MULTILEVEL OBJECT TRACKING ON BIG GRAPH DATA USING INTERVAL TYPE-2 FUZZY SYSTEMS IN WIRELESS MULTIMEDIA SENSOR NETWORKS

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

BY

CİHAN KÜÇÜKKEÇECİ

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

AUGUST 2020

Approval of the thesis:

MULTILEVEL OBJECT TRACKING ON BIG GRAPH DATA USING INTERVAL TYPE-2 FUZZY SYSTEMS IN WIRELESS MULTIMEDIA SENSOR NETWORKS

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

Prof. Dr. Halil Kalıpçılar Dean, Graduate School of **Natural and Applied Sciences**	________
Prof. Dr. Halit Oğuztüzün Head of Department, **Computer Engineering**	________
Prof. Dr. Adnan Yazıcı Supervisor, **Computer Engineering, METU**	________

Examining Committee Members:

Assoc. Prof. Dr. İsmail Sengör Altıngövde Computer Engineering, METU	________
Prof. Dr. Adnan Yazıcı Computer Engineering, METU	________
Prof. Dr. Murat Koyuncu Information Systems Engineering, Atılım University	________
Assoc. Prof. Dr. Sinan Kalkan Computer Engineering, METU	________
Assoc. Prof. Dr. Tufan Kumbasar Control and Automation Engineering, ITU	________

Date:

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

Name, Surname:	Cihan Küçükkeçeci
Signature :	

ABSTRACT

MULTILEVEL OBJECT TRACKING ON BIG GRAPH DATA USING INTERVAL TYPE-2 FUZZY SYSTEMS IN WIRELESS MULTIMEDIA SENSOR NETWORKS

Küçükkeçeci, Cihan
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Adnan Yazıcı

August 2020, 146 pages

Wireless multimedia sensor networks (WMSN) are the key elements of automation systems applied in different domains from home security to immigrant surveillance at a border station. In most of the applications, sensor data needs to be processed for data analytics. However, the interpretation of raw sensor data and unveiling the information inside remains a challenging issue from many aspects. As the interval of the sensor data is frequent, data needs to be treated as big data because of the volume and velocity. Unfortunately, traditional approaches do not perform well in big data analytics, especially in extracting the complex relationships between data.

In this dissertation, a novel fuzzy object tracking approach which is developed using a big graph data model is proposed by utilization of a multilevel fusion. This approach consists of three main steps: intra-node fusion, inter-node fusion, and object trajectory construction. Intra-node fusion exploits object detection and tracking in each sensor while inter-node fusion uses spatiotemporal data along with neighbor sensors. Then, all trajectories from all sensor nodes are integrated using fuzziness to construct trajectories in the common ground-plane across the wireless multimedia sensor network. Since uncertainty naturally exists in trajectory data, fuzzy logic systems have been studied on the extracted trajectories as well as for further analytics like trajectory prediction and anomaly detection.

A prototype system was implemented and several experiments were conducted to evaluate the performance of the proposed approach with both synthetic and real world datasets. The results show that usage of third-level fusion, in addition to inter-node and intra-node fusions provides significantly better performance for object tracking in WMSN applications. GeoLife Trajectories and Maritime Cadastre datasets were used as input of two different real world use cases to perform experiments, and results validate that interval type-2 fuzzy logic utilization improves performance in both trajectory extraction and analytics.

Keywords: big data analytics, Internet of things, fuzzy logic, nosql databases, graph model, wireless multimedia sensor networks

ÖZ

ÇOKLU ORTAM DUYARGA AĞLARINDA ARALIK TİP-2 BULANIK SİSTEMLER KULLANARAK BÜYÜK ÇİZGE VERİLERDE ÇOKKATMANLI NESNE TAKİBİ

Küçükkeçeci, Cihan
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Adnan Yazıcı

Ağustos 2020 , 146 sayfa

Kablosuz çoklu ortam duyarga ağları (KÇODA), ev güvenliğinden, bir sınır karakolundaki göçmen gözetimine kadar farklı alanlarda uygulanan otomasyon sistemlerinin temel unsurlarıdır. Bu tip uygulamaların birçoğunda, veri analizi için sensör verilerinin işlenmesi gerekir. Ancak, ham sensör verilerinin yorumlanması ve içerisinde barındırdığı bilgilerin açığa çıkarılması birçok açıdan zor bir konudur. Sensör verilerinin ölçüm aralığı sık olduğundan, hacim ve hız nedeniyle verinin büyük veri olarak ele alınması gerekir. Ne yazık ki, geleneksel yaklaşımlar büyük veri analizlerinde, özellikle de veriler arasındaki karmaşık ilişkilerin çıkarılmasında pek de iyi performans göstermezler.

Bu tezde, büyük grafik veri modeli kullanılarak geliştirilen yeni bir bulanık nesne izleme yaklaşımı, çokkatmanlı füzyondan yararlanılarak önerilmektedir. Bu yaklaşım üç ana adımdan oluşur: düğüm-içi füzyon, düğümler-arası füzyon ve nesne yörüngesi oluşturulması. Düğümler-arası füzyon, her sensör düğümde nesne algılama ve izlemeden faydalanırken, düğümler-arası füzyon, komşu sensörler aracılığıyla zamanmekansal verileri kullanır. Daha sonra, tüm sensör düğümlerindeki tüm yörüngeler, kablosuz çoklu ortam duyarga ağı boyunca uzanan ortak yer düzlemindeki yörüngeleri oluşturmak için bulanıklık kullanılarak birleştirilir. Yörünge verilerinde belirsizlik doğası gereği mevcut olduğundan, çıkartılmış yörüngelerin yanı sıra yörünge tahmini ve anomali tespiti gibi diğer analitikler için bulanık mantık sistemleri çalışılmıştır.

Prototip bir sistem geliştirildi ve hem sentetik hem de gerçek dünya veri kümeleriyle, önerilen yaklaşımın performansını değerlendirmek için çeşitli deneyler yapıldı. Sonuçlar göstermektedir ki, düğümler-arası ve düğüm-içi füzyonlara ek olarak üçüncü katman füzyon kullanımı, KÇODA uygulamalarındaki nesne izleme için önemli ölçüde daha iyi performans sağlamaktadır. Deneyleri yapmak için GeoLife Trajectories ve Maritime Cadastre veri kümeleri, farklı gerçek dünya senaryolarına girdi olarak kullanıldı, ve sonuçlar aralık tip-2 bulanık mantık kullanımının hem yörünge çıkarmada hem de analizde performansı arttırdığını doğrulamaktadır.

Anahtar Kelimeler: büyük veri analizi, şeylerin Interneti, bulanık mantık, nosql veritabanları, çizge modeli, kablosuz çoklu ortam duyarga ağları

To my family...

ACKNOWLEDGMENTS

There are many people who are sharing the success and pride of this thesis, and these people were with me whatever it takes to put things on track for both in my thesis and my life. One of them is my supervisor Prof.Dr.Adnan Yazıcı unhesitatingly. I feel sincerely beholden to him for his endless advice, constructive criticisms, and excellent feedback which had crucial effects on putting this dissertation into its current form.

Another one is undoubtedly my family who shared every moment of this journey with me without taking a moment to pause. I sincerely thank each of my family members for supporting and believing in me throughout my life.

And finally, I owe the members of my thesis monitoring committee, the juries of my thesis defense, my friends, my relatives, my colleagues, and everyone who finds something valuable for him/herself in this dissertation a debt of gratitude.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xvi

LIST OF FIGURES xviii

LIST OF ABBREVIATIONS xxi

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem 2

1.2 Contributions 3

1.3 Overview of the Thesis 5

1.4 Declaration 6

2 BACKGROUND AND RELATED WORK 7

2.1 Internet of Things 7

2.2 Wireless Multimedia Sensor Networks 8

2.3 NoSQL Graph Databases 9

2.4 Big Data 10

2.5 Big Data Analytics 11

2.6 Machine Learning Algorithms 12

2.7 Fuzzy Logic System 14

3 GRAPH-BASED DATA MODEL 17

3.1 Overview 18

3.2 The Graph-Based Big Data Model 19

3.3 Graph Database Selection 21

3.4 Model Implementation Over Graph Database 21

3.5 Experimental Work 27

3.5.1 Comparison to Relational Data Model 27

3.5.1.1 Concepts Based Query 29

3.5.1.2 Video Based Query 29

3.5.1.3 Recursive Query 31

3.5.2 Doubling Sensed Raw Data Size 33

3.6 Remarks 33

4 DATA SIMULATION 37

4.1 Reference WSN System 37

4.2 Generic Infrastructure for Simulator 39

4.3 System Architecture 40

4.4 Simulation 42

4.4.1 Network Topology Simulation 42

4.4.2 Data Flow Simulation 44

4.4.3 Scenario Generator 45

4.5 A Case Study: Surveillance Application 47

4.6 Simulator Application 49

4.7 Remarks 52

5 OBJECT TRACKING AND TRAJECTORY ANALYTICS 53

5.1 Overview 53

5.2 Object Tracking 54

5.2.1 Intra-Node Fusion 55

5.2.2 Inter-Node Fusion 56

5.2.3 Rule Engine 56

5.2.3.1 Definitions 59

5.2.3.2 Rule Generation and Implementation 60

5.2.3.3 Sample Scenario Execution 63

5.2.4 Object Trajectory Construction 67

5.3 Interval Type-2 (IT2) Fuzzy Logic Trajectory Analytics 71

5.3.1 Fuzzy Geometry 72

5.3.2 Extending Object Tracking with Fuzziness 74

5.3.3 IT2 Fuzzy Logic based Classification 82

5.3.4 Trajectory Prediction 84

5.3.5 Anomaly Detection 85

5.4 Real World Use Cases 87

5.4.1 GeoLife Dataset 88

5.4.2 Marine Cadastre Dataset 89

5.5 Remarks 92

6 EVALUATION 95

6.1 Multilevel Object Tracking Experiments 95

6.1.1 Synthetic Data 96

6.1.2 Real World Dataset 101

6.2 Fuzzy Logic Experiments 104

6.2.1 Global Trajectory Construction 104

6.2.2 Fuzzy Trajectory Extraction 110

6.2.3 Prediction and Anomaly Detection Analytics 112

6.3 Remarks 115

7 CONCLUSIONS 117

7.1 Discussion 118

7.2 Future Work 119

REFERENCES 121

APPENDICES

A FUZZY RULES 131

A.1 Type-1 Fuzzy Rules 131

A.2 Interval Type-2 Fuzzy Rules 135

B COMPLEXITY ANALYSIS OF ALGORITHMS 139

B.1 Rule Engine Algorithm 140

B.2 Intra-Node Fusion Algorithm 141

B.3 Inter-Node Fusion Algorithm 142

B.4 Object Trajectory Construction Algorithm 143

CURRICULUM VITAE 145

LIST OF TABLES

TABLES

Table 3.1 Compare OrientDB and Neo4j Community Editions 22

Table 3.2 Node and Edge Types 24

Table 3.3 Test Results (Numbers are in milliseconds) 33

Table 4.1 Defined Message Queues 40

Table 4.2 Sample Thresholds To Identify Objects 47

Table 4.3 Simulated Entity Types 50

Table 4.4 Sensed Values at Node(30,10) 51

Table 4.5 Sensed Values at Node(40,10) 51

Table 5.1 Rule Condition Parameter Values for Each Trajectory 67

Table 5.2 Sample Rule Engine Execution Scores for Each Trajectory 67

Table 5.3 Fuzzy-based Rule Engine Parameter Values for Each Trajectory . . . 81

Table 5.4 Fuzzified Input Parameters for Each Trajectory 81

Table 5.5 Output Values of the Fuzzy-based Rule Engine for Each Trajectory . 82

Table 5.6 Mapping from Transportation Mode to Concept Type 90

Table 6.1 Algorithm Scores with Various Parameters 101

Table 6.2 Mapping from Transportation Mode to Concept Type 103

Table 6.3 F-measure Comparison for Concept Types 104

Table 6.4 The Number of Rules Comparison for Type-1 and Interval Type-2 . 106

Table 6.5 Less Number of Rules in IT2 compared to T1 109

Table 6.6 PICP and PINAW evaluation for the IT2 fuzzy algorithm. 116

Table A.1 T1 Fuzzy Rules Used in Intra-Node Fusion (Part 1/2) 132

Table A.2 T1 Fuzzy Rules Used in Intra-Node Fusion (Part 2/2) 133

Table A.3 T1 Fuzzy Rules Used in Inter-Node Fusion 134

Table A.4 IT2 Fuzzy Rules Used in Intra-Node Fusion (Part 1/2) 136

Table A.5 IT2 Fuzzy Rules Used in Intra-Node Fusion (Part 2/2) 137

Table A.6 IT2 Fuzzy Rules Used in Inter-Node Fusion 138

LIST OF FIGURES

FIGURES

Figure 2.1 5 Vs of big data. 10

Figure 2.2 Type-1 fuzzy logic system. 14

Figure 2.3 Interval type-2 fuzzy logic system. 15

Figure 3.1 Graph-based data model. 20

Figure 3.2 Relational Database Schema 28

Figure 3.3 OrientDB query time and simulation duration. 34

Figure 4.1 Reference WMSN System Architecture [1] 38

Figure 4.2 System architecture. 41

Figure 4.3 Simulated wireless sensor network. (a) Grid-based deployment. (b) Random deployment. 43

Figure 4.4 Data flow simulation. 45

Figure 4.5 Scenario generation. (a) Using scenario drawer. (b) Using JSON file. 46

Figure 4.6 Surveillance application simulation. 48

Figure 4.7 Data simulator application. 49

Figure 5.1 Fuzzy logic system. 61

Figure 5.2 Sample fuzzy member functions. 62

Figure 5.3 Rule engine execution on a sample scenario. 64

Figure 5.4 Multilevel fusion step-by-step trajectory detection. (a) Actual trajectory. (b) Intra-node fusion. (c) Inter-node fusion. (d) Trajectory construction. 70

Figure 5.5 An IT2 Fuzzy Logic System. 71

Figure 5.6 A fuzzy point derived from possible positions. 73

Figure 5.7 A fuzzy trajectory representation. 73

Figure 5.8 Object Models for Fuzzy Trajectories. 74

Figure 5.9 Input member functions for IT2 FLS. 80

Figure 5.10 Success rates of classification algorithms in prediction with-/without parameter tuning. 85

Figure 5.11 (a) Trained trajectories for analytics. (b) A sample anomaly in which the object leaves suddenly while moving on Trajectory-3. 87

Figure 5.12 Utilization of real world dataset. 89

Figure 5.13 Sample AIS data exchange representation. 90

Figure 5.14 A sample trajectory from maritime dataset. 93

Figure 6.1 Scenario samples generated using developed simulator. (a) Straight trajectories. (b) Circular trajectories. (c) Zig zag trajectories. (d) Wavy trajectories. 97

Figure 6.2 F-measure scores for both grid-based and randomly deployed nodes. 98

Figure 6.3 Comparison with Kalman filter and Particle filter. (a) Straight trajectory. (b) Wavy trajectory. 99

Figure 6.4 Error rates comparison with Kalman filter and Particle filter. (a) Straight trajectory. (b) Wavy trajectory. 100

Figure 6.5 Object trajectory construction. (a) F-Scores. (b) Precision comparison with inter-node fusion. 102

Figure 6.6 Fuzzy logic usage compared to non-fuzzy approach in trajectory construction. (a) Synthetically generated data. (b) GeoLife Trajectories dataset. (c) Marine Cadastre dataset. 105

Figure 6.7 T-Test results. (a) For Baseline (Rule-Based Engine) and Type-1 Fuzzy Logic. (b) For Type-1 Fuzzy Logic and Interval Type-2 Fuzzy Logic. 110

Figure 6.8 Comparison of representation of regular and fuzzy trajectories. (a) Regular trajectory as a line. (b) Fuzzy trajectory as a polygon. 111

Figure 6.9 F-measure score comparison for proposed IT2 prediction algorithm on synthetically generated data. 112

Figure 6.10 F-measure score comparison for proposed IT2 prediction algorithm on GeoLife Trajectories dataset. 113

Figure 6.11 F-measure score comparison for proposed IT2 prediction algorithm on Marine Cadastre dataset. 114

Figure 6.12 Anomaly detection success rates of fuzzy algorithms on synthetically generated data. 114

Figure B.1 Complexity analysis of Rule Engine algorithm. 140

Figure B.2 Complexity analysis of Intra-Node Fusion algorithm. 141

Figure B.3 Complexity analysis of Inter-Node Fusion algorithm. 142

Figure B.4 Complexity analysis of Object Trajectory Construction algorithm. 143

LIST OF ABBREVIATIONS

ABBREVIATIONS	
CEP	Complex Event Processing
ITS	Intelligent Transport Systems
FVSA	Fused Video Surveillance Architecture
IoT	Internet of Things
API	Application Program Interface
IT2	Interval Type-2
FLS	Fuzzy Logic System

CHAPTER 1

INTRODUCTION

Wireless Multimedia Sensor Network (WMSN) is a very effective technology commonly used in automation systems applied in different domains from monitoring of an industrial site to home security for detecting unexpected visitors, from traffic control in a metropolis to immigrant surveillance at a border station. Considering the fact that these sensor nodes are 24/7 active and generating data each minute or even each second, it easily becomes a big data subject because of the volume, velocity and variety. Since relational databases have some drawbacks in big data applications, NoSQL databases are the savior in scalability and performance issues.

Big data was a huge hit in the last decade since the valuable information hidden in large volumes of data got attraction by both commercial and academia. As big data analytics evolved and hidden information in complex relationships was needed to be revealed, the graph databases emerged as the key technology from other NoSQL database types.

Surveillance is one of the most mission critical application domains with connected data with multiple dimensions like space and time. Therefore, there are many research studies in data analytics on surveillance applications for monitoring a zone. However, extracting the information from the big sensor data and using that knowledge to help better understanding the area for predictive analytics remains a challenging issue. To facilitate such challenges, this thesis proposes a novel fuzzy object tracking approach which is developed using a graph-based big data model by utilization of a multilevel fusion. This approach consists of three main steps: intra-node fusion, inter-node fusion, and object trajectory construction. Intra-node fusion exploits object detection and tracking in each sensor while inter-node fusion uses spatiotemporal data along with neighbor sensors. Since uncertainty naturally exists in trajectory data, an interval type-2 fuzzy rule based engine is developed to be used in both intra-node and inter-node algorithms to fuse trajectories. Then, all trajectories from all sensor nodes are integrated to construct fuzzy trajectories in the common ground-plane across the wireless multimedia sensor network. Extracted trajectories are utilized in advanced analytics like prediction and anomaly detection.

Two different real world use cases were experimented using GeoLife Trajectory dataset for surveillance on the ground and Maritime dataset for monitoring the vessels. Usage of real world application scenarios as well as the synthetically generated data, justified that the interval type-2 fuzzy logic utilization improves performance in both trajectory extraction and data analytics. My thesis increased the performance 10%-30% on real world datasets compared to the baseline which does not use any fuzzy logic system.

In this chapter, firstly motivation for this thesis is declared. Then, the problem formulation and definition is discussed. After that contributions are given in the following section. And finally, the organization of the thesis is presented.

1.1 Motivation and Problem

Sensors are present in various forms all around the world such as mobile phones, surveillance cameras, smart televisions, intelligent refrigerators and blood pressure monitors. Usually, most of the sensors are a part of some other systems with similar sensors that compose the sensor networks. Many studies have already been done on sensor networks in diverse domains like fire detection, city surveillance, and early warning systems. All those applications position sensor nodes and collect their data for a long time period with real-time data flow, which is considered as big data.

Big data may be structured or unstructured and needs to be stored for further processing and analyzing. However, analyzing multimedia big data is a challenging task requiring a high-level modeling to efficiently extract valuable information from data. Especially in surveillance applications, sensor nodes produce (near) realtime data which usually needs to be treated as big data because of the volume and velocity of data.

The hidden information inside the big data is revealed by data analytics. Some analytics on the surveillance domain can answer the following research questions;

- How can the path of an identified object be extracted?
- Where was the object at a given specific time?
- What is the possible position of an identified object after 10 minutes?
- Does the identified object behave abnormally?

To be able to deal with those problems, it is also needed to simulate and model the real system to generate synthetic data because of the following reasons;

- A long period of time, maybe years, is needed to store enough data to be analyzed
- Different kinds of scenarios may never happen during that time period

Another problem is the uncertainty in trajectories. Simultaneous detection and tracking of objects moving through a trajectory using wireless sensor nodes without any equipment attached on the object like GPS or RFID are challenging research topics, because of the reason that detailed analysis of spatiotemporal sensor data is required. There are several studies on trajectory extraction but uncertainty of the trajectory points is still an open issue.

1.2 Contributions

Considering the defined problems and scope of the thesis, main contributions of this dissertation are as follows:

- A new object tracking algorithm to extract trajectories on big surveillance data that consists of three main algorithms: intra-node fusion, inter-node fusion, and object trajectory construction. Intra-node fusion exploits object detection and tracking in each sensor. The output of intra-node fusion is used together with spatiotemporal data along with neighbor sensors to generate inter-node trajectories. Then, all the trajectories from all sensor nodes are fused to construct fuzzy trajectories for detected objects in the common ground-plane across the WMSN. Usage of fuzzy logic and multilevel approach in trajectory extraction performed better than the Kalman filter and Particle filter which are widely used in this problem scope.

- Extracted trajectories can be used for further analytics. Since uncertainty naturally exists in trajectory data, and analytics like prediction and anomaly detection need increased fuzziness to be able to handle indefinite information in a logically correct manner, interval type-2 fuzzy logic-based approach is proposed to foreseen trajectories and detect the outliers. An object following a trajectory causes an anomaly if it leaves the trajectory path. GeoLife Trajectories and Maritime Cadastre datasets were used to perform experiments, and results of these experiments show that using fuzzy based approach performs better than the compared classification algorithms.

- A graph-based model for big data is proposed to be able to do analytics on data with complex relations. Accordingly NoSQL graph database is used for the purpose of addressing the big data related problems. The network infrastructure model is intertwined with the sensed and fused data model to reveal the information from data. Although there have been some related studies in the literature about the surveillance systems in the big data context, to the best of our knowledge, there has not been an applicable big graph data model for WMSNs using a graph database yet.

- A simulation infrastructure is implemented for simulating multimedia wireless sensor networks and sensor node execution using both generated synthetic data and available real world datasets. Two different datasets are used for verification in order to test how applicable and effective are the proposed algorithms in the real world use cases. In addition, developed simulator application is being used by many other master and doctorate thesis.

1.3 Overview of the Thesis

This thesis is organized as follows: first chapter is the background information and related works, and next chapter is about the proposed graph-based big data model. Chapter 4 describes the developed big data compatible simulator using the reference real wireless multimedia sensor network. And final chapter proposes multi level fusion tracking algorithm and big data analytics using well-known classification algorithms.

The dissertation starts with an introduction and gives the motivation for this thesis afterward. Then, the problem and contributions of this dissertation are provided. Thereafter, Chapter 2 gives background information about the key topics and related works to this dissertation.

In Chapter 3, the first step of the research, defining a graph-based data-model and selecting a suitable graph database is introduced. An overview about the related works is given in Section 3.1. In Section 3.2, a new graph-based data model is proposed. Comparison and selection of a graph database is presented in Section 3.3. Then, proposed data model is implemented on selected graph database in Section 3.4. Finally, conclusions are given accordingly in Section 3.6.

Next step of thesis study is using the implementation of proposed graph-based data model, and this guides the reader to Chapter 4 where a comprehensive simulator to generate both network and data flow is introduced and developed. In this chapter, firstly specifications of the reference wireless sensor network system are given in Section 4.1. Then, building components and generic infrastructure of the simulator are provided in Section 4.2 and 4.3, respectively. Thereafter, in Section 4.4 system simulation which covers both network topology and data flow generation is given. In Section 4.5 a case study in surveillance, and in Sections 4.6 developed JavaFX data simulator application are described. Then concluding remarks are also given in the final Section 4.7 of this chapter.

Chapter 5 composed of many algorithms to track objects using semi-automatic fuzzy rule engine which this dissertation mainly based upon. First, an overview is given in Section 5.1. Then, a novel multilevel object tracking approach is provided in Section

5.2, and extracted trajectories are used for analytics in Section 5.3.3. Empirical studies on applying interval type-2 fuzzy logic on extracted trajectories are discussed in Section 5.3. Finally, at the end of this chapter brief remarks are provided in Section 5.5.

Chapter 6 composed of several experiments to benchmark the proposed multilevel object tracking algorithms and trajectory analytics. First, in Section 6.1, a synthetic data generated by the data simulator and a real world dataset were used evaluate the proposed algorithm. Then, various experiments were conducted using interval type-2 fuzzy logic in Section 6.2.

The final chapter of this dissertation, Chapter 7, encompasses conclusions, discussion, and foreseen future work which conclude the dissertation.

1.4 Declaration

The content of this thesis is the result of the author's original work, except where referenced or stated otherwise. Parts of this dissertation have been previously published by the author.

CHAPTER 2

BACKGROUND AND RELATED WORK

There are several topics that this dissertation is based on. As sensor networks intertwine with Internet of Things (IoT), big data is interlaced with NoSQL databases. In this chapter, background information for key topics are provided.

In this chapter, firstly description of Internet of Things and how it is evolved is described. Then, sensor networks is declared in the following section. After that, NoSQL databases including Graph databases are provided. And last but not least, big data and analytics of the big data are discussed.

2.1 Internet of Things

Kevin Ashton is the person who used the Internet of Things (IoT) naming for the first time in 1999 [2] and it roughly means that devices, called Things, connected to the Internet for various purposes from home security to weather forecast, and from shopping to healthcare. In all use cases, Things will use the Internet instead of humans to provide a service. The sensors, RFIDs, and nanotechnology help this mission to be accomplished by taking away the need for human-entered data.

Pankesh et al.[3] propose a domain model for IoT to make a common understanding. To define the model, they reference to the real world applications and summarize under three headings; Intermittent Sensing for RFID kind of technologies, Regular Data Collection for smart devices to interact with humans, and Sense-Compute-Actuate loops for machine-to-machine communications between smart devices.

As parallel to Sense-Compute-Actuate cycle, Sensor-as-a-Service (Senaas) notion is defined by Sarfraz et al. [4]. They virtualized the sensors as services by an abstraction on technical details of sensors. They trigger services with an event in the sensor and compute it to reply an action. Their IoT virtualization framework is validated by a case study.

Atzori et al. [5] prepare a comprehensive survey about IoT. They identify the enabling technologies as sensing, identification and communication systems like RFID, WiFi, and sensors. In addition, middleware applications using Service Oriented Architecture (SOA) are important for data distribution. In the end, they list open issues such as privacy, addressing of things and non-standardized applications for the future.

IoT is fully connected to sensor technology and the researches about sensor networks directly or indirectly improve the IoT. Hong et al. [6] propose an approach to Internet of Things using IP-based wireless sensor networks. They realize that IoT probably has the same problems that Internet itself had in the past. So, they identify problems like mobility, security, time synchronization, and IP6 adaptation. They also share evaluation results of an implementation of their proposed SNAIL platform.

2.2 Wireless Multimedia Sensor Networks

A set of sensors called sensor nodes connected to each other or connected to leading gateways is simply called a sensor network (SN). If sensors have capability of collecting multimedia data and have a communication infrastructure among the sensors, it is called multimedia sensor network (MSN). If sensors are connected to each other using wireless technology then it is a wireless multimedia sensor network (WMSN).

Akyıldız et al. [7] discuss the state of the art of research on WMSNs as well as the challenges. The challenges related to WSN deployment configurations are summarized by Perera et al. [8] in their research. Another survey paper [9] enlists the challenging issues to design middleware systems for WSN. A couple of the identified challenges are as follows:

- Data fusion
- Resource management

- Scalability and network topology
- Security
- Quality of Service
- Limited power

From the database point of view, Ramesh et al. [10] define a database layer on top of sensor network so that a database query is mapped to traversing sensor nodes in the WMSN.

2.3 NoSQL Graph Databases

NoSQL databases grouped under four categories which are key-value store (e.g. Amazon's Simple DB), big table databases (e.g. Google Cloud Bigtable), document-oriented (e.g. CouchDB) and graph databases (e.g. OrientDB). Graph databases [11] consist of nodes and edges (relations between nodes) which store data as properties. Most of graph databases provide the capability to label nodes and edges. NoSQL Graph databases provide many ways to query data such as;

- User interface via SQL-like query language (Cypher for Neo4j, SQL for OrientDB)
- User interface via Graph visualization to interact with the nodes and edges
- Application program interface (API) to programmatically connect to database

Unfortunately, there is not any standardized way of querying, so that you have to write database specific queries every time. There is an open-source framework Apache TinkerPop to provide graph computing capabilities for graph databases. Gremlin is a part of the TinkerPop to traverse the graphs. And by the support of most of the graph databases, any gremlin query can be written once and work on every graph database.

Graph databases are generally preferred to handle social networks, fraud detection, graph-based operations, real-time recommendations and hierarchical relations.

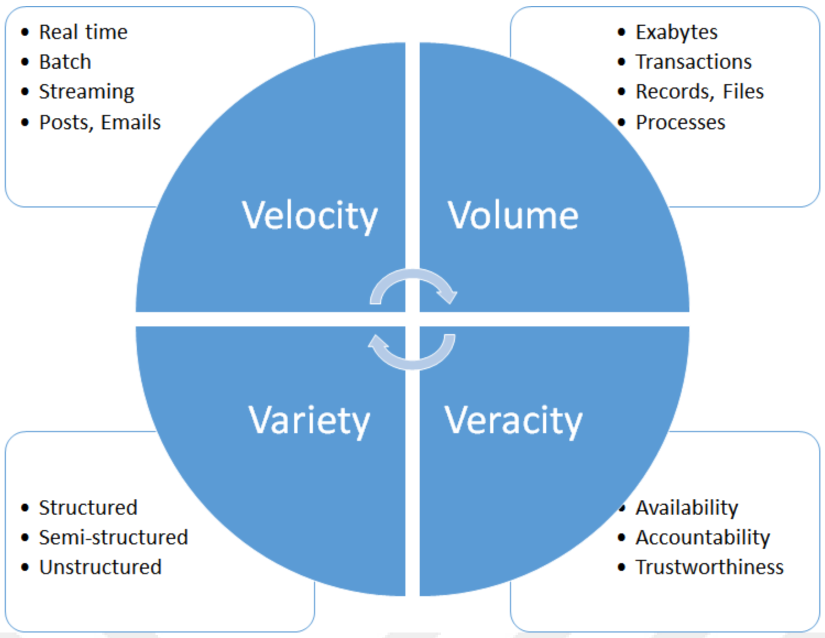


Figure 2.1: 5 Vs of big data.

Moniruzzaman et al. [12] evaluate NoSQL databases in the aspect of big data analytics. In their survey, they enlist different types of NoSQL databases according to characteristics (features and benefits of NoSQL databases), classification (key-value, document, column-based and graph); and evaluation with a matrix on the basis of few attributes like design, integrity, indexing, distribution, and system.

2.4 Big Data

The definition of the Big Data is all in the name. An extremely large set of data is called Big Data, and it is not exactly something new but the fact of the need to process and analyze that data that holds out hidden knowledge inside is the reason why it is the buzzword of recent years. Big Data is defined by a number of Vs; Velocity, Variety, Volume, Value and Veracity as shown in Figure 2.1.

Volume is the quantity of stored data. Velocity is the speed of data generation or processing. Variety is the type and structure of the data. Value is the importance of information that data provides. Veracity is the variation in quality of data.

The survey paper [13] points the relation between IoT and big data. For example, jet aircraft engines produce one terabyte of data per flight using various sensors. Think about a huge number of flights in a day all around the world and then you can have really big data. HP prepared a business-value white paper related to big data. According to the paper, 1 trillion sensors, roughly 150 sensors for every person will be existed by 2030. The generated data will be mostly unstructured data and the value of it depends on how the information is extracted from the data. As the number of sensors increases,much more storage and processing will be required. And all of these creates some new challenging issues.

Many papers [14, 15], state that relational database management systems are inadequate for big data and NoSQL databases are the solution at least for the time being.

2.5 Big Data Analytics

There are various challenges of applications to work on big data like storage, management, security, analytics, and processing [16, 17]. Big data analytics is the ability of extracting gold plated information from huge dataset and have some conclusions that will support in decision-making. This can be challenging since volume and velocity cause big problems for data analysis because of technological limitations. Additionally, the variety of data types needs different approaches and algorithms for different sources. Therefore, advanced analytical methods and techniques are needed to process big data.

Depending on the used data mining techniques and algorithms, big data serves different kinds of information. Both historical statistics and predictive forecast can be achieved by big data analytics. There are three main types of big data analytics methods.

- Descriptive analytics uses big data to represent the current state of the situation as a statistical report using statistical methods, like standard deviation, median, variance of values recorded in big data [18]. Historical big data is used in descriptive analytics to apply statistical methods to extract patterns and understand the past behaviors Descriptive analytics are kind of looking into history and expose what happened.

- Predictive analytics helps to statistically model and forecast the future possibilities. Learning models of predictive analytics can be supervised, unsupervised, and semi-supervised.

- Prescriptive analytics is about optimization to analyze the cause-effect relationship and produce improvement policies. This analytics method is brand new compared to other methods and not common yet.

In surveillance applications, most challenging and widely used analytics are the trajectory analytics [19]. Trajectory extraction is a descriptive analytics used by the raw track data collected by sensors or radar systems. Trajectory prediction and anomaly detection are in the scope of predictive analytics as there is an estimation of the next location of moving objects.

2.6 Machine Learning Algorithms

Machine Learning algorithms have been widely used on temporal data models such as time series and spatiotemporal data for years. Classification, regression, predictive learning, outlier detection, and forecasting are some of the tasks that these algorithms are applied.

Random Forest

A random forest [20] consists of a set of decision trees. Each tree is created using randomly generated feature vectors which are individually sampled.

If lots of trees are used in the forest, many of the features are included. This results in helping limit the generalization error due to variance and bias. If we didn't use randomly generated features, then trees would be highly correlated. The reason of that correlation is that few features could be particularly predictive and thus, the same features would be chosen in many of the base trees.

Random forests are assumed to be more robust than the decision trees. Because of the fact that they aggregate mulitple decision trees to limit overfitting as well as error due to bias and therefore yield useful results.

Naive Bayes

The Naive Bayesian [21] algorithm provides an approach to represent, use and learn probabilistic knowledge. The approach is designed for using during the initiation tasks in which the objective is to accurately predict which class test cases belong to. Such a classification can be seen as a specialized form of Bayesian network, described as naive because it is based on two major simplifying assumptions. The former assumes that the predictive attributes are independent with regard to class conditionally, and the latter accepts that no hidden or latent features can affect the prediction process.

Decision Table Decision tables [22] is a popular method for machine learning of classification and regression. It is widely used because of its many advantageous such as being easy to interpret, handling continuous and discrete features, having no need for feature scaling. As with other classification methodologies, this method has a two-step process, namely training and classification. In training phase, pre-labeled training data is analyzed by the classification algorithm to create a model. In the classification phase, using the test data, classification rules are applied to give a decision in order to generate the final output.

Logistic Regression

In statistics, logistic regression [23] is a statistical method to analyse a set ot data which is composed of one ore more independent variables. The output is a binary value that can be "0" or "1" to represent fail or pass, lose or win, dead or alive kinds of information.

In cases where the output can have more than two values, then multinomial logistic regression can be used.

Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) [24] is a neural network which is composed of several layers and neurons.

From architectural point of view, several neurons with nonlinear activation function are hierarchically connected to each other in MLP. It uses back-propagation approach for training.

Sequential Minimal Optimization (SMO)

Sequential minimal optimization (SMO) [25] is generally used in the training phase of the support vector machines. SMO transforms nominal attributes into binary attributes to fulfil the necessary values. As default, all the attibutes are normalized. Since SVM is widely used in character recognition and face recognition problems, its training is more complex and time consuming compared to the SMO though.

2.7 Fuzzy Logic System

A fuzzy logic system (FLS) provides a nonlinear mapping of an input data to a scalar output data [26]. In Figure 2.2, a type-1 (T1) fuzzy logic system representation is given. T1 FLS has four main steps: fuzzification, rule evaluation, knowledge base, and defuzzification.

During the fuzzification stage, the actual inputs are converted to fuzzy membership functions. The knowledge base provides the list of rules in the application domain. The big challenge is to design the correct list of rules for the problem. Then, experts were used to filter and organize the rules generated automatically in the knowledge base. The fuzzy logic system inputs are being processed by the rule evaluation process, which uses the knowledge base to combine fuzzy inputs and produce outputs mapped to fuzzy membership functions. In the defuzzification stage, the fuzzy outputs of the rule evaluation step are mapped to crisp outputs.

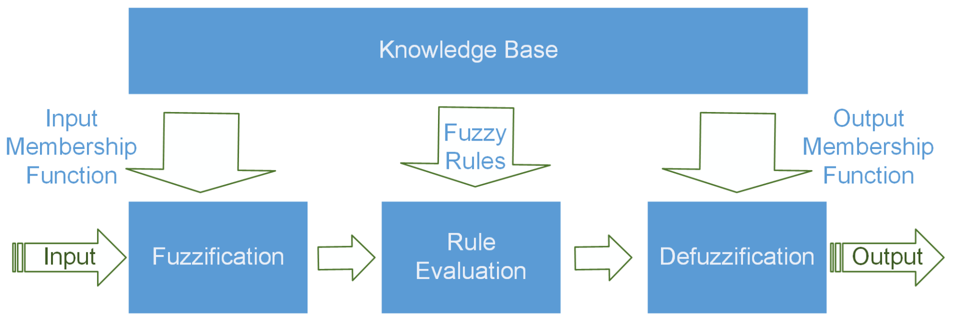


Figure 2.2: Type-1 fuzzy logic system.

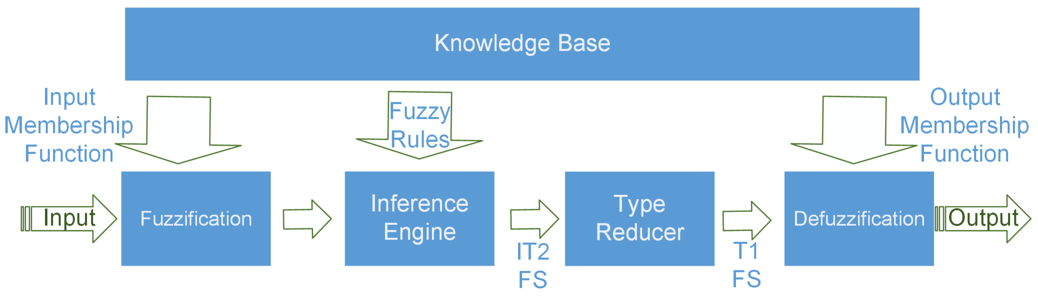


Figure 2.3: Interval type-2 fuzzy logic system.

Despite being a cure for most of uncertainty cases, researches has shown that there are restrictions in the ability of type-1 fuzziness to model the uncertainties [27]. The reason of the drawback is that membership grades of type-1 fuzziness are crisp values, and for more uncertain cases, type-2 fuzziness [28] evolved by making the membership functions fuzzy. Because of the computational complexity, interval type-2 (IT2) FLS [29] are more popular and widely used.

There are five components in an IT2 FLS; fuzzifier, fuzzy rules, inference engine, type-reducer and defuzzifier. Figure 2.3 shows a representative diagram of an IT2 FLS which is similar to type-1 counterpart, but the main difference is that inference engine works on IT2 fuzzy rules in the rule base. Accordingly, a type-reducer is needed to convert the outputs of the inference engine, which are IT2 FSs, into a T1 FS before defuzzification step.

CHAPTER 3

GRAPH-BASED DATA MODEL

A graph consists of nodes and edges, which shows the relations between nodes. Therefore, graphs are very efficient and convenient to handle data which has valuable information in relations. Many application domains like social networks, recommendation systems, and fraud detection in money transactions graphs are widely used and data is stored in graph databases. While storage of the data is an important task, processing the streaming data and taking action for mission-critical applications are also crucial. In order to process that kind of data, it is needed to identify the data flow well.

A graph-based big data model is proposed in a generic manner for handling multimedia wireless sensor network data. Graph-based data model suits well for advanced analytics like graph mining, and prediction using the complex relationships between data. For that purpose, big sensor data is stored in a graph database in such a way that proposed graph model represents both the sensor network topology and the data flow among the nodes. The applicability of this solution is illustrated with a prototype implementation and experimental evaluations. A number of experiments have been done for measuring the accuracy and efficiency of this solution. Simulation results show that proposed multimedia wireless sensor network model is applicable in large-scale real life scenarios.

This chapter is organized as follows: next section provides an overview over related work. Then, a new graph-based big model is proposed in Section 3.2. To implement the model, the top graph databases are evaluated in Section 3.3. Implementation of proposed model is presented in Section 3.4. Section 3.5 presents the experimental results and evaluations. Finally, conclusions are drawn in Section 3.6.

3.1 Overview

Over the years, various methods are used for wireless sensor network data representation and management [30, 31, 32, 33]. Yang et al. [34] propose a hybrid data model to store their wireless sensor network data. NoSQL part of the hybrid model is stored with a key-value structure to provide higher scalability and better performance. Christine et al. [35] discuss big data with spatial data received from wireless sensors using real life scenarios.

Renzo et al. [36] present a survey paper on graph database models. They compare graph database models to other database models like a relational model and then compare the representative graph database models. Mark et al. [37] introduce a graph based model which is called HyperNode. They also define a language HNQL (HyperNode Query Language) to query and update their model. Arati et al. focus on the information retrieval from sensor networks and propose hybrid protocol which is called APTEEN [38]. They have noticed that achieving an efficient model needs to replace the conventional flat topology with a graph based model.

PipeNet [39] is multi layered wireless sensor network application on pipeline scenario which is the similar multi layer architecture as proposed but in another domain. They have developed the system to analyze the collected multi-modal data for detection of the leaks. Another research about WSN related to surveillance domain is a survey paper written by Felemban [40]. His survey research enlists the literature for experimenting work done in border surveillance and intrusion detection using the technology of WSN. This thesis differs from the existing works by employing a graph based approach for surveillance domain and focusing on the simulation of the big data.

The wireless sensor networks have been modeled using graphs in several studies. Punyasha et al. [41] proposed an energy-efficient node distribution strategy to avoid unnecessary data transmission. They have modeled the network topology as well as the energy levels of sensor nodes. Mario et al. [42] analyzed the performance of localization methods used in WSNs. They have modeled the wireless sensor networks as a graph to be used to assess how the localization algorithms perform in different network topologies. This dissertation not only models the network topology, but also the data flow transmitted in the wireless sensor networks in order to do analytics on a complete graph-based data model.

3.2 The Graph-Based Big Data Model

In order to model and simulate a multimedia wireless multimedia sensor network from the point of network topology and the streaming data among the sensor nodes, a reference WMSN system, which is designed and developed in CENG Multimedia Laboratory, was used. Wireless nodes are connected to each other and equipped with many multimedia and scalar sensors.

There are three main types of nodes; sensor node, gateway, and sink. The sink node is the base station and many clusters are connected to the sink. Gateway nodes are the cluster head, which is connected to a set of clustered sensor nodes [43].

On top of network topology, the data flow, from a sensor node to gateway and from gateway to gateway (multi-hop) or sink, was modeled. The data in the reference system is gathered using a camera and acoustic, seismic and PIR scalar sensors. There is a multi-layered mechanism to ingest data from sensor nodes to the sink node. The big graph-based data model is comprehensively designed to include the lifecycle of data from sensing to analyzing.

Figure 3.1 shows a snapshot of proposed graph-based NoSQL database to represent a sensor data ingested from a sensor node, fused and sent to the gateway. **Actual Data** represents a real world physical event which causes a change in sensors deployed on the sensor node and is used as a ground truth for simulation data. **Sensor Raw Data** is the sensed data in the sensor node that represents the digitalized data captured from the real world. If there is a camera recording of raw data, multimedia data is represented by **Multimedia Sensor Data**. Raw Data is connected to the **Sensor Node**, which sensed data by itself. After applying multi-layer fusion to raw data, **Fused Data** is formed and enriched by each fusion layers. In order to fully cover the lifecycle of the data, the fused data is being analyzed and an **Action** like triggering an alarm or sending a notification message to another system is created.

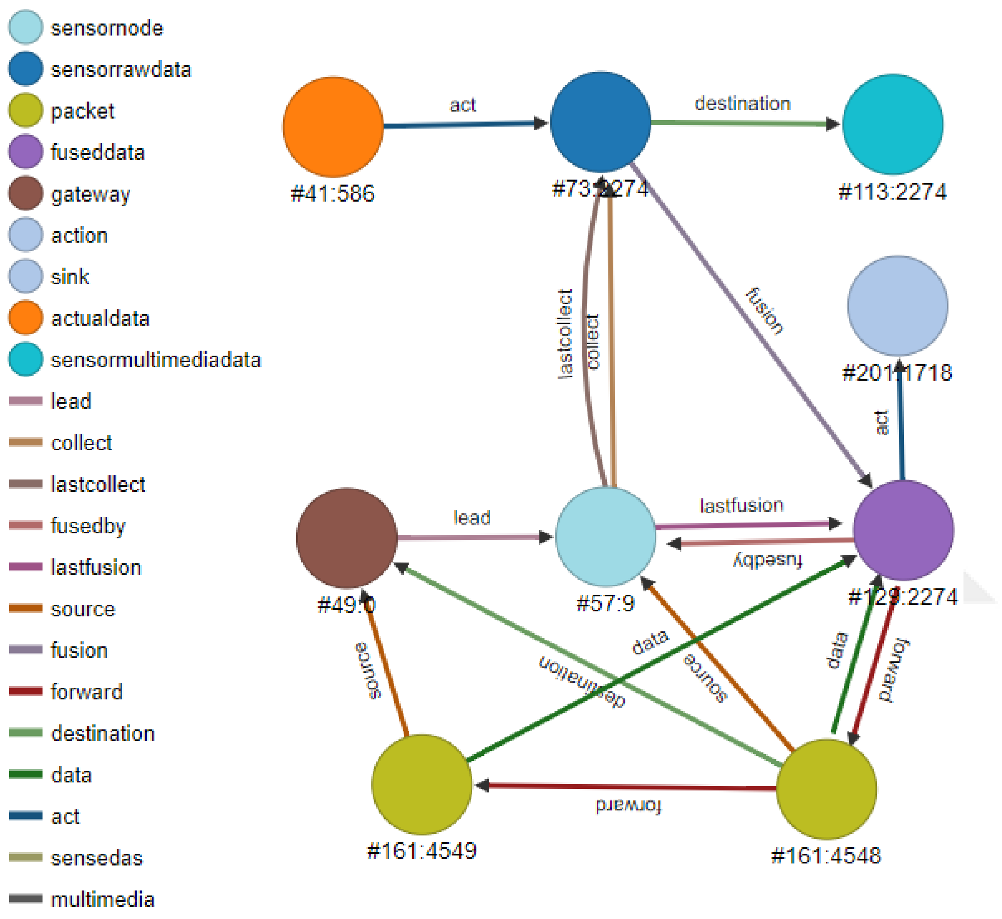


Figure 3.1: Graph-based data model.

A graph database is a storage environment and data model is designed as a graph-based big data model in parallel to the NoSQL database selection. Data stored in graph databases is used for further analytical processes like tracking and event detection.

3.3 Graph Database Selection

This thesis includes applying the currently available databases to proposed graph-based big data model. Salem et al. [44] compare a set of databases like Cougar and TinyDB. Li-Yung Ho et al. [45] propose a distributed graph database based on an open-source graph database which is called Neo4j. The options are limited if you are looking for a graph database. Neo4j, Titan, and OrientDB are featured open-source graph databases.

Neo4j is a well-known graph database and used by many researchers. In addition, Neo4j is relatively easier to be used rapidly by developing some small pieces of code. Spring Framework support is really helpful to put things together very fast.

Titan is another open-source option for a graph database but its development is stopped and discontinued in early 2015. Therefore, it is not preferred to utilize Titan.

OrientDB is another open-source graph database which is not as popular as Neo4j for now but has many advantages over it. Table 3.1 shows the comparison of OrientDB and Neo4j Community Editions which is provided by the official website of OrientDB. From all those compared features, "Multi-Master Replication", "SQL" and "Elastic Scalability with Zero Configuration" are the most important features for us to choose OrientDB.

3.4 Model Implementation Over Graph Database

Nodes (vertices) and relations (edges) are defined in graph databases to store data. Compared to the traditional RDBMS approach, every row in a table is replaced with a node and its properties. There are edges to represent cross-table references.

Table 3.1: Compare OrientDB and Neo4j Community Editions

Feature	OrientDB	Neo4j
Graph Database Support	Yes	Yes
Transaction Support	Yes	Yes
Hooks	Yes	Yes
User Management for Security	Yes	No
SQL Support	Yes	No
Full-text Search Support	Yes	Yes
Spatial Data Support	Yes	Yes
Database Triggers	Yes	No
Multiple Index Support	Yes	No
Multi-Master Replication	Yes	No
Sharding	Yes	No
Elastic Scalability	Yes	No
Data Types Extension	Yes	No
Server-Side Function Support	Yes	No
Embedded Mode without Restrictions	Yes	No
Sequences	Yes	No

At the first step, the node and edge types are defined. Node types are; Sink, Gateway, SensorNode, SensorRawData, SnFusedData (*SensorFusedData*), GwFusedData (*GatewayFusedData*) and SinkFusedData. Edge types are; Lead, Collect, LastCollected, Next, Fusion, FusedBy, LastFusion, Reported and Forwarded. The edge types are defined for the usage between specific nodes. Table 3.2 lists the edge types in the graph database.

Each sensor node has the capability to hold temporarily a set of data which are sensed by sensors like *PIR*, *seismic*, *acoustic* and camera. That capability is provided by an in-memory database. For the fusion at the sensor node level, this in-memory database is used as a cache to analyze the changes in scalar sensors and provide some additional data to the first level fusion. The algorithm of the first level fusion is shown in Algorithm 1. The fusion result is stored in the *fusedData* including the *video*, *silhouette*, *foreground* and low level *features*.

Algorithm 1 Sample first level fusion algorithm executed on sensor nodes

```
1: procedure FIRSTLF($PIR, seismic, acoustic, threshold$)
   ▷ $PIR$ identifies if there is a movement or not, two integers $seismic$ and $acoustic$
   values are scalar data sensed by the node, $threshold$ is used to identify that there
   is an object with enough sound and vibration on sensor node
2:     $fusedData$ = $\emptyset$
3:     if $PIR = true$ and $seismic \geq threshold$ and $acoustic \geq threshold$ then
4:         $fusedData.video$ = startVideoRecording()
5:         $fusedData.frame$ = selectFrame($fusedData.video$)
6:         $fusedData.frmFeat$ = findLowLevelFeatures($fusedData.frame$)
7:         $fusedData.fgnd$ = selectForeground($fusedData.frame$)
8:         $fusedData.fgndFeat$ = findLowLevelFeatures($fusedData.fgnd$)
9:         $fusedData.silhouette$ = extractSilhouette($fusedData.fgnd$)
10:    end if
11:    return $fusedData$;
12: end procedure
```

Sensor nodes apply the first level fusion and reports the output *fusedData* to leading gateway. The gateway applies the second level fusion as in Algorithm 2. The purpose of fusion at the gateway is a refinement of the sensor fused data before sending to

Table 3.2: Node and Edge Types

Edge Type	From Node	To Node
Lead	Gateway	SensorNode
Lead	Gateway	Gateway
Lead	Sink	Gateway
Collect	SensorNode	SensorRawData
LastCollected	SensorNode	SensorRawData
LastFusion	SensorNode	SnFusedData
LastFusion	Gateway	GwFusedData
Next	SensorRawData	SensorRawData
Next	SnFusedData	SnFusedData
Fusion	SensorRawData	SnFusedData
Fusion	SnFusedData	GwFusedData
Fusion	GwFusedData	SinkFusedData
FusedBy	SnFusedData	SensorNode
FusedBy	GwFusedData	Gateway
FusedBy	SinkFusedData	Sink
Reported	SnFusedData	Gateway
Forwarded	GwFusedData	Gateway
Forwarded	GwFusedData	Sink

the sink node. As a sample refinement algorithm, removing the duplications and normalizing the data according to scalar data can be written.

Algorithm 2 Sample second level fusion algorithm executed on gateways

```
1:  procedure SECONDLF($fusedDataList, threshold$)
    ▷ The list of $fusedData$ reported by leading sensor nodes, $threshold$ is used to
    identify the difference between two scalar value
2:      $filteredDataList$ = []
3:      for $i = 0$ to $fusedDataList.size$ do
4:          $current$ = $fusedDataList[i]$
5:          $previous$ = previous element of $current$
6:
7:          $diff$ = $current.acoustic$ - $previous.acoustic$
8:          $diffRate$ = $current.acoustic$ * $threshold/100$
9:          if $diff < diffRate$ then
10:             mark $current$ as duplicate and drop
11:         else
12:             add $current$ to $filteredDataList$
13:         end if
14:     end for
15:     return $filteredDataList$
16: end procedure
```

The output of the second level fusion is reported to the sink node. As the final decision maker, the sink correlates the concepts forwarded by gateways and decides that if an *action* is necessary or not. If an action is required, notification of the operator is triggered as given in Algorithm 3. All those three fusion algorithms are sample algorithms to provide the proof of concept execution of all phases of the simulated environment.

All those three fusion algorithms are sample algorithms to provide the proof of concept execution of all phases of the simulated environment.

Algorithm 3 Sample third level fusion algorithm executed on the sink

```
1:  procedure THIRDLF($filteredDataList, threshold$)
    ▷ The list of $fusedData$ reported by reporting gateways, $threshold$ is used to identify the difference between two scalar value
2:      $actionList$ = []
3:      for $i = 0$ to $filteredDataList.size$ do
4:          $current$ = $fusedDataList[i]$
5:          $previous$ = previous element of $current$
6:          if $current.acoustic > threshold$ and $previous.acoustic > threshold$ then
7:              $action$ = createAction($current$)
8:              add $action$ to $actionList$
9:              notify operator using $action$
10:         end if
11:     end for
12:     return $actionList$
13: end procedure
```

3.5 Experimental Work

A test environment was setup to make some experiments on proposed graph model. Test environment specifications were;

- Intel i7-4710HQ Quad Core CPU
- 16 GB DDR3 RAM
- 240 GB SSD Storage
- 4 GB NVIDIA 860GTX GPU

Test environment had three database systems which were;

- OrientDB v2.2.17 (Graph database)
- Neo4j v2.3.12 (Graph database)
- MySQL v5.7.26 (Relational database)

The simulator explained in the next chapter was used to simulate a multimedia wireless sensor network with synthetic raw data. Sensor nodes were placed in a square shaped area, and gateways were located in the center of each group of sensor nodes. The sink node was placed in the center of the whole area as shown in Figure 4.6.

There was 25 million sensor nodes which were leaded by 2,500 gateways, and the sink node was responsible to collect all data in the simulation environment.

3.5.1 Comparison to Relational Data Model

This experiment was done on all three databases installed in the test environment. Many queries was run on both the graph data model and relational data model. Figure 3.2 shows the relational database representative of the graph data model on MySQL.

Below queries are randomly selected sample queries and Table 3.3 shows the test results of the experiment.

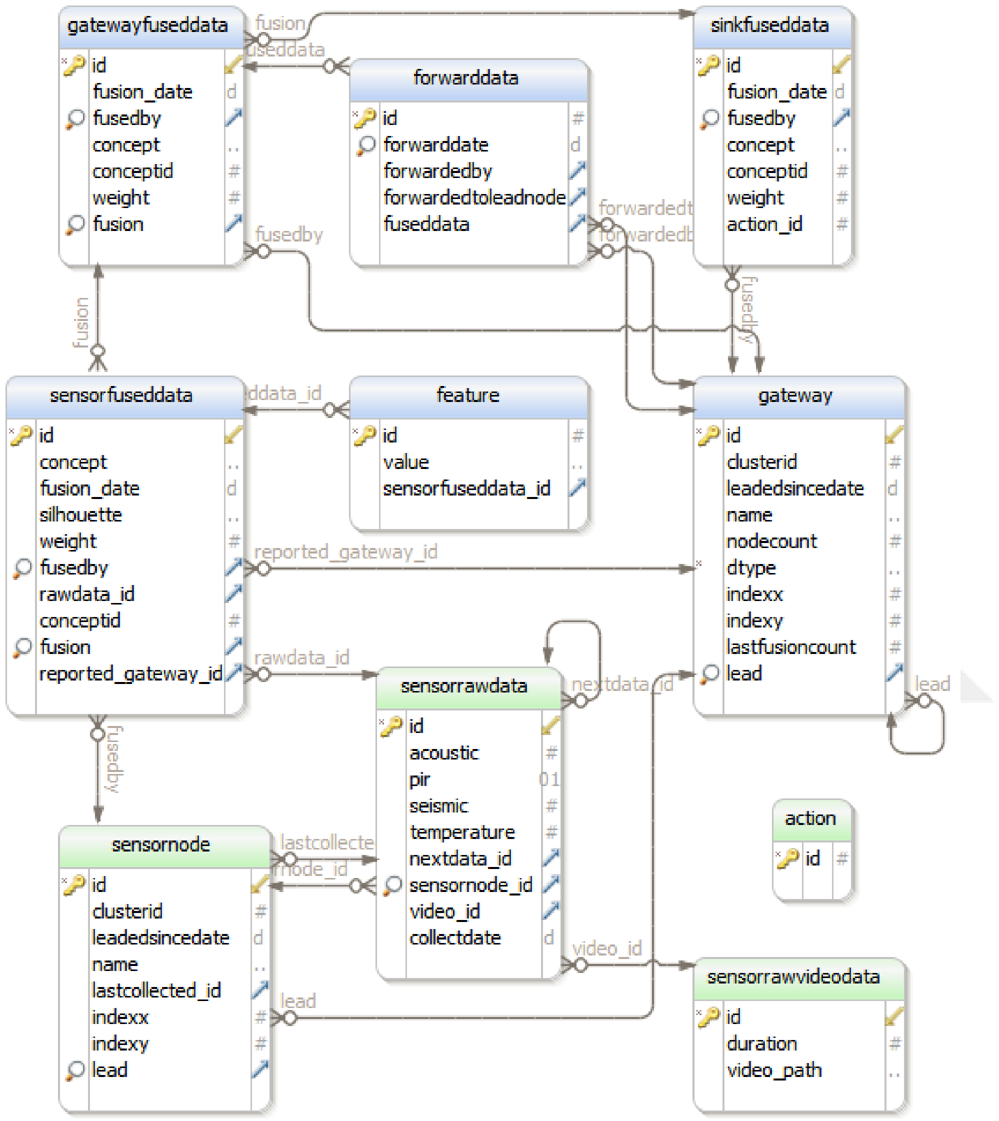


Figure 3.2: Relational Database Schema

3.5.1.1 Concepts Based Query

This query finds the specific type of objects with the highest probability of its detection time and location.

OrientDB Query:

```
SELECT concept, weight, fusionDate,
out("fusedby").indexX, out("fusedby").indexY
FROM fuseddata
WHERE weight>0.90 AND concept = "Vehicle"
ORDER BY fusionDate
```

Neo4j Query:

```
MATCH (b:fuseddata)-[:fusedby]->(sd:sensornode)
WHERE b.concept = "Human"
AND b.weight>0.90
RETURN b.concept, b.weight, b.fusionDate,
sd.indexX, sd.indexY
```

Relational SQL Query:

```
SELECT b.concept, b.weight, a.fusion_Date,
sd.indexx, sd.indexy
FROM sinkfuseddata a, gatewayfuseddata g,
sensorfuseddata b, sensornode sd
WHERE a.concept = 'Vehicle'
AND a.weight>0.90 AND a.id = g.fusion
AND g.id = b.fusion AND b.fusedby = sd.id
ORDER BY a.fusion_Date ASC
```

3.5.1.2 Video Based Query

This query finds the possible explosions by identifying continued high volume around the surveillance area with their recorded video paths and video duration. The value bigger than 15 is assumed to be a high volume sound.

OrientDB Query:

```
SELECT in("collect").name[0], in("collect").indexX[0],
in("collect").indexY[0], acoustic,
out("video").videoPath[0],
out("video").videoDurationSec[0]
FROM sensorrawdata
WHERE acoustic>15
AND out("next").acoustic[0]>15
AND out("next").out("next").acoustic[0]>15 ORDER BY name
```

Neo4j Query:

```
MATCH
(sn:sensornode)-[:collect]->(sa:sensorrawdata)-[:next]->
(sb:sensorrawdata)-[:next]->(sc:sensorrawdata)-[:video]->
(sv:sensorrawvideodata)
WHERE sa.acoustic>15
AND sb.acoustic>15 AND sc.acoustic>15
RETURN sn.name, sa.acoustic, sn.indexX, sn.indexY,
sv.videoPath, sv.videoDurationSec ORDER BY sn.name
```

Relational SQL Query:

```
SELECT s.name, s.indexx, s.indexy, ra.collectDate,
ra.acoustic, v.video_path, v.duration
FROM sensornode s, sensorrawdata ra, sensorrawdata rb,
sensorrawdata rc, sensorrawvideodata v
WHERE ra.acoustic>15 AND rb.acoustic>15
AND rc.acoustic>15 AND s.id = ra.sensornode_id
AND ra.id = rb.next_id AND rb.id = rc.next_id
AND rc.video_id = v.id ORDER BY s.name ASC
```

3.5.1.3 Recursive Query

This query finds the detected “Human” typed objects with high accuracy and calculates the distance of the sensor node to the sink node.

OrientDB Query:

```
SELECT $nodeId, out('fusedby').name[0],
fusionDate, $deep.count
FROM fuseddata
LET $nodeId = out('fusedby').@rid,
$deep = SELECT COUNT(*)
FROM (TRAVERSE in('lead') FROM $nodeId)
WHERE concept = 'Human' AND weight > 0.9
```

Neo4j Query:

```
MATCH p=(a:sink)-[:lead*]->(b:gateway)
WITH b.name as gname, length(p) AS depth
MATCH (sfd:fuseddata)-[:fusedby]->
(sn:sensornode)<-[:lead]-(g:gateway)
WHERE sfd.concept = "Human"
AND sfd.weight>0.9 AND g.name = gname
RETURN gname, sn.name, sfd.fusionDate, depth
```

Relational SQL Query:

```
WITH RECURSIVE search_graph(id, name, lead, depth)
AS SELECT g.id, g.name, g.lead, 0
FROM gateway g WHERE g.lead is null
UNION ALL
SELECT g.id, g.name, g.lead, 0 FROM gateway g
WHERE g.lead is null
UNION ALL SELECT g.id, g.name, g.lead, sg.depth + 1
FROM gateway g, search_graph sg
WHERE g.lead = sg.id
```

```
SELECT s.name, s2.name, s2.indexX, s2.indexY, s.depth
FROM search_graph s INNER JOIN
SELECT DISTINCT sn.id, sn.name, sn.indexX, sn.indexY,
sn.lead
FROM sinkfuseddata sfd, gatewayfuseddata gfd,
sensorfuseddata srfd, sensornode sn
WHERE srfd.fusion = gfd.id
AND gfd.fusion = sfd.id AND srfd.fusedby = sn.id
AND sfd.concept = 'Human' AND sfd.weight>0.9 s2
ON s2.lead = s.id
```

Table 3.3 shows the performance results of example queries. For the first query is a simple range query which is focused on the basic query performance. OrientDB performed better than Neo4j because of the under-hood architecture of OrientDB which is a multi-model graph database. And, graph model performed better than the relational model since there was no join operation as promised by the NoSQL databases.

For the second query, Neo4j performed better than OrientDB and the graph model was again better than MySQL. That time again graph databases beat relational databases because of their join-free query capability. To understand why Neo4j is faster than OrientDB, it is needed to dive into queries. The query was type of neighbors and neighbors of neighbors query, which is a typical graph matching problem considering paths of length 1 or 2. In PostgreSQL, a relational table with id backed by an index is used. Neo4j performed better because of its "index-free adjacency" for the edges.

The last query was to test the recursive SQL type of a query. The graph-based model was much faster than the relational model. Neo4j failed for this recursive query. There could be some optimizations possible while using the Cypher language (the query language of Neo4j) but it was not possible to find them in the available Neo4j documentation.

Table 3.3: Test Results (Numbers are in milliseconds)

Query	OrientDB	Neo4j	MySQL
Concept Based	209	618	938
Video Based	355	145	422
Recursive	4,293	79,812	36,469

3.5.2 Doubling Sensed Raw Data Size

An OrientDB graph database was selected for the execution of experiments. Previous experiments were applied on generated synthetic data of one month where each sensor node can sense data with 5 minutes of period. Now, the simulation duration was increased from 1 month to 5 months step by step and diagnosed the query performance.

Figure 3.3 shows the chart of query times affected by the increased simulation duration. The query time was increased and it was better than linear which was fairly well compared to the doubled data size.

3.6 Remarks

A graph-based big data model was proposed to represent the multimedia sensor networks, and data model was implemented to simulate multimedia wireless sensor networks. The sensor data retrieved from the network and the video data streamed from cameras were treated like big data. The storage environment of aforesaid big data was selected as a graph database which suits well to the proposed graph model.

The network topology and the data flow between each sensor nodes, gateways and sink were modeled so that all static and kinetic data was stored within the sensor network which must be assumed to be big data. The database to store big data was the NoSQL graph database. Because graph databases are good at representation of complex relations and scalable to store big multimedia sensor data.

Proposed model was tested on both graph databases and a relational database. Test

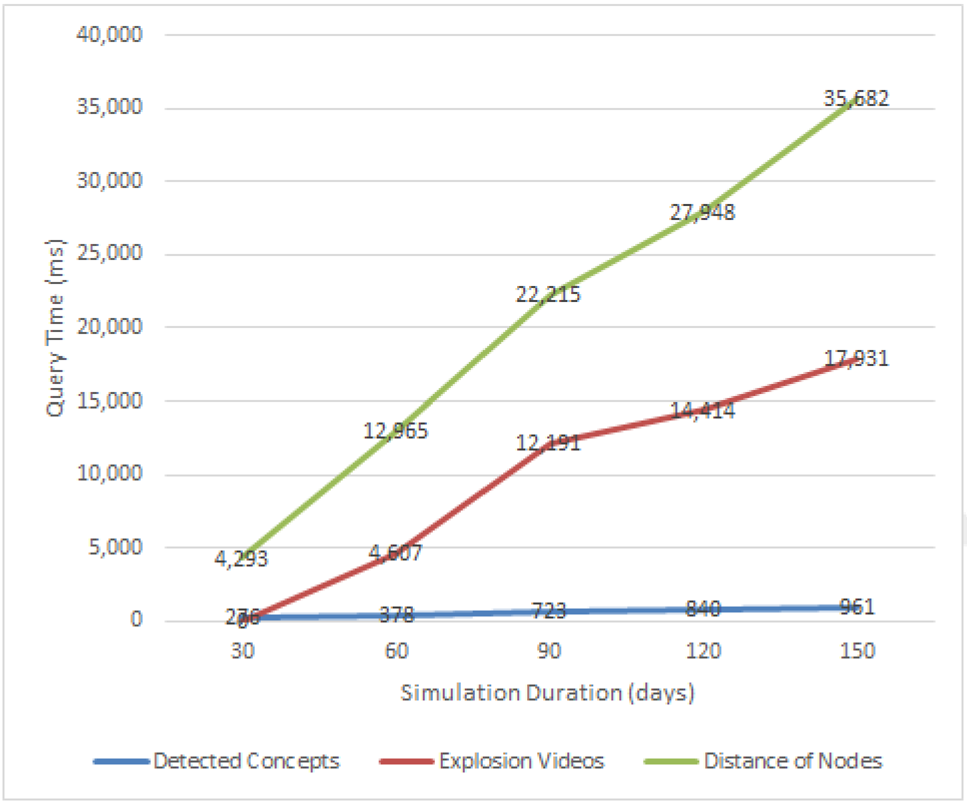


Figure 3.3: OrientDB query time and simulation duration.

results showed that graph database model performs better that the relational database model. To decide which graph database was more convenient and efficient,two well-known graph databases, OrientDB and Neo4j, were chosen for experiments. Neo4j is the market leading graph database, but it did not fit into the main requirement, storing complex multimedia big data. Because Neo4j supports high availability with the master-slave approach, which can scale vertically. But, in order to survive in the big data world, the master-master approach is needed, which OrientDB supports. In addition to that, experiments showed that OrientDB performs better than Neo4j.

Graph-based big data model successfully survived even with millions of data nodes. Many complex query scenarios were tested on synthetic data and millions of data could be efficiently queried, in a few seconds.

Proposed graph data model in this Chapter is the baseline of my thesis. Simulation of the data flow with all its connection between sensor nodes was well represented by the graph-based data model and all data stored in the graph database wa naturally indexed by relationships, which provides faster access to data as compared with a relational database.

Graph data model provides a flexible design for analyzing data based on relationships. While looking for a relationship between two nodes, graph model is obviously more efficient than other models as it is by nature of being a graph connected to each other with edges/relations. Therefore graph data analytics is the a effective way to explore complex relationships in data and mostly preferred. Since the focus of my dissertation is data analytics on surveillance applications, graph based model fits quite well for both wireless multimedia sensor network modeling and data flow from sensor node to sink node with all relationship information.

CHAPTER 4

DATA SIMULATION

By using the proposed graph-based data model and selected graph database in previous chapter, we developed a simulator to simulate wireless sensor network deployment and data flow on the network.

There are three main components of the simulator; network topology generator, sensor data generator, and scenario generator. The network topology generator is used to create a simulation for a multimedia wireless sensor network infrastructure which is composed of sensor nodes and gateways. Sensor data generator produces synthetic data with position, PIR, seismic and acoustic information with end-to-end data flow simulation from sensing at a sensor node to storage at the sink node. And, scenario generator component is developed to build an execute custom scenarios to generate experimental data for different use cases.

This chapter is organized as follows: the next section provides the information related to real wireless sensor network configuration. Section 4.2 gives the overview of the generic infrastructure, and system architecture is detailed in Section 4.3, respectively. In Section 4.4, components of the simulator is given. A case study for surveillance domain is described in Section 4.6 to realize the proposed model. Finally, conclusions are given in Section 4.7.

4.1 Reference WSN System

The reference WMSN system is composed of wireless multimedia sensors and some scalar sensors. The system is designed as a multi-tier automated surveillance system.

The first layer is the sensing layer with scalar sensors including acoustic, seismic, and PIR. The second layer is triggered by first layer. Multimedia sensors like camera and microphone are used capture video and audio. After applying the fusion, object type and location of the sensor is extracted to be provided to the next layer, which is called the sink layer. The sink layer provides the capability to do analytics on all collected and generated information in the network.

The overall architecture is given in Figure 4.1. Sensor nodes are connected to the gateway nodes via ZigBee (IEEE 802.15.4) interfaces. A special thread for serial messaging is developed to send the events from sensor nodes to the gateway. The gateway prepares XMPP messages using the gathered events coming from leading nodes via broadcast messages. Those XMPP messages are transferred to the sink node. The XMPP messages and multimedia data is transfered over IP based (IEEE 802.11) connection.

A Raspberry Pi (RPi) has been used as the hardware of a sensor node. Model B board with 512 MB has been used and includes these hardware components:

- CPU: ARM1176 700MHz

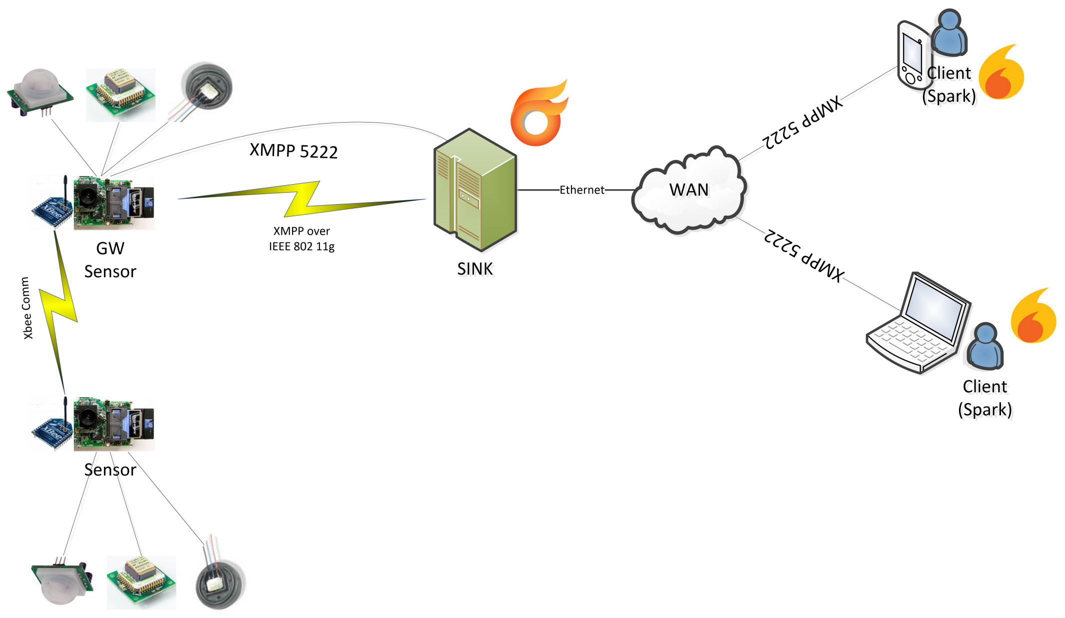


Figure 4.1: Reference WMSN System Architecture [1]

- RAM: 512 MB SDRAM (shared with GPU)
- Storage: SD Card slot
- Network: 10/100 Mb Ethernet
- Other interface: 2x USB 2.0 ports, Video and audio outputs, GPIO ports,
- Power: 5V 700-mA microUSB

The following list is the components installed on a node in the reference WSN system to fulfill its functions:

- Sensors: Vibration, Motion (PIR), Acoustic
- Camera: Raspicam
- Network: Xbee ZigBee (IEEE 802.15.4), Wi-Fi (IEEE 802.11)
- Power Source: 4400-mAh 5V 1A power bank
- Audio Input: Microphone
- Software: XMPP for instant message

Communication between sensor nodes, gateway and sink are established using ZigBee interfaces. All nodes are equipped with ZigBee (IEEE 802.15.4) which is based on low-bandwidth radio transmission. The line of sight of ZigBee can be up to of 1.5 Km. at outdoor applications and transmission rates can be around 250 Kbps at most.

In reference WSN system, sensor node and gateway roles are all predefined, there is not any dynamic gateway selection. Because different roles may need different kinds of hardware components.

4.2 Generic Infrastructure for Simulator

The simulator is developed by using Java 1.8 as maven projects to use Apache Maven as the dependency management framework. To develop the simulator application of this thesis beyond OrientDB, a generic infrastructure which can be easily adapted to other database systems was designed. To achieve that, business logic without any dependency to OrientDB was developed.

Table 4.1: Defined Message Queues

Queue Name	Process	Queue Element
Scalar Data	Collected Raw Data	SensorRawData
Fused	Level 1 and 2 Fusion	SnFusedData
Forward	Forwarding to Sink	GwFusedData
Action	Level 3 Fusion	SinkFusedData

There are two managers called DataManager and NetworkManager. DataManager defines the necessary interfaces to populate data for the underlying database. NetworkManager has the business logic to construct network topology and defines the necessary interfaces to create network entities for the underlying database.

As there is a data flow between sensor nodes and gateways and sink, in order to cope with the bottleneck of high throughput of streaming data, Apache ActiveMQ message queues was positioned between each process. Message queues are defined as seen on Table 4.1 below. Fusion logic is developed on top of messaging queues and there are 3 business logic handlers; Level1and2Fusion, GatewayForward and Level3Fusion.

4.3 System Architecture

To manage the big data and high throughput of many wireless multimedia sensors, a scalable system architecture supported by NoSQL databases and messaging queues was developed. Figure 4.2 shows a visualization of proposed multilayer system architecture composed of four layers; Detection Layer, Message Layer, Data Layer, and Analysis Layer. Each layer is responsible for different parts of the entire process of analyzing big data in the proposed architecture.

The sense layer is composed of sensor nodes, gateways, and a sink. The raw data generator, producing synthetic data with position, PIR, seismic and acoustic information, simulates the sensor nodes activated by a moving object. Raw data is created for each sensor node with the sensor data calculated as a function of the distance to the position of the moving object. The detected data is collected at the sink using

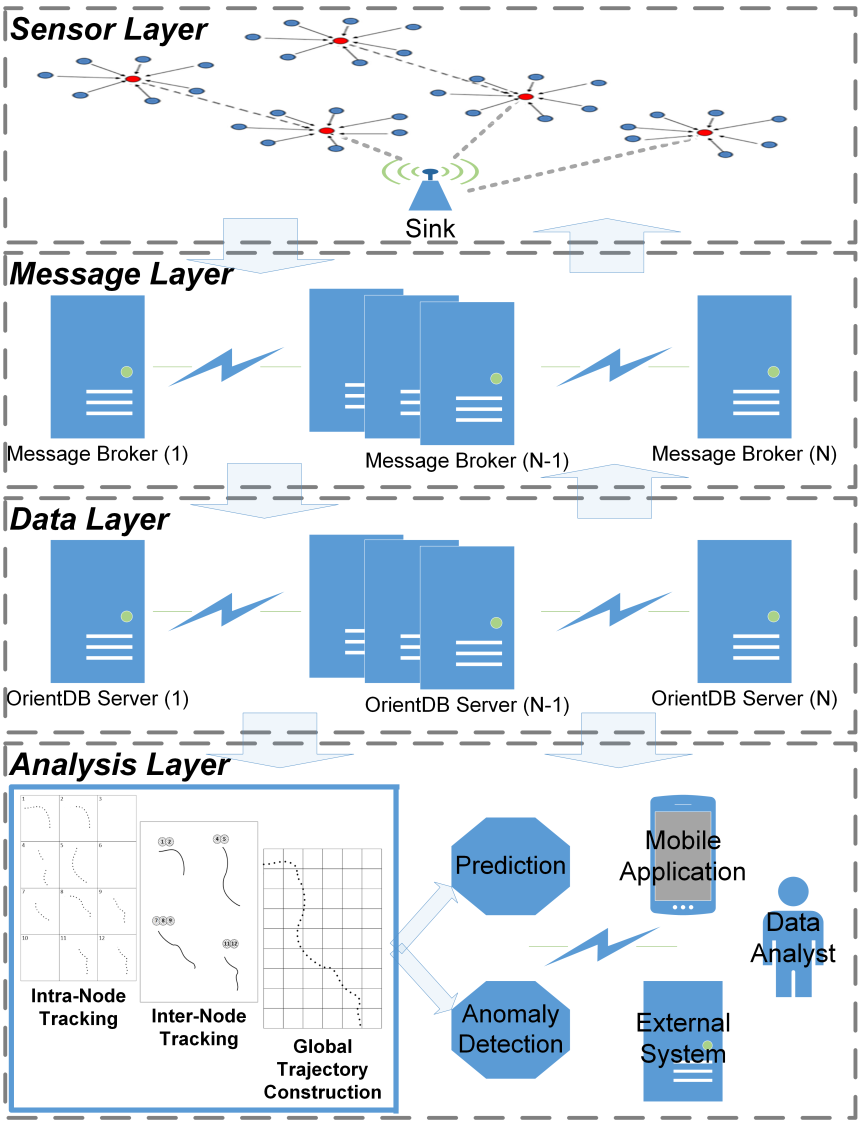


Figure 4.2: System architecture.

the message brokers that are positioned in the message layer. In order to cope with bottlenecks due to the high throughput of streaming sensor data, message queues are positioned between each process. The message data is orchestrated and sent to the data layer for retention by the message brokers.

In the data layer, NoSQL databases built using the OrientDB graph database system are clustered to process big graph data. The graph structure better supports connectivity analysis. Because object tracking is completely tied to the connection between each piece of data that is detected, graph models work better by analyzing the relationship between different sensor nodes or by identifying anomalies in a trajectory. Because the graph-based big data model allows you to query based on node attributes and relationships between them, a sophisticated prediction model can be developed and integrated into different domain applications. Since the streaming data is distributed by the message layer, the data layer must be aligned with the clustered big data architecture.

Finally, in the analysis layer, the data stored in the graph database is used for data analytics by the data scientist, the external system, or the mobile application for end-user reports and live portals. The object tracking algorithms, detailed in the last chapter of this dissertation, are executed in the analysis layer.

4.4 Simulation

To develop and experiment the graph data model, a simulator was developed which is mainly focused on data simulation but also supports network topology simulation for both grid and random node deployment.

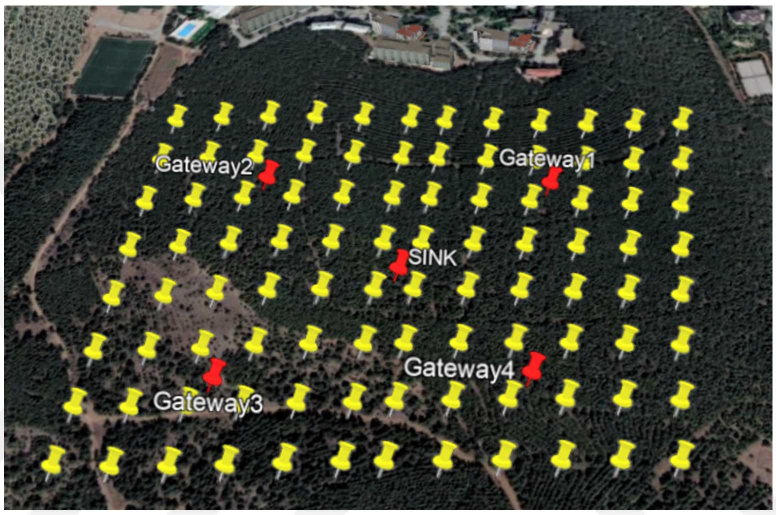
4.4.1 Network Topology Simulation

For data simulation, first of all it is needed to deploy sensor nodes, gateways, and a sink node on a simulated area.

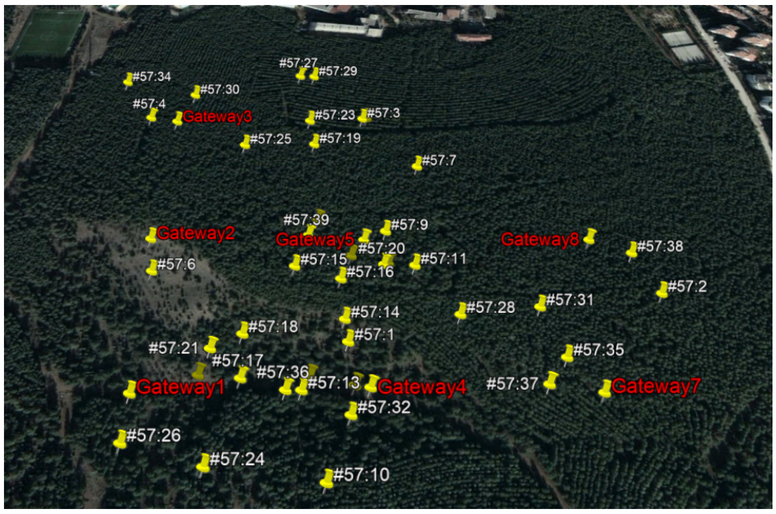
It is assumed that the WMSN is distributed with a grid layout in a simulation area. Grid layout employed in this thesis is commonly used in research studies [46, 47] as it is more convenient to monitor the whole area without any gap in between the sensor nodes.

A testbed which is part of the territory of the university is selected. Figure 4.3b shows a map view of the monitored area with 96 sensor nodes, composed of 4 different clusters with a gateway node and a sink node in the center. The distance between the two sensor nodes is estimated at about 60 meters.

Since main focus is the grid-based deployment of sensor nodes, the performance of



(a)



(b)

Figure 4.3: Simulated wireless sensor network. (a) Grid-based deployment. (b) Random deployment.

proposed approach was also measured for randomly distributed wireless sensor networks (Figure 4.3a). To make it more realistic and well distributed, 10% of the nodes are positioned distantly from each other and distribute the rest of the nodes without any limitation, except that two sensor nodes cannot be closer than 2 meters.

4.4.2 Data Flow Simulation

Sensor Data Generator produces synthetic data with position, PIR, seismic and acoustic data. The data flow is simulated as if the data were detected by sensor nodes, close to the position of the data generated. Raw sensor data is created for each sensor node with the scalar data calculated as a function of the distance to the position of the raw data. From a sensor node to a gateway and from a gateway to a sink node, the data stream comprising the multilevel data fusion process is simulated to generate more realistic data. In addition, the simulator uses the WGS84 geodetic datum, which represents latitude and longitude coordinates, to support real world scenarios.

The data flow occurs from the sensor nodes to gateways and from gateways to the other gateways (multi hop) and finally to the sink node. Each sensor node holds a set of data sensed by the sensors and camera of the node. Sensor nodes include embedded programs for handling correlation, transformation, and aggregation on the raw data, which is called first level fusion. The sensor fused data are reported to the leading gateway by all of its connected sensor nodes.

The gateway waits for all sensor nodes to report. When all reports are ready, the gateway applies an aggregation or filtering on the received data. The second-level fusion is done at this point and the output of the fusion is a summary of that cluster. The gateway fused data are forwarded to the sink node for a final decision.

Similar to the second-level fusion, the sink waits for all gateways' fused data. By applying some patterns to detect anomalies or other kinds of analysis are done at the third level fusion. The output of the last level fusion is an action like triggering an alarm or a notification message to another system.

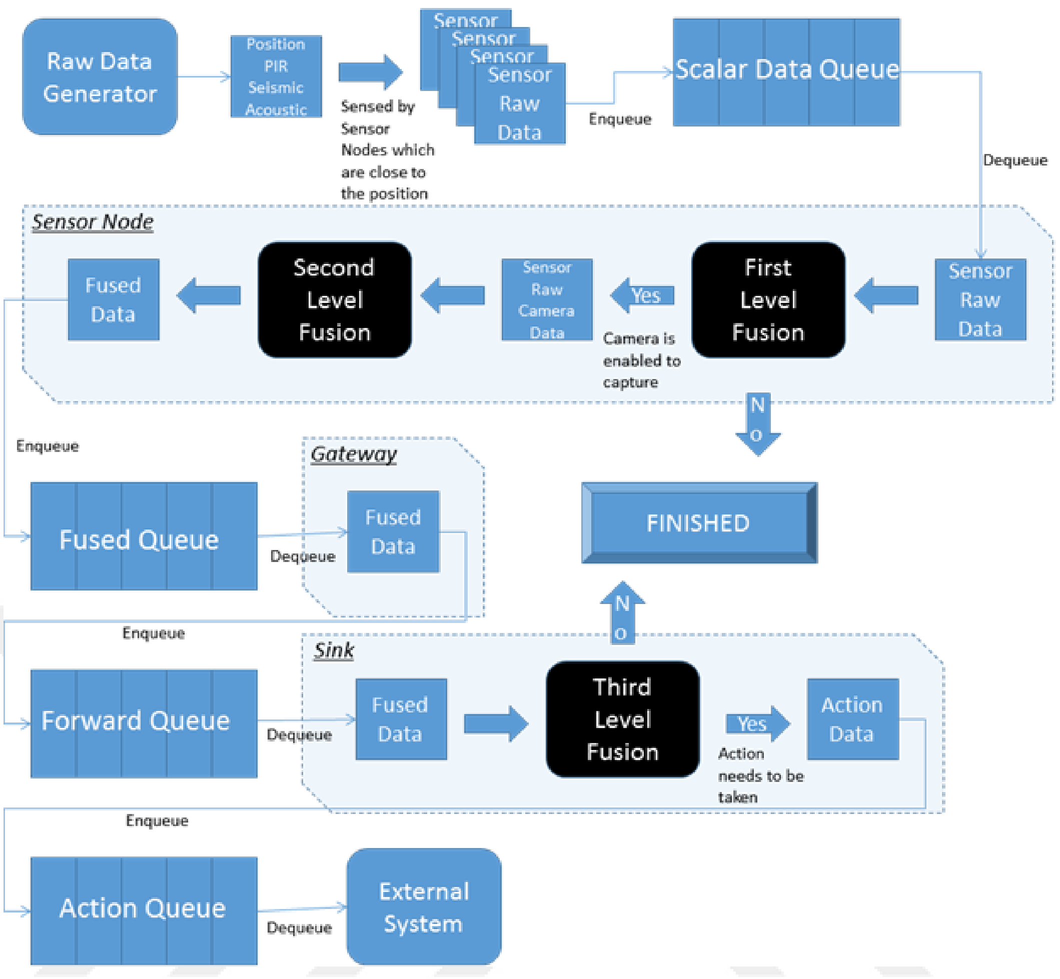


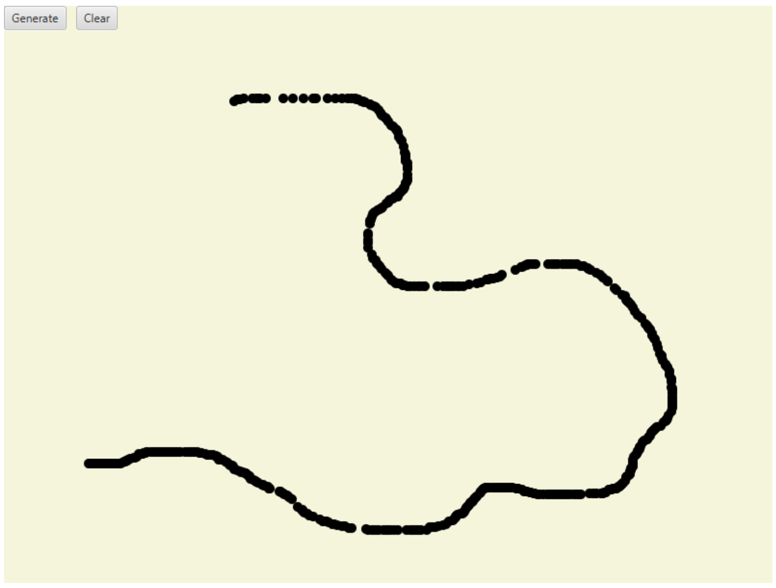
Figure 4.4: Data flow simulation.

4.4.3 Scenario Generator

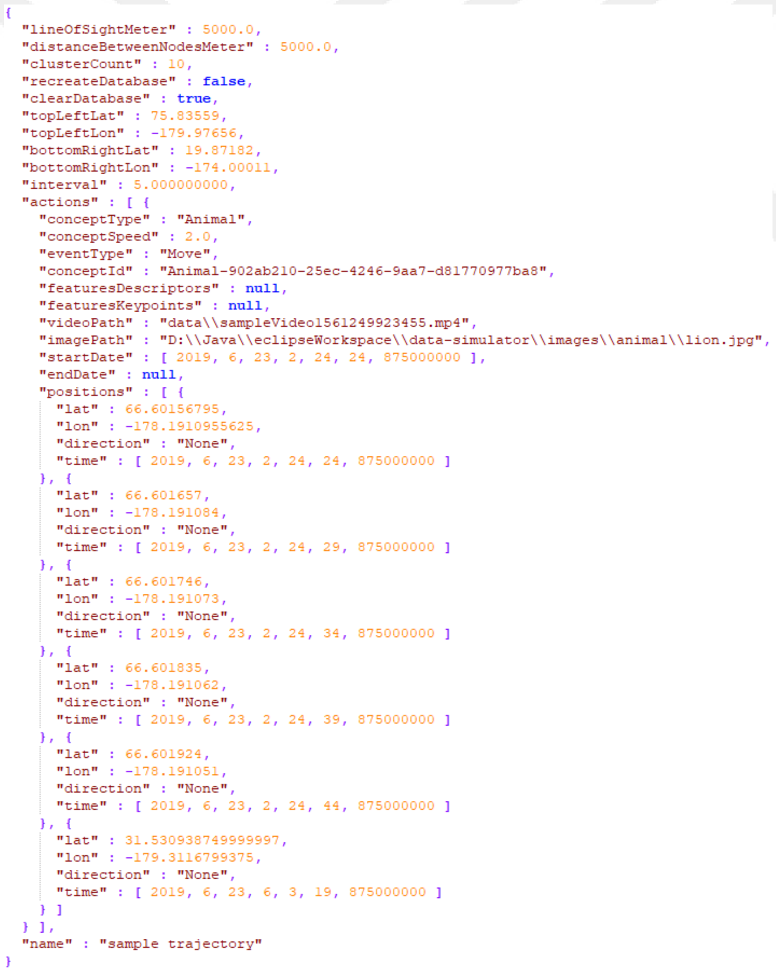
This component utilizes generation of custom use cases to be able to experiment various data. Scenarios can be created by both using a drawer and a JSON file (Figure 4.5a).

Scenario drawer provides a whiteboard which enables free style drawing using mouse or any pointer on touch screen monitors. It is possible to select type of the object, and specify date for the scenario. Then, you can generate the data for the scenario and export it as a JSON file.

A scenario can be modified by opening and editing the JSON file, or a new scenario can be created from scratch by writing the scenario information in JSON format into



(a)



```json
{
  "lineOfSightMeter" : 5000.0,
  "distanceBetweenNodesMeter" : 5000.0,
  "clusterCount" : 10,
  "recreateDatabase" : false,
  "clearDatabase" : true,
  "topLeftLat" : 75.83559,
  "topLeftLon" : -179.97656,
  "bottomRightLat" : 19.87182,
  "bottomRightLon" : -174.00011,
  "interval" : 5.000000000,
  "actions" : [ {
    "conceptType" : "Animal",
    "conceptSpeed" : 2.0,
    "eventType" : "Move",
    "conceptId" : "Animal-902ab210-25ec-4246-9aa7-d81770977ba8",
    "featuresDescriptors" : null,
    "featuresKeypoints" : null,
    "videoPath" : "data\\sampleVideo1561249923455.mp4",
    "imagePath" : "D:\\Java\\eclipseWorkspace\\data-simulator\\images\\animal\\lion.jpg",
    "startDate" : [ 2019, 6, 23, 2, 24, 24, 875000000 ],
    "endDate" : null,
    "positions" : [ {
      "lat" : 66.60156795,
      "lon" : -178.1910955625,
      "direction" : "None",
      "time" : [ 2019, 6, 23, 2, 24, 24, 875000000 ]
    }, {
      "lat" : 66.601657,
      "lon" : -178.191084,
      "direction" : "None",
      "time" : [ 2019, 6, 23, 2, 24, 29, 875000000 ]
    }, {
      "lat" : 66.601746,
      "lon" : -178.191073,
      "direction" : "None",
      "time" : [ 2019, 6, 23, 2, 24, 34, 875000000 ]
    }, {
      "lat" : 66.601835,
      "lon" : -178.191062,
      "direction" : "None",
      "time" : [ 2019, 6, 23, 2, 24, 39, 875000000 ]
    }, {
      "lat" : 66.601924,
      "lon" : -178.191051,
      "direction" : "None",
      "time" : [ 2019, 6, 23, 2, 24, 44, 875000000 ]
    }, {
      "lat" : 31.530938749999997,
      "lon" : -179.3116799375,
      "direction" : "None",
      "time" : [ 2019, 6, 23, 6, 3, 19, 875000000 ]
    } ]
  } ],
  "name" : "sample trajectory"
}
```

(b)

Figure 4.5: Scenario generation. (a) Using scenario drawer. (b) Using JSON file.

a file. Simulator provides the capability to import JSON files to generate data from the scenario file.

4.5 A Case Study: Surveillance Application

Surveillance systems need robust and scalable infrastructure. To achieve that, all data flow and data itself are needed to be analyzed and modelled.

A set of sensors and video cameras are needed to monitor the whole city. Assume that, sensors and cameras are clustered according to the districts and each cluster forwards sensed data to the HQ (Head Quarter) which is the operation center. At the HQ, an alarm is triggered, or a notification is sent to the officers to early detection of violence.

Sensor types can be seismic, acoustic and PIR (Passive Infrared) which are types of scalar sensors. In addition to them, video cameras or thermal cameras can be added to critical locations. As the default, cameras are switched off. According to the sensed information from scalar sensors, the predefined conditions can be extracted using rule-based approaches. The motion or environmental change may be detected and interpreted to activate the camera by providing a rough prediction of the moving object.

Moving object can e categorized as; Animal, Human, and Vehicle. The data collected from scalar sensors are analyzed to guess the category of the objects according to predefined thresholds (Table 4.2).

After activation of the camera by analyzing scalar sensor data, video and audio streams

Table 4.2: Sample Thresholds To Identify Objects

Object Type	PIR	Seismic (Hz)	Acoustic (dB)
Animal	True	5 - 20	5 - 30
Human	True	21 - 55	31 - 50
Vehicle	True	>35	>50

from the camera are started to be processed. That processing in the sensor node is called the first level fusion. After fusion, the concept of the moving object, and maybe even a silhouette, are revealed. Fusion output is reported to the gateway which is the leading node of that district.

A gateway is connected to a set of sensors and cameras. The described operations are done for all leaded sensors so that the gateway receives many concepts and silhouettes. By applying some algorithms like filtering the duplicate concepts or aggregating them to normalize the received data, the gateway accomplishes the second level fusion. After fusion, more accurate concepts and silhouettes are provided by many sensors. Fusion output is forwarded to the HQ (Headquarter) for a final decision.

As there are many districts in a city, there are many gateways which are far away from the HQ and not directly connected to it. In that case, data is forwarded over other gateways. At the HQ, the received data from all districts are analyzed, aggregated or filtered for the purpose of detecting anomalies or finding some patterns, which is called the third level fusion. At the end, the fusion comes to the conclusion and an alarm is triggered to the officers or the external system of the armed forces is notified.

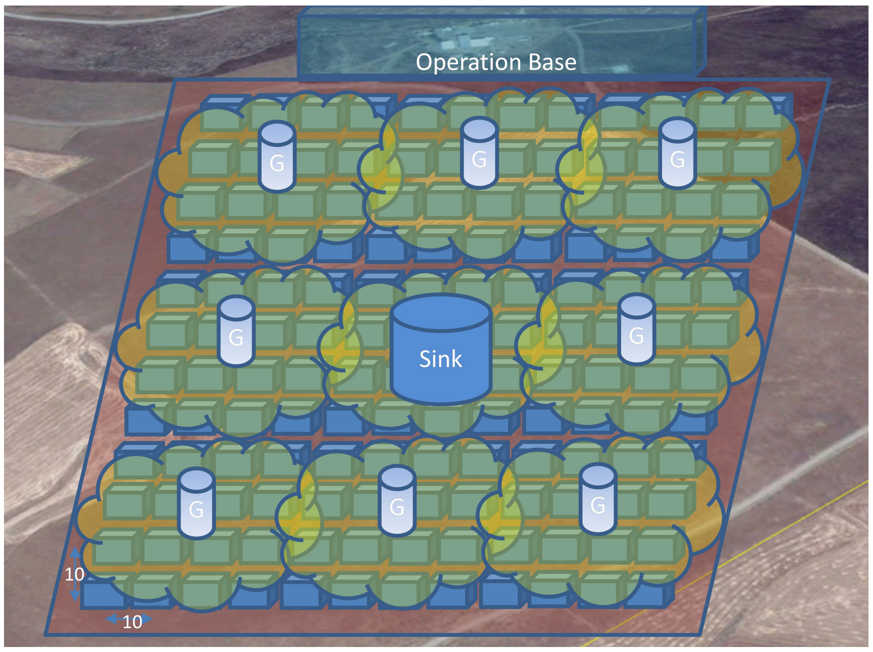
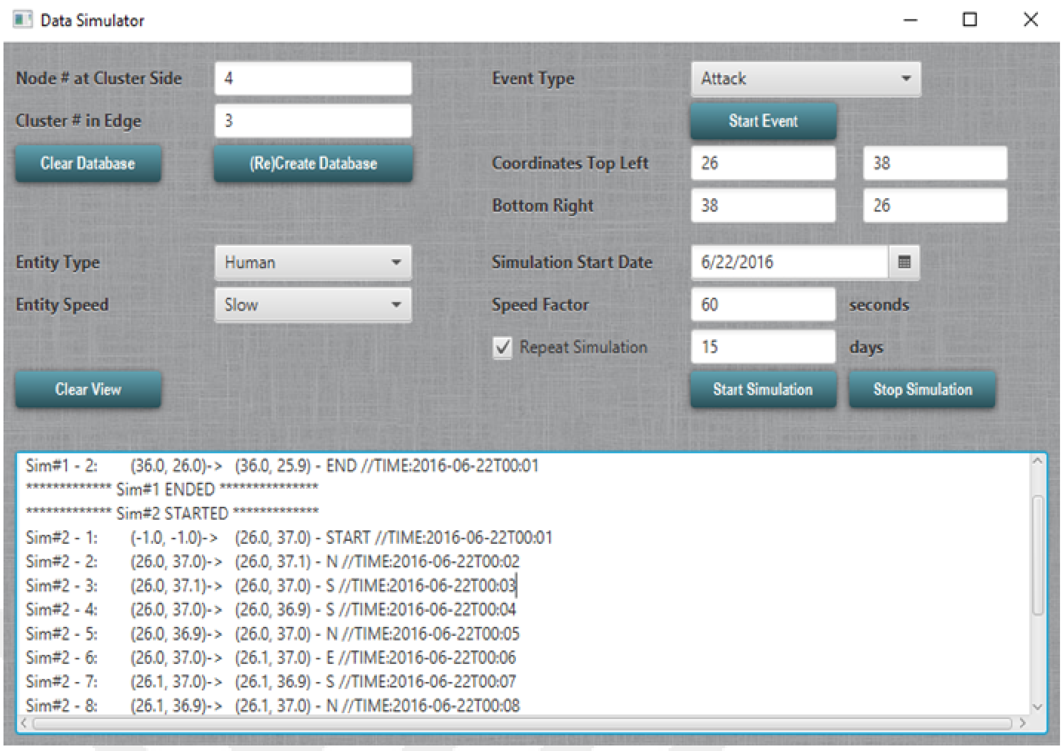


Figure 4.6: Surveillance application simulation.



Node # at Cluster Side: 4
Cluster # in Edge: 3
Event Type: Attack
Coordinates Top Left: 26 38
Bottom Right: 38 26
Entity Type: Human
Entity Speed: Slow
Simulation Start Date: 6/22/2016
Speed Factor: 60 seconds
[✓ Repeat Simulation] 15 days

```
Sim#1 - 2:    (36.0, 26.0)->  (36.0, 25.9) - END //TIME:2016-06-22T00:01
************* Sim#1 ENDED **************
************* Sim#2 STARTED *************
Sim#2 - 1:    (-1.0, -1.0)->  (26.0, 37.0) - START //TIME:2016-06-22T00:01
Sim#2 - 2:    (26.0, 37.0)->  (26.0, 37.1) - N //TIME:2016-06-22T00:02
Sim#2 - 3:    (26.0, 37.1)->  (26.0, 37.0) - S //TIME:2016-06-22T00:03
Sim#2 - 4:    (26.0, 37.0)->  (26.0, 36.9) - S //TIME:2016-06-22T00:04
Sim#2 - 5:    (26.0, 36.9)->  (26.0, 37.0) - N //TIME:2016-06-22T00:05
Sim#2 - 6:    (26.0, 37.0)->  (26.1, 37.0) - E //TIME:2016-06-22T00:06
Sim#2 - 7:    (26.1, 37.0)->  (26.1, 36.9) - S //TIME:2016-06-22T00:07
Sim#2 - 8:    (26.1, 36.9)->  (26.1, 37.0) - N //TIME:2016-06-22T00:08
```

Figure 4.7: Data simulator application.

To be able to cover the whole area, nodes are usually distributed with a grid layout. Figure 4.6 shows a simulation for surveillance of an HQ with 16 sensor nodes in each cluster that has a gateway in the middle, and 9 gateways in total with the sink node in the center. Between two sensor nodes, the distance is simulated as 10 meters.

4.6 Simulator Application

Figure 4.7 is the screenshot of the simulator application to generate the network topology in Figure 4.6. "(Re)Create Database" button cleans the database and generates sink, gateways and sensor nodes according to the given parameter related to node count and cluster count.

"Start simulation" button starts simulation by sending an entity from one of the edges of the area. Then, the entity moves randomly according to its speed and simulation ends when the entity moves out of the area. Possible moves are going to north, south, east, or west and don't move.

It is possible to run parallel simulations by clicking the button at any time. And if "Repeat Simulation" checkbox is checked, a new simulation is automatically started when current simulation ends.

There are several Entity types to simulate. Each entity has its own speed, acoustic and seismic values. Table 4.3 show each type and its simulation values. "Entity Speed" selection combobox decreases or increases the speed value of the simulating entity. Acoustic and seismic values are defined by the selection of "Entity Type" combobox and according to entity's distance from the sensor node, those values are recalculated to degrade its effect on the sensor node.

Assume that simulation put an "Animal" entity at (37,120) position. The entity is between the sensor nodes at (30,120) and (40,120). Table 4.4 and Table 4.5 show the sensed values between those two nodes for Animal Type.

Another important capability of the simulation tool is Event generation. Events can be generated at any time while the simulation is running. Currently, there are two types of events.

- Attack: As seen in Figure 4.6, operational base is located at north. If something comes from south and directly moves toward the base, this movement is an Attack event for us.
- Smuggling: If a group of human is moving together with a group of animal and they are coming from west and going in the direction of south-east, this movement is smuggling event for us.

Table 4.3: Simulated Entity Types

Entity Type	Speed	Acoustic	Seismic
Human	1	20	10
Animal	2	40	20
Vehicle	4	70	80
GroupOfHuman	1	60	30
GroupOfAnimal	2	80	60

Table 4.4: Sensed Values at Node(30,10)

ANIMAL	30,10	31,10	32,10	33,10	34,10	35,10	36,10	37,10	38,10	39,10	40,10
Acoustic	20	18	16	14	12	10	8	6	4	2	0
Seismic	40	36	32	28	24	20	16	12	8	4	0

Table 4.5: Sensed Values at Node(40,10)

ANIMAL	30,10	31,10	32,10	33,10	34,10	35,10	36,10	37,10	38,10	39,10	40,10
Acoustic	0	2	4	6	8	10	12	14	16	18	20
Seismic	0	4	8	12	16	20	24	28	32	36	40

When “Start Event” button is pressed, selected event is started to be simulated. Additional event types can be added for further analysis.

4.7 Remarks

The chapter is the about the simulation which includes a simulator to produce synthetic big sensor multimedia data and the simulation infrastructure which represents the objects moving in the multimedia sensor networks.

A wireless sensor network with millions of data was simulated to test the proposed graph data model. The query performance was tested with many complex scenarios, and it was shown that generated millions of synthetic data can be efficiently queried on the graph data model.

CHAPTER 5

OBJECT TRACKING AND TRAJECTORY ANALYTICS

A new object tracking method for surveillance applications is proposed using a big data model and a multilevel fusion. The proposed approach is based on three main algorithms: intra-node fusion, inter-node fusion, and object trajectory construction.

Intra-node fusion exploits the detection and tracking of objects in each sensor. The output of the intra-node fusion is used with spatiotemporal data as well as neighboring sensors to generate inter-node trajectories. Then, the fused data from all the sensor nodes are combined to construct global trajectories for the detected objects in the monitored area on the WMSN. The proposed method tracks moving objects, such as vehicles, animals and humans, using wireless multimedia sensors for surveillance purposes.

This chapter is organized as follows: next section provides an overview. Section 5.2 details the proposed object tracking approach, and extracted trajectories are used for analytics in Section 5.3.3, respectively. Interval Type-2 Fuzzy Logic usage for trajectory extraction and analytics is provided in Section 5.3. Finally, remarks are given in Section 5.5.

5.1 Overview

Being very related to this chapter, the issue of multi-level track fusion has also been studied by a number of researchers [48, 49, 50]. In the first level, an object is detected from an image and calculate it's location. Then trajectories generated by sensor nodes in the same local area are fused. At the final step, global trajectories are constructed.

The main difference between this thesis and the previous studies mentioned here is that all of these studies use track identification created in the first level and continues on based upon that assumption, but a more realistic and challenging approach which don't use any pre-identification is proposed. All trajectory information, like spatiotemporal data, speed, direction and low-level features are extracted to be able to track objects.

Regarding the object tracking approaches in Wireless Sensor Networks, Fayyaz [51] presents a good survey on them. He categorizes tracking according to network architecture, the algorithms used for tracking, sensor types, number of tracked objects and wireless communication technologies. On the other hand, Mazimpaka and Timpf [52] focus on tracking aspect of the trajectory mining methods and applications. They point out that coping with massive big data is an open research issue for object tracking. Valsamis et al. [53] compare a couple of predictive analytics on real-time big data for trajectory prediction in the literature. They only focus on the maritime data for vessels which are real-time spatiotemporal time series. Besides, they used multi-scan machine learning algorithms to train their models.

Trajectory modeling and prediction have been studied by many researchers [54, 55, 56]. Chengyang et al. [57] modeled trajectories in a way that all original trajectories transformed into fixed-length sequences. On the contrary, this thesis supports using various lengths of trajectories. Another study proposed STWalk [58] which learns trajectories in spatiotemporal data graphs by traversing the graphs based on the timestamps. Yufan et al. [59] developed a method based on the long-short term memory (LSTM) prediction and used the detected trajectories in order to detect anomalies. This thesis differentiates from other works since it uses a novel interval type-2 fuzzy logic based approach for trajectory extraction, and not only the anomaly detection, but also the trajectory prediction has been studied.

5.2 Object Tracking

The proposed object tracking approach was developed specifically for the surveillance requirements, but it can also be easily adapted to other application areas. An object's tracking is mainly focused on detecting an object when it enters the surveillance zone, from the moment it enters the zone until it leaves it. The tracking begins when the object is detected by a sensor node using its physical sensors. According to the algorithm implemented on the sensor nodes, a camera is activated if an object is detected or if it is a false alarm caused by noise in the environment. Next, a snapshot of the detected object is taken to ensure the possible presence of an object. The snapshot of the connected camera is important for accurately detecting and then tracking objects, as this multimedia data is used to estimate the approximate position of the detected object using object localization algorithms, which use the position of the sensor node, including ground height, camera lens specifications, and camera viewing angle.

The challenge of tracking an object without an associated identifier, such as GPS or RFID, is that each physical event occurring on a sensor deployed on a node is completely anonymous. Main goal was to look for relationships and correlations between entire anonymous sensor data and to identify not only the objects but also their movement over the time passing through the surveillance zone.

The main idea of the proposed object tracking approach is to make use of the position and time of the sensor node in accordance with the fusion outputs, i.e., the concept types, the features of the low-level, and speed of the object to solve the problem of whether the object has already been detected or whether it is a completely new object. The proposed object tracking algorithm solves this problem by using the multilevel fusion approach with fuzzy rules. The first level of the fusion is the intra-node fusion which processes data for each sensor node and the second level is the fusion between the neighbors of a sensor node. Finally, the third level fusion is the construction of the object's trajectory in order to finalize the tracking and to extract the global trajectories of the objects. In the following subsections, the algorithms as well as the fuzzy rule engine used in each fusion level are explained.

5.2.1 Intra-Node Fusion

Intra-node fusion is the first level trajectory fusion that uses the local trajectories detected by a sensor node. The last point of each local trajectory is analyzed with the current sensor data to check if it corresponds to the existing trajectories, which are stored as time-ordered trees.

An object moving around a sensor node triggers the sensors equipped on the node and creates sensor data. The sensor data represents an actual object and it is examined as if it belongs to any existing local trajectories, which means that it is currently being tracked by the sensor node, or the new sensor data initiates a new trajectory to be added into the local trajectories store, which means that a new tracking is being started. The rule engine uses time, direction, distance and velocity to make a decision about whether new sensor data belongs to a new or an existing trajectory. There can be multiple objects in a trajectory. In that case objects are differentiated using the time, speed and direction of the objects.

During the analysis, the geodesic distance between the current object and the last object of the trajectory is calculated. The direction of movement and the elapsed time between two objects are used during the analysis. The speed of the object is referenced according to the object type, but the actual speed is calculated using the elapsed time and distance.

5.2.2 Inter-Node Fusion

Inter-node fusion is the second level trajectory fusion that looks for trajectory correlations between different sensor nodes. Each node is compared to its neighbor nodes. Moreover, only the nodes in the same direction of the compared trajectory are considered to optimize the performances.

The inter-node fusion algorithm is presented in the Algorithm 5. This algorithm mainly merges the trajectories identified by the sensor node s and its neighbor nodes. Trajectories from two sensor nodes are compared to identify if they can be fused or not.

5.2.3 Rule Engine

A generic rule engine for intra-node and inter-node fusions was designed and developed. The rule engine applies a list of rules on a given object for tracking purposes.

Algorithm 4 Intra-Node Fusion Algorithm

```
 1: procedure INTRANODEFUSION($n, TT[]$)
    ▷ The trajectory tree-list ($TT[]$)] is used to identify if the sensed object ($n$) is
    already being tracked or a new object.
 2:     $matchFound \leftarrow false$                                    ▷ Initialize variable
 3:     for each trajectory $T$ in $TT[]$ do
 4:         $last \leftarrow$ Last sensed item in the trajectory $T$
 5:         $score \leftarrow ruleEngine(last, n)$
 6:         if $score > \lambda$ then
    ▷ New data belongs to an existing trajectory
 7:             $matchFound \leftarrow true$
 8:             insert $n$ into $T$
 9:             exit loop
10:         end if
11:     end for
12:     if $matchFound = false$ then                                  ▷ New trajectory found
13:         $T \leftarrow \{\}$
14:         insert $n$ into $T$
15:         insert $T$ into $TT[]$
16:     end if
17: end procedure
```

Algorithm 5 Inter-Node Fusion Algorithm

```
1: procedure INTERNODEFUSION($n, s, \epsilon$)
2:     $N[] \leftarrow findNeighbors(s)$
3:     $\theta \leftarrow findDirection(L_s, L_n)$
4:     for neighbor node $sn$ in $N[]$ do
5:         $\theta_s \leftarrow findDirection(L_s, L_{sn})$
6:         if $\theta_s$ in the same direction with $\theta$ then
7:             $T_s[] \leftarrow$ Trajectories of sensor node $s$
8:             $T_{sn}[] \leftarrow$ Trajectories of sensor node $sn$
9:             for each trajectory $t_s$ in $T_s[]$ and $t_{sn}$ in $T_{sn}[]$ do
10:                $M_t \leftarrow$ Merge trajectories $t_s$, $t_{sn}$
11:                $p \leftarrow \emptyset$
12:                $c \leftarrow$ Last sensed item in the trajectory $M_t$
13:                $same \leftarrow true$
14:                for the last $\epsilon$ items do
15:                    $p \leftarrow$ Previous item of $c$ in the trajectory $M_t$
16:                    if $p$ is $null$ then
17:                        exit loop
18:                    end if
19:                    $score \leftarrow ruleEngine(p, c)$
20:                    if $score < \lambda$ then
21:                        $same \leftarrow false$
22:                        exit loop
23:                    end if
24:                    $c \leftarrow p$
25:                end for
26:                if $same = true$ then
27:                    Apply merged trajectory $M_t$ in $T_s[]$ and $T_{sn}[]$
28:                end if
29:            end for
30:        end if
31:    end for
32: end procedure
```

Each rule is classified in a category that has a positive or negative effect on the score of the object being processed. Specifying different rules for the rule engine enables the use of the rule engine by different algorithms and fusion levels.

5.2.3.1 Definitions

A rule (R) is a representation of an object state written in the form of a condition statement that gives a Boolean value (true / false). Each rule is associated with a rule category and defined as follows:

$$R = [cond, cat] \tag{5.1}$$

where

- $cond$ is the rule condition
- cat is the rule category

A category of rules (RC) is an abstraction based on the basic properties of objects moving along trajectories. Categories allow us to group rules and assign different priorities between rules. A rule category is defined as follows:

$$RC = [e, \varrho] \tag{5.2}$$

where

- e is the identifier for positive or negative effect on the score
- ϱ is the weight of the rule category

The score (S) is the output of the rule engine that is calculated by applying all the rules for the movement of a detected object. The score value is an aggregation of all category scores and calculated using the following equations:

$$S = \sum_{n=1}^{i} f(\sum_{m=1}^{j} g(R_m, RC_n), RC_n) \tag{5.3}$$

where i is the number of rule categories, j is the number of rules and,

$$g(R, RC) = \begin{cases} 0, & \text{if R.cat} \neq \text{RC} \\ 0, & \text{if R.cat} = \text{RC}, \text{R.cond} = \text{false} \\ 1, & \text{if R.cat} = \text{RC}, \text{R.cond} = \text{true} \end{cases}$$

$$f(s, RC) = \begin{cases} \text{s} \times \text{RC}.\varrho, & \text{if RC.e} = \text{positive} \\ \text{-s} \times \text{RC}.\varrho, & \text{if RC.e} = \text{negative} \end{cases}$$

5.2.3.2 Rule Generation and Implementation

A fuzzy logic rule generator that provides fuzzy rules, generated semi-automatically using trained trajectories, was developed, and then the generated rules were fine-tuned by experts.

A fuzzy inference system design and optimization tool, Fuzzy Inference System Professional (FisPro) [60, 61], was used to generate and induce fuzzy rules using training trajectories, and then the generated rules were adapted by assigning rule categories by experts. FisPro is an open-source application to create fuzzy inference systems in order to use those systems for reasoning. A fuzzy inference system (FIS) is a powerful interface between symbolic and numerical spaces. The ability of FIS to integrate the human expert knowledge with its fuzzy rules is one of the key aspects of the success of fuzzy systems. Besides, FIS represents the behavior of the system in a human-understandable way.

As shown in Figure 5.1, the fuzzy logic system has four basic steps: fuzzification, rule evaluation, knowledge base, and defuzzification. During the fuzzification stage, the actual inputs are converted to fuzzy membership functions. The fuzzy logic system inputs are being processed by the rule evaluation process, which uses the knowledge base to combine fuzzy inputs and produce outputs mapped to fuzzy membership functions. The knowledge base provides the list of rules in the application domain. The challenge is to design the correct list of rules for the problem. To deal with it, experts were used to filter and organize the rules generated automatically. Finally, in the defuzzification stage, the fuzzy outputs of the rule evaluation step are mapped to crisp

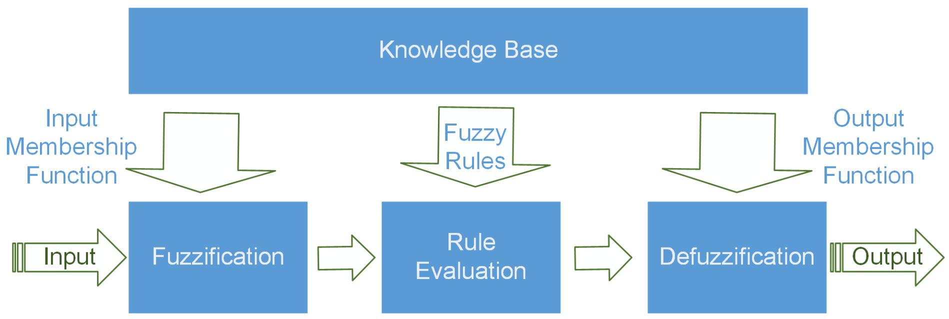


Figure 5.1: Fuzzy logic system.

outputs.

Input member functions are created for each rule category. Figure 5.2 shows examples of fuzzy input member functions that are used to generate fuzzy rules. Crisp location data in meters can be mapped to fuzzy values, such as "no change", "small change" "medium change" or "big change". Similar to the change of location, the time difference between two sensor data is compared and mapped to fuzzy values. Domain experts were involved to define the boundaries of input member functions.

Below is a list of sample fuzzy rules which were used in the rule engine for fusion. The full list of fuzzy rules can be found in Appendix A.

- IF sameDirection AND newTime AND smallChange AND humanSpeed THEN sameTrajectory
- IF sameDirection AND newTime AND bigChange AND humanSpeed THEN sameTrajectory
- IF sameDirection AND newTime AND noChange AND humanSpeed THEN sameTrajectory
- IF differentDirection AND realtime AND noChange AND humanSpeed THEN sameTrajectory
- IF sameDirection AND realtime AND bigChange AND vehicleSpeed THEN sameTrajectory
- IF sameDirection AND oldTime AND bigChange AND animalSpeed THEN sameTrajectory

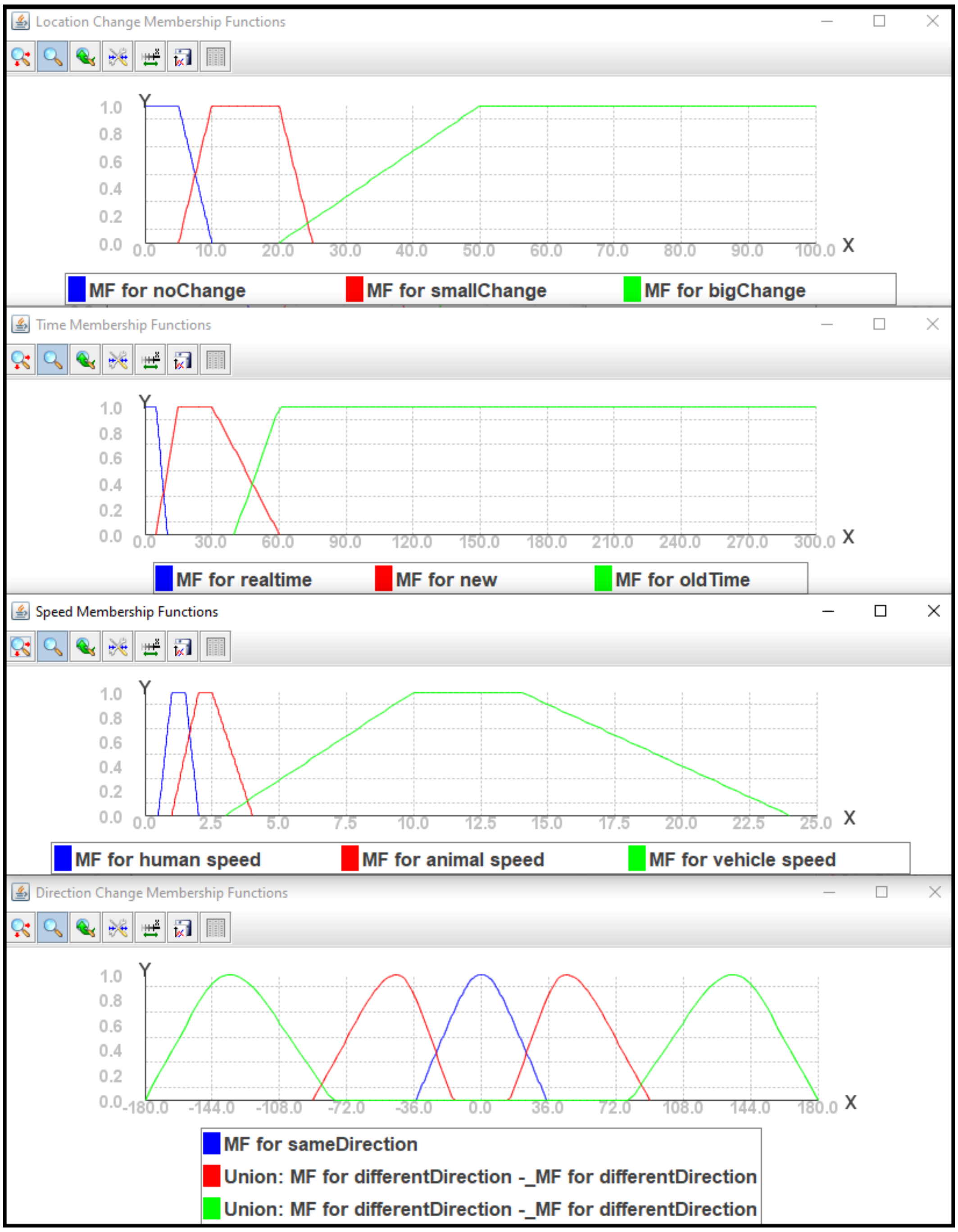


Figure 5.2: Sample fuzzy member functions.

- IF sameDirection AND newTime AND bigChange AND animalSpeed THEN sameTrajectory
- IF sameDirection AND newTime AND smallChange AND animalSpeed THEN sameTrajectory

FisPro tool was used for rule generation and several rule induction methods are available in the tool. The OLS algorithm [62] converts the input data into a fuzzy rule. Then it selects the convenient rule with the least squares criterion using the linear regression as well as the Gram-Schmidt orthogonalization. After this step, an optimization is made by applying a second pass to conclude the selected rules.

Finally concluded rules are linked to one of the rule categories described below:

- Direction: The direction of movement of the detected object is used in the rules. (e.g., are they moving on the same direction?)
- Time: The time-based rules are associated with this category. (e.g., has the object been seen in last 2 hours?)
- Velocity: The speed of the object is used in the rules to calculate its possible location and decide whether it moves or waits in the same location. (e.g., how far this object can go with this speed in that specific time?)
- Feature: The low-level SIFT [63] features of silhouette image are used in the rules. (e.g., check the similarity of silhouette images!)
- Concept Type: The type of object concept is used in some rules. (e.g., check if the concept types are identical or not?)
- Distance: The distance-based calculations are used to create rules. (e.g., is it possible to move from previous location to current location in that specific time?)

5.2.3.3 Sample Scenario Execution

Suppose there are two sensor nodes SN_1 and SN_2 positioned in (2,0) and (2,4) in a grid-based area. Each sensor node maintains a local track store for recognized trajectories used in track fusion calculations.

Let the trajectory T_1 is in the local track store of the sensor node SN_1 while T_2 and T_3 are in SN_2. Three trajectories can be defined as arrays of location and time pairs like below;

- T_1 = { [(0,4)-12:00:00], [(4,4)-12:00:02] }
- T_2 = { [(4,2)-12:00:01], [(3,0)-12:00:07] }
- T_3 = { [(0,2)-12:00:06], [(2,3)-12:00:08] }

Figure 5.3 shows an example scenario area with sensor nodes (SN_1, SN_2) and recognized trajectories (T_1, T_2, T_3). The red dashed line represents the trajectory of the moving object (T_{OBJ}) where (3,3) is the last location of the object at 12:00:09.

Suppose there are three categories of rules and for each rule category, a rule is defined in the rule engine. The categories of rules can be defined as follows:

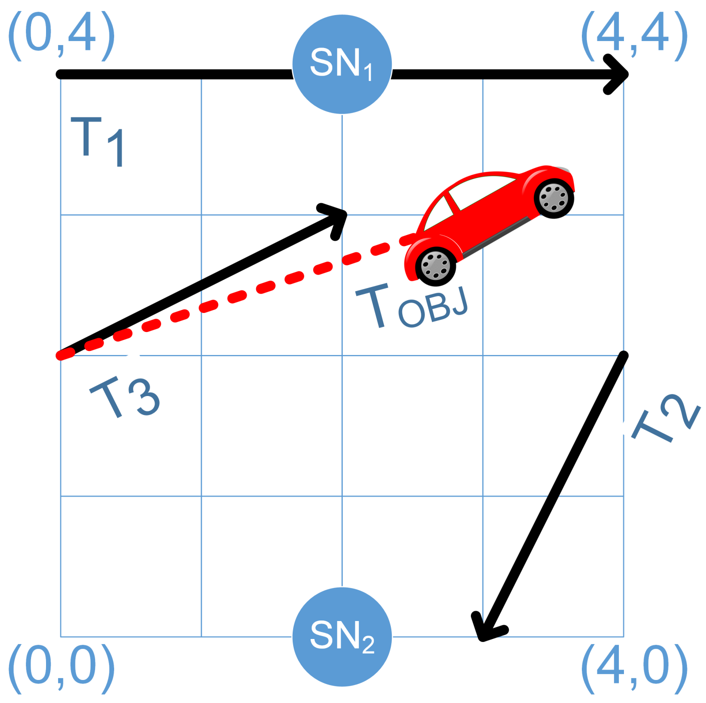


Figure 5.3: Rule engine execution on a sample scenario.

- ***RC-1:*** Velocity
  e: Positive(+), ϱ: 0.4
- ***RC-2:*** Direction
  e: Positive(+), ϱ: 0.3
- ***RC-3:*** Time
  e: Negative(-), ϱ: 0.5

The rules can be defined as follows (refer to the Algorithm 6 for the variable definitions in *cond*);

- ***R-1:*** Check if the distance between the previous location and the location of the newly detected object is within the maximum and minimum limits that can be moved by the newly detected object
  cond: $d <= v_{max} \times \mathit{\Delta} t$ AND $d >= v_{min} \times \mathit{\Delta} t$
  cat: RC-1
- ***R-2:*** Check if the relative direction of the location of the newly detected object from the previous location is in the same direction as the trajectory
  cond: $\theta = p.direction$
  cat: RC-2
- ***R-3:*** Check whether the previous detection time is obsolete, as if it is earlier than the detection time of the new object movement
  cond: $\mathit{\Delta} t > 5$ (5 minutes)
  cat: RC-3

Table 5.1 shows the computed values of parameters which are used in rule conditions. *d* represents the distance between trajectory's last object and the moving object. *v* represents the velocity calculated by location and time pairs of the trajectory. For this sample scenario, v_{max} and v_{min} are computed by using a fixed 30% upper and lower bound on *v*. θ is the relative direction between trajectory's last object and moving object.

Finally, execution of the scenario shown in Figure 5.3 using the rule engine results the scores in Table 5.2 for each trajectory. According to the final scores, the new object

Algorithm 6 Rule Engine Algorithm

```
1: procedure RULEENGINE($p, c$)
   ▷ The last object in the trajectory, the previous object ($p$), and the current object ($c$) are compared to indicate whether the current object belongs to the trajectory or not.
2:     $L_c \leftarrow (c.x, c.y)$                              ▷ Location of the current object
3:     $L_p \leftarrow (p.x, p.y)$                              ▷ Location of the previous object
4:     $d \leftarrow distanceOnGeoid(L_c, L_p)$
5:     $\theta \leftarrow findDirection(L_c, L_p)$
6:     $v \leftarrow (p.vx, p.vy)$
7:     $\varDelta t \leftarrow p.t - c.t$
8:     $R[] \leftarrow$ List of semi-automatically generated rules
9:     $S[] \leftarrow \{\}$                                   ▷ Used for rule category scores
10:    for each rule $r$ in $R[]$ do
11:        $valid \leftarrow r.process(d, \theta, v, \varDelta t, \varepsilon)$
12:        if $valid = true$ then                              ▷ Rule is accepted
13:            $i \leftarrow$ Index of rule $r$'s category in S[]
14:            $S[i] + +$
15:        end if
16:    end for
17:    $score \leftarrow 0$
18:    for each rule category $ct$ do
19:        $i \leftarrow$ Index of rule category $ct$ in S[]
20:        if $ct.e = Positive$ then
21:            $score \leftarrow score + (ct.\varrho * S[i])$
22:        else
23:            $score \leftarrow score - (ct.\varrho * S[i])$
24:        end if
25:    end for
26:    return $score$
27: end procedure
```

Table 5.1: Rule Condition Parameter Values for Each Trajectory

Parameter	T_1	T_2	T_3
d	1.4	3.0	1.0
v	2	0.4	1.1
θ	SW	N	E

Table 5.2: Sample Rule Engine Execution Scores for Each Trajectory

	T_1	T_2	T_3
R-1	0.4	0.0	0.4
R-2	0.0	0.0	0.3
R-3	-0.5	0.0	0.0
Score	-0.1	0.0	0.7

belongs to the trajectory T_3, which has a higher probability for the trajectory of the new object than the other possible trajectories.

5.2.4 Object Trajectory Construction

The third level of the fusion is to cluster and construct the object trajectories generated at the inter-node fusion level using the rule engine outputs. Some inter-node trajectories may be fused into other trajectories because they may overlap or be complementary to one another. The purpose of this last level of fusion is to detect the final global trajectories in the monitored area. Global object trajectories can be used as input for intra-node and inter-node fusion. Another use is to detect anomalies if a newly detected object trajectory does not correspond to any learned global trajectory. The clustering algorithm is an unsupervised learning algorithm in which relationships are discovered from an unlabeled dataset. The results of previous detections are used as input parameters for future calculations.

The construction of global trajectories is shown in Algorithm 7, and the main idea of the algorithm is based on the calculation of Hausdorff Similarity [64] between

Algorithm 7 Object Trajectory Construction Algorithm

```
1: procedure OBJECTTRAJECTORYCONSTRUCTION($T[], GLB[]$)
   ▷ Global object trajectories ($GLB$) are constructed using the inter-node fusion generated trajectories ($T$).
2:     for each trajectory $t$ in $T[]$ do
3:         if $t$ is a short trajectory then
4:             Merge $t$ with other trajectories
5:         end if
6:     end for
7:     for each global trajectory $gt$ in $GLB[]$ do
8:         $ML[] \leftarrow \{\}$
9:         $T[] \leftarrow similarLengthTraj(GLB[], gt.length)$
10:        for each trajectory $t$ in $T[]$ do
11:            $m \leftarrow hausdorffSimilarity(gt, t)$
12:            if $m > SimilarityThreshold$ then
13:                Mark as candidate trajectory
14:                insert $t$ into $ML[]$
15:            end if
16:        end for
17:        if $ML[].size > GlobalTrajectoryThreshold$ then
18:            $glbTraj \leftarrow fastDTWCluster(ML[])$
19:            insert $glbTraj$ into $GLB[]$
20:        end if
21:    end for
22: end procedure
```

trajectories with similar lengths. The calculated value is normalized in the interval [0, 1]. Higher measures indicate a high degree of similarity and it can be identified that they can be grouped together in a global candidate trajectory. If several candidate trajectories are detected, they can construct a global trajectory.

K-Nearest Neighbors (kNN) algorithm has been used by many studies for trajectories [65, 66, 67, 68]. It is a supervised classification algorithm and can be used for either prediction or extraction of the trajectories. A trajectory is compared with the other trajectories based on the Euclidean distance, but in our approach we are using our rule based approach in order to use additional input parameters like speed, direction and time as well as the distance.

Since the length of trajectories can be different, the Dynamic Time Warping (DTW) distance has been used to calculate the trajectory similarities. DTW aligns trajectories by warping the time axis continuously till an optimal similarity has been detected. If there is a high similarity, then centroids of the global trajectories were computed by using the standard K-Means way.

Figure 5.4 shows how to detect a step by step trajectory using the multilevel fusion approach. Suppose there are 12 sensor nodes that monitor an area and an object passes through the surveillance zone (a). In order to describe the detection of a trajectory of an object, a metaphor of the puzzle game can be used. The inter-node fusion algorithm detects many pieces that are connected, but they belong to different parts of the puzzle. Therefore, the algorithm is not aware of the situation as a whole (b). Each sensor node detects and fuses its local data, a range limited to a single node. Intra-node fusion is a phase in which you begin to merge different groups and form shapes or images in the puzzle, but the entire image is still missing (c). The sensor data from many sensor nodes are fused in the last step and finally the construction of the overall object trajectory is performed. Thus, with this last step, all the pieces of the puzzle are connected to each other and a certain number of trajectories are determined. These trajectories detected by many sensor nodes are then fused to identify the overall trajectory.

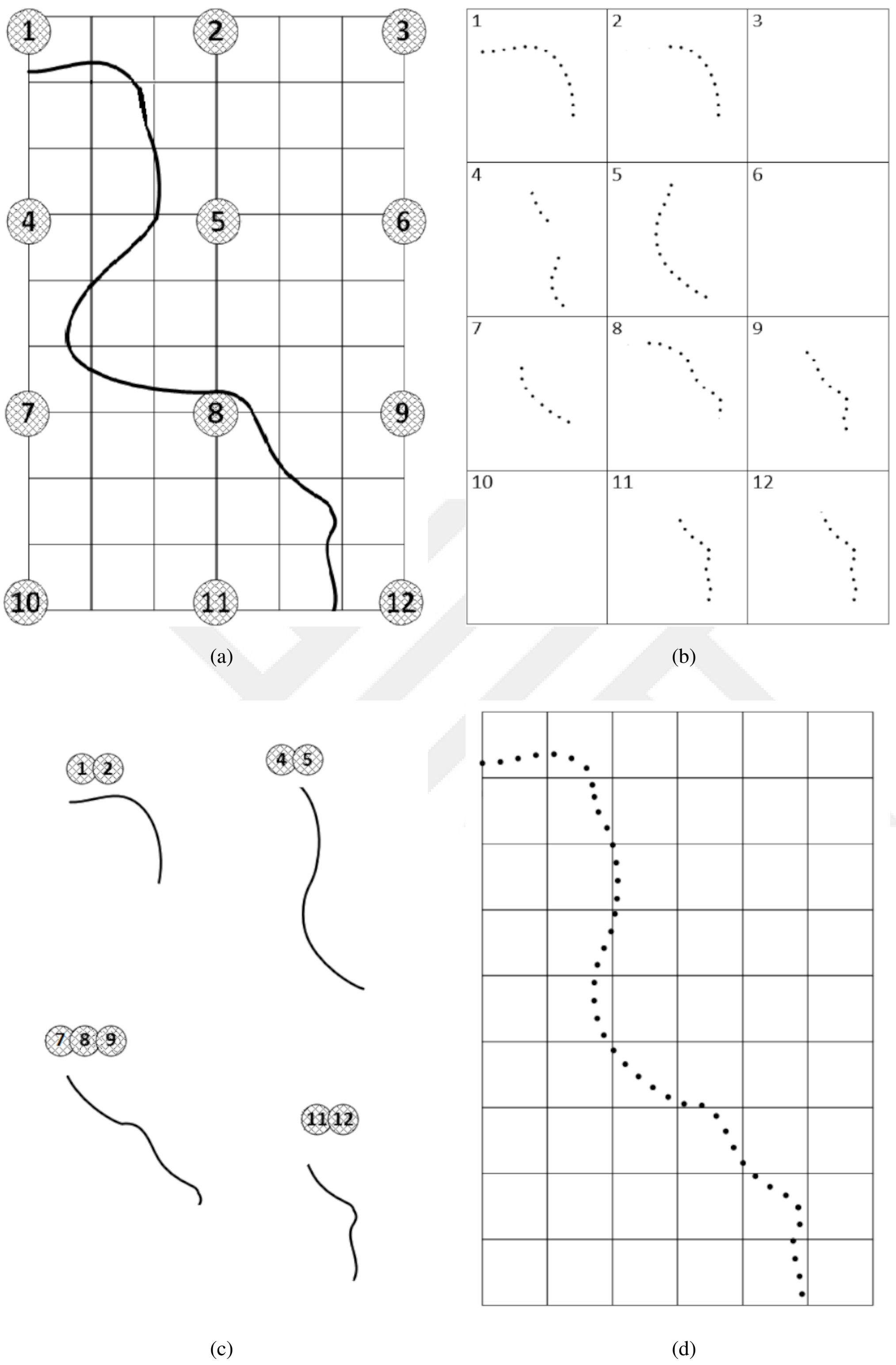


Figure 5.4: Multilevel fusion step-by-step trajectory detection. (a) Actual trajectory. (b) Intra-node fusion. (c) Inter-node fusion. (d) Trajectory construction.

5.3 Interval Type-2 (IT2) Fuzzy Logic Trajectory Analytics

A fuzzy logic system (FLS) can be defined as mapping from crisp values to vague values for deducing based on a set of rules, and then remapping to crisp values to provide a scalar output. There are five components in an interval type-2 (IT2) FLS; fuzzifier, fuzzy rules, inference engine, type-reducer and defuzzifier (5.5).

Firstly, the crisp input data is fuzzified into a set of fuzzy variables by using input membership functions, and this step is called Fuzzification. Afterwards, the inference engine is executed using a set of rules to produce fuzzy outputs. Then, these resulting outputs are utilized by the type-reducer to calculate the centroids of the fuzzy values, which is known as the type-reduced sets. Lastly, in the defuzzification step, the resulting reduced output is used to produce crisp outputs using the output membership functions.

IT2 FLS can be thought of as a set of embedded type-1 (T1) fuzzy logic systems [69]. Therefore, IT2 FLS is more adaptive and it can realize more complex input-output relationships which cannot be achieved by type-1 fuzziness.

Accordingly, number of rules in the rulebase will be less if IT2 FLS is used instead of T1 FLS [70, 71], because the ability of representing more uncertainties using the footprint of uncertainty (FOU) enables to cover the input/output situations with fewer fuzzy sets. This has been experimented in the next Chapter in Section 6.2.

In the scope of IT2 studies, firstly fuzzy geometry was introduced to the data model. Then, the rule engine was updated to use interval type-2 fuzzy logic system. Last but

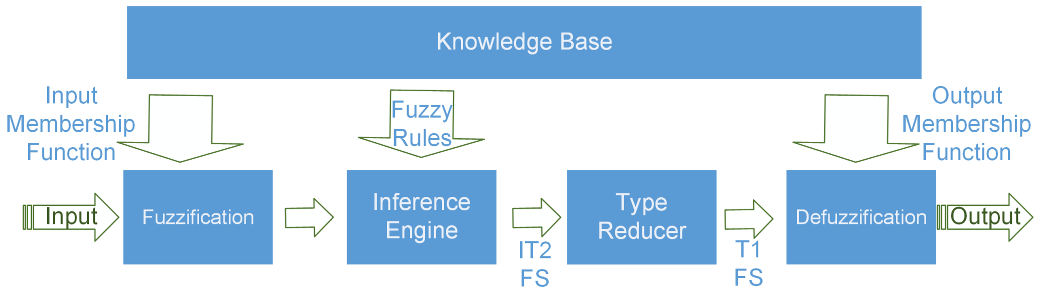


Figure 5.5: An IT2 Fuzzy Logic System.

not least, new Maritime Dataset was used to show that this dissertation can be adjusted to other application domains without an effort. To benchmark the performance gain after using IT2 fuzzy logic, several experiments given in the next Chapter in Section 6.2 was conducted.

5.3.1 Fuzzy Geometry

The position of an object can be represented as a fuzzy point [72], which is a point with an uncertainty for its position, but possible positions are known with a degree of certainty which is called a membership value. Therefore, a fuzzy point can be represented using a circle to encompass the fuzzy point 5.6.

For a fuzzy point $FP(x, y, \mu)$, position is derived from a set of possible positions using the following equations:

$$FP_x = \frac{\sum_{n=1}^{i} x_i * \mu_i}{i} \tag{5.4}$$

$$FP_y = \frac{\sum_{n=1}^{i} y_i * \mu_i}{i} \tag{5.5}$$

$$FP_\mu = \frac{\sum_{n=1}^{i} \mu_i}{i} \tag{5.6}$$

where i is the number of possible positions, $\mu \in [0, 1]$ and represents the membership value which is the Euclidean distance between a sensor node and the location of a sensed object.

By using two fuzzy points, a straight fuzzy line with uncertain boundaries can be defined [73]. If more than two fuzzy points are concatenated, a fuzzy trajectory FT can be represented as given in the following equation;

$$FT = [FP_1(x_1, y_1, \mu_1), FP_2(x_2, y_2, \mu_2), ..., FP_n(x_n, y_n, \mu_n))] \tag{5.7}$$

where n is the number of fuzzy points in the trajectory (Figure 5.7). While a trajectory is represented as a line, a fuzzy trajectory is visualized as an area. Each fuzzy point of the fuzzy trajectory is represented as a circle whose radius is computed according to the degree of certainty of the fuzzy point.

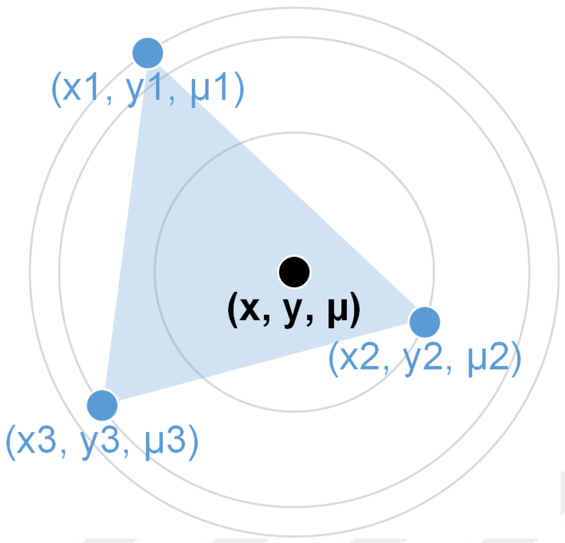


Figure 5.6: A fuzzy point derived from possible positions.

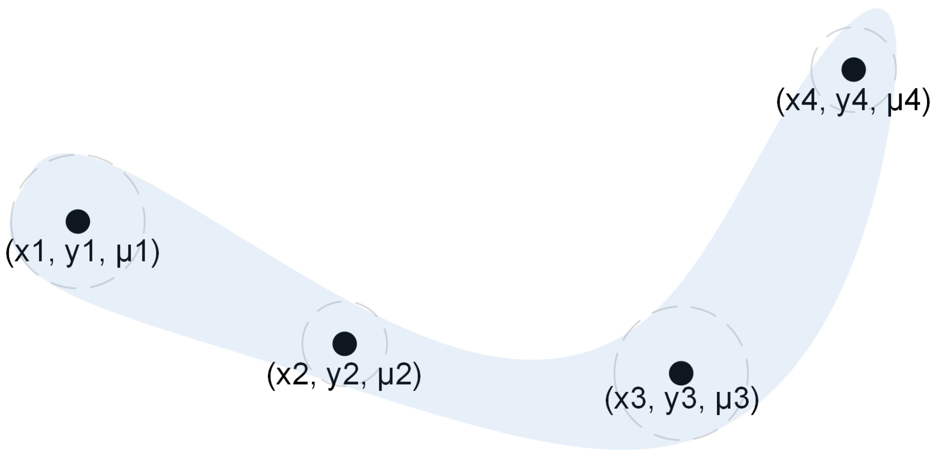


Figure 5.7: A fuzzy trajectory representation.

5.3.2 Extending Object Tracking with Fuzziness

The data of a moving object gathered from multiple sensor nodes has many values for the specific location of an object at an exact time. In other words, there are many variations of sensor data for a moving object. As discussed in previous sections, fuzzy point can be used to represent the estimated position of an object at a specific time. Over and above, by using a set of fuzzy points followed one after another, we can have a fuzzy trajectory.

To be able to deal with fuzzy trajectories, the data model had to be enriched by adding fuzziness. Firstly, new data models for fuzzy geometry was introduced as shown in

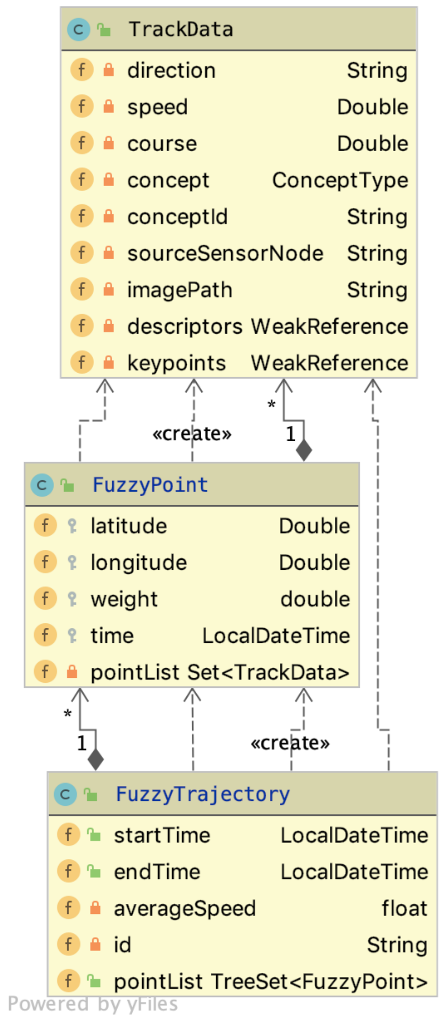


Figure 5.8: Object Models for Fuzzy Trajectories.

Figure 5.8. TrackData model represents the data sensed by a sensor node. Since there can be many sensor data for a fuzzy point, there is a many-to-one relationship between TrackData and FuzzyPoint. A similar relationship exists for FuzzyPoint and FuzzyTrajectory as a set of fuzzy points is necessary to form a fuzzy trajectory.

Another improvement was adding interval type-2 fuzzy logic to object tracking. The rule engine was extended to use IT2 FLS to fuse trajectories. First, a type-1 fuzzy logic system has been implemented in order to identify the input and output membership functions. Then, by using the T1 FLS as the base implementation, IT2 FLS has been introduced by extending the membership functions by defining lower and upper boundaries.

Type-1 input membership functions are;

- Direction MF: The direction change between two consecutive data (Unit: Degree)
  - sameDirection
    - Gaussian [-35, 0, 35]
  - similarDirection
    - Gaussian [-90, -45, -15]
    - Gaussian [15, 45, 90]
  - differentDirection
    - Trapezoidal [-180, -180, -135, -80]
    - Trapezoidal [80, 135, 180, 180]
- Time MF: The time difference between two consecutive data (Unit: Seconds)
  - realTime
    - Trapezoidal [0, 0, 5, 10]
  - newTime
    - Trapezoidal [5, 15, 30, 60]
  - oldTime
    - Trapezoidal [40, 60, MAX, MAX]

- Location MF: The distance between two consecutive data (Unit: Meters)
  - sameLocation
    - Trapezoidal [0, 0, 5, 10]
  - smallChangeLocation
    - Trapezoidal [5, 10, 20, 25]
  - bigChangeLocation
    - Trapezoidal [20, 50, 500, 500]
- Speed MF: The change in speed between two consecutive data (Unit: Percentage of the change)
  - sameSpeed
    - Trapezoidal [0, 0, 10, 20]
  - smallChangeSpeed
    - Trapezoidal [10, 20, 30, 40]
  - bigChangeSpeed
    - Trapezoidal [40, 50, 100, 100]

Type-1 output membership functions are;

- Similarity MF: The similarity of the two consecutive data
  - differentTrajectory
    - Gaussian [0.0, 0.0, 0.4]
  - similarTrajectory
    - Gaussian [0.25, 0.5, 0.75]
  - sameTrajectory
    - Gaussian [0.6, 1.0, 1.0]

The interval type-2 FLS has been constructed by blurring the type-1 FLS by shifting the left and right MFs uniformly. In this thesis, symmetrical footprint of uncertainty (FOU) was used for simplicity. For instance; The *SameDirection* Gaussian antecedent in type-1 FLS has been extended from left and right to introduce the upper and lower MFs in interval type-2 FLS.

$$\textit{Gaussian [-35, 0, 35]} \longrightarrow \begin{cases} \textit{Upper - Gaussian [-40, 0, 40]} \\ \textit{Lower - Gaussian [-30, 0, 30]} \end{cases}$$

Interval Type-2 input membership functions are;

- Direction MF: The direction change between two consecutive data (Unit: Degree)
  - sameDirection
    - Upper - Gaussian [-40, 0, 40]
    - Lower - Gaussian [-30, 0, 30]
  - similarDirection
    - Union
      - Upper - Gaussian [-95, -45, -10]
      - Lower - Gaussian [-85, -45, -20]
    - Union
      - Upper - Gaussian [10, 45, 95]
      - Lower - Gaussian [20, 45, 85]
  - differentDirection
    - Union
      - Upper - Gaussian [-180.0, -135, -75]
      - Lower - Gaussian [-175.0, -135, -85]
    - Union
      - Upper - Gaussian [75, 135, 180]
      - Lower - Gaussian [85, 135, 175]
- Time MF: The time difference between two consecutive data (Unit: Seconds)
  - realTime
    - Upper - Trapezoidal [0, 0, 5, 12]

    - Lower - Trapezoidal [0, 0, 5, 8]
  - newTime
    - Upper - Trapezoidal [3, 15, 30, 63]
    - Lower - Trapezoidal [7, 15, 30, 57]
  - oldTime
    - Upper - Trapezoidal [38, 60, MAX, MAX]
    - Lower - Trapezoidal [42, 60, MAX, MAX]

- Location MF: The distance between two consecutive data (Unit: Meters)
  - sameLocation
    - Trapezoidal [0, 0, 5, 10]
  - smallChangeLocation
    - Trapezoidal [5, 10, 20, 25]
  - bigChangeLocation
    - Trapezoidal [20, 50, 500, 500]

- Speed MF: The change in speed between two consecutive data (Unit: Percentage of the change)
  - sameSpeed
    - Trapezoidal [0, 0, 10, 20]
  - smallChangeSpeed
    - Trapezoidal [10, 20, 30, 40]
  - bigChangeSpeed
    - Trapezoidal [40, 50, 100, 100]

Interval Type-2 output membership functions are;

- Similarity MF: The similarity of the two consecutive data
  - differentTrajectory
    - Upper - Trapezoidal [0.0, 0.0, 0.2, 0.45]

    * Lower - Trapezoidal [0.0, 0.0, 0.1, 0.45]
  - similarTrajectory
    * Upper - Trapezoidal [0.3, 0.45, 0.55, 0.7]
    * Lower - Trapezoidal [0.3, 0.5, 0.5, 0.7]
  - sameTrajectory
    * Upper - Trapezoidal [0.5, 0.7, 1.0, 1.0]
    * Lower - Trapezoidal [0.5, 0.8, 1.0, 1.0]

Figure 5.9 shows the input member functions of the fuzzy logic system. In Section 6.2.1, the results of comparison between without fuzzy logic, with type-1 fuzzy logic and interval type-2 fuzzy logic were given.

Similar to the type-1 fuzzy rule generation explained in Section 5.2.3.2, same steps were followed to generate IT2 fuzzy rules using the training data. The difference between the T1 and IT2 is the input membership functions which are being used to transform crisp values into fuzzy input parameters.

Below is a list of sample fuzzy rules which were used in the rule engine for fusion. The full list of IT2 fuzzy rules can be found in Appendix A.2.

- IF similarDirection AND smallChangeLocation AND realTime AND smallChangeSpeed THEN sameTrajectory
- IF differentDirection AND smallChangeLocation AND realTime AND smallChangeSpeed THEN sameTrajectory
- IF differentDirection AND sameLocation AND realTime AND sameSpeed THEN sameTrajectory
- IF differentDirection AND smallChangeLocation AND realTime AND bigChangeSpeed THEN similarTrajectory
- IF similarDirection AND sameLocation AND realTime AND bigChangeSpeed THEN sameTrajectory
- IF sameDirection AND sameLocation AND realTime AND smallChangeSpeed THEN sameTrajectory

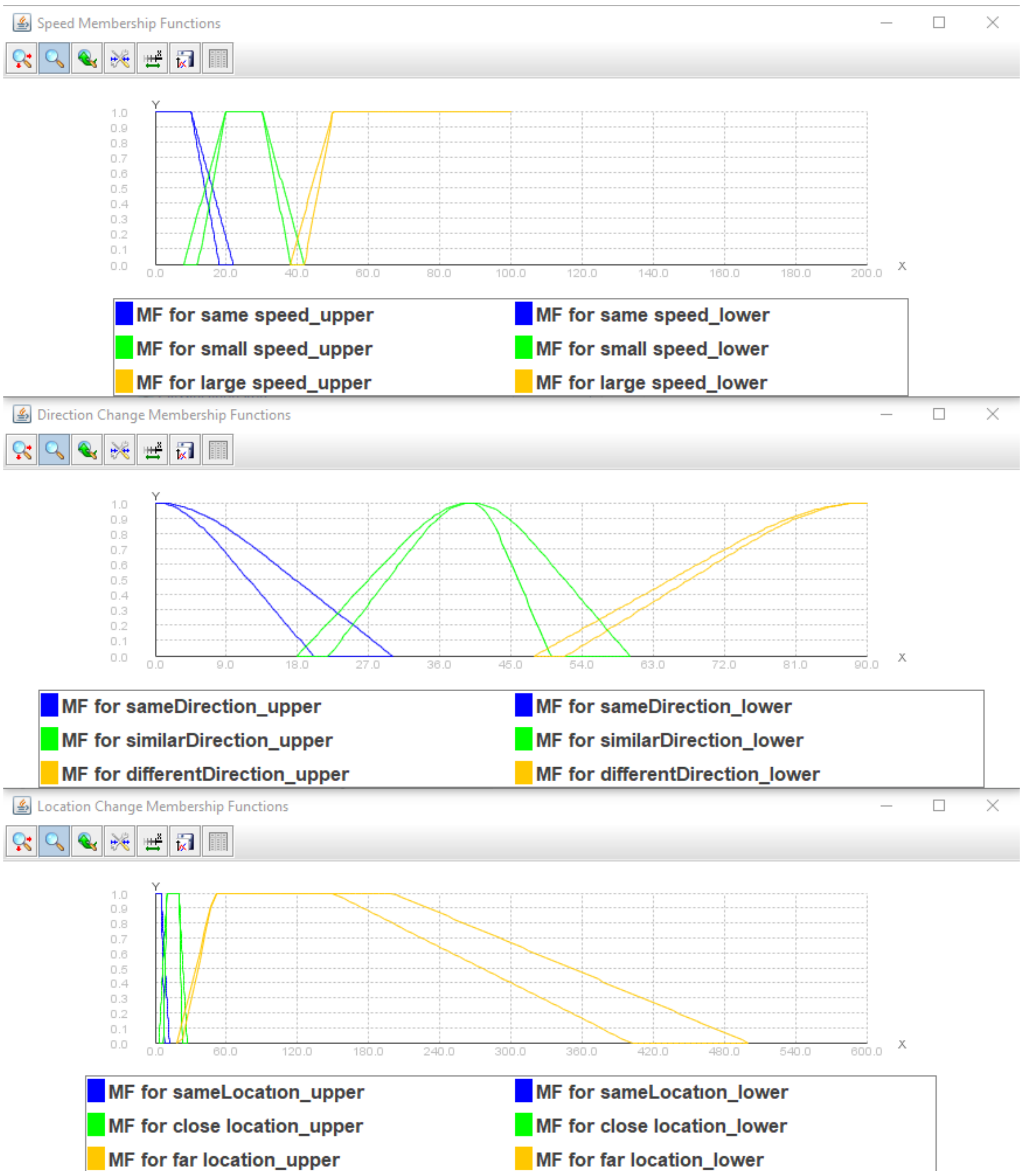


Figure 5.9: Input member functions for IT2 FLS.

Table 5.3: Fuzzy-based Rule Engine Parameter Values for Each Trajectory

Parameter	T_1	T_2	T_3
Calculation Direction Change:	| 0-18 |	| 244-18 |	| 26-18 |
Direction Change (degree)	18	222	8
Time Difference (seconds)	7	2	1
Location Difference (meters)	1.4	3.0	1.0
Calculation Speed Change:	| (2-0.2)/2 |	| (0.4-1.5)/0.4 |	| (1.1-1)/1.1 |
Speed Change (%)	%90	%275	%9

Table 5.4: Fuzzified Input Parameters for Each Trajectory

Parameter	T_1	T_2	T_3
Direction Change (degree)	sameDirection	differentDirection	sameDirection
Time Difference (seconds)	realTime	realTime	realTime
Location Difference (meters)	sameLocation	sameLocation	sameLocation
Speed Change (%)	bigChangeSpeed	bigChangeSpeed	sameSpeed

IT2-based rule engine has been used in multilevel object tracking in order to fuse the trajectories. For the sample scenario specified in Section 5.2.3.3,

Table 5.3 shows the computed values of parameters which are used as input for the interval type-2 based rule engine. *Direction Change* represents the angle difference in degrees between the last object of a trajectory and the moving object. *Time Difference* represents the time difference in seconds between the time of the last object of a trajectory and the detection time. *Location Difference* is the distance in meters from the location of the last object of a trajectory and the location of the moving object. *Speed Change* is the change of velocity in percentage.

These crisp values are transformed into fuzzy input parameters as given in Table 5.4.

Finally, the execution of the IT2 FLS system using these fuzzy inputs for the scenario shown in Figure 5.3 results the scores in Table 5.5 for each trajectory. According to the final scores, the new object belongs to the trajectory T_3, which has a higher probability for the trajectory of the new object than the other possible trajectories.

Table 5.5: Output Values of the Fuzzy-based Rule Engine for Each Trajectory

	T_1	T_2	T_3
Similarity	0.6	0.4	0.9

5.3.3 IT2 Fuzzy Logic based Classification

A new classification algorithm given in Algorithm 8 was developed using the interval type-2 fuzzy logic in order to enhance the trajectory prediction and anomaly detection analytics.

This classification algorithm has two parts, initialization and assessment. During the initialization part, an internal trajectory model is generated for each class in order to efficiently classify a point. During assessment part, trajectory models are used to calculate the angle, distance and speed input values are computed and provided to the IT2 fuzzy logic system to get the classification output.

The input of the FLS is the global trajectories extracted from the multilevel object tracking algorithm. By using the input member functions, characteristics of the trajectory is mapped to fuzzy values and as an output, the similarity of the compared trajectories are generated in a fuzzy manner.

Rule base of the IT2 FLS has the following fuzzy rules to classify the data;

- IF sameDirection AND sameLocation THEN sameTrajectory
- IF sameDirection AND sameLocation AND sameSpeed THEN sameTrajectory
- IF sameDirection AND closeLocation AND sameSpeed THEN sameTrajectory
- IF similarDirection AND sameLocation AND sameSpeed THEN sameTrajectory
- IF sameDirection AND sameLocation AND smallSpeedChange THEN sameTrajectory
- IF sameDirection AND farLocation AND sameSpeed THEN differentTrajectory

Algorithm 8 IT2 Fuzzy Logic Classification Algorithm

```
1: procedure IT2FUZZYCLASSIFIER($n[], T[]$)
   ▷ Trained by the global trajectories ($T[]$)] and classifies the instance ($n$) using the IT2 FLS.
2:     $fuzzyEngine \leftarrow initializeIT2FLS()$
3:     $output \leftarrow []$                          ▷ Classification output array for each trajectory
4:     $lat \leftarrow$ n[0]                                                        ▷ Latitude
5:     $lon \leftarrow$ n[1]                                                        ▷ Longitude
6:     $time \leftarrow$ n[2]                                                 ▷ Time difference
7:     $minDistance \leftarrow$ Maximum value for initialization
8:     $closestPoint = \emptyset$
9:     $i = 0$
10:    for each trajectory $t$ in $T[]$ do
11:        for each point $p$ in trajectory $t$ do
12:            $\rho 1 \leftarrow cartesianDistance(lat, lon, p)$
13:            if $\rho 1 < minDistance$ then                         ▷ Find closest point
14:                $minDistance \leftarrow \rho 1$
15:                $closestPoint \leftarrow p$
16:                $\theta 1 \leftarrow calculateAzimuth(t, p)$
17:                $\alpha 1 \leftarrow getSpeed(t, p)$
18:            end if
19:        end for
20:        $last \leftarrow$ Last classified instance in the classifier
21:        $\rho 2 \leftarrow cartesianDistance(lat, lon, closestPoint)$
22:        $\theta 2 \leftarrow calculateAzimuth(lat, lon, last)$
23:        $\alpha 2 \leftarrow getSpeed(T, p)$
24:        $output[i] \leftarrow fuzzyEngine.get(\theta 2 - \theta 1, \alpha 2 - \alpha 1, \rho 2 - \rho 1)$
25:        increment $i$
26:    end for
27:    return $output$
28: end procedure
```

- IF differentDirection AND farLocation AND bigSpeedChange THEN differentTrajectory
- IF similarDirection AND farLocation AND bigSpeedChange THEN differentTrajectory
- IF differentDirection AND closeLocation AND bigSpeedChange THEN differentTrajectory
- IF differentDirection AND farLocation AND bigSpeedChange THEN differentTrajectory

Extracted global trajectories are used to train well-known classification algorithms for the purpose of benchmarking the proposed IT2 classification algorithm. The selected classification algorithms in our benchmarks are;

- Random Forest [20]
- Decision Table [22]
- Sequential Minimal Optimization (SMO) [25]
- Naive Bayes [21]
- Logistic Regression [23]
- Multilayer Perceptron (MLP) [24]

Experimental results to show the performance gain after using IT2 fuzzy logic are given in the next Chapter in Section 6.2.

5.3.4 Trajectory Prediction

Classification algorithms are used for prediction in many studies [74, 75]. Well-known classification algorithms were trained using the extracted global trajectories, and once the learning phase is over, a new sensor data was classified to predict the possible trajectory of the object.

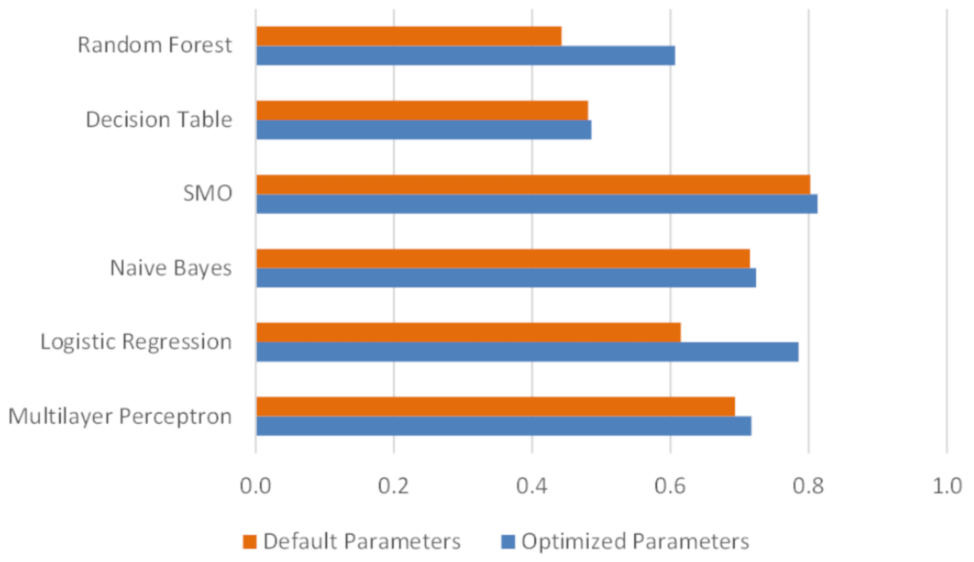


Figure 5.10: Success rates of classification algorithms in prediction with/without parameter tuning.

The parameters of the classification algorithms were tuned to increase the success rates of the predictions of the trajectories. Figure 5.10 shows the performance of each classification algorithm with and without tuned parameters. The success rate is calculated using the number of correct identifications of the trajectories. The SMO algorithm is more efficient than the other machine learning algorithms used in the experiments and which have been very successful. Moreover, it is important to emphasize that only the SMO algorithm can identify an equal possibility of overlapping trajectories, which is an important case for the prediction. Executing algorithms with optimized parameters gives better results than expected, but optimizing logistic regression is one of the most affected by overall success.

The proposed prediction algorithm based on IT2 fuzzy logic classification is given in Algorithm 9. Proposed algorithm performed better than the known algorithms. Experimental results are given in the next Chapter in Section 6.2.3.

5.3.5 Anomaly Detection

Anomaly detection aims to find the patterns which are unexpected and this can help to identify problems accurately or early detection of conflict situations to take action

Algorithm 9 IT2 Trajectory Prediction Algorithm

```
1: procedure IT2PREDICT($n[], T[]$)
   ▷ Trains the IT2 Fuzzy Classifier with global trajectories ($T[]$)] and classifies the instance ($n$). Returns the predicted trajectory.
2:     $classified[] \leftarrow it2FuzzyClassifier(n, T)$ ▷ Classification output array for each trajectory
3:     $max = 0$
4:     $ind = 0$
5:     for each classification value $c$ in $classified[]$ do
6:         if $c > max$ then                ▷ Find the highest classification score
7:             $max \leftarrow c$
8:             $ind \leftarrow$ index of classification value
9:         end if
10:    end for
11:    return $T[ind]$
12: end procedure
```

before potential problems occur.

Anomalies can occur for both objects and trajectories. An object moving along a trajectory can deviate from an existing trajectory and it can be called an object anomaly. Besides, there can be completely a new trajectory which has never been identified before and can be matched with any of the existing global trajectories, which is assumed as a trajectory anomaly.

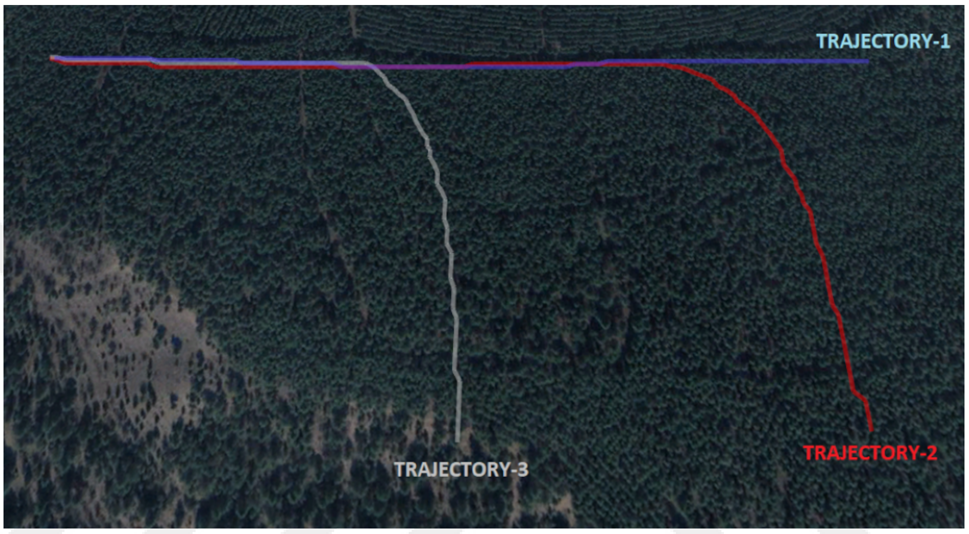
By using the global trajectories as the ground truth, it can be determined whether an object is lined up on a path or not [76]. If the object deviates from the trajectory, this can be accepted as an anomaly sign (Figure 5.11b) for the sample trajectories given in Figure 5.11a.

The proposed anomaly detection algorithm based on IT2 fuzzy logic classification is given in Algorithm 10. Proposed algorithm performed better than some of the known algorithms. Experimental results are given in the next Chapter in Section 6.2.3.

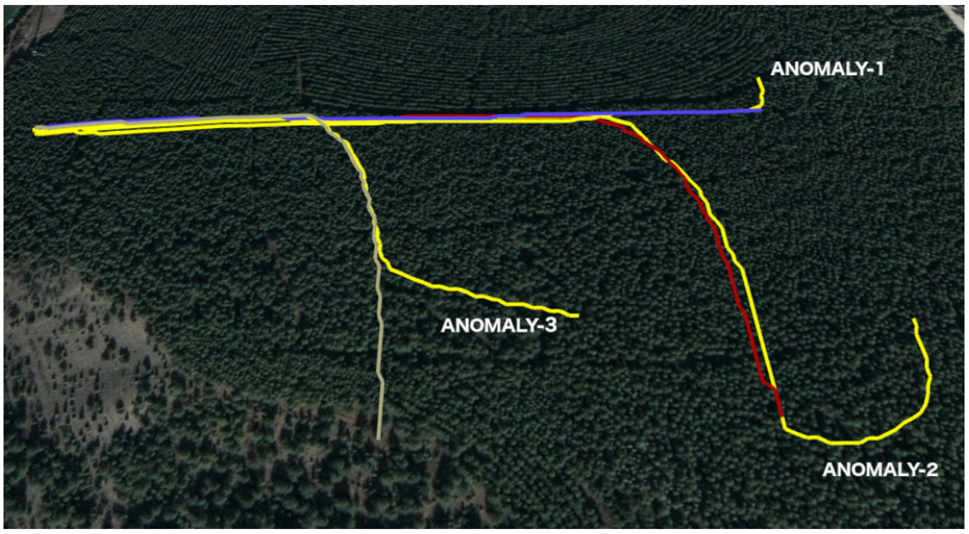
5.4 Real World Use Cases

I used two different real world dataset in the scope of this dissertation; GeoLife Trajectory dataset and Maritime Cadastre dataset. Former was used for surveillance application in a rural area. Latter is used for monitoring the vessels in a zone.

To be able to use these datasets, I imported the dataset through my simulator to mimic



(a)



(b)

Figure 5.11: (a) Trained trajectories for analytics. (b) A sample anomaly in which the object leaves suddenly while moving on Trajectory-3.

Algorithm 10 IT2 Anomaly Detection Algorithm

```
1: procedure IT2ANOMALY($n[], T[]$)
   ▷ Trains the IT2 Fuzzy Classifier with global trajectories ($T[]$)] and classifies the instance ($n$). Returns if there is an anomaly or not.
2:     $\alpha \leftarrow it2Predict(n, T)$                ▷ Predicted trajectory
3:     $\beta \leftarrow$ Previously predicted trajectory
4:     if $\alpha! = \beta$ then                    ▷ Possible anomaly.
5:         $s_\beta \leftarrow$ Latest scores of trajectory $\beta$
6:         $s_\alpha \leftarrow$ Latest scores of trajectory $\alpha$
7:         if $s_\beta$ is decreased AND $s_\alpha$ is increased then
8:             return true
9:         end if
10:    end if
11:    return false;
12: end procedure
```

as if data is coming from the wireless multimedia sensors (Figure 5.12). This operation removed the identifier on the data, and fully anonymized the input data which makes it more challenging problem.

Both GeoLife Trajectory dataset and Maritime Cadastre dataset are not exploited by multiple sensors. Instead, each trajectory corresponds to a single object carrying a geographic location providing device. To make it usable for the experiments, the original trajectories were transformed into sensor data that can be detected by several sensors via the simulator. The simulator reads actual trajectory as an input and generates motion at the exact location of the actual trajectory based on the type of object. Then, the sensor nodes that can detect motion at that location are triggered as if there is an object.

5.4.1 GeoLife Dataset

GeoLife dataset [77] was collected by the Microsoft Research as part of the GeoLife project. It is a GPS trajectory dataset which is produced by 182 users in 5 years.

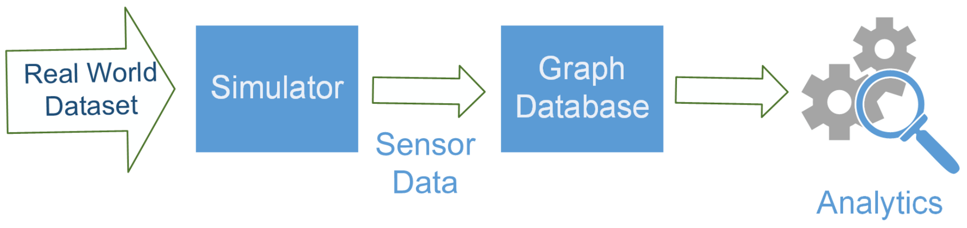


Figure 5.12: Utilization of real world dataset.

From 2007 to 2012, 17,621 trajectories that contains more than 1 million Km. distance and almost 50K hours of duration. The trajectories were recorded in a dense representation which is every 1-20 seconds per point.

Each folder of the dataset contains GPS log data of a user, and each folder has a number of PLT files in it. Each PLT file is named by starting time of a trajectory and contains the GPS data. PLT files are typical text files with a GPS data in each line which has Latitude, Longitude, Altitude and Date information. For more information about content of the PLT files, please consult the [77].

An example data from the dataset is:

```
39.984611,116.318026,0,493,39744.1204861111,2008-10-23,02:53:30
39.984608,116.317761,0,493,39744.1205439815,2008-10-23,02:53:35
39.984563,116.317517,0,496,39744.1206018519,2008-10-23,02:53:40
39.984539,116.317294,0,500,39744.1206597222,2008-10-23,02:53:45
39.984606,116.317065,0,505,39744.1207175926,2008-10-23,02:53:50
```

In order to use GeoLife dataset in this thesis, the transport mode labels in the dataset were mapped to simulation concept types used in the surveillance application domain. Table 5.6 shows the mapping of the concept types "Vehicle" and "Human" because there is no label associated with the type of concept "Animal" in the dataset.

5.4.2 Marine Cadastre Dataset

Publicly available maritime dataset is an Automatic Identification System (AIS) dataset [78] from the U.S. Coast Guard. It has been collected using a navigation (AIS) device

Table 5.6: Mapping from Transportation Mode to Concept Type

Transportation Mode	Concept Type
Walk	Human
Car	Vehicle
Taxi	Vehicle
Bus	Vehicle
Train	Vehicle

which communicates the location of vessels in real time. Data is publicly available in "www.marinecadastre.gov".

AIS is an automated system for exchanging the navigational information between ships and shore stations like Vessel Traffic Services (VTS) stations (Figure 5.13). Information such as ship type, location, time, and speed have been extracted from the raw AIS data and provided as a dataset.

Each data is downloadable from the website of the U.S. Coastal guard as an Excel Sheet File. The maritime data is available since 2009 and each year updated. Records are filtered to one minute and formatted in monthly files by Universal Transverse Mercator (UTM) zone. Below is the column names in which data provided;

- MMSI: This is the Maritime Mobile Service Identity, which is the unique iden-

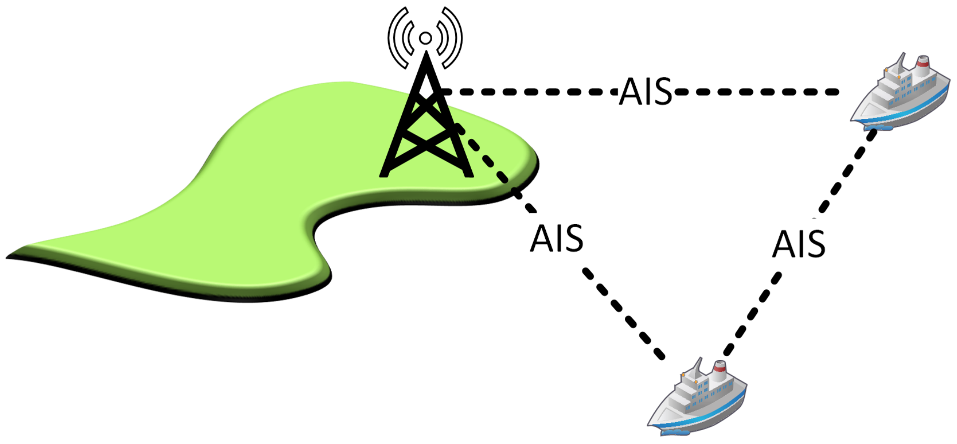


Figure 5.13: Sample AIS data exchange representation.

tifier for vessels.

- BaseDateTime: Data timestamp
- LAT: Geographical latitude
- LON: Geographical longitude
- COG: Course Over Ground
- SOG: Speed Over Ground
- Heading: Current heading of the AIS vessel at the time of the last message
- VesselName: Name of the vessel
- IMO: IMO ship identification number
- CallSign: Callsign of the vessel
- VesselType: Ship type
- Status: Navigation status
- Length: Length of the vessel
- Width: Width of the vessel
- Draft: Name of the vessel
- Cargo: Cargo payload information

Some of the columns which are not a direct input for the algorithm was filtered. Besides, the MMSI number was used as the identifier of moving vessels since most of the AIS systems use the MMSI, not the IMO number, which is another unique number for vessels. After that, the data simulator was adapted to use maritime dataset as input.

Two new Java-based console applications were implemented to prepare the maritime dataset for experiments. These applications are VesselDataCleaner and VesselDataImporter.

- VesselDataCleaner: This application removes the unused columns, such as width, length, cargo, and callsign, from CSV files. Also, it provides filtering trajectories according to their length.
- VesselDataImporter: This application reads the maritime dataset stored in CSV files and utilizes the data simulator functionality to import data as if it is sensed from the sensor nodes.

An example data from the dataset is;

> 366940480,2017-01-04T13:51:07,52.41575,-174.60041,9.1,-154.0,251.0,EARLY DAWN,IMO7821130,WDB7319,1001,undefined,32.95,8.82,4.0,31

After removing the unused columns, only the columns necessary for our algorithms are left, like below;

> 366940480,2017-01-04T13:51:07,52.41575,-174.60041

Figure 5.14 shows a sample trajectory from maritime dataset. It can be seen that some part of the trajectory are wider, some of them not, which is a result of using the fuzzy points in trajectory fusion calculations. The width of each fuzzy point is derived from the set of possible positions by taking into account the degree of certainty values.

5.5 Remarks

An unsupervised object tracking approach which is developed using the graph-based big data model was proposed. Management and storage of the big graph data in a NoSQL graph-based big database system, namely OrientDB, was implemented and an efficient object tracking approach was proposed. The approach for tracking objects consist of three main algorithms as intra-node fusion, inter-node fusion, and object trajectory construction.

Usage of two different datasets shows that this dissertation is not limited to only the Surveillance domain, but it can also be used for other domains. As a matter of fact

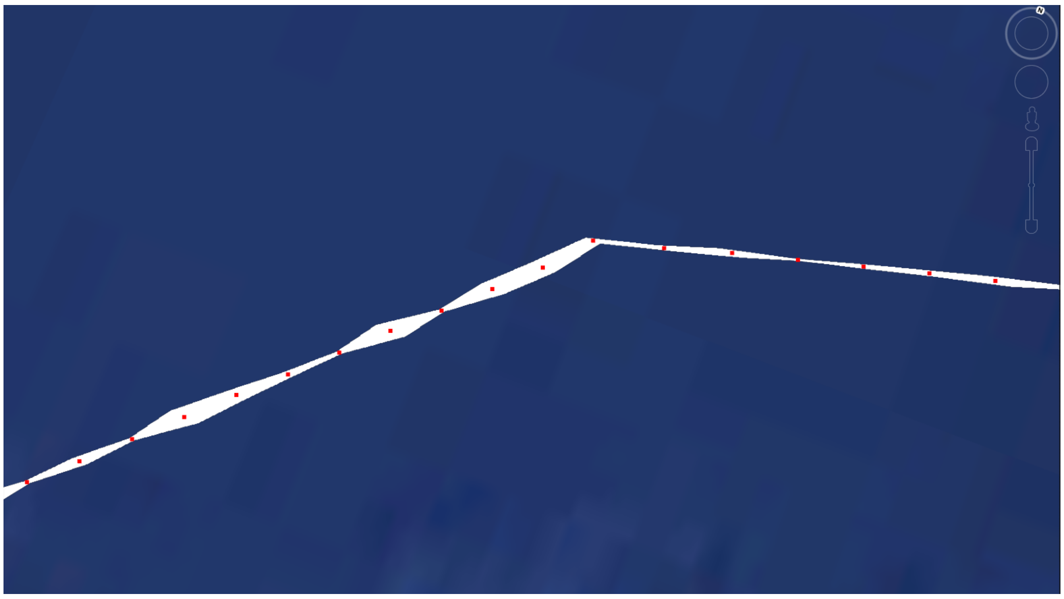


Figure 5.14: A sample trajectory from maritime dataset.

that spatiotemporal data is not bounded to any domain, this thesis can be used without any limitation on the application domain.

Data model extended to support fuzziness and trajectories are represented as fuzzy trajectories to be able to define trajectories not as a line but as a polygon, which adds the width aspect to a trajectory.

Trajectory prediction and anomaly detection analytics were applied on extracted global trajectories using well-known classification algorithms. Then, an IT2 FLS was proposed to make more accurate predictions and enhance anomaly detection analytics.

CHAPTER 6

EVALUATION

This chapter presents experimental studies which were conducted to evaluate the performance of proposed multilevel object tracking algorithm and trajectory analytics. In addition to those, interval type-2 fuzzy logic extension on top of the existing implementation was tested to benchmark the improvements provided by the fuzziness.

6.1 Multilevel Object Tracking Experiments

A test environment to test proposed object tracking approach was prepared, and experiments are realized in two different aspects. The first aspect was to visualize the performance of the algorithm for different types of scenarios using the Scenario Builder component of the simulator, which is described in Chapter 4. In parallel with this, test results are also compared with the Kalman filter tracking approach [79, 80]. With the second aspect of the experiments, the performance of the proposed tracking algorithms was evaluated with a real world data set. Because there are many public datasets, most tracking applications are just video data sets such as Multi-Object Tracking [81] or dataset with device tag, such as GPS or RFID, that do not conform to this dissertation's problem area. In addition, some data sets have focused on interconnected roads designed for wheeled vehicles and pedestrian traffic [82] but in this experiments, a rural area was monitored which has no information on roads. Therefore, to evaluate algorithms with real data, a dataset was needed to be adapted to the surveillance application domain. The GeoLife trajectory dataset was used because it has different types of transport modes and a wide variety of trajectory lengths.

6.1.1 Synthetic Data

The simulator described in previous chapter was used to produce synthetic data. Both grid-based and random distributed wireless sensor networks were setup to evaluate the performance for different topologies.

Figure 6.1 shows example scenarios generated by the simulator. Various scenarios were created using different trajectory models and concept types to evaluate the performance of proposed algorithms. The following formulas were used to calculate the precision/recall values and the F-measure.

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

where *TP* is the number of true positives, *FP* is the number of false positives, and *FN* is the number of false negatives. For a better evaluation, F-measure or balanced F-score was used to combine precision and recall values in a metric.

$$F = 2.\frac{Precision.Recall}{Precision + Recall} \tag{6.3}$$

Figure 6.2 represents the result of experiments by calculating the F-measure scores of each scenario group for both grid-based and random deployment of sensors. Proposed approach detects straight trajectories because they are easy to track with respect to more complex trajectories. The detection performance of the circular and zigzag trajectories are slightly lower than those of the other types. The zigzag pattern has sharp turns and is not expected and treated as new objects. For circular patterns, the behavior is not expected to move back and forth at the same point. However, the overall performance looks promising. From the point of distribution of the nodes, if sensor nodes are deployed randomly, the scores decrease. The loss of performance in random positioning has two main causes. First, there are undetected areas and if the trajectory of the object falls in this area, it causes disconnection. Second, some densely positioned areas create multiple possible trajectories for a single trajectory and make inter-node fusion more difficult to manage. To summarize, randomly deployed nodes are not as good as grid positioning, but this is an expected result.

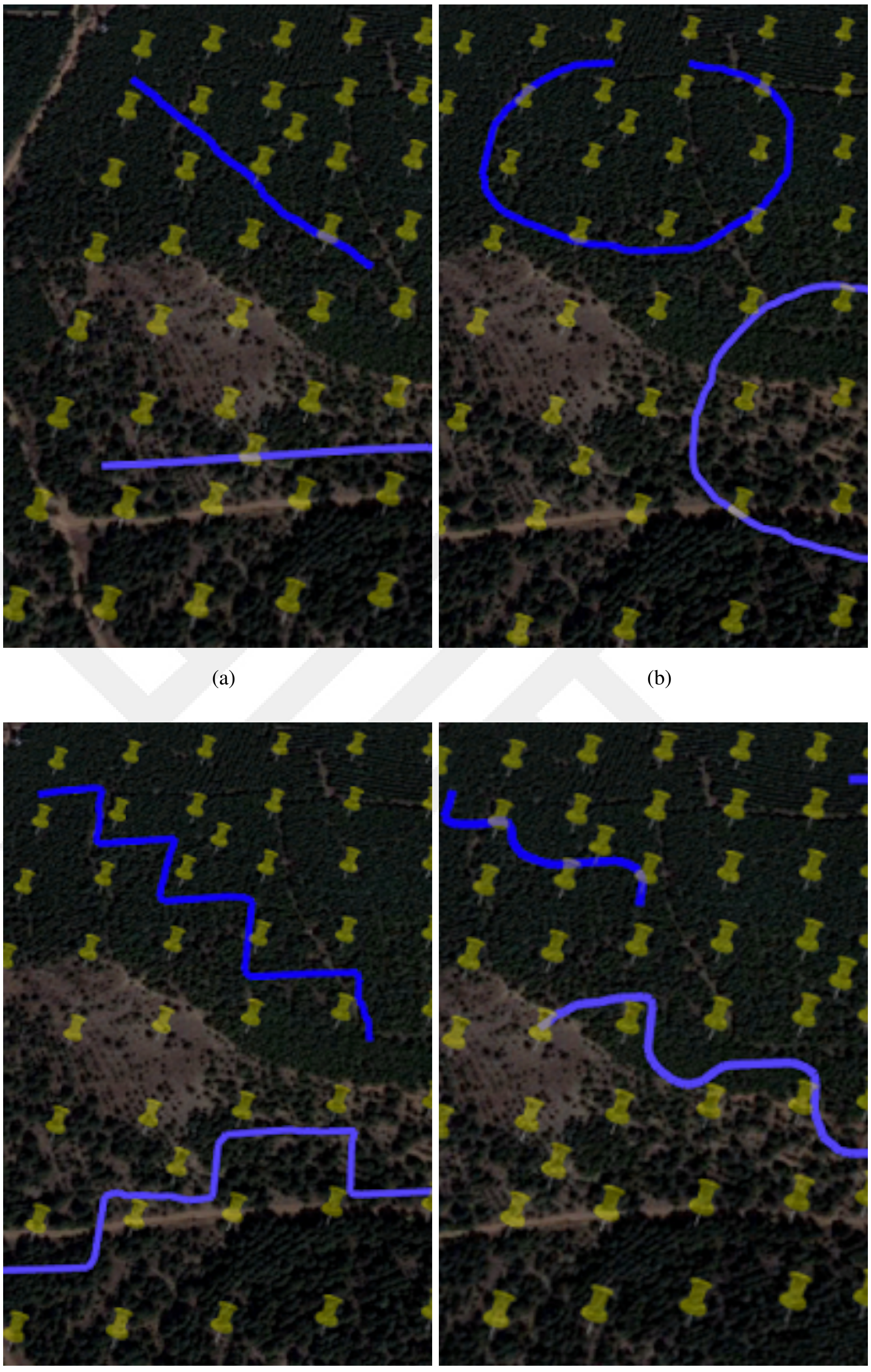


Figure 6.1: Scenario samples generated using developed simulator. (a) Straight trajectories. (b) Circular trajectories. (c) Zig zag trajectories. (d) Wavy trajectories.

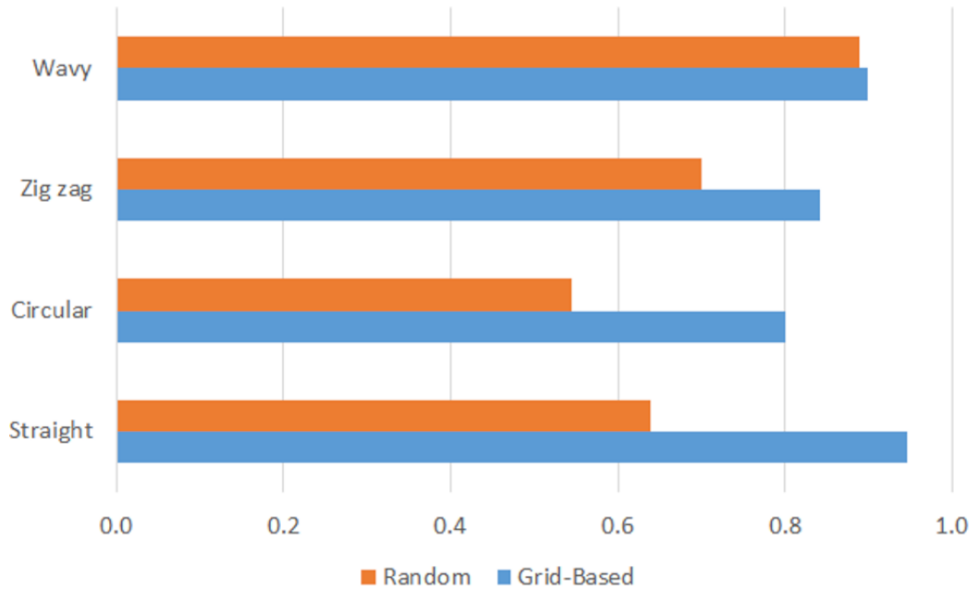
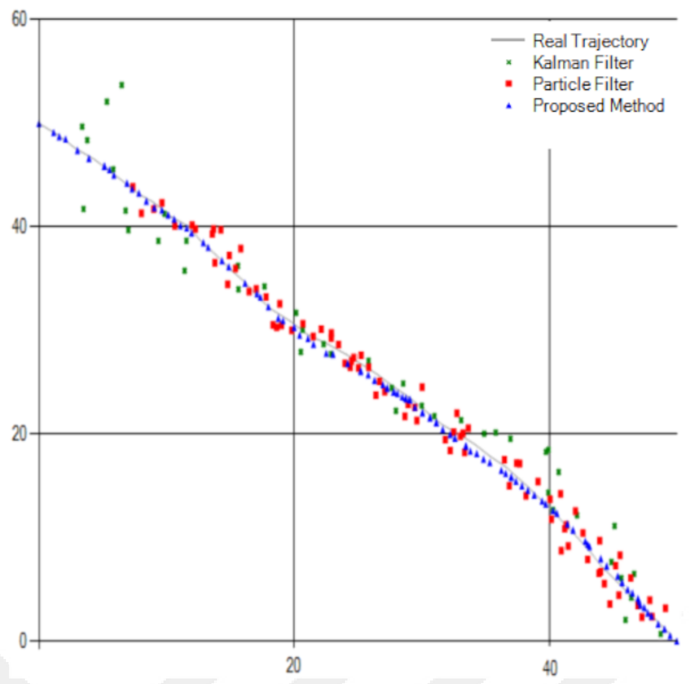


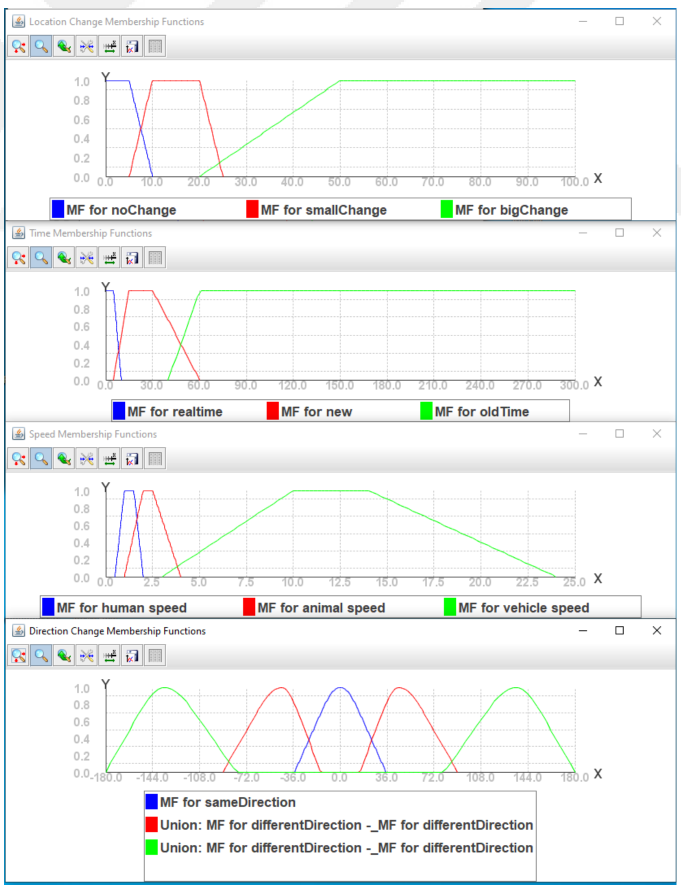
Figure 6.2: F-measure scores for both grid-based and randomly deployed nodes.

Since Kalman filters and Particle filters are widely used in object tracking [83, 84], the performance of proposed algorithms were compared with these two filters. The Kalman filter tries to balance the motion model and the measurements to provide a better estimate of trajectories. It uses linear projections with Gaussian noise to increase efficiency while the particle filter uses a sequential Monte Carlo method. Both algorithms recursively update the state estimate. Kalman filter uses the system model and the sensor observations to estimate the current state from the previous states. Particle filter uses random sampling to generate different system states, then assigns high weights to the states supported by the sensor data.

The results of performance comparisons and error rates are given in Figure 6.3 and 6.4. Error rates are calculated by the Euclidean distance between real trajectory and the output of the algorithm. The results show that the proposed approach better detects the trajectory of the object, with error rates lower than those of Kalman and particles filters. In addition, it appears from the experimental results that even object maneuvers, the proposed object tracking algorithms can continue to track objects with a relatively low error rate. Kalman filter and particle filter algorithms need early recovery time to produce optimized predictions. For this reason, some of the initial values of the particle filter are ignored.

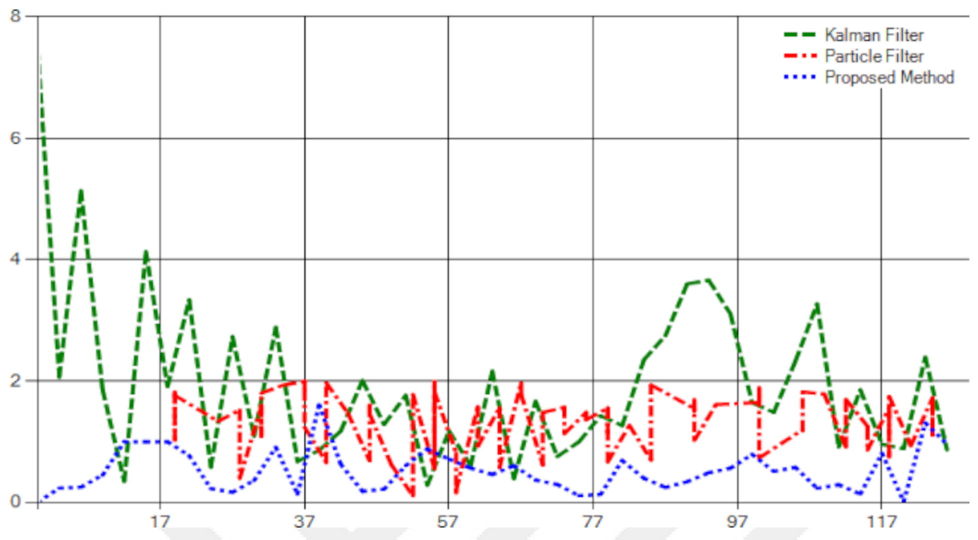


(a)

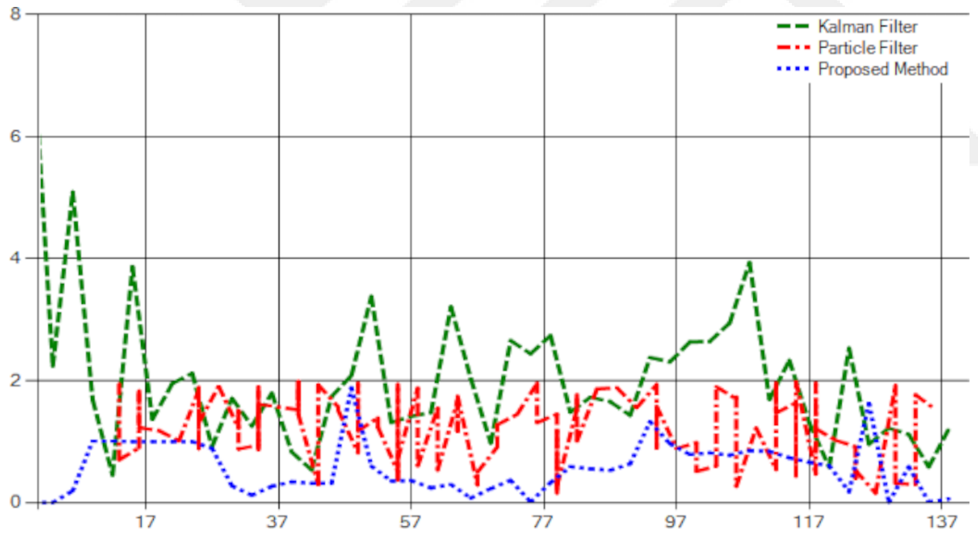


(b)

Figure 6.3: Comparison with Kalman filter and Particle filter. (a) Straight trajectory. (b) Wavy trajectory.



(a)



(b)

Figure 6.4: Error rates comparison with Kalman filter and Particle filter. (a) Straight trajectory. (b) Wavy trajectory.

Table 6.1: Algorithm Scores with Various Parameters

ε	NodeBuffer=5	NodeBuffer=10
0.35	0.322058681	0.483088022
0.40	0.999631145	0.999631145
0.45	0.999631145	0.481668622
0.50	0.499815572	0.321112415

6.1.2 Real World Dataset

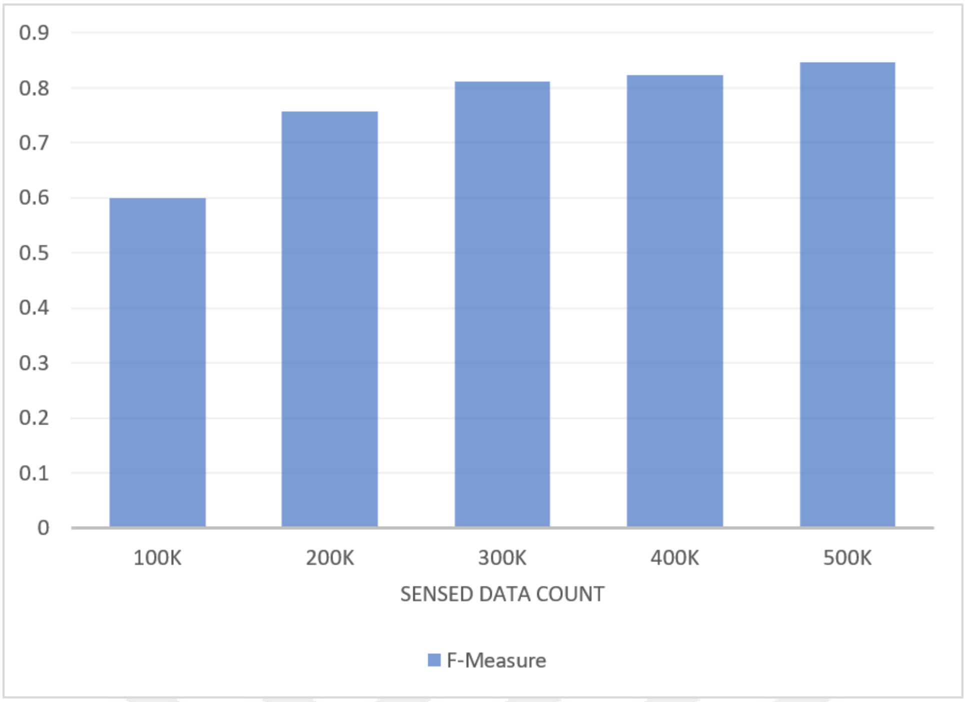
In order to evaluate the performance of the proposed object tracking algorithms, a confusion matrix was used. The selection of the ε and *NodeBuffer* parameters affects the sensitivity of the algorithm. ε is the threshold of the distance between the actual distance and the calculated distance using speed and time. *NodeBuffer* is the parameter to specify the number of historical sensor data used to identify new sensor data. that is, whether these data correlate with previous detections or not.

The scores of the parameters are calculated by normalizing the Hausdorff Distance between real trajectory and extracted trajectory with the number of tracks found. Table 6.1 illustrates the computed scores for different parameter values of ε and *NodeBuffer*. The results show that the tracking algorithm works the best with $\varepsilon = 0.40$. *NodeBuffer* depends on the ε parameter as a pivot. For bigger ε values *NodeBuffer* is negatively correlated and with lower ε values *NodeBuffer* is positively correlated.

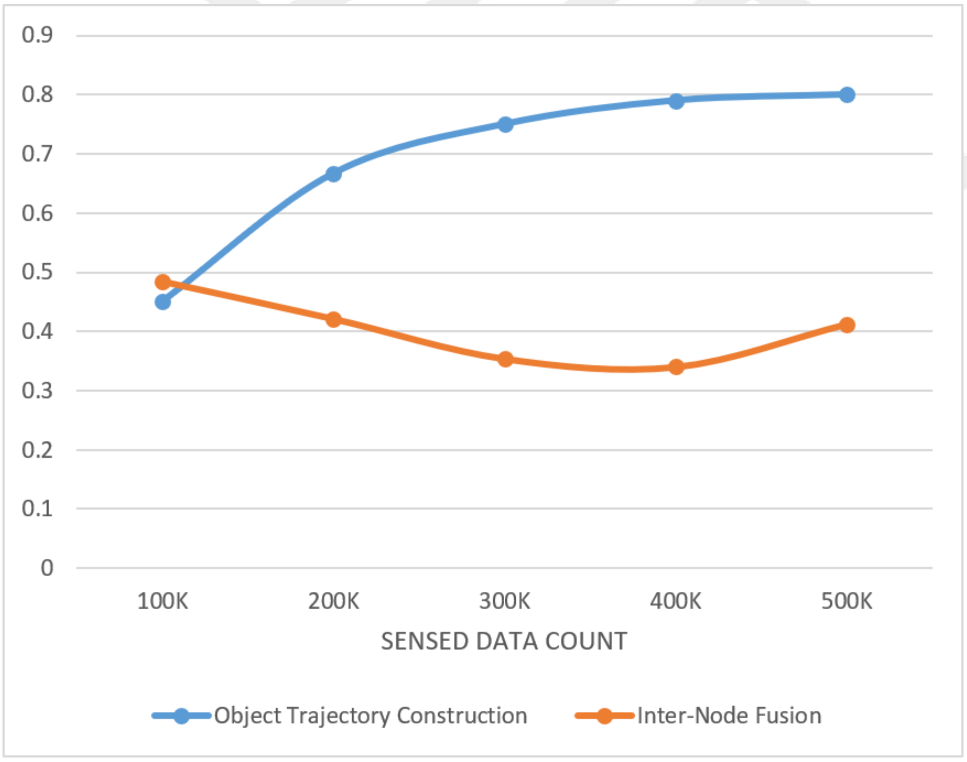
Therefore, parameters $\varepsilon = 0.40$ and *NodeBuffer=5* were used as default values to monitor the performance of the algorithm in GeoLife Trajectories dataset.

The measurement scores F are given in Figure 6.5a for different sizes of datasets. The algorithm duplicates small pieces of longer tracks as different tracks. As a result, tracks are correctly detected with smaller, noisy tracks. This provides lower accuracy performance, but the effect of noisy data decreases as more sensor data is processed.

To show the positive effect of multilevel tracking, especially that of constructing the trajectory of the object, the results were measured for fusion with and without third level trajectory fusions. Improvement of the performance of inter-node fusion over



(a)



(b)

Figure 6.5: Object trajectory construction. (a) F-Scores. (b) Precision comparison with inter-node fusion.

Table 6.2: Mapping from Transportation Mode to Concept Type

Transportation Mode	Concept Type
Walk	Human
Car	Vehicle
Taxi	Vehicle
Bus	Vehicle
Train	Vehicle

intra-node fusion has very little effect since it is impossible to continuously track an object without inter-node fusion. Indeed, this is because intra-node tracking is simply limited by the detection capability of the sensor nodes. The result of experiments relating to the effect of the construction of the trajectory of the object is presented in Figure 6.5a. From the experimental results, it has been observed that the construction of the object's trajectory significantly improves the overall performance of the object tracking approach.

In order to see the improved performance of the object tracking algorithms with different types of concepts, the transport mode labels in the dataset were mapped to simulation concept types. Table 6.2 shows the mapping of the concept types "Vehicle" and "Human" because there is no label associated with the type of concept "Animal" in the dataset.

Table 6.3 displays the F-measure scores for a selected set of datasets based on the scenarios chosen for validating the proposed approach. These datasets have trajectories of the concept types "Vehicle" and "Human". Depending on the results obtained, the trajectories belonging to the human type are well detected with respect to the type of vehicle. The best performance of human objects is the difference in speed between moving objects. Since a human typically moves more slowly than a vehicle, a sensor node can track humans more easily and more accurately than a vehicle. In other words, the vehicle sensor data collected by the sensors is much more scarce than the human sensor data.

Table 6.3: F-measure Comparison for Concept Types

Dataset ID	Vehicle	Human
062	0.89	0.91
085	0.87	0.87
128	0.89	0.91
153	0.91	0.93

6.2 Fuzzy Logic Experiments

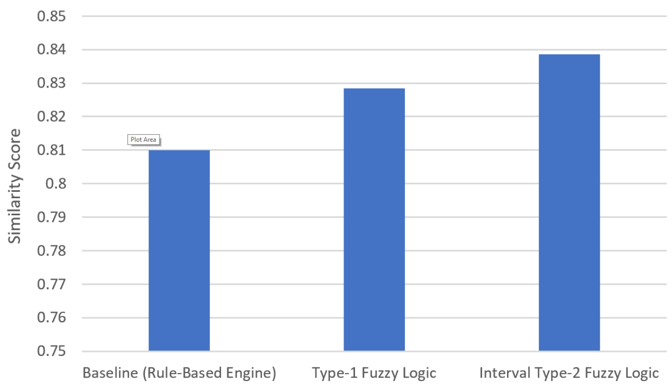
To test fuzzy data model and proposed interval type-2 fuzzy classification algorithm, several experiments was conducted. In the first experiment, the object trajectory construction was tested by applying the fuzziness to trajectory concept. Afterwards, experiments on classification algorithms were extended to include the new IT2 FLS based algorithm for both prediction and anomaly detection.

6.2.1 Global Trajectory Construction

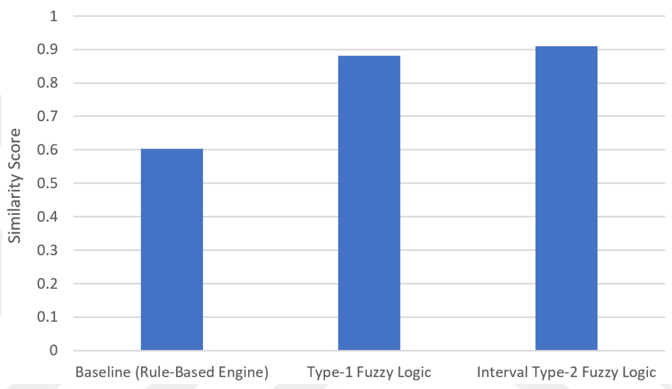
While generating the global trajectories, throughout this thesis study, first a rule based engine without using fuzzy logic has been used in multilevel object tracking, and this was the baseline of this study. Then, type-1 fuzzy logic have been added by replacing the existing rule-based engine with a semi-automatic fuzzy based rule engine. Finally, fuzziness in this dissertation was improved by using interval type-2 fuzzy logic, and this experiment evaluates all three approaches for global trajectory construction. Experiments were conducted using both real world datasets and synthetic trajectories by applying k-fold cross validation.

Figure 6.6 gives the results of the experiment for each datasets. Scores are evaluated by computing Hausdorff Similarity between the extracted global trajectory and actual trajectories using the following equations:

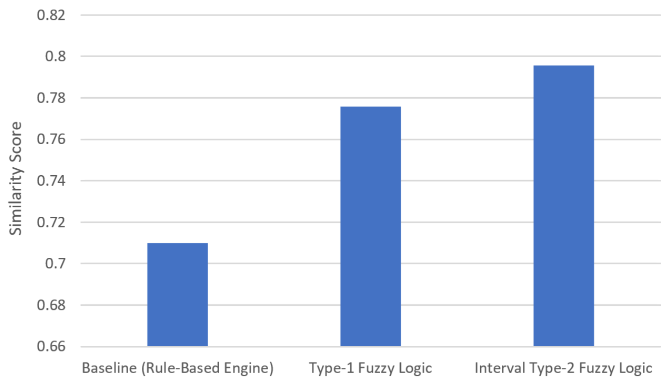
$$S = (\sum_{n=1}^{i}(\sum_{m=1}^{j} H(T_{glb}, T_m))/j)/i \tag{6.4}$$



(a)



(b)



(c)

Figure 6.6: Fuzzy logic usage compared to non-fuzzy approach in trajectory construction. (a) Synthetically generated data. (b) GeoLife Trajectories dataset. (c) Marine Cadastre dataset.

Table 6.4: The Number of Rules Comparison for Type-1 and Interval Type-2

Dataset	Type-1	Inverval Type-2
Synthetic	41	32
GeoLife Trajectories	68	40
Marine Cadastre	24	21

where H is the number of Hausdorff Similarity, j is the number of actual trajectories for a test scenario, and i is the number of test scenarios.

Even though interval type-2 performed best in Figure 6.6a, results are very close to each other for with and without fuzzy logic systems. The reason for these similar scores is that these fuzzy logic systems are built on top of the baseline by using the synthetically generated data. The performance on the GeoLife dataset is better than the Maritime dataset in this experiment which can be explained by the high frequency of the GeoLife dataset. While the average interval for sensor data is around 5-10 seconds in GeoLife trajectories, the time between two maritime data is around 1-3 minutes. Therefore the construction of the trajectories on the GeoLife dataset outperforms as it has denser data.

Table 6.4 gives the comparison of number of rules generated for each datasets on both type-1 and interval type-2 fuzzy logic systems. It is clearly seen that the number of rules in the rulebase is less if IT2 FLS is used instead of T1 FLS. Depending on the rotations and differences in the navigational properties such as speed and direction changes, number of the generated fuzzy rules are increasing or decreasing.

The number of rules in the rulebase of FLS will be less if IT2 FLS is used instead of T1 FLS, because the ability of representing more uncertainties using the footprint of uncertainty (FOU) which enables to cover the input/output situations with fewer fuzzy sets. In order to justify this, assume that there was a simple trajectory such as a straight line. We could train the T1 and IT2 fuzzy logic systems to generate automatically the fuzzy rules as specified in Section 5.2.3.2. There would be 6 fuzzy rules in order to setup the type-1 fuzzy logic system.

- IF sameDirection AND sameLocation AND realTime THEN sameTrajectory

- IF sameDirection AND sameLocation AND realTime THEN sameTrajectory
- IF similarDirection AND sameLocation AND realTime AND smallChange-Speed THEN similarTrajectory
- IF similarDirection AND smallChangeLocation AND newTime AND small-ChangeSpeed THEN similarTrajectory
- IF similarDirection AND bigChangeLocation AND newTime AND smallChange-Speed THEN similarTrajectory
- IF similarDirection AND bigChangeLocation AND newTime AND smallChange-Speed THEN differentTrajectory
- IF similarDirection AND smallChangeLocation AND realTime AND small-ChangeSpeed THEN similarTrajectory

The interval type-2 fuzzy logic system would use only 4 fuzzy rules.

- IF sameDirection AND sameLocation AND realTime THEN sameTrajectory
- IF sameDirection AND sameLocation AND realTime THEN sameTrajectory
- IF similarDirection AND sameLocation AND realTime AND smallChange-Speed THEN similarTrajectory
- IF similarDirection AND smallChangeLocation AND newTime AND small-ChangeSpeed THEN similarTrajectory
- IF similarDirection AND bigChangeLocation AND newTime AND smallChange-Speed THEN similarTrajectory

Missing two rules in IT2 FLS have been covered by the other IT2 fuzzy rules. Table 6.5 shows the mapping of how one IT2 fuzzy rule covers two T1 fuzzy rules. The former IT2 fuzzy rule uses FOU of the Time Input MF, and the latter IT2 fuzzy rule utilizes the ability of representing more uncertainties of the Similarity Output MF. For instance, Time MF for T1 an IT2 FLS have been given in Section refsec:45b.

FOU defined by *newTime* in the Time MF of interval type-2 FLS covers the values identified by both *realTime* and *newTime* in the Time MF of the type-1 FLS. Because of the the overlap between the upper/lower boundaries of *newTime* in IT2 and the fuzzy values *realTime* and *newTime* in T1, one IT2 fuzzy rule can replace two T1 fuzzy rules as given in Table 6.5.

T1 - Time MF:

- *realTime*
  - Trapezoidal [0, 0, 5, 10]
- *newTime*
  - Trapezoidal [5, 15, 30, 60]

IT2 - Time MF:

- *newTime*
  - Upper - Trapezoidal [3, 15, 30, 63]
  - Lower - Trapezoidal [7, 15, 30, 57]

In order to satisfy that the results of the experiment are significant, an one-tailed paired T-Test was conducted on the experiment results. With the t-test, we expected to show that the interval type-2 fuzzy logic statistically performs better than the type-1 fuzzy logic, and type-1 fuzzy logic performs better than the baseline, rule-based engine without fuzzy logic.

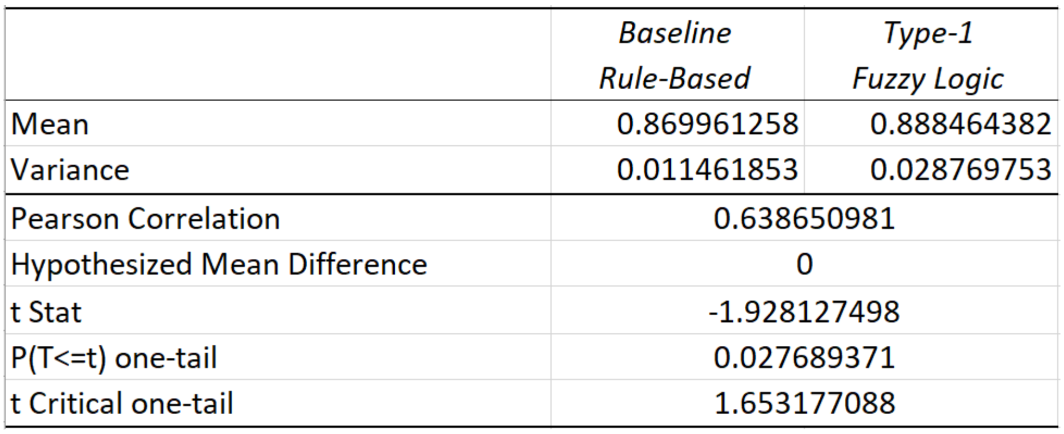
Microsoft Excel has been used to calculate the t-test values and Figure 6.7 gives the T-Test results.

As the p value in Figure 6.7a is less than the 0.05 which is the scientific threshold for T-Test, type-1 fuzzy logic algorithm significantly performs better than the baseline algorithm which is a rule-based engine without any fuzzy logic.

Figure 6.7b is the T-Test results for type-1 fuzzy logic and interval type-2 fuzzy logic. In science, 0.95 confidence is the threshold and our p value is 0.00009 which is far less than the 0.05 which means Interval type-2 fuzzy logic statistically outperforms.

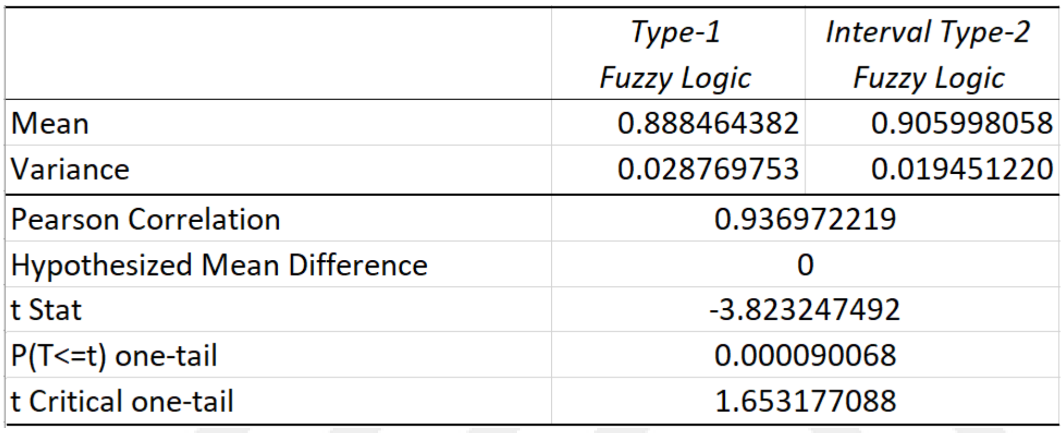
Table 6.5: Less Number of Rules in IT2 compared to T1

IT2 Fuzzy Rule	Covered Type-1 Fuzzy Rules
IF similarDirection AND smallChangeLocation AND newTime AND smallChangeSpeed THEN similarTrajectory	• IF similarDirection AND smallChangeLocation AND newTime AND smallChangeSpeed THEN similarTrajectory • IF similarDirection AND smallChangeLocation AND realTime AND smallChangeSpeed THEN similarTrajectory
IF similarDirection AND bigChangeLocation AND newTime AND smallChangeSpeed THEN similarTrajectory	• IF similarDirection AND bigChangeLocation AND newTime AND smallChangeSpeed THEN similarTrajectory • IF similarDirection AND bigChangeLocation AND newTime AND smallChangeSpeed THEN differentTrajectory



	Baseline Rule-Based	*Type-1 Fuzzy Logic*
Mean	0.869961258	0.888464382
Variance	0.011461853	0.028769753
Pearson Correlation	0.638650981	
Hypothesized Mean Difference	0	
t Stat	-1.928127498	
P(T<=t) one-tail	0.027689371	
t Critical one-tail	1.653177088	

(a)



	Type-1 Fuzzy Logic	*Interval Type-2 Fuzzy Logic*
Mean	0.888464382	0.905998058
Variance	0.028769753	0.019451220
Pearson Correlation	0.936972219	
Hypothesized Mean Difference	0	
t Stat	-3.823247492	
P(T<=t) one-tail	0.000090068	
t Critical one-tail	1.653177088	

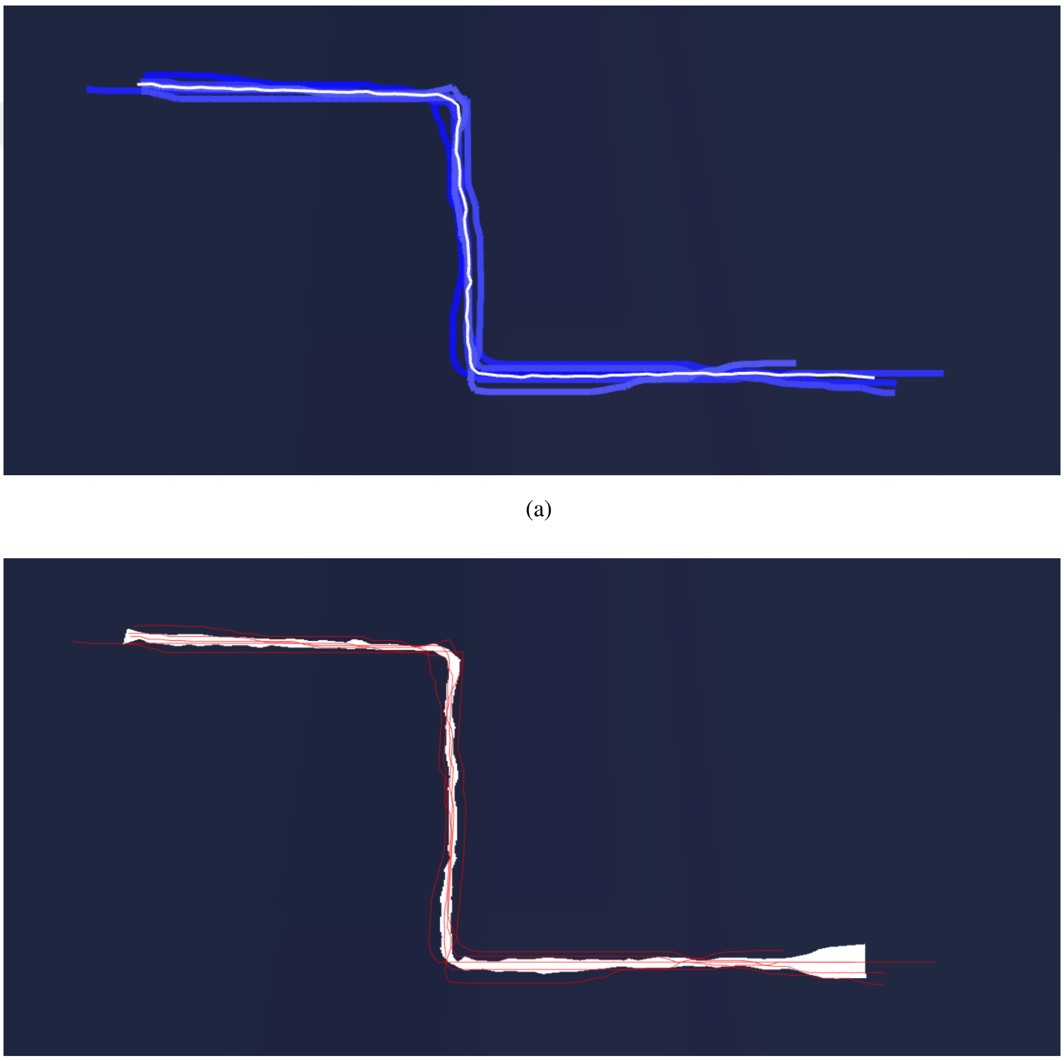
(b)

Figure 6.7: T-Test results. (a) For Baseline (Rule-Based Engine) and Type-1 Fuzzy Logic. (b) For Type-1 Fuzzy Logic and Interval Type-2 Fuzzy Logic.

6.2.2 Fuzzy Trajectory Extraction

After introducing the fuzzy geometry, to be able to see the fuzzy trajectories, the trajectory extraction code had to be adapted to export trajectories into KML not as a line, but as a polygon. Figure 6.8 shows representation of an extracted global trajectory as both a regular and a fuzzy trajectory on synthetically generated data. In Figure 6.8a, thick blue trajectories represent the source trajectories detected by the sensor nodes, and the white trajectory is the extracted global trajectory. In Figure 6.8b, thin red trajectories represent the source trajectories detected by the sensor nodes, and the thick white trajectory is the extracted global fuzzy trajectory.

It can be seen that the last part of the fuzzy trajectory is exceptionally wider compared to the other segments throughout the trajectory. The reason for the higher uncertainty is some of the trajectories are ended before or after the fuzzy trajectory's last point. Therefore, the distance between the last point of the shorter trajectory and the longer trajectory are mapped to the same fuzzy point that is the last point of the fuzzy trajectory, so the coverage of the fuzzy point is effected by the sparsely positioned trajectory data.



(a)

(b)

Figure 6.8: Comparison of representation of regular and fuzzy trajectories. (a) Regular trajectory as a line. (b) Fuzzy trajectory as a polygon.

6.2.3 Prediction and Anomaly Detection Analytics

Well-known classification algorithms were used as baseline for the type-1 fuzziness and interval type-2 fuzzy logic based classification algorithms. Experiments were conducted to test whether some improvements could be gained in prediction performance, or not.

Figure 6.9 shows the result of the experiments on synthetically generated data. Results show that type-1 and interval type-2 FLS perform better than the other algorithms. The reason might be that these fuzzy logic systems are built using the synthetic data.

Figure 6.10 and Figure 6.11 shows the result of the experiments on real world datasets. Using fuzzy logic in prediction surely increased the accuracy of forecasting the actions to be taken. In the GeoLife Trajectories dataset, results show that interval type-2 FLS performs better than the type-1 FLS and most of the other algorithms. Type-1 is also better than the SMO and the Logistic Regression algorithms. In this experiment, the performance on the Maritime Cadastre dataset is better than the GeoLife Trajectories dataset which can be clarified by the blunter trajectories with more linear fashion.

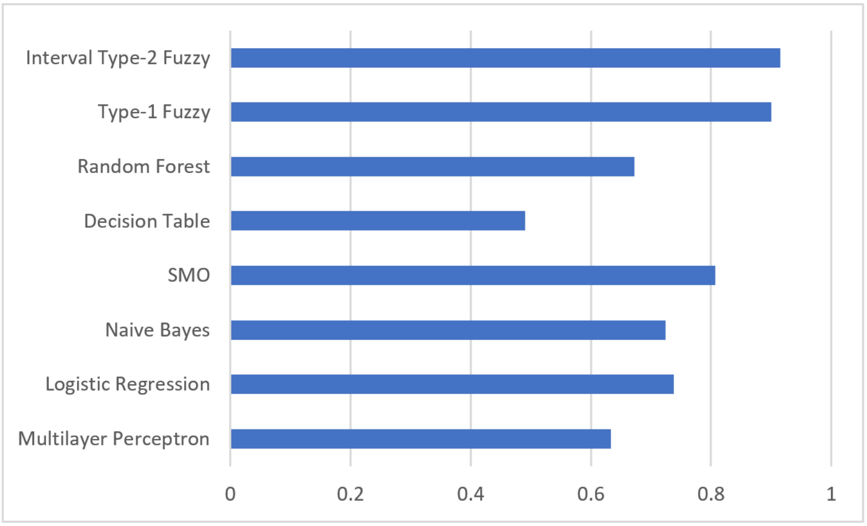


Figure 6.9: F-measure score comparison for proposed IT2 prediction algorithm on synthetically generated data.

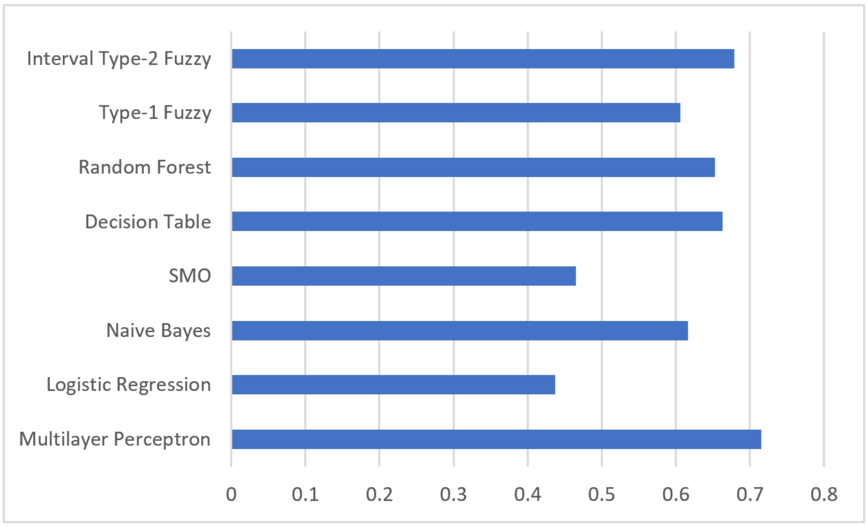


Figure 6.10: F-measure score comparison for proposed IT2 prediction algorithm on GeoLife Trajectories dataset.

It can be also observed that not only the fuzzy based algorithms but also the other classification algorithms. Therefore, another reason why the algorithms outperform on the Maritime dataset is that the distance between two different Maritime trajectories is further than the GeoLife trajectories.

In another experiment, the anomaly detection performance was analyzed by collecting the outputs of the classification algorithms including the type-1 and interval type-2 fuzzy algorithms in order to identify whether the fuzzy logic is able to capture the anomaly, or not.

Figure 6.12 shows results of the classifications for each algorithm on anomaly scenarios. A synthetic scenario has been generated in order to produce an anomaly where a moving object changes its trajectory from one to another. Success rate has been calculated by whether the algorithm can detect the anomaly as soon as it starts or not.

Almost all the algorithms, except SMO and MultiLayer Perceptron, achieved to classify trajectories, but to identify the anomalies immediately, fuzzy logic algorithms, Random Forest and Naive Bayes are the best. Logistic Regression and Multilayer Perceptron have oscillation problem which makes them unreliable for the anomaly detection case. Proposed IT2 FLS can be used in coordination with Random Forest or Naive Bayes algorithms to gain some performance in trajectory analytics.

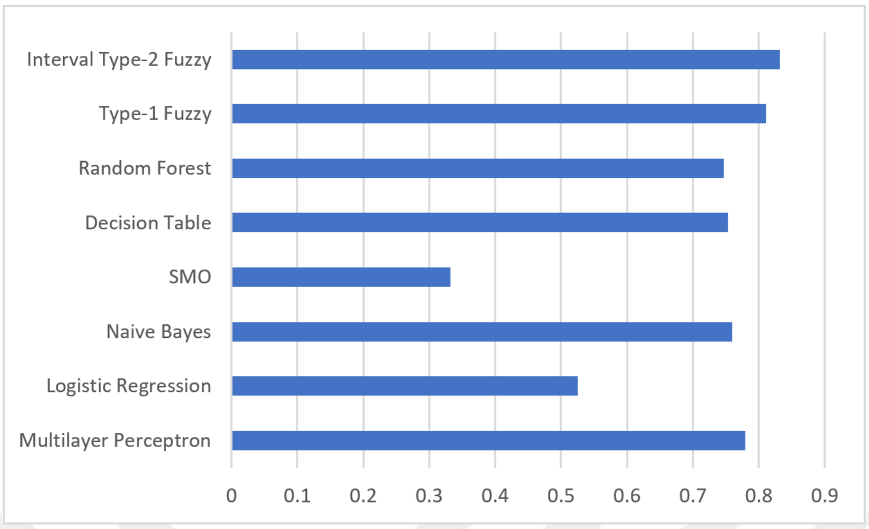


Figure 6.11: F-measure score comparison for proposed IT2 prediction algorithm on Marine Cadastre dataset.

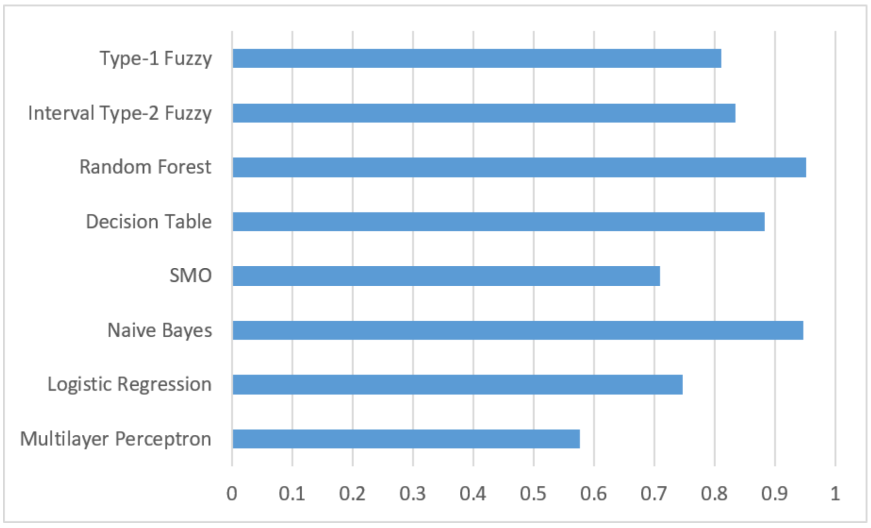


Figure 6.12: Anomaly detection success rates of fuzzy algorithms on synthetically generated data.

The prediction intervals (PI) are composed of an upper and lower boundary with a certain probability level. The prediction interval coverage probability (PICP) and the prediction interval normalized average width (PINAW) metrics are the performance indicators to quantitatively evaluate the performance of PI.

PICP is used to describe the reliability of the constructed PIs by quantifying the number of measured values belong to the interval defined by the IT2 fuzzy logic system.

$$PICP = \frac{1}{n}\sum_{n=1}^{k} \mu_k \tag{6.5}$$

where n is the number of measurements, and μ_k = 1 if $y_k \in [L_k, U_k]$, otherwise μ_k = 0. For the kth testing input, y is the target value, L and U are the lower and upper bound of each PI. A PICP value with closer to the 1.0 has a lower false rate.

PINAW is used to measure the width of the interval and characterizes the sharpness of the PIs. In other words, it is the average width of PIs as a percentage of the underlying target range.

$$PINAW = \frac{1}{nr}\sum_{n=1}^{k}(U_k - L_k) \tag{6.6}$$

where $r = y_{max} - y_{min}$ in which represents the range of the target values and is used to normalize the average width of PIs.

The evaluation of the PICP and PINAW metrics of several test cases for various object types are given in Table 6.6. It shows that most of the results were aligned with the expected maximum coverage probability with the minimum interval width. Depending on the object type, there can be performance differences which is expected since number of collected sensor data might be decreased and fuzziness and unpredictability can be increased.

6.3 Remarks

Many experiments were conducted on various scenarios using real-life datasets, which are GeoLife Trajectories and Maritime Cadastre Datasets, and as well as synthetically created dataset using the simulator developed for this thesis. The results of the experiments on both datasets show that proposed object tracking approach performs quite well and it is robust to be used for other application domains. From the experiments that It was observed that, the multi-layer fusion approach boosts the performance of object tracking.

Regarding the trajectory analytics, experimental results showed that to do analytics on trajectories there is not one best algorithm for all cases. While an algorithm is successful for one case, it fails for the other case. Performance of the fuzzy prediction algorithm is better than the well-known classification algorithms for our application domain and dataset.

Lastly, interval type-2 fuzzy logic experiments validated that proposed interval type-2 fuzzy logic system performs better in prediction, and also can be used for anomaly detection in coordination with Random Forest or Naive Bayes algorithms.

Table 6.6: PICP and PINAW evaluation for the IT2 fuzzy algorithm.

Object	Test Cases		
Type	Case1	Case2	Case3
Type1:			
PICP (%)	68.97	53.84	46.78
PINAW (%)	9.30	11.04	16.16
Type2:			
PICP (%)	71.81	56.81	53.75
PINAW (%)	10.32	8.95	10.06
Type3:			
PICP (%)	62.04	58.70	38.67
PINAW (%)	13.32	26.22	21.92

CHAPTER 7

CONCLUSIONS

This first aim of this thesis was to represent the wireless sensor networks with multimedia data in a graph-based big data model. The implementation was composed of two main modules. The former module was the implementation of the proposed data model, and the latter module was the simulation which includes a simulator to produce synthetic big sensor multimedia data and the simulation infrastructure which represents the objects moving in a wireless multimedia sensor network.

The focus domain for this thesis was the surveillance applications, and proposed graph-based data model was designed accordingly. The network topology and the data flow between each sensor nodes, gateways and sink were modeled, therefore all static and kinetic data within the WMSN was able to be stored, which can be treated as big data. The database to store big data was the NoSQL graph database, called OrientDB. Because graph databases are good at representation of complex relations and scalable to store big multimedia sensor data.

The WMSN prototype system with millions of data was simulated to test the proposed graph-base data model. The query performance was tested with many complex scenarios and it was shown that generated millions of synthetic data can be efficiently queried on proposed graph-based data.

Also, a new unsupervised object tracking approach using the big graph-based data model was introduced. The object tracking approach includes three main algorithms: intra-node fusion, inter-node fusion, and object trajectory construction. To validate the proposed algorithms, several experiments was performed on different scenarios using real datasets, GeoLife Trajectories and Marine Cadastre, as well as synthetically generated data using the simulator developed for this thesis. The results of the experiments on both datasets showed that the proposed object tracking approach is working well and is robust for use in other application domains. From the experiments it was observed that the proposed multi-level fusion approach improves the performance of object tracking.

Trajectory analyzes, such as prediction and anomaly detection, were applied on extracted global trajectories, and well-known classification algorithms were compared for different trajectories. Experimental results showed that the best algorithm for predicting all possible trajectories does not exist in all cases.

Afterwards, fuzzy logic was used to leverage the performance of both trajectory generation and trajectory analytics. Experiments showed that usage of fuzzy logic with type-1 or interval type-2 provides performance improvement.

7.1 Discussion

The research in this dissertation is based on the research project supervised by TUBITAK (The Scientific and Technological Research Council of Turkey) under Grant No. 114R082. The aim of the project was increasing accuracy of transferred information and the wireless network energy efficiency. A wireless sensor network consuming less energy than currently used was constructed and realized.

This dissertation starts with studies on big data analytics and how to store and utilize the generated data in the wireless sensor network. This section is an introduction to sensor networks, big data, NoSQL databases. After that, thesis motivation, the problem, and contributions of this dissertation are provided.

Thereafter, graph based big data model is presented. This chapter covers comparison of relational database and NoSQL graph datases as well as the graph database tool selection for this thesis.

As the focus of this dissertation is data analytics, a data simulator was needed and developed for the purpose of generating big data for a period of time. In addition to be able to feed some external datasets in the format of wireless sensor networks, data flow in the network has to be simulated as if data is collected and fused by the sensor nodes. In this sense, a simulator is designed and implemented in Java language by using several software products like graph databases, and publish/subscribe engines.

Then the proposed approach for object tracking and global trajectory extraction is introduced. Baseline of the algorithm is challenged by the fuzzy logic with both type-1 and interval type-2 systems. In this context, multi-level tracking algorithm and fuzzy logic systems are provided in detail.

7.2 Future Work

A possible future work on analytics is to propose a hybrid approach using the best performing classification algorithm according to the nature of the trajectory. This way, trajectory analytics can be utilized by the advantage of various algorithms.

There are other machine learning algorithms based on Long-Short Term Memory (LSTM) or Recurrent Neural Networks (RNNs) have been published recently for trajectory prediction. It can be considered to analyze these new algorithms to improve the performance of the proposed approach.

In addition, as another possible research topic can be usage of container based deployments. Installation of OrientDB and Apache ActiveMQ servers can be moved into the Docker containers which can provide easy to go setup and scaling.

Another future work can be investigation of automatically calculating the ε and *Node-Buffer* parameters based on the dataset. This way, some enhancement in the performance of the algorithms can be achieved.

REFERENCES

[1] M. Civelek, *A Lightweight Wireless Multimedia Sensor Network Architecture With Object Detection And Classification Capability*. PhD thesis, Middle East Technical University, 2017.

[2] K. Ashton, "That 'internet of things' thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[3] P. Patel, A. Pathak, T. Teixeira, and V. Issarny, "Towards application development for the internet of things," in *Proceedings of the 8th Middleware Doctoral Symposium*, p. 5, ACM, 2011.

[4] S. Alam, M. M. Chowdhury, and J. Noll, "Senaas: An event-driven sensor virtualization approach for internet of things cloud," in *Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on*, pp. 1–6, IEEE, 2010.

[5] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[6] S. Hong, D. Kim, M. Ha, S. Bae, S. J. Park, W. Jung, and J.-E. Kim, "Snail: an ip-based wireless sensor network approach to the internet of things," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 34–42, 2010.

[7] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer networks*, vol. 51, no. 4, pp. 921–960, 2007.

[8] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, and P. Christen, "Sensor discovery and configuration framework for the internet of things paradigm," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pp. 94–99, IEEE, 2014.

[9] X. Li and S. Moh, "Middleware systems for wireless sensor networks: A com-

parative survey," *Contemporary Engineering Sciences*, vol. 7, no. 13, pp. 649–660, 2014.

[10] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker, "The sensor network as a database," tech. rep., Citeseer, 2002.

[11] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*. " O'Reilly Media, Inc.", 2015.

[12] A. Moniruzzaman and S. A. Hossain, "Nosql database: New era of databases for big data analytics-classification, characteristics and comparison," *arXiv preprint arXiv:1307.0191*, 2013.

[13] M. Chen, S. Mao, and Y. Liu, "Big data: a survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[14] J. Han, E. Haihong, G. Le, and J. Du, "Survey on nosql database," in *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pp. 363–366, IEEE, 2011.

[15] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins, "A comparison of a graph database and a relational database: a data provenance perspective," in *Proceedings of the 48th annual Southeast regional conference*, p. 42, ACM, 2010.

[16] U. Sivarajah, M. M. Kamal, Z. Irani, and V. Weerakkody, "Critical analysis of big data challenges and analytical methods," *Journal of Business Research*, vol. 70, pp. 263–286, 2017.

[17] S. Khalifa, Y. Elshater, K. Sundaravarathan, A. Bhat, P. Martin, F. Imam, D. Rope, M. Mcroberts, and C. Statchuk, "The six pillars for building big data analytics ecosystems," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 33, 2016.

[18] M. H. ur Rehman, V. Chang, A. Batool, and T. Y. Wah, "Big data reduction framework for value creation in sustainable enterprises," *International Journal of Information Management*, vol. 36, no. 6, pp. 917–928, 2016.

[19] S. A. Ahmed, D. P. Dogra, S. Kar, and P. P. Roy, "Trajectory-based surveillance analysis: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, pp. 1985–1997, July 2019.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 338–345, Morgan Kaufmann Publishers Inc., 1995.

[22] R. Kohavi, "The power of decision tables," in *Proceedings of the 8th European Conference on Machine Learning*, ECML'95, (Berlin, Heidelberg), pp. 174–189, Springer-Verlag, 1995.

[23] S. Le Cessie and J. C. Van Houwelingen, "Ridge estimators in logistic regression," *Applied statistics*, pp. 191–201, 1992.

[24] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on neural networks*, vol. 3, no. 5, pp. 683–697, 1992.

[25] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Advances in neural information processing systems*, pp. 507–513, 1998.

[26] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, pp. 345–377, March 1995.

[27] J. M. Mendel and R. I. B. John, "Type-2 fuzzy sets made simple," *IEEE Transactions on Fuzzy Systems*, vol. 10, pp. 117–127, April 2002.

[28] H. Hagras, "Type-2 flcs: A new generation of fuzzy controllers," *IEEE Computational Intelligence Magazine*, vol. 2, pp. 30–43, Feb 2007.

[29] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: Theory and design," *Trans. Fuz Sys.*, vol. 8, pp. 535–550, Oct. 2000.

[30] O. Diallo, J. J. Rodrigues, and M. Sene, "Real-time data management on wireless sensor networks: a survey," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 1013–1021, 2012.

[31] C. Hu, Z. Xu, Y. Liu, L. Mei, L. Chen, and X. Luo, “Semantic link network-based model for organizing multimedia big data,” *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 376–387, 2014.

[32] B. Bostan-Korpeoglu, A. Yazici, I. Korpeoglu, and R. George, “A new approach for information processing inwireless sensor network,” in *22nd International Conference on Data Engineering Workshops (ICDEW’06)*, pp. 34–34, IEEE, 2006.

[33] P. Zhang, Z. Yan, and H. Sun, “A novel architecture based on cloud computing for wireless sensor network,” in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering. Atlantis Press*, pp. 472–475, 2013.

[34] Y. Li, C. Wu, L. Guo, C.-H. Lee, and Y. Guo, “Wiki-health: A big data platform for health,” *Cloud Computing Applications for Quality Health Care Delivery*, p. 59, 2014.

[35] C. Jardak, P. Mähönen, and J. Riihijärvi, “Spatial big data and wireless networks: experiences, applications, and research challenges,” *IEEE Network*, vol. 28, no. 4, pp. 26–31, 2014.

[36] R. Angles and C. Gutierrez, “Survey of graph database models,” *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, p. 1, 2008.

[37] M. Levene and G. Loizou, “A graph-based data model and its ramifications,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 5, pp. 809–823, 1995.

[38] A. Manjeshwar and D. P. Agrawal, “Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks.,” in *Ipdps*, vol. 2, p. 48, 2002.

[39] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, “Pipenet: A wireless sensor network for pipeline monitoring,” in *2007 6th International Symposium on Information Processing in Sensor Networks*, pp. 264–273, IEEE, 2007.

[40] E. Felemban, “Advanced border intrusion detection and surveillance using wireless sensor network technology,” *International Journal of Communications, Network and System Sciences*, vol. 6, no. 5, p. 251, 2013.

[41] P. Chatterjee, S. C. Ghosh, and N. Das, “Load balanced coverage with graded node deployment in wireless sensor networks,” *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 2, pp. 100–112, 2017.

[42] M. L. Ruz, J. Garrido, J. Jiménez, R. Virrankoski, and F. Vázquez, “Simulation tool for the analysis of cooperative localization algorithms for wireless sensor networks,” *Sensors*, vol. 19, no. 13, p. 2866, 2019.

[43] S. A. Sert, H. Bagci, and A. Yazici, “Mofca: Multi-objective fuzzy clustering algorithm for wireless sensor networks,” *Applied Soft Computing*, vol. 30, pp. 151–165, 2015.

[44] S. Hadim and N. Mohamed, “Middleware: Middleware challenges and approaches for wireless sensor networks,” *IEEE distributed systems online*, vol. 7, no. 3, p. 1, 2006.

[45] L.-Y. Ho, J.-J. Wu, and P. Liu, “Distributed graph database for large-scale social computing,” in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 455–462, IEEE, 2012.

[46] E. Masazade, R. Niu, and P. K. Varshney, “Dynamic bit allocation for object tracking in wireless sensor networks,” *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5048–5063, 2012.

[47] O. Ozdemir, R. Niu, and P. K. Varshney, “Dynamic bit allocation for target tracking in sensor networks with quantized measurements,” in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 2906–2909, IEEE, 2010.

[48] G. Kayumbi, N. Anjum, and A. Cavallaro, “Global trajectory reconstruction from distributed visual sensors,” in *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pp. 1–8, IEEE, 2008.

[49] N. Anjum and A. Cavallaro, "Trajectory association and fusion across partially overlapping cameras," in *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, pp. 201–206, IEEE, 2009.

[50] D. Cheng, Y. Gong, J. Wang, Q. Hou, and N. Zheng, "Part-aware trajectories association across non-overlapping uncalibrated cameras," *Neurocomputing*, vol. 230, pp. 30–39, 2017.

[51] M. Fayyaz, "Classification of object tracking techniques in wireless sensor networks," *Wireless Sensor Network*, vol. 3, no. 04, p. 121, 2011.

[52] J. D. Mazimpaka and S. Timpf, "Trajectory data mining: A review of methods and applications," *Journal of Spatial Information Science*, vol. 2016, no. 13, pp. 61–99, 2016.

[53] A. Valsamis, K. Tserpes, D. Zissis, D. Anagnostopoulos, and T. Varvarigou, "Employing traditional machine learning algorithms for big data streams analysis: the case of object trajectory prediction," *Journal of Systems and Software*, vol. 127, pp. 249–257, 2017.

[54] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "Stgat: Modeling spatial-temporal interactions for human trajectory prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6272–6281, 2019.

[55] C. E. Verdonk Gallego, V. F. Gómez Comendador, M. A. Amaro Carmona, R. M. Arnaldo Valdés, F. J. Sáez Nieto, and M. García Martínez, "A machine learning approach to air traffic interdependency modelling and its application to trajectory prediction," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 356 – 386, 2019.

[56] B. Zhu, C. Qian, X. Pan, and H. Chen, "A trajectory-based deep sequential method for customer churn prediction," in *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, ICMLT 2020, (New York, NY, USA), p. 114–118, Association for Computing Machinery, 2020.

[57] C. Qian, R. Jiang, Y. Long, Q. Zhang, M. Li, and L. Zhang, "Vehicle trajectory modelling with consideration of distant neighbouring dependencies for desti-

nation prediction," *International Journal of Geographical Information Science*, vol. 33, no. 10, pp. 2011–2032, 2019.

[58] S. Pandhre, H. Mittal, M. Gupta, and V. N. Balasubramanian, "Stwalk: Learning trajectory representations in temporal graphs," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, CoDS-COMAD '18, (New York, NY, USA), p. 210–219, Association for Computing Machinery, 2018.

[59] Y. Ji, L. Wang, W. Wu, H. Shao, and Y. Feng, "A method for lstm-based trajectory modeling and abnormal trajectory detection," *IEEE Access*, vol. 8, pp. 104063–104073, 2020.

[60] S. Guillaume and B. Charnomordic, "Learning interpretable fuzzy inference systems with fispro," *International Journal of Information Sciences*, vol. 181, no. 20, pp. 4409–4427, 2011. Special Issue on Interpretable Fuzzy Systems.

[61] S. Guillaume and B. Charnomordic, "Fuzzy inference systems: an integrated modelling environment for collaboration between expert knowledge and data using fispro," *Expert Systems with Applications*, vol. 39, pp. 8744–8755, August 2012.

[62] J. Hohensohn and J. M. Mendel, "Two-pass orthogonal least-squares algorithm to train and reduce fuzzy logic systems," *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, pp. 696–700 vol.1, 1994.

[63] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision(ICCV)*, vol. 02, p. 1150, 09 1999.

[64] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[65] K. Mironov and M. Pongratz, "Fast knn-based prediction for the trajectory of a thrown body," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, pp. 512–517, 2016.

[66] G. Xu, X. Wang, X. Guo, S. Liang, and F. Wei, “The model of potential violation discovery based on knn and spatio-temporal trajectory of commercial vehicle,” in *2019 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 1466–1471, 2019.

[67] A. L. Duca, C. Bacciu, and A. Marchetti, “A k-nearest neighbor classifier for ship route prediction,” in *OCEANS 2017 - Aberdeen*, pp. 1–6, 2017.

[68] H. Ramadhan, Y. Yustiawan, and J. Kwon, “A constrained k-nearest neighbor approach for semantic indoor trajectory extraction,” in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 13–16, 2020.

[69] N. N. Karnik, J. M. Mendel, and Q. Liang, “Type-2 fuzzy logic systems,” *IEEE transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 643–658, 1999.

[70] D. Wu, “On the fundamental differences between interval type-2 and type-1 fuzzy logic controllers,” *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 5, pp. 832–848, 2012.

[71] H. Hagras, “Type-2 flcs: A new generation of fuzzy controllers,” *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 30–43, 2007.

[72] J. Buckley and E. Eslami, “Fuzzy plane geometry i: Points and lines,” *Fuzzy Sets and Systems*, vol. 86, no. 2, pp. 179 – 187, 1997.

[73] D. Ghosh and D. Chakraborty, “Analytical fuzzy plane geometry i,” *Fuzzy Sets and Systems*, vol. 209, pp. 66 – 83, 2012. Theme : Fuzzy numbers and Analysis.

[74] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, “Learning-based approach for online lane change intention prediction,” in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 797–802, IEEE, 2013.

[75] B. T. Morris and M. M. Trivedi, “Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis,” in *Advanced Video and Signal Based Surveillance, 2008. AVSS’08. IEEE Fifth International Conference on*, pp. 154–161, IEEE, 2008.

[76] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[77] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory.," *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, 2010.

[78] U. C. Guard, "Vessel traffic data," 2009-2017.

[79] S. Vasuhi and V. Vaidehi, "Target tracking using interactive multiple model for wireless sensor network," *Information Fusion*, vol. 27, pp. 41–53, 2016.

[80] K. Hirpara and K. Rana, "Energy-efficient constant gain kalman filter based tracking in wireless sensor network," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

[81] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831.

[82] M. A. Brovelli, M. Minghini, M. Molinari, and P. Mooney, "Towards an automated comparison of openstreetmap with authoritative road datasets," *Transactions in GIS*, vol. 21, no. 2, pp. 191–206, 2017.

[83] T. Zhang, S. Liu, C. Xu, B. Liu, and M.-H. Yang, "Correlation particle filter for visual tracking," *IEEE Transactions on Image Processing*, vol. 27, no. 6, pp. 2676–2687, 2018.

[84] P. Prasad and A. Gupta, "Moving object tracking and detection based on kalman filter and saliency mapping," in *Data Engineering and Intelligent Computing*, pp. 639–646, Springer, 2018.

APPENDIX A

FUZZY RULES

A.1 Type-1 Fuzzy Rules

Input Member Functions:

- Direction MF - [sameDirection, similarDirection, differentDirection]
- Time MF - [newTime, oldTime, realTime]
- Location MF - [sameLocation, smallChange, bigChange]
- Speed MF - [sameSpeed, smallChange, bigChange]

Output Member Functions:

- Similarity MF - [sameTrajectory, differentTrajectory]

Table A.1: T1 Fuzzy Rules Used in Intra-Node Fusion (Part 1/2)

Direction (Input)	Location (Input)	Time (Input)	Speed (Input)	Similarity (Output)
sameDirection	sameLocation	realTime	-	sameTrajectory
differentDirection	sameLocation	realTime	smallChange	similarTrajectory
differentDirection	smallChange	newTime	smallChange	similarTrajectory
differentDirection	bigChange	newTime	smallChange	similarTrajectory
differentDirection	bigChange	newTime	smallChange	differentTrajectory
similarDirection	sameLocation	realTime	smallChange	similarTrajectory
similarDirection	smallChange	newTime	smallChange	similarTrajectory
similarDirection	bigChange	newTime	smallChange	similarTrajectory
similarDirection	bigChange	newTime	smallChange	differentTrajectory
similarDirection	sameLocation	realTime	sameSpeed	similarTrajectory
differentDirection	sameLocation	realTime	sameSpeed	similarTrajectory
similarDirection	smallChange	newTime	bigChange	similarTrajectory
similarDirection	bigChange	newTime	bigChange	similarTrajectory
similarDirection	smallChange	newTime	sameSpeed	similarTrajectory
similarDirection	bigChange	newTime	sameSpeed	similarTrajectory
differentDirection	smallChange	newTime	sameSpeed	similarTrajectory
similarDirection	sameLocation	newTime	smallChange	similarTrajectory
similarDirection	sameLocation	realTime	bigChange	similarTrajectory
similarDirection	bigChange	newTime	bigChange	differentTrajectory
similarDirection	bigChange	newTime	sameSpeed	differentTrajectory
sameDirection	smallChange	newTime	sameSpeed	similarTrajectory
sameDirection	bigChange	newTime	smallChange	similarTrajectory
sameDirection	smallChange	newTime	smallChange	similarTrajectory
sameDirection	bigChange	newTime	smallChange	differentTrajectory
sameDirection	bigChange	newTime	bigChange	differentTrajectory

Table A.2: T1 Fuzzy Rules Used in Intra-Node Fusion (Part 2/2)

Direction (Input)	Location (Input)	Time (Input)	Speed (Input)	Similarity (Output)
sameDirection	sameLocation	newTime	bigChange	similarTrajectory
sameDirection	smallChange	newTime	bigChange	similarTrajectory
similarDirection	smallChange	newTime	smallChange	differentTrajectory
differentDirection	smallChange	newTime	smallChange	differentTrajectory
similarDirection	sameLocation	newTime	smallChange	differentTrajectory
differentDirection	sameLocation	newTime	smallChange	differentTrajectory
sameDirection	sameLocation	newTime	smallChange	differentTrajectory
sameDirection	smallChange	newTime	smallChange	differentTrajectory
differentDirection	sameLocation	newTime	smallChange	similarTrajectory
similarDirection	smallChange	newTime	bigChange	differentTrajectory
similarDirection	sameLocation	newTime	bigChange	differentTrajectory
sameDirection	sameLocation	newTime	bigChange	differentTrajectory
differentDirection	sameLocation	newTime	bigChange	differentTrajectory
sameDirection	smallChange	newTime	bigChange	differentTrajectory
differentDirection	smallChange	newTime	bigChange	differentTrajectory
differentDirection	bigChange	newTime	bigChange	differentTrajectory

Table A.3: T1 Fuzzy Rules Used in Inter-Node Fusion

Direction (Input)	Location (Input)	Time (Input)	Speed (Input)	Similarity (Output)
sameDirection	sameLocation	realTime	-	sameTrajectory
differentDirection	sameLocation	realTime	smallChange	sameTrajectory
differentDirection	smallChange	newTime	smallChange	sameTrajectory
differentDirection	bigChange	newTime	smallChange	sameTrajectory
differentDirection	bigChange	newTime	smallChange	similarTrajectory
similarDirection	smallChange	newTime	smallChange	sameTrajectory
similarDirection	sameLocation	realTime	smallChange	sameTrajectory
similarDirection	bigChange	newTime	smallChange	sameTrajectory
similarDirection	bigChange	newTime	smallChange	similarTrajectory
similarDirection	sameLocation	newTime	smallChange	sameTrajectory
similarDirection	sameLocation	realTime	sameSpeed	sameTrajectory
similarDirection	smallChange	newTime	sameSpeed	sameTrajectory
similarDirection	bigChange	newTime	sameSpeed	sameTrajectory
similarDirection	smallChange	newTime	bigChange	similarTrajectory
differentDirection	sameLocation	realTime	sameSpeed	sameTrajectory
differentDirection	smallChange	newTime	sameSpeed	sameTrajectory
similarDirection	smallChange	newTime	bigChange	sameTrajectory
similarDirection	sameLocation	realTime	bigChange	sameTrajectory
similarDirection	bigChange	newTime	bigChange	sameTrajectory
differentDirection	sameLocation	realTime	bigChange	sameTrajectory
sameDirection	smallChange	newTime	sameSpeed	sameTrajectory
sameDirection	smallChange	newTime	smallChange	sameTrajectory
sameDirection	bigChange	newTime	smallChange	sameTrajectory
sameDirection	bigChange	newTime	smallChange	similarTrajectory
differentDirection	sameLocation	newTime	smallChange	sameTrajectory

A.2 Interval Type-2 Fuzzy Rules

Input Member Functions:

- Direction MF - [sameDirection, similarDirection, differentDirection]
- Time MF - [newTime, oldTime, realTime]
- Location MF - [sameLocation, smallChange, bigChange]
- Speed MF - [sameSpeed, smallChange, bigChange]

Output Member Functions:

- Similarity MF - [sameTrajectory, similarTrajectory, differentTrajectory]

Table A.4: IT2 Fuzzy Rules Used in Intra-Node Fusion (Part 1/2)

Direction (Input)	Location (Input)	Time (Input)	Speed (Input)	Similarity (Output)
sameDirection	sameLocation	realTime	sameSpeed	sameTrajectory
differentDirection	sameLocation	realTime	smallChange	similarTrajectory
differentDirection	smallChange	newTime	smallChange	similarTrajectory
differentDirection	bigChange	newTime	smallChange	similarTrajectory
similarDirection	sameLocation	realTime	smallChange	similarTrajectory
similarDirection	smallChange	newTime	smallChange	similarTrajectory
similarDirection	bigChange	newTime	smallChange	similarTrajectory
similarDirection	sameLocation	newTime	smallChange	similarTrajectory
similarDirection	smallChange	newTime	sameSpeed	similarTrajectory
similarDirection	sameLocation	realTime	bigChange	similarTrajectory
similarDirection	smallChange	newTime	bigChange	similarTrajectory
similarDirection	bigChange	newTime	bigChange	similarTrajectory
similarDirection	smallChange	realTime	bigChange	differentTrajectory
similarDirection	bigChange	newTime	sameSpeed	similarTrajectory
differentDirection	sameLocation	realTime	sameSpeed	similarTrajectory
differentDirection	smallChange	newTime	sameSpeed	similarTrajectory
differentDirection	bigChange	newTime	sameSpeed	differentTrajectory
sameDirection	smallChange	newTime	smallChange	similarTrajectory
sameDirection	smallChange	newTime	sameSpeed	similarTrajectory
sameDirection	bigChange	newTime	sameSpeed	similarTrajectory
differentDirection	sameLocation	realTime	bigChange	similarTrajectory
similarDirection	sameLocation	realTime	sameSpeed	similarTrajectory
sameDirection	bigChange	newTime	smallChange	differentTrajectory
sameDirection	sameLocation	newTime	bigChange	similarTrajectory
differentDirection	sameLocation	newTime	smallChange	differentTrajectory

Table A.5: IT2 Fuzzy Rules Used in Intra-Node Fusion (Part 2/2)

Direction (Input)	Location (Input)	Time (Input)	Speed (Input)	Similarity (Output)
sameDirection	sameLocation	newTime	smallChange	differentTrajectory
differentDirection	bigChange	newTime	bigChange	differentTrajectory
sameDirection	bigChange	newTime	bigChange	differentTrajectory
sameDirection	smallChange	newTime	bigChange	differentTrajectory
differentDirection	sameLocation	newTime	bigChange	differentTrajectory
differentDirection	smallChange	newTime	bigChange	differentTrajectory
similarDirection	sameLocation	newTime	bigChange	differentTrajectory

Table A.6: IT2 Fuzzy Rules Used in Inter-Node Fusion

Direction (Input)	Location (Input)	Time (Input)	Speed (Input)	Similarity (Output)
sameDirection	sameLocation	realTime	sameSpeed	sameTrajectory
differentDirection	sameLocation	realTime	smallChange	sameTrajectory
differentDirection	smallChange	newTime	smallChange	sameTrajectory
similarDirection	sameLocation	realTime	smallChange	sameTrajectory
differentDirection	bigChange	newTime	smallChange	sameTrajectory
similarDirection	smallChange	newTime	smallChange	sameTrajectory
similarDirection	bigChange	newTime	smallChange	sameTrajectory
similarDirection	smallChange	newTime	sameSpeed	sameTrajectory
similarDirection	sameLocation	realTime	sameSpeed	sameTrajectory
similarDirection	sameLocation	realTime	bigChange	similarTrajectory
similarDirection	smallChange	newTime	bigChange	similarTrajectory
similarDirection	smallChange	realTime	bigChange	sameTrajectory
similarDirection	bigChange	newTime	bigChange	sameTrajectory
differentDirection	sameLocation	realTime	sameSpeed	sameTrajectory
differentDirection	smallChange	newTime	sameSpeed	sameTrajectory
differentDirection	sameLocation	realTime	bigChange	sameTrajectory
sameDirection	smallChange	newTime	sameSpeed	sameTrajectory
similarDirection	bigChange	newTime	sameSpeed	sameTrajectory
sameDirection	sameLocation	realTime	smallChange	sameTrajectory
sameDirection	bigChange	newTime	sameSpeed	sameTrajectory
sameDirection	smallChange	newTime	smallChange	sameTrajectory
sameDirection	bigChange	newTime	smallChange	sameTrajectory

APPENDIX B

COMPLEXITY ANALYSIS OF ALGORITHMS

Analysis of the algorithms is an important task in complexity theory to provide an estimation of the computer resources required by an algorithm while solving a problem. Algorithm analysis is the determination of the amount of time known as time complexity, which is the efficiency or execution time of an algorithm listed as a function that relates the input length to the number of steps. In other words, it is the process of analyzing the algorithm's problem-solving capacity in terms of the size of memory required and the time needed during computation. In general, we perform the following types of analysis:

- Worst case, the maximum number of steps performed for execution

- Best case, the minimum number of steps performed for execution

- Average case, an average number of steps needed for execution

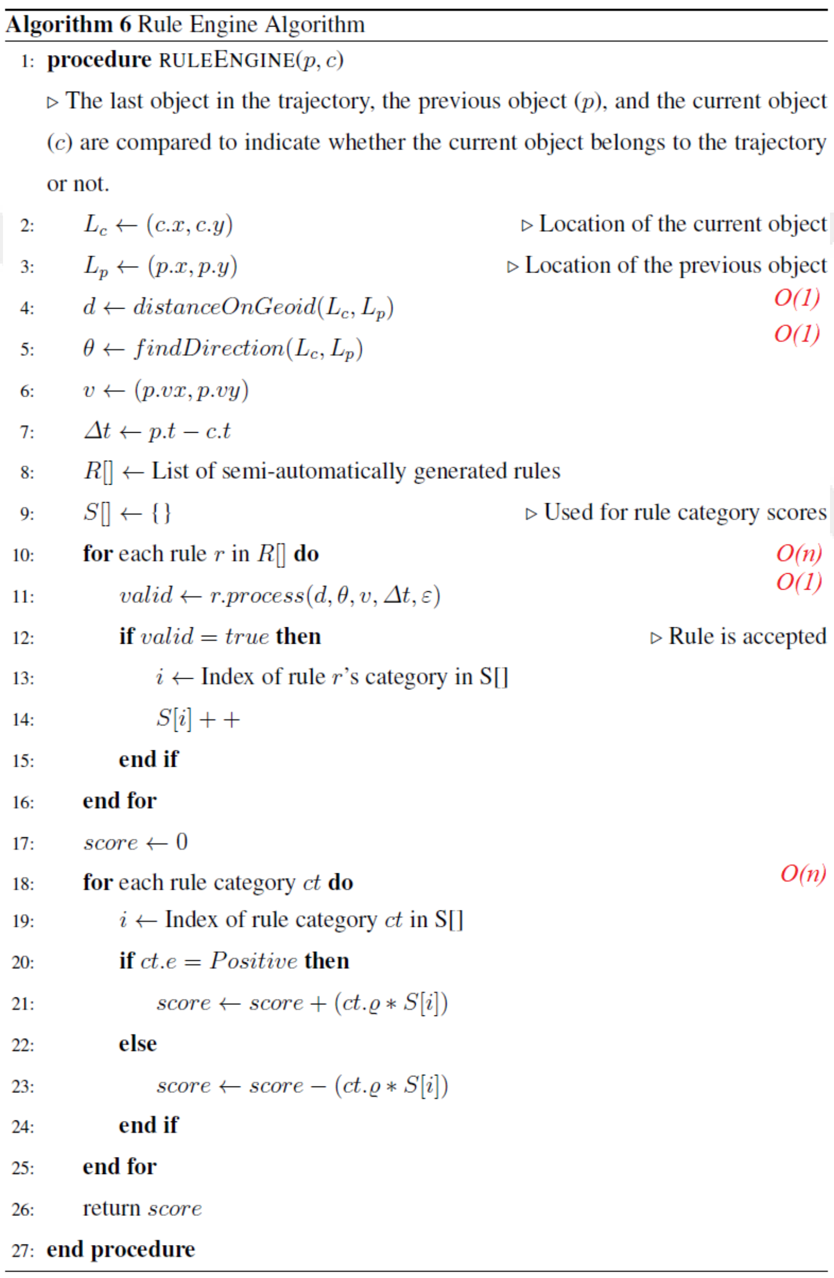
To be able to asses the resource consumed by an algorithm while execution, there are several strategies to be used and Asymptotic Analysis is one of them for assessment of the complexity of the algorithms.

The asymptotic behavior of a function refers to the growth of that function when the number of steps becomes large. We generally ignore small values of n, as we generally want to estimate the slowness of the program on large inputs.

B.1 Rule Engine Algorithm

Rule Engine algorithm iterates over the list of rules and executes each rule which is an execution of an instruction.

As shown in the below figure that the 10th and 18th lines of the algorithm iterates n times, the complexity of this algorithm is $O(n)$.



Algorithm 6 Rule Engine Algorithm

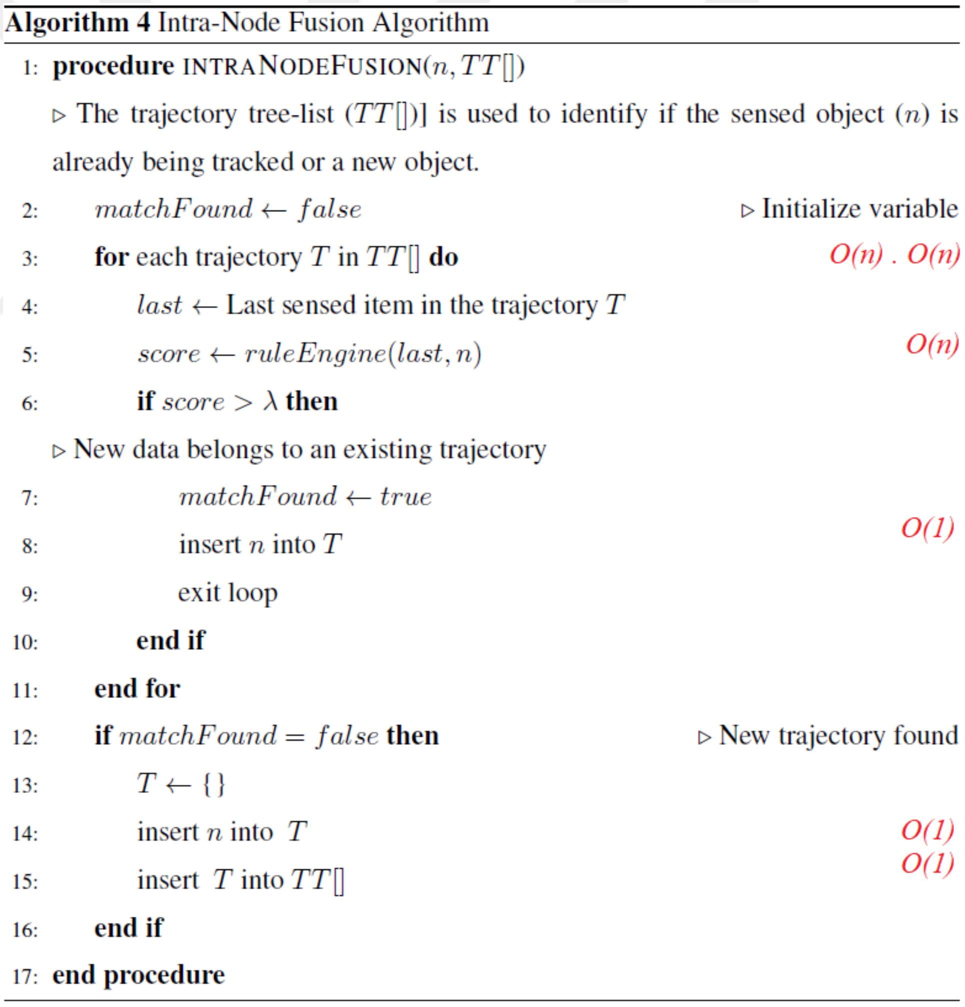
```
 1: procedure RULEENGINE($p, c$)
        ▷ The last object in the trajectory, the previous object ($p$), and the current object
        ($c$) are compared to indicate whether the current object belongs to the trajectory
        or not.
 2:     $L_c \leftarrow (c.x, c.y)$                              ▷ Location of the current object
 3:     $L_p \leftarrow (p.x, p.y)$                              ▷ Location of the previous object
 4:     $d \leftarrow distanceOnGeoid(L_c, L_p)$                                     O(1)
 5:     $\theta \leftarrow findDirection(L_c, L_p)$                                  O(1)
 6:     $v \leftarrow (p.vx, p.vy)$
 7:     $\Delta t \leftarrow p.t - c.t$
 8:     $R[] \leftarrow$ List of semi-automatically generated rules
 9:     $S[] \leftarrow \{\}$                                   ▷ Used for rule category scores
10:     for each rule $r$ in $R[]$ do                                              O(n)
11:         $valid \leftarrow r.process(d, \theta, v, \Delta t, \varepsilon)$                   O(1)
12:         if $valid = true$ then                                     ▷ Rule is accepted
13:             $i \leftarrow$ Index of rule $r$'s category in S[]
14:             $S[i]++$
15:         end if
16:     end for
17:     $score \leftarrow 0$
18:     for each rule category $ct$ do                                             O(n)
19:         $i \leftarrow$ Index of rule category $ct$ in S[]
20:         if $ct.e = Positive$ then
21:             $score \leftarrow score + (ct.\varrho * S[i])$
22:         else
23:             $score \leftarrow score - (ct.\varrho * S[i])$
24:         end if
25:     end for
26:     return $score$
27: end procedure
```

Figure B.1: Complexity analysis of Rule Engine algorithm.

B.2 Intra-Node Fusion Algorithm

Intra-Node Fusion algorithm iterates over the trajectory list and uses Rule Engine algorithm to decide whether trajectories can be merged or not.

As shown in the below figure that the 3rd line iterates n times over the $ruleEngine$ algorithm whose complexity is $O(n)$, the complexity of this algorithm is $O(n^2)$.



Algorithm 4 Intra-Node Fusion Algorithm

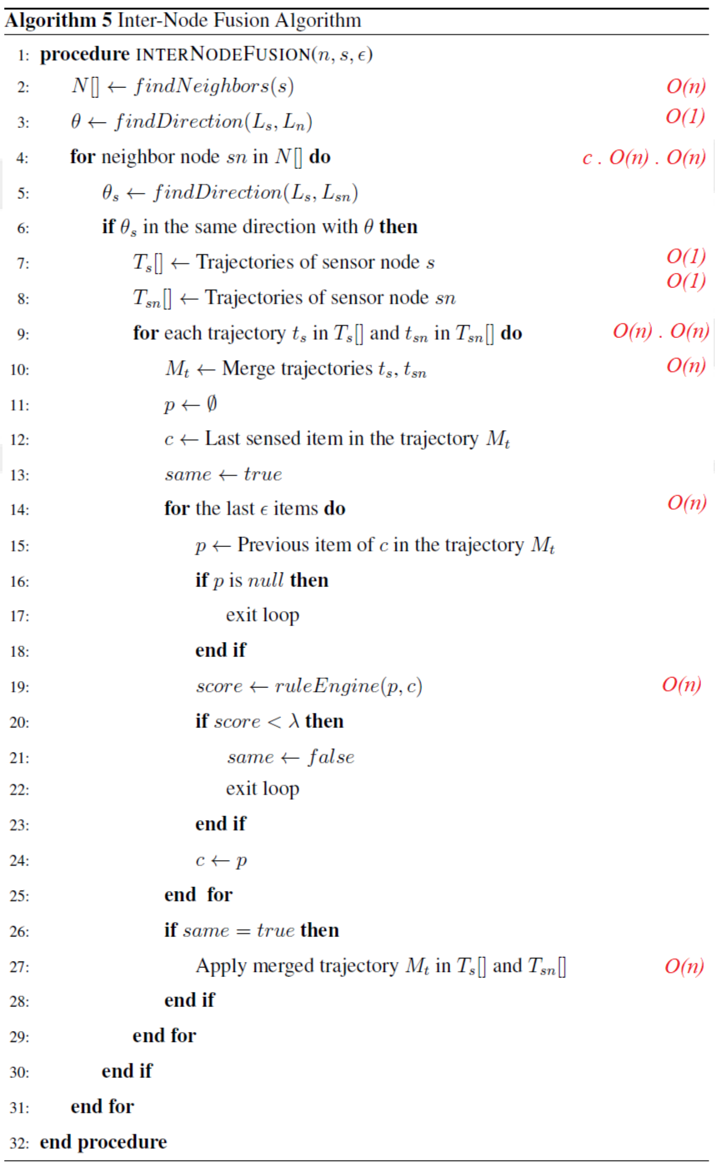
```
 1: procedure INTRANODEFUSION($n, TT[]$)
    ▷ The trajectory tree-list ($TT[]$)] is used to identify if the sensed object ($n$) is
    already being tracked or a new object.
 2:     $matchFound \leftarrow false$                                  ▷ Initialize variable
 3:     for each trajectory $T$ in $TT[]$ do                            O(n) . O(n)
 4:         $last \leftarrow$ Last sensed item in the trajectory $T$
 5:         $score \leftarrow ruleEngine(last, n)$                      O(n)
 6:         if $score > \lambda$ then
    ▷ New data belongs to an existing trajectory
 7:             $matchFound \leftarrow true$
 8:             insert $n$ into $T$                                     O(1)
 9:             exit loop
10:         end if
11:     end for
12:     if $matchFound = false$ then                                ▷ New trajectory found
13:         $T \leftarrow \{\}$
14:         insert $n$ into $T$                                         O(1)
15:         insert $T$ into $TT[]$                                      O(1)
16:     end if
17: end procedure
```

Figure B.2: Complexity analysis of Intra-Node Fusion algorithm.

B.3 Inter-Node Fusion Algorithm

Inter-Node Fusion algorithm iterates over the neighbor sensor nodes to find the similar trajectories by comparing the trajectories using the rule engine.

As shown in the below figure that the 4th line iterates over the neighbor sensor nodes, which is a constant number of nodes ($1 <= c <= 8$), to compare the trajectories with each other whose complexity is $O(n^2)$, the complexity of this algorithm is $O(c.n^2)$, which is $O(n^2)$.



Algorithm 5 Inter-Node Fusion Algorithm

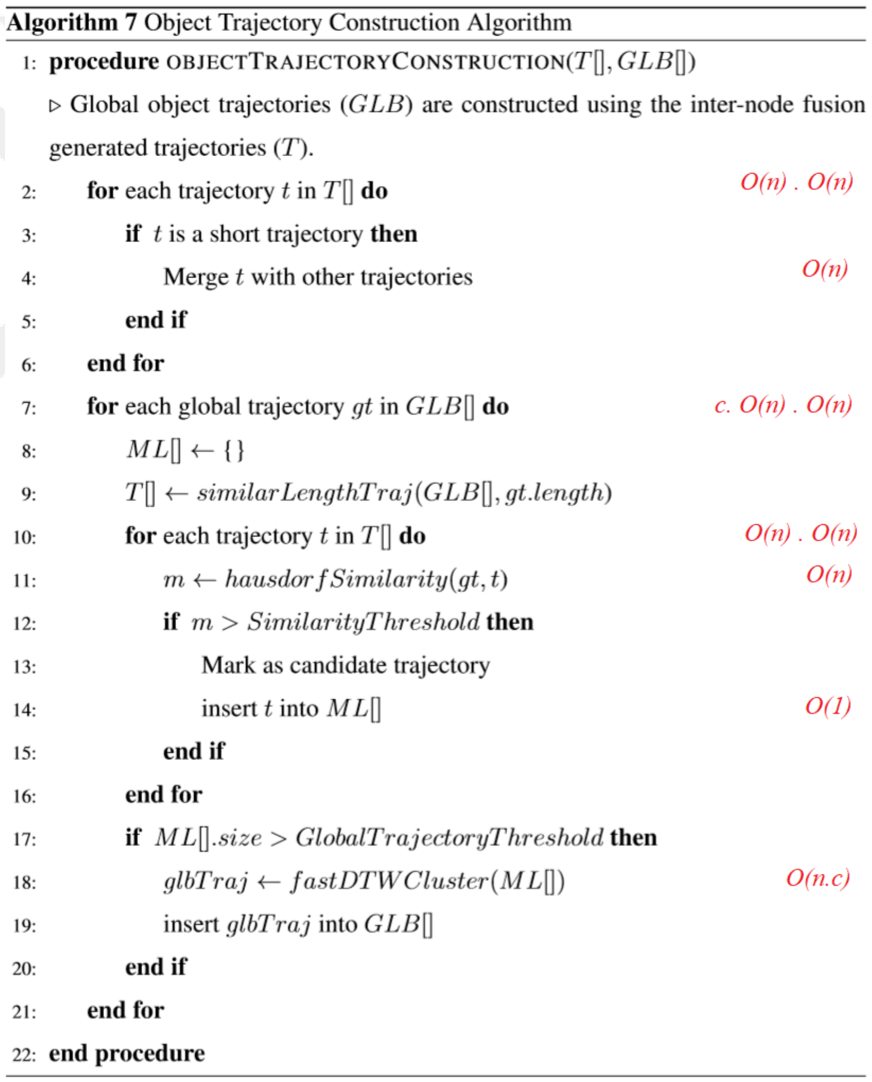
```
1:  procedure INTERNODEFUSION($n, s, \epsilon$)
2:      $N[] \leftarrow findNeighbors(s)$                                   O(n)
3:      $\theta \leftarrow findDirection(L_s, L_n)$                               O(1)
4:      for neighbor node $sn$ in $N[]$ do                            c . O(n) . O(n)
5:          $\theta_s \leftarrow findDirection(L_s, L_{sn})$
6:          if $\theta_s$ in the same direction with $\theta$ then
7:              $T_s[] \leftarrow$ Trajectories of sensor node $s$                 O(1)
8:              $T_{sn}[] \leftarrow$ Trajectories of sensor node $sn$               O(1)
9:              for each trajectory $t_s$ in $T_s[]$ and $t_{sn}$ in $T_{sn}[]$ do     O(n) . O(n)
10:                 $M_t \leftarrow$ Merge trajectories $t_s$, $t_{sn}$                 O(n)
11:                 $p \leftarrow \emptyset$
12:                 $c \leftarrow$ Last sensed item in the trajectory $M_t$
13:                 $same \leftarrow true$
14:                 for the last $\epsilon$ items do                          O(n)
15:                     $p \leftarrow$ Previous item of $c$ in the trajectory $M_t$
16:                     if $p$ is $null$ then
17:                         exit loop
18:                     end if
19:                     $score \leftarrow ruleEngine(p, c)$                       O(n)
20:                     if $score < \lambda$ then
21:                         $same \leftarrow false$
22:                         exit loop
23:                     end if
24:                     $c \leftarrow p$
25:                 end for
26:                 if $same = true$ then
27:                     Apply merged trajectory $M_t$ in $T_s[]$ and $T_{sn}[]$        O(n)
28:                 end if
29:             end for
30:         end if
31:     end for
32: end procedure
```

Figure B.3: Complexity analysis of Inter-Node Fusion algorithm.

B.4 Object Trajectory Construction Algorithm

Object Trajectory Construction algorithm iterates over the inter-node trajectories to find the matching global trajectories.

As shown in the below figure that the 7th line iterates over the existing global trajectories, which is a constant number of global trajectories ($1 <= c <= 10$), to compare with the candidate trajectories by using the Hausdorff Similarity algorithm whose complexity is linear, the complexity of this algorithm is $O(c.n^2)$, which is still $O(n^2)$.



Algorithm 7 Object Trajectory Construction Algorithm

```
1:  procedure OBJECTTRAJECTORYCONSTRUCTION($T[], GLB[]$)
        ▷ Global object trajectories ($GLB$) are constructed using the inter-node fusion
        generated trajectories ($T$).
2:      for each trajectory $t$ in $T[]$ do                              O(n) . O(n)
3:          if $t$ is a short trajectory then
4:              Merge $t$ with other trajectories                        O(n)
5:          end if
6:      end for
7:      for each global trajectory $gt$ in $GLB[]$ do                    c. O(n) . O(n)
8:          $ML[] \leftarrow \{\}$
9:          $T[] \leftarrow similarLengthTraj(GLB[], gt.length)$
10:         for each trajectory $t$ in $T[]$ do                          O(n) . O(n)
11:             $m \leftarrow hausdorfSimilarity(gt, t)$                 O(n)
12:             if $m > SimilarityThreshold$ then
13:                 Mark as candidate trajectory
14:                 insert $t$ into $ML[]$                               O(1)
15:             end if
16:         end for
17:         if $ML[].size > GlobalTrajectoryThreshold$ then
18:             $glbTraj \leftarrow fastDTWCluster(ML[])$                O(n.c)
19:             insert $glbTraj$ into $GLB[]$
20:         end if
21:     end for
22: end procedure
```

Figure B.4: Complexity analysis of Object Trajectory Construction algorithm.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Küçükkeçeci, Cihan
Nationality: Turkish
Date and Place of Birth: 1983, Konya
Marital Status: Married
Phone: +905326163114

EDUCATION

Degree	**Institution**	**Year of Graduation**
M.Sc.	Computer Engineering, Bilkent University	2007
B.Sc.	Computer Engineering, Hacettepe University	2005

PROFESSIONAL EXPERIENCE

Year	**Place**	**Enrollment**
2019-Present	Realdolmen NV	Integration Architect/Technical Lead
2017-2019	Luciad NV	Senior Software Engineer
2015-2017	Ayesaş	Project Manager/Senior Software Expert Engineer
2012-2015	ATOS	Software Team Lead/Senior Software Engineer
2009-2012	Milsoft	Lead Software Engineer
2008-2009	Turkish Army	Computer Engineer (Military Service)
2005-2007	Bilkent University	Research Assistant/Software Engineer

PUBLICATIONS

International Journal Publications

C. Küçükkeçeci and A. Yazici, “Multilevel Object Tracking in Wireless Multimedia Sensor Networks for Surveillance Applications Using Graph-based Big Data.“ IEEE Access, 2019.

C. Küçükkeçeci and A. Yazıcı, “Big data model simulation on a graph database for surveillance in wireless multimedia sensor networks,” Big Data Research, vol. 11, pp. 33 – 43, 2018.

International Conference Publications

C. Küçükkeçeci and A. Yazici, “A graph-based big data model for wireless multimedia sensor networks,“ in Advances in Big Data - Proceedings of the 2nd INNS Conference on Big Data, October 23-25, 2016, Thessaloniki, Greece, pp. 205–215.