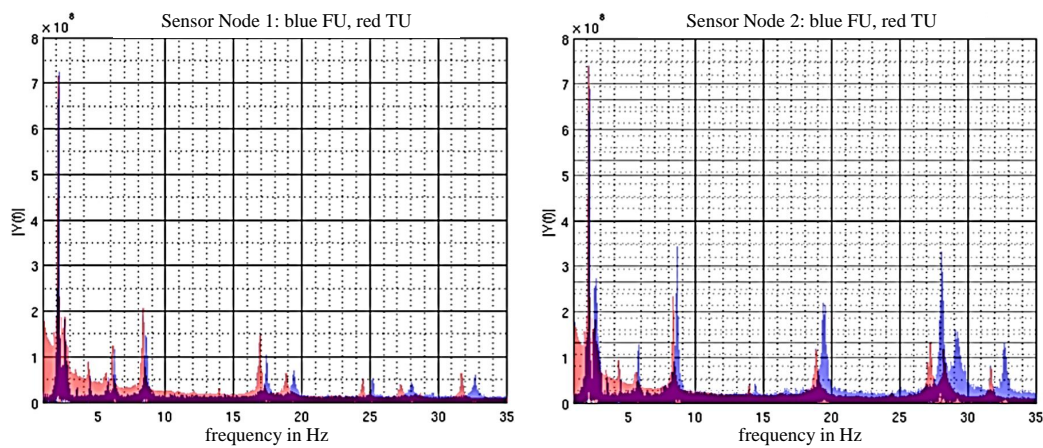


Figure 7.30.: Frequency spectrum results of our AVS-Extrem hardware compared with the reference system (RS485) in order to validate the capabilities to reflect the natural frequencies, adapted from [83].

natural frequencies of our system to the TU Berlin reference system during ambient excitation of the bridge in order to investigate real world capabilities. We placed two SN directly next to each other on the bridge of the Berlin Volksparksteg. The SN were placed as depicted in Figure 7.28 (right hand); we could not place the SNs over the full width of the bridge. We had to place one SN on the side and one SN in the middle of the bridge as the wired setup of the TU Berlin would have hindered the passers-by during jogging and walking. Of course, we could force the passers-by to jump or walk over the cables but unfortunately, the wired construction of the TU Berlin was extremely sensitive, so that even slight disturbances of the cables could alter the measuring behavior. Hence, we prevented passers-by from stepping on the cable by simply leaving an open corridor for them to use. All SNs were mounted on aluminum angles and placed on the ground. During the entirety of the experiment we did not move or touch any part involved in the measurement.

In the first step, we evaluated the maximum necessary sampling frequency, which depends on the arising natural frequencies of the bridge. For this purpose we sampled for 40 minutes with the reference system acceleration data in a preliminary setup experiment as recommended by [164] at

150 Hz. The result proves that the subsequent investigation can be performed at each SN with 70 Hz to cover a maximum detectable frequency of 35 Hz (see Nyquist–Shannon sampling theorem) which covers the relevant frequencies for this bridge, see Figure 7.29.

We collected the acceleration data with our AVS-Extrem SN while passers-by walked over the bridge and cars passed the road under the bridge. By comparing the results (please refer to Figure 7.30), we can see that both SNs calculate frequencies with a constant error between them of 3.5% to 3.8%. These errors are caused by manufacturing and can be eliminated by an initial offset calibration. As a result, we have a random error δ_{random} of 0.3% ($3.5\% < X < 3.8\%$, $\delta_{\text{random}} = 0.3\%$). Based on these results, we can say that the natural frequency extraction works well for our purposes if we compensate for the constant error in the tests with a preliminary calibration routine. We could successfully calculate the natural frequency as a classic and global assessing parameter for the bridge at the AVS-Extrem SN.

7.4.3 Experimental Setup

The goal for the bridge setup was to build a database of raw data from the acceleration sensors on the bridge in order to later apply data analysis methods. The bridge was equipped with four AVS-Extrem SNs. The SN are enclosed in black plastic boxes to prevent animals or bad weather conditions to disturb the long term deployment, see Figure 7.31(b). In contrast to the previously introduced application (Fence Surveillance, Sport Device, and Therapeutic Setup) the SNs of this experiment are equipped with power cables to connect the SN to a constant power source. This setup was necessary to enable further long term research and to sample with a constant 150 Hz without the need for regular battery maintenance. As our institute is not located near the wooden oak research bridge, this setup turned out to be feasible for our research.

Each SN continuously samples with a rate of 150 Hz to cover a maximum detectable frequency of 75 Hz. Each sample contains the acceleration value for all three axes x , y , and z as a 16 bit integer value resulting in a payload of 6 byte per sample. Taking the sample rate into account, this results in a data volume of 900 byte per second for each node.

For the control center we also use one of our wireless SN as gateway. The node was connected to a PC which reads the incoming samples from the SN during training and stores these samples in log files. During the detection, this gateway only transmits the event types which can be represented by simple numbers.

We gathered raw data of multiple events initiated by the bridge excitation, with each event being defined by a different bridge setup. We performed a single jump in the middle of the bridge to perform the excitation, see Figure 7.31 (a).

In order to investigate the gathered bridge data for the evaluation framework, the SNs transmit the data to a raspberry pie based base station which preprocessed the raw data into event data chunks. We have access to these data via the VPN of the University or using a website that grants immediate access and push services for later event reports.

7.4.3.1 Event Experiments

Based on the preliminary system validation in Section 7.4.2, we started to set up subsequent experiments with a wooden bridge located on the grounds of the Federal Institute for Materials Research & Testing - Fabekstraße Berlin (BAM). The advantage of this bridge is that we are able to change certain elements of the bridge by unscrewing or adding some parts or even slightly

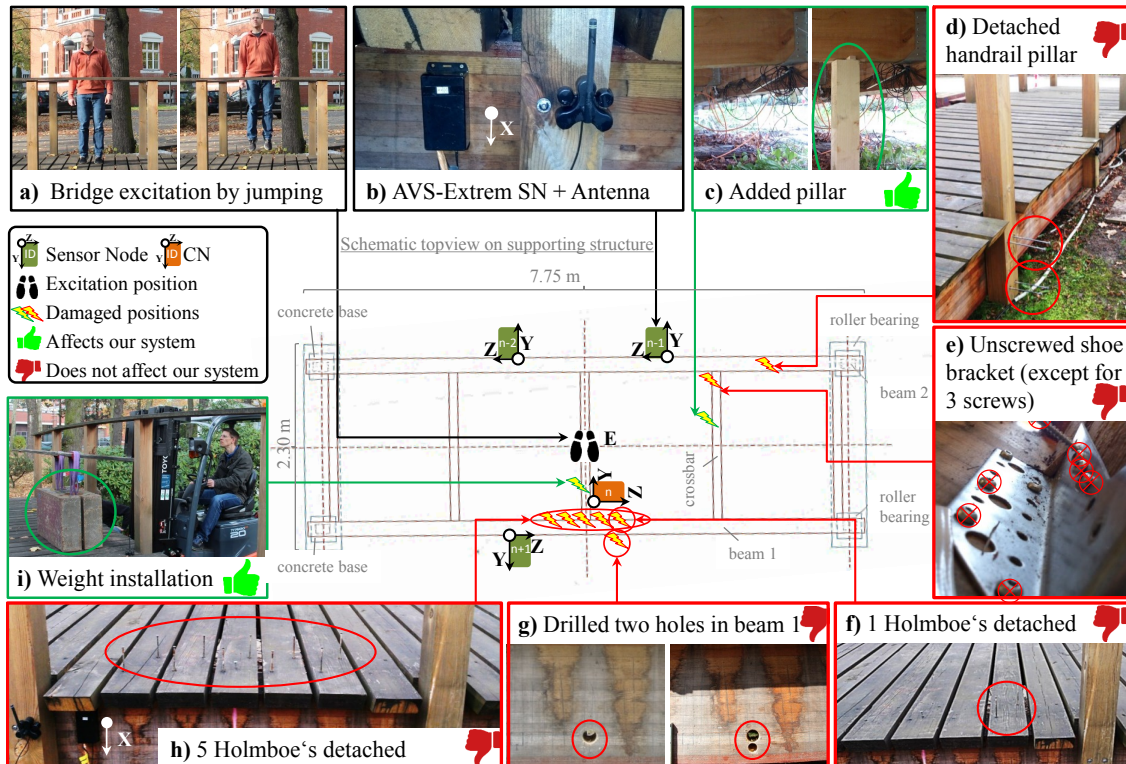


Figure 7.31.: Setup of the bridge experiments and event investigation. Some forms of bridge damage or changes did not affect the detection system. Added weight and pillars are recognizable. The bridge sketch is reproduced with permission by the project documentation of the IaFB Berlin - Institut für angewandte Forschung im Bauwesen.

damaging the bridge. For all changes, we had to request a permission from the [BAM](#) which meant that we were limited in our experiments when it came to drastic changes such as destroying the bridge. However, we are very grateful for the opportunity to change even a few parameters of the bridge. The subsequent experiments were performed in order to answer two main questions: (i) which changes in the bridge our system is able to detect and (ii) whether we can distinguish between these detectable changes.

Relevant Event Identification Our first investigation covers the relevant damage, changes, or additions to bridge that we could track with the most recommended characteristic of a bridge, the natural frequency. The goal of this experiment was to find out which changes to the bridge our system is able to detect, or which of these events cause significant changes in the raw data or the derived natural frequencies. The first natural frequency of the wooden bridge is determined at 11.9 Hz at a temperature of 12°C. Changes in the natural frequency ensure that the core property of the bridge is affected. This allows us to elaborate with an ensured repeatable behavior of the bridge that can be influenced by changing the characteristic of the bridge.

In Figure 7.31, we show all general event types that have been investigated with the natural frequency. Only the event in Figure 7.31 (c) (added pillar) changed the first natural frequency to 12.7 Hz and the event in Figure 7.31 (i) (weight installation) changed the first natural frequency to 13.5 Hz if exposing the system to a single jump (see Figure 7.31 (a)). All other events such as detaching a handrail pillar (Figure 7.31 (d)) or unscrewing a shoe bracket to let one crossbar

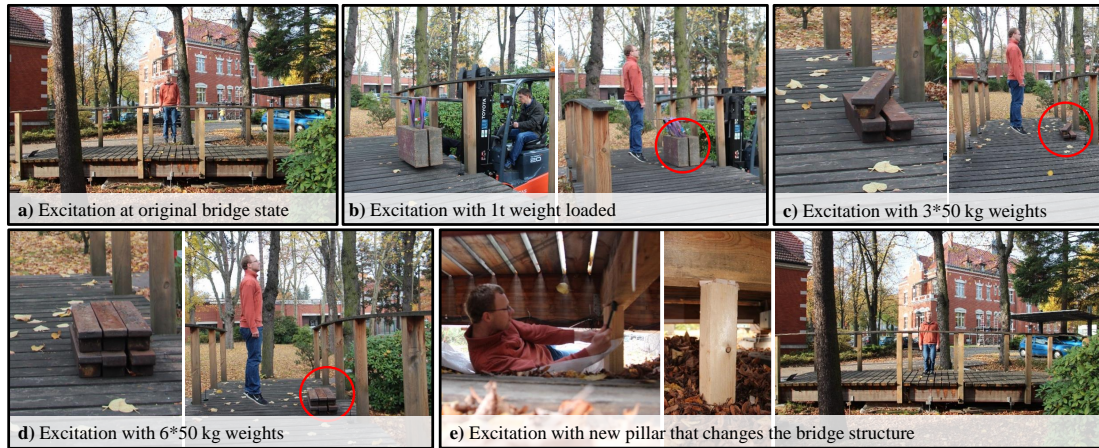


Figure 7.32.: 5 performed experiments (a) to (e) at the wooden bridge. The changes to the bridge are investigated by forcing the bridge to oscillate with a single jump.

just stay loosely with the shoe bracket did not affect the natural frequencies of 11.9 Hz at all. We further detached one (Figure 7.31 (f)) to five (Figure 7.31 (h)) wooden Holmboes, but no change in the natural frequency could be measured by exposing the system to a single jump. We were also allowed to damage the bridge by drilling two small holes, but this change did not effect the natural frequencies either. All subsequent experiments had to be performed with the irreversible change to the bridge in the form of the two holes depicted in Figure 7.31 (g). Hence we continue to investigate our system with the event types of weight installation (see 7.31 (c)) and an added pillar (see 7.31 (i)) while all other events are discarded from further investigations. The reason why we could not detect any effects within the collected data during the discarded event is that the bridge is massively constructed for its small size, hence only exceptionally changes to the bridge will cause any changes to the oscillation behavior at all or sensor with much higher precision are needed to detect any effects during the discarded events.

Our main goal is to show the principal functionality for structural health applications in general. Bridges in particular vary greatly in terms of material, structure, and size, so it is important to know the bridge's inherent characteristics in order to provide meaningful features. We decided to provide only frequency features in order investigate the impact on our events within the frequency domain. For the area of research for SHM, we want to motivate experts to investigate our approach with their in-depth knowledge about structural behavior.

Event Definition It is not proven that the discarded events as described above are not detectable with our proposed approach at all, as this depends strongly on the bridge in general and thus on the used features. We want to ensure that our selected changes to the bridge influence the classic parameters of natural frequency in the area of SHM [158, 165], hence we actually focus exclusively on events that affect the natural frequency of the bridge. We added only frequency based features to the feature selection as these features originate from the frequency domain and help reflect the bridge's oscillation.

Bridges in particular have very diverse characteristics owing to material, structure, and size, so it is important to know the bridge's inherent characteristics in order to provide meaningful features. In future investigations, SHM professionals are called upon to investigate bridges by providing in-depth knowledge about structural behavior to feed the Distributed Event Detection with more

SIGNALS <i>s</i>		Frequency $F(s)$			
		Normalized $N(s)$			
		$N(F(X))$	$N(F(Y))$	$N(F(Z))$	$N(F(\vec{p}))$
Feature-Types	Intensity	$\frac{1}{(n+1)_{(2,7)}}$		$\frac{1}{(n-1)_{(2,5)}}$	
	Histogram			$\frac{1}{(n)_{(2,2)}}$	
	Energy			$\frac{1}{(n-1)_{(2,6)}}$	
	Peak				
	Peak-To-Peak		$\frac{1}{(n)_{(1,3)}}$		
	1. Natural Frequency	$\frac{1}{(n)_{(-,1)}}$			$\frac{1}{(n)_{(-,4)}}$

Legend:

# (relative node pos.) _(binID,vector pos.)	Features of Neighborhood Nodes (need to be distributed) ----->
# (central node) _(binID,vector pos.)	Features of Central Node Only (no distribution necessary) ----->
	Available Features ----->

Figure 7.33.: Selected features for the bridge surveillance based on features of the frequency domain.

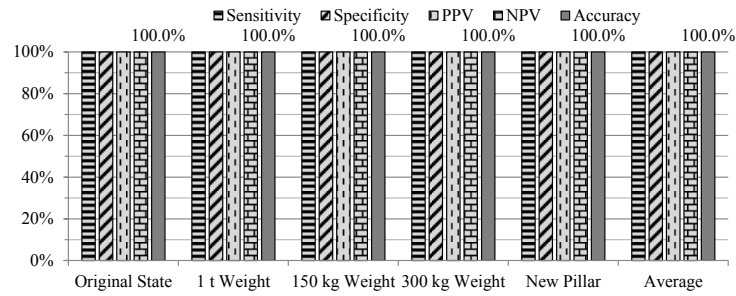
diverse and SHM-specific features. With the following experiments, we want to show that the Distributed Event Detection works properly with the most common parameters.

For our second investigation, we derived 5 events from our previous experiments and will expose the system to a single jump 20 times for each of the 6 event types. The experiments are performed in a fixed order as depicted in Figure 7.32. We start with the unchanged bridge as depicted in Figure 7.32 (a) and perform 20 excitations by jumping at position E in order to approach a dirace impulse, (see Figure 7.31 (E)). For the second event in Figure 7.32 (b) we had to load a weight of 1 t onto the weight installation position with a forklift (see red cross in Figure 7.31). For the third event, we exchanged the 1 t weight with three 50 kg metal rods at the weight installation, see Figure 7.32 (c). As depicted in Figure 7.32 (d) we added three more 50 kg metal rods to the bridge to increase the weight from 150 kg to 300 kg. For the subsequent event e), all weights were removed and we added a pillar to the bridge structure which could be seen as a reverse experiment to simulate a pillar elimination event. We exposed the bridge to 20 more jumps, see 7.32 (e).

7.4.4 Results

The results depicted in Figure 7.34 clearly show that all events can be differentiated perfectly.

The feature selection removed one SN as this node $n - 2$ did only deliver redundant information. Hence, only three SN are necessary for that example application. As depicted in Figure 7.33, the feature selection has selected 3 features from the z-axis, 2 from the x-axis, and 1 feature from the y-axis. The wooden bridge is constructed with one rigid side using a concrete base (left side of the schematic top-view in Figure 7.31) while the other side is flexible and movable, with a roller bearing to provide a movable contact point between the superstructure and the substructure. This roller bearing explains why the feature selection makes use of z-axis features to such an extent. The bridge moves back and forth alongside the roller bearing (z-axis) after the excitation as well as up and down the x-axis. The y-axis adds only one peak-to-peak feature to solve the classification problem. The natural frequency feature is used for the x-axis of SN n as well as the 3D natural frequency



Results of Evaluation Framework based on Frequency Features only

Evaluation Framework		Detected				
		Original State	1 t Weight	150 kg Weight	300 kg Weight	New Pillar
To Detect	Original State	20				
	1 t Weight		20			
	150 kg Weight			20		
	300 kg Weight				20	
	New Pillar					20

Figure 7.34.: Classification results of bridge events based on the feature selection setup with frequency based features (top) including the corresponding confusion matrix view (bottom).

combination of all axes $|\mathcal{N}(F(\vec{v}))|$. As most features are extracted from \mathcal{SN} it makes sense to assign this node the role of the \mathcal{CN} as this minimizes the number of features to be transmitted to \mathcal{SN} .

7.4.5 Evaluation

We thus present the adaptation of our distributed approach to \mathcal{SHM} as a new and additional approach to monitoring bridges. Nevertheless, we want to emphasize that the carefully selected investigated events show the applicability of the Distributed Event Detection in a very early proof of concept for that specific \mathcal{SHM} domain.

In \mathcal{SHM} , it is of special importance to be able to deploy a long-lived system in order to prevent high maintenance costs. Thanks to optimized energy consumption in distributed event detection, the required longevity is guaranteed in this scenario. Even energy harvesting techniques will benefit from the conceptual reduction of communication of the proposed approach as the need for charging processes is reduced if the energy depletes more slowly. This could actually lead to smaller, less expensive, and better integrated energy harvesting devices.

Based on the positive results, we want to motivate the extension of research into the applicability of Distributed Event Detection on events of sudden failure of load bearing elements of halls, bridges, or buildings after changes in weather, traffic, or groundwater. Such accidents can lead to loss of lives, which makes their early detection very important. Events that surpass the load limit of a structure can be reliably recognized using strain gauges as well as acceleration sensors for modal and vibration analysis in the lower frequency spectrum. The proposed Distributed Event Detection could conceivably be applied to dams or pipelines and could also contribute to the detection of manipulation. Distributed Event Detection should primarily add to and improve upon prevalent

visual inspection methods.

7.4.6 Lessons Learned

The process of preliminary signal and event investigation helped figure out a well-adapted setup for the bridge, which resulted in an optimal detection result. We found a perfect bridge setup for the Distributed Event Detection system while knowing which types of events do not fit into the detection portfolio. Nevertheless, it is not said that [SHM](#) researchers can not provide other dedicated features for bridges to support other event types. One strength of the Distributed Event Detection system lies in its high extensibility of features and sensors, which leads to a high flexibility for future applications. As every sensor can be integrated into the event detection process, more and newer sensor types, which may be more suitable for a bridge surveillance system, can be integrated and help extend the capabilities of the Distributed Event Detection system. For the given bridge application it is worth mentioning that we investigated a very specialized and selected scenario which reflects a small problem fraction of the [SHM](#) domain. We want to emphasize that application development for adaptable systems like ours benefit and depend very much from domain experts' knowledge. Interdisciplinary cooperations take on an important role in the area of research where new technology has to be validated towards its applicability for real world applications.

CHAPTER 8

Conclusions and Future Work

The final chapter of this thesis gives a summary of the main contributions comprising the system derivation, the system realization, the theoretic system analysis, and the deployment analysis in order to answer the initial hypothesis, which we want to repeat here:

“An autonomous resource-constrained WSN can be developed and deployed for a wide range of applications under real world conditions to observe and detect events cooperatively with an in-network communication-based event detection approach that clearly enhances network lifetime expectations and offers a reasonable detection accuracy.”

In order to answer this hypothesis, all relevant requirements have been extracted based on a comprehensive related work investigation. The mandatory requirements have been considered and have been subsequently realized in the Distributed Event Detection system realization comprising two frameworks. The first framework is a supervised training framework (Evaluation Framework) that calculates a corresponding model for all trained events offline. The second framework (Distributed Event Detection Framework) is an event detection framework with a generally similar structure for the real WSN that applies a cooperative in-network event detection in order to fuse distributed event characteristics which are extracted according to the trained classification model. The theoretic system analysis shows the system’s low energy demands, that the system is effective even in large scale networks with increased multi-hop length, and that it provides high classification results even in case of inoperative SNs. The system’s impact towards single-SN detection system is shown. The deployment analysis covers an area of four applications in different environmental structures and different ways SNs are affected by events to show the system’s functionality under real world conditions. The application dependent detection performance and the lessons learned show the detection system’s aspect of feasible detection accuracy and wide transferability and applicability.

The presented system extends the known information fusion approaches of raw data, feature, and classification fusion (see [10–12]) by integrating them on the basis of the known input-output scheme of Dasarathy et al. [13]. The proposed Distributed Event Detection system shows high lifetime and classification performance. In case of one of the rare adapted applications that run counter to our recommended reference setup, the proposed system can alternatively apply the underlying information fusion approaches in order to provide the according lifetime that possibly outperformed our system in the above shown analyses.

The Distributed Event Detection system gives larger WSNs a push to make use of the distributed

		Event Requirements																									
		# of Events		Information Fusion			WSN Scale					Realization				Applicability			Algorithm								
		Single Event	Multiple	Raw Data	Decision/State	Classification	Feature	None	One Sensor	Small (2-10)	Mid (11-39)	Large (>39)	Concept	Simulation	Offline Algorithm	Laboratory Test	Real World Test	Theoretic	Specific Application	Domain	Trainable	Threshold Based	Anomaly Detection	Model Based	Fuzzy Based	Pattern Recognition	Hierarchical PR
Event Detection Architectures	Local	3	3		4	1	1	1	2	2	1		1		1	3	2	1	1	4		2		2	1	1	
	Centralized	4	4	7		1	1	4		2	1	1	2	2	2		3	1	6	1		4	1	2		1	
	Decentralized	3	5	2	4	2		2		3	1	1		3	2	3		1	2	2	1		1	3	1	1	2
	Distributed	3	3	2	2	1	1	1		1	1	1	1	3	1	1	1	1	3	1	1	2		1		1	2
	Local + Centralized	1	3		1	2				2		1		1		1	1		2		1	1					2
	Local + Decentralized	5	4		7	4		6		2		2	1	5	1	2	1	4	4	2		3		1	2	1	2
	Decentralized + Centralized	1		1	1			1						1					1						1		



 Fulfilled mandatory requirements for the proposed Distributed Event Detection
 Additionally covered requirements for the proposed Distributed Event Detection

Figure 8.1.: Compressed visualization of all investigated related work papers by summing up the covered requirements within the corresponding architecture type complemented by the resulting features of the realized system.

computational potential in order to evaluate events independently from external infrastructures with the benefit of preserved energy due to reduced data transmissions.

8.1 System Requirements

A generic list of requirements is derived from the investigated related work in order to provide comprehensive recommendations for designing a cooperative Distributed Event Detection system for WSNs.

We investigated the related work for existing event detecting approaches and systems for WSN. We organized the existing related work in seven architectural approaches and investigated the quality of the systems using six categories of multiple event detection requirements.

We compared the related work with our experiences and expectations for a profound Distributed Event Detection system to find a concluding set of requirements. In order to provide future developments for Distributed Event Detection systems for WSNs, we created a generic list of requirements to provide comprehensive recommendations for designing a cooperative Distributed Event Detection System for WSNs. To the best of our knowledge, none of the investigated event detection approach can provide all the derived requirements.

The system’s requirements comprise a Distributed Event Detection architecture as introduced with the ability to differentiate multiple events by performing a trainable, distributed, and feature-based pattern recognition algorithm. The realized system should be deployed with a real world test by ideally using a WSN with at least 11 SNs in order to evaluate a realistic environment for WSN applications. As mentioned by [56, 57], a real world test or deployment is highly recommended in order to show that an algorithm for WSNs withstands real world influences.

8.2 System Realization

A Distributed Event Detection system has been realized comprising two frameworks. The Evaluation Framework calculates a global knowledge-motivated classification model for all trained events offline. The Distributed Event Detection Framework applies a cooperative in-network event detection by fusing distributed event features which are extracted according to the feature selection within the Evaluation Framework. The event fusion and classification is performed by a **CN** that is dynamically filtered by the **CN-Filter**. An additional quality filter helps reduce the potentially increasing in-network traffic if the **SNs** observing the event have a very similar perspective on the event. The application filter allows us to reduce traffic based on application dependent event relevances.

In addition to the mandatory requirements for Distributed Event Detection systems, we fulfilled multiple additional requirements summarized in Figure 8.1 with blue pins.

Number of Events: The introduced system fulfills the derived mandatory requirements by the ability to differentiate *multiple numbers of events* shown with 3, 5 6, and a maximum of 10 events in real world deployments.

Information Fusion: The second requirement asked to use *features* as input for the *information fusion*, which is provided with a comprehensive feature vector that is conceptually integrated by using distance metrics. The feature vectors are assembled over multiple detecting **SNs**. Additionally, the provided *distributed in-network event evaluation architecture* enables the distribution and fusion of the features driven by an integrated filter system to autonomously find the most capable **SN** – called **CN** – that performs the information fusion as well as the event report.

WSN Scale: We deployed the proposed system by using tailored embedded hardware including a serious casing for multiple *small WSN scale* deployments; 2 (Sport Device), 3 (Therapeutic Exercises), 4 (Bridge Surveillance), and one *large WSN scale* deployment with 49 **SNs** (Fence Surveillance).

Realization: We provided a distributed information fusion *concept* for the distributed event detection and to emphasize the potential of the proposed system *theoretically*, we evaluated our proposed approach by investigating the *conceptual* logic of our system by deriving multiple information fusion approaches from the **DFD** model that express all hierarchical sub-components of our proposed system. All information fusion approaches are evaluated on a concept basis to investigate the maximum classification accuracy capabilities.

To provide a comprehensive and highly performant training, the Distributed Event Detection system is implemented *offline* in order to pre-evaluate the expected classification accuracy and to create a classification model according to the trained events. All applications have been *realized* by performing a deployment with a *real world test* for each of the four applications.

Applicability: The proposed system is provided by a high *applicability* made possible through the Evaluation Framework that covers the ability to create new applications by a supervised *training*. The requirements emphasized the need to test the applicability in general, hence we showed the universal applicability of the proposed Distributed Event Detection by deploying 4 applications

which differ in their environmental structure and the way SNs are affected by the corresponding events.

Algorithm: In order to cover the needs for adapting the system to a large amount of applications, *pattern recognition* techniques are recommended to be used as an efficient event detection and assessment algorithm. The proposed system provides and evaluates common pattern recognition algorithms such as the KNN K=1, KNN K=3, NB and the Prototype Classifier. The example resulting applications implements the ED based prototype classifier that has been evaluated to perform best for the current restrictions of WSNs and applications investigated.

8.3 Theoretic System Analysis

If necessary, the hierarchical structure of the proposed system allows falling back to any of the underlying information fusion approaches in order to provide the event detection on a lower level of cooperation, as all compared systems are reused and extended as necessary modules within the Distributed Event Detection system.

We can conclude by investigating predetermined requirements (e.g. network size, events per hour or in-network communication intensity) that the proposed system can outperform classical information fusion approaches with a resulting smaller communication load, an increased lifetime, and a higher applicability.

We showed that the Distributed Event Detection reduces the network load to the BS at the cost of increased in-network communication. We analyzed the trade-off between the increased in-network communication and the reduced communication to the BS in detail with respect to multiple influencing factors to provide relevant parameters that define the range of applicability. The technical limits of our system define concrete scenario requirements, which we provide using a parameter set that enables us to apply the proposed system to a vast amount of applications, with classic information fusion approaches being outperformed regarding energy efficiency and lifetime.

We conclude that the proposed system has the potential to leverage WSNs to be ready for new and trainable distributed and cooperative applications that outperform standard information fusion approaches in their energy efficiency. Further, potential benefits of the proposed system include a new level of autonomy to WSNs as the decision-making ability of the proposed system can directly notify and activate additional systems inside or outside the WSN without the need to communicate with a dedicated headquarter or BS.

The results show clearly that under ideal conditions from two-hop distances and onward (see Fig. 5.6) and from 3-hop distances and onward for the worst case scenario (see Fig. 5.4), the network load is lower for our approach compared to all introduced information fusion scenarios. Smaller networks with 1-hop distances are only advised to apply the Classification Fusion and Feature Fusion approaches.

The additional in-network communication limits the applicability of our proposed approach especially if we have to initiate a second in-network communication or if the number of relevant events that have to be notified increase close to 100 % of all arising events. The RN of our approach has the task to relay all network packets to the BS. The RN of the Distributed Event Detection outperforms the lifetime of all investigated information fusion approaches clearly in every investigated setup. The CN has to deal more often with a less effective energy footprint, but a clear majority of the investigated setups outperforms the competing approaches as at least one of the

competing SNs depletes before the first SN of the proposed system depletes. Two derived reference parameter setups for a broad range of application with up to 50 events per hour, 64 affected SNs but also indicates that the probability of a second in-network communication should stay below 7% or the probability of a critical event should stay below 10%. As a matter of fact optimal setup recommendations have to be calculated based on the applications requirements individually.

The Distributed Event Detection system offers a feasible detection sensitivity of 82.3% with even a single SN. In order to increase this sensitivity by about 11.4 percentage points, the costs of 5 additional SNs, including the in-network communication between the nodes, have to be raised.

As a second in-network communication is an indicator for redundant sensor data over multiple SNs which should indicate a necessary revision of the deployment setup for the affected applications. Instead, the goal of the Distributed Event Detection is in its core to benefit most from diverse sensor readings of multiple and different perspectives which will influence the classification results positively as well as the resulting energy demands; hence, we state that applications with diverse sensor readings based on multiple distinct perspectives are to be preferred for adapting the proposed Distributed Event Detection system.

8.4 Deployment Analysis

For the deployment and analysis we provided a platform architecture comprising a modular hardware layer as well as an extensible application layer. The tailored design of two subsequently developed SNs with appropriate housings for deployments allowed us to perform repeatable experiments with reliable results under real world conditions.

We showed the universal applicability of the proposed Distributed Event Detection with the deployment of four applications which span an area of application in different environmental structures and different ways in which SNs are affected by the corresponding events.

For all four deployed applications, we introduced the experimental setup, presented the results along with an evaluation, and presented our lessons learned. Depending on the application, we go into detail in order to highlight specific historical development results of the Fence Surveillance, the casing setup of the Sport Device, application related requirements for faulty events during the Therapeutic Exercises, and to highlight preliminary system experiments for relevant event identification at the Bridge Surveillance. The lessons learned discussions for all of our four deployments give recommendations and further insights for concrete applications of the Distributed Event Detection system.

Fence Surveillance: The results of the Fence Surveillance of the newly proposed system outperform for a setup with 10 differentiated events with 70.5% sensitivity and 96.7% specificity. The Fence Surveillance application shows very promising results of up to 91.1% sensitivity and 96.2% specificity if the 10 events are categorized into relevance based categories like *critical*, *intentional*, and *uncritical*. We compared our results with our previous approaches of past years and can state that the current system works very well but still cannot guarantee 100% reliable results even if the recommended high quality constructions of fence elements are used. Hence, it is recommended to use our system as an additional security system for existing camera-based or virtual fences as our additional and independent detection system offers high redundant sensory based observation and without the single point of failure a cable based system would imply for existing security systems.

The real impact of a Distributed Event Detection system depends on the application, while we recognized a sensitivity increase of about 11 % for the Fence Surveillance System.

Sport Device: The Wireless Motion-Based Training device application shows varying results depending on the user. An advanced test person achieves objectively better performance results (avg. accuracy 94 %) compared with a beginner test person (avg. accuracy 88 %). Hence, such applications intend to motivate the user to complete the training or to repeat the training in case of too many errors for example. Instead of giving a critical warning, such applications have the ability to remind and support the user to explicitly repeat a specific order of techniques. The device motivated the test persons to repeat the training more often than necessary and with the investigated training device, the correct and incorrect order of the techniques can be detected. The main advantage is still the trainability of the proposed solution. A martial arts instructor can train a series of incorrect techniques according the correctly performed series of techniques. The device is able to differentiate very well between these techniques and furthermore does not require any infrastructure to give an appropriate feedback.

Therapeutic Exercises: The results of the three Therapeutic Exercises reached 100 % when performed correctly. By introducing artificial motion errors, we discovered that using defined rejection boundaries, it is possible to decide whether a motion was correctly performed depending on the metric distance. Especially during a therapeutic exercise, the focus is more on finding out how the motion is performed rather than whether the correct motion is performed. Nevertheless, we consider Quantified Self users to be early adopters of the Distributed Event Detection idea as the approach will be very suitable if the SNs have to cooperatively assess a human movement or have to detect a specific and dangerous movement in advance in order to warn the user, as is the case with people affected by hemiparesis, rescue forces, or people with orthotics or prosthetics. The main advantage should be mentioned once more: the trainability. All incorrectly performed exercises as well as all correctly performed exercises can be trained by experts in order to give the user concrete feedback if e.g. a highly health-damaging motion is performed. The device is able to differentiate very well between these techniques and does not require any infrastructure to give an appropriate feedback.

Bridge Surveillance: The Bridge Surveillance evaluated five a priori selected and evaluated changes to a wooden bridge that could be detected with 100 % accuracy, which shows the applicability of the Distributed Event Detection in a very early proof of concept for the SHM domain. The results will ideally motivate experts from the area of SHM to consider our proposed system as an alternate approach for event-driven SHM surveillance. Although it is very unlikely that trainings will be the path to provide a classification model for a bridge in the future, it is more likely that experts will create a classification model using historical data that can be pre-evaluated offline beforehand using the Evaluation Framework.

8.5 Future Work

The presented Distributed Event Detection has two major improvable sections to work on; these are system *refinements* to optimize the existing system, and system *extensions* to upgrade the existing system with new components.

8.5.1 Refinements

The applicability of the proposed Distributed Event Detection System can be increased by implementing new features, hence an increased feature pool is beneficial for every event detection pool, as more features provide a better opportunity to reflect the arising events more precisely and according to their needs.

The current state of the Distributed Event Detection should be ported to a state-of-the-art operating system like RIOT OS [166], which is the official successor to the Firekernel [4] used in this thesis. RIOT intends to be the Linux for embedded systems in the near future and should therefore be an optimal base for a broad introduction of the proposed system to the embedded developer community.

8.5.2 Extensions

Future research will find a huge potential in the abilities to evaluate concurrently arising events within a system and to differentiate those from each other. As an example, we can use the Fence Surveillance approach. In case of two concurrent intruders the SNs currently have to be trained to such events to be effectively able to classify two concurrently arising events. Other applications that record audio data to detect events based on the variety of sounds for example run into the problem that numerous sounds arise simultaneously. Those sounds do not only have to be segmented, they have to be distinguished to detect multiple distinct and relevant events.

An enhanced event segmentation based on the given Distributed Event Detection System is important to assess the course of events instead of distinguishing events from each other. Particularly applications in the area of health care, sports, or rehabilitation will benefit from this system extension.

An improved feedback system based on sound, light, or vibration could enhance concrete interactive applications for humans to gain a better translation of the detection results into an easily comprehensible feedback.

List of Figures

1.1	General Distributed Event Detection Idea: Multiple sensor nodes observe an arising event from different perspectives to fuse the acquired data based on an a priori trained model.	5
1.2	Centralized Event Detection at the base station causes high network load (left). The Distributed Event Detection (right) reduces the network load to the base station at the cost of increased in-network communication.	7
2.1	Seven event detection architectures and their taxonomic relation in the context of global and local event knowledge.	12
2.2	Related work organized by event detection architecture and requirements.	28
2.3	Compressed visualization of all related work papers investigated by summing up the requirements covered within the corresponding architecture type.	29
3.1	Standard training and pattern recognition scheme adapted from [10]	34
3.2	Example depiction of the nearest prototype classifier and the K nearest neighbor classifier (KNN).	38
4.1	Distributed event detection concept reduces traffic in multi-hop networks during event reporting [5].	44
4.2	Distributed Event Detection System, composed of the Evaluation and the Distributed Event Detection Framework, adapted from [5].	46
4.3	Different signal types are derived from the raw data by applying a normalization, creating a magnitude signal and a frequency transformation, adapted from [5,76]	48
4.4	Process of <i>reference vector</i> and <i>selected features bit-mask</i> creation. By applying the <i>selected features bit-mask</i> the Distributed Event Detection is able to recompile the <i>fused feature vector</i> , adapted from [55].	57
4.5	Model error examination during the quality extraction process of an unknown event, adapted from [76].	64
5.1	Performance comparison of the Euclidean distance with the Mahalanobis distance under optimal conditions and under real world conditions simulated with the LOOCV, adapted from [59].	70
5.2	Input-output scheme of Dasarathy et al. [13] applied to our hierarchical structured information fusion approaches.	73
5.3	[Standard Case] Comparison of payload in [byte] expected for varying hop distances. Critical events arise with probability of 50 %.	81
5.4	[Worst Case] Comparison of payload in [byte] expected for varying hop distances. Critical events arise with probability of 100 %, adapted from [5].	81

5.5	[Standard Case] Comparison of payload in [packets] expected for varying hop distances. Critical events arise with probability of 50 %.	82
5.6	[Worst Case] Comparison of payload in [packets] expected for varying hop distances. Critical events arise with probability of 100 %, adapted from [5].	82
5.7	Energy consumption of essential event processing tasks during the event detection as a basis for our theoretical considerations.	84
5.8	[Best Case] Lifetime investigation with varying parameter $\#_E$ to investigate the influence of the number of arising events per hour, adapted from [5].	91
5.9	[Standard Case] Lifetime investigation with varying parameter $\#_E$ to investigate the influence of the number of arising events per hour.	92
5.10	[Worst Case] lifetime investigation with varying parameter $\#_E$ to investigate the influence of the number of arising events per hour.	93
5.11	[Best Case] Lifetime investigation with varying parameter $\#_{SN_{EV}}$ to investigate the influence of the number of SN affected by an event.	94
5.12	[Standard Case] Lifetime investigation with varying parameter $\#_{SN_{EV}}$ to investigate the influence of the number of SNs affected by an event.	96
5.13	[Worst Case] Lifetime investigation with varying parameter $\#_{SN_{EV}}$ to investigate the influence of the number of SNs affected by an event.	97
5.14	[Best Case] Lifetime investigation with equally varying parameters $\#_{SN_{EV}}$ and $\#_{CNF}$ to investigate the influence of the number of affected SNs and the equal number of SNs passing the CN-Filter by an event	99
5.15	[Standard Case] Lifetime investigation with equally varying parameters $\#_{SN_{EV}}$ and $\#_{CNF}$ to investigate the influence of the number of affected SNs and the equal number of SNs passing the CN-Filter by an event.	100
5.16	[Worst Case] Lifetime investigation with equally varying parameters $\#_{SN_{EV}}$ and $\#_{CNF}$ to investigate the influence of the equal number of affected SNs and the number of SNs passing the CN-Filter by an event.	101
5.17	[Best Case] Lifetime investigation with varying parameter P_{EV_C} to investigate the influence of different probabilities of arising critical events.	103
5.18	[Standard Case] Lifetime investigation with varying parameter P_{EV_C} to investigate the influence of different probabilities of arising critical events.	104
5.19	[Worst Case] Lifetime investigation with varying parameter P_{EV_C} to investigate the influence of different probabilities of arising critical events.	105
5.20	[Best Case] Lifetime investigation with varying parameter P_{CNF} to investigate the influence of different probabilities of passing the CN-Filter.	106
5.21	[Standard Case] Lifetime investigation with varying parameter P_{CNF} to investigate the influence of different probabilities of passing the CN-Filter.	107
5.22	[Worst Case] Lifetime investigation with varying parameter P_{CNF} to investigate the influence of different probabilities of passing the CN-Filter.	107
5.23	Two reference setups with an assumed minimized and a maximized second in-network communication. The lifetime calculation is performed with increasing parameter $\#_{SN_{EV}}$ to investigate the number of affected SNs.	110
5.24	[General Areal Categories] 10 fence events grouped in 3 event categories: Uncritical, Intentional and Critical, applied from [5].	112

5.25	[Construction Site Categories] 10 fence events grouped in 3 event categories: Un-critical, Critical and Time-Dependent, applied from [5].	113
5.26	Evaluation of four classifiers with the evaluation framework. Three setups with the averaged metrics are compared: 10 events, three General Areal Categories and three Construction Site Categories, applied from [5].	115
5.27	Confusion matrix for the four classifiers using all 10 events	115
5.28	Confusion matrix for the four classifiers using the three General Areal Categories, applied from [5]	115
5.29	Confusion matrix for the four classifiers using the three Construction Site Categories.	116
5.30	Selected features for the fused and distributed feature vector from the Evaluation Framework for the trained classes, applied from [5]	117
5.31	Reference vectors extracted during the feature selection process for each class, showing involved and missing features because of structural feature absence and SNs.	118
5.32	Missing feature compensation method comparison with prototype classifier, adapted from [5].	121
5.33	Sensitivity performance comparison with 32 combinations of inoperative wireless sensor nodes, applied from [5].	122
5.34	Box plot evaluation of the six different numbers of active SNs, applied from [5]. .	122
6.1	Platform Architecture depiction composed of the Hardware and Application Layer with references to in-depth publications for further details.	126
6.2	AVS-Extrem hardware sensor node [102]	127
6.3	F4VI2 SN and casing with component description in concept and assembled development stage, applied from [127].	129
6.4	AVS Extrem sensor node and casing: Disassembled, assembled and fence integration visualization	130
6.5	F4VI2 SN and casing with component description in concept and assembled development stage, applied from [127].	132
6.6	Performance comparison of both used sensor nodes, adapted from [127].	132
6.7	Latency experiment results of processing time created by five differently combined parameters: location, radio mode, activity, adapted from [87].	134
6.8	PDR calculation from experiment results created by five differently combined parameters: location, radio mode and activity, adapted from [87].	134
7.1	Taxonomy of investigated applications based on the environmental structure and manner in which events affect SNs.	137
7.2	SN integration into a construction site fence during sunshine and rain weather, applied from [5].	139
7.3	Physical sensor network structure is driven by the fence structure. Event distribution, training and evaluation location depiction, applied from [5].	141
7.4	Comparison of the classification results of all 10 events provided by the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower), adapted from [5].	142

7.5	All 10 fence events compared in confusion matrices. Left: Theoretic results of the Evaluation Framework. Right: Deployment results of the Distributed Event Detection Framework.	142
7.6	Comparison of the classification results of the 3 derived General Areal Categories in the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower).	143
7.7	Comparison of the classification results of the 3 derived Construction Site Categories in the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower).	143
7.8	Comparison of the confusion matrices of the 3 derived General Areal Categories in the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower).	144
7.9	Comparison of the confusion matrices of the 3 derived Construction Site Categories in the evaluation framework (upper) and the real world deployment that uses the Distributed Event Detection framework (lower).	144
7.10	Classification capabilities comparison of seven deployed different fence event detection systems with different setup parameters, adapted from [5].	145
7.11	Classification capabilities comparison within a Receiver Operating Characteristic space graph of seven deployed different fence event detection systems, adapted from [5].	149
7.12	Fence buildup challenges comprise a vast variety of over time changing parameters that can theoretically be covered by an increased training effort, adapted from [55].	150
7.13	Optimal conditions for a fence surveillance application at the example location of the former Berlin-Tempelhof Airport (IATA: THF).	151
7.14	a) shows the real world applications while (b) adapts the technical realization with the Wireless Motion-Based Training with a coupled as well as an uncoupled device. Our experiments are based in the coupled stick variant, applied from [87].	152
7.15	The enhanced casing to create a training device based on the fence casing depicted in Figure 6.4.	153
7.16	Six different fighting techniques trained 20 times and evaluated 50 times, applied from [87].	154
7.17	Selected features for the Wireless Motion-Based Training device, only relevant features and signals are presented.	155
7.18	Classification results of 6 stick fight techniques evaluated with 50 technique for each technique by an advanced test person, applied from [87].	156
7.19	Classification results of 6 stick fight techniques evaluated with 50 technique for each technique by a beginner test person.	156
7.20	Confusions matrix for classification problem of 6 stick fight techniques evaluated with 50 technique for each technique by an advanced test person, applied from [87].	156
7.21	Confusions matrix for classification problem of 6 stick fight techniques evaluated with 50 technique for each technique by a beginner test person.	156
7.22	The Bowman start position (a): Extension at the elbow, joint(end position (c): Flexion at the elbow joint.	160
7.23	Armpattern Classic Frog: Practicing the primary movement of the right arm [150].	160
7.24	Dumbbell curl exercise: Movement of the arm, adapted from [60, 86].	161

7.25	Selected features for human body motions, only relevant features and signals are presented.	162
7.26	Results of correctly (top left) and incorrectly performed exercises (top right) including the corresponding confusion matrix view (bottom) for both experiments.	164
7.27	Euclidean distance result investigation for rejection boundary definition (red line). Comparison for all exercises for the correctly and incorrectly performed exercises as well as misclassified exercises, adapted from [59].	165
7.28	Setup of the bridge experiment at the Volksparksteg in Berlin Figure 7.28 (left hand) to validate our AVS-Extrem hardware concerning the capabilities to extract proper natural frequencies by comparing the system to a reference system, adapted from [83].	168
7.29	Results of our preliminary bridge measurements show a maximum relevant frequency of 35 Hz in the frequency spectrum of the Volksparksteg, adapted from [83].	169
7.30	Frequency spectrum results of our AVS-Extrem hardware compared with the reference system (RS485) in order to validate the capabilities to reflect the natural frequencies, adapted from [83].	169
7.31	Setup of the bridge experiments and event investigation. Some forms of bridge damage or changes did not affect the detection system. Added weight and pillars are recognizable. The bridge sketch is reproduced with permission by the project documentation of the IaFB Berlin - Institut für angewandte Forschung im Bauwesen.	171
7.32	5 performed experiments (a) to (e) at the wooden bridge. The changes to the bridge are investigated by forcing the bridge to oscillate with a single jump.	172
7.33	Selected features for the bridge surveillance based on features of the frequency domain.	173
7.34	Classification results of bridge events based on the feature selection setup with frequency based features (top) including the corresponding confusion matrix view (bottom).	174
8.1	Compressed visualization of all investigated related work papers by summing up the covered requirements within the corresponding architecture type complemented by the resulting features of the realized system.	178

List of Tables

5.1	Comparison of the applicability of the Euclidean and Mahalanobis distance within an event detection system for Wireless Sensor Networks.	71
5.2	Classification Capabilities derived from the Input Detail Level for Fusion	77
5.3	Detailed list of measured and derived energy consumption values of event processing tasks during the event detection as a basis for our theoretical considerations.	85

APPENDIX A

Abbreviations

ACK	Acknowledgement Packet	70
ADC	Analog-to-Digital Converter	34
AODV	Ad hoc On-Demand Distance Vector	125
AVS-Extrem	Autonome vernetzte Sensorsysteme in Extrembedingungen (mst-AVS) ..	127
BAM	Federal Institute for Materials Research & Testing - Fabekstraße Berlin	170
BMBF	German Federal Ministry of Education and Research	132
BPEL	Business Process Execution Language	16
BS	Base Station	1
CBC	Cipher Block Chaining	126
CFDFA	Cluster-based Fuzzy Decision Fusion Algorithm	23
CH	Cluster Head	12
CMSIS	ARM Cortex Microcontroller Software Interface Standard	49
CN	Central Node	7
CPU	Central Processing Unit	128
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance	85
DAI	Data In	77
DAI-DAO	Data In > Data Out	73
DAI-FEO	Data In > Feature Out	74
DEI	Decision In	77
DEI-DEO	Decision In > Decision Out	74
DFD	Data-Feature-Decision	73
dps	Degrees per Second	129
DSO	Digital Sampling Oscilloscope	85
DSP	Digital Signal Processing	128
DSR	Dynamic Source Routing	125
DSS	Decision Support System	15
DST	Destination of a Packet	87
ED	Euclidean Distance	35
FEI	Feature In	75
FEI-DEO	Feature In > Decision Out	74
FEI-FEO	Feature In > Feature Out	74
FEM	Finite Element Model	167
FFT	Fast Fourier Transform	21
FPU	Floating Point Unit	128
GFZ	German Centre for Geosciences	15
GITEWS	German Indonesian Tsunami Early Warning System	15
GPS	Global Positioning System	15
ISM	Industrial, Scientific and Medical	127
KNN	K Nearest Neighbor	38
LBC	Leaky Bucket Counter	14
LED	Light-Emitting Diode	129
LOOCV	Leave-one-out Cross-Validation	57

LOS	Line of Sight.....	62
LZWC	Lempel–Ziv–Welch Codec.....	78
MANET	Mobile Ad Hoc Network.....	51
MCU	Micro Controller Unit.....	3
MD	Mahalanobis Distance.....	35
MEC	Mobile Edge Computing.....	111
MEMS	Microelectromechanical systems.....	14
MMR	Micro Mesh Routing.....	86
MMR	Micro Mesh Routing.....	86
MPU	Maximize Unsafe Path Routing Protocol.....	86
MSB	Modular Sensor Board.....	146
NB	“naïve” Bayes.....	114
NPV	Negative Predictive Value.....	114
PCA	Principal Component Analysis.....	15
PCB	Printed Circuit Board.....	127
PD	Power Down.....	128
PDR	Packet Delivery Ratio.....	9
PEGASIS	Power-Efficient Gathering in Sensor Information Systems.....	86
PIR	Passive Infrared.....	34
PPV	Positive Predictive Value.....	114
PWM	Pulse-Width Modulation.....	129
RAM	Random Access Memory.....	127
RN	Relay Node.....	5
ROC	Receiver Operating Characteristic.....	149
RPL	IPv6 Routing Protocol for Low power and Lossy Networks.....	86
RSSI	Received Signal Strength Indicator.....	14
RTOS	Real-Time Operating System.....	125
RTT	Round-Trip Time.....	134
RX	Receive.....	85
SFSM	Sequential Floating Search Method.....	56
SHM	Structural Health Monitoring.....	1
SIP	System-in-a-Package.....	128
SMD	Surface-Mount Device.....	129
SN	Sensor Node.....	1
SOA	Service-Oriented Architecture.....	16
SRAM	Static Random Access Memory.....	128
SRC	Source of a Packet.....	87
SRD	Short Range Devices.....	127
TCP	Transmission Control Protocol.....	135
TDOA	Time Difference of Arrival.....	17
TU Berlin	Technische Universität Berlin.....	168
TX	Transmit.....	85
UART	Universal Asynchronous Receiver/Transmitter.....	128
USB	Universal Serial Bus.....	128
VIVE	Validierung des Innovationpotentials verteilter Ereigniserkennung in drahtlosen Sensornetzen (Project-ID: 03V0139).....	132
WBAN	Wireless Body Area Networks.....	61
WEKA	Waikato Environment for Knowledge Analysis.....	114
WOR	Wake-On-Radio.....	85
WSN	Wireless Sensor Network.....	1

APPENDIX B

Zusammenfassung

Drahtlose Sensornetze (engl. WSNs) [1, 2] bestehen aus mehreren drahtlos vernetzten, batteriebetriebenen und sensorbasierten Kleinstcomputern – Sensorknoten genannt. Sensorknoten sind typischerweise mit mindestens einer Sensorart ausgestattet, um autonom ihre Umgebung zu beobachten, indem sie Sensordaten erfassen und diese auf der eingebetteten Hardware ggf. verarbeiten und innerhalb des drahtlos vernetzten Sensornetzes auszutauschen.

Ereigniserkennung ist ein beobachtender und bewertender Prozess von realen Vorfällen oder Phänomenen. Drahtlose Sensornetze haben das Potenzial Ereignisse in der Umgebung zu erfassen und zu detektieren, um uns z.B. insbesondere im Rahmen von Sicherheitsfragen zu unterstützen. Dabei sind z.B. Brückenüberwachungssysteme zur Schadenserkenkung zu nennen [3] oder Arealüberwachungssysteme, um z.B. Feuerwehkräfte in Gebäuden zu unterstützen [4], oder Zaunüberwachungssysteme, um Einbruchversuche zu detektieren [5, 143]. Ereigniserkennung in drahtlosen Sensornetzen ist insbesondere deswegen herausfordernd, weil eine übergreifende Bedeutung verschiedener Messungen von unterschiedlichen Orten erzeugt werden soll. Eine einfache Strategie basierend auf redundanter Datenerhebung ist nur bei schwellwertbasierten Ereignissen wie der Feuererkennung sinnvoll. Wohingegen musterbasierte Ereignisse, wie z.B. Einbruchereignisse an Zäunen, von unterschiedlichen Perspektiven auf das Ereignis profitieren.

Die Anforderungen an ereignisüberwachende drahtlose Sensornetze stehen im Widerspruch zu deren Eigenschaften. Dies ist insbesondere dann der Fall, wenn die Laufzeit dieser Netze besonders hoch und eine hochakurate Ereigniserkennung mit zudem miniaturisierten Sensorknoten gefordert ist, deren Ressourcen (Energie, Rechenleistung, Speicher) zudem stark beschränkt sind.

Die vorliegende Arbeit präsentiert ein verteiltes Ereigniserkennungssystem, das den Evaluierungsprozess für die Ereigniserkennung von der Basisstation in das Sensornetz hineinverlagert. Das verteilte Evaluierungskonzept reduziert dadurch die nötige Kommunikationslast, die zu einer Basisstation anfällt, auf eine einzige zu übertragene Ereignismeldung und erhöht damit die Lebensdauer der am stärksten belasteten Sensorknoten und des gesamten Sensornetzes.

Es werden zwei Frameworks vorgestellt, die die Realisierung des verteilten Ereigniserkennungskonzeptes ermöglichen und das Ziel haben, das globale Wissen über die verteilten Ereignisse zu erhalten. Die Frameworks ermöglichen dies, indem sie die vielfältigen Ereignisperspektiven der Sensorknoten trotz Datenreduktion erhalten können. Das *Evaluationsframework* beinhaltet u.a. eine automatisierte Erstellung des Ereignismodells basierend auf Daten eines überwachten Trainings, um eine hohe Übertragbarkeit des Systems zu ermöglichen. Das zweite Framework, *Verteilte Ereigniserkennung*, ermöglicht die tatsächliche Umsetzung realer Anwendungen, indem die Ereigniserkennung auf den Sensorknoten implementiert ist und das zuvor trainierte Modell zur Erkennung genutzt wird. Dabei tauschen unter anderem die vom Ereignis betroffenen Sensorknoten Ereignisinformationen innerhalb der Sensornetzes aus, um die Ereignisse kooperativ zu klassifizieren und je nach Relevanz zu melden.

Es werden die technischen Grenzen des vorgestellten Systems präsentiert, indem der Trade-off zwischen den Einsparungen der Kommunikation zur Basisstation und dem ansteigenden netzinternen Kommunikationsaufwand untersucht wird. Verglichen mit klassischen Ansätzen der Informationsfusion kann dem vorgestellten System eine deutliche Leistungssteigerung im Sinne der Energieeffizienz aufgrund des konzeptionell reduzierten Kommunikationsaufwandes zugeschrieben werden. Alle im Vergleich stehenden klassischen Ansätze der Informationsfusion werden von dem vorgestellten verteilten Ansatz konzeptionell integriert und modular erweitert.

Das vorgestellte verteilte Ereigniserkennungssystem erzielt eine hohe Ereigniserkennungssensitivität von mehr als 80 % bis hin zu 100 % abhängig von der jeweiligen Anwendung. Mittels vier real umgesetzter Anwendungen werden die Funktionalität, die Erkennungsleistung sowie die zusätzlichen applikationsabhängigen Erkenntnisse in Bezug auf die Anwendbarkeit der verteilten Ereigniserkennung evaluiert.

ANHANG C

Erklärung

Ich versichere, dass ich die vorliegende Dissertation auf Grundlage der in der Arbeit angegebenen Hilfsmittel und Hilfen selbständig verfasst habe. Die Arbeit ist nicht in einem früheren Promotionsverfahren eingereicht worden.

Berlin,

Bibliography

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Sensor Networks,” *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–116, Aug. 2002.
- [2] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.
- [3] S. Arangio and F. Bontempi, “Structural Health Monitoring of a Cable-Stayed Bridge with Bayesian Neural Networks,” *Structure and Infrastructure Engineering, Taylor & Francis*, vol. 11, no. 4, pp. 575–587, 2015.
- [4] H. Will, K. Schleiser, and J. Schiller, “A Real-time Kernel for Wireless Sensor Networks Employed in Rescue Scenarios,” in *Proceedings of the 34th Conference on Local Computer Networks*, ser. LCN. Zürich, Switzerland: IEEE, Oct. 2009.
- [5] N. Dziengel, M. Seiffert, M. Ziegert, S. Adler, S. Pfeiffer, and J. Schiller, “Deployment and Evaluation of a Fully Applicable Distributed Event Detection System in Wireless Sensor Networks,” *Ad Hoc Networks, Elsevier*, Sep. 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.adhoc.2015.08.017>
- [6] K. Peter, “Meilenstein bei der Produktion von MEMS-Sensoren,” *telematik-markt.de*, Feb. 2015. [Online]. Available: <http://telematik-markt.de/telematik/meilenstein-bei-der-produktion-von-mems-sensoren>
- [7] T. Rault, A. Bouabdallah, and Y. Challal, “Energy Efficiency in Wireless Sensor Networks: A Top-Down Survey,” *Computer Networks, Elsevier*, vol. 67, pp. 104–122, Jul. 2014. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S1389128614001418>
- [8] J. Yick, B. Mukherjee, and D. Ghosal, “Wireless Sensor Network Survey,” *Computer Networks, Elsevier*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S1389128608001254>
- [9] N. Dziengel, M. Ziegert, S. Adler, Z. Kasmi, and J. Schiller, “Herausforderungen der Verteilten Ereigniserkennung in Drahtlosen Sensornetzen,” in *Mikrosystemtechnik Kongress 2011*, ser. MST. Baiona: VDI VDE IT, 2011.
- [10] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, Nov. 2000.
- [11] H. Niemann, *Klassifikation von Mustern*, 2nd ed. Springer, Jul. 1983.

- [12] A. Webb and K. Copsey, *Statistical Pattern Recognition*. Wiley, 2011. [Online]. Available: <http://books.google.de/books?id=WpV9Xt-h3O0C>
- [13] B. V. Dasarathy, "Sensor Fusion Potential Exploitation-Innovative Architectures and Illustrative Applications," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 24–38, 1997. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=554206
- [14] Y. Kim, J. Kang, D. Kim, E. Kim, P. K. Chong, and S. Seo, "Design of a Fence Surveillance System Based on Wireless Sensor Networks," in *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, ser. Autonomics '08. Brussels, Belgium, Belgium: ICST, 2008, pp. 4:1–4:7. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1487652.1487656>
- [15] A. Yousefi, S. Member, A. Dibazar, and T. Berger, "Intelligent Fence Intrusion Detection System: Detection of Intentional Fence Breaching and Recognition of Fence Climbing," in *Proceedings of the Conference on Technologies for Homeland Security*, ser. HST. IEEE, 2008, pp. 620–625.
- [16] T. Z. Redhwan, M. Chowdhury, and H. A. Rahman, "A Neyman-Pearson approach to the development of low cost earthquake detection and damage mitigation system using sensor fusion," in *Proceedings of the 21st International Conference on Electronics, Circuits and Systems*, ser. ICECS. IEEE, 2014, pp. 191–194. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7049954
- [17] K. Kapitanova, S. H. Son, and K.-D. Kang, "Using Fuzzy Logic for Robust Event Detection in Wireless Sensor Networks," *Ad Hoc Networks, Elsevier*, vol. 10, no. 4, pp. 709–722, Jun. 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.adhoc.2011.06.008>
- [18] K. B. Hein, L. B. Hormann, and R. Weiss, "Using a leaky bucket counter as an advanced threshold mechanism for event detection in wireless sensor networks," in *Proceedings of the 10th Workshop on Intelligent Solutions in Embedded Systems*, ser. WISES. IEEE, 2012, pp. 51–56. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6273604
- [19] S. Zoller, M. Wachtel, F. Knapp, and R. Steinmetz, "Going all the way-Detecting and transmitting events with wireless sensor networks in logistics," in *Proceedings of the 38th Conference on Local Computer Networks Workshops*, ser. LCN Workshops. IEEE, 2013, pp. 39–47. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6758496
- [20] J. Gupchup, A. Terzis, R. C. Burns, and A. S. Szalay, "Model-Based Event Detection in Wireless Sensor Networks," *Computing Research Repository (CoRR)*, vol. abs/0901.3923, 2009. [Online]. Available: <http://arxiv.org/abs/0901.3923>
- [21] A. Khelil, F. K. Shaikh, B. Ayari, and N. Suri, "MWM: A Map-based World Model for Wireless Sensor Networks," in *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, ser. Autonomics '08. Brussels, Belgium, Belgium: ICST, 2008, pp. 5:1–5:10. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1487652.1487657>
- [22] Y. Li, Z. Wang, and Y. Song, "Wireless Sensor Network Design for Wildfire Monitoring," in *6th World Congress on Intelligent Control and Automation*, ser. WCICA. Dalian, China: IEEE, 2006. [Online]. Available: <http://dx.doi.org/10.1109/WCICA.2006.1712372>

- [23] C. Bo, Z. Peng, Z. Da, and C. Junliang, “The Complex Alarming Event Detecting and Disposal Processing Approach for Coal Mine Safety Using Wireless Sensor Network,” *International Journal of Distributed Sensor Networks*, Hindawi, vol. 2012, pp. 1–12, 2012. [Online]. Available: <http://www.hindawi.com/journals/ijdsn/2012/280576/>
- [24] C. Yang, J. Cao, X. Liu, L. Chen, and D. Chen, “A high quality event capture scheme for WSN-based structural health monitoring,” in *Proceedings of the Global Communications Conference*, ser. GLOBECOM. IEEE, 2012, pp. 622–627. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6503182
- [25] T. T. T. Zan, H. B. Lim, K.-J. Wong, A. J. Whittle, and B.-S. Lee, “Event Detection and Localization in Urban Water Distribution Network,” *Sensors Journal, IEEE*, vol. 14, no. 12, pp. 4134–4142, Dec. 2014. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6901220>
- [26] F. Viani, F. Robol, M. Salucci, E. Giarola, S. De Vigili, M. Rocca, F. Boldrini, G. Benedetti, and A. Massa, “WSN-based early alert system for preventing wildlife-vehicle collisions in alps regions-From the laboratory test to the real-world implementation,” in *Proceedings of the 7th European Conference on Antennas and Propagation*, ser. EuCAP. IEEE, 2013, pp. 1913–1916. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6546622
- [27] J. Lauterjung, U. Münch, and A. Rudloff, “The Challenge of Installing a Tsunami Early Warning System in the Vicinity of the Sunda Arc, Indonesia,” *Natural Hazards and Earth System Sciences, Copernicus Publications*, vol. 10, no. 4, pp. 641–646, 2010. [Online]. Available: <http://www.nat-hazards-earth-syst-sci.net/10/641/2010/nhess-10-641-2010.pdf>
- [28] B. Deosarkar, N. Yadav, and R. Yadav, “Clusterhead Selection in Clustering Algorithms for Wireless Sensor Networks: A Survey,” in *Proceedings of the International Conference on Computing, Communication and Networking*, ser. ICCCN. IEEE, Dec 2008, pp. 1–8.
- [29] X. Liu, “A Survey on Clustering Routing Protocols in Wireless Sensor Networks,” *Sensors*, vol. 12, no. 12, pp. 11 113–11 153, Aug. 2012. [Online]. Available: <http://www.mdpi.com/1424-8220/12/8/11113/>
- [30] H. Ghasemzadeh, V. Loseu, and R. Jafari, “Collaborative Signal Processing for Action Recognition in Body Sensor Networks: A Distributed Classification Algorithm Using Motion Transcripts,” in *Proceedings of the 9th International Conference on Information Processing in Sensor Networks*, ser. IPSN. Stockholm, Sweden: ACM/IEEE, Apr. 2010. [Online]. Available: <http://doi.acm.org/10.1145/1791212.1791242>
- [31] J. Radak, B. Ducourthial, and V. Cherfaoui, “Early detection of dangerous events on the road using distributed data fusion,” in *Proceedings of the Vehicular Networking Conference*, ser. VNC. IEEE, 2014, pp. 17–24. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7013304
- [32] J. Yin, D. H. Hu, and Q. Yang, “Spatio-temporal Event Detection Using Dynamic Conditional Random Fields,” in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, ser. IJCAI. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1321–1326. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1661445.1661657>

- [33] M. Waelchli, P. Skoczylas, M. Meer, and T. Braun, "Distributed Event Localization and Tracking with Wireless Sensors," in *Proceedings of the 5th International Conference on Wired/Wireless Internet Communications*, ser. WWIC. Coimbra, Portugal: ACM, May 2007, pp. 247–258.
- [34] H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao, "Target Classification and Localization in Habitat Monitoring," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, ser. ICASSP. Hong Kong, China: IEEE, Apr. 2003.
- [35] J. W. Ko and Y.-H. Choi, "A Grid-Based Distributed Event Detection Scheme for Wireless Sensor Networks," *Sensors*, vol. 11, no. 11, pp. 10048–10062, 2011. [Online]. Available: <http://www.mdpi.com/1424-8220/11/11/10048>
- [36] F. Martincic and L. Schwiebert, "Distributed Event Detection in Sensor Networks," in *Proceedings of the International Conference on Systems and Networks Communications*, ser. ICSNC. Tahiti, French Polynesia: IEEE, Oct. 2006.
- [37] K.-X. Thuc and K. Insoo, "A Collaborative Event Detection Scheme Using Fuzzy Logic in Clustered Wireless Sensor Networks," *{AEU} - International Journal of Electronics and Communications*, vol. 65, no. 5, pp. 485 – 488, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1434841110001445>
- [38] M. Li and Y. Liu, "Underground Coal Mine Monitoring with Wireless Sensor Networks," *Transactions on Sensor Networks, ACM*, vol. 5, pp. 10:1–10:29, Apr. 2009. [Online]. Available: <http://doi.acm.org/10.1145/1498915.1498916>
- [39] Y. Li, C. Ai, C. Vu, Y. Pan, and R. Beyah, "Delay-Bounded and Energy-Efficient Composite Event Monitoring in Heterogeneous Wireless Sensor Networks," *Transactions on Parallel and Distributed Systems, IEEE*, vol. 21, no. 9, pp. 1373–1385, Sep. 2010. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5204076>
- [40] O. Vargas-Fallas and F. J. Torres-Rojas, "Distributed detection of debris flows on a wireless sensor network," in *Proceedings of the International Work Conference on Bio-inspired Intelligence*, ser. IWOBI. IEEE, 2014, pp. 111–118. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6913948
- [41] M. Zoumboulakis and G. Roussos, "Escalation: Complex Event Detection in Extremely Resource-Constrained Wireless Sensor Networks," *Mobile Networks and Applications, Springer-Verlag New York, Inc.*, vol. 16, no. 2, pp. 194–213, Apr. 2011. [Online]. Available: <http://dx.doi.org/10.1007/s11036-010-0268-0>
- [42] A. Y. Yang, S. Iyengar, S. Sastry, R. Bajcsy, P. Kuryloski, and R. Jafari, "Distributed segmentation and classification of human actions using a wearable motion sensor network," in *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, ser. CVPRW. IEEE, 2008, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4563176
- [43] D. M. Doolin and N. Sitar, "Wireless Sensors for Wildfire Monitoring," in *Proceedings of the SPIE Symposium on Smart Structures & Materials / NDE '05*, San Diego, CA, USA, Mar. 2005.

- [44] M. F. Duarte and Y. H. Hu, "Vehicle Classification in Distributed Sensor Networks," *Journal of Parallel and Distributed Computing, Elsevier*, vol. 64, no. 7, Jul. 2004.
- [45] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh, "Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, ser. SenSys. San Diego, CA, USA: ACM, Nov. 2005.
- [46] A. Tavakoli, J. Zhang, and S. H. Son, "Group-Based Event Detection in Undersea Sensor Networks," in *Proceedings of the 2nd International Workshop on Networked Sensing Systems*, ser. INSS. San Diego, CA, USA: ACM, Jun. 2005.
- [47] M. Moradi, J. Ghaisari, J. Askari, and M. Gorji, "A New Method for Detection of a Distributed Event in Wireless Sensor Networks," in *Proceedings of the 19th Iranian Conference on Electrical Engineering*, ser. ICEE. IEEE, 2011, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5955565
- [48] H. Malazi, K. Zamanifar, and S. Dulman, "FED: Fuzzy Event Detection model for Wireless Sensor Networks," *International Journal of Wireless & Mobile Networks, AIRCC*, vol. 3, no. 6, pp. 29–45, Dec. 2011. [Online]. Available: <http://www.airccse.org/journal/jwmn/1211wmn03.pdf>
- [49] S. Zhang, H. Chen, Q. Zhu, and J. Jia, "A Fuzzy-Decision Based Approach for Composite Event Detection in Wireless Sensor Networks," *The Scientific World Journal, Hindawi*, vol. 2014, pp. 1–20, 2014. [Online]. Available: <http://www.hindawi.com/journals/tswj/2014/816892/>
- [50] C.-O. Hong and Y.-H. Choi, "Proximity-Based Robust Event Detection in Wireless Sensor Networks," *International Journal of Distributed Sensor Networks, Hindawi*, vol. 2014, pp. 1–7, 2014. [Online]. Available: <http://www.hindawi.com/journals/ijdsn/2014/632397/>
- [51] A. V. U. Phani Kumar, A. M. Reddy V, and D. Janakiram, "Distributed Collaboration for Event Detection in Wireless Sensor Networks," in *Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing*, ser. MPAC '05. New York, NY, USA: ACM, 2005, pp. 1–8. [Online]. Available: <http://doi.acm.org/10.1145/1101480.1101491>
- [52] D. Schuldhuis, H. Leutheuser, and B. M. Eskofier, "Classification of daily life activities by decision level fusion of inertial sensor data," in *Proceedings of the 8th International Conference on Body Area Networks*. ICST, 2013, pp. 77–82. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2555332>
- [53] Z. Hao, Z. Zhang, and H.-C. Chao, "A Cluster-Based Fuzzy Fusion Algorithm for Event Detection in Heterogeneous Wireless Sensor Networks," *Journal of Sensors, Hindawi*, 2015. [Online]. Available: <http://www.hindawi.com/journals/js/aa/641235/>
- [54] F. Wang, J. Liu, and L. Sun, "Ambient Data Collection with Wireless Sensor Networks," *EURASIP Journal on Wireless Communications and Networking, Springer*, vol. 2010, pp. 4:1–4:10, February 2010.

- [55] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller, "A System for Distributed Event Detection in Wireless Sensor Networks," in *Proceedings of the 9th International Conference on Information Processing in Sensor Networks*, ser. IPSN. Stockholm, Sweden: ACM/IEEE, Apr. 2010.
- [56] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network Aggregation Techniques for Wireless Sensor Networks: A Survey," *Wireless Communications, IEEE*, vol. 14, no. 2, pp. 70–87, April 2007.
- [57] R. V. Kulkarni, A. Föster, and G. K. Venayagamoorthy, "Computational Intelligence in Wireless Sensor Networks: A Survey," *Communications Surveys and Tutorials, IEEE*, vol. 13, no. 1, pp. 68–96, Jan. 2011. [Online]. Available: <http://dx.doi.org/10.1109/SURV.2011.040310.00002>
- [58] B. D. Ripley and N. L. Hjort, *Pattern Recognition and Neural Networks*, 1st ed. New York, NY, USA: Cambridge University Press, 1995.
- [59] F. Schneider, "Enhancement of an Event Detection System for Wireless Sensor Networks with an Optimized Similarity Measure." Master's thesis: Department of Mathematics and Computer Science, Freie Universität Berlin, 2014.
- [60] M. Seiffert, N. Dziengel, M. Ziegert, R. Kerz, and J. Schiller, "Towards Motion Characterization and Assessment Within a Wireless Body Area Network," in *Internet and Distributed Computing Systems*, ser. Lecture Notes in Computer Science, G. D. Fatta, G. Fortino, W. Li, M. Pathan, F. Stahl, and A. Guerrieri, Eds. Springer International Publishing, Sep. 2015, no. 9258, pp. 63–74. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-23237-9_7
- [61] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," Tech. Rep. 8, 1966.
- [62] G. Navarro, "A Guided Tour to Approximate String Matching," *Computing Surveys, ACM*, vol. 33, no. 1, pp. 31–88, Mar. 2001. [Online]. Available: <http://doi.acm.org/10.1145/375360.375365>
- [63] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
- [64] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Morgan Kaufmann, Feb. 2011.
- [65] F. Hermans, N. Dziengel, and J. Schiller, "Quality Estimation Based Data Fusion in Wireless Sensor Networks," in *Proceedings of the 6th International Conference on Mobile Adhoc and Sensor Systems*, ser. MASS. IEEE, 2009, pp. 1068–1070.
- [66] B. Ojeda-Magaña, R. Ruelas, A. Corona Nakamura, M., W. Carr Finch, D., and L. Gómez-Barba, "Pattern Recognition in Numerical Data Sets and Color Images through the Typicality Based on the GKPFM Clustering Algorithm," *Mathematical Problems in Engineering, Hindawi*, vol. 2013, no. 716753, 2013.

- [67] I. Rish, "An Empirical Study of the naive Bayes Classifier," in *Workshop on Empirical Methods in Artificial Intelligence*, ser. IJCAI, vol. 3. IBM New York, 2001, pp. 41–46. [Online]. Available: <http://www.cc.gatech.edu/~isbell/reading/papers/Rish.pdf>
- [68] G. F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 5th ed. Harlow, England ; New York: Addison-Wesley, 2005.
- [69] T. Cover and P. Hart, "Nearest Neighbor Pattern Classification," *Information Theory, IEEE Transactions*, vol. 13, no. 1, pp. 21–27, January 1967.
- [70] N. Dziengel, G. Wittenburg, and J. Schiller, "Towards Distributed Event Detection in Wireless Sensor Networks," in *Adjunct Proceedings of the 4th International Conference on Distributed Computing in Sensor Systems*, ser. DCOSS. Santorini Island, Greece: IEEE/ACM, Jun. 2008. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.172.2015>
- [71] A. K. Pat, P. Langley, K. Wagstaff, and J. Yoo, "Generalized Clustering, Supervised Learning, and Data Assignment," in *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2001.
- [72] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou, C. Hayward, I. Rudan, H. Campbell, A. F. Wright, J. F. Wilson, F. Agakov, P. Navarro, and C. S. Haley, "Application of High-Dimensional Feature Selection: Evaluation for Genomic Prediction in Man," *Scientific Reports*, vol. 5, p. 10312, May 2015. [Online]. Available: <http://www.nature.com/doifinder/10.1038/srep10312>
- [73] P. Langley and others, *Selection of Relevant Features in Machine Learning*. Defense Technical Information Center, 1994. [Online]. Available: <http://www.aaii.org/Papers/Symposia/Fall/1994/FS-94-02/FS94-02-034.pdf>
- [74] R. Kohavi, "Wrappers for Performance Enhancement and Oblivious Decision Graphs," Ph.D. dissertation, Citeseer, 1995. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.2725&rep=rep1&type=pdf>
- [75] C. Wartenburger, "Experimentelle Analyse Verteilter Ereigniserkennung in Sensornetzen." Diploma thesis: Department of Mathematics and Computer Science, Freie Universität Berlin, 2008.
- [76] M. Seiffert, "A Model Driven Classifier for Distributed Event Detection Systems in Wireless Sensor Networks." Diploma thesis: Department of Mathematics and Computer Science, Freie Universität Berlin, 2012.
- [77] Y. Anzai, *Pattern Recognition and Machine Learning*. Academic Press, 1992.
- [78] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy Conservation in Wireless Sensor Networks: A Survey," *Ad Hoc Networks, Elsevier*, vol. 7, no. 3, pp. 537–568, May 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.adhoc.2008.06.003>
- [79] G. Wittenburg, N. Dziengel, and J. Schiller, "Demo Abstract: In-network Training and Distributed Event Detection in Wireless Sensor Networks," in *Proceedings of the 6th Conference on Embedded Network Sensor Systems*, ser. SenSys. New York, NY, USA: ACM, 2008, pp. 387–388. [Online]. Available: <http://doi.acm.org/10.1145/1460412.1460465>

- [80] S. Aksoy and R. M. Haralick, "Feature Normalization and Likelihood-Based Similarity Measures for Image Retrieval," *Pattern Recognition Letters, Elsevier*, vol. 22, pp. 563–582, 2001.
- [81] J. Cooley and J. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.
- [82] "Decimation-in-time (DIT) Radix-2 FFT," Jones, Douglas L., Illinois, 1992. [Online]. Available: <http://cnx.org/content/m12016/latest/>.
- [83] S. Stapelfeld, "Evaluierung der Einsatzfähigkeit eines Drahtlosen Sensornetzes in der Brückenüberwachung." Master's thesis: Department of Mathematics and Computer Science, Freie Universität Berlin, 2012.
- [84] "CMSIS MCU Software Standard v3.01," ARM, Cambridge. [Online]. Available: <https://silver.arm.com/browse/CMSIS>
- [85] R. Wenzel, "Evaluation der DSP Funktionalitäten des Cortex M4 Prozessors für die digitale Signalverarbeitung in drahtlosen Sensornetzwerken." Bachelor's thesis: Department of Mathematics and Computer Science, Freie Universität Berlin, 2014.
- [86] N. Dziengel, M. Seiffert, and S. Kolano, "A Method and a Feedback System for the Assessment of Motions of a Versatile System," PCT/EP 2015/069767, Nov. 2014, EP Patent 14 193 446.3.
- [87] N. Dziengel, M. Ziegert, M. Seiffert, J. Schiller, and G. Wittenburg, "Integration of Distributed Event Detection in Wireless Motion-Based Training Devices," in *Proceedings of the 1st International Conference on Consumer Electronics*, ser. ICCE. Berlin, Germany: IEEE, 2011. [Online]. Available: <http://dx.doi.org/10.1109/ICCE-Berlin.2011.6031860>
- [88] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Hoboken, NJ: Wiley-Interscience, 2004.
- [89] D. L. Hall and S. A. H. McMullen, *Mathematical Techniques in Multisensor Data Fusion (Artech House Information Warfare Library)*. Norwood, MA, USA: Artech House, Inc., 2004.
- [90] P. Pudil, J. Novovičová, and J. Kittler, "Floating Search Methods in Feature Selection," *Pattern Recognition Letters, Elsevier Science Inc.*, vol. 15, no. 11, pp. 1119–1125, Nov. 1994. [Online]. Available: [http://dx.doi.org/10.1016/0167-8655\(94\)90127-9](http://dx.doi.org/10.1016/0167-8655(94)90127-9)
- [91] H. Peng, F. Long, and C. Ding, "Feature Selection Based on Mutual Information Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1226–1238, 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1453511
- [92] P. Pudil, F. J. Ferri, J. Novovicova, and J. Kittler, "Floating Search Methods for Feature Selection with Nonmonotonic Criterion Functions," in *Proceedings of the 12th International Conference on Pattern Recognition*, ser. ICPR. IEEE, 1994, pp. 279–283.
- [93] M. Gütlein, E. Frank, M. Hall, and A. Karwath, "Large Scale Attribute Selection Using Wrappers," 2009.

- [94] G. Wittenburg, N. Dziengel, and C. Wartenburger, "Method and Sensor Network for Attribute Selection for an Event Recognition," European patent EP 2,392,120 B1 (PCT/EP2010/050920, WO 2010/086325 (05.08.2010 Gazette 2010/31)), Mar. 2014, EP Patent 2 392 120 (B1). [Online]. Available: <https://depatisnet.dpma.de/DepatisNet/depatisnet?action=pdf&docid=EP000002392120B1>
- [95] A. Jain and D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Performance," *Transactions on Pattern Analysis and Machine Intelligence, IEEE*, vol. 19, no. 2, pp. 153–158, 1997. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=574797
- [96] R. Kohavi and G. H. John, "Wrappers for Feature Subset Selection," *Artificial Intelligence, Elsevier*, vol. 97, no. 1–2, pp. 273 – 324, 1997. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S000437029700043X>
- [97] R. Kohavi, "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, ser. IJCAI, San Mateo, USA, 1995.
- [98] M. H. Quenouille, "Approximate Tests of Correlation in Time-Series 3," vol. 45. Cambridge University Press, 7 1949, pp. 483–484.
- [99] G. Wittenburg, N. Dziengel, and C. Wartenburger, "Verfahren und Sensornetz zur Merkmalsauswahl für eine Ereigniserkennung," German patent DE 10 2009 006 560 B4 (WO 2010/086325 A1, US 20120036242 A1), Jun. 2011. [Online]. Available: <http://depatisnet.dpma.de/DepatisNet/depatisnet?action=pdf&docid=DE102009006560B4>
- [100] —, "Method and Sensor Network for Attribute Selection for an Event Recognition," Feb. 2012, US Patent App. 13/146,623. [Online]. Available: <https://depatisnet.dpma.de/DepatisNet/depatisnet?action=pdf&docid=US020120036242A1>
- [101] N. Dziengel, N. Schmittberger, J. Schiller, and M. Guenes, "Secure Communications in Event-Driven Wireless Sensor Networks," in *Proceedings of the 3rd International Symposium on Sensor Networks and Applications (SNA) in conjunction with the Twenty-Fourth International Conference on Computer Applications in Industry and Engineering (CAINE)*, 2011.
- [102] N. Dziengel, M. Ziegert, S. Adler, Z. Kasmı, S. Pfeiffer, and J. Schiller, "Energy-Aware Distributed Fence Surveillance for Wireless Sensor Networks," in *Proceedings of the 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, ser. ISSNIP. IEEE, Dec 2011, pp. 353–358.
- [103] R. D. Maesschalck, D. Jouan-Rimbaud, and D. Massart, "The mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems, Elsevier*, vol. 50, no. 1, pp. 1 – 18, 2000. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0169743999000477>
- [104] L. Devroye, L. Györfı, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.
- [105] P. C. Mahalanobis, "On the Generalised Distance in Statistics," in *Proceedings National Institute of Science, India*, vol. 2, no. 1. Springer, Apr. 1936, pp. 49–55. [Online]. Available: http://www.new.dli.ernet.in/rawdataupload/upload/insa/INSA_1/20006193_49.pdf

- [106] D. Disatnik and S. Benninga, “Shrinking the covariance-matrix.” *The Journal of Portfolio Management, LLC*, vol. 33, no. 4, pp. 55–63, 2007.
- [107] J. Schiller, *Mobile Communications*, 2nd ed. Addison Wesley, 2003.
- [108] M. Kolar and E. P. Xing, “Estimating Sparse Precision Matrices from Data with Missing Values,” 2012.
- [109] R. Little and D. Rubin, *Statistical Analysis With Missing Data*, ser. Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series. Wiley, 1987. [Online]. Available: <https://books.google.de/books?id=w40QAQAIAAJ>
- [110] T. Schneider, “Analysis of Incomplete Climate Data: Estimation of Mean Values and Covariance Matrices and Imputation of Missing Values,” *Journal of Climate*, vol. 14, no. 5, Mar. 2001. [Online]. Available: <http://authors.library.caltech.edu/3973/1/SCHNjc01b.pdf>
- [111] A. T. Hoang and M. Motani, “Collaborative Broadcasting and Compression in Cluster-Based Wireless Sensor Networks,” *Transactions on Sensor Networks, ACM*, vol. 3, no. 3, Aug. 2007. [Online]. Available: <http://doi.acm.org/10.1145/1267060.1267065>
- [112] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, “Information Fusion for Wireless Sensor Networks: Methods, Models, and Classifications,” *ACM Computer Surveys (CSUR)*, vol. 39, no. 3, Sep. 2007. [Online]. Available: <http://doi.acm.org/10.1145/1267070.1267073>
- [113] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, “Practical Data Compression in Wireless Sensor Networks: A Survey,” *Journal of Network and Computer Applications, Elsevier*, vol. 35, no. 1, pp. 37–59, Jan. 2012. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S1084804511000555>
- [114] M. E. Assi, A. Ghaddar, S. Tawbi, and G. Fadi, “Resource-Efficient Floating-Point Data Compression Using MAS in WSN,” *International Journal of Ad hoc, Sensor & Ubiquitous Computing, AIRCC*, vol. 4, no. 5, pp. 13–28, Oct. 2013. [Online]. Available: <http://www.aircse.org/journal/ijasuc/papers/4513ijasuc02.pdf>
- [115] G. Campobello, O. Giordano, A. Segreto, and S. Serrano, “Comparison of Local Lossless Compression Algorithms for Wireless Sensor Networks,” *Journal of Network and Computer Applications, Elsevier*, vol. 47, pp. 23–31, Jan. 2015. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S1084804514002203>
- [116] J. Wilke and C. Haas, “Energy-efficiency of Concast Communication in Wireless Sensor Networks,” in *10th Annual Conference on Wireless On-demand Network Systems and Services*, ser. WONS. Banff, AB, Canada: IEEE, March 2013, pp. 34–38. [Online]. Available: <http://dx.doi.org/10.1109/WONS.2013.6578318>
- [117] N. Dziengel, G. Wittenburg, S. Adler, Z. Kasmi, M. Ziegert, and J. H. Schiller, *Event Detection in Wireless Sensor Networks*. Taylor & Francis LLC, CRC Press, 2012.
- [118] L. Mottola and G. P. Picco, “Logical Neighborhoods: A Programming Abstraction for Wireless Sensor Networks,” in *Distributed Computing in Sensor Systems*. Springer, 2006, pp. 150–168. [Online]. Available: http://link.springer.com/chapter/10.1007/11776178_10

- [119] S. Lindsey and C. S. Raghavendra, "PEGASIS: Power-Efficient Gathering in Sensor Information Systems," in *Aerospace conference proceedings, 2002. IEEE*, vol. 3. IEEE, 2002, pp. 3–1125. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1035242
- [120] T. Tsvetkov, "RPL: IPv6 Routing Protocol for Low Power and Lossy Networks," *Sensor Nodes—Operation, Network and Application (SN)*, vol. 59, p. 2, 2011. [Online]. Available: <https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2011-07-1.pdf#page=67>
- [121] A. Jamil, D. J. Parish, R. C. Phan, I. Phillips, J. Whitley, and G. Oikonomou, "Maximise Unsafe Path Routing Protocol for Forest Fire Monitoring System Using Wireless Sensor Networks," in *Proceedings of the 3rd International Conference on Networked Embedded Systems for Every Application*, ser. NESEA. IEEE, 2012, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6474018
- [122] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller, "Max-margin Classification of Data with Absent Features," *Journal of Machine Learning Research*, vol. 9, pp. 1–21, Jun. 2008. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1390681.1390682>
- [123] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern Classification with Missing Data: A Review," *Neural Compututer Applications, Springer*, vol. 19, no. 2, pp. 263–282, Mar. 2010. [Online]. Available: <http://dx.doi.org/10.1007/s00521-009-0295-6>
- [124] N. Dziengel, M. Ziegert, Z. Kasmi, F. Hermans, S. Adler, G. Wittenburg, and J. Schiller, "A Platform for Distributed Event Detection in Wireless Sensor Networks," in *Proceedings of the 1st International Workshop on Networks of Cooperating Objects*, ser. CONET, Stockholm, Sweden, Apr. 2010.
- [125] G. Wittenburg, N. Dziengel, S. Adler, Z. Kasmi, M. Ziegert, and J. H. Schiller, "Cooperative Event Detection in Wireless Sensor Networks." *IEEE Communications Magazine*, vol. 50, no. 12, pp. 124–131, 2012.
- [126] M. Baar, H. Will, B. Blywis, T. Hillebrandt, A. Liers, G. Wittenburg, and J. Schiller, "The ScatterWeb MSB-A2 Platform for Wireless Sensor Networks," Freie Universität Berlin, Berlin, Germany, Tech. Rep. B 08-15, Sep. 2008.
- [127] M. Ziegert, N. Dziengel, M. Seiffert, S. Pfeiffer, and J. Schiller, "A Developer and a Reference Board for Distributed Motion Evaluation in Wireless Sensor Networks," in *Proceedings of the 1st International Conference on Industrial Technology*, ser. ICIT. Seville, Spain: IEEE, 2015.
- [128] "SMB380 Datasheet," Bosch, Reutlingen, Germany. [Online]. Available: http://www.bosch.de/start/content/language2/downloads/news_0505_1_Sensortec_Datasheet_en.pdf
- [129] InvenSense Inc., "MPU-9150 Product Specification Revision 4.3," 2013.
- [130] "LPC2387 Product Datasheet Ref. 5.1," NXP, Eindhoven, Netherlands, 2013. [Online]. Available: <http://www.nxp.com/documents/datasheet/LPC2387.pdf>

- [131] “CC1101 Datasheet,” Texas Instruments, Texas, USA. [Online]. Available: <http://focus.ti.com/lit/ds/swrs061f/swrs061f.pdf>
- [132] “SHT1x Datasheet ver. 5,” Sensirion, 2011.
- [133] “Molex Series 503500 Datasheet Rev. 1,” Molex, Japan, 2010.
- [134] “LTC4150 Coulomb Counter/ Battery Gas Gauge,” Linear Technology, California, Milpitas, 2010.
- [135] “FSA03 Application Notes ver. 1.0.5,” Falcom, 2011.
- [136] “FT232R USB UART IC Datasheet Version 2.11,” Future Technology Devices International Limited, 2010.
- [137] “STM32F415xx, STM32F417xx Datasheet Rev 5, DocID022063,” STMicroelectronics, 2015.
- [138] “A1101R08C 1101 Series,” Anaren Integrated Radio, 2010.
- [139] “WS2812B Intelligent Control LED Integrated Light Source, Datasheet,” Worldsemi, 2013.
- [140] T. Hillebrandt, “Untersuchung und Simulation des Zeit- und Energieverhaltens eines MSB430-H Sensornetzwerkes,” Master’s thesis, Department of Mathematics and Computer Science, Freie Universität Berlin, December 2007.
- [141] C. J. Doppler, “Über das farbige Licht der Doppelsterne und einiger anderer Gestirne des Himmels,” *Psychological Bulletin, Nabu Press*, vol. 5, no. 2, pp. 465–482, 1843.
- [142] G. Wittenburg, K. Terfloth, F. L. Villafuerte, T. Naumowicz, H. Ritter, and J. Schiller, “Fence Monitoring - Experimental Evaluation of a Use Case for Wireless Sensor Networks,” in *Proceedings of the 4th European Conference on Wireless Sensor Networks*, ser. EWSN, Delft, The Netherlands, Jan. 2007. [Online]. Available: <http://portal.acm.org/citation.cfm?id=1758141>
- [143] N. Dziengel, M. Ziegert, S. Adler, Z. Kasmi, M. Seiffert, and J. Schiller, “Verteilte Ereigniserkennung in Drahtlosen Sensornetzen überwacht Zaun-, Reha- und Sportanwendungen,” *Fachzeitschrift für Aufbau- und Verbindungstechnik in der Elektronik PLUS, Leuze Verlag*, vol. 14, no. 1, pp. 629–635, Jan. 2011.
- [144] K. Terfloth, G. Wittenburg, and J. Schiller, “FACTS - A Rule-Based Middleware Architecture for Wireless Sensor Networks,” in *Proceedings of the 1st International Conference on COMMunication System softWARE and MiddlewaRE*, ser. COMSWARE. New Delhi, India: IEEE, Jan. 2006. [Online]. Available: <http://dx.doi.org/10.1109/COMSWA.2006.1665162>
- [145] J. Schiller, A. Liers, and H. Ritter, “ScatterWeb: A Wireless Sensornet Platform for Research and Teaching,” *Computer Communications*, vol. 28, pp. 1545–1551, Apr. 2005.
- [146] A. LaMarca and E. de Lara, “Location Systems: An Introduction to the Technology Behind Location Awareness,” *Synthesis Lectures on Mobile and Pervasive Computing*, vol. 3, no. 1, pp. 1–122, 2008. [Online]. Available: <http://dx.doi.org/10.2200/S00115ED1V01Y200804MPC004>

- [147] A. W. Salmoni, R. A. Schmidt, and C. B. Walter, “Knowledge of Results and Motor Learning: A Review and Critical Reappraisal.” *Psychological Bulletin*, vol. 95, no. 2, pp. 355–386, 1984.
- [148] “Wii at Nintendo.” [Online]. Available: <http://www.nintendo.com/wii>
- [149] E. A. Heinz, K. Kunze, M. Gruber, D. Bannach, and P. Lukowicz, “Using Real-Time Recognition Tasks in Games of Martial Arts – An Initial Experiment,” in *Proceedings of the 2nd Symposium on Computational Intelligence and Games*, ser. CIG. IEEE, 2006, pp. 98–102.
- [150] I. Spirgi-Gantert and B. Suppé, *FBL Klein-Vogelbach functional kinetics: therapeutische Übungen: [auf DVD: Videos und Übungsblätter]*, 6th ed., ser. Physiotherapie. Berlin ;Heidelberg ;New York, NY: Springer Medizin, 2012.
- [151] A. Schwarzenegger and B. Dobbins, “The New Encyclopedia of Modern Bodybuilding,” *Simon & Schuster*, 2012.
- [152] W. Gary, “Quantified Self, self knowledge through numbers,” *QS Quantified Self*, Sep. 2015. [Online]. Available: <http://quantifiedself.com/>
- [153] J. Kim, A. Swartz, J. P. Lynch, J.-J. Lee, and C.-G. Lee, “Rapid-to-Deploy Reconfigurable Wireless Structural Monitoring Systems using Extended-Range Wireless Sensors,” *Smart Structures and Systems, Techno-Press*, vol. 6, no. 5-6, pp. 505–524, 2010.
- [154] M. Kurata, J. Kim, Y. Zhang, J. P. Lynch, G. Van der Linden, V. Jacob, E. Thometz, P. Hipley, and L.-H. Sheng, “Long-term Assessment of an Autonomous Wireless Structural Health Monitoring System at the new Carquinez Suspension Bridge,” in *SPIE Smart Structures and Materials + Nondestructive Evaluation and Health Monitoring*. International Society for Optics and Photonics, 2011, pp. 798 312–798 312.
- [155] H. Joshi, B. Burlacu, M. Prabhu, S. Yoon, M. Sichitiu, R. Dutta, and S. Rizkalla, “Wireless Structural Health Monitoring System Design, Implementation and Validation,” *Constructed Facilities Lab (CFL), North Carolina State University (NCSU), Raleigh, North Carolina, Research Report RD-06-02*, 2006.
- [156] T. Nagayama, B. Spencer, G. Agha, and K. Mechitov, “Model-Based Data Aggregation for Structural Monitoring Employing Smart Sensors,” in *Proceedings of the 3rd International Conference on Networked Sensing Systems*, ser. INSS. Transducer Research Foundation TRF, May 2006, pp. 203–210.
- [157] S. Su, *Decentralized Damage detection in Civil Infrastructure using Multi-scale Wireless Sensor Networks*. University of Notre Dame, 2011.
- [158] R. A. Swartz and J. P. Lynch, “Damage Characterization of the Z24 Bridge by Transfer Function Pole Migration,” in *Proceedings of the 26th International Modal Analysis Conference*, ser. IMAC XXVI. Orlando, FL, USA: Springer, 2008.
- [159] B. Spencer Jr, S. Cho, and S.-H. Sim, “Wireless Monitoring of Civil Infrastructure Comes of Age,” *Structure Magazine*, vol. 13, 2011.
- [160] S. W. Doebling, C. R. Farrar, M. B. Prime *et al.*, “A Summary Review of Vibration-Based Damage Identification Methods,” *Shock and vibration digest*, vol. 30, no. 2, pp. 91–105, 1998.

-
- [161] T. Kijewski-Correa, M. Haenggi, and P. Antsaklis, “Multi-scale wireless sensor networks for structural health monitoring,” 2005, pp. 16–18.
- [162] T. Kijewski-Correa and S. Su, “BRAIN: A Bivariate Data-Driven Approach to Damage Detection in Multi-Scale Wireless Sensor Networks,” *Journal of Smart Structures and Systems, Technopress*, vol. 5, no. 4, 2009.
- [163] F. Neitzel, W. Niemeier, S. Weisbrich, and M. Lehmann, “Investigation of Low-Cost Accelerometer, Terrestrial Laser Scanner and Ground-Based Radar Interferometer for Vibration Monitoring of Bridges,” in *Proceedings of the 6th European Workshop on Structural Health Monitoring*, ser. EWSHM. DGZfP, pp. 542–551. [Online]. Available: <http://www.ndt.net/article/ewshm2012/papers/fr2c2.pdf>
- [164] C. Handel, “Dynamische Messungen von Eisenbahnbrücken.” Projekt ComTest: Österreichische Forschungsförderungsgesellschaft mbH, 2007. [Online]. Available: http://www.regelplanung.at/B45/Dynamische_Messung_von_Eisenbahnbruecken.pdf
- [165] G. De Roeck, B. Peeters, and J. Maeck, “Dynamic Monitoring of Civil Engineering Structures,” ser. IASS-IACM, Chania, Greece, 2000.
- [166] E. Baccelli, O. Hahm, M. Wählisch, M. Günes, and T. C. Schmidt, “RIOT: One OS to Rule Them All in the IoT,” in *INRIA, Research Report, No. RR-8176*, 2012.