8-20-2019 10:00 AM

# Data Analytics and Performance Enhancement in Edge-Cloud Collaborative Internet of Things Systems

Tianqi Yu, *The University of Western Ontario*

Supervisor: Wang, Xianbin, *The University of Western Ontario*
Co-Supervisor: Shami, Abdallah, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering

## Recommended Citation

Yu, Tianqi, "Data Analytics and Performance Enhancement in Edge-Cloud Collaborative Internet of Things Systems" (2019). *Electronic Thesis and Dissertation Repository*. 6466.
https://ir.lib.uwo.ca/etd/6466

# Abstract

Based on the evolving communications, computing and embedded systems technologies, Internet of Things (IoT) systems can interconnect not only physical users and devices but also virtual services and objects, which have already been applied to many different application scenarios, such as smart home, smart healthcare, and intelligent transportation. With the rapid development, the number of involving devices increases tremendously. The huge number of devices and correspondingly generated data bring critical challenges to the IoT systems. To enhance the overall performance, this thesis aims to address the related technical issues on IoT data processing and physical topology discovery of the subnets self-organized by IoT devices.

First of all, the issues on outlier detection and data aggregation are addressed through the development of recursive principal component analysis (R-PCA) based data analysis framework. The framework is developed in a cluster-based structure to fully exploit the spatial correlation of IoT data. Specifically, the sensing devices are gathered into clusters based on spatial data correlation. Edge devices are assigned to the clusters for the R-PCA based outlier detection and data aggregation. The outlier-free and aggregated data are forwarded to the remote cloud server for data reconstruction and storage. Moreover, a data reduction scheme is further proposed to relieve the burden on the trunk link for data uploading by utilizing the temporal data correlation. Kalman filters (KFs) with identical parameters are maintained at the edge and cloud for data prediction. The amount of data uploading is reduced by using the data predicted by the KF in the cloud instead of uploading all the practically measured data.

Furthermore, an unmanned aerial vehicle (UAV) assisted IoT system is particularly designed for large-scale monitoring. Wireless sensor nodes are flexibly deployed for environmental sensing and self-organized into wireless sensor networks (WSNs). A physical topology discovery scheme is proposed to construct the physical topology of WSNs in the cloud server to facilitate performance optimization, where the physical topology indicates both the logical connectivity statuses of WSNs and the physical locations of WSN nodes. The physical topology discovery scheme is implemented through the newly developed parallel Metropolis-Hastings random walk based information sampling and network-wide 3D localization algorithms, where UAVs are served as the mobile edge devices and anchor nodes. Based on the physical topology constructed in the cloud, a UAV-enabled spatial data sampling scheme is further proposed to efficiently sample data from the monitoring area by using denoising autoencoder (DAE). By deploying the encoder of DAE at the UAV and decoder in the cloud, the data can be partially sampled from the sensing field and accurately reconstructed in the cloud.

In the final part of the thesis, a novel autoencoder (AE) neural network based data outlier detection algorithm is proposed, where both encoder and decoder of AE are deployed at the edge devices. Data outliers can be accurately detected by the large fluctuations in the squared error generated by the data passing through the encoder and decoder of the AE.

**Keywords:** data processing, topology discovery, machine learning, edge-cloud collaborative computing, Internet of Things systems

# Lay Summary

Based on the evolving communications, computing and embedded systems technologies, the Internet of Things (IoT) can interconnect not only physical users and devices but also virtual services and objects, which have already been pervasively deployed. With the rapid development, the number of involving devices increases tremendously. The huge number of devices and generated data bring critical challenges. To enhance the overall performance, this thesis aims to address the related issues on IoT data processing and physical topology discovery of the subnets self-organized by IoT devices.

Firstly, the issues on outlier detection and data aggregation are addressed through the development of recursive principal component analysis based data analysis framework. The framework is developed in a cluster-based structure to fully exploit the spatial data correlation. Moreover, a temporal data correlation based reduction scheme is further proposed to reduce the amount of data uploading, which is implemented by using the data predicted by the Kalman filters in the cloud instead of uploading all the practically measured data.

Furthermore, an unmanned aerial vehicle (UAV) assisted IoT system is designed for large-scale monitoring, where UAVs are served as the mobile edge devices. Specifically, wireless sensor nodes are flexibly deployed for environmental sensing and self-organized into wireless sensor networks (WSNs). The physical topology of WSNs unveils the logical connectivity statuses of WSNs and the physical locations of nodes, which can facilitate system performance optimization. Thus, a physical topology discovery scheme is proposed to construct the physical topology in the cloud. Moreover, a UAV-enabled spatial data sampling scheme is further proposed to efficiently sample data from the monitoring area by using denoising autoencoder (DAE). By deploying the encoder of DAE at the UAV and decoder in the cloud, the data can be partially sampled from the area and accurately reconstructed in the cloud.

In the final part, a novel autoencoder based data outlier detection algorithm is proposed, where both encoder and decoder of autoencoder are deployed at the edge devices. Data outliers can be accurately detected by the large fluctuations in the squared error generated by the data passing through the encoder and decoder.

To my parents

# Acknowledgments

I would like to express my deepest appreciation to my supervisor, Dr. Xianbin Wang, for all his guidance, patience and support. It was his enlightening supervisions that inspired me to explore novel research areas and broadened my views in the research area. It was also his guidance and encouragement that helped me get prepared for my future career with all the professional skills. It was a wonderful and rewarding journey to learn from him.

I also feel grateful to my co-supervisor, Dr. Abdallah Shami. Thanks to his professional insights and technical guidance, I was able to achieve exiting research findings and implement my ideas into practice. It was my honor to work with him.

Sincere thanks to Dr. Rachid Benlamri, Dr. Kostas Kontogiannis, Dr. Anestis Dounavis and Dr. Jagath Samarabandu for being my examination committee. I highly appreciate their precious time and constructive suggestions on my thesis and research.

Additionally, I would like to thank every course instructor and administrative staff that I met at The University of Western Ontario. I cannot complete this degree without their assistance and kindness during these four years.

I would also like to express my thanks and gratitude to my research group colleagues, who helped me a lot in both work and daily life just like my brothers and sisters. I was so lucky to meet such a big and warm research group. I would also like to extend my thanks to all my friends, who gave me strong support whenever and wherever I needed help.

As always, I feel so grateful to my parents and my family. I highly appreciate their love and support throughout not only this degree but also my life. They are always there for me and always back me up.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **IoT** | Internet of Things |
| **VANET** | vehicular ad hoc networks |
| **WSN** | wireless sensor network |
| **UAV** | unmanned aerial vehicle |
| **PCA** | principal component analysis |
| **PC** | principal component |
| **SPE** | squared prediction error |
| **DAE** | denoising autoencoder |
| **KF** | Kalman filter |
| **MHRW** | Metropolis-Hastings random walk |
| **MDS** | multidimensional scaling |
| **GPS** | global positioning system |
| **D2D** | device-to-device |
| **LTE** | long term evolution |
| **Wi-Fi** | wireless fidelity |
| **RF** | radio frequency |
| **LDA** | latent Dirichlet allocation |
| **OSA** | obstructive sleep apnea |
| **TIHM** | technology integrated healthcare management |
| **AMI** | advanced metering infrastructure |
| **SIoT** | social Internet of Things |
| **OSN** | online social network |
| **PO** | physical object |
| **AO** | abstract object |
| **VO** | virtual object |
| **CVO** | composite virtual object |
| **ISMA** | intelligent sensing model for anomaly detection |
| **SVM-RBF** | support vector machine-radial basis function |
| **IIoT** | industrial Internet of Things |
| **M2M** | machine-to-machine |
| **CS** | compressive sensing |
| **DFD** | distributed fault detection |
| **MVN** | multivariate normal |
| **CDF** | cumulative distribution function |
| **MSE** | mean squared error |
| **GM** | grey model |
| **ARMA** | autoregressive and moving average |
| **SVD** | singular value decomposition |
| **RSS** | received signal strength |

# List of Abbreviations (Cont'd)

| | |
|---|---|
| **LOS** | line-of-sight |
| **FOM** | free-space outdoor model |
| **AE** | autoencoder |
| **RSSI** | received signal strength indicator |
| **LQI** | link quality indicator |
| **IRLS** | iterative reweighted least squares |
| **GD** | gradient descent |
| **AI** | artificial intelligence |
| **QoS** | quality of service |

# Chapter 1

# Introduction

## 1.1 Overview of Internet of Things Systems

By utilizing the rapidly developing communications, computing, and embedded systems technologies, Internet of Things (IoT) systems are able to interconnect not only physical users and devices, but also virtual objects and services, which can finally change the way of living in many different kinds of scenarios, such as smart city, smart healthcare, and Industry 4.0 [1]. Due to the pervasive deployment and enlarging scale of IoT systems, the number of involving devices keeps increasing in an explosive trend, which expects to reach 18 billion in 2022 [2]. The tremendous increment in the number of devices brings huge challenges to the performance of IoT systems. Before the detailed investigations on the challenges, the fundamentals of IoT systems are firstly introduced with the conceptual system architecture, which provides a blueprint of the whole system.

The ultimate goal of IoT systems is to make timely and reliable decisions and provide customized services by fully utilizing the information collected from objects and environments. In order to achieve the goal, a paradigm of IoT systems should at least comprise the following components, *i.e.*, sensing, communication, and analytics layers, as shown in Fig.1.1.

- **Sensing layer** composed of IoT end devices is the most fundamental component in the IoT systems, which is responsible for sensing and collecting the environmental information, and also reacting to the feedback and instructions. These IoT end devices are

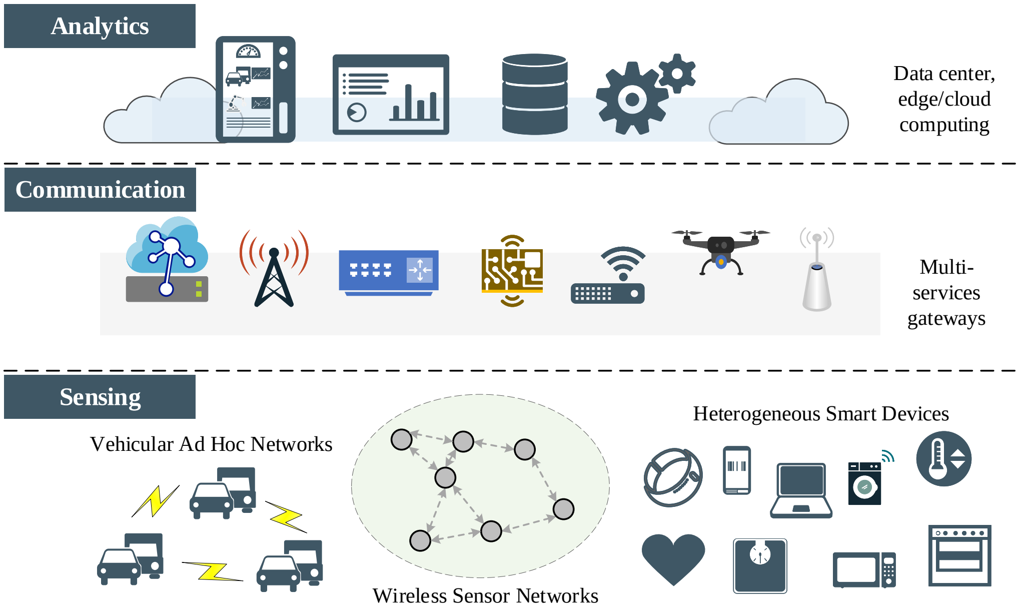

Analytics
Data center, edge/cloud computing
Communication
Multi-services gateways
Sensing
Vehicular Ad Hoc Networks
Wireless Sensor Networks
Heterogeneous Smart Devices

Figure 1.1: Diagram of IoT systems includes sensing, communications, and analytics layers.

heterogeneous with different capabilities of computing, communications, and storage, which can be self-organized into several subnets, such as vehicular ad hoc networks (VANETs) and wireless sensor networks (WSNs). Therefore, it is a tough task to effectively coordinate and manage the massive amount of heterogeneous IoT end devices and subnets in the sensing layer.

- **Communication layer** supported by multi-services gateways is the backbone network for IoT data communications, including the uplink for the uploading of sensing data and the downlink for the delivery of feedback. Due to the heterogeneous feature, multiple kinds of communication protocols may coexist in the IoT systems. Thus, multi-services gateways are needed to facilitate the data communications throughout the systems, such as the femto base stations, wireless access points, and mobile gateways (*e.g.*, unmanned aerial vehicle (UAV)).

- **Analytics layer** is responsible for IoT data processing and analysis. In IoT systems, data analytics can be flexibly executed. IoT data can be locally processed at the IoT end devices, though the IoT end devices are with limited resources and can only provide elementary processing. IoT data can also be uploaded to the remote data center, *e.g.*, cloud computing platform, for comprehensive processing and analysis, while processing and communications from the remote platform can lead to high latency. Thus, edge computing has been introduced into IoT systems as a compromise, which is more close to the IoT end devices and can provide real-time responses to the local devices. Meanwhile, edge computing can provide preliminary data processing, so that the tasks of cloud computing can be partially offloaded to the edge and the burden of data uploading on the trunk link can be relieved. Thus, collaborative computing is a promising solution to the computation-intensive tasks in the IoT systems, which needs to be seriously analyzed and designed.

## 1.2 Challenges in the Explosive Growth of IoT Systems

The pervasive deployment and increasing system scale boost the explosive growth of IoT systems, which brings huge technical challenges to the IoT systems alongside the enormous amount of involving devices. In order to enhance the system performance on real-time, reliability and scalability, some of the existing and potential technical challenges in the IoT systems are unveiled as follows.

- **Interoperability among massive and heterogeneous devices:** IoT systems typically comprise a huge number of devices with different capabilities of communications, computing, and storage. Without effective interoperability among these devices, the reliability and scalability of IoT systems would be degraded dramatically. The specific aspects of interoperability in the IoT systems include the communications and coordination among the massive amount of end devices in the sensing layer, the interactions between the sensing layer and the intermediate multi-services gateways, the communications and cooperation among the gateways, and the interactions between the gateways and the remote system data and control center. Therefore, multiple communication protocols and coordination mechanisms need to be customized for the IoT systems to guarantee the interoperability among the massive and heterogeneous devices.

- **Autonomous organization and management:** IoT end devices can be self-organized into subnets, such as VANETs and WSNs, which outstandingly improve the flexibility and scalability of IoT systems. However, due to the random and scalable features, the information of self-organized subnets, *e.g.*, network topology, can hardly be known in the system control center in advance. Furthermore, the dynamic and resource-constrained IoT end devices tend to change the subnets in unpredictable ways. Thus, unbiased information sampling schemes are needed to collect the information of the devices and subnets, so that the system control center can have a better knowledge of the entire system for device management, event management, and system performance optimization.

- **IoT data processing and analysis:** The huge number of IoT end devices continuously generate a massive amount of IoT data, which challenges the IoT systems on timely and reliable data processing and analysis. Providing the weak capabilities of some IoT end devices, the IoT data has to be uploaded to the remote data center, *e.g.*, cloud computing platform, for comprehensive data analytics and storage. However, the overwhelming amount of data uploading imposes a heavy burden on the trunk link, which may even result in system crashes. Furthermore, given the complex and dynamic environmental situations of the deployment fields, IoT end devices are vulnerable to different kinds of attacks and inner malfunctions, which can finally taint the IoT data. The abnormal IoT data can lead the data-driven IoT systems into unsafe conditions. Therefore, it is necessary to develop appropriate system architecture and corresponding algorithms for data processing and analysis in the IoT systems.

- **Collaborative computing:** In order to reduce system investment and operating costs, certain IoT devices are built with limited resources. Thus, resource-constrained devices can hardly be used to complete computation-intensive tasks. Collaborative computing is a promising solution, which can finally provide timely and reliable services to users and devices by optimally utilizing the distributed system resources in IoT systems. Specifically, the implementation of collaborative computing in IoT systems relies on the resource awareness of individual devices, optimal resource allocation, and task offloading. Each of the technical fundamentals needs to be seriously investigated.

- **Security and privacy protection:** The resource-constrained IoT devices are vulnerable to different kinds of attacks, due to the transparent wireless communication interfaces and lack of protection mechanisms. The malicious attacks can easily occur during the procedure of data communications, which may either steal the IoT data by eavesdropping or mislead the IoT systems by spoofing, tampering or dropping the IoT data. Moreover, the privacy of users would also be exposed to the adversaries through the compromised devices. Considering the pervasive deployment of IoT systems, malicious attacks can bring huge security and privacy threats to the industry, environment, and society. However, the existing security and privacy protection mechanisms are too complex to be applied to the IoT systems directly. Therefore, lightweight and distributed security and privacy protection mechanisms need to be tailored for the IoT systems to protect data confidentiality and user privacy.

## 1.3 Research Objectives of the Thesis

Considering the challenges mentioned above, technical issues on IoT data processing and topology management of the self-organized subnets would be addressed in the thesis. The specific research objectives are identified as follows.

- **Design of IoT system architecture:** The general requirements of IoT system architecture design are dynamic, flexible, and scalable, due to the dynamic and heterogeneous features of the huge number of IoT end devices [3]. Beyond the general demands, the specific needs of IoT data processing and topology management should be involved in the design as well, since the IoT system architecture has deterministic effects on the dataflow of IoT data processing and also the efficiency of device coordination and management. The existing conceptual architectures of IoT systems include the cloud-based architecture and edge-cloud collaborative architecture, while the latter is more appropriate for the IoT systems with requirements of large-scale and real-time analytics. Although the conceptual architecture exists, the functionalities of each system components and the collaborations among them still need to be seriously considered and carefully designed for specific applications.

- **Development of IoT data processing algorithms:** Due to the explosive increment in the number of heterogeneous IoT end devices, data processing in IoT systems meets the challenges of high volume and taint. Therefore, data processing algorithms, particularly for data aggregation and data outlier detection, need to be developed in order to reduce the amount of data uploading and clean the tainted data. Due to the context-aware capabilities of IoT systems and mild change of physical environments, IoT data are generally labeled with timestamp and location and are temporally and spatially correlated, which can be exploited to develop the data processing algorithms.

- **Development of topology management schemes:** Due to the randomly deployed and self-organized features of IoT end devices, particularly, wireless sensor nodes, it is difficult to control the physical topology of the subnets in IoT systems. However, awareness of physical topology is important, since physical topology indicates both the physical locations and connection statuses of IoT end devices, which can be utilized to facilitate the IoT systems with performance optimization such as device management and real-time event detection. Thus, the development of topology management schemes is a necessity. Since IoT end devices are context-aware, unbiased sampling of the device information can be exploited to construct the physical topology at the system control center.

## 1.4 Technical Contributions of the Thesis

The main contributions of this thesis are summarized as follows.

- In order to aggregate the redundant data and detect the outliers in IoT systems, a cluster-based data analysis framework is proposed using recursive principal component analysis (R-PCA). More specifically, at a cluster head, spatially correlated sensor data collected from cluster members are aggregated by extracting the principal components (PCs). The data outliers are identified by the abnormal squared prediction error (SPE) score, which is defined as the square of residual value after extraction of PCs. With R-PCA, the parameters of the PCA model can be recursively updated to adapt to the changes in IoT systems. The cluster-based data analysis framework also releases the computational and processing burdens on sensor nodes.

- Denoising autoencoder (DAE) neural network is an extension of PCA on nonlinear data correlation. By using DAE, a UAV enabled spatial data sampling scheme is proposed for large-scale IoT systems. More specifically, a UAV-enabled edge-cloud collaborative IoT system architecture is firstly developed for data processing in large-scale IoT monitoring systems, where the UAV is utilized as a mobile edge computing device. Based on the system architecture, the UAV-enabled spatial data sampling scheme is further proposed, where wireless sensor nodes of the large-scale IoT systems are clustered by a newly developed bounded-size K-means clustering algorithm. A neural network model, *i.e.*, DAE, is applied to each cluster for data sampling and reconstruction by exploiting either linear or nonlinear spatial correlation among data samples. Taking advantage of the DAE neural network model, the accuracy and efficiency of spatial data sampling are dramatically improved. Furthermore, similar to PCA, the squared error generated by passing through the autoencoder (AE) neural network can also be used to identify the data outliers.

- The R-PCA and DAE based algorithms are proposed based on the spatial data correlation, while an edge computing enabled temporal IoT data reduction scheme is further proposed by the exploitation of temporal data correlation. More specifically, IoT data are firstly modeled as multivariate normal distribution in the cloud. Dual Kalman filters (KF) with identical parameters are then deployed at both the cloud and edge platforms. The same predictions are simultaneously triggered by the dual KFs at both platforms. Only the measured IoT data out of the predicted range is further uploaded from edge to cloud. Otherwise, predicted values at both platforms are used instead of measurements. By using this approach, the amount of data uploading is reduced so that the burden on the bandwidth of the trunk link is relieved.

- In order to build up the physical topology in the cloud, a cloud-orchestrated physical topology discovery scheme is proposed for the large-scale IoT systems by using UAV. More specifically, the large-scale monitoring area is firstly split into several subregions for UAV-enabled data collection. Within the subregions, parallel Metropolis-Hastings random walk (MHRW) is developed to gather the information of nodes, including their

IDs and neighbor tables. The collected information is forwarded to the cloud through UAV for the initial construction of logical topology. After that, a network-wide 3D localization algorithm is further developed based on the logical topology and multidimensional scaling method, termed as Topo-MDS, where the UAV equipped with a global positioning system (GPS) chipset is served as a mobile anchor to locate the sensor nodes. The physical topology can be successfully formed up by using the proposed scheme. Based on the physical topology constructed in the cloud, the target areas can be timely located when abnormal events occur.

## 1.5 Thesis Outline

The remainder of this thesis is organized as follows.

A comprehensive study of data analytics in IoT systems is conducted in Chapter 2. The fundamentals of IoT data analytics are firstly elucidated, which comprises IoT data characteristics, IoT data challenges and taxonomy of IoT data analytics. Afterwards, the system architectures that can support effective and efficient data analytics in IoT systems are analyzed, including the cloud-based architecture and edge-cloud collaborative architecture. Finally, the existing applications such as smart city and smart healthcare are investigated from the perspectives of system design and shortcomings of performance.

In Chapter 3, a cluster-based data analysis framework is proposed using R-PCA, which can aggregate the redundant data and detect the data outliers simultaneously. More specifically, at a cluster head, sensor data collected from cluster members are highly correlated in the spatial domain and thus aggregated by extracting the PCs, and potential data outliers are identified by the abnormal SPE score, which is defined as the square of residual value after extraction of PCs. With R-PCA, the parameters of the PCA model can be recursively updated to adapt to the changes in IoT systems. The cluster-based data analysis framework also relieves the computational and processing burdens on sensor nodes. Practical databases based simulations have indicated that the proposed framework efficiently aggregates the correlated sensor data with high recovery accuracy. The data outlier detection accuracy is also improved by the proposed method compared to other existing algorithms.

In Chapter 4, a temporal IoT data reduction scheme empowered by edge computing is proposed to reduce the amount of data uploaded to the cloud. More specifically, IoT data are firstly modeled as multivariate normal distribution in the cloud. Dual KF with identical parameters are then deployed at both cloud and edge ends. The same predictions are triggered by the dual KFs at both ends simultaneously. Only the measurements out of the predicted ranges are further uploaded from edge to cloud. Otherwise, predicted values at both ends are used instead of measurements. A simple IoT system prototype has been developed for performance evaluation. Experimental results indicate that the proposed scheme significantly reduces the number of packets uploaded to the cloud platform while ensures the data accuracy.

In Chapter 5, a cloud-orchestrated physical topology discovery scheme for large-scale IoT systems using UAVs is proposed, in order to build up the physical topology in the cloud. More specifically, first of all, the large-scale monitoring area is split into subregions for UAV-enabled data collection. Within the subregions, parallel MHRW is developed to gather the information of nodes, including their IDs and neighbor tables. The collected information is then forwarded to the cloud through UAV for the initial construction of logical topology. After that, a network-wide 3D localization algorithm is further developed based on the logical topology and multidimensional scaling method, termed as Topo-MDS, where the UAV equipped with a GPS chipset is served as a mobile anchor to locate the nodes. Simulation results indicate that the parallel MHRW improves both the efficiency and accuracy of logical topology construction. Besides, the Topo-MDS algorithm dramatically improves the 3D localization accuracy, as compared to the existing algorithms in the literature.

In Chapter 6, a UAV enabled spatial data sampling scheme is proposed using DAE neural network. More specifically, a UAV-enabled edge-cloud collaborative IoT system architecture is firstly developed for data processing in large-scale IoT monitoring systems, where the UAV is utilized as a mobile edge computing device. Based on the system architecture, the UAV-enabled spatial data sampling scheme is further proposed, where wireless sensor nodes of the large-scale IoT systems are clustered by a newly developed bounded-size K-means clustering algorithm. A neural network model, *i.e.*, DAE, is applied to each cluster for data sampling and reconstruction by the exploitation of either linear or nonlinear spatial correlation among data samples. Simulations have been conducted and the results indicate that the proposed scheme has improved the data reconstruction accuracy under the same sampling ratio without introducing extra complexity, as compared to the compressive sensing based method.

Based on the system architecture and the dataflow proposed in Chapter 6, an AE neural network based data outlier detection algorithm is further developed in Chapter 7. By using AE, the spatial correlation of data can be fully utilized to improve the data outlier detection accuracy. Performance evaluation has been conducted based on the oceanic atmospheric data. Simulation results indicate that the developed scheme can accurately detect the data outliers.

Finally, all the contributions are summarized in Chapter 8, where the future research directions are identified as well.

# Chapter 2

# Data Analytics in IoT Systems

With the pervasive deployment of IoT technology, the number of connected IoT end devices increases in an explosive trend, which continuously generates a massive amount of data. Timely data analytics can provide useful information for decision making in the IoT systems, which is able to enhance both the system efficiency and reliability. More specifically, data analytics in IoT systems is utilized to effectively and efficiently process the discrete IoT data series and provide services such as data classification, pattern analysis, and tendency prediction. However, the continuous generation of data from heterogeneous devices brings huge technical challenges to IoT data analytics. Thus, how to timely and fully process and analyze the massive and heterogeneous IoT data needs to be seriously considered in the design of IoT systems. This chapter provides a comprehensive study of data analytics in IoT systems. A fundamental introduction to data analytics in IoT systems is firstly elucidated, including the characteristics of IoT data, the challenges of IoT data, and the taxonomy of IoT data analytics. IoT system architectures suitable for data analytics are thoroughly analyzed then. Finally, a comprehensive survey on the existing applications of data analytics in IoT systems is conducted from the perspectives of system design and shortcomings of performance.

## 2.1 Introduction to Data Analytics in IoT Systems

With the rapid development of communications and embedded systems technologies, IoT systems have been pervasively deployed in different kinds of application scenarios. The number

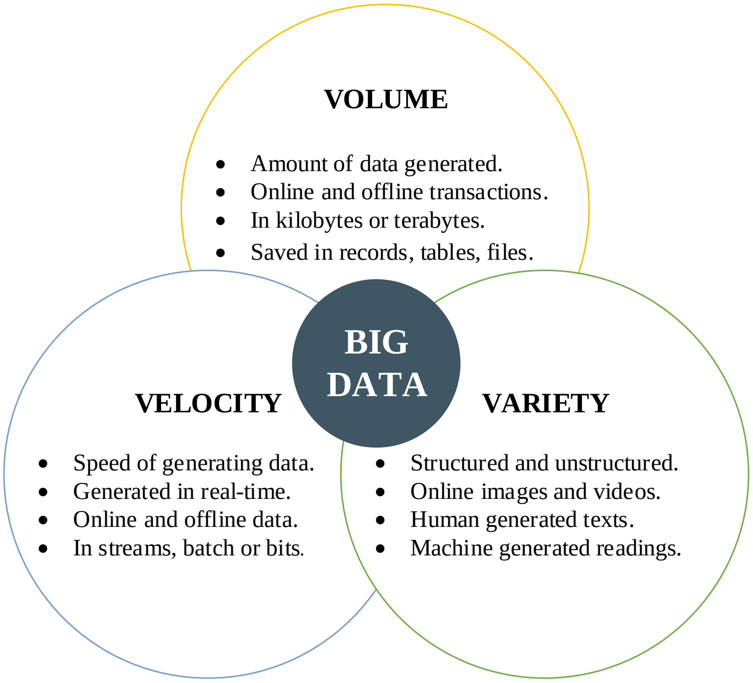

VOLUME
Amount of data generated.
Online and offline transactions.
In kilobytes or terabytes.
Saved in records, tables, files.
BIG DATA
VELOCITY
Speed of generating data.
Generated in real-time.
Online and offline data.
In streams, batch or bits.
VARIETY
Structured and unstructured.
Online images and videos.
Human generated texts.
Machine generated readings.

Figure 2.1: Three Vs of big data: volume, velocity, and variety.

of involving IoT end devices keeps increasing in an explosive trend. These devices directly interact with the real world and continuously generate a massive amount of data, which brings huge challenges to the data analytics in IoT systems, particularly the data analytics with a critical requirement of completion time. Thus, data analytics needs to be seriously considered in the IoT systems. In this section, data analytics in IoT systems is analyzed from the perspectives of IoT data characteristics, IoT data challenges and taxonomy of IoT data analytics.

### 2.1.1 IoT Data Characteristics

With the tremendous increment in the number of IoT end devices, a massive amount of IoT data are generated as a consequence. However, due to the unique characteristics of IoT data, data analytics in IoT systems is not identical to the conventional big data analytics. Thus, the characteristics of IoT data are firstly identified in this subsection.

The renowned properties of big data are the three Vs, namely, volume, velocity, and variety, as depicted in Fig.2.1 [4]. Though they have three Vs in common, IoT data still have several aspects different from the conventional big data [5]. The unique characteristics of IoT data are listed as follows [6].

- *Large scale:* With the pervasive deployments of large-scale IoT systems, a large number of IoT end devices are involved in the systems and continuously generate a massive amount of data. In most of IoT systems, not only the real-time data but also the historical data are needed to provide the descriptions of user patterns, environmental trends, *etc.* Thus, both the real-time and historical data have to be processed, analyzed and stored in the IoT systems, which finally labels the characteristic of large scale to IoT data.

- *Heterogeneity:* The sensing layer of an IoT system as shown in Fig.1.1 is in high diversity, which comprises heterogeneous devices and subnets. Different from the traditional homogeneous wireless networks, data generated by the heterogeneous IoT devices are not identical in formats and even unstructured, which finally results in heterogeneity.

- *Temporal and spatial correlation:* IoT data are generally labeled with both location information and timestamp, as most of the IoT systems are context-aware. The labeled IoT data are highly correlated in temporal and spatial domains because the environmental parameters sensed and sampled by the IoT end devices are varied in mild trends. Providing the statistical characteristic of temporal and spatial correlation, IoT data can be easily processed with the statistical tools and the machine learning methods.

- *Taint:* Due to the low-cost feature of IoT end devices, these tiny devices are vulnerable to different kinds of attacks and also inner malfunctions, which can finally lead to abnormal IoT data. Therefore, data pre-processing, particularly data cleaning, is generally needed before eventually performing data analysis.

### 2.1.2 IoT Data Challenges

Providing the unique characteristics of IoT data, the technical challenges on data collection, data analytics and data usage that IoT systems can meet are stated as follows.

- *Data collection:* In the large-scale IoT systems, "things", namely, the connected IoT end devices are the most fundamental components. So the first challenge comes with data collection from the massive amount of heterogeneous end devices. It has to be

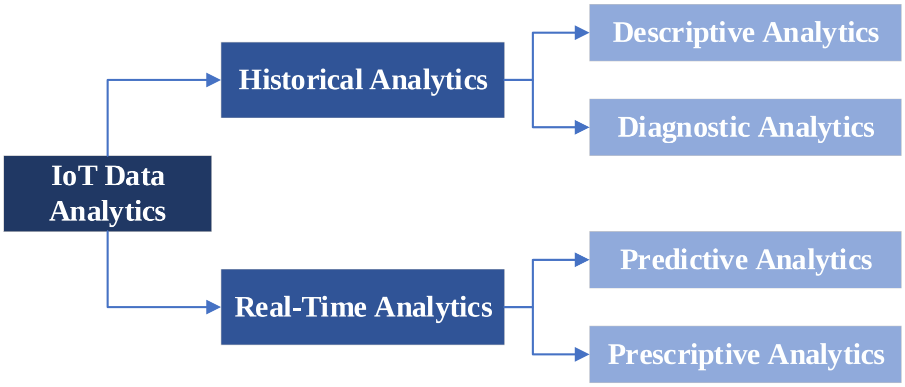


Figure 2.2: Taxonomy of data analytics in IoT systems.

recognized who, where, when and why generating these data. Besides, during the stage of data collection, data accuracy, integration and security need to be ensured as well.

- *Data analytics:* The collected IoT data are then uploaded to the cloud platform through edge devices. During this stage, it is necessary to seriously consider how to store the data with different structures and formats, and how to process and analyze the data with appropriate tools.
- *Data usage:* Since most of the IoT applications are data-driven, how to manage the ownership of data, how to legally share data with others and how to provide efficient and useful feedback to the actuators need to be considered in the system design as well.

### 2.1.3 Taxonomy of IoT Data Analytics

*Analytics* refers to “the scientific process of transforming data into insights for the purpose of making better decisions” [7]. In terms of IoT data analytics, it is the computational process of transforming the IoT data collected from the heterogeneous IoT end devices into insights through data processing and analysis, for decision making in the IoT systems. The history of IoT data analytics is as long as the emergence of IoT systems. Therefore, several efforts have been spared on the processing and analysis of IoT data. According to the different requirements of dataset and completion time, data analytics in IoT systems can be classified into historical analytics and real-time analytics as shown in Fig.2.2. Moreover, in conventional big data analytics, considering the different processing stages, the analytics can be categorized into four types, namely, descriptive analytics, diagnostic analytics, predictive analytics, and prescriptive analytics [8]. These four types are also integrated into the taxonomy diagram of IoT data analytics, according to the type of dataset usage, the requirement of processing time and the stage of the processing procedure.

**Historical analytics** is based on the IoT data that have been collected and stored in the database for a certain while, which can be further classified into descriptive analytics and diagnostic analytics. Descriptive analytics is the fundamental of IoT data processing, which uncovers the patterns behind the raw data. Diagnostic analytics is used to discover the reasons behind certain patterns.

- *Descriptive analytics:* Descriptive analytics is the process of transforming raw data collected from multiple data sources into useful information, which describes the past. For example, a clinic records the number of patients that were hospitalized last month. However, the findings of the descriptive analytics simply describe the fact, without inferring the reasons behind. Therefore, descriptive analytics only can hardly support the highly data-driven application scenarios of the IoT systems, where other types of data analytics are still needed.

- *Diagnostic analytics:* At the stage of diagnostic analytics, historical data from multiple data sources are jointly analyzed with the diagnostic tools to find out the reasons behind the facts provided by descriptive analytics. By exploitation of diagnostic analytics, it is possible to identify the hiding data patterns and underlying relations among data, which can provide in-depth insights into a particular problem. In the meantime, IoT systems should have detailed information at their disposal, otherwise, data collection may turn out to be individual for every issue and time-consuming.

**Real-time analytics** in IoT systems focuses on the design of IoT system architecture that must complete the data analytics and return responses within a certain time frame, which is known as the deadline. According to the requirements of different applications, the deadline could range from nanosecond in computer network communications to millisecond in medical diagnosis. Missing the deadline will violate the system requirements, while completion of a task much earlier than the deadline may also deteriorate the system performance. Fast responses and precise timing control are typical features of real-time analytics.

Real-time analytics can be further classified into predictive analytics and prescriptive analytics, which have more critical requirements on real-time responses as compared to descriptive and diagnostic analytics. Besides, instead of the historical data stored in the database, predictive and prescriptive analytics rely more on the real-time data continuously and timely collected from the IoT end devices. Based on the patterns identified by descriptive analytics, predictive analytics can predict future patterns by using real-time data. Prescriptive analytics is the final stage of IoT data analytics, which makes decisions based on the results of predictive analytics and provides the corresponding reaction and feedback.

- *Predictive analytics:* Based on the findings of descriptive and diagnostic analytics, predictive analytics serves as a forecasting tool, which can support the detection of tendencies and the prediction of future trends. Taking advantage of the predictive analytics, an industrial IoT system, for instance, can identify the machines that are most likely to break down, and prepare reactions in advance to minimize the potential loss. Although predictive analytics has numerous advantages, it is worth to aware of the risks of wrong predictions, since the accuracy of prediction highly depends on the data quality and stability of the situation. Therefore, it is necessary to treat the prediction carefully and optimize it continuously.

- *Prescriptive analytics:* The objective of prescriptive analytics is to prescribe what actions to take so that a potential issue can be eliminated and a promising trend can be fully utilized. An example of prescriptive analytics is that a large-scale IoT surveillance system can timely prevent the occurrence of bad accidents and react to unpreventable emergencies with prepared plans. However, prescriptive analytics requires not only historical data, but also external information due to the nature of statistical algorithms. Furthermore, prescriptive analytics generally uses sophisticated tools, such as the deep learning methods, which brings high computational complexity to the system. Therefore, the design of an IoT system should jointly consider the expected added values brought by prescriptive analytics and the additional consumptions alongside.

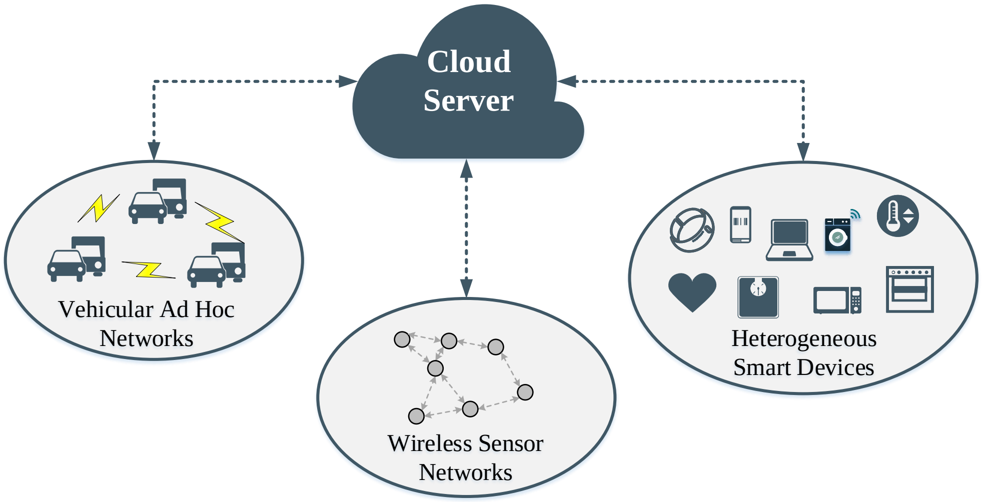


Figure 2.3: A general cloud-based IoT system architecture.

## 2.2 Architectures for Data Analytics in IoT Systems

The design of system architecture determines the dataflow in the IoT systems, which finally affects the processing and completion time of data analytics. Therefore, the architecture design needs to be seriously considered for data analytics in IoT systems. In this section, the general architectures of IoT systems that are able to support effective and efficient data analytics are extensively surveyed and analyzed, which migrates from the traditional cloud-based architecture to the newly developed edge-cloud collaborative architecture.

### 2.2.1 Cloud-based IoT System Architecture

In the initial deployment stage of IoT systems, cloud-based IoT system architecture is the dominating architecture. As shown in Fig.2.3, the system architecture consists of two major parts, namely, IoT end devices and the cloud computing platform.

- *IoT end devices* and the self-organized subnets are the fundamental components of the IoT systems, which have direct interactions with the physical environments through sensors and actuators. For example, in the case of a smart home system, temperature sensors sample the indoor temperature and upload the measurements to the cloud through either a wired gateway or wireless access point. The air conditioner can react to the feedback from the cloud, and adjust the temperature accordingly.

- *Cloud computing platform* is the remote data and control center in the IoT systems. IoT data collected from the IoT end devices are comprehensively processed at the cloud, while the results are sent back to the IoT end devices as feedback. Given the strong computing capability of the cloud server, it can support the comprehensive IoT data analytics and the massive amount of IoT data storage.

However, with the tremendous increase in the number of IoT end devices, the cloud-based IoT systems have met the following limitations which prevent them from being pervasively deployed in the large-scale application scenarios with critical requirements of real-time processing and analysis [9]:

- *Unstable cloud connection:* Cloud computing platform is remotely located, which can lead to the weak stability of the connections between cloud and IoT end devices. For example, in VANETs, handover of the fast-moving vehicles can result in the temporary absence of cloud computing service. The unstable cloud connection can lead to messy coordination of smart vehicles and finally incur bad traffic accidents. Thus, the IoT systems face a huge challenge – how to ensure normal operations in the absence of cloud connection.

- *Limited bandwidth:* Although the cloud server has the capability of processing the massive amount of data, the procedure of data uploading still challenges the bandwidth of the trunk link. In the case of industrial IoT systems, the huge amount of data imposes a heavy burden on the underlying network bandwidth, while overwhelming data can finally lead to system crash. Therefore, it is necessary to pre-process, especially effectively compress the IoT data first, instead of simply uploading all the data to the cloud.

- *High latency:* The data processing, analysis, and storage center is remotely located at the cloud server, which incurs unavoidable latency due to the procedure of data processing and communication. While in some systems, for instance, smart healthcare, real-time responses are needed for emergency cases, especially for elders living alone. Hence, how to reduce latency and provide real-time responses is also a critical challenge in certain IoT systems.

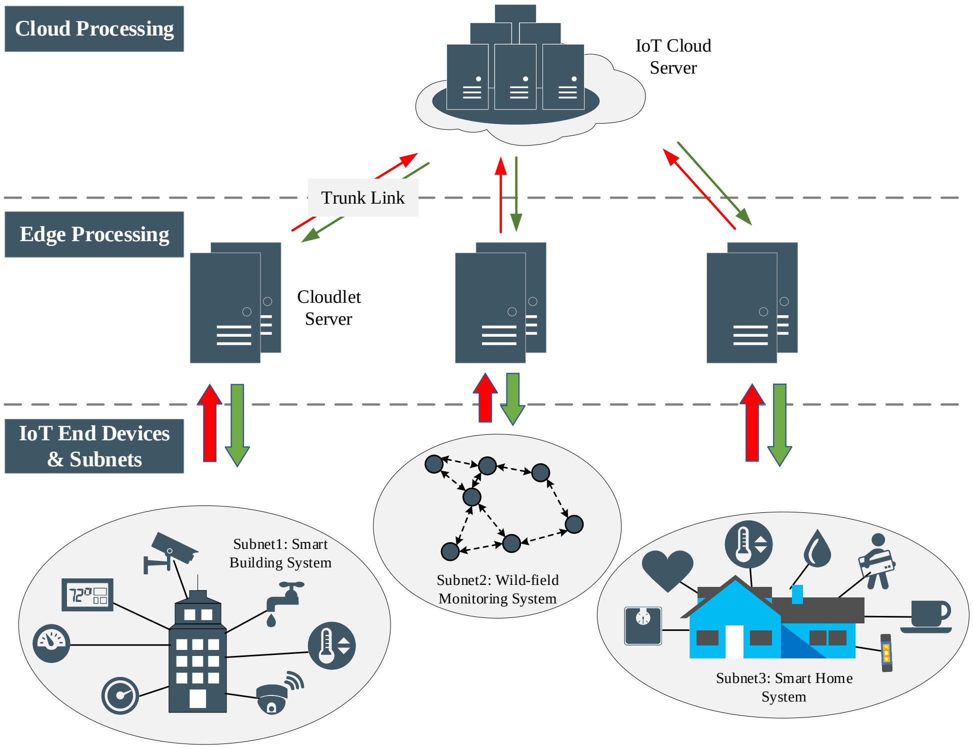

Cloud Processing
IoT Cloud Server
Trunk Link
Edge Processing
Cloudlet Server
IoT End Devices & Subnets
Subnet1: Smart Building System
72°
Subnet2: Wild-field Monitoring System
Subnet3: Smart Home System

Figure 2.4: A general edge-cloud collaborative system architecture.

Due to the technical limitations, the cloud-based architecture can hardly meet the critical requirements of real-time and massive data analytics in large-scale IoT systems. In this condition, edge computing has been introduced into the system as a promising solution, which enables the local and real-time processing for IoT end devices and offloading computational tasks from the cloud platform [10]. The correspondingly developed edge-cloud collaborative IoT system architecture is presented in the next subsection.

### 2.2.2 Edge-Cloud Collaborative IoT System Architecture

A general edge-cloud collaborative architecture for data analytics in heterogeneous IoT systems is depicted in Fig.2.4. The system architecture mainly consists of heterogeneous IoT end devices, edge computing devices, and the cloud computing platform, which are detailed below.

- *Heterogeneous IoT end devices and subnets* are still functioning as the fundamental layer in the edge-cloud collaborative architecture and directly interact with the physical environments. Due to the pervasive deployments of IoT systems, IoT end devices are heterogeneous with quite different capabilities (*e.g.*, computing, communication, and storage). Hence, device-to-device (D2D) communications among these devices request the support of multiple communication protocols (*e.g.*, ZigBee, LTE, and WiFi).

- *Edge computing devices* ("edge devices" for short) have been introduced into the IoT systems as a potential and promising solution, considering the technical limitations of cloud-based IoT systems. In the newly developed edge-cloud collaborative IoT system, edge computing devices locate in the intermediate layer, which can provide local and real-time processing to IoT end devices and can also execute preliminary data analytics so that the tasks can be offloaded from the cloud platform and the burden of trunk link can be relieved. In the system architecture proposed in Fig.2.4 [11], the lightweight cloudlet servers are utilized as the edge computing devices. In addition to the cloudlet server, any device that has the capabilities of computing, communication, and storage can be utilized as the edge device, for example, a femto base station, lightweight server and smart gateway. Even the UAV can serve as a mobile edge device.

- *Cloud computing platform* is the legacy of cloud-based architecture, which still serves as the remote data and control center in the edge-cloud collaborative IoT system architecture. Since edge devices have limited computing and storage capabilities, the cloud platform is responsible for the complex and comprehensive data analytics and the massive amount of data storage.

Functions of the major components in the system architecture have been explained in detail. The interactions, namely, data communications, among them are further given as follows.

- *IoT end devices and edge devices:* Edge devices are equipped with RF modules of different communication protocols, which can support the data uploading from heterogeneous IoT end devices. As aforementioned, edge devices serve as the intermediate layer in the edge-cloud collaborative IoT system architecture. Therefore, besides the data uploading, edge devices are also responsible for sending and relaying the reaction and feedback generated by either edge devices or the cloud platform back to the IoT end devices.

- *Edge devices and cloud platform:* Edge devices upload the pre-processed data to the cloud so that the burden on the trunk link can be relieved. Cloud platform sends back the results of comprehensive data processing and analysis then. As compared to the cloud platform, the capabilities of edge devices are weaker. Thus, the data processing speed of the edge is slower than that of the cloud. While as mentioned in the previous subsection, data offloading to the cloud can incur extra latency due to the procedure of data communication. Therefore, it is necessary to balance the trade-off between processing time and communication time, when optimizing the task offloading of data analytics.

## 2.3 Applications of Data Analytics in IoT Systems

A comprehensive survey on the existing applications of data analytics in IoT systems is conducted in this section, which includes smart city, smart healthcare, industrial IoT, social network, and environmental monitoring. These applications are analyzed from the perspectives of system design and shortcomings of performance.

### 2.3.1 Smart City

Hut architecture as depicted in Fig.2.5 is specifically designed for the smart city, which can provide the service of real-time data processing based on the historical data analytics [12]. For example, in abnormal event detection, the historical batch data are used to learn the normal patterns so that the abnormality of real-time data streams can be timely and accurately identified. Two specific use cases using hut architecture are analyzed as instances. One is the Madrid transportation system, where 3000 traffic sensors are deployed on the M30 ring road by Madrid city council. Based on the descriptive analytics of the historical traffic data collected by the sensors, bad traffic is detected in real-time in order to prevent the worse congestion and facilitate public transportation. The other case is the Taiwan energy management system, where malfunctioning electronic devices and unusual appliance usages are monitored and detected in real-time through excessive power dissipation.

There is another work that also focuses on the prevention of traffic congestions in Madrid [13]. Different from the above work [12], not only traffic data from the city council of Madrid but also media data from Twitter and weather data are jointly considered to predict and prevent

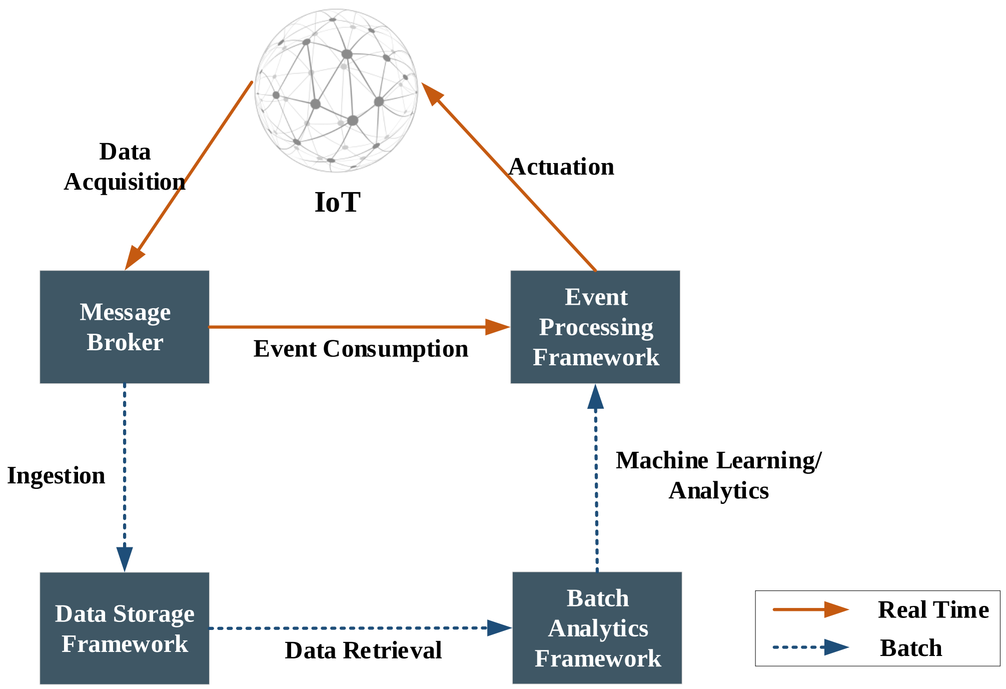


Figure 2.5: Hut architecture for data analytics in smart city.

traffic congestions using the Bayesian network in a real-time way. With the comprehensive consideration of multiple factors and utilization of the Bayesian network model, the prediction of traffic congestions is more accurate. In [13], multiple data streams are jointly utilized for the same aim, namely, prediction of traffic congestions. In order to fully extract the relations among multiple data streams in the smart city, latent Dirichlet allocation (LDA), a topic extraction method that is generally used in text analysis, is exploited to uncover the underlying structure of the multiple data streams [14].

Although several efforts have been spared on the application of data analytics in the smart city, there are some challenges remaining as listed below.

- *Factor selection:* Smart city is a complex scenario with multiple data streams of different physical factors. It is a critical challenge to select the proper factors for the specific target. In [13], traffic, media and weather data streams are utilized to predict the traffic congestions. In [14], LDA based method is utilized to uncover the relation between traffic and weather data streams. The relations among the mentioned factors may be easy to aware from common senses. While for some other data streams, the underlying relations may not be perceptual. It is necessary to discover a scientific way to uncover

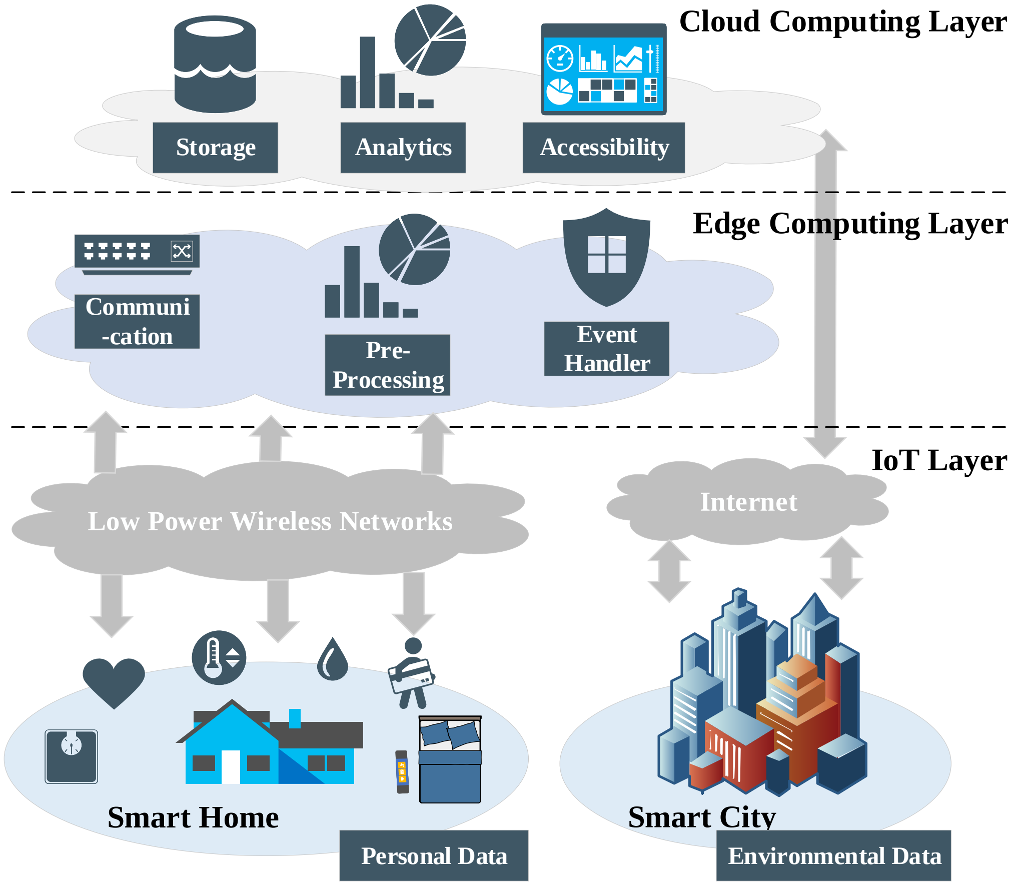


Figure 2.6: An edge-cloud collaborative IoT system architecture for OSA detection.

the underlying relations so as to improve the accuracy of prescriptive analytics.

- *Time window selection:* In the smart city, most of the applications have requirements of real-time analytics, *e.g.*, traffic coordination. However, the amount of data generated is huge, which imposes a heavy burden on data communications and can lead to the high latency of data analytics. Therefore, the selection of a proper time window for data collection is also among the most critical challenges. It is necessary to develop a method that can adaptively adjust the time window, which can automatically decrease to capture times of high interest in a finer granularity and adjust again in times of low interest.

### 2.3.2 Smart Healthcare

Smart healthcare is among the most promising application scenarios where IoT systems can change the way of living [15]. IoT technology enabled smart healthcare system has already been utilized to do long-term monitoring of chronic diseases. While for spasmodic diseases, particularly the real-time emergency event detection for elders living alone, it has higher requirements on the capability of real-time analytics, which needs to be seriously considered in the design of IoT-enabled smart healthcare system. There have been a few works in this area as analyzed below.

In [16], a real-time monitoring architecture is proposed for obstructive sleep apnea (OSA, a severe sleep disorder) detection based on the collaboration of edge and cloud computing, as depicted in Fig.2.6. For real-time OSA detection, multiple related factors are monitored including sleep environment (collected by smart city system), sleep status, physical activities and physiological parameters (collected by the smart home system). Edge and cloud play different roles in processing the measurements of these factors. More specifically, cloud computing with stronger capability is responsible for batch processing enabled pattern recognition and event prediction. Edge computing as analyzed in the section of architecture design is more close to the monitoring devices, which is utilized to implement the real-time OSA detection and reduce the latency of reaction and feedback. Through the edge computing enabled real-time detection, lives can be saved from OSA.

Another commonly occurred disease of elders is dementia, which affects 46 million people around the world. In [17], an IoT system is specifically designed for dementia care, termed as TIHM (technology integrated healthcare management). TIHM involves the families with dementia patients, clinics and hospitals with healthcare experts, small and medium-sized IoT companies, and academic groups with healthcare, economic, security, and technical experts. The system architecture of TIHM is quite similar to the OSA detection system (Fig.2.6), real-time data of environments, patients' physiological parameters, and their daily lifestyles are collected through environmental sensors, medical devices, wearable technologies, and interactive applications. Lightweight servers provided by the IoT companies are functioned as edge computing devices, while the TIHM project has a more powerful backend server providing the service of cloud computing. Based on the data analytics, the needs of dementia patients can be identified in an early stage, which allows the clinical team to provide a timely response and prevent the patients from exacerbating ill health.

Smart healthcare systems can improve the quality of life and scientifically extend the life-time of patients. However, the issue of the privacy protection of patients' information is remaining unsolved.

### 2.3.3 Industrial IoT

Industrial IoT (IIoT) is the leverage and reality of IoT technology in the context of industrial transformation. On one hand, the transformation can optimize the performance and boost productivity while cutting the total cost. On the other hand, it is able to predict and prevent potential machinery failures [18].

From the technical perspective, IIoT paves the way to connect all the industrial assets, such as machines and control systems, through the evolving machine-to-machine (M2M) and industrial communication technologies [19]. More specifically, the IIoT can facilitate the process automation domain in the following three aspects, namely, supervision, closed-loop networked control, and interlocking. However, closed-loop networked control and interlocking are highly sensitive to delay and require bounded delay at the millisecond level (10-100 ms), which imposes a heavy burden on the real-time analytics in IIoT systems [20]. In order to meet the critical requirement of real-time analytics, a three-tier IIoT system architecture has been specifically designed for delay mitigation, as depicted in Fig.2.7 [19]. In terms of the functions of each tier in the architecture, the edge tier defines the domain in which IIoT components interact with each other, which consists of sensors, actuators, and controllers interconnected by independent local area networks to an IIoT edge gateway. The IIoT edge devices are in turn connected to the platform tier for global coverage. Finally, the platform tier takes advantage of the service network to establish connections with the enterprise tier that implements domain-specific applications and provides interfaces to the end-users. The latency level incurred by the processing at each tier is also labeled in Fig.2.7. It can be seen that the edge tier can complete tasks within milliseconds, which can meet the critical requirements of bounded delay in closed-loop networked control and interlocking applications.

Although the three-tier IIoT system architecture has been widely accepted for delay mitigation, the explosive growth of IIoT applications, especially in terms of their scale and complexity, has dramatically increased the difficulty in ensuring the desired real-time performance.

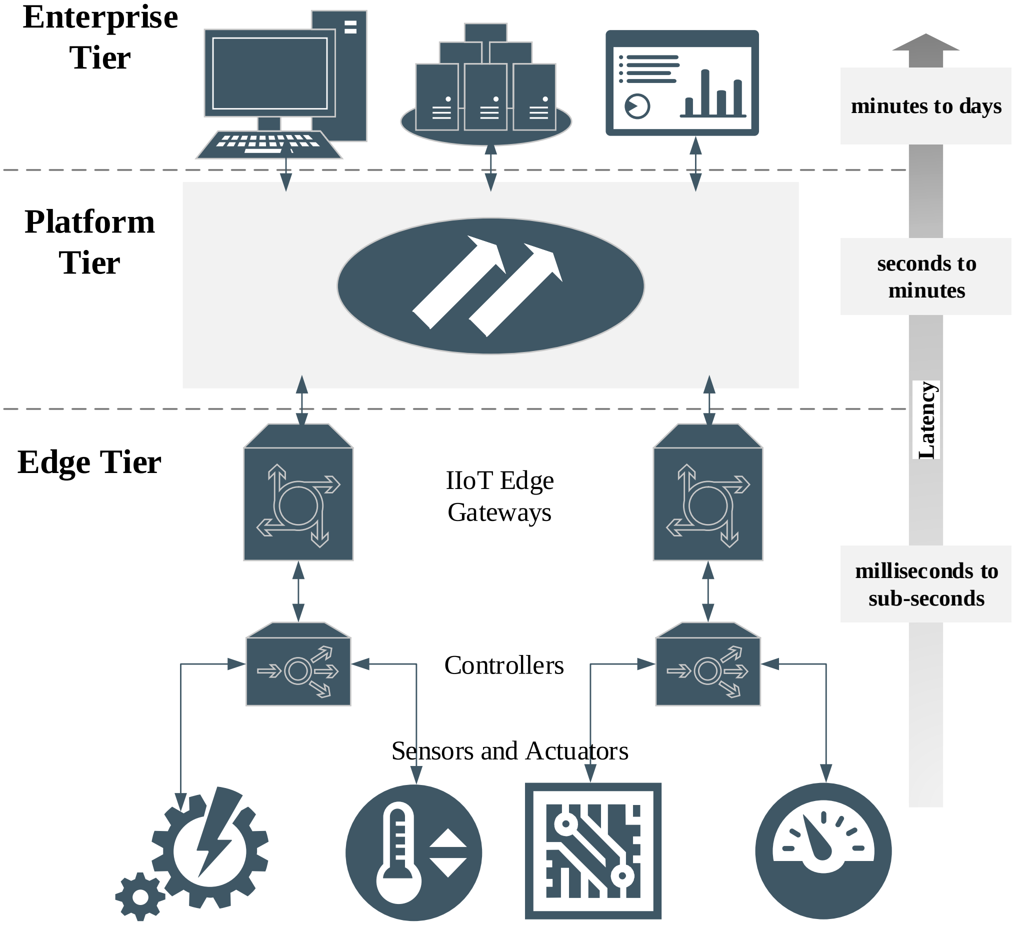


Figure 2.7: Three-tier IIoT system architecture.

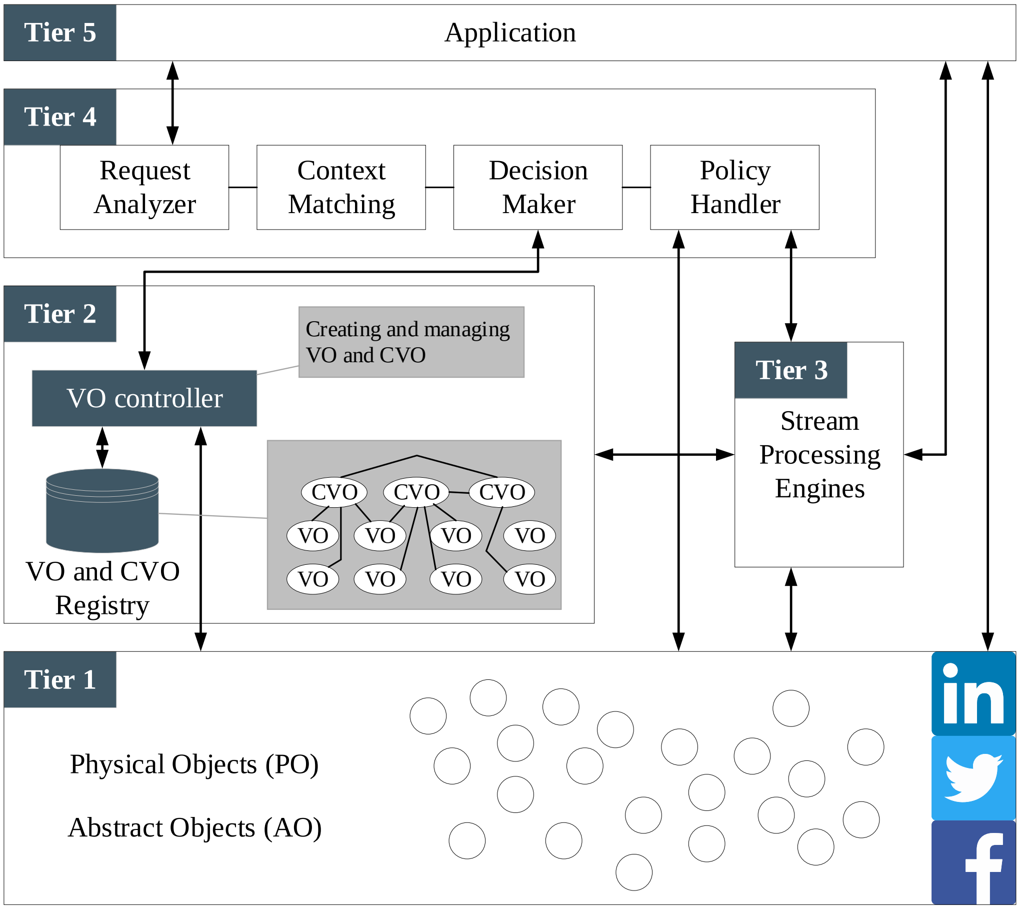


Figure 2.8: Architecture for data analytics in SIoT system.

In addition to the challenge of real-time performance, energy-efficient operations, interoperability among heterogeneous IIoT devices, and security and privacy all need to be seriously considered in the IIoT systems.

### 2.3.4 Social Network

With the pervasive deployment of IoT, not only people but also physical and virtual objects are interconnected through the evolving communications and embedded systems technologies [21]. In such a condition, the social IoT (SIoT) system has been proposed [22]. Similar to the online social network (OSN) for people, SIoT introduces the concept of social relationships into objects. However, before fully implementing the concept of SIoT, an SIoT system architecture needs to be developed, where IoT end devices can be controlled, managed and monitored in a real-time and cognitive way. There have been a few works focusing on this problem as analyzed below.

As shown in Fig.2.8, an SIoT system architecture has been proposed to intelligently and cognitively create, manage, control and monitor the SIoT objects in real-time [23]. In the proposed architecture, real-world objects are termed as physical objects (POs), while the services that need special skills are termed as abstract objects (AOs). PO and AO jointly compose tier 1 of the system architecture, which have direct interactions with the real-world and are then virtually represented as virtual objects (VOs) in tier 2. The new services incurred by the combination of VOs are termed as composite VOs (CVOs). Tier 3 is the most important component in the system architecture, namely, stream processing engines, which is the part that enables real-time analytics. Tier 4 is the decision making layer, which is executed based on the results provided by the stream processing engines. Tier 5 indicates the applications and services that the SIoT system can provide. The architecture provides a social interaction framework for IoT end devices functioning similar to the OSN for people and supports the real-time data stream processing in the meantime. However, there are still many aspects, especially the applications in tier 5 that need to be further investigated.

Anomaly detection in the cross-platform SIoT systems has already been analyzed as a case study [24]. In the enlarging cross-platform SIoT systems, the number of heterogeneous connected devices has been increasing tremendously, which brings a high risk of information loss and malicious access to the systems. In [24], an intelligent sensing model for anomaly detection (ISMA) has been proposed for the cross-platform SIoT systems, where anomalies refer to the malicious users misleading the systems with fraudulent information. The ISMA strategy deliberately induces faulty data (termed as cognitive tokens) to attract malicious users and then identifies and classifies the anomalies with the error-based outlier filters. A common login system for different platforms in SIoT system is introduced into the whole architecture as a part of collaborative anomaly identification across different platforms. A fair play point approach is used for the determination of anomalies, which improves the anomaly detection accuracy, as compared to the existing methods, for example, SVM-RBF (support vector machine-radial basis function) and sigmoid approach. However, this work still depends on the historical data for off-site evaluations, which needs to be further developed to meet the requirements of real-time services to provide anomaly detection with continuous user monitoring.

### 2.3.5 Environmental Monitoring

IoT technology has been pervasively applied to environmental monitoring such as oceanic atmosphere monitoring and forest fire surveillance, taking the advantages of low cost and flexible deployment of IoT enabled systems [25, 26]. In [27], an edge-cloud collaborative IoT system architecture is proposed for data analytics in environmental monitoring, where UAVs are deployed and utilized as mobile edge devices. Wireless sensor nodes and the cloud platform are involved for environmental sensing and complex data analytics respectively. Moreover, a UAV-enabled spatial data sampling scheme is further developed based on the system architecture, in order to overcome the challenge of accurate and efficient data sampling and reconstruction. Taking advantage of the UAVs, urgent tasks of data analytics can be timely completed at the mobile edge devices.

Furthermore, one of the most significant functions of data analytics in IoT-enabled environmental monitoring system is disaster detection and management. IoT enabled natural disaster management approaches have been surveyed and summarized in [28], such as early warning, notification, knowledge aggregation, remote monitoring, and victim localization. Data analytics, particularly real-time analytics, plays a key role in the disaster management system for real-time decision making, which can save lives and protect personal belongings.

The main technical challenge in such systems is also the issue of security, since personal and private data are collected for environmental monitoring and disaster detection. Thus, beyond efficient and collaborative, the system architecture needs to be secure.

## 2.4 Chapter Summary

In this chapter, data analytics in IoT systems has been thoroughly studied. The characteristics of data analytics in IoT systems have been elucidated firstly. Afterwards, the IoT system architectures for data analytics are investigated, where both the traditional cloud-based architecture and the newly developed edge-cloud collaborative architecture are analyzed. By exploitation of the edge-cloud collaborative architecture, data analytics have been applied in several application scenarios, which have been extensively surveyed and analyzed from the perspectives of system design and shortcomings of performance.

# Chapter 3

# Recursive Principal Component Analysis based Data Outlier Detection and Sensor Data Aggregation in IoT Systems

IoT is emerging as the underlying technology of our connected society, which enables many advanced applications. In IoT-enabled applications, information of application surroundings is gathered by networked sensors, especially wireless sensors due to their advantage of infrastructure-free deployment. However, the pervasive deployment of wireless sensor nodes generates a massive amount of sensor data, and data outliers are frequently incurred due to the dynamic nature of wireless channels. As the operation of IoT systems relies on sensor data, data redundancy and data outliers could significantly reduce the effectiveness of IoT applications or even mislead systems into unsafe conditions. In this chapter, a cluster-based data analysis framework is proposed using R-PCA, which can aggregate the redundant data and detect the outliers in the meantime. More specifically, at a cluster head, spatially correlated sensor data collected from cluster members are aggregated by extracting the PCs, and potential data outlier is determined by the abnormal SPE score, which is defined as the square of residual value after extraction of PCs. With R-PCA, the parameters of the PCA model can be recursively updated to adapt to the changes in IoT systems. The cluster-based data analysis framework also relieves the computational and processing burdens on sensor nodes. Practical databases based simulations have confirmed that the proposed framework efficiently aggregates the correlated sensor data with high recovery accuracy. The data outlier detection accuracy is also improved by the proposed method as compared to the other algorithms in the literature.

## 3.1 Introduction

In the anticipated era of IoT, not only people but also physical and virtual things are interconnected based on the evolving sensing, communication and processing technologies, which directly enables many advanced applications in the connected society, including intelligent transportation systems, smart buildings, and smart grids [29–31]. In IoT-enabled applications, sensor networks are the most important component, since critical information from both external surroundings and inner systems is sampled by networked sensors. Currently, more wireless sensors are applied than wired sensors in IoT applications, as wireless sensors can be self-organized into WSNs and randomly deployed without the requirement of additional infrastructure. From the technical point of view, WSNs are supported by the low-power and low-cost devices using ZigBee, Bluetooth and Wi-Fi technologies, which normally work in unlicensed bands. The use of unlicensed bands brings additional deployment flexibility of WSNs but introduces uncontrollable interference simultaneously [32]. Given the low processing capability of sensor nodes, data outliers could occur as a result of the increased interference, the delay incurred by uncoordinated channel access, or simply the malfunctions of sensor nodes.

Data outliers are the sensor readings that do not follow the normal pattern of sensor data in IoT systems, namely, deviate dramatically from the ground truth [33]. According to the frequency of occurrence, data outliers can be classified into random outliers and continuous outliers. A random outlier may be caused by occasionally failed communication. Continuous outliers are possibly caused by the low battery or malfunction on hardware. In industrial IoT systems, these collected data outliers could mislead the whole IoT system into unsafe conditions, since the actuators are driven by the information inferred from sensor data [34]. Therefore, data outliers have to be detected in real-time in order to avoid erroneous decisions made by the IoT systems.

Furthermore, pervasive and redundant deployment of wireless sensor nodes imposes a challenge for efficient sensor data collection. Normally, IoT systems are resource-constrained, while a significant proportion of network resources are consumed by data transmissions [35]. Hence, a well-designed data aggregation algorithm could reduce the number of data transmissions while ensuring data accuracy [36, 37]. Since data outlier detection and data aggregation are among the most critical requirements of IoT data processing [38, 39], there are increasing research interests in this area, which are summarized in Section 3.2. Considering the low detection accuracy and high computational complexity of existing algorithms, a novel data analysis framework is still needed for performance enhancement of data outlier detection schemes.

In this chapter, a cluster-based real-time data analysis framework using R-PCA is proposed for data outlier detection and data aggregation, which exploits the spatial correlation in sensor data. With R-PCA, spatially correlated sensor data can be aggregated by extracting the PCs, and the transformation basis is recursively updated to track the IoT system changes. In the meantime, SPE score, which is defined as the square of residual value after extraction of PCs, can be used to identify the data outliers since any abnormal data could lead to significantly increased SPE score. As compared to the previous work [40], an outlier detection threshold and an R-PCA based outlier diagnosis algorithm are further proposed to accurately identify the specific data outlier.

The proposed data analysis framework works along with cluster-tree network topology. Sensor data are diagnosed and aggregated by R-PCA based algorithms at the cluster head, and then outlier-free and aggregated sensor data are forwarded to the data center. All the sensor data are recovered at the data center for further analysis. Simulations have been conducted based on the databases provided by NDBC-TAO [41] and Intel Lab [42]. Simulation results confirm that the proposed data analysis framework improves the outlier detection accuracy, as compared to univariate and multivariate outlier detection algorithms in the literature. Besides the accurate outlier detection, our proposed scheme can efficiently aggregate the redundant data and ensure the recovery accuracy at the data center as well.

The rest of the chapter is organized as follows. In Section 3.2, the related work on data outlier detection and sensor data aggregation are surveyed and summarized. The concept of PCA is clarified in Section 3.3. Section 3.4 elaborates the details of the proposed data analysis framework. The performance evaluation of the proposed framework is analyzed in Section 3.5. Finally, all the contributions of this chapter are summarized in Section 3.6.

## 3.2 Related Work

Data outlier detection and sensor data aggregation have been widely considered as the main technical challenges of IoT systems. Several studies have been carried out in this area using different methods. Since the proposed R-PCA based data outlier detection and aggregation method mainly utilizes the spatial correlation of sensor data, spatial correlation based data outlier detection algorithms and data aggregation algorithms are briefly overviewed below.

### 3.2.1 Spatial Correlation based Data Outlier Detection

Existing studies on spatial correlation based outlier detection can be summarized into the following categories.

- *Majority Voting* is a classical spatial correlation based data outlier detection method. A local sensor node is detected as abnormal when its reading is substantially different from the majority of its neighbors. For instance, in distributed fault detection (DFD) algorithm [43], general and differential Euclidean distances between sensor data generated from local sensor node and its neighbors were used as outlier detection criteria. The local node was detected as abnormal when the Euclidean distances between most of its neighbors were over a certain threshold. However, due to the excessive dependence of a sensor node on its neighbors, the data outlier detection accuracy was low when the network was sparse.

- *Classifiers* are applied to detect data outlier by training a “normal” model and then classifying the under detecting data into normal and abnormal. Support vector machine (SVM) is among the most commonly used classifiers, especially the lightweight quarter-sphere SVM. Generally, the radius of the quarter sphere is trained by “normal” data, and any sensor data that falls out of the quarter sphere is detected as abnormal [44, 45]. The major concern of the classifier-based outlier detection is its high computational complexity, since the local-based algorithms are processed at sensor nodes.

- *PCA* Data outliers can generate dramatic variations in the residual value after the extraction of principal components. Therefore, PCA combined with detection criteria (*e.g.*, SPE score, $T^2$ score) can be used to detect data outliers. In [46], an anomaly detection algorithm was proposed based on the combination of the conventional PCA method and the SPE score.

The abnormal data was detected by the significant SPE score. In [47], kernel PCA was adopted to mitigate the linear limitation of sensor data, where the geometric information was considered as well. Given the dynamic conditions in WSNs, PCA with a static transformation basis could not track the variations [46,47]. Hence, a robust recursive fault detection algorithm was proposed by *S.Chan et al.* [48], which particularly focused on the sensitivity to minor system changes and robustness to dramatic data faults. However, the robust recursive fault detection algorithm in [48] was loaded on a single sensor node. Given the constrained processing capacity and power supply of a sensor node, the proposed algorithm in [48] was too complex to be implemented. In this chapter, R-PCA [49] is applied to track the changes in the IoT systems, while outlier detection and diagnosis algorithms are proposed based on the clusters that typically have more computational resources.

### 3.2.2 Spatial Correlation based Sensor Data Aggregation

In this subsection, existing studies on sensor data aggregation are summarized. Besides the conventional spatial correlation based data aggregation algorithms, both compressive sensing and PCA based algorithms are also reviewed.

- In *Conventional Data Aggregation* algorithms, sensor nodes are clustered by spatial correlation, but sensor data are simply aggregated by basic operations, such as *mean and median*, without full exploitation of data correlation and accuracy for data aggregation. For instance, *Sun et al.* [50] proposed a trust-based framework for data aggregation in WSNs. Every sensor data was assigned a weight, according to the trustworthiness ranked by comparison with historical data and neighbor data. Afterwards, the weighted mean value calculated at the aggregator was used as the aggregated data.

- *Compressive Sensing (CS)* transforms raw sensor data into the sparse domain at the sender, to reduce the overall communication overload, while increases the complexity of receiver for data recovery. *Xiang et al.* [51] proposed a CS-based data aggregation scheme. Particularly, they adopted diffusion wavelets to sparsify the raw sensor data, which further reduced the communication overload while the data recovery faced high computational complexity. Besides, sparsity might exist in the environments, but the compressive sensing method was still limited by the restricted isometry property [52].

• *PCA* has been widely used to aggregate sensor data, due to the essence of principal component extraction. A PCA-based hierarchical data aggregation algorithm was proposed in [53]. At each level, sensor data from the lower level was aggregated and forwarded. However, only the temperature reading was investigated without consideration of multivariate sensor readings in current IoT systems. Considering the multivariate data aggregation, a novel principal components-based context compression (PC3) algorithm was proposed [54]. PC3 algorithm was also able to adaptively update the transformation basis of PCA by alternating the learning and compression operations at sensor nodes. However, PC3 was too complex to be deployed on a sensor node with limited computational capacity. In this chapter, an improved R-PCA algorithm is proposed relying on cluster processing, which recursively updates the transformation basis with only the newest data so that the memory occupied by historical data in [54] could be released.

## 3.3 Principal Component Analysis

Before developing the R-PCA and the data analysis framework, conventional PCA is introduced in this section for clarification of the concept. PCA is a normally used mathematical tool for correlated data aggregation [55]. The reduction in data dimension is obtained by projecting the raw data matrix ($\mathbf{X}$) into a subspace defined by the extracted PCs. That is

$$\mathbf{Y}_{[l\times n]} = \mathbf{P}_{[l\times m]}\mathbf{X}_{[m\times n]} \ (l < m), \tag{3.1}$$

where $\mathbf{X}$ is the raw data matrix consisted of $m$ physical variables and $n$ samples, $\mathbf{P}$ is the transformation basis consisted of PCs, and $\mathbf{Y}$ is the projection of $\mathbf{X}$ in the subspace termed as score matrix. In other words, the aim of PCA is to derive a transformation basis $\mathbf{P}$ that can make the projection of $\mathbf{X}$, *i.e.*, $\mathbf{Y} = \mathbf{PX}$, linearly uncorrelated and less-dimensional. $\mathbf{P}$ is obtained through the following calculations.

First, the covariance matrix of $\mathbf{X}$ is calculated as

$$\mathbf{C_X} = \frac{1}{n-1}\overline{\mathbf{X}}\,\overline{\mathbf{X}}^T, \tag{3.2}$$

where $\mathbf{X}$ is normalized to a zero-mean and unit-variance matrix $\overline{\mathbf{X}}$, in order to mitigate the impact of different scales.

Afterwards, the covariance matrix $\mathbf{C_X}$ is decomposed by eigenvalue decomposition,

$$\mathbf{C_X} = \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T, \tag{3.3}$$

where the eigenvectors $\{\mathbf{e}_i, i = 1, 2, \ldots, m\}$ of $\mathbf{C_X}$ are organized as columns in $\mathbf{E} = \left[\mathbf{e}_1^T, \mathbf{e}_2^T, \ldots, \mathbf{e}_m^T\right]$ and $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues $\{\lambda_i, i = 1, 2, \ldots, m\}$ of $\mathbf{C_X}$.

The first $l(l < m)$ largest columns in $\mathbf{E}$ are considered as principal components. $l$ is the number of principal components and determined by the cumulative percentage formula as

$$\gamma = \frac{\sum_{i=1}^{l} \tilde{\lambda}_i}{\sum_{i=1}^{m} \tilde{\lambda}_i} \times 100\%, \tag{3.4}$$

where $\tilde{\lambda}_i$ is the $i$th largest diagonal element in the reordered eigenvalue matrix ($\tilde{\boldsymbol{\Lambda}}$) and $\gamma$ is determined by the specific application requirements, *e.g.*, 80%. Eigenvector and eigenvalue matrices are then reordered and reduced into $\tilde{\mathbf{E}}_l = \left[\tilde{\mathbf{e}}_1^T, \tilde{\mathbf{e}}_2^T, \ldots, \tilde{\mathbf{e}}_l^T\right]$, and $\tilde{\boldsymbol{\Lambda}}_l = diag([\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_l])$ accordingly.

Finally, *transformation basis* $\mathbf{P}$ is set to the transposition of the reordered and reduced eigenvector matrix, *i.e.*, $\mathbf{P} = \tilde{\mathbf{E}}_l^T$. Since the subspace is defined by the extracted principal components, data matrix $\overline{\mathbf{X}}$ can be projected into the subspace by the transformation basis $\mathbf{P}$ as $\mathbf{Y} = \mathbf{P}\overline{\mathbf{X}} = \tilde{\mathbf{E}}_l^T\overline{\mathbf{X}}$.

*Residual value* is the remaining after extracting principal components from raw data matrix, which is formulated as

$$\epsilon = \overline{\mathbf{X}} - \mathbf{P}^T\mathbf{Y} = \overline{\mathbf{X}} - \tilde{\mathbf{E}}_l\tilde{\mathbf{E}}_l^T\overline{\mathbf{X}}. \tag{3.5}$$

Since principal components are the variables with largest variances, the residual value (3.5) tends to be small.

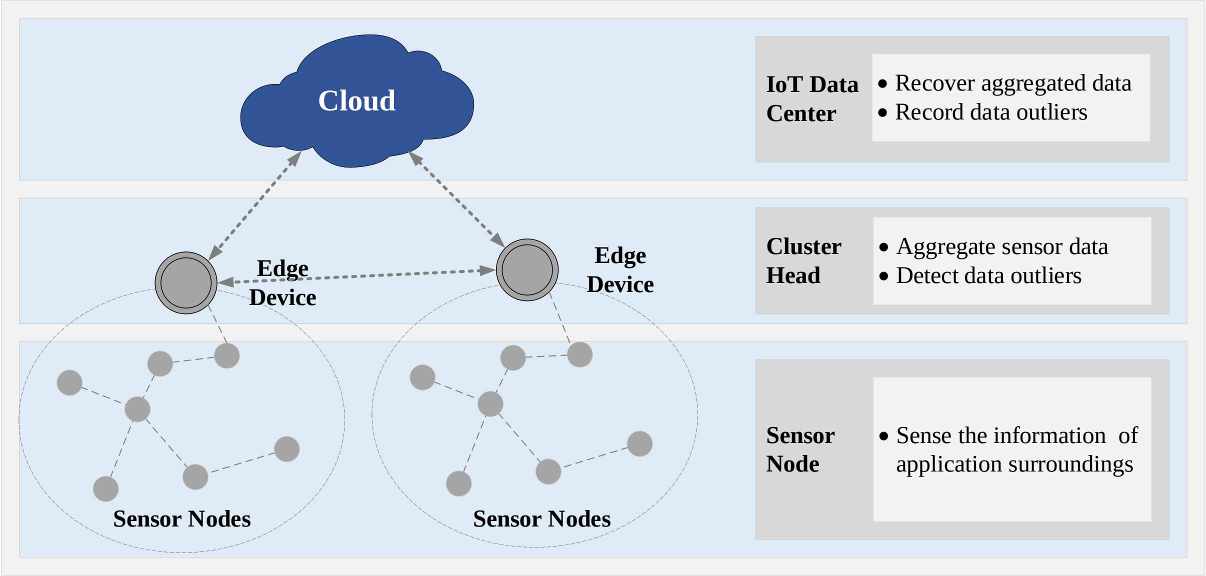


Figure 3.1: R-PCA based multivariate data analysis framework for outlier detection and data aggregation.

*SPE score* is defined from the residual value and used as the outlier detection criterion,

$$SPE = ||\epsilon||_2^2 = ||\overline{\mathbf{X}} - \tilde{\mathbf{E}}_l\tilde{\mathbf{E}}_l^T\overline{\mathbf{X}}||_2^2. \qquad (3.6)$$

SPE score is sensitive to data outliers, since a potential data outlier can generate a dramatic variation on the residual value.

## 3.4 Proposed Data Analysis Framework Using Recursive Principal Component Analysis

In order to detect data outliers and aggregate sensor data, a cluster-based data analysis framework is proposed here by using R-PCA, as shown in Fig.3.1.

As discussed in Section 3.3, the dimension of sensor data can be aggregated by projecting raw sensor data into a subspace defined by principal components, and data outliers can be detected by the abnormal SPE score. However, due to the dynamic conditions in IoT systems, the static transformation basis in conventional PCA cannot adapt to the system changes. Therefore, R-PCA is further developed, where all the parameters in the PCA model are recursively updated to adapt to the variations in the IoT systems. Although R-PCA introduces extra computations into the IoT systems, these computations are offloaded to cluster heads and the IoT data center instead of creating an additional burden on sensor nodes, which are more affordable and practical in IoT systems.

By exploitation of R-PCA, a data analysis framework is proposed along with the cluster-tree network topology, as shown in Fig.3.1. Sensor nodes are gathered into clusters by a clustering algorithm, such as the adaptive k-means algorithm [56]. Edge devices with stronger computational capacity are assigned to the clusters and served as cluster heads, which are responsible for sensor data aggregation and data outlier detection. The cloud computing platform is served as the data center of the whole IoT system. At the IoT data center, all the aggregated sensor data are recovered and outliers are recorded for further analysis.

More details on the data processing at each component of the proposed data analysis framework are explained as follows.

### 3.4.1 Data Sampling at Sensor Nodes

Given the limited computational capacity and power supply of a sensor node, a sensor node is designed to be simply responsible for sampling the application surroundings and transmitting the sampled sensor data to its cluster head in the proposed data analysis framework.

Considering the general condition that multiple sensors are embedded on one sensor node, the data matrix generated by a sensor node $i$ is multivariate and mathematically expressed as

$$\mathbf{X}_i = \begin{bmatrix} x_{i,1}(1) & x_{i,1}(2) & \dots & x_{i,1}(n) \\ x_{i,2}(1) & x_{i,2}(2) & \dots & x_{i,2}(n) \\ \vdots & \vdots & \ddots & \vdots \\ x_{i,m}(1) & x_{i,m}(2) & \dots & x_{i,m}(n) \end{bmatrix}, \tag{3.7}$$

where $m$ is the number of physical variables (like temperature and humidity), while $n$ is the number of samples.

At a certain time $t$, data vector generated by sensor node $i$ is $\mathbf{X}_i(t) = \left[x_{i,1}(t), x_{i,2}(t), \dots, x_{i,m}(t)\right]^T$.

## 3.4.2 Data Outlier Detection and Aggregation at Cluster Head

At cluster head, sensor data from its members are collected $\{\mathbf{X}_1, \ldots, \mathbf{X}_k\}$, where $k$ is the number of nodes in the cluster. Since the correlation between neighbor sensor data with the same physical property is higher, the cluster head then reorganizes the sensor readings into $\{\hat{\mathbf{X}}_1, \ldots, \hat{\mathbf{X}}_m\}$ according to the physical properties, where $m$ is the number of physical variables. Each new matrix $\hat{\mathbf{X}}_j$ $(j = 1, 2, \ldots, m)$ becomes

$$\hat{\mathbf{X}}_j = \begin{bmatrix} \hat{x}_{1,j}(1) & \hat{x}_{1,j}(2) & \ldots & \hat{x}_{1,j}(n) \\ \hat{x}_{2,j}(1) & \hat{x}_{2,j}(2) & \ldots & \hat{x}_{2,j}(n) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{x}_{k,j}(1) & \hat{x}_{k,j}(2) & \ldots & \hat{x}_{k,j}(n) \end{bmatrix}. \tag{3.8}$$

Data processing on $\hat{\mathbf{X}}_j$ is listed in Algorithm 1 and 2, which can be parted into two major phases, namely, initialization and recursion. In initialization phase, parameters in R-PCA model are initialized. In the recursion phase, parameters are kept updating for data outlier detection and sensor data aggregation. Details are explained as follows. *For simplification,* $\hat{\mathbf{X}}_j$ *is remarked as* $\mathbf{X}_{[k\times n]}$ *in the following statements.*

#### 3.4.2.1 Initialization Phase

**Normalization** Given the raw sensor data matrix $\mathbf{X} = \left[\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_k^T\right]^T$, where $k$ is the number of sensor nodes and $\mathbf{x}_i$ $(i = 1, 2, \ldots, k)$ is the vector of $n$ samples from sensor node $i$, $\mathbf{x}_i = [x_i(1), x_i(2), \ldots, x_i(n)]$. $\mathbf{X}$ is normalized to a zero-mean and unit-variance matrix $\overline{\mathbf{X}}$, in order to mitigate the impact of different scales. The normalization is given by

$$\overline{x}_i(j) = \frac{x_i(j) - \mu_i}{\sigma_i}, j = 1, \ldots, n, \tag{3.9}$$

where $\mu_i$ and $\sigma_i$ are the mean and standard deviation of $\mathbf{x}_i$.

**Parameters Initialization** The eigenvector matrix $\mathbf{E}$ and eigenvalue matrix $\boldsymbol{\Lambda}$ of $\mathbf{C}_X$ are initialized by (3.2)-(3.3).

$\mu_{SPE}$ and $\sigma_{SPE}$ are initialized by calculating the mean and standard deviation of series

$\{SPE(j), j = 1, 2, \ldots, n\}$, while $SPE(j)$ is calculated as

$$SPE(j) = \|\overline{\mathbf{x}}(j) - \tilde{\mathbf{E}}_l \tilde{\mathbf{E}}_l^T \overline{\mathbf{x}}(j)\|_2^2, \tag{3.10}$$

where $\overline{\mathbf{x}}(j) = [\overline{x}_1(j), \overline{x}_2(j), \ldots, \overline{x}_k(j)]^T$.

#### 3.4.2.2 Recursion Phase

**Normalization** The newly collected sensor data at time instance $t$, $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_k(t)]^T$ need to be normalized as well. However, before normalization, the mean and variance of raw sensor data are updated first. Since $\mu_i = \frac{1}{n}\sum_{j=1}^{n} x_i(j)$ , therefore the mean value is recursively updated by

$$\mu_i(t) = (1 - \beta) \cdot \mu_i(t-1) + \beta \cdot x_i(t), \tag{3.11}$$

where $\beta = \frac{1}{t}$ is the forgetting factor [57]. Similarly, the variance is recursively updated by

$$\sigma_i^2(t) = (1 - \beta) \cdot \sigma_i^2(t-1) + \beta \cdot (x_i(t) - \mu_i(t))^2. \tag{3.12}$$

$x_i(t)$ is normalized to $\overline{x}_i(t)$ by $\mu_i(t)$ and $\sigma_i(t)$ then.

**Outlier Detection** SPE score of $\overline{\mathbf{x}}(t)$ is calculated by (3.10). The estimated range of $SPE(t)$ is relative to the confidence level $(1 - \alpha)$ as

$$P\{|SPE(t) - \mu_{SPE}| \leq \xi_\alpha \cdot \sigma_{SPE}\} = 1 - \alpha, \tag{3.13}$$

since the SPE score follows Gaussian distribution. The value of $\xi_\alpha$ is set to 3 here, which means the SPE score falls in the range $[\mu_{SPE} - \xi_\alpha \cdot \sigma_{SPE},\ \mu_{SPE} + \xi_\alpha \cdot \sigma_{SPE}]$ with 99.7% confidence. It is still an adaptive threshold, since the mean value ($\mu_{SPE}$) and standard deviation ($\sigma_{SPE}$) of SPE score are kept updating with the newly collected sensor data.

If the SPE score ($SPE(t)$) is out of the estimated range, an outlier is detected and Algorithm 2 is called for further outlier diagnosis. Otherwise, parameters are updated with the newly collected sensor data.

**Outlier Diagnosis** The outlier diagnosis algorithm is further proposed, since the over-threshold SPE(t) in Algorithm 1 can only indicate the abnormal of $\overline{\mathbf{x}}(t)$. With Algorithm 2, the specific abnormal component $\overline{x}_i(t)$ is diagnosed. Calculate SPE score of each component in $\overline{\mathbf{x}}(t) = [\overline{x}_1(t), \overline{x}_2(t), \ldots, \overline{x}_k(t)]$ by

$$SPE_i(t) = \|\overline{x}_i(t) - \tilde{\mathbf{E}}_{l,i}\tilde{\mathbf{E}}_{l,i}^T\overline{x}_i(t)\|_2^2, \; i = 1, 2, \ldots, k, \tag{3.14}$$

where $k$ is the number of sensor nodes, and $\tilde{\mathbf{E}}_{l,i}$ is the corresponding row vector in eigenvector matrix $\tilde{\mathbf{E}}_l$. And then divide it by SPE(t),

$$\eta_i = SPE_i(t)/SPE(t), \; i = 1, 2, \ldots, k. \tag{3.15}$$

If the result $\eta_i$ is over a pre-defined weight $\xi$, the specific component $\overline{x}_i(t)$ is diagnosed as an outlier. Record the time and label of data outlier with $(t, i)$ for further statistical analysis. Sensor node that frequently generates outliers should be manually diagnosed.

**Parameters Update of SPE Score** $\mu_{SPE}$ and $\sigma_{SPE}$ are updated by (3.16) and (3.17),

$$\mu_{SPE}(t) = (1 - \beta) \cdot \mu_{SPE}(t - 1) + \beta \cdot SPE(t), \tag{3.16}$$

$$\sigma_{SPE}^2(t) = (1 - \beta) \cdot \sigma_{SPE}^2(t - 1) + \beta \cdot (SPE(t) - \mu_{SPE}(t))^2, \tag{3.17}$$

where $\beta = \frac{1}{t}$ is the forgetting factor.

**Eigenvectors and Eigenvalues Update** The covariance matrix of $\mathbf{X}$ at time instance $t$ is

$$\begin{aligned} \mathbf{C}_\mathbf{X}(t) &= \frac{1}{t-1}\sum_{j=1}^{t}\overline{\mathbf{x}}(j)\overline{\mathbf{x}}^T(j) \\ &= (1 - \varepsilon)\mathbf{C}_\mathbf{X}(t - 1) + \varepsilon \cdot \overline{\mathbf{x}}(t)\overline{\mathbf{x}}^T(t) \\ &= \mathbf{C}_\mathbf{X}(t - 1) + \varepsilon \cdot (\overline{\mathbf{x}}(t)\overline{\mathbf{x}}^T(t) - \mathbf{C}_\mathbf{X}(t - 1)), \end{aligned} \tag{3.18}$$

where $\varepsilon$ is the modifying factor and usually < 0.01.

The eigenvector decomposition of covariance matrix at time $t$ is given by

$$\begin{aligned}\mathbf{C_X}(t) &= \mathbf{E}(t)\mathbf{\Lambda}(t)\mathbf{E}^T(t)\\ &= \mathbf{E}(t-1)\mathbf{\Lambda}(t-1)\mathbf{E}^T(t-1) + \varepsilon(\overline{\mathbf{x}}(t)\overline{\mathbf{x}}^T(t)\\ &- \mathbf{E}(t-1)\mathbf{\Lambda}(t-1)\mathbf{E}^T(t-1))\\ &= \mathbf{E}(t-1)[(1-\varepsilon)\mathbf{\Lambda}(t-1)]\mathbf{E}^T(t-1) + \varepsilon\overline{\mathbf{x}}(t)\overline{\mathbf{x}}^T(t),\end{aligned} \tag{3.19}$$

let $\mathbf{A}(t) = \overline{\mathbf{x}}^T(t)\mathbf{E}(t-1)$, then

$$\mathbf{C_X}(t) = \mathbf{E}(t-1)[(1-\varepsilon)\mathbf{\Lambda}(t-1) + \varepsilon\mathbf{A}^T(t)\mathbf{A}(t)]\mathbf{E}^T(t-1). \tag{3.20}$$

It can be further decomposed as

$$(1-\varepsilon)\mathbf{\Lambda}(t-1) + \varepsilon\mathbf{A}^T(t)\mathbf{A}(t) = \mathbf{U}(t)\mathbf{\Sigma}(t)\mathbf{U}^T(t). \tag{3.21}$$

Therefore, the eigenvectors and eigenvalues are updated by

$$\mathbf{E}(t) = \mathbf{E}(t-1)\mathbf{U}(t), \tag{3.22}$$

$$\mathbf{\Lambda}(t) = \mathbf{\Sigma}(t). \tag{3.23}$$

Let

$$\mathbf{U}(t) = \mathbf{I} + \mathbf{H}_1(t), \tag{3.24}$$

$$\mathbf{\Sigma}(t) = (1-\varepsilon)\mathbf{\Lambda}(t-1) + \mathbf{H}_2(t). \tag{3.25}$$

Based on the first-order perturbation theory [58], the $\mathbf{H}_1$ and $\mathbf{H}_2$ are derived as

$$\mathbf{H}_1(t) = \begin{cases} 0 & ,i=j,\\ \frac{\varepsilon\mathbf{A}_i(t)\mathbf{A}_j(t)}{(1-\varepsilon)(\mathbf{\Lambda}_j^2(t-1)-\mathbf{\Lambda}_i^2(t-1))+\varepsilon(\mathbf{A}_j^2(t)-\mathbf{A}_i^2(t))} & ,i\neq j.\end{cases} \tag{3.26}$$

and

$$\mathbf{H}_2(t) = \begin{cases} \varepsilon\mathbf{A}_i^2(t) & ,i=j,\\ 0 & ,i\neq j.\end{cases} \tag{3.27}$$

Finally, $\mathbf{E}$ and $\mathbf{\Lambda}$ are updated by (3.22)-(3.27).

**Data Aggregation** After outlier detection and outlier diagnosis, the outlier-free sensor data are further aggregated and transmitted to the data center,

$$\mathbf{y}(t) = \tilde{\mathbf{E}}_l^T(t)\mathbf{x}(t), \tag{3.28}$$

where the dimension of sensor data is reduced from $k$ to $l$.

**Algorithm 1** R-PCA based Outlier Detection Algorithm

```
Initialization:
normalize $\mathbf{X} \Rightarrow \overline{\mathbf{X}} \sim \mathcal{N}(0, 1)$
calculate $\mathbf{E}$ and $\mathbf{\Lambda}$ of $\frac{1}{n-1}\overline{\mathbf{X}}\,\overline{\mathbf{X}}^T$
initialize $\mu_{SPE}$ and $\sigma_{SPE}$
Recursion:
update $\mu$, $\sigma$ and normalize $\mathbf{x}(t)$
rank $\mathbf{\Lambda}$, $\mathbf{E}$ and calculate the number of PCs, $l$
reduce $\tilde{\mathbf{E}} \rightarrow \tilde{\mathbf{E}}_l$ and $\tilde{\mathbf{\Lambda}} \rightarrow \tilde{\mathbf{\Lambda}}_l$
calculate $SPE(t)$
if $|SPE(t) - \mu_{SPE}| > \xi_\alpha \cdot \sigma_{SPE}$ then
    outlier detected
    call Algorithm 2
else
    update $\mathbf{E}$ and $\mathbf{\Lambda}$
    update $\mu_{SPE}$ and $\sigma_{SPE}$
end if
```

**Algorithm 2** Outlier Diagnosis Algorithm

```
for $i = 1 : k$ do
    calculate $SPE_i(t)$ and $\eta_i$
    if $\eta_i > \xi \cdot \sum_{j=1}^{k} \eta_j$ then
        outlier detected
        record outlier with time $t$ and label $i$
    end if
end for
```

### 3.4.3 Data Recovery at IoT Data Center

In initialization phase, the initialized eigenvector matrix $\mathbf{E}$ and eigenvalue matrix $\mathbf{\Lambda}$ are transmitted to IoT data center. The data center simultaneously updates the eigen matrices for accurate recovery of the aggregated sensor data,

$$\tilde{\mathbf{X}} = \tilde{\mathbf{E}}_l \mathbf{Y}. \tag{3.29}$$

In terms of the data outliers, each received outlier $(t, i)$ is recorded. Further investigations need to be conducted to analyze the possible reasons, to restore the abnormal behaviors.

## 3.5 Performance Evaluation

In this section, practical databases based simulations have been conducted to evaluate the performance of the proposed data analysis framework. Since the framework is proposed for data outlier detection and aggregation, the detection accuracy of data outlier and recovery accuracy of aggregated data are investigated in the first two subsections. Given the cluster-based structure of the proposed framework, the influence of different numbers of clusters within the network is further evaluated. The complexity analysis is done at the end.

### 3.5.1 Detection Accuracy of Data Outlier

Data outlier detection is one of the major functions of the data analysis framework. Thus, the accuracy of outlier detection is investigated in detail. More specifically, the practical databases and evaluation metrics used are first introduced. Afterwards, the detection accuracy of the proposed algorithm is compared with both univariate and multivariate outlier detection algorithms. The developed outlier detection threshold is compared with the conventional threshold as well.

#### 3.5.1.1 Databases & Metrics

In order to evaluate the outlier detection accuracy, two practical databases are used, namely, NDBC-TAO [41] and Intel Lab [42].

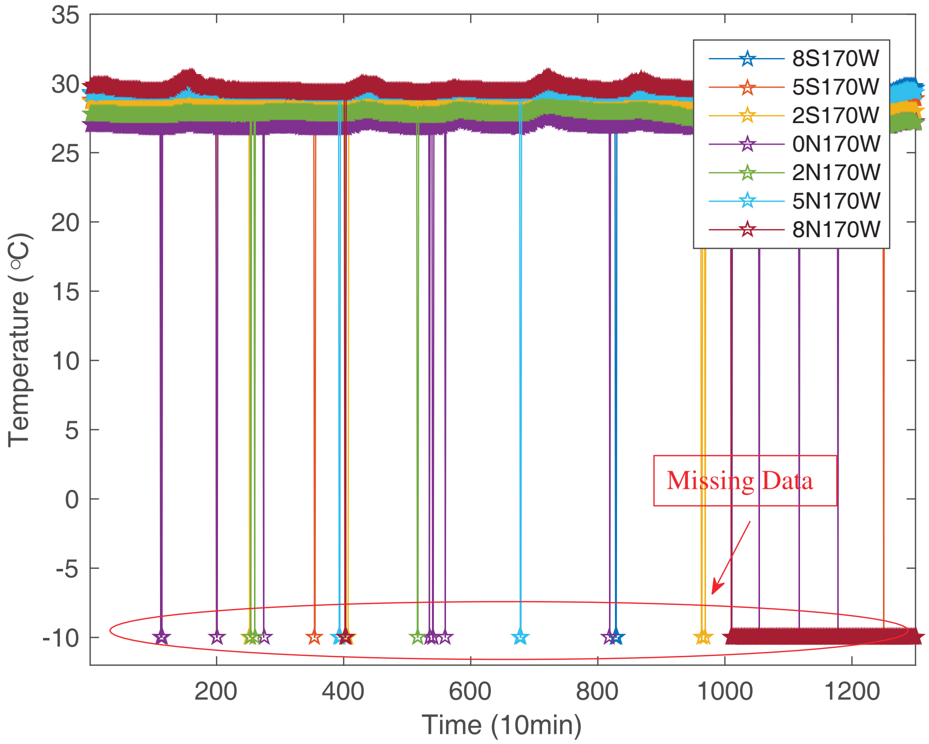


Figure 3.2: Sea surface temperature measurements collected from the NDBC-TAO project with random and continuous outliers.

**NDBC-TAO** In the NDBC-TAO project, meteorological and oceanographic sensors are deployed at the Pacific Ocean to monitor the climate. In the univariate test, sea surface temperature readings from 7 stations at 170W are used. The missing data in the original database are treated as outliers. The sensor readings including outliers are shown in Fig.3.2.

**Intel Lab** 54 sensor nodes are distributed in an indoor environment and sensor readings are regularly collected every 30s. In the multivariate test, temperature, humidity and voltage readings are used. Outliers are simulated by randomly generating missing data in the database. The randomly selected outliers are set to 0, and data outliers randomly occur with 3% to 30% times of the period.

In terms of the evaluation metrics, true positive rate (TPR) and false positive rate (FPR) are adopted, where TPR refers to the ratio of outliers successfully detected to the total number of outliers and FPR is the ratio of normal data mistakenly detected as abnormal to the total amount of normal data.

Table 3.1: [NDBC-TAO] Random Outlier Detection

| | True Positive Rate | False Positive Rate |
|---|---|---|
| DFD | 0.6296 | 0.0036 |
| iDFD | 1.0000 | 0.2813 |
| PCA | 1.0000 | 0.2162 |
| R-PCA | 1.0000 | 0.0386 |

Table 3.2: [NDBC-TAO] Mixed Outlier Detection

| | True Positive Rate | False Positive Rate |
|---|---|---|
| DFD | 0.0846 | 0.0030 |
| iDFD | 1.0000 | 0.3473 |
| PCA | 1.0000 | 0.2340 |
| R-PCA | 1.0000 | 0.0331 |

#### 3.5.1.2 Univariate Outlier Detection

To evaluate the performance of univariate outlier detection, R-PCA based algorithm is compared to the existing spatial correlation based algorithms, DFD [43], iDFD [59] and PCA-based algorithm with SPE score as outlier detection criterion [46].

As shown in Fig.3.2, in the NDBC-TAO database, there are two types of data outliers. One is the randomly missing data, which may be incurred by the occasionally failed communication. The other is the continuously missing data, which may be caused by the malfunctions of the sensor node. In the first test, we focus on the detection of randomly missing data by using the first half of the data in Fig.3.2. The test result is listed in Table 3.1. From Table 3.1, we can see that iDFD, PCA and R-PCA based algorithms all accurately detect the missing data in the NDBC-TAO database, while the FPR of R-PCA is the lowest. This result indicates that the R-PCA makes the fewest mistakes on the classification of normal and abnormal data. The reason is that the recursively updated transformation basis not only accurately captures the spatial correlation between sensor readings but also tracks the trend of data variations.

In another test, all the data shown in Fig.3.2 are included, where both randomly and continuously missing data exist. This test brings more challenges to the outlier detection algorithms. The test result is listed in Table 3.2. From Table 3.2, we can tell that the conventional DFD algorithm can hardly handle the continuously missing data detection, since the TPR is

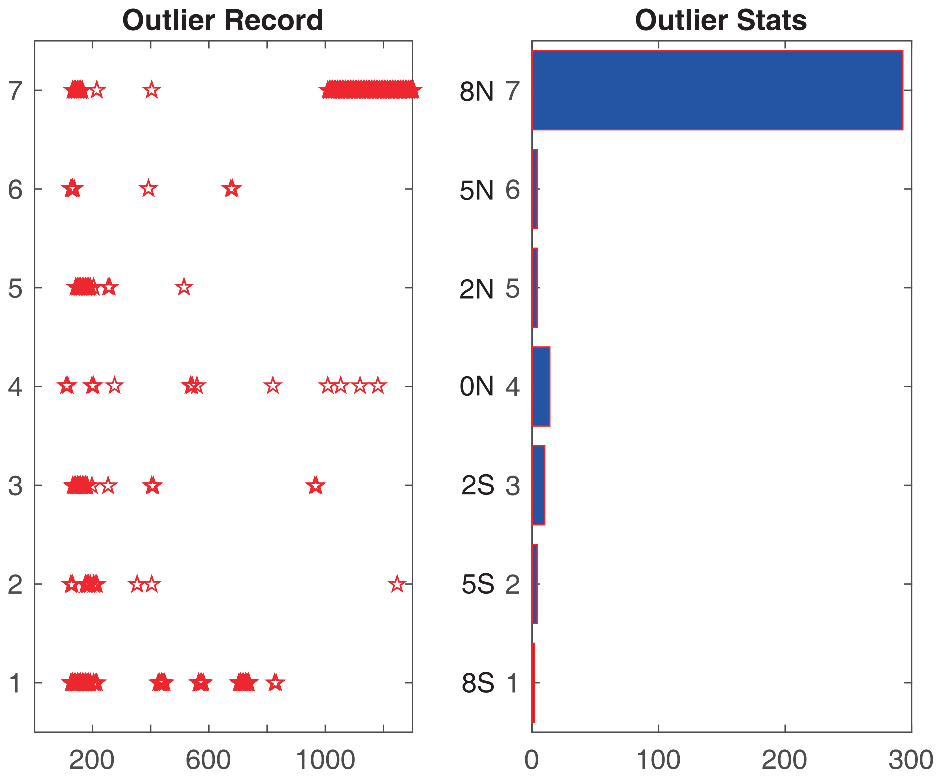


Figure 3.3: Statistical results of the univariate data outlier detection in the NDBC-TAO.

about 8.46%. All the other three algorithms detect the outliers accurately with 100% TPR. The R-PCA based algorithm still outperforms iDFD and PCA on FPR, with the existence of continuous data missing.

The time and label of detected data outliers and the statistical results of sensor nodes are plotted in Fig. 3.3. From the "outlier record", we can notice that the continuous data missing has occurred at sensor node "8N170W" since the 1000th time step. Besides, the number of outliers in the "outlier stats" at "8N170W" is much larger than other sensor nodes. Many possible reasons may lead to the lasting missing, like out of battery, malfunction in the communication module, which needs further manual restoration.

#### 3.5.1.3 Multivariate Outlier Detection

In the multivariate test, the outlier detection algorithm implemented based on sensor nodes locally is selected as benchmark [48], termed as conventional R-PCA based algorithm (CR-PCA). As compared to CR-PCA, the outlier detection algorithm in the proposed cluster-based

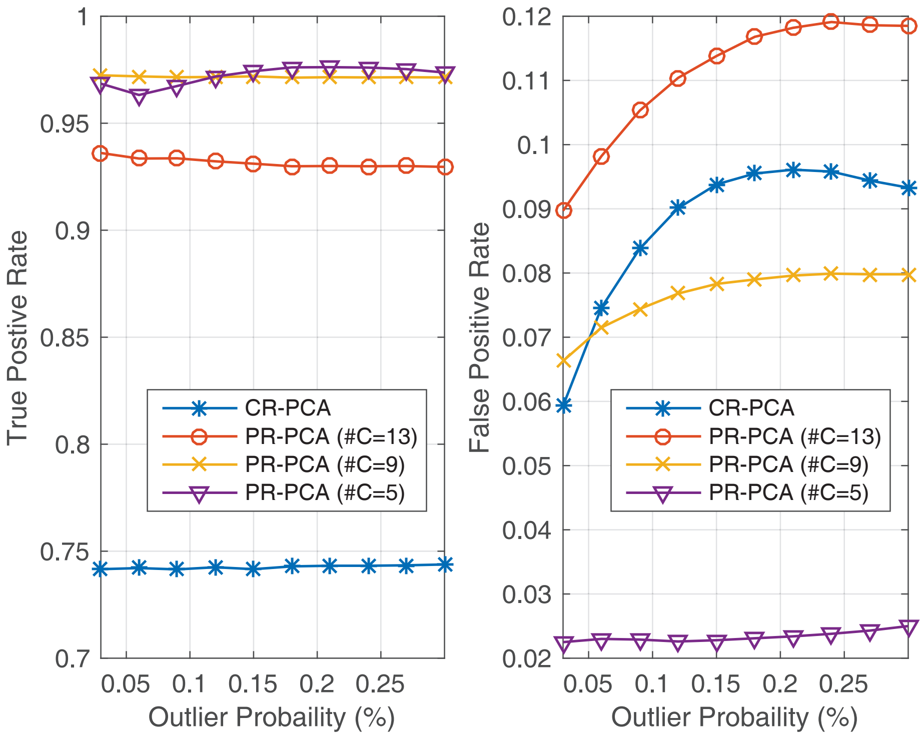


Figure 3.4: Comparison on the TPR (left) and FPR (right) between the CR-PCA and the PR-PCA with different numbers of clusters (#$C$) under different outlier probabilities.

data analysis framework is termed as the proposed R-PCA based algorithm (PR-PCA). Considering the PR-PCA is cluster-based, different numbers of clusters are considered. The comparisons on detection accuracy are shown in Fig.3.4.

From Fig.3.4, it can be seen that the TPR of PR-PCA outperforms that of CR-PCA. The performance is even better with fewer numbers of clusters. These results indicate that in the Intel database, the correlation between temperature, humidity and voltage readings of a sensor node, is not as strong as the spatial correlation between neighbor sensor readings. Besides, fewer numbers of clusters mean larger cluster size, which implies that more neighbors within a cluster improve the detection accuracy.

#### 3.5.1.4 Threshold

In Algorithm 1, a detection threshold of abnormal SPE score is proposed by exploitation of the mean and standard deviation values of SPE score. Here, the proposed threshold (termed as $\sigma$) is compared to the conventional detection threshold of SPE score (termed as $\delta^2$) [55]. The $\delta^2$

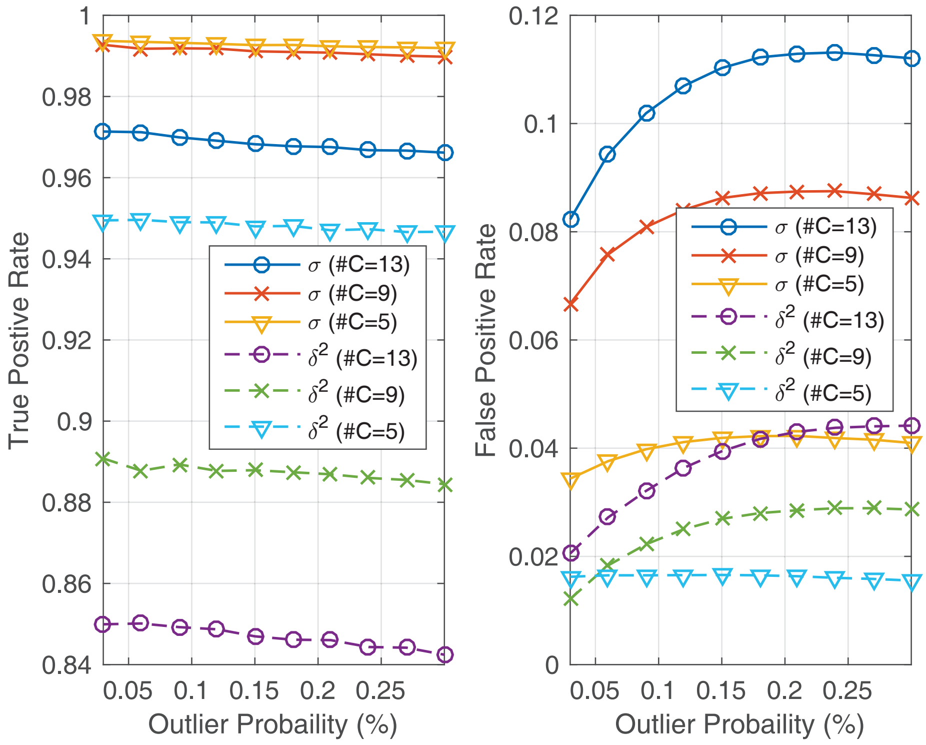


Figure 3.5: Comparison on the TPR (left) and FPR (right) between different detection thresholds of SPE score with different numbers of clusters (#$C$) under different outlier probabilities.

threshold is defined by

$$\delta_\alpha^2 = \theta_1\left(\frac{c_\alpha\sqrt{2\theta_2 h_0^2}}{\theta_1} + 1 + \frac{\theta_2 h_0(h_0-1)}{\theta_1^2}\right)^{1/h_0}, \tag{3.30}$$

where $c_\alpha$ is the coefficience for the Gaussian distribution with confidence level $(1-\alpha)$, and $h_0 = 1 - 2\theta_1\theta_3/3\theta_2^2$,

$$\theta_1 = \sum_{j=l+1}^{k} \tilde{\lambda}_j,\ \theta_2 = \sum_{j=l+1}^{k} \tilde{\lambda}_j^2,\ \theta_3 = \sum_{j=l+1}^{k} \tilde{\lambda}_j^3. \tag{3.31}$$

Specifically, SPE score is detected by

$$\begin{cases} if\ SPE > \delta_\alpha^2 & outlier, \\ if\ SPE \leq \delta_\alpha^2 & normal. \end{cases} \tag{3.32}$$

We assume that both detection thresholds work at the same confidence level 99.7%, namely, $\xi_\alpha$ and $c_\alpha$ are equal to 3. Comparisons are made between $\sigma$ and $\delta^2$ thresholds on outlier detection accuracy without outlier diagnosis (Algorithm 2), and results are shown in Fig.3.5. From Fig.3.5, it can be concluded that the proposed $\sigma$ detection threshold improves the TPR by about 9% as compared to the $\delta_\alpha^2$ threshold, while the FPR of the proposed $\sigma$ detection threshold is about 5% higher. Although each threshold has its advantage, the proposed $\sigma$ threshold is less complex in computation than $\delta^2$ threshold.

### 3.5.2 Recovery Accuracy of Aggregated Data

In the proposed data analysis framework, sensor data are aggregated by R-PCA at cluster heads, so that the network resources consumed by correlated and redundant sensor data transmissions can be reduced. The aggregated sensor data are finally recovered at the IoT data center. Recovery accuracy is a normally used metric to evaluate the quality of data aggregation algorithms. In this work, recovery accuracy is mathematically defined by *relative recovery error (rre)*, which is the relative difference between original and recovered data matrices. More specifically, as mentioned in Section 3.3, the reduced-dimensional matrix $\mathbf{Y}$ is generated by projecting $\overline{\mathbf{X}}$ into the subspace as $\mathbf{Y} = \tilde{\mathbf{E}}_l^T \overline{\mathbf{X}}$. At the IoT data center, the data matrix is then recovered by the inverse processing, *i.e.*, $\tilde{\mathbf{X}} = \tilde{\mathbf{E}}_l \mathbf{Y}$. So the rre is mathematically given by

$$rre = \frac{||\tilde{\mathbf{X}} - \overline{\mathbf{X}}||_2}{||\overline{\mathbf{X}}||_2}. \tag{3.33}$$

In this test, the sensor data provided by the Intel lab are used. Details of the database are explained in Subsection 3.5.1.1. The comparison between PR-PCA and the benchmark CR-PCA is shown in Fig.3.6, which demonstrates the instant *rre* along the timeline. The statistical values of Fig.3.6 are summarized in Table 3.3.

Table 3.3: Relative Recovery Error

| | *rre_mean* | *rre_min* | *rre_median* | *rre_max* |
|---|---|---|---|---|
| CR-PCA | 0.0995 | 0.0699 | 0.0989 | 0.1364 |
| PR-PCA | 0.0849 | 0.0313 | 0.0796 | 0.1883 |

From Fig.3.6, it can be seen that the *rre* curves of both CR-PCA and PR-PCA based algorithms fluctuate dramatically. But it can still tell that nearly 90% *rre* values of PR-PCA are below those of CR-PCA, which is proved by Table 3.3 as well. Table 3.3 indicates that the

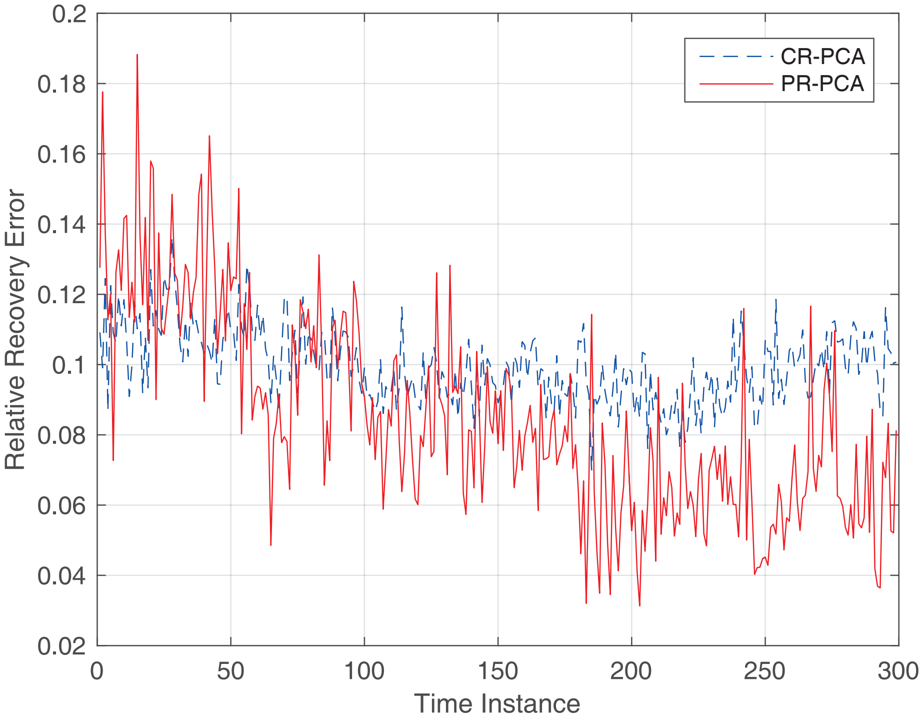


Figure 3.6: Comparison on the relative recovery error between the CR-PCA and the PR-PCA.

mean value of the PR-PCA based algorithm is about 15% smaller than that of CR-PCA based algorithm. This result further strengthens the conclusion in Subsection 3.5.1 that the correlation between temperature, humidity and voltage readings of a sensor node, is not as strong as the spatial correlation between neighbor sensor readings.

### 3.5.3 Discussion on the Number of Clusters

The data analysis framework is proposed based on cluster-tree topology. From Fig.3.4, we can notice that the performance of outlier detection varies with different numbers of clusters. Thus, in this subsection, we further investigate the influence of different numbers of clusters on data recovery accuracy and network energy consumption.

Data recovery accuracy is still evaluated by the *rre* (3.33). In terms of the network energy consumption, energy consumed by all sensor nodes and cluster heads are considered. Micaz mote is used as the node energy consumption model [54]. More specifically, the energy consumed by transmitting and receiving are 720 and 110 nJ/bit, respectively. Each CPU instruction costs 4 nJ/instruction. Given the data packet, we assume that the packet header is 56 bits, the

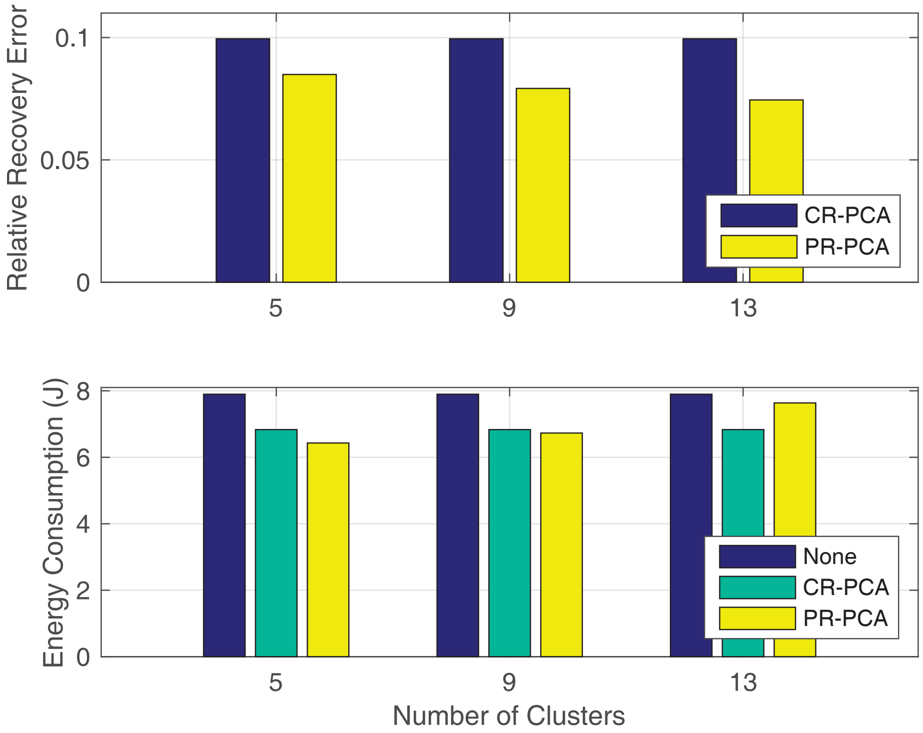


Figure 3.7: Comparison on the relative recovery error and the network energy consumption of the CR-PCA and the PR-PCA under different numbers of clusters.

preamble overhead is 160 bits and each component occupies 32 bits of payload. The network energy consumed by the baseline network without any data aggregation algorithm (None) and the network energy consumed by a network with CR-PCA are selected as benchmarks.

Fig.3.7 shows the influence of different numbers of clusters on *rre* and network energy consumption with different data aggregation algorithms.

From Fig.3.7, we can see that in terms of the number of clusters, there is a trade-off between the *rre* and network energy consumption. That is with the increment in the number of clusters, the *rre* decreases while the network energy consumption increases. The reason is that with more clusters, the average number of sensor nodes within a cluster decreases. With fewer sensor nodes in a cluster, the aggregation degree of sensor data decreases, which means a larger amount of sensor data is transmitted throughout the network. The network energy consumption increases as a result. Network energy consumption of PR-PCA based algorithm is even higher than that of CR-PCA when the 54 sensor nodes are clustered into 13 clusters. However, with a larger number of clusters, the *rre* decreases. This is because, with stricter clustering threshold, fewer sensor nodes are gathered into the same cluster, but the spatial correlation between sensor data generated by these sensor nodes is stronger, which further improves the recovery accuracy of aggregated data. In practical scenarios, the number of clusters is determined by the requirements of specific applications.

### 3.5.4 Complexity Analysis

In this work, R-PCA is exploited instead of PCA, to recursively update the parameters in the PCA model to track the changes in the IoT systems. Simultaneously, extra computations are introduced as a consequence. Given a data matrix $\mathbf{X}_{[k \times n]}$, the computational complexity of PCA is $O(k^3)$, which is dominated by the eigenvalue decomposition. In R-PCA, the complexity of the initialization phase is still $O(k^3)$. In terms of the recursion phase, a single round of recursion is not computational-complex as $O(k^3)$, since eigenvectors and eigenvalues are obtained by first-order perturbation theory based update instead of eigenvalue decomposition. But the $t$ times of recursion brings extra time complexity of $O(k^2 t)$.

The mathematical calculations in R-PCA, including matrix multiplication, eigenvalue decomposition, and the sorting, are affordable for a full-functioned CPU. However, given the weak computational capacity of the sensor node, the calculations are quite heavy burdens. The previous algorithms with adaptively updated PCA [48, 54] loaded on sensor nodes can lead to long computational delay and fast battery draining. The cluster-based data analysis framework proposed in this work offloads the complex calculations from sensor nodes to cluster heads and the IoT data center, which relieves the heavy burden on sensor nodes.

## 3.6 Chapter Summary

In this chapter, a cluster-based data analysis framework using R-PCA to overcome the challenges of data outlier detection and redundant sensor data aggregation in the IoT systems has been proposed. More specifically, sensor nodes are gathered into clusters, and all sensor data are transmitted to cluster heads. At a cluster head, spatially correlated sensor data are diagnosed and aggregated by the R-PCA based algorithms, and then the outlier-free and aggregated data are forwarded to the IoT data center. All the aggregated data are recovered and outliers are recorded for further analysis at the data center. With the development of R-PCA, the parameters of the PCA model are able to be recursively updated in real-time, which finally improves the performance of the data analysis framework. The cluster-based framework also relieves the computational burden on sensor nodes. Simulation results prove that the proposed data analysis framework precisely aggregates sensor data with high recovery accuracy. Besides, the outlier detection accuracy is also improved compared to other existing data outlier detection algorithms.

# Chapter 4

# A Novel Edge Computing Enabled Temporal Data Reduction Scheme in IoT Systems

The recent advancement of IoT technologies has enabled many emerging applications. These advanced applications generate a massive amount of data at the edge of IoT networks, which usually need to be relayed to the cloud for data analytics. However, uploading all these IoT data to the cloud platform imposes a heavy burden on the underlying network. The unavoidable long delay from data exchange and processing significantly reduces the time-responsiveness of real-time IoT applications. Thus, edge computing has been introduced to IoT applications as an intermediate between end devices and cloud for primary IoT data processing. In this chapter, a temporal IoT data reduction scheme through edge computing is proposed to reduce the total amount of IoT data uploaded to the cloud. More specifically, IoT data are firstly modeled as multivariate normal distribution by the cloud. Dual Kalman filters (KFs) with identical parameters are then deployed at both the edge and cloud platforms. The same predictions are simultaneously triggered by the dual KFs at both platforms. Only the measured IoT data out of the predicted range is further uploaded from edge to cloud. Otherwise, predicted values at both platforms are used instead of measurements. A simple prototype IoT system is developed for performance evaluation. Experimental results indicate that the proposed scheme significantly reduces the number of packets uploaded to the cloud platform with high data accuracy.

## 4.1 Introduction

With the rapid development of sensing and communication technologies, IoT can interconnect not only people but also physical objects and virtual processes, which creates many emerging applications including smart healthcare, smart home, and smart cities [60, 61]. The real-time data processing nature of these advanced applications impose new challenges on current IoT architectures, particularly in meeting the latency, privacy and bandwidth requirements of IoT applications [10]. For instance, a smart healthcare system requires high privacy and security of user data, real-time response to the emergency situation and high bandwidth for uploading the massive amount of daily monitoring sensor data. The current IoT architecture cannot suffice these requirements, due to the limitation of bandwidth, unavoidable long delay and high cost for massive data uploading. Due to these challenges, edge computing has been proposed as a complement, and a number of institutions have made joint efforts on its development.

Edge computing can be considered as an extension of cloud computing from the core network to edge network, which generally provides services of data computing, storage, and analysis between end devices and traditional cloud computing platform [62]. In terms of the aforementioned challenges, primary data processing at edge computing platform can provide timely response to end devices at local networks, and also reduce the amount of data uploaded to the cloud platform so as to save the network bandwidth, as shown in Fig.4.1. As summarized in [35], the advantages of edge computing are highlighted by better cognition, high efficiency, high agility, and low latency, since edge devices can be flexibly located to provide its services.

There are a few recent studies focused on edge computing enabled data processing frameworks in IoT systems. A smart data structure for big data management deployed on edge devices has been proposed in [63]. Similarly, a pre-cloud data processing module was proposed in [64], which acted as an edge engine. The engine supported several data processing units, including data collection, storage, mining, *etc.* The proper location of the engine in the IoT architecture was investigated as well. However, most of the existing works, like [63, 64], are still focused on the general frameworks of edge computing enabled data processing in IoT systems without further theoretical analysis, operation process design or performance verification. Based on a commonly used edge-enabled IoT system framework, a data reduction scheme is

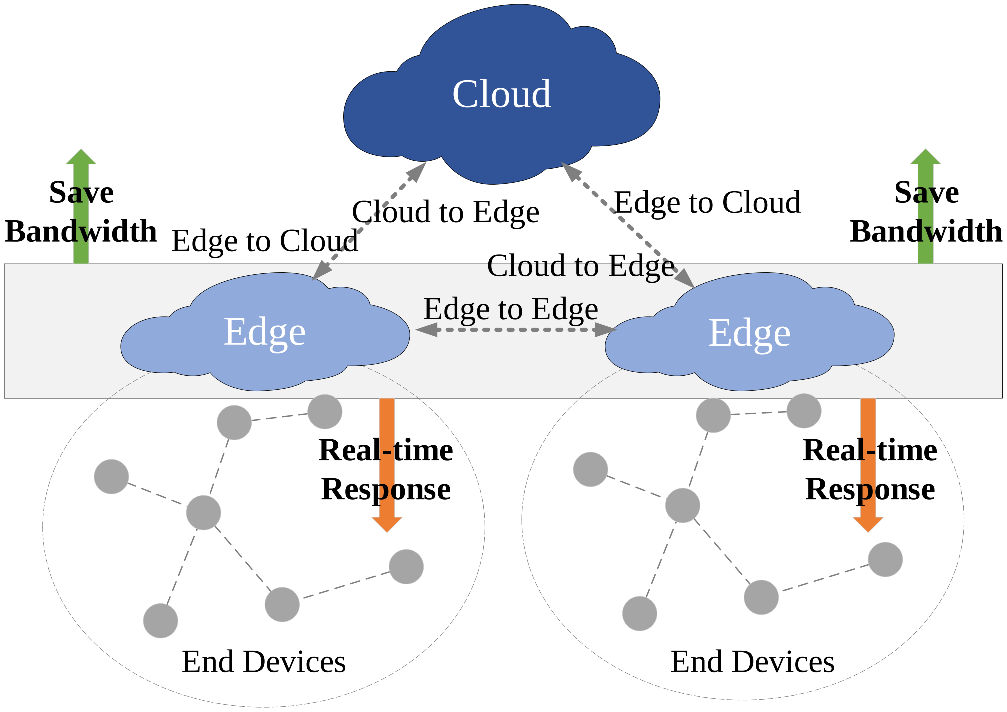


Figure 4.1: An edge computing enabled IoT system architecture.

designed based on the temporal correlation between time-series IoT data, and an experimental platform is developed for performance evaluation.

In this chapter, a novel temporal data reduction scheme is proposed by the exploitation of the primary data processing at edge computing platform, so that the total amount of IoT data to be uploaded could be minimized. More specifically, IoT data are firstly modeled as multivariate normal distribution by the cloud. Dual KFs with identical parameters are then deployed at both the edge and cloud platforms. The same predictions for the data measured by end devices are simultaneously triggered by the dual KFs at both platforms, based on historical data and intrinsic temporal data correlation. Only the measured data out of the predicted range is further uploaded from edge to cloud. Otherwise, predicted values at both ends are used instead of measurements. A simple prototype IoT system is developed for experimental evaluation, which consists of end devices, edge devices, and the cloud platform. Experimental results indicate that the proposed scheme minimizes the number of packets uploaded to the cloud platform with identical data accuracy, as compared to the benchmarks, GM and ARMA.

The rest of this chapter is organized as follows. The model of the edge computing enabled

IoT system is presented in Section 4.2. In Section 4.3, the novel temporal data reduction scheme is further proposed. Practical experiments based performance evaluation is investigated in Section 4.4. Finally, all the contributions are summarized in Section 4.5.

## 4.2 System Model

As shown in Fig.4.1, an edge computing enabled IoT system architecture normally consists of the following functional blocks, end devices, edge computing devices, and the cloud computing platform, and the interactive interfaces between these blocks [35].

• *End devices* are the most fundamental and important component in IoT systems. Generally, IoT end devices are equipped with sensing and communication modules, such as wireless sensor nodes and smartphones. Due to the weakness in computational capacity, IoT end devices are mainly responsible for sensing and reporting the physical surroundings. In this work, wireless sensor nodes are regarded as the IoT end devices, which periodically transmit the physical measurements to edge platform. Sensor data sampled from $n$ sensor nodes at time instance $t$ are marked as $\mathbf{X}_{\{1,\ldots,n\},t} = [x_{1,t}, x_{2,t}, \ldots, x_{n,t}]^T$.

• *Edge computing devices* are located at the network edge and act as an intermediate between end devices and the remote cloud platform, as shown in Fig.4.1. Since the edge is enabled for primary data collection, storage and processing, it can not only provide real-time response to end devices but also save the bandwidth by reducing the amount of data uploaded to the cloud [65]. In this work, dual KFs with identical parameters are set up at both the edge and cloud ends. The same predictions are simultaneously triggered at both ends. Instead of uploading all the IoT data indiscriminately, only the data out of the predicted confidence interval are uploaded from edge to cloud for further analysis. In most cases, predicted values are used at both ends.

• *Cloud computing platform* plays the role of a remote data center, which is generally responsible for complex processing and analysis of IoT data. Although the cloud meets the challenges of bandwidth limitation and high latency, it still cannot be substituted given its strong computational capacity. In the cloud, IoT data are modeled as multivariate normal distribution and kept update by KFs, which makes data query and data management easier.

## 4.3 A Novel Edge Computing Enabled Temporal Data Reduction Scheme

In this section, the novel edge computing enabled temporal data reduction scheme is proposed, which consists of three major phases. Firstly, sensor data are modeled as multivariate normal (MVN) distribution based on historical data in the cloud. Afterwards, Kalman filters with identical parameters are deployed at both edge and cloud, which simultaneously predict and update the mean vector and covariance matrix of the MVN model. In the running phase, the confidence interval of the future data measured by sensor nodes is estimated by the predicted mean vector and covariance matrix. Only the measured data out of predicted interval is further uploaded from edge to cloud. Otherwise, predicted values are used on both platforms. More details on these three phases are explained as follows.

**Phase I** In phase I, all the measured sensor data are uploaded to cloud through edge devices, and modeled as multivariate normal distribution at cloud based on the normality analysis of collected sensor data. Suppose that $m$ samples are collected from $n$ sensor nodes, and then these data are modeled as a data matrix $\mathbf{X}$ with $n \times m$ dimensions at cloud, *e.g.*, $\mathbf{X}_{i,j}(i \in [1, n], j \in [1, m])$ is the $j$th sensor reading from node $i$.

However, $n$ can be huge for a large-scale IoT system. It is inefficient and meaningless to set up an MVN model with $n$ variables, so several downsized MVN models with $k$ variables are built instead. The value of $k$ is determined by the multivariate normality test. In our work, $k$ is the largest value that can make the submatrix $\mathbf{X}'_{[k\times m]}(k < n)$ pass the Mardia's normality test [66]. Specifically, Mardia's skewness and kurtosis of submatrix $\mathbf{X}'_{[k\times m]}(k < n)$ and corresponding p-values are calculated. The skewness is calculated as

$$S = \frac{1}{6m}\sum_{i=1}^{m}\sum_{j=1}^{m}\left[(\mathbf{x}_i - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_j - \boldsymbol{\mu})\right]^3, \tag{4.1}$$

where $\mathbf{x}_{i_{[k\times 1]}}$ is the $i$th column vector of $\mathbf{X}'$, $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix of $\mathbf{X}'$,

$$\boldsymbol{\Sigma} = \frac{1}{m-1}\sum_{i=1}^{m}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T. \tag{4.2}$$

The corresponding p-value is given by

$$p_S = 1 - \mathcal{F}_{\chi^2}(S, v), \tag{4.3}$$

where $\mathcal{F}_{\chi^2}$ is the cumulative distribution function (*CDF*) of chi-square distribution, and $v$ is the degrees of freedom,

$$v = \frac{1}{6}k(k+1)(k+2). \tag{4.4}$$

Mardia's kurtosis is calculated as

$$K = \sqrt{\frac{m}{8k(k+2)}} \left\{ \frac{1}{m} \sum_{i=1}^{m} \left[ (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) \right]^2 - k(k+2) \right\}, \tag{4.5}$$

and p-value of kurtosis is given by

$$p_K = 2 \times (1 - \Phi(|K|)), \tag{4.6}$$

where $\Phi$ is the *CDF* of standard normal distribution.

If p-values of the skewness and kurtosis, $p_S$, $p_K$, are larger than a certain significant level, then the $k$-variate $\mathbf{X}'$ is considered to be following the multivariate normal distribution. The probability density function $f(x)$ is given by

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}_0|^{1/2}} exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1} (\mathbf{x} - \boldsymbol{\mu}_0)), \tag{4.7}$$

where $\boldsymbol{\mu}_0 \in \Re^k$ and $\boldsymbol{\Sigma}_0 \in \mathbf{S}_{++}^k$ are the initial mean vector and covariance matrix of the multivariate normal distribution, which are calculated by the historical sensor data.

**Phase II** In phase II, Kalman filters with identical parameters are simultaneously built at both the edge and cloud platforms, in order to keep predicting and updating the mean vector and covariance matrix of the MVN model built in phase I. Maintaining a Kalman filter consists of two major procedures: prediction and update. In the prediction procedure, mean vector at time $t$ ($\hat{\boldsymbol{\mu}}_{t|t-1}$) is estimated by

$$\hat{\boldsymbol{\mu}}_{t|t-1} = \mathbf{F}_t \hat{\boldsymbol{\mu}}_{t-1|t-1} + \mathbf{B}_t \mathbf{U}_t, \tag{4.8}$$

and its corresponding covariance ($\mathbf{P}_{t|t-1}$) is calculated as

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^T + \mathbf{Q}_t, \tag{4.9}$$

where $\mathbf{F}_t$ is the state transition model, $\mathbf{B}_t$ is the control-input model applied to the control vector $\mathbf{U}_t$, and $\mathbf{Q}_t$ is the covariance of white Gaussian noise in prediction procedure. Based on (4.8) (4.9), the optimal Kalman gain is calculated as

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t^T (\mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^T + \mathbf{R}_t)^{-1}, \tag{4.10}$$

where $\mathbf{H}_t$ is the observation model and $\mathbf{R}_t$ is the covariance of observation noise. Accordingly, $\hat{\boldsymbol{\mu}}_{t|t}$ and $\mathbf{P}_{t|t}$ are updated with the optimal Kalman gain ($\mathbf{K}_t$) and the mean vector of measured data ($\boldsymbol{\mu}_t$) by

$$\hat{\boldsymbol{\mu}}_{t|t} = \hat{\boldsymbol{\mu}}_{t|t-1} + \mathbf{K}_t (\boldsymbol{\mu}_t - \mathbf{H}_t \hat{\boldsymbol{\mu}}_{t|t-1}), \tag{4.11}$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_{t|t-1}. \tag{4.12}$$

Simultaneously, the covariance matrix $\boldsymbol{\Sigma}$ is kept predicting and updating in the same way. Namely, both mean vector and covariance matrix are predicted by the values from previous time instance, and are updated after receiving newly measured data. However, the specific values used to update the model are determined by the edge computing platform in Phase III.

**Phase III** In phase III, the newly measured sensor data, $\mathbf{x}_t$, is transmitted to edge first. Based on the mean vector and covariance matrix predicted by Kalman filters, $\mathcal{N}(\boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})$, the confidence interval of $\mathbf{x}_t$ is further derived at edge. The edge device determines whether the received $\mathbf{x}_t$ falls in the confidence interval. Only when $\mathbf{x}_t$ is out of the predicted interval, it would be further uploaded to the cloud.

Given a MVN model, the confidence interval $\mathbf{C}$ is determined by

$$\mathbf{C} = \hat{\boldsymbol{\mu}}_{t|t-1} \pm \sqrt{\chi_k^2(\alpha)\lambda_i}\mathbf{e}_i, \; i = 1, \ldots, k, \tag{4.13}$$

where $\chi_k^2(\alpha)$ is the chi-square distribution with $k$ degrees of freedom and $1-\alpha$ confidence level. $\hat{\boldsymbol{\mu}}_{t|t-1}$ is the predicted mean vector. $\lambda_i$ and $\mathbf{e}_i$ are the eigenvalue and eigenvector of the predicted covariance matrix $\hat{\boldsymbol{\Sigma}}_{t|t-1}$,

$$\hat{\boldsymbol{\Sigma}}_{t|t-1} = \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T. \tag{4.14}$$

In order to simplify the determination, $\mathbf{x}_t$ is transformed from the original coordinate system into the coordinate system defined by eigenvectors, namely,

$$\tilde{\mathbf{x}}_t = \mathbf{E}^T(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{t|t-1}). \tag{4.15}$$

Correspondingly, the bounds of confidence interval in the new coordinate system are

$$\mathbf{C} = (\underbrace{0,\ldots,0}_{i-1}, \pm\sqrt{\chi_k^2(\alpha)\lambda_i}, \underbrace{0,\ldots,0}_{k-i}), i = 1,\ldots,k. \tag{4.16}$$

If $\mathbf{x}_t$ falls in the confidence interval, Kalman filters are updated with the predicted values $(\hat{\boldsymbol{\mu}}_{t|t-1}, \hat{\boldsymbol{\Sigma}}_{t|t-1})$ at both the edge and cloud platforms, and error caused by the substitution is calculated. Otherwise, $\mathbf{x}_t$ is forwarded to the cloud platform, and Kalman filters at both the edge and cloud platforms are updated with $\mathbf{x}_t$ in the meantime. The pseudocode of all the three phases is summarized in Algorithm 3.

## 4.4 Performance Evaluation

In the first part of this section, multivariate normality of sensor data is analyzed using the practical database provided by Intel lab [67]. Afterwards, a simple prototype IoT system is developed for the performance evaluation of the proposed temporal data reduction scheme.

### 4.4.1 Multivariate Normality Analysis

Mardia's skewness and kurtosis (4.1)-(4.6) are used to test the multivariate normality of sensor data [66]. In order to evaluate the effect of the number of variables on normality, p-values of skewness and kurtosis are calculated with temperature readings from different numbers of sensor nodes, where the sensor data are provided by Intel lab. Significant level is set to 0.05

**Algorithm 3** A Novel Edge Computing Enabled Temporal Data Reduction Scheme

```
**PHASE I:** (*In Cloud, Historical Data*)
set up $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$
**PHASE II:** (*At Edge and Cloud, Historical Data*)
set up KFs with trained $\mathbf{F}_t$s and $\mathbf{H}_t$s
**PHASE III:** (*At Edge, End Devices* → *Edge*, $\mathbf{x}_t$)
predict by KFs, $\hat{\boldsymbol{\mu}}_{t|t-1}$ and $\hat{\boldsymbol{\Sigma}}_{t|t-1}$
calculate confidence interval $\mathbf{C}$ of $\mathcal{N}(\hat{\boldsymbol{\mu}}_{t|t-1}, \hat{\boldsymbol{\Sigma}}_{t|t-1})$
$\hat{\boldsymbol{\Sigma}}_{t|t-1} = \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^T$
$\mathbf{C} = \hat{\boldsymbol{\mu}}_{t|t-1} \pm \sqrt{\chi_k^2(\alpha)\lambda_i}\mathbf{e}_i, \ i = 1, \ldots, k$
**if** $\mathbf{x}_t \notin$ bounds $\mathbf{C}$ **then**
    edge → cloud, $\mathbf{x}_t$
    update models with $\mathbf{x}_t$
**else**
    update models with $\hat{\boldsymbol{\mu}}_{t|t-1}$ and $\hat{\boldsymbol{\Sigma}}_{t|t-1}$
    calculate error
**end if**
```

Table 4.1: P-Values of Skewness and Kurtosis

| $k$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $p_S$ | 0.4948 | 0.2704 | 0.1343 | 0.0882 | 0.0415 |
| $p_K$ | 0.2185 | 0.2611 | 0.2998 | 0.2802 | 0.2702 |
| $k$ | 6 | 7 | 8 | 9 | 10 |
| $p_S$ | 0.0143 | 0.0095 | 0.0036 | 0.0057 | 0.0105 |
| $p_K$ | 0.2678 | 0.2768 | 0.2925 | 0.2640 | 0.2503 |

without losing the generality. P-values ($p_S$ and $p_K$) are listed in Table 4.1. From Table 4.1, it can be seen that $p_S$ decreases with the increment in the number of variables, and it cannot even pass the test when the number of variables is larger than 4.

For better visualization of the multivariate normality, chi-square quantile-quantile (Q-Q) plot is introduced here. If $k$-variate sensor data follow the multivariate normal distribution, then the squared Mahalanobis distance of sensor data follows the chi-square distribution with $k$ degrees of freedom, where the squared Mahalanobis distance is calculated as

$$d_i^2 = (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}), \ i = 1, \ldots, m. \tag{4.17}$$

If $d^2 \sim \chi_k^2$, the chi-square Q-Q plot should be approximately a straight line through the

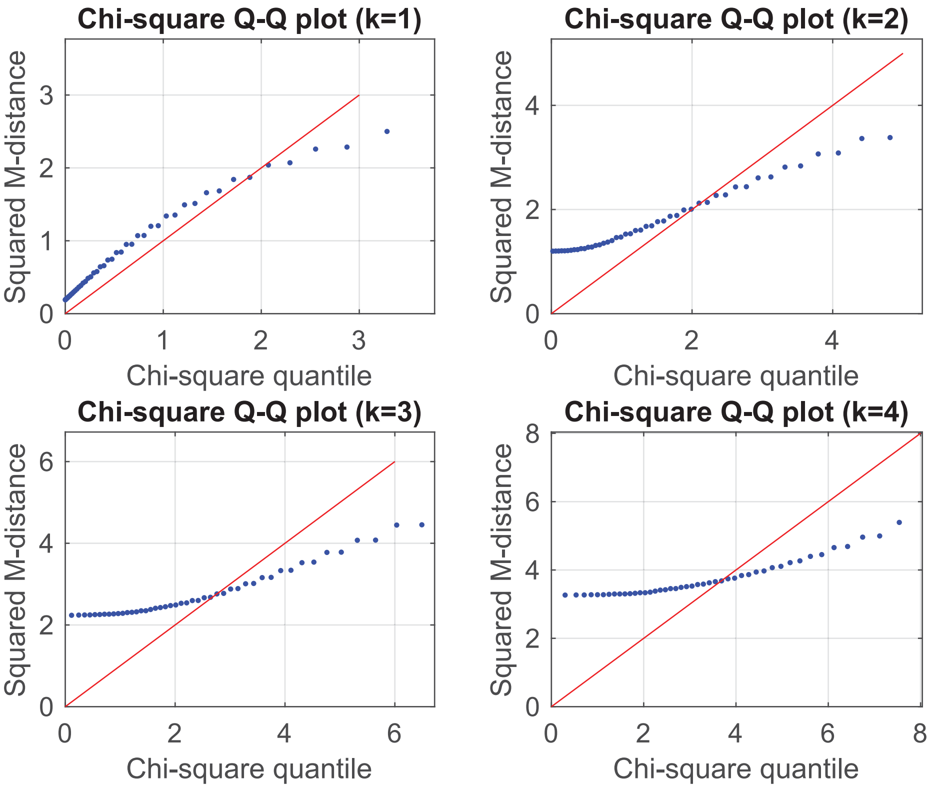


Figure 4.2: Chi-square Q-Q plot generated by the squared Mahalanobis distance among data from different numbers of sensor nodes (k=1,2,3,4).

origin with slope 1. The Q-Q plots of $k = 1, 2, 3, 4$ are shown in Fig.4.2. It can be visualized that the slope deviates from 1 more dramatically with the increment in the number of variables, which matches the trend of p-values.

## 4.4.2 Experimental Evaluation

### 4.4.2.1 Experimental Platform Setup

A simple IoT system prototype is developed, which consists of four sensor nodes (end devices), a gateway (edge device) and the Google BigQuery database (cloud platform). Each sensor node embeds temperature (TMP36), humidity (HIH5031) and brightness (TEMT6000) sensors, while XBee-S2 is used as the ZigBee RF module. One Raspberry Pi gateway is deployed as the edge device [68], and the Google BigQuery database acts as the cloud platform [69].

Four sensor nodes are randomly deployed in an indoor room. Sensor data are sampled every 30s, transmitted to Raspberry Pi for primary analysis and later uploaded to Google BigQuery.

#### 4.4.2.2 Evaluation Metrics

In the experiments, two evaluation metrics are considered. One is the number of packets uploaded from edge to cloud, since the aim of our proposed scheme is to reduce the data transmission. The other is the mean squared error (MSE) caused by downsized data transmission,

$$\varepsilon = \frac{1}{q}\sum_{i=1}^{q} ||\hat{\mathbf{x}}_i - \mathbf{x}_i||_2^2. \tag{4.18}$$

#### 4.4.2.3 Analysis of Confidence Level

The effect of different confidence levels $(1 - \alpha)$ and different numbers of variables $(k)$ on the performance of the proposed scheme are shown in Fig.4.3. From Fig.4.3, we can see that the number of packets uploaded reduces with the increment in confidence level. This is due to that the increment in confidence level enlarges the confidence interval, which results in more measured data falling in the confidence interval and fewer data transmissions. Since less measured data are uploaded, the models at both the edge and cloud platforms have fewer opportunities to be updated with the measured data, which finally makes the MSE of data increase. Furthermore, with the increment in the number of variables $(k)$, the number of packets uploaded reduces and the error increases correspondingly.

#### 4.4.2.4 Comparisons with GM and ARMA

GM (grey model) and ARMA (autoregressive and moving average) are two normally used prediction models for time series and introduced in this experiment as benchmark methods. Since the confidence interval used in the proposed scheme is not applicable in GM and ARMA, the bias of measured data from predicted value (threshold $\xi$, $|\mathbf{x}_t - \hat{\mathbf{x}}_t| \in [\mathbf{0}, \xi * \hat{\mathbf{x}}_t]$) is used instead as the transmission threshold. Here, $k$ is set to 4.

The experimental results are shown in Fig.4.4 and Fig.4.5, where Fig.4.4 shows the comparison on the number of packets uploading and Fig.4.5 is on the MSE caused by downsized data transmission. From these two figures, we can see that with the increment in the threshold, the number of packets uploaded decreases and the MSE increases, which are in the same trends as the results in Fig.4.3 and reasons behind these trends are identical.

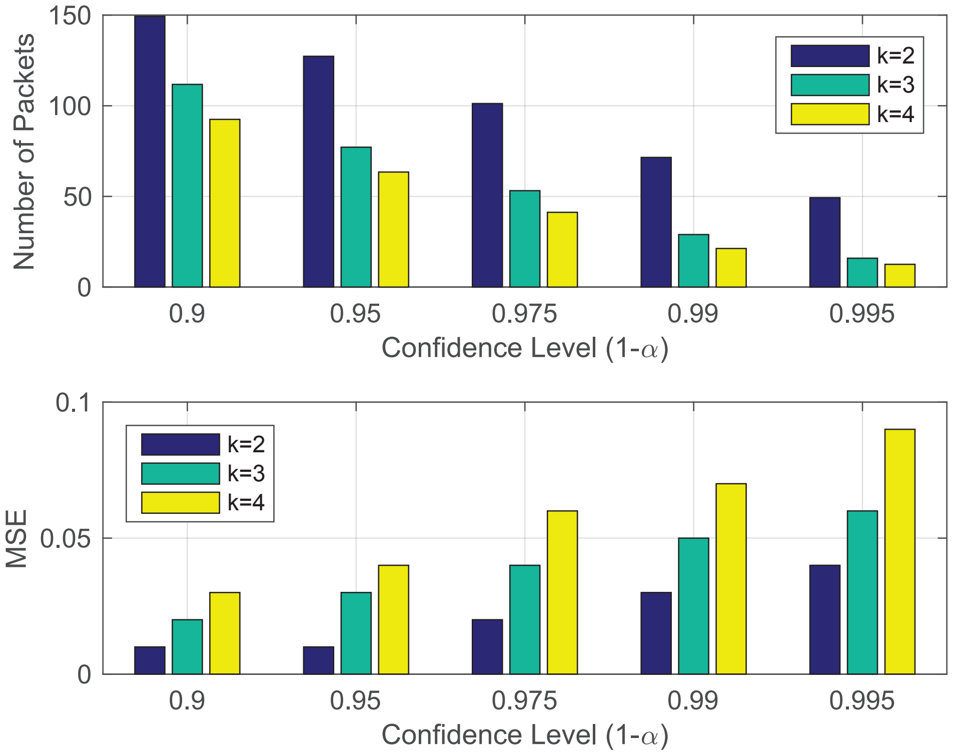


Figure 4.3: The effect of the confidence level $(1 - \alpha)$ on the number of packets uploading to the cloud and the mean squared error (MSE).

In terms of the comparisons between the proposed scheme (marked as MVN) and the benchmark methods (GM and ARMA), with the application of MVN, the number of packets uploading is 77.9% and 97% fewer than GM and ARMA, but the MSE generated correspondingly is 50% and 12.5% higher. However, the mean squared error generated by MVN is the lowest in the extreme case that no packets are uploaded from edge to cloud and all the sensor data recorded at cloud are the values predicted by the models, which is 92.8% and 95.5% lower than GM and ARMA respectively. Given the joint analysis of the number of packets uploaded and the mean square error, the proposed scheme outperforms the benchmark methods.

## 4.5 Chapter Summary

In this chapter, a novel edge computing enabled temporal data reduction scheme has been proposed, which has significantly reduced the number of packets uploaded to the cloud platform in current IoT systems. More specifically, sensor data are firstly modeled as multivariate normal

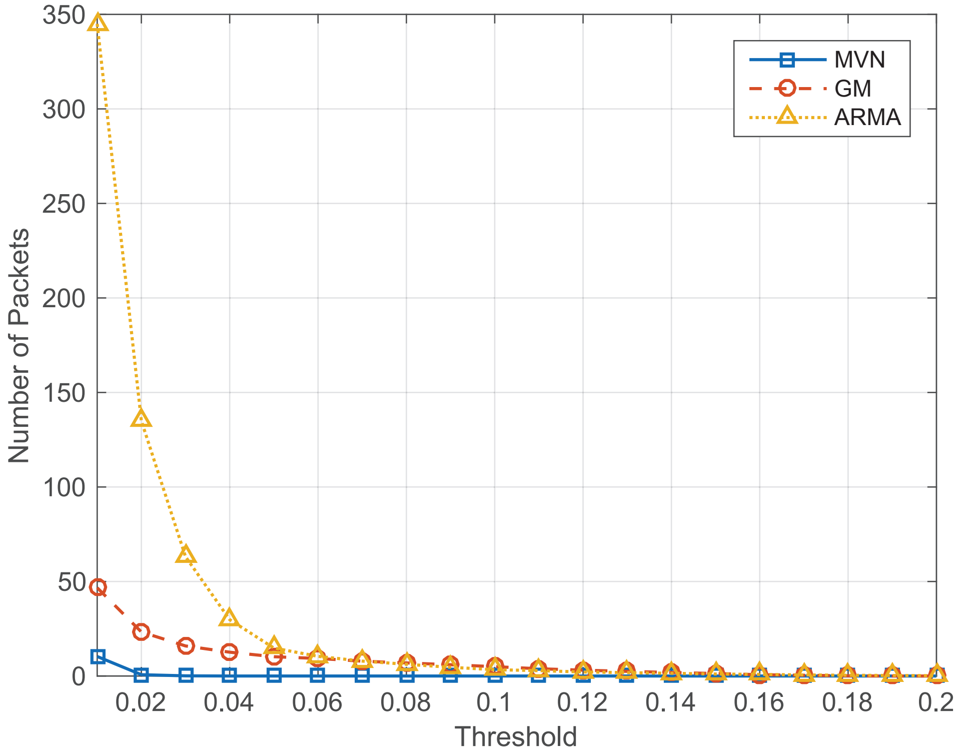


Figure 4.4: Comparison on the number of packets uploading between the proposed MVN-based scheme and the benchmark methods (GM and ARMA).

distribution by the cloud. Dual KFs with identical parameters are then deployed at both the edge and cloud platforms for prediction. The same predictions for the future data measured by end devices are simultaneously triggered at both platforms. Only the measured data out of the predicted range is further uploaded to the cloud. Otherwise, predicted values at both platforms are used instead of measurements. In order to evaluate the performance, a simple prototype IoT system is developed and GM and ARMA are selected as benchmark methods. Experimental results indicate that the proposed scheme significantly reduces the number of packets uploaded to the cloud platform with high data accuracy.

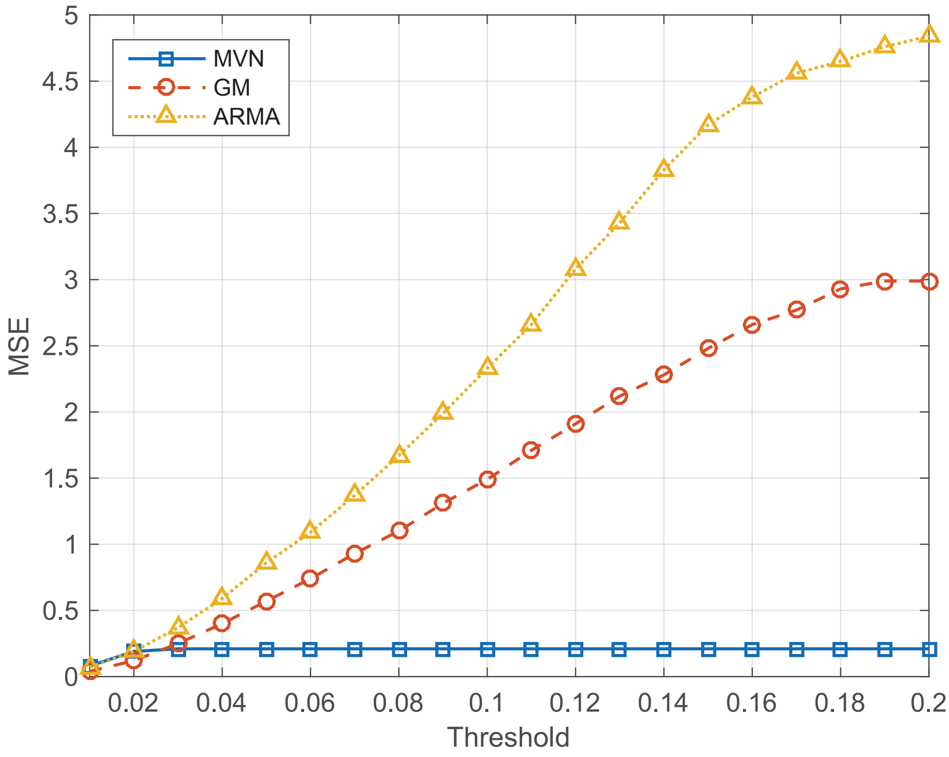


Figure 4.5: Comparison on MSE between the proposed MVN-based scheme and the benchmark methods (GM and ARMA).

# Chapter 5

# Cloud-Orchestrated Physical Topology Discovery of Large-Scale IoT Systems Using UAVs

WSNs have been rapidly integrated into IoT systems, empowering rich and diverse applications such as large-scale environmental monitoring. However, due to the random deployment of sensor nodes, the physical topology of the WSNs cannot be controlled and typically remains unknown to the IoT cloud server. Therefore, in order to derive the physical topology at the cloud for effective real-time event detection, a cloud-orchestrated physical topology discovery scheme for large-scale IoT systems using UAVs is proposed in this chapter. More specifically, the large-scale monitoring area is firstly split into a number of subregions for UAV-enabled data collection. Within the subregions, parallel Metropolis-Hastings random walk (MHRW) is developed to gather the information of WSN nodes, including their IDs and neighbor tables. The collected information is then forwarded to the cloud through UAVs for the initial generation of logical topology. Thereafter, a network-wide 3D localization algorithm is further developed based on the discovered logical topology and multidimensional scaling method, termed as Topo-MDS, where the UAVs equipped with GPS are served as mobile anchors to locate the sensor nodes. Simulation results indicate that the parallel MHRW improves both the efficiency and accuracy of logical topology discovery. In addition, the Topo-MDS algorithm dramatically improves the 3D location accuracy, as compared to the existing algorithms in the literature.

## 5.1 Introduction

With the rapid development of sensing, communications and computing technologies, large-scale IoT systems have been gradually deployed in diverse applications, including environmental monitoring such as forest and ocean surveillance [70, 71]. In enabling large-scale environment monitoring, wireless sensor nodes are pervasively used due to their advantages of low power, low cost, and ease of deployment. These nodes are generally self-organized into WSNs for cost-effective data collection. Due to limited coverage and computing resources of a single WSN, standalone WSNs are gradually integrated into interconnected complex systems enabled by IoT technology [72]. This development brings the capability of large-scale network monitoring and management, as well as abundant computational resources of the IoT cloud platform [73]. Particularly, by the exploitation of cloud computing, sensor data gathered by the WSN nodes in target areas can be visualized and processed ubiquitously, so that unexpected events can be promptly detected and located.

The physical topology of the large-scale IoT system is often needed in the cloud, since it indicates not only the logical connectivity statuses (*i.e.*, logical topology) but also the physical locations of sensor nodes. However, due to the random deployment nature of WSNs, the physical topology of a large-scale IoT system is extremely hard to control during the deployment stage. Therefore, the development of a physical topology discovery scheme becomes an urgent necessity in improving the operational effectiveness of large-scale IoT applications.

In the literature, several studies have investigated logical topology discovery in WSNs. In [74], a topology map was derived from the virtual coordinate system of the WSN by singular value decomposition (SVD). Different from the virtual coordinate based method, the number of nodes estimation and topology discovery were implemented based on gathering the node IDs and coordinates in [75]. However, both efficiency and accuracy of these algorithms were not ideal, especially for the large-scale IoT systems. To overcome this difficulty, a parallel MHRW based logical topology discovery algorithm is developed in this chapter. More specifically, the target area is divided into several subregions by the cloud. The centroid of each subregion is selected as UAV [76] hovering point, while the beacon signal is broadcast. Sensor nodes are clustered into these subregions according to the received signal strength (RSS) of detected

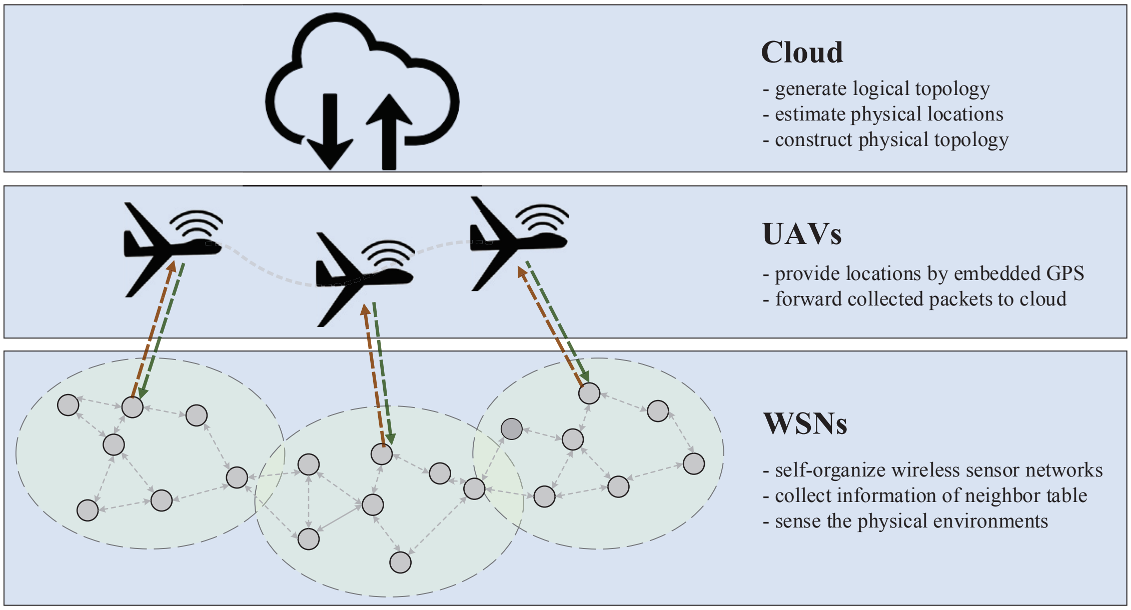

Cloud
- generate logical topology
- estimate physical locations
- construct physical topology
UAVs
- provide locations by embedded GPS
- forward collected packets to cloud
WSNs
- self-organize wireless sensor networks
- collect information of neighbor table
- sense the physical environments

Figure 5.1: A general architecture of the cloud-orchestrated large-scale IoT systems.

beacons. Parallel MHRW paths are simultaneously conducted in the subregions to gather the information of sensor nodes, including IDs and neighbor tables. The collected information is forwarded to the IoT cloud through the UAV for the initial generation of the logical topology.

In addition to the logical topology, the physical locations of sensor nodes are essential for the physical topology construction of WSNs. In the proposed system, one or multiple UAVs are utilized as mobile anchors to facilitate the localization. Given the UAVs equipped with GPS chipsets, 3D coordinates of sensor nodes can be estimated (3D, *i.e.*, latitude, longitude and altitude). There have been several studies focused on UAV-assisted 3D localization. In [77], multi-lateration method was used to derive the 3D coordinates. Each sensor node captured the broadcast beacons from the UAVs, which contained the real-time 3D location information of the UAVs. In [78], a mobile beacon-based 3D localization algorithm using the multidimensional scaling method was proposed, termed as MBL-MDS. Similarly, beacons from UAVs were captured for localization. The major difference was the exploitation of the multidimensional scaling method. In both works, each sensor node has to record the beacons and locate itself by the proposed algorithms. However, such methods would significantly increase the load of complexity and cost on the sensor nodes.

A network-wide 3D localization algorithm is then proposed based on the discovered logical topology and multidimensional scaling method, termed as Topo-MDS. By contrast, the localization is processed in the cloud in a centralized way. In the proposed Topo-MDS, distance matrices of subregions are established firstly. Multidimensional scaling and linear transformation methods are then used in turn to derive the physical coordinates. Combined with the logical topology discovered by parallel MHRW, the physical topology of the WSNs is finally built in the IoT cloud. Numerical simulations have been conducted in 3D-space scenarios. Simulation results have indicated that the parallel MHRW based logical topology discovery algorithm improves both estimation efficiency and accuracy. Additionally, the proposed Topo-MDS algorithm dramatically improves the 3D localization accuracy, as compared to the existing 3D localization algorithms in the literature.

The rest of this chapter is organized as follows. In Section 5.2, the proposed cloud-orchestrated large-scale IoT system is described in detail. In Section 5.3, the logical topology discovery algorithm is proposed based on the parallel MHRW. Topo-MDS localization algorithm is developed in Section 5.4. Based on Section 5.2-5.4, Section 5.5 details the proposed physical topology discovery scheme, which consists of initialization, parallel MHRW and construction procedures. Convergence time and topology estimation accuracy of the proposed scheme are evaluated in Section 5.6. Section 5.7 concludes this chapter.

## 5.2 Cloud-Orchestrated Large-Scale IoT Systems Using UAVs

A general architecture of the proposed cloud-orchestrated IoT systems is shown in Fig.5.1, which consists of three major components, namely, wireless sensor nodes, UAVs and a cloud platform. Details of each component are given below.

• *Wireless sensor nodes* are the fundamental components of IoT systems. These nodes are randomly deployed in monitoring areas and self-organized into WSNs to gather environmental information. For example, in a forest fire surveillance system, temperature, smoke, and olfactory sensors are used for fire detection [79]. Since several efforts have been done on the selection of suitable sensors, this work emphasizes more on the physical topology construction of the WSNs. In the proposed system as depicted in Fig.5.1, wireless sensor nodes are supposed to be homogeneous and have the capability of peer-to-peer communications. Each node

Table 5.1: Neighbor Table of Sensor Node $v_i$

| Neighbor List | Neighbor ID | Received Signal Strength (RSS) |
|---|---|---|
| $Nb(v_i)_1$ | $id_{Nb(v_i)_1}$ | $RSS(Nb(v_i)_1 \rightarrow v_i)$ |
| $Nb(v_i)_2$ | $id_{Nb(v_i)_2}$ | $RSS(Nb(v_i)_2 \rightarrow v_i)$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $Nb(v_i)_{d(v_i)}$ | $id_{Nb(v_i)_{d(v_i)}}$ | $RSS(Nb(v_i)_{d(v_i)} \rightarrow v_i)$ |

maintains a neighbor table, which contains the IDs of its neighbors and also the corresponding RSS. Table 5.1 shows an example of the neighbor table of node $v_i$, where $d(v_i)$ is the number of neighbors and $Nb(v_i)$ is the set of neighbors.

• *GPS-embedded UAVs* are served as mobile anchors to facilitate localization and topology discovery, and as mobile relays to forward information from sensor nodes to the cloud.

In order to reduce the manufacturing cost, sensor nodes are generally built without GPS chipsets. However, the awareness of location information in the cloud is a necessity for real-time event detection. With the knowledge of RSS information (*e.g.*, Table 5.1), relative coordinates of sensor nodes can be calculated. To further locate the actual coordinates, several physical locations known as anchor points are imperative. In this case, GPS-embedded UAVs are exploited to provide real-time 3D location information. In the cloud-orchestrated IoT system, the flight paths of UAVs are designed in advance by the cloud. The UAVs are correspondingly programmed to autonomously fly to the target positions without additional human intervention [80]. Furthermore, UAVs are able to carry different RF modules and support different wireless communication protocols. For instance, UAVs have the capability of communicating with sensor nodes in a self-organized way through ZigBee modules [81] and possibly serve as relays to forward the information to the cloud [82].

• *IoT cloud platform* is the remote data and control center for the IoT systems, leveraging cloud computing to achieve complex data processing and analysis, as well as coordination of UAV flight paths. By exploitation of the proposed scheme, the physical topology of the WSNs is finally built in the cloud. Benefited from the cloud, sensor data sampled from the target areas can be timely and efficiently visualized and located, since physical topology illustrates both the logical topology of the randomly deployed WSNs and the physical locations of actual nodes.

## 5.3 Logical Topology Discovery by Subregion-based Parallel Metropolis-Hastings Random Walk

The physical topology consists of the information of logical topology and physical location. Thus, a logical topology discovery algorithm is firstly proposed in this section, which is implemented through subregion-based parallel MHRW.

### 5.3.1 Modeling of a WSN as a Graph

A WSN is modeled as an undirected graph $G = (V, E)$, where $V$ is the set of vertices and $E$ is the set of edges. In the WSN, sensor nodes are modeled as vertices, and wireless communication links between nodes are modeled as edges. Given $n$ nodes within a WSN, the set of vertices is expressed as $V = \{v_1, v_2, \dots, v_n\}$, and the number of vertices is represented by $|V| = n$. Similarly, the set of edges $E = \{e_1, e_2, \dots, e_m\}$ represents the wireless communication links, and the number of links is $|E| = m$. Besides the vertex and edge, the degree of a vertex is also a non-trivial consideration in graph-based investigations. Here, the degree of a vertex is modeled by the number of valid neighbors of a sensor node and marked as $d(v_i)$. The value of $d(v_i)$ may vary from the initial setup, due to the physical obstacles, malfunctions and dead nodes. Thus, only the nodes with valid wireless communication capability are defined as valid neighbors.

### 5.3.2 Metropolis-Hastings Random Walk on a Graph

The simple random walk process biases towards nodes with higher degrees. Hence, MHRW is developed to modify the transition probability matrix, so that the process can converge to the desired uniform stationary distribution and remove the degree bias [83]. The transition probability in MHRW is given by

$$P^{MH}_{v_i,v_j} = \begin{cases} \frac{1}{d(v_i)} \cdot \min(1, \frac{d(v_i)}{d(v_j)}), & if\ v_j \in\ Nb(v_i), \\ 1 - \sum_{k \neq i} P^{MH}_{v_i,v_k}, & if\ v_j = v_i, \\ 0, & otherwise. \end{cases} \tag{5.1}$$

Based on (5.1), we can derive that a packet forwarded along the MHRW path in a WSN can return back to the starting node with an expected time of $|V|$. Let the packet gather the information of each node that it visits, including node IDs and neighbor tables. By collecting the returning packets, information specific to the sensor nodes can be collected. Based on the connected statuses stated in neighbor tables, the logical topology can be established at the IoT cloud platform. Logical adjacency matrix ($\tilde{\mathbf{C}}$) of the WSN is generated by

$$\tilde{\mathbf{C}}_{v_i,v_j} = \begin{cases} 1, & if\ e(v_i, v_j)\ exists, \\ 0, & otherwise, \end{cases} \tag{5.2}$$

where the existence of $e(v_i, v_j)$ means that vertices $v_i$ and $v_j$ can wirelessly communicate.

### 5.3.3 Logical Topology Discovery by Subregion-based Parallel Metropolis-Hastings Random Walk Processes

The expected return time $|V|$ indicates that the time is proportional to the number of vertices. Therefore, with the increment in the scale of WSNs, the convergence time of global MHRW would be monotonously increased as a consequence. To deal with the low efficiency of global MHRW, a novel subregion-based parallel MHRW is proposed. Details of the proposed algorithm are stated as follows.

Sensor nodes are firstly clustered into subregions. The target field is uniformly divided into N subregions by the cloud, and centroids of the subregions are selected as UAV hovering points. At each hovering point, UAV broadcasts a beacon signal. Each sensor node records all the detected beacons and selects the certain subregion, from which the RSS of the beacon signal is the highest. RSS is chosen as the clustering metric, since RSS measurement can be easily achieved without modifications on sensor nodes.

Afterwards, parallel MHRW processes are simultaneously taken place at the subregions. In each subregion, the cloud randomly selects a sensor node as the starting point and a packet is generated correspondingly. Packets are then forwarded within the subregions following the MHRW rule to gather the information of sensor nodes, as stated in Subsection 5.3.2. After the packets return, the UAVs are functioned as relays and forward the packets to the IoT cloud. The logical topology is finally established in the cloud by (5.2) based on the collected information.

The convergence time of global and parallel MHRW and the accuracy of logical topology discovery would be investigated in Subsection 5.6.3 and 5.6.4.

## 5.4 Topo-MDS: Logical Topology and Multidimensional Scaling based 3D Localization

In order to construct the physical topology of the WSNs in the cloud, physical location information needs to be aware as well. Based on the logical topology discovered in Section 5.3, a UAV-assisted network-wide 3D localization algorithm is further developed to estimate the physical coordinates of sensor nodes. The proposed algorithm exploits UAV hovering points as anchor locations, so that distance matrices of the subregions can be built. Based on the distance matrices, the relative and physical coordinates are estimated in turn by multidimensional scaling and linear transformation methods.

### 5.4.1 Relative Location Estimation by Multidimensional Scaling

Multidimensional scaling (MDS) method can be used to calculate the relative coordinates, when the distances between sensor nodes are able to be estimated [84]. Specifically, the distance matrix **D** is formulated as

$$\mathbf{D} = \left[\begin{array}{ccc|ccc} \tilde{d}_{v_1,v_1} & \dots & \tilde{d}_{v_1,v_n} & \tilde{d}_{v_1,a_1} & \dots & \tilde{d}_{v_1,a_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \tilde{d}_{v_n,v_1} & \dots & \tilde{d}_{v_n,v_n} & \tilde{d}_{v_n,a_1} & \dots & \tilde{d}_{v_n,a_m} \\ \hline \tilde{d}_{a_1,v_1} & \dots & \tilde{d}_{a_1,v_n} & d_{a_1,a_1} & \dots & d_{a_1,a_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \tilde{d}_{a_m,v_1} & \dots & \tilde{d}_{a_m,v_n} & d_{a_m,a_1} & \dots & d_{a_m,a_m} \end{array}\right], \tag{5.3}$$

where $\tilde{d}_{v_i,v_j}(i, j = 1, 2, \dots, n)$ is the estimated distance between sensor nodes $v_i$ and $v_j$, $\tilde{d}_{v_i,a_j}(i = 1, 2, \dots, n, j = 1, 2, \dots, m)$ is the estimated distance between sensor node $v_i$ and anchor point $a_j$, and $d_{a_i,a_j}(i, j = 1, 2, \dots, m)$ is the actual distance between anchor points $a_i$ and $a_j$.

Let $\mathbf{B}$ denote the double centralization of $\mathbf{D}$ by

$$\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{D}^2\mathbf{J}, \tag{5.4}$$

where

$$\mathbf{J} = \mathbf{I}_{n+m} - \frac{1}{n+m}\mathbf{1}_{n+m} \cdot \mathbf{1}_{n+m}^T, \tag{5.5}$$

where $\mathbf{I}_{n+m}$ is an $(n+m) \times (n+m)$ identity matrix and $\mathbf{1}_{n+m} = [1, 1, \ldots, 1]^T$ is an $(n+m) \times 1$ unit vector.

And then, the relative coordinates $\hat{\hat{\mathbf{P}}}$ can be obtained by minimizing the squared error expression,

$$\hat{\hat{\mathbf{P}}} = \arg\min_{\overline{\mathbf{P}}} ||\mathbf{B} - \overline{\mathbf{P}}\overline{\mathbf{P}}^T||^2, \tag{5.6}$$

where $\overline{\mathbf{P}} = \mathbf{J}\mathbf{P}$ is the centralized coordinate matrix, and $\mathbf{P}$ is the coordinate matrix $\mathbf{P}_{[(n+m)\times 3]} = [\mathbf{p}_{v_1}, \ldots, \mathbf{p}_{v_n}, \mathbf{p}_{a_1}, \ldots, \mathbf{p}_{a_m}]^T$.

The minimization problem in (5.6) can be solved by decomposing $\mathbf{B}$ using the eigenvalue decomposition,

$$\mathbf{B} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \tag{5.7}$$

where $\mathbf{\Lambda}$ is the ordered diagonal eigenvalue matrix $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \ldots, \lambda_{n+m}), \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{n+m}$, and $\mathbf{U}$ is the eigenvector matrix containing the corresponding eigenvectors as columns. Finally, the relative coordinates are estimated by

$$\hat{\hat{\mathbf{P}}} = \mathbf{U}_3\mathbf{\Lambda}_3^{1/2}, \tag{5.8}$$

where $\mathbf{U}_3$ consists of the first three columns in $\mathbf{U}$ and $\mathbf{\Lambda}_3^{1/2} = diag(\lambda_1^{1/2}, \lambda_2^{1/2}, \lambda_3^{1/2})$.

### 5.4.2 Physical Location Estimation by Linear Transformation

$\hat{\hat{\mathbf{P}}}$ in (5.8) is the relative coordinates. The physical coordinates $\tilde{\mathbf{P}}$ can be estimated by

$$\tilde{\mathbf{P}} = c\mathbf{R}\hat{\hat{\mathbf{P}}} + \mathbf{t} \cdot \mathbf{1}_{n+m}^T, \tag{5.9}$$

where the scale $c$ is set to 1 for simplification, and the parameters $\mathbf{R}$ and $\mathbf{t}$ can be obtained by minimizing the squared error of linear transformation of the anchor points as

$$\arg\min_{c,\mathbf{R},\mathbf{t}} \sum_{i=1}^{m} \|c\mathbf{R}\hat{\hat{\mathbf{p}}}_{a_i} + \mathbf{t} - \mathbf{p}_{a_i}\|_2^2, \tag{5.10}$$

where $\hat{\hat{\mathbf{p}}}_{a_i} = [\hat{\hat{x}}_{a_i}, \hat{\hat{y}}_{a_i}, \hat{\hat{z}}_{a_i}]^T$ and $\mathbf{p}_{a_i} = [x_{a_i}, y_{a_i}, z_{a_i}]^T$ are the relative and physical coordinates of anchor point $a_i$. $m$ is the number of anchor points. Arun's method [85] is introduced to solve the problem defined by (5.10).

Firstly, the physical and relative coordinate matrices of anchor nodes, $\mathbf{P}'_{[3\times m]}$ and $\hat{\hat{\mathbf{P}}}'_{[3\times m]}$, are off-mean by

$$\overline{\mathbf{P}'} = \mathbf{P}' - \boldsymbol{\mu}_p \cdot \mathbf{1}_m^T, \tag{5.11}$$

and

$$\overline{\hat{\hat{\mathbf{P}}}'} = \hat{\hat{\mathbf{P}}}' - \boldsymbol{\mu}_{\hat{\hat{p}}} \cdot \mathbf{1}_m^T, \tag{5.12}$$

where $\boldsymbol{\mu}_p = \frac{1}{m}\sum_{i=1}^{m} \mathbf{p}_{a_i}$ and $\boldsymbol{\mu}_{\hat{\hat{p}}} = \frac{1}{m}\sum_{i=1}^{m} \hat{\hat{\mathbf{p}}}_{a_i}$.

Afterwards, $\overline{\mathbf{P}'}\,\overline{\hat{\hat{\mathbf{P}}}'}^T$ is decomposed by SVD,

$$\overline{\mathbf{P}'}\,\overline{\hat{\hat{\mathbf{P}}}'}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T. \tag{5.13}$$

Finally, the rotation matrix $\mathbf{R}$ and the translation matrix $\mathbf{t}$ are calculated as

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T,\ \mathbf{t} = \boldsymbol{\mu}_p - \mathbf{R}\boldsymbol{\mu}_{\hat{\hat{p}}}. \tag{5.14}$$

$\mathbf{R}$ and $\mathbf{t}$ are substituted into (5.9) to estimate the physical coordinates of $n$ sensor nodes.

### 5.4.3 Topo-MDS: Logical Topology and Multidimensional Scaling based Network-Wide 3D Localization Algorithm

Based on logical topology discovered in Section 5.3 and multidimensional scaling method, a network-wide 3D localization algorithm termed as Topo-MDS is further proposed. UAV hovering points are used as anchor locations to locate the sensor nodes as stated in (5.10). The specific procedures of the Topo-MDS algorithm are detailed as follows.

I Given $k$ sensor nodes in a subregion (sensor nodes are clustered in Section 5.3), a distance matrix is initialized as $\mathbf{D}_{1_{[(k+1)\times(k+1)]}}$ and the first UAV hovering point is recorded as $ml_1$.

II When the UAV moves to the 2nd location, the distance matrix is expanded to $\mathbf{D}_{2_{[(k+2)\times(k+2)]}}$ and the location is recorded as $ml_2$. Similarly, while the UAV moves to the $m$th location, the distance matrix is enlarged to $\mathbf{D}_{m_{[(k+m)\times(k+m)]}}$ and UAV location is recorded as $ml_m$.

III Given the distance matrix $\mathbf{D}_m$, the relative coordinates can be calculated by (5.3)-(5.8) in Subsection 5.4.1 through multidimensional scaling method. With $m$ recorded locations ($ml_{1,\ldots,m}$) and the relative coordinates, the locations of $k$ sensor nodes can be estimated by (5.9)-(5.14) through linear transformation in Subsection 5.4.2.

## 5.5 Proposed Physical Topology Discovery Scheme

The physical topology discovery scheme is proposed in this section, based on the investigations of parallel MHRW based logical topology discovery and Topo-MDS algorithm based 3D localization in Section 5.3 and 5.4, respectively. The proposed scheme consists of three phases, initialization, parallel MHRW and construction, as summarized in Algorithm 4. Moreover, specific details of each procedure are explained in the following paragraphs.

**Initialization** According to the IEEE802.15.4e TSCH protocol, sensor nodes indicate their existence and share information by sending enhanced beacons during the period of network initialization [86]. Based on received beacons, each node sets up its neighbor table ($tb$) that contains neighbor IDs and RSS of signals from its neighbors, as stated in Table 5.1. Afterwards, the target area is uniformly divided into several subregions by the cloud. UAV hovers at the centroid of each subregion and broadcasts a beacon signal. Sensor nodes are clustered according to the RSS of beacon signals. In subregion $i$, a sensor node is selected as the starting point of MHRW by the cloud, marked as $s_i$. A packet ($Pk_i$) is generated at the starting node $s_i$, while ID and neighbor table of the node are added to the packet in the meantime.

**Algorithm 4** Physical Topology Discovery Scheme

```
Initialization:
uniformly divide the target area into N subregions
UAV hovers at the centroid of each subregion and broadcasts a beacon signal
sensor nodes are clustered into the subregions according to the RSS of detected beacons
generate a packet $Pk_i$ at the cloud-selected starting node $s_i$ in subregion $i$ ($i = 1, 2, \dots, N$)
add ID ($id_{s_i}$) and neighbor table ($tb_{s_i}$) of $s_i$ to $Pk_i$
Parallel MHRW:
forward packet $Pk_i$ in subregion $i$ by MHRW rule
if node $v \in$ subregion $i$ & node $v \notin Pk_i$ then
    add $id_v$ and $tb_v$ to $Pk_i$
end if
Construction:
discover the logical topology $\Rightarrow$ adjacency matrix $\tilde{\mathbf{C}}$
estimate the physical locations of sensor nodes $\Rightarrow$ coordinate matrix $\tilde{\mathbf{P}}$
construct the physical topology ($\tilde{\mathbf{C}}, \tilde{\mathbf{P}}$) in the IoT cloud
```

**Parallel MHRW** Starting nodes for the $N$ subregions are selected by the cloud in the phase of initialization. $N$ parallel random walk processes are simultaneously conducted according to the Metropolis-Hastings rule as defined by (5.1). After each time of packet transmission, the node ID of the packet receiver is checked. If the node ID (*e.g.*, $id_v$) is not on the list of packet $Pk_i$, then ID and neighbor table of the packet receiver (node $v$) are added to packet $Pk_i$.

**Construction** After packet $Pk_i$ returns back to its starting node $s_i$, the collected packet would be forwarded to the IoT cloud platform through UAV. In the cloud, the physical topology is constructed by the collected node IDs and neighbor tables. More specifically, the logical topology is firstly established by the connectivity statuses stated in neighbor tables, as investigated in Section 5.3. The physical coordinates of sensor nodes are then calculated by the UAV-assisted Topo-MDS algorithm developed in Section 5.4. Finally, the physical topology is constructed by the combination of logical topology and estimated 3D locations of sensor nodes.

## 5.6 Performance Evaluation

Simulations have been conducted to evaluate the proposed physical topology discovery scheme from different aspects, including convergence time, logical topology estimation accuracy and

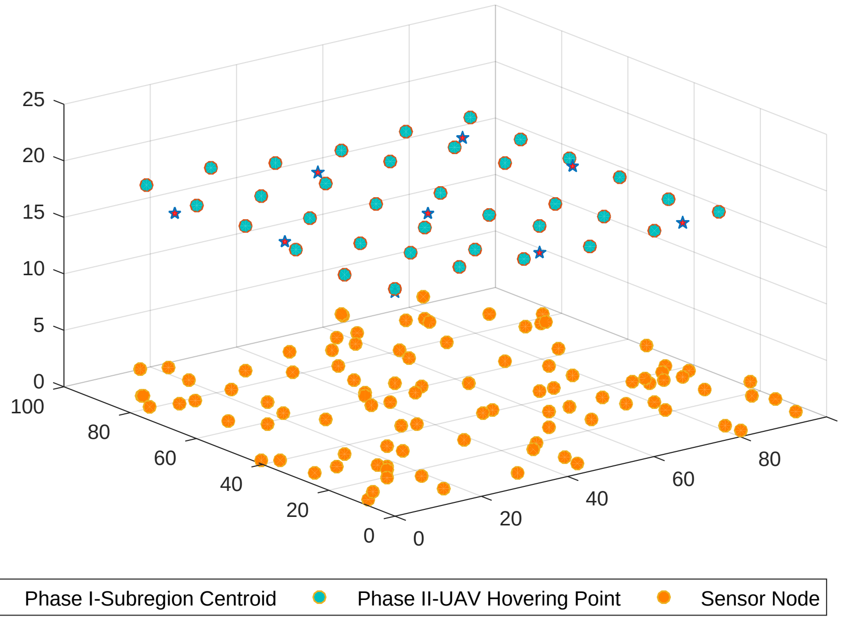


Figure 5.2: Deployment of UAV hovering points and sensor nodes in the 3D scenario.

3D location accuracy. System settings and wireless communication channel models used in the simulations are firstly given in Subsection 5.6.1 and 5.6.2.

### 5.6.1 Simulation Settings

In the simulation, 100 sensor nodes are randomly deployed in a 3D space [100m×100m×1m]. These nodes are supposed to be homogeneous with transmitting power ranging from -10, -5 to 0dBm and receiving sensitivity -90dBm [87].

UAV hovering points are deployed as shown in Fig.5.2. In the phase of logical topology discovery, the UAV hovers at the latitude-longitude centroid of the subregion in 20m height for sensor nodes clustering and data collection. Later in the localization phase, the UAV hovers at designed intervals and heights as anchor locations. The hovering bias of UAV is ±1.5m in latitude and longitude, and ±0.5m in altitude [88].

### 5.6.2 Wireless Communication Channel Models

For the signal propagation from UAV to sensor nodes (SNs), and peer-to-peer wireless communication channels among SNs, the two-ray ground model and the free-space outdoor model are respectively used, taking into account the different signal propagation environments.

For the UAV-SN air-to-ground signal propagation, the two-ray ground model is a commonly used channel model [77,89], which considers both the line-of-sight (LOS) and ground-reflected rays. For wireless communications among SNs, the signal propagation channel quality is not as ideal as UAV-SN, given the potential near-ground scatters. Instead of the two-ray ground model, the free-space outdoor model (FOM) is thus adopted. This is a channel model designed specifically for WSNs in the outdoor open areas, which jointly considers the effect of the free-space propagation, ground reflection, RSS uncertainty, and antenna radiation impact [90].

**Two-ray Ground Model** For large distance $d$, the received power $P_r$ (in dBm) can be expressed by the two-ray ground model as [89],

$$P_r(dBm) = P_t + 10log(G_tG_r) + 20log(H_tH_r) - 40log(d), \tag{5.15}$$

where $P_t$ is the transmitting power. $d$ is the horizontal distance between transmitter and receiver. $G_t$ and $G_r$ are the antenna gains of transmitter and receiver, $G_t = G_r = 1$. $H_t$ and $H_r$ are the antenna heights of transmitter and receiver.

**Free-Space Outdoor Model** The received power (in dBm) is modeled by [90],

$$P_r(dBm) = P_t + 20log(\frac{\lambda}{4\pi d}) + 10log(K_1^2 + K_2^2\Gamma^2 + 2K_2\Gamma cos(\frac{2\pi}{\lambda}\Delta L)) + X_\sigma, \tag{5.16}$$

where $\lambda$ is the propagation wavelength, and $K_1$ and $K_2$ are coefficients irregularity in antenna radiation pattern. $\Delta L$ is the path difference between LOS and ground-reflected rays. $X_\sigma$ is the RSS uncertainty that follows Gaussian distribution. $\Gamma$ is the ground reflection coefficient,

$$\Gamma = \frac{\sin\theta - \sqrt{(\varepsilon - jx_\Gamma) - \cos^2\theta}}{\sin\theta + \sqrt{(\varepsilon + jx_\Gamma) - \cos^2\theta}}, \tag{5.17}$$

where parameters of average ground are used without losing generality, $\varepsilon = 15, x_\Gamma = 3.75 \times 10^{-2}$. $\theta$ is the reflection angle.

Table 5.2: Comparison between Global and Parallel MHRW on Convergence Time (Hops)

| Transmitting Power, $P_t$ (dBm) | -10 | -5 | 0 |
|---|---|---|---|
| Global MHRW | 666 | 742 | 1145 |
| Parallel MHRW (#$c$=4) | 264 | 309 | 340 |
| Parallel MHRW (#$c$=9) | 229 | 271 | 307 |

### 5.6.3 Convergence Analysis

The convergence of random walk is analyzed by the Geweke's diagnostics [91]. In Geweke's diagnostics, convergence is evaluated by the difference between the first 10% and last 50% of a data sequence. Mostly, the difference is calculated by Z statistics as

$$Z = \frac{E[\mathbf{X}_1] - E[\mathbf{X}_2]}{\sqrt{Var[\mathbf{X}_1] + Var[\mathbf{X}_2]}}, \tag{5.18}$$

where $\mathbf{X}_1$ is the first 10% of data sequence $\mathbf{X}$ and $\mathbf{X}_2$ is the last 50%. The first 10% is determined as convergence when the Z-score falls in the range $[-1, 1]$. Here, the number of node IDs that have been collected is used as the convergence evaluation metric $\mathbf{X}$.

The convergence time is defined as the number of hops that the random walk path has visited before it is convergent. In terms of the convergence time, the global random walk is not efficient enough, especially for large-scale WSNs [92]. That is why the subregion-based parallel MHRW is further proposed. The convergence time of the parallel MHRW is determined by the number of hops of the longest path. The comparison on the convergence time between the global MHRW and the subregion-based parallel MHRW with different numbers of subregions (#$c$) is listed in Table 5.2, where different transmitting powers are considered.

From Table 5.2, it is clear that the convergence time of parallel MHRW (#$c$=4) is reduced by 60.4% as compared to the global MHRW. With the number of subregions increasing from 4 to 9, the convergence time saved further increases to 65.6%. This is because the more subregions, the more parallel random walk processes are simultaneously conducted, leading to the dramatic reduction in the convergence time. Moreover, in parallel MHRW (#$c$=4), with the transmitting power increasing from -10 to 0dBm, the convergence time increases from 264 to 340 hops. The reason is that the increment in the transmitting power decreases the difference between node connectivity degree in the WSN. The more uniform network converges the slower [91].

Table 5.3: Topology Estimation Error of Topology Preserving Map Method

| $P_t$ (dBm) | case 1 | case 2 | case 3 | case 4 |
|---|---|---|---|---|
| -10 | 0.9850 | 1.0377 | 0.9682 | 0.8887 |
| -5 | 0.8696 | 0.7902 | 0.8537 | 0.7240 |
| 0 | 0.9431 | 0.9431 | 0.9441 | 0.3562 |

### 5.6.4 Logical Topology Estimation Analysis

In order to evaluate the estimation accuracy of the parallel MHRW based logical topology discovery algorithm, the estimation error is defined as

$$\varepsilon_c = ||\mathbf{C} - \tilde{\mathbf{C}}||_1/||\mathbf{C}||_1, \tag{5.19}$$

where $\mathbf{C}$ and $\tilde{\mathbf{C}}$ are the logical adjacency matrices of actual and estimated network topologies.

Before evaluating the proposed algorithm, the topology preserving map method [74] is used as the benchmark. All four cases proposed in [74] are tested in the randomly deployed scenario. The transmitting powers ($P_t$) are set to -10, -5 and 0 dBm, respectively. The experimental results are listed in Table 5.3. It indicates that case 4 performs the best. However, the lowest estimation error in case 4 is still as high as 0.3562, which indicates that the method proposed in [74] is not suitable for the logical topology discovery in the randomly deployed scenario.

The logical topology estimation error of the proposed algorithm with different transmitting powers is shown in Fig.5.3, where #$c$=9. It can be seen that the estimation error finally converges to 0, which is much lower than the results in Table 5.3. In terms of Fig.5.3, at the beginning, error generated by $P_t$ =-10 dBm is the largest. This is because, with lower transmitting power, the average node degree is lower, so that the connectivity information sampled from the neighbor tables is less. However, given the MHRW converges faster with lower transmitting power, the path of $P_t$ =-10 dBm converges to 0 the fastest.

### 5.6.5 Physical Location Estimation Analysis

In Subsection 5.6.5, accuracy of 3D localization is investigated. Before evaluating the proposed Topo-MDS algorithm, RSS-based distance estimation model and evaluation metric are introduced as follows.

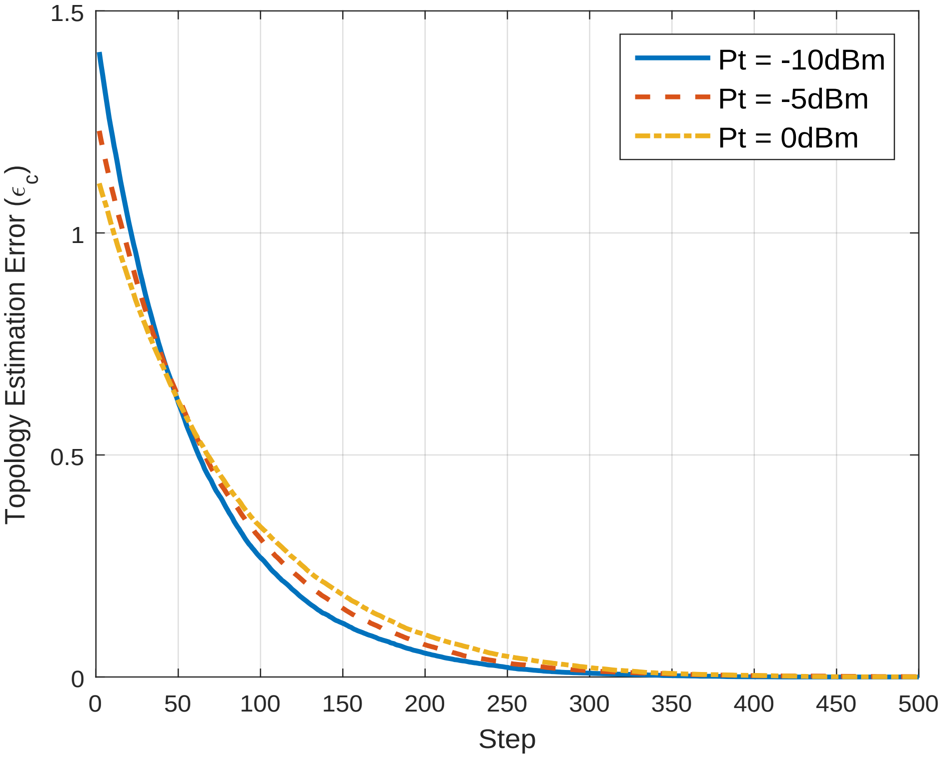


Figure 5.3: Logical topology estimation error $\varepsilon_c$ with different transmitting powers $P_t$.

**RSS-based Distance Estimation** Distance between sensor nodes and distance between sensor node and UAV are both estimated by the RSS. The model in [78, 93] is adopted as the RSS-based distance estimation model, where the estimated distance $\tilde{d}_{v_i,v_j}$ is assumed to be affected by estimation offset $\eta_{v_i,v_j}$ as

$$\tilde{d}_{v_i,v_j} = d_{v_i,v_j} + \eta_{v_i,v_j}, \tag{5.20}$$

where the offset $\eta_{v_i,v_j}$ follows Gaussian distribution, $\eta_{v_i,v_j} \sim \mathcal{N}(0, \sigma^2_{v_i,v_j})$ and $\sigma^2_{v_i,v_j} = (\gamma \cdot d_{v_i,v_j})^2$. $\gamma$ is the ratio of standard deviation of the distance estimation offset to the actual distance $d_{v_i,v_j}$.

**Evaluation Metric** Sensor nodes are located in three dimensions by the Topo-MDS algorithm proposed in Section 5.4, while location error is quantified by the average Euclidean distance between actual and estimated 3D locations,

$$\varepsilon_p = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{p}_{v_i} - \tilde{\mathbf{p}}_{v_i}\|_2, \tag{5.21}$$

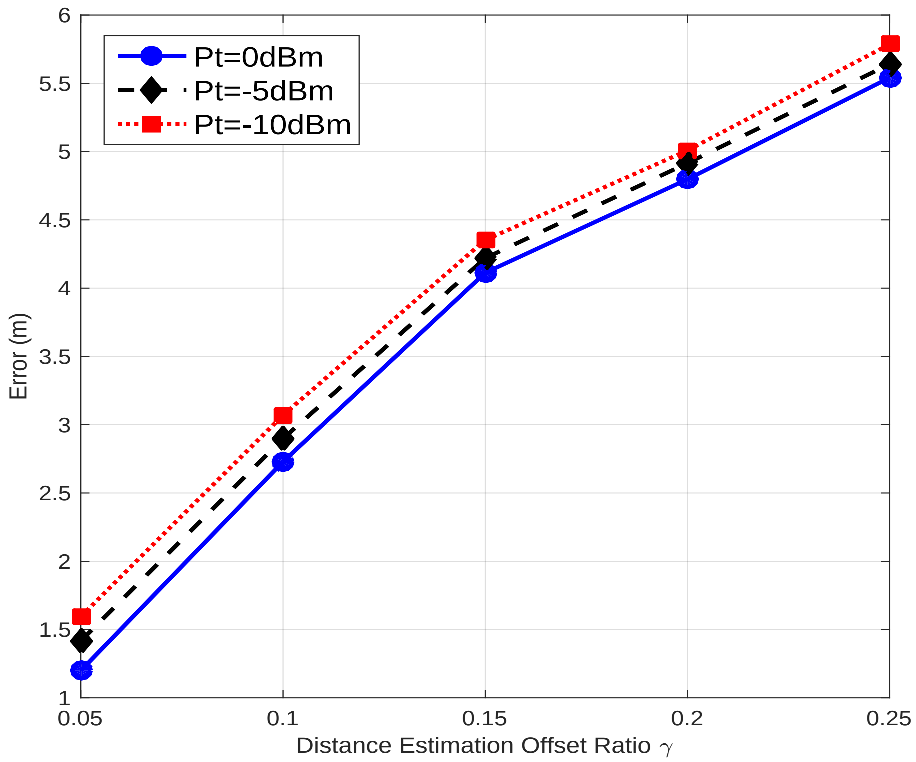


Figure 5.4: Influence of the transmitting power $P_t$ and the distance estimation offset ratio $\gamma$ on 3D location error.

where $\mathbf{p}_{v_i}$ is the ground-truth coordinate of node $v_i$, *i.e.*, $\mathbf{p}_{v_i} = (x_{v_i}, y_{v_i}, z_{v_i})$, and $\tilde{\mathbf{p}}_{v_i}$ is the estimated coordinate.

**Evaluation of Performance Influence Factors** Factors that may have effects on the 3D location estimation accuracy are investigated, including transmitting power of sensor nodes ($P_t$), distance estimation offset ratio ($\gamma$), the number of subregions ($\#c$) and UAV hovering interval. UAV hovering interval is the distance between the hovering point at the latitude-longitude plane. Simulation results are shown in Fig.5.4 and Fig.5.5.

In Fig.5.4, we evaluate the influence of transmitting power and the distance estimation offset. The transmitting power $P_t$ increases from -10, -5 to 0dBm and distance estimation offset ratio $\gamma$ ranges from 5% to 25%, while the interval is fixed to 10m and the number of subregions is 9. From Fig.5.4, it can be seen that the location error of the proposed Topo-MDS algorithm decreases with the increment in transmitting power. The reason is that the increment in transmitting power enlarges the average number of neighbor sensor nodes so

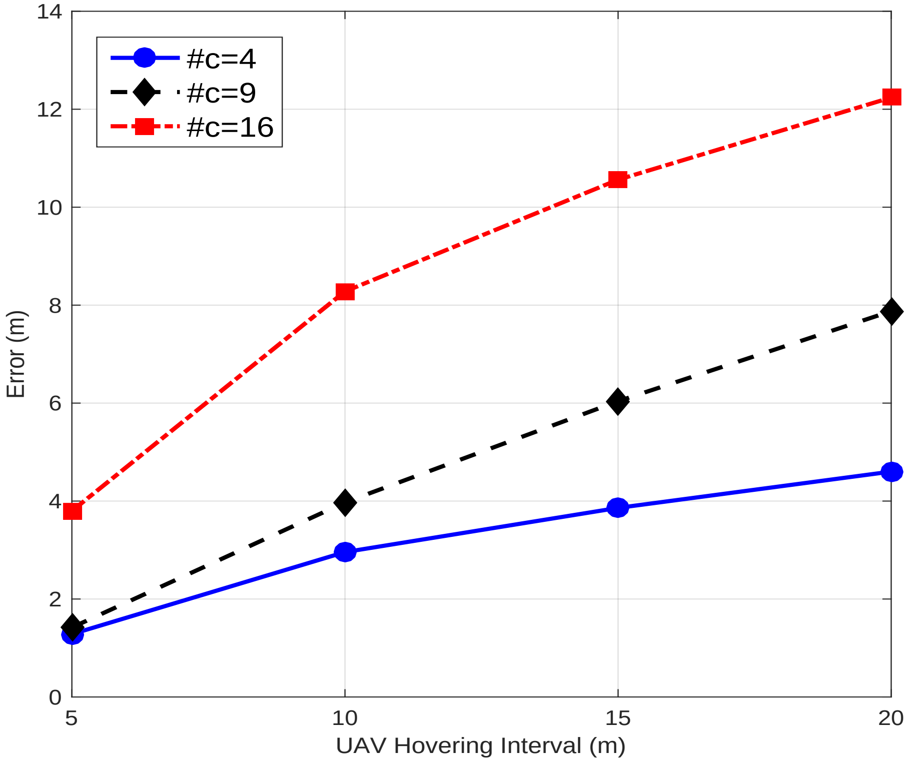


Figure 5.5: Influence of the number of subregions #$c$ and the UAV hovering interval (m) on 3D location error.

that the fundamental distance matrix (5.3) is expanded. Besides, the location error increases monotonously with the growth in distance estimation offset ratio $\gamma$. As stated in Topo-MDS, the 3D coordinates of sensor nodes are derived from the distance matrix. Thence, the larger distance estimation offset finally results in the increasing location error.

In Fig.5.5, the effects of the UAV hovering interval and the number of subregions are investigated. The UAV hovering interval enlarges from 5 to 20m, and the number of subregions (#$c$) increases from 4 to 16, while $P_t$ is set to 0dBm and distance estimation offset ratio is 15%. Fig.5.5 indicates that the increasing UAV hovering interval enlarges the location error. This is because UAV hovering points are used as anchors, while the increasing interval reduces the number of anchors. Furthermore, it can be noticed that more subregions result in larger location error. The reason is that the larger number of subregions downsizes the fundamental distance matrix (5.3) of each subregion. Combined with the result from Subsection 5.6.3, the increment in the number of subregions (#$c$) accelerates the convergence rate, while decreases the location accuracy. The trade-off would be balanced by the requirements of specific applications.

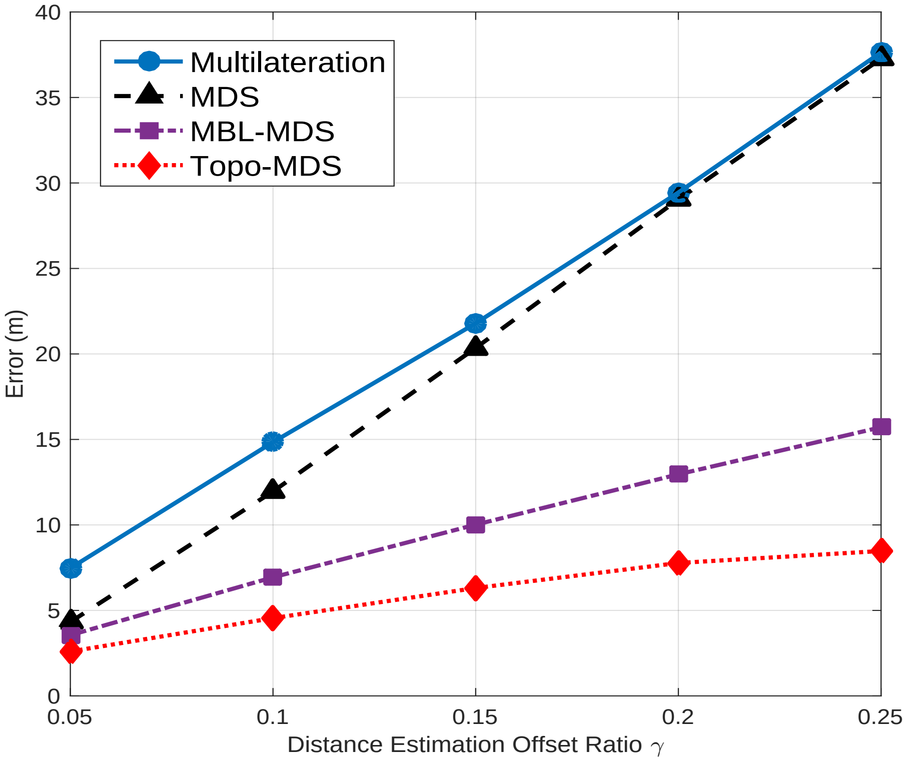


Figure 5.6: Scenario 1 (UAV heights ∈[20m 50m]): comparison between Topo-MDS and benchmark algorithms (multi-lateration, MDS, MBL-MDS) on 3D location error.

**Performance Comparison with Benchmark Methods** In order to better evaluate the performance of the proposed Topo-MDS, multi-lateration based 3D localization algorithm [77], MDS-based algorithm, and mobile beacon based 3D localization with MDS (MBL-MDS) algorithm [78] are selected as the benchmark methods. Two different scenarios of UAV hovering in the benchmark works [77, 78] are considered. In scenario 1 [77], UAV flies at a random altitude (range=[20m,50m]). In scenario 2 [78], UAV hovers on a plane, where the altitude is fixed to 20m. In both scenarios, UAV hovering biases are involved. Transmitting power is set to 0dBm, the interval is 15m and 9 subregions are separated. Fig.5.6 and Fig.5.7 demonstrate the location error of the proposed Topo-MDS and benchmark algorithms in two scenarios, respectively.

From Fig.5.6 and Fig.5.7, we can observe that the location error of the proposed Topo-MDS algorithm is the lowest in both scenarios. MBL-MDS algorithm [78] performs better in scenario 2 than scenario 1, since it is particularly proposed for the case that UAV hovers on a plane. But even in scenario 2, the location error of MBL-MDS is still higher than Topo-MDS.

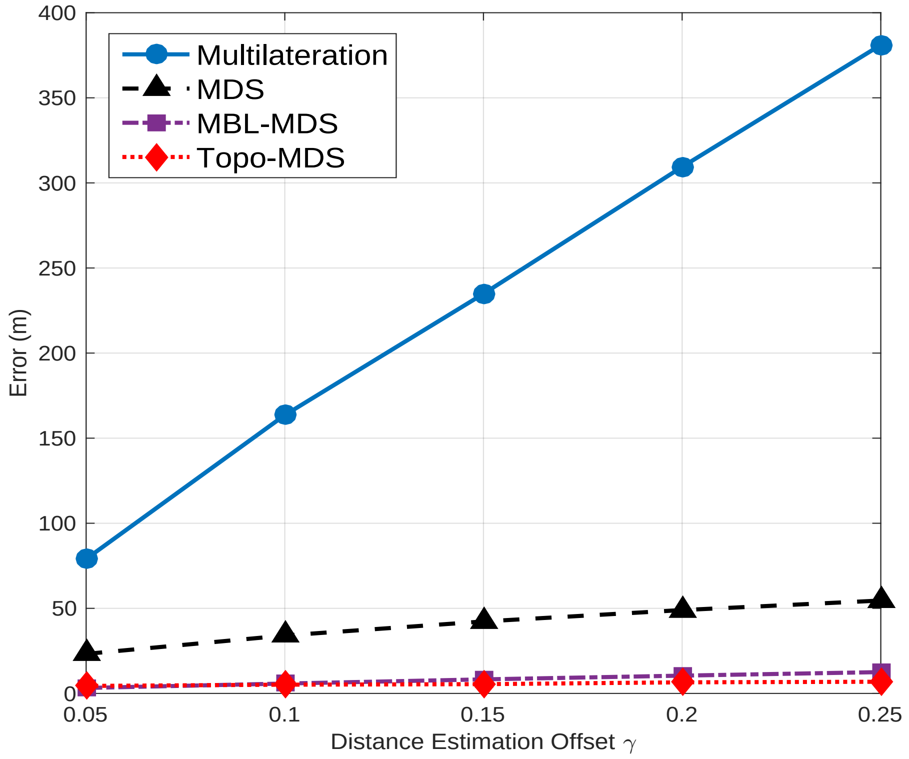


Figure 5.7: Scenario 2 (UAV heights =20m): comparison between Topo-MDS and benchmark algorithms (multi-lateration, MDS, MBL-MDS) on 3D location error.

The reason is that in MBL-MDS, a sensor node locates itself by recording the beacons and calculating its distances to UAV hovering points. In Topo-MDS, the distance matrix is set up based on not only the distances to UAVs but also the distances between neighbor sensor nodes. The network-wide approach improves the accuracy of 3D localization. Although the location error of multi-lateration based algorithm [77] is acceptable in scenario 1, the error in scenario 2 is extremely high, which implies that the multi-lateration based algorithm is not suitable for scenario 2 at all. This is because UAV hovering on a plane leads to the near singularity of the system equations in the multi-lateration based algorithm.

**Complexity Analysis** In addition to the aforementioned factors, the computational complexity of the proposed algorithm needs to be analyzed as well. In particular, the complexity of fundamental multidimensional scaling and linear transformation calculations is dominated by the eigenvalue decomposition. Given an $N \times N$ symmetric matrix, the complexity is $O(N^3)$ for eigenvalue decomposition. Thus, for Topo-MDS algorithm, the complexity is $O((k+m)^3)$, where $k$ and $m$ are the numbers of sensor nodes and UAV hovering points in a subregion respectively. In terms of MDS-based and MBL-MDS algorithms, the calculation is processed at each sensor node individually, *i.e.*, the number of sensor nodes is 1 ($k = 1$) and the complexity is exponentially relevant to the number of anchor locations received from UAVs ($m'$). Therefore, the complexity of the MDS-based algorithm is $O(m'^3)$. For MBL-MDS, the complexity is reduced to $O(\kappa \cdot m''^3)$, where $m''$ is the number of anchors selected in each subset ($m'' \ll m'$) and $\kappa$ is the number of subsets ($1 \leq \kappa \leq \lfloor m'/m'' \rfloor$). The multi-lateration based algorithm is also locally executed, but the computational complexity is much lower as $O(m')$.

The proposed Topo-MDS algorithm expands the distance matrix to improve location accuracy, but the computational complexity is increased in the meantime. Although the complexity of Topo-MDS is higher than other algorithms, it is implemented in the cloud platform instead of individual sensor nodes, so the increment in complexity is acceptable.

## 5.7 Chapter Summary

In this chapter, a cloud-orchestrated physical topology discovery scheme has been developed for large-scale IoT systems. The proposed discovery scheme consists of two parts, namely, logical topology discovery and network-wide 3D localization. In terms of the logical topology discovery, parallel MHRW is developed to improve the accuracy and efficiency of the discovery algorithm. For network-wide localization, a UAV-assisted 3D localization algorithm is proposed based on discovered logical topology and multidimensional scaling method, termed as Topo-MDS algorithm. Extensive simulations have been conducted to evaluate the efficiency and accuracy of the logical topology discovery, as well as the 3D localization. The results indicate that the parallel MHRW improves both the convergence rate and estimation accuracy, as compared to other benchmark methods. Besides, the 3D localization accuracy is also dramatically improved by the proposed Topo-MDS, as compared to the multi-lateration, MDS-based and MBL-MDS algorithms.

# Chapter 6

# UAV-Enabled Spatial Data Sampling in Large-Scale IoT Systems Using Denoising Autoencoder Neural Network

IoT technology has been pervasively applied to environmental monitoring, due to the advantages of low cost and flexible deployment of IoT enabled systems. In many large-scale IoT systems, accurate and efficient data sampling and reconstruction are among the most critical requirements, since this can relieve the data rate of trunk link for data uploading while ensuring data accuracy. To address the related challenges, a UAV enabled spatial data sampling scheme has been proposed in this chapter using denoising autoencoder (DAE) neural network. More specifically, a UAV-enabled edge-cloud collaborative IoT system architecture is firstly developed for data processing in large-scale IoT monitoring systems, where UAV is utilized as a mobile edge computing device. Based on this system architecture, the UAV-enabled spatial data sampling scheme is further proposed, where the wireless sensor nodes of large-scale IoT systems are clustered by a newly developed bounded-size K-means clustering algorithm. A neural network model, *i.e.*, DAE, is applied to each cluster for data sampling and reconstruction, by the exploitation of either linear or nonlinear spatial correlation among data samples. Simulations have been conducted and the results indicate that the proposed scheme has improved data reconstruction accuracy under the same sampling ratio without introducing extra complexity, as compared to the compressive sensing based method.

## 6.1 Introduction

With the advantages of low cost and flexible deployment, large-scale IoT systems have been widely applied to environmental monitoring [94]. A general architecture of such system consists of a large number of connected wireless sensor nodes and a cloud platform, where the sensor nodes as data collection layer are pervasively deployed in the target areas for environmental sensing and sampling, while the cloud platform is utilized as the remote data center for data processing and analysis [95].

However, considering the harsh environment of operation fields, wireless communications between sensor nodes are vulnerable to different kinds of obstacles and interference. Additionally, with the enlarging scale of the IoT system, the tremendous amount of data uploading imposes a heavy burden on the bandwidth requirement of the trunk link. Thus, accurate and efficient data sampling and reconstruction are among the most critical technical demands for the design and operation in the cloud-enabled IoT systems. In order to overcome this challenge, UAV has been introduced into the large-scale IoT systems as a mobile edge computing device [82]. Here the UAV-enabled edge device serves as the intermediate layer [35]. Given the special location of the intermediate layer, the UAV can support real-time responses for the sensor nodes and offload tasks from the cloud by preliminary data processing and analysis. Through the deployment of UAV, an edge-cloud collaborative IoT system architecture has been developed for data processing in large-scale IoT monitoring systems.

Based on this system architecture, a novel spatial data sampling scheme has been further proposed, which can reduce the amount of data sampled at sensor nodes and relieve the bandwidth requirement of the link between UAV and cloud. The principle behind the proposed scheme is the spatial and temporal correlation between sensor data. In a complex environment, the correlation between different types of physical sensor data is not as simple as linearity [96]. Therefore, a neural network model, *i.e.* denoising autoencoder (DAE) [97], is utilized in our work, which has the capability of compressing both linearly and nonlinearly correlated data.

The proposed sampling scheme consists of three phases, namely, system initialization, model training, and data sampling. During the first phase, a UAV hovers above the target area served by the large-scale IoT system and the cloud. All sensor nodes keep active and upload data to the cloud through UAV. Based on the collected data, sensor nodes are clustered by the newly developed bounded-size K-means clustering algorithm. In the second phase, certain sensor nodes within each cluster are selected as data sampling representatives. DAE models for the clusters are trained in the cloud. Parameters of encoders in DAE models are sent to the UAV, while parameters of decoders are kept in the cloud. In the phase of data sampling, data are sampled from selected representatives and then encoded by the UAV before being forwarded to the cloud. The full dataset is finally decoded and reconstructed in the cloud. With the support of cluster formation and UAV, the efficiency of data sampling can be improved. Performance evaluation is conducted, where compressive sensing as a conventional data sampling method in IoT systems is utilized as the benchmark method. According to the numerical results, the proposed scheme has dramatically improved the data reconstruction accuracy under the same sampling ratio without introducing additional computational complexity.

The contributions of this chapter are summarized as follows:

- A UAV-enabled edge-cloud collaborative IoT system architecture is developed for data processing in large-scale IoT systems, which overcomes the critical challenges of cloud-enabled IoT systems, including high latency, bandwidth overload and unstable connection to the cloud.

- A novel spatial data sampling scheme has been proposed for efficient data sampling and reconstruction in the large-scale IoT monitoring systems. In order to fully exploit the spatial data correlation, DAE neural network has been selected as the fundamental data sampling and reconstruction model. With DAE, the sampled data can be precisely reconstructed in the cloud. In the meantime, by locating the encoder in DAE at the UAV, the amount of data uploaded to the cloud is dramatically reduced and thus the burden on the trunk link is relieved.

- A novel bounded-size K-means clustering algorithm has been developed specifically for cluster formation and the cluster-based spatial data sampling in the proposed scheme. In the novel clustering algorithm, the lower and upper bounds of cluster size are predetermined, which considers the effect of cluster size on the intra-cluster communications and data sampling.

The remaining of this chapter is organized as follows. Section 6.2 summarizes the related work on spatial data sampling in IoT systems. In Section 6.3, DAE neural network model is detailed. The architecture of the UAV-enabled edge-cloud collaborative IoT system is developed in Section 6.4. The novel spatial data sampling scheme is then proposed in Section 6.5. Performance evaluation is conducted in Section 6.6. Finally, Section 6.7 concludes the work.

## 6.2 Related Work

Spatial correlation based data sampling in the remote sensing field has been well studied in recent years. According to the different fundamental models used, related work is classified into the following categories.

***Compressive Sensing (CS)*** is a data compression technique that can map high-dimensional data into the sparse domain by utilizing a random sensing matrix. In CS-based methods, the sensing field is considered as sparse domain, where data are sparsely sampled from the field and fully recovered at the receiver. Compressive data gathering was the first CS-based method for large-scale WSNs [98]. WSNs were deployed as the data collection layer of IoT systems. Data were converted to the sparse domain by DCT (discrete cosine transform) and compressed along the multi-hop routing path. In [99], the authors proposed a well-developed CS-based framework for data sensing, sampling and recovery, where PCA was used to generate the sparse domain. A cluster-based random sampling algorithm was proposed in [100]. The sparse matrix was generated at the sink by random sampling at both intra-cluster and inter-cluster levels.

As stated, several research efforts have been spared on CS-based data sampling in IoT systems, while the weaknesses of these methods are mostly due to the intrinsic constraints of CS technique. The application of CS is limited by the restricted isometry property. However, the sparse domain sometimes may not exist for data sampled from complex circumstances. Additionally, although mapping data into a special sparse domain can further compress data, the complexity of the data recovery algorithm will be dramatically increased as a result.

***Principal Component Analysis (PCA)*** is a linear correlation based feature extraction model. Therefore, PCA and variations of PCA based spatial data aggregation have been widely used in WSNs and IoT systems. In [101], distributed compressive-project PCA was proposed in cooperation with the second-order data-coupled clustering algorithm for efficient data collection in large-scale WSNs. Similarly, the authors in [102] proposed a cluster-based framework as well, aiming at outlier-free data aggregation in IoT systems. The difference was that recursive PCA was used in [102] for adaptively updating PCA models.

***Autoencoder (AE)*** is a neural network model for feature extraction, which can be considered as nonlinear PCA. Given the outstanding performance on data modeling and processing, neural network models have attracted attention from both industrial and academic institutions. In terms of spatial data sampling in large-scale IoT systems, AE has been used in replace of PCA given the nonlinear processing capability. In [103], the authors proposed a data compression algorithm with error bound guarantee, where data were spatially compressed by AE-based nonlinear feature extraction.

However, both PCA and AE based methods sample full dataset from the sensing field, and then spatially compress data at a cluster head or fusion center. By contrast, CS-based methods have the capability of sparsely sampling from the sensing field directly, so that both sampling and communication related processing and cost can be further saved. By exploitation of DAE, the proposed scheme can also sample a subset of data directly from the field. As compared to CS, the data reconstruction accuracy has been improved under the same sampling ratio.

## 6.3 Denoising Autoencoder Neural Network

The fundamental mathematical model behind the proposed scheme is DAE, which is a neural network model that can be used to reconstruct the full dataset from the sampled subset [97]. In this section, DAE is explained based on the introduction to basic AE.

### 6.3.1 Basic Autoencoder

AE is a neural network model for feature extraction. The difference to the PCA model is that AE has the capability of dealing with nonlinear data correlation. As a neural network model, AE is also consisted of input, hidden and output layers, while the special case is that the target output of AE is exactly its input. A general structure of AE with a single hidden layer is

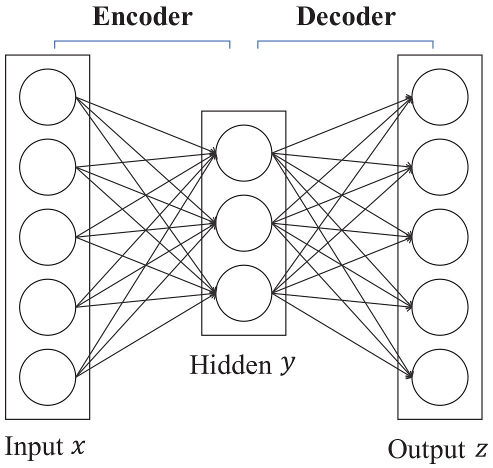


Figure 6.1: A general structure of the autoencoder neural network with a single hidden layer.

shown in Fig.6.1, where the projection from the input layer to the hidden layer is termed as the encoder, while the projection from the hidden layer to the output layer is termed as the decoder.

The mapping function of the encoder is expressed as

$$\mathbf{y} = f_\theta(\mathbf{x}) = f(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}_f), \tag{6.1}$$

where $\mathbf{x}$ is the input vector in $n$ dimensions, while $\mathbf{y}$ is the hidden layer readout with $k$ units. $f(\cdot)$ is a nonlinear activation function, and sigmoid function is generally adopted. $\mathbf{W}_{[k\times n]}$ is the input weight matrix, and $\mathbf{b}_f$ is the input bias vector.

Correspondingly, the mapping function of the decoder is given by

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = g(\mathbf{V} \cdot \mathbf{y} + \mathbf{b}_g), \tag{6.2}$$

where $\mathbf{z}$ is the output vector with the same dimension as input $\mathbf{x}$. $g(\cdot)$ is the activation function of the decoder. Both identity and sigmoid function are frequently used. $\mathbf{V}_{[n\times k]}$ is the output weight matrix, and $\mathbf{b}_g$ is the output bias vector.

To find out the optimal parameter sets $\theta = \{\mathbf{W}, \mathbf{b}_f\}$ and $\theta' = \{\mathbf{V}, \mathbf{b}_g\}$, the cost function of basic AE is given by

$$\mathcal{J}_{\theta,\theta'} = \frac{1}{m}\sum_{i=1}^{m} ||\mathbf{z}^{(i)} - \mathbf{x}^{(i)}||_2^2, \tag{6.3}$$

which penalizes the squared error between input $\mathbf{x}$ and output $\mathbf{z}$. $m$ is the size of training dataset.

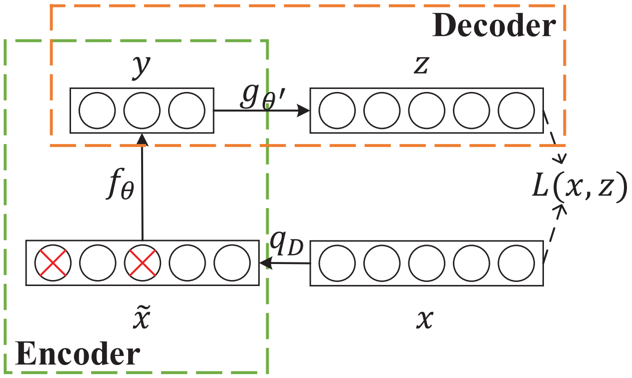


Figure 6.2: A general structure of the denoising autoencoder.

### 6.3.2 Denoising Autoencoder

Based on the basic AE, DAE is further proposed by *P. Vincent et al.* [97] to extract features and reconstruct original data from **corrupted** data as shown in Fig.6.2.

Original data $\mathbf{x}$ is corrupted to $\tilde{\mathbf{x}}$ by

$$\tilde{\mathbf{x}} = q_D(\mathbf{x}), \tag{6.4}$$

where $q_D$ is corruption function. In our data sampling scheme, $q_D$ is defined as a mask function that makes $\tilde{\mathbf{x}}$ a subset of $\mathbf{x}$.

As shown in Fig.6.2, the corrupted data vector $\tilde{\mathbf{x}}$ is encoded to $\mathbf{y}$ and then decoded to $\mathbf{z}$ by

$$\mathbf{y} = f_\theta(\tilde{\mathbf{x}}), \ \mathbf{z} = g_{\theta'}(\mathbf{y}). \tag{6.5}$$

Since the objective of DAE is to recover the original data $\mathbf{x}$ from the corrupted data $\tilde{\mathbf{x}}$, the cost function is defined as the squared error between original $\mathbf{x}$ and reconstructed $\mathbf{z}$ as

$$\mathcal{J}_{\theta,\theta'} = \frac{1}{m}\sum_{i=1}^{m} ||\mathbf{z}^{(i)} - \mathbf{x}^{(i)}||_2^2 = \frac{1}{m}\sum_{i=1}^{m} ||g_{\theta'}(f_\theta(\tilde{\mathbf{x}}^{(i)})) - \mathbf{x}^{(i)}||_2^2. \tag{6.6}$$

Mini-batch based gradient descent algorithm [104] is used to solve the problem and learn the parameters. Though the training procedure occupies certain computational load and memory, it is executed in the cloud platform and does not impose an additional burden on the sensor nodes nor the UAVs.

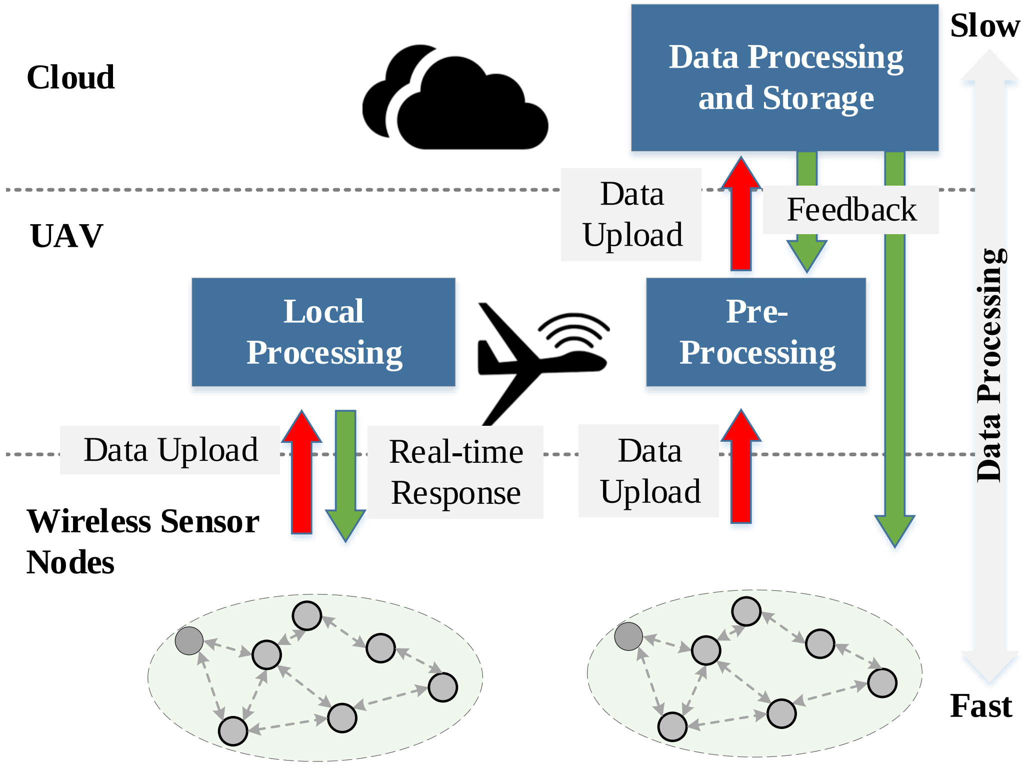


Figure 6.3: UAV-enabled edge-cloud collaborative architecture for data processing in large-scale IoT monitoring systems.

## 6.4 UAV-enabled Edge-Cloud Collaborative IoT System Architecture

A UAV-enabled edge-cloud collaborative IoT system architecture for data processing in large-scale IoT monitoring systems is developed as shown in Fig.6.3, which consists of three major components, namely, wireless sensor nodes as end devices, UAVs as mobile edge devices and IoT cloud platform. Details of each component are given below.

- *IoT cloud platform* is the remote data and control center for the IoT system, leveraging cloud computing to achieve complex data processing and analysis, cluster formation for wireless sensor nodes, as well as coordination of UAV flight paths. Particularly, since the training process of the DAE models is too complex to be loaded on either sensor nodes or UAVs, the parameter sets are learned through the training in the cloud. The parameters of encoders in DAE models are then sent to UAV for data encoding. The parameters of decoders are kept in the cloud for data reconstruction.

- *UAVs* are utilized as mobile edge computing devices, which can support both local processing for the local events with critical real-time requirements and preliminary processing to offload the computational tasks from the cloud so as to relieve the bandwidth requirements of the underlying trunk link. In terms of wireless communications, UAVs are able to carry different RF modules and support different protocols. For instance, UAVs have the capability of communicating with sensor nodes in a self-organized way through ZigBee modules and possibly serve as relays to forward the information to the cloud. Therefore, in the proposed scheme, UAV is utilized to collect and encode the sampled data before uploading them to the cloud. Depending on the service areas of the large-scale IoT systems, one or multiple UAVs could be used. Multiple UAVs can improve the efficiency of data sampling and encoding. However, the exploitation of multiple UAVs introduces more cost on device management and may also incur the issue of multiple-UAV cooperation to the IoT systems.

- *Wireless sensor nodes* are the fundamental components in IoT systems, which are normally deployed in the target areas in a random or predetermined way to sense and sample environmental information. For instance, in a forest fire surveillance system, temperature, smoke and humidity sensors are utilized for fire detection. These nodes are able to be self-organized into WSNs. Furthermore, a WSN is modeled as an undirected graph $G = (V, E)$ here. Sensor nodes are modeled as vertices $V$, and wireless communication links between nodes are modeled as edges $E$. The degree of a vertex is modeled by the number of valid neighbors of a sensor node. Only the nodes with valid wireless communication capability are defined as valid neighbors.

## 6.5 UAV-Enabled Spatial Data Sampling Scheme Using Denoising Autoencoder Neural Network

A UAV-enabled spatial data sampling scheme for large-scale IoT monitoring systems is proposed in this section. As stated in Algorithm 5, the scheme consists of three phases, namely, system initialization, model training, and data sampling. The dataflow in three phases is shown in Fig.6.4. More details are given in the following paragraphs.

**Algorithm 5** UAV-Enabled Spatial Data Sampling Using DAE

```
System Initialization:
set up UAV-IoT communication system
construct the physical topology of WSN in the cloud
UAV hovers above the target area as mobile relay and forwards raw data samples from
sensor nodes to the cloud
cluster sensor nodes by Algorithm 6
Model Training:
rank the link quality based on RSSI and LQI at UAV
select communication and data sampling representatives
send dissociation_notification to the remaining ones
train $\{\theta, \theta'\}$ with random masks $q_D$ in the cloud
send $\theta = \{\mathbf{W}, b_f\}$ to UAV for data encoding
Data Sampling:
collect data $\tilde{\mathbf{x}}$ from representatives to UAV
if RSSI or LQI is below threshold then
    trigger model training procedure
else
    encode data by $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$, and forward $\mathbf{y}$ to the cloud
end if
the original data is reconstructed by $g_{\theta'}(\mathbf{y})$ in the cloud
```

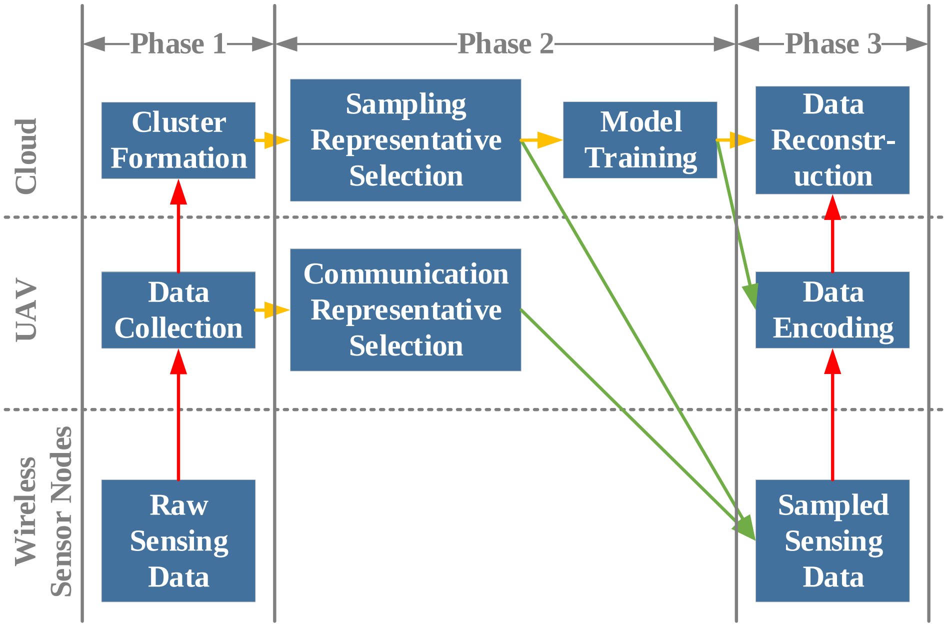


Figure 6.4: Dataflow of the UAV-enabled spatial data sampling scheme.

## 6.5.1 System Initialization

Wireless communications between the components in the IoT system are set up first. More specifically, wireless sensor nodes embedded with ZigBee RF modules are randomly deployed in the target area and self-organized into WSNs. UAV hovers above the target area, and wirelessly communicates with the nodes and the cloud through ZigBee and Wi-Fi, respectively.

### 6.5.1.1 Physical Topology Construction

Considering the randomness and self-organization features, the physical topology of the WSNs cannot be known in advance, which needs to be constructed in the cloud by the exploitation of the physical topology discovery scheme proposed in the previous work [25]. Physical topology provides the physical locations of sensor nodes and the logical topology of the WSNs.

### 6.5.1.2 Raw Data Collection

UAV keeps hovering above the target area and broadcasting beacon signal. According to IEEE802.15.4, sensor nodes would passively scan the channel, and send *association_request* to the UAV once the beacon signal is detected [105]. After the association is set up, raw data packets are transmitted from the sensor node to the UAV. UAV measures and records the RSSI (received signal strength indicator) and LQI (link quality indicator) of the received data packet, and then forwards the packet to the cloud.

### 6.5.1.3 Clustering

Based on the physical locations and raw data obtained in the first two steps, sensor nodes are clustered by the newly proposed bounded-size K-means clustering algorithm in the cloud. Pseudocode is listed in Algorithm 6. In the proposed clustering algorithm, sizes of generated clusters are bounded in the range [MIN_CZ, MAX_CZ], which are predetermined lower and upper bounds respectively.

Physical distance between locations and Euclidean distance between data are jointly utilized as the clustering criterion,

$$||\mathbf{l}_i - \mathbf{l}_{C_j}||_2 + \beta||\mathbf{d}_i - \mathbf{d}_{C_j}||_2 \leq \varepsilon, \tag{6.7}$$

where $\mathbf{l}_i$ and $\mathbf{d}_i$ are the location and data of sensor node $i$, while $\mathbf{l}_{C_j}$ and $\mathbf{d}_{C_j}$ indicate the average location and data centroids of cluster $j$. $\beta$ is the weight to balance these two metrics and $\varepsilon$ is the threshold. Particularly, all the collected data are normalized first to remove the impact of different scales.

**Algorithm 6** Bounded-Size K-means Clustering Algorithm

```
 1: Input: node set $S$, lower bound MIN_CZ, upper bound MAX_CZ, initial value and offset
    of $\varepsilon$ ($\varepsilon_{INI}$, $\varepsilon_{OFFSET}$)
 2: initialize $K = 1$, $\varepsilon = \varepsilon_{INI}$, $S_1$ as centroid of cluster 1
 3: while minimal cluster size < MIN_CZ do
 4:   for each node $S_i$ in $S$ do
 5:     for cluster $j = 1 : K$ do
 6:       if Eq.(6.7) satisfied and size of $j$ <MAX_CZ then
 7:         assign $S_i$ to cluster $j$, update centroids of $j$
 8:         break
 9:       end if
10:     end for
11:     if $S_i$ is not assigned to existing clusters then
12:       $K = K + 1$, and assign $S_i$ as centroid of cluster $K$
13:     end if
14:   end for
15:   $\varepsilon = \varepsilon + \varepsilon_{OFFSET}$
16: end while
17: Output: K and generated clusters
```

The procedure of cluster formation using Algorithm 6 is further explained as follows.

- The first cluster is formed up by regarding the location and data of the first sensor node as cluster centroids.

- For the remaining nodes in the network, if a node satisfies the clustering criterion of a cluster and the cluster size is not beyond the upper bound MAX_CZ (line 6 in Algorithm 6), the node is assigned to such cluster and the cluster centroids are updated with the new average values of location and data. If a node cannot be assigned to any existing clusters, a new cluster is formed with the location and data of such node as cluster centroids.

Table 6.1: Record of Link Quality

| Device ID | MAC Address | RSSI | LQI | Cluster ID |
|---|---|---|---|---|

- The proposed bounded-size K-means clustering algorithm is an iterative algorithm, the condition of termination is that the minimal cluster size of the generated clusters is larger than or equal to the lower bound MIN_CZ.

For the generated clusters, the dataset of cluster $j$ at time $t$, $\mathbf{x}_j^{(t)}$, is the concatenation of data from member sensor nodes, which is considered as the original data vector in DAE.

## 6.5.2 Model Training

Within each cluster, two types of representatives are selected for communication with the UAV and data sampling, respectively. Communication representatives are chosen by the UAV according to link quality, while data sampling representatives are determined by the cloud. Based on the selections, corresponding DAE models are trained for the clusters.

### 6.5.2.1 Communication Representative Selection

During the phase of system initialization, RSSI and LQI are measured and recorded at UAV as shown in Table 6.1.

RSSI and LQI are jointly used to evaluate the link quality, which is calculated as

$$quality = \frac{RSSI}{RSSI\_MAX} + \frac{LQI}{LQI\_MAX}, \tag{6.8}$$

where RSSI and LQI indicate the power strength of the received signal and the success of received packet demodulation respectively. In communication protocols such as IEEE802.11 and IEEE802.15.4, RSSI and LQI are both defined in range 0x00~0xFF, namely, RSSI_MAX=0xFF, LQI_MAX=0xFF, where a higher value indicates better quality. In practical applications, chipset manufacturers can self-define the value of RSSI_MAX and LQI_MAX. However, by scaling RSSI and LQI, the *quality* defined in (6.8) always ranges from 0 to 2 and 2 indicates the best link quality.

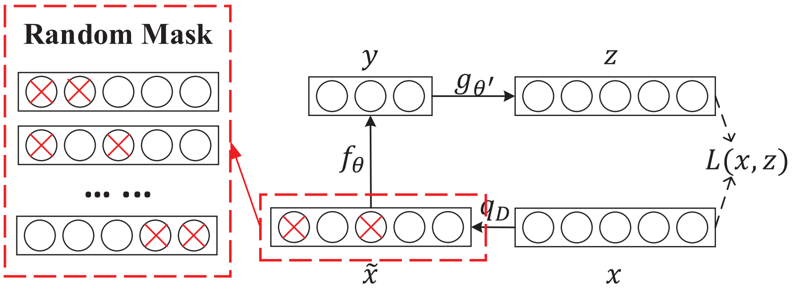


Figure 6.5: Masks are randomly generated for the training of DAE models.

Based on the ranking of *quality*, the sensor node with the best link quality in a cluster is selected as the communication representative. The working mode of the selected node is converted to the coordinator. The remaining sensor nodes within the same cluster upload data through the coordinator instead of communicating with UAV directly. In this way, the time duration of UAV-enabled data sampling can be reduced.

#### 6.5.2.2 Data Sampling Representative Selection

Given a cluster $j$ ($j = 1, 2, \dots, K$), $NC_j$ sensor nodes are contained. $NR_j$ out of $NC_j$ sensor nodes are selected as representatives for data sampling. Based on the knowledge of logical topology, the degree of each sensor node can be calculated. Within each cluster, order the member sensor nodes according to node degree. Node with the highest degree and the lowest $(NR_j - 1)$ ones are selected as data sampling representatives.

The communication representative only communicates with the selected sampling representatives for data uploading, and sends *disassociation_notification* to the remaining ones.

#### 6.5.2.3 Model Training

Random masks are generated to project original data vector $\mathbf{x}_j^{(t)}$ to subset $\tilde{\mathbf{x}}_j^{(t)}$, as shown in Fig.6.5. In terms of the masks, a fraction of original $\mathbf{x}_j^{(t)}$ would be dropped off, namely, $(NC_j - NR_j)$ out of $NC_j$ in $\mathbf{x}_j^{(t)}$ would be replaced by **nan** (not a number). Taking Fig.6.5 as an example, the original data vector is

$$\begin{aligned}\mathbf{x}_j^{(t)} = [\mathbf{d}_1^{(t)}; \mathbf{d}_2^{(t)}; \dots; \mathbf{d}_5^{(t)}] = [&d_{1,1}^{(t)}, d_{1,2}^{(t)}, \dots, d_{1,p_1}^{(t)},\\ &d_{2,1}^{(t)}, d_{2,2}^{(t)}, \dots, d_{2,p_2}^{(t)}, \dots\dots, d_{5,1}^{(t)}, d_{5,2}^{(t)}, \dots, d_{5,p_5}^{(t)}],\end{aligned} \tag{6.9}$$

and the sampling subset is

$$\begin{aligned}\tilde{\mathbf{x}}_j^{(t)} = [\mathbf{nan}; \mathbf{d}_2^{(t)}; \mathbf{nan}; \mathbf{d}_4^{(t)}; \mathbf{d}_5^{(t)}] = [&nan, \dots, nan, d_{2,1}^{(t)}, d_{2,2}^{(t)}, \dots, d_{2,p_2}^{(t)}, \\ &nan, \dots, nan, d_{4,1}^{(t)}, d_{4,2}^{(t)}, \dots, d_{4,p_4}^{(t)}, d_{5,1}^{(t)}, d_{5,2}^{(t)}, \dots, d_{5,p_5}^{(t)}],\end{aligned} \tag{6.10}$$

where $\mathbf{d}_i^{(t)}$ is the data vector generated by sensor node $i$ in cluster $j$ at time $t$, and $p_i$ is the number of measured physical variables. Namely, the dimension of $\mathbf{d}_i^{(t)}$ is $p_i$.

DAE model parameter sets $\{\theta_j, \theta'_j\}$ of cluster $j$ are learned by minimizing the cost function,

$$\mathcal{J}_{\theta_j,\theta'_j} = \frac{1}{m}\sum_{t=1}^{m} \|g_{\theta'_j}(f_{\theta_j}(q_{Dt}(\mathbf{x}_j^{(t)}))) - \mathbf{x}_j^{(t)}\|_2^2, \tag{6.11}$$

where $q_{Dt}$ is the mask randomly generated at time $t$. $f(\cdot)$ is sigmoid function, and $g(\cdot)$ is linear function. $m$ is the amount of historical data samples archived in the cloud for training. Mini-batch gradient descent algorithm is applied to solve (6.11). $\theta = \{\mathbf{W}, \mathbf{b}_f\}$ is sent to the UAV for data encoding. $\theta' = \{\mathbf{V}, \mathbf{b}_g\}$ is maintained in the cloud for data reconstruction.

## 6.5.3 Data Sampling

Dataflow of spatial data sampling and reconstruction has been shown in Fig.6.4. Data processing at each component is specifically provided as follows.

### 6.5.3.1 Data Sampling

Based on the clusters setup and representatives selected in Subsection 6.5.1 and 6.5.2, data are collected from the data sampling representatives to the communication representatives and then forwarded to the UAV.

### 6.5.3.2 Data Encoding

The collected data samples are encoded at the UAV by

$$\mathbf{y}^{(t)} = \frac{1}{1 + e^{-(\mathbf{W}\tilde{\mathbf{x}}^{(t)} + \mathbf{b}_f)}}, \tag{6.12}$$

where $\mathbf{W}$ and $\mathbf{b}_f$ are the parameters obtained from the training in Subsection 6.5.2.3. $\mathbf{y}^{(t)}$ is forwarded to the cloud.

Simultaneously, RSSI and LQI of the received data packet are evaluated as well. If either RSSI or LQI is below a pre-defined threshold, a warning is sent to the cloud. The model training procedure is re-triggered cooperatively by UAV and cloud.

#### 6.5.3.3 Data Reconstruction

In the cloud platform, data from each cluster is reconstructed by

$$\mathbf{z}^{(t)} = \mathbf{V}\mathbf{y}^{(t)} + \mathbf{b}_g, \tag{6.13}$$

where $\mathbf{V}$ and $\mathbf{b}_g$ are the parameters learned and maintained from the training in Subsection 6.5.2.3.

## 6.6 Performance Evaluation

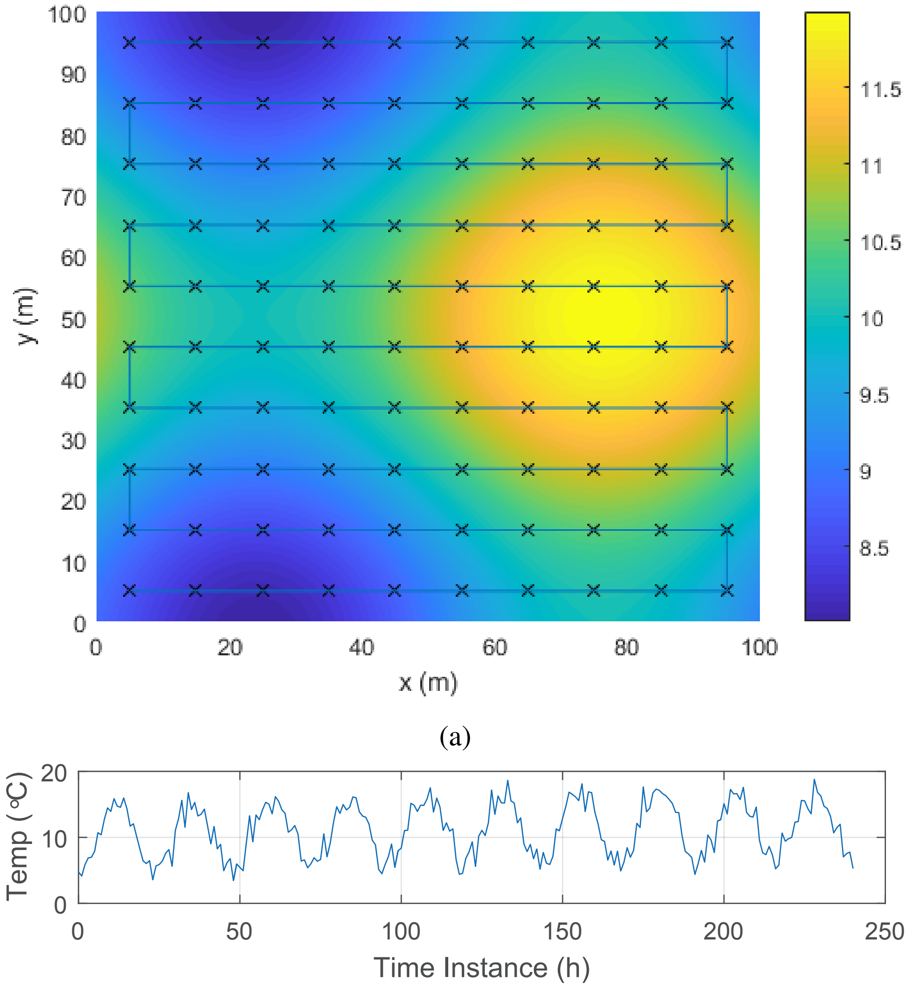
Simulations are conducted in this section to analyze the clustering result and accuracy of final data reconstruction, based on the simulation settings given in Subsection 6.6.1.

### 6.6.1 Simulation Settings

#### 6.6.1.1 Fundamental Settings

Fig.6.6 shows both the geographical distribution and temporal variance of the temperature field. Fig.6.6(a) is a 100m×100m field, where temperature varies continuously. The temporal trend in Fig.6.6(b) indicates the variance of the mean value of the geographical temperature field within 10 days. The unit of the horizontal axis in Fig.6.6(b) is an hour. 100 sensor nodes are randomly deployed in the area (not shown). The altitude coordinate of a sensor node is the height of the deployed location. The transmitting power of sensor nodes is homogeneously set to -10dBm and the receiver sensitivity is -90dBm.

UAV flight path is also demonstrated in Fig.6.6(a). UAV hovers above the target area with



(a)

(b)

Figure 6.6: Geographical distribution (a) and temporal variance (b) of the temperature field.

an even interval. The hovering interval has a direct influence on the localization accuracy [25] but does not have much effect on the following investigations. Hence, the interval is set to 10m without losing generality. The hovering height is 20m above the field. The hovering bias is ±1.5m in latitude and longitude and ±0.5m in altitude.

#### 6.6.1.2 Wireless Communication Channel Models

For the signal propagation from UAV to sensor nodes and peer-to-peer channels among sensor nodes, two-ray ground and free-space outdoor models are respectively used, considering the different signal propagation environments.

For the air-to-ground signal propagation from UAV to the sensor node, the two-ray ground model is commonly used, which considers both the line-of-sight and ground-reflected rays. For wireless communications among sensor nodes, the signal propagation channel quality is worse, given the potential near-ground scatters. Instead of the two-ray ground model, the free-space outdoor model is thus adopted. This is a channel model designed specifically for WSNs in the outdoor open areas, which jointly considers the effect of the free-space propagation, ground reflection, RSS uncertainty, and antenna radiation impact.

**Two-ray Ground Model** For large distance $d$, the received power $P_r$ (in dBm) can be derived by the two-ray ground model as [89],

$$P_r(dBm) = P_t + 10log(G_tG_r) + 20log(H_tH_r) - 40log(d), \tag{6.14}$$

where $P_t$ is the transmitting power. $d$ is the horizontal distance between transmitter and receiver. $G_t$ and $G_r$ are the antenna gains of transmitter and receiver, $G_t = G_r = 1$. $H_t$ and $H_r$ are the antenna heights of transmitter and receiver.

**Free-Space Outdoor Model** The received power is modeled as [90],

$$P_r(dBm) = P_t + 20log(\frac{\lambda}{4\pi d}) + 10log(K_1^2 + K_2^2\Gamma^2 + 2K_2\Gamma cos(\frac{2\pi}{\lambda}\Delta L)) + X_\sigma, \tag{6.15}$$

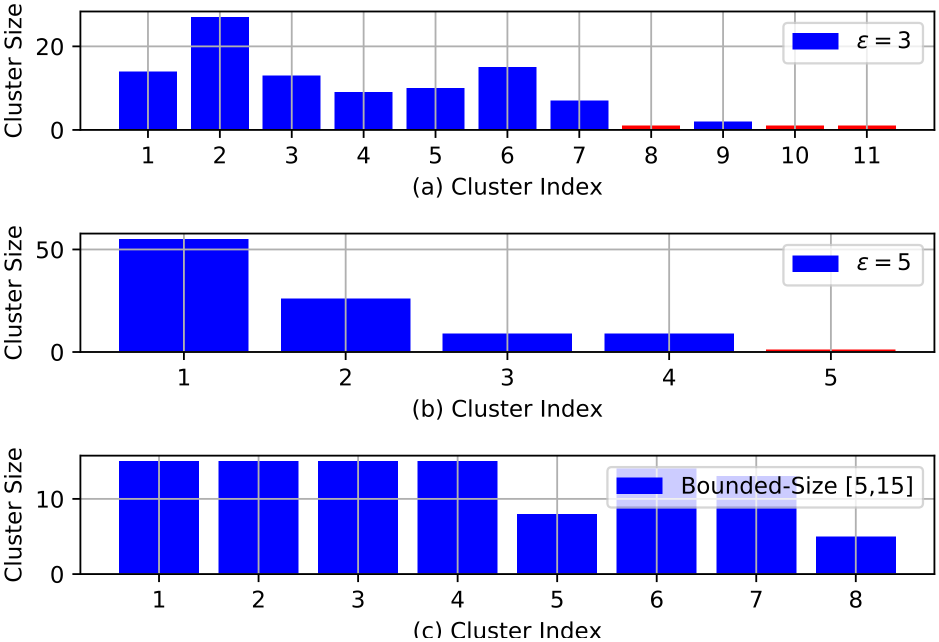


Figure 6.7: Comparison between traditional threshold-based clustering algorithm (a) $\varepsilon$=3 (b) $\varepsilon$=5 and the proposed bounded-size K-means clustering algorithm (c) [5, 15] and $\varepsilon_{INI}$=3.

where $\lambda$ is the propagation wavelength, and $K_1$ and $K_2$ are coefficients irregularity in antenna radiation pattern. $\Delta L$ is the path difference between LOS and ground-reflected rays. $X_\sigma$ is the RSS uncertainty that follows Gaussian distribution. $\Gamma$ is the ground reflection coefficient,

$$\Gamma = \frac{\sin\theta - \sqrt{(\varepsilon - jx_\Gamma) - \cos^2\theta}}{\sin\theta + \sqrt{(\varepsilon + jx_\Gamma) - \cos^2\theta}}, \tag{6.16}$$

where parameters of average ground are used without losing generality, $\varepsilon = 15, x_\Gamma = 3.75 \times 10^{-2}$. $\theta$ is the reflection angle.

### 6.6.2 Clustering Analysis

The proposed bounded-size K-means clustering algorithm is analyzed in this subsection. Since the proposed algorithm is threshold-based, the traditional threshold-based clustering algorithm [56] is selected as the benchmark. The improvement of the proposed clustering algorithm as compared to the benchmark method is firstly provided. Influence of the parameters including lower bound, upper bound, $\varepsilon_{INI}$ and $\varepsilon_{OFFSET}$ on the clustering results is further investigated.

With the traditional clustering algorithm, when the threshold $\varepsilon$ is set to 3, the number of

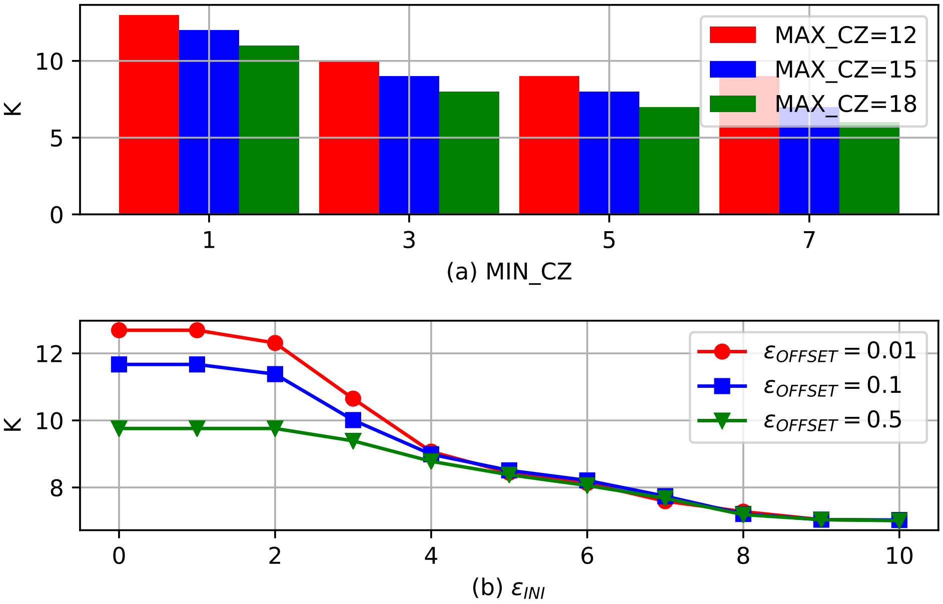


Figure 6.8: Influence of the parameters on clustering results: (a) upper bound (MAX_CZ) and lower bound (MIN_CZ); (b) initial value and offset of clustering threshold $\varepsilon$ ($\varepsilon_{INI}$ and $\varepsilon_{OFFSET}$).

sensor nodes in each of the generated clusters is shown in Fig.6.7(a), which illustrates that eleven clusters are generated and three of them contain only a single node as highlighted in red. When $\varepsilon = 5$, five clusters are generated and there is one cluster containing a single node as shown in Fig.6.7(b). The results in Fig.6.7(a) and (b) indicate that with the increment in threshold $\varepsilon$, the number of clusters with single node decreases indeed. However, it may result in some huge clusters in the meantime. The huge cluster refers to the cluster with an extremely large amount of sensor nodes, for example, in Fig.6.7(b), cluster 1 containing 55 sensor nodes.

In our proposed data sampling scheme, the communication representative in each cluster is functioned as a coordinator and directly communicates with the UAV, while the other cluster members communicate with the coordinator locally. Therefore, in the huge clusters, the intra-cluster communications would be overload with multiple hops and also vulnerable to environmental interference. In addition to the huge clusters, in the cluster with a single node, the single node has to be regarded as both data sampling representative and communication representative in the meantime, which can result in the early death of such node. Hence, we have added new attributes in the proposed clustering algorithm, namely, the upper and lower bounds of cluster size [MIN_CZ, MAX_CZ]. As shown in Fig.6.7(c), when the bounds are set to $[5, 15]$, $\varepsilon_{INI} = 3$, and $\varepsilon_{OFFSET} = 0.01$, sizes of the generated clusters are more balanced.

Influence of the parameters on clustering results is shown in Fig.6.8, where (a) shows the effect of the lower and upper bounds [MIN_CZ, MAX_CZ] with $\varepsilon_{INI} = 3$ and $\varepsilon_{OFFSET} = 0.01$, while (b) shows the influence of $\varepsilon_{INI}$ and $\varepsilon_{OFFSET}$ with MIN_CZ=2 and MAX_CZ=15. From Fig.6.8(a) it can be seen that with the increment in MIN_CZ, the number of clusters generated (namely, K in Fig.6.8) decreases, which is due to the iteratively increased threshold $\varepsilon$. In addition, given the fixed MIN_CZ, with the increment in MAX_CZ, the number of clusters generated reduces, which is because the clustering result is mainly affected by the setting of MAX_CZ in such condition. From Fig.6.8(b) we can notice that with the increment in $\varepsilon_{INI}$, the number of clusters generated decreases. In the meantime, with the increasing $\varepsilon_{OFFSET}$, the value of K converges faster. The reason is that with a higher threshold $\varepsilon$, more sensor nodes would satisfy the threshold and be gathered into the same cluster and the "huge" clusters are then bounded by MAX_CZ. Overall, the clustering result is jointly affected by these parameters, which need to be seriously predetermined by the requirements of specific applications.

### 6.6.3 Data Reconstruction Analysis

Data reconstruction accuracy is investigated in this subsection. Bounds on the clustering algorithm are set to [2,15], $\varepsilon_{INI} = 3$, $\varepsilon_{OFFSET} = 0.01$ and $\beta = 0.1$, and 10 clusters are generated. The DAE model of each cluster is trained by the mini-batch gradient descent algorithm, where the batch size is set to 48. The length of the training dataset is 480 (about 20 days), while the length of the testing dataset is 120 (5 days).

Fig.6.9 is demonstrated as an example, which shows the original temperature readings from 15 sensor nodes within a cluster (labeled 1~15), and also the sampled and reconstructed values. It indicates that with 12 sensor nodes selected as data sampling representatives, the reconstructed data can have an accurate approximation of the original data.

In order to quantatively evaluate the reconstruction accuracy, *data reconstruction error* is defined by the average squared $l2$-norm of difference between reconstructed and original data,

$$error = \frac{1}{T}\sum_{t=1}^{T} ||\mathbf{z}^{(t)} - \mathbf{x}^{(t)}||_2^2, \tag{6.17}$$

where $\mathbf{z}$ and $\mathbf{x}$ are reconstructed and original data vectors. $T$ is the length of the testing dataset.

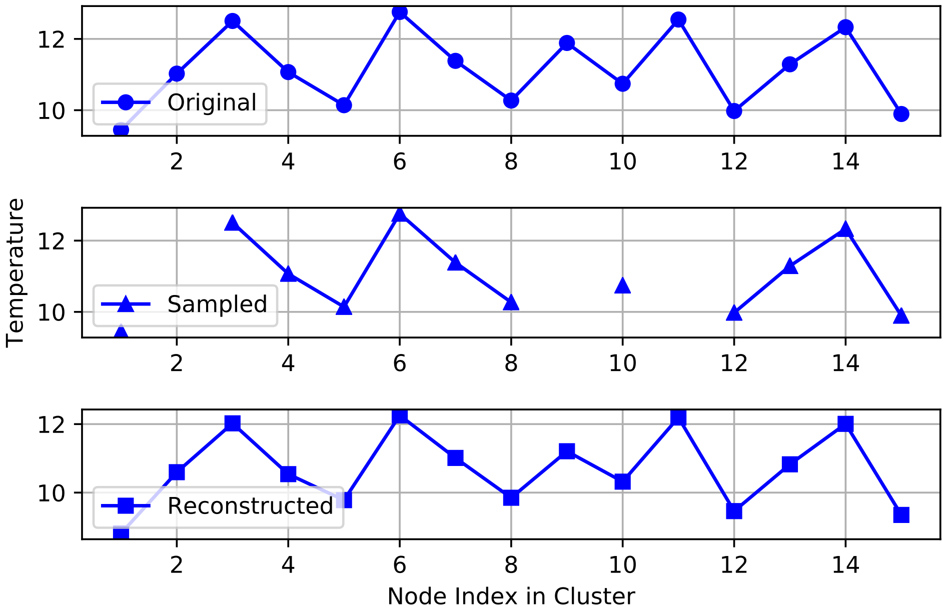


Figure 6.9: Original, sampled, and reconstructed temperature values (°$C$) from 15 sensor nodes within a cluster.

Table 6.2: Comparison between Different Data Sampling Representative Selection Criteria

| Method | Proposed | Highest | Lowest | Random |
|---|---|---|---|---|
| Sampling Ratio = 0.6 | 0.0943 | 0.1056 | 0.1003 | 0.1206 |
| Sampling Ratio = 0.7 | 0.0217 | 0.0277 | 0.0241 | 0.0249 |
| Sampling Ratio = 0.8 | 0.0137 | 0.0140 | 0.0146 | 0.0165 |

#### 6.6.3.1 Data Sampling Representative Selection Analysis

As proposed in Subsection 6.5.2.2, the node with the highest degree and the nodes with the lowest degrees in each cluster are selected as the data sampling representatives. Data reconstruction error generated by using the proposed selection criterion is evaluated here, as compared to other selection criteria, including the selection of nodes with highest degrees, selection of nodes with lowest degrees and random selection. Comparison under different sampling ratios is listed in Table 6.2, where the *sampling ratio* refers to the ratio of the number of representatives over the total number of sensor nodes in the cluster.

It can be seen that the data reconstruction error dramatically decreases with the increment in the sampling ratio, while for different selection criteria the difference in error is trivial. The reason is that during the training procedure of the DAE model, random masks are used. Therefore, from the perspective of data reconstruction, there is only a minor difference between

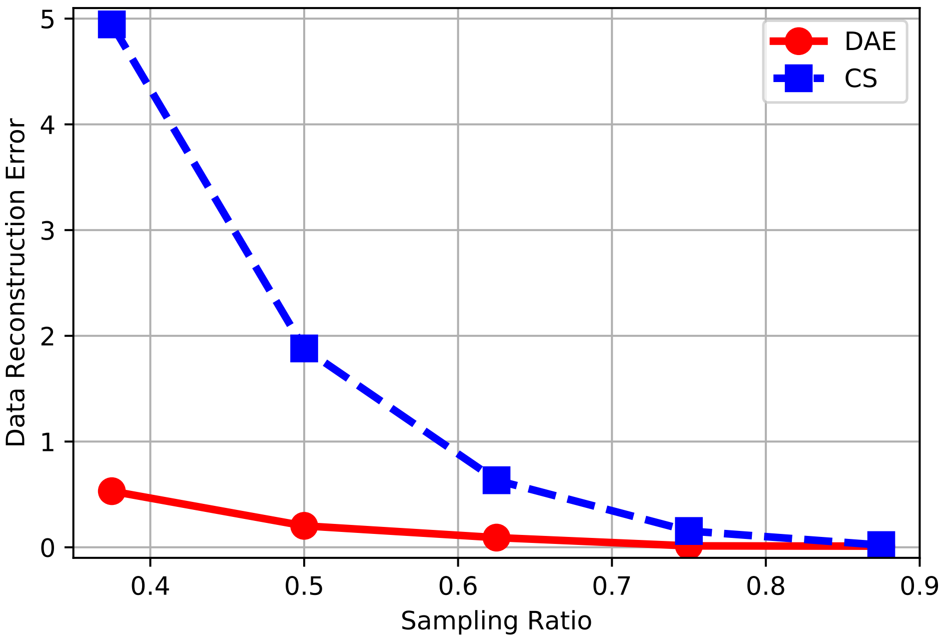


Figure 6.10: Comparison on the data reconstruction error between the proposed DAE-based scheme and the CS-based method under different sampling ratios.

these selection criteria. The proposed scheme mainly concerns the physical meanings of the data samples in the actual applications. In the clusters of sensor nodes, the node with the highest degree is located at the hot spot of the cluster and can represent its densely distributed neighbor nodes, while the nodes with lowest degrees are possibly located at the edge of the cluster or the area with sparse node distribution which can hardly be represented by others. That is the reason why the data samples measured by these nodes are collected.

#### 6.6.3.2 Comparison with Compressive Sensing

Comparison on the error generated by two methods under different sampling ratios is shown in Fig.6.10, where DAE represents our proposed scheme and CS refers to the CS-based benchmark method. It can be seen that with the increment in the sampling ratio, the data reconstruction error decreases. The reason is that with a higher sampling ratio, the uncertain proportion of collected data is less, which further improves the reconstruction accuracy. Additionally, the error curves of “DAE” and “CS” indicate that the proposed scheme outperforms the CS-based method. Especially when the sampling ratio is low as 0.375, the data reconstruction error of the proposed DAE-based scheme is 89.3% less than that of the CS-based method.

In terms of the complexity analysis, the computational complexity of the CS-based method is dominated by the recovery algorithm. Therefore, the overall complexity is determined by the selection of the recovery algorithm. In our simulation, the iterative reweighted least squares (IRLS) algorithm is exploited for data recovery [106]. While for the DAE-based method, the computational complexity of the proposed method is dominated by the model training procedure, where the mini-batch gradient descent (GD) algorithm is used to learn the parameters. IRLS and mini-batch GD are both iterative algorithms. IRLS algorithm needs fewer iterations to converge, while the cost of IRLS at each iteration is higher [107]. Therefore, the comparison on the computational complexity between IRLS and mini-batch GD is determined by the features of data.

## 6.7 Chapter Summary

In order to address the challenge of accurate and efficient data sampling and reconstruction in large-scale IoT systems, a cluster-based spatial data sampling scheme has been proposed using DAE neural network, by the exploitation of the spatial data correlation. UAV is utilized as the mobile edge device and an edge-cloud collaborative data processing architecture is then developed, where wireless sensor nodes and the cloud platform are involved for environmental sensing and complex data analysis respectively. In order to form up suitable clusters for the proposed data sampling scheme, a novel bounded-size K-means clustering algorithm is proposed. A neural network model, DAE, is adopted to fully exploit the spatial data correlation and perform data sampling and reconstruction for each cluster. More specifically, the encoders in DAE models are deployed at the UAV for encoding the data collected from sampling representatives, while the decoders are located in the cloud for data reconstruction. Simulations have been conducted, where the proposed bounded-size K-means clustering algorithm and spatial data sampling scheme are both investigated. Numerical results indicate that the proposed scheme improves the data reconstruction accuracy under the same sampling ratio, as compared to the compressive sensing based method.

# Chapter 7

# Autoencoder Neural Network-based Data Outlier Detection in Edge-Cloud Collaborative IoT Systems

Due to the advantages of low cost and easy deployment, IoT systems have been pervasively deployed for large-scale environmental monitoring, where a huge number of IoT end devices are involved. In such systems, the cloud computing platform is generally utilized as the remote data and control center. However, the huge number of IoT end devices generate a massive amount of data, which brings huge challenges to the systems on the underlying network bandwidth of the trunk link and real-time data analytics. In order to overcome these challenges, an edge-cloud collaborative IoT system architecture is proposed in this chapter for the large-scale environmental monitoring, where edge computing is the intermediate layer. Edge computing supported by edge devices can provide local and real-time processing to the end devices, and can also provide preliminary data analytics to offload the computational tasks from the cloud and reduce the amount of data uploading. Based on the proposed system architecture, an autoencoder (AE) neural network-based data outlier detection scheme is newly developed, where the spatial correlation of data can be fully utilized to improve the data outlier detection accuracy by using AE. Performance evaluation has been conducted based on the practical oceanic atmospheric data. Simulation results indicate that the developed scheme can detect the data outlier accurately.

## 7.1 Introduction

With the rapid development of IoT technology, IoT systems have been widely used in environmental monitoring. Thus, the number of IoT end devices involved in the systems increases in an explosive trend, which consequently generates a massive amount of IoT data. Data processing, analysis, and storage of the massive amount of IoT data bring huge technical challenges to the IoT systems on the network bandwidth, real-time analytics, and connectivity stability [10]. To address the issue, edge computing has been integrated into large-scale IoT monitoring systems. In the newly developed edge-cloud collaborative IoT systems, edge computing supported by edge devices, such as smart gateways, lightweight servers, and base stations, is served as the intermediate layer, which is closer to the IoT end devices than the remote cloud platform. Therefore, the edge devices can provide local and real-time processing to the end devices, so that the delay incurred by the interaction between end devices and the remote cloud platform can be reduced. On the other side, the edge devices can also provide preliminary data analytics, so that the amount of data uploaded to the cloud platform can be reduced and the bandwidth burden on the trunk link can be relieved.

In this chapter, the edge-cloud collaborative IoT system architecture for large-scale environmental monitoring is firstly proposed, which is composed of wireless sensor nodes, edge devices, and the cloud computing platform. The functions of each component and the inter-communications are explained in detail. Based on the system architecture, a novel data outlier detection algorithm using AE neural network is further developed.

In large-scale IoT monitoring systems, data outlier refers to the data that do not follow the normal pattern or trend in either spatial or temporal domains. Several factors can incur the data outliers, such as the abnormal events in the target monitoring area and the inner malfunctions of sensor nodes. Data outliers can lead the data-driven IoT systems into unsafe conditions. Therefore, it is necessary to detect the data outliers timely and accurately. Considering the capabilities of edge devices and their special locations in the IoT systems, a novel edge computing enabled data outlier detection algorithm is proposed. An artificial neural network model (*i.e.*, AE) is used to fully exploit the spatial correlation of the environmental monitoring data. The reason why AE is used is the complexity of the monitored environment. Although the collected data has strong regularity, it may not be linearly correlated. As a neural network, AE can deal with both linearly and nonlinearly correlated data. Considering the computational complexity of neural network model training, the training process is executed at the cloud and the obtained models and parameters are sent back to the edge devices for data outlier detection. At the edge devices, the data outliers are detected through the large fluctuations generated by the process of AE encoding and decoding. Simulations have been conducted based on practical oceanic atmospheric data, where the numerical results indicate that the AE based data outlier detection algorithm proposed in this chapter can detect the data outliers accurately.

The remaining of this chapter is organized as follows. In Section 7.2, the edge-cloud collaborative IoT system architecture is detailed. The AE-based data outlier detection algorithm is then developed in Section 7.3. In Section 7.4, performance evaluation based on the practical oceanic atmospheric database is conducted. Finally, Section 7.5 concludes the work.

## 7.2 Edge-Cloud Collaborative IoT System Architecture

A general edge-cloud collaborative IoT system architecture for large-scale environmental monitoring is developed as shown in Fig.7.1, which consists of three major components, namely, wireless sensor nodes, edge devices, and the cloud computing platform. Functions of each component and the intercommunications are given below.

- *Wireless sensor nodes* are the most fundamental and important components of the large-scale IoT monitoring systems. The wireless sensor nodes are randomly deployed in the monitoring area, and self-organized into WSNs to timely sense and collect the environmental information. For example, in the forest fire surveillance system GreenOrbs, wireless sensor nodes are deployed on the trees, where each node is built with sensors including temperature, humidity, light intensity, and carbon dioxide titer to monitor the forest and detect the forest fire [108]. In the edge-cloud collaborative IoT system architecture, the wireless sensor nodes mainly communicate with the edge devices, and the communication with the cloud platform is also relayed by the edge devices. On the one hand, requirements on the communication capabilities of wireless sensor nodes can be reduced. On the other hand, data processing capabilities of the edge devices can be fully utilized.

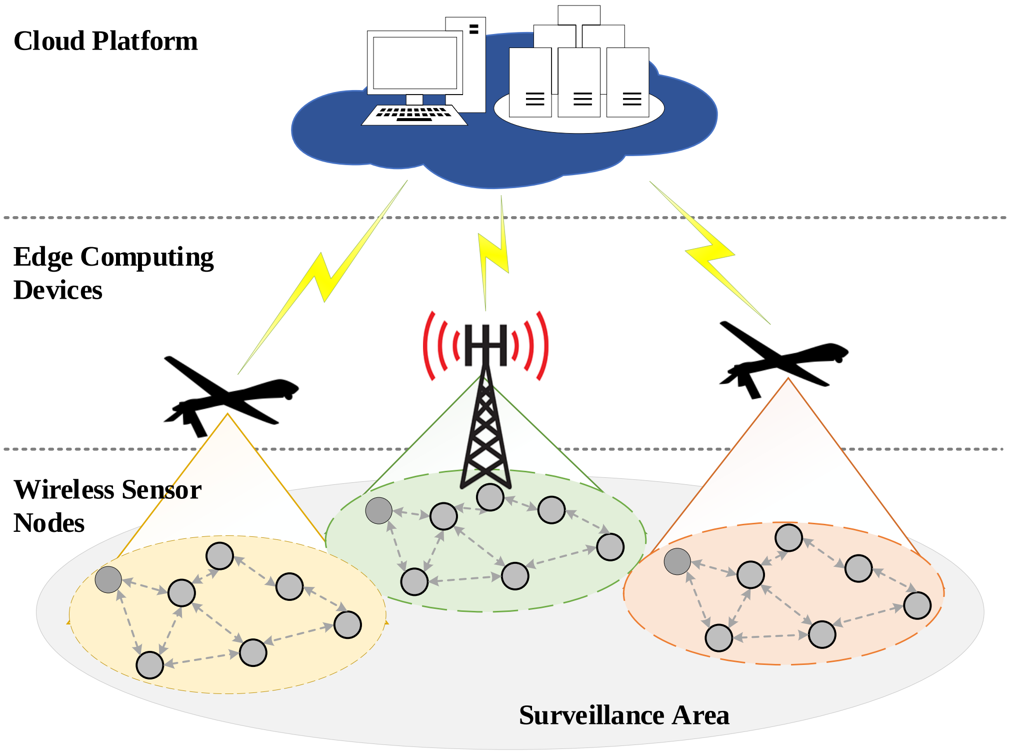


Figure 7.1: A general architecture of edge-cloud collaborative IoT systems.

- *Edge devices* In a large-scale IoT monitoring system, ground base stations deployed around the wireless sensor nodes can serve as edge devices. While in some inaccessible areas where the infrastructures of base stations can hardly be deployed, UAVs can be utilized as mobile edge devices to provide edge computing services. From the perspective of system architecture, the edge devices play the role of the intermediate layer, which can provide local and real-time processing services to the wireless sensor nodes so that the delay incurred by the communication with the remote cloud platform can be reduced. In the meantime, it can also provide preliminary data processing, which can offload the computing tasks from the cloud platform, and also reduce the amount of data uploaded to the cloud platform to relieve the bandwidth burden on the trunk link.

In the proposed data outlier detection algorithm, edge devices firstly serve as relay to forward the environmental monitoring data collected by the wireless sensor nodes to the cloud platform. Afterwards, the data outlier detection is executed at the edge devices. Whenever the data are uploaded to the edge devices, the data are identified as normal or outlier. Once a data outlier is detected, the edge device sends an outlier warning and the raw data to the cloud.

- *Cloud platform* serves as the remote data and control center in the systems responsible for comprehensive data analytics and massive data storage, given its superior computing power and huge storage space. In the proposed data outlier detection algorithm, an AE model is used to extract and exploit the spatial correlation of the environmental monitoring data. As an artificial neural network model for feature extraction, the model training process of AE is in high computational complexity. Wireless sensor nodes and edge devices can hardly provide the required computing power and resource consumption. Therefore, the model training process is executed at the cloud. The obtained model parameters are sent back to the edge devices for data outlier detection.

## 7.3 Autoencoder based Data Outlier Detection

In this section, the definition of data outlier in the IoT systems is firstly given. Afterwards, the structure of AE is explained. Finally, the AE based data outlier detection algorithm is proposed.

### 7.3.1 Data Outlier

In IoT systems, data outlier refers to the sensor data that do not follow the normal trend, which means the data that do not conform to the regularity of sensor data in temporal or spatial domains [33]. Fig.7.2 shows the sea surface temperature data measured from 7 different monitoring stations at 170W on the Pacific Ocean by the Tropical Atmosphere Ocean (TAO) project supported by the National Oceanic and Atmospheric Administration (NOAA). The normal temperature range is between 28°C and 30°C, and the variation is mild. However, when a data outlier occurs, the temperature measurement abruptly decreases to -9.99°C. Many factors can lead to data outliers, which may be abnormal events in the monitoring area such as forest fire, the inner malfunctions of the sensor nodes such as the damage on hardware modules and low power, and even the interference during wireless communications. In this chapter, only the data outlier detection is focused, while the reasons behind the data outliers are not diagnosed.

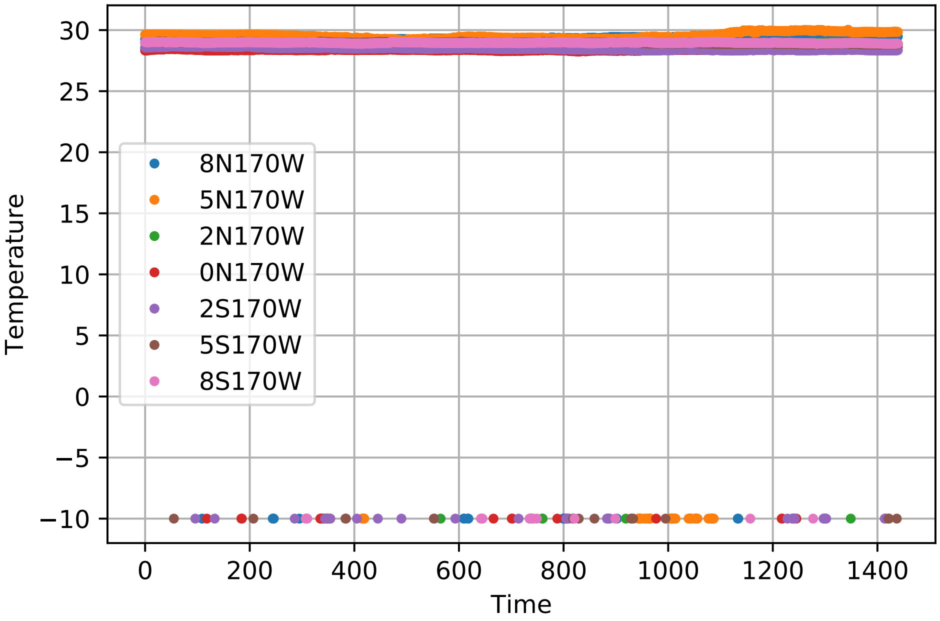


Figure 7.2: Sea surface temperature measurements from 7 monitoring stations at 170W in the TAO project.

### 7.3.2 Basic Autoencoder

AE is a neural network model for feature extraction. The difference to the PCA model is that AE has the capability of dealing with nonlinear data. As a neural network model, AE is also consisted of input, hidden and output layers, while the special case is that the target output of AE is exactly its input. Particularly, the projection from the input layer to the hidden layer is termed as the encoder, while the projection from the hidden layer to the output layer is termed as the decoder.

The mapping function of the encoder is expressed as

$$\mathbf{y} = f_\theta(\mathbf{x}) = f(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}_f), \tag{7.1}$$

where $\mathbf{x}$ is the input vector in $n$ dimensions, while $\mathbf{y}$ is the hidden layer readout with $k$ units. $f(\cdot)$ is a nonlinear activation function, and sigmoid function is generally adopted. $\mathbf{W}_{[k\times n]}$ is the input weight matrix, and $\mathbf{b}_f$ is the input bias vector.

Correspondingly, the mapping function of the decoder is given by

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = g(\mathbf{V} \cdot \mathbf{y} + \mathbf{b}_g), \tag{7.2}$$

where $\mathbf{z}$ is the output vector with the same dimension as input $\mathbf{x}$. $g(\cdot)$ is the activation function of the decoder. Both identity and sigmoid function are frequently used. $\mathbf{V}_{[n\times k]}$ is the output weight matrix, and $\mathbf{b}_g$ is the output bias vector.

To find out the optimal parameter sets $\theta = \{\mathbf{W}, \mathbf{b}_f\}$ and $\theta' = \{\mathbf{V}, \mathbf{b}_g\}$, the cost function of basic AE is given by

$$\mathcal{J}_{\theta,\theta'} = \frac{1}{m}\sum_{i=1}^{m} ||\mathbf{z}^{(i)} - \mathbf{x}^{(i)}||_2^2, \tag{7.3}$$

which penalizes the squared error between input $\mathbf{x}$ and output $\mathbf{z}$. $m$ is the size of training dataset.

### 7.3.3 Proposed Autoencoder based Data Outlier Detection Algorithm

**Algorithm 7** Proposed Autoencoder based Data Outlier Detection Algorithm

```
System Initialization:
upload raw data samples from sensor nodes to the cloud
Model Training:
normalize $\mathbf{X} \Rightarrow \overline{\mathbf{X}}$
cluster wireless sensor nodes
train models and obtain parameter sets $\theta = \{\mathbf{W}, b_f\}$ and $\theta' = \{\mathbf{V}, b_g\}$
send clustering results and parameter sets $\theta = \{\mathbf{W}, b_f\}$ and $\theta' = \{\mathbf{V}, b_g\}$ to edge devices
Data Outlier Detection:
upload data $\mathbf{x}$ to edge devices and normalize to $\overline{\mathbf{x}}$
calculate $\varepsilon = ||g_{\theta'}(f_\theta(\overline{\mathbf{x}}^{(t)})) - \overline{\mathbf{x}}^{(t)}||_2^2$
if $\varepsilon > \xi$ then
    trigger warning and upload $\mathbf{x}$ to the cloud
else
    upload $f_\theta(\mathbf{x}^{(t)})$ to the cloud
end if
```

In this subsection, the AE-based data outlier detection algorithm is proposed and explained in detail. Fig.7.3 shows the dataflow of the proposed data outlier detection algorithm in the edge-cloud collaborative IoT systems. As shown in Fig.7.3, the algorithm consists of three main phases, namely, system initialization, model training, and data outlier detection.

#### 7.3.3.1 System Initialization

During the phase of system initialization, the wireless sensor nodes indicate their existence by broadcasting the beacon signal and self-organized into WSNs. Afterwards, the wireless sensor

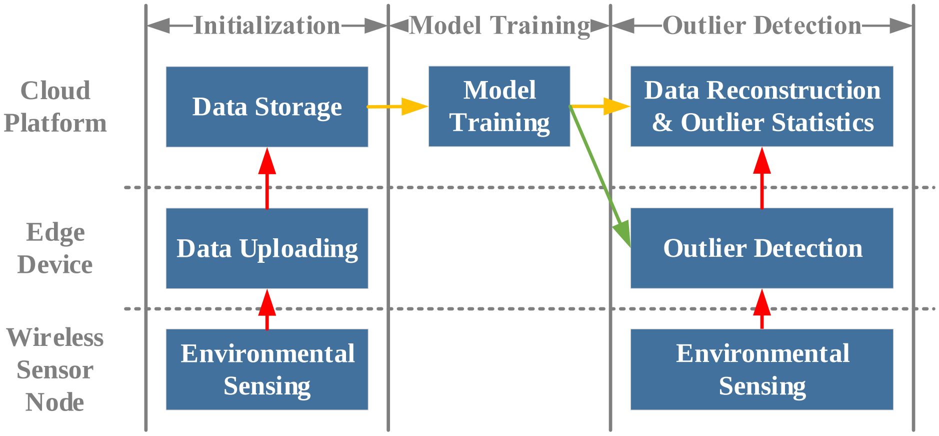


Figure 7.3: Dataflow of the AE-based data outlier detection algorithm.

nodes periodically sense and collect the environmental monitoring data, where the data vector generated by node *i* at the time instance *t* is

$$\mathbf{x}_i^{(t)} = [x_{i,1}^{(t)}, x_{i,2}^{(t)}, \ldots, x_{i,p}^{(t)}], \tag{7.4}$$

where *p* is the number of physical variables measured by node *i*.

The data are regularly uploaded to the cloud platform through the edge device that is closest to the node. The data are then stored in the cloud platform. During this phase, the edge devices only function as relays, which are responsible for collecting the sensing data and uploading the data directly to the cloud without local processing or other pre-processing operations.

#### 7.3.3.2 Model Training

The model training process is executed in the cloud, considering the computational complexity. The model training process is based on the historical data stored during the first phase, and the data matrix of the wireless sensor node *i* archived during time *T* is

$$\mathbf{X}_i = [\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}, \ldots, \mathbf{x}_i^{(T)}]. \tag{7.5}$$

First of all, the data vector is normalized to eliminate the impact of different scales,

$$\overline{x}_{i,j}^{(t)} = \frac{x_{i,j}^{(t)} - \min(\mathbf{x_{i,j}})}{\max(\mathbf{x_{i,j}}) - \min(\mathbf{x_{i,j}})}, \tag{7.6}$$

where $\mathbf{x}_{i,j} = [x_{i,j}^{(1)}, x_{i,j}^{(2)}, \ldots, x_{i,j}^{(T)}]$ refers to the historical data vector of physical variable $j$ measured by node $i$ during time period $T$. $\min(\mathbf{x_{i,j}})$ and $\max(\mathbf{x_{i,j}})$ are the minimum and maximum values of $\mathbf{x}_{i,j}$.

Based on the spatial correlation of the sensing data, a clustering algorithm such as the bounded-size K-means clustering algorithm proposed in Chapter 6 is used to cluster the wireless sensor nodes, which would generate $K$ clusters. Based on the clustering results, an AE model is built for each cluster, and the model parameters are trained. For a cluster $k, k = 1, 2, \ldots, K$, the parameters $\theta_k = \{\mathbf{W}, \mathbf{b}_f\}$ and $\theta'_k = \{\mathbf{V}, \mathbf{b}_g\}$ would be obtained. The clustering results and the parameters of the encoders and decoders obtained from the training process would be sent back to the edge devices for data outlier detection.

#### 7.3.3.3 Data Outlier Detection

The data outlier detection process is mainly executed at the edge devices, which can improve the real-time performance of the data outlier detection and relieve the bandwidth burden on the trunk link connected to the cloud platform.

The AE generates a certain error when decode and reconstruct the data that have passed through the encoder. If there is a data outlier in the original data, the generated error would fluctuate significantly as compared to the error generated by the normal data. The data outlier detection algorithm is proposed by the exploitation of the error fluctuation. At time $t$, the squared error is calculated as

$$\varepsilon = \|g_{\theta'}(f_\theta(\overline{\mathbf{x}}^{(t)})) - \overline{\mathbf{x}}^{(t)}\|_2^2, \tag{7.7}$$

where $\overline{\mathbf{x}}^{(t)}$ is the original data at $t$ after normalization, and $g_{\theta'}(f_\theta(\overline{\mathbf{x}}^{(t)}))$ is the data after passing through the encoder and the decoder.

The specific steps of the data outlier detection algorithm are given as follows. Firstly, the data are normalized to eliminate the impact of different scales. The normalized data are then substituted into the squared error equation (7.7), where the parameters of the AE encoders and decoders stored at the edge devices are used. If the squared error exceeds a preset threshold, the data sample is identified as an outlier, and then the raw data and an outlier warning are

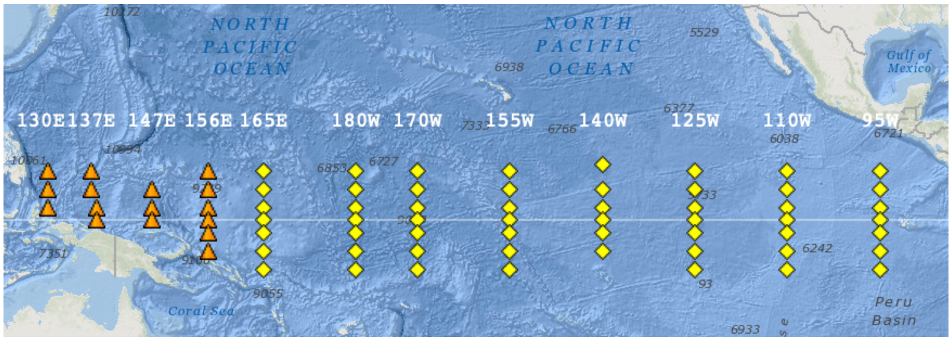


Figure 7.4: Deployment of the monitoring stations in the TAO project.

simultaneously uploaded to the cloud platform for further analysis. If the squared error value is lower than the threshold, the encoded data are uploaded to the cloud platform, and the data are decoded and reconstructed in the cloud so that the amount of data uploading would be reduced. The pseudocode of the AE-based data outlier detection algorithm is listed in Algorithm 7.

## 7.4 Performance Evaluation

### 7.4.1 Simulation Settings

Sea temperature measurements from the TAO project are used to analyze the detection accuracy of the proposed AE-based data outlier detection algorithm [109]. The deployment of the monitoring stations is shown in Fig.7.4.

Seven monitoring stations located at 170W are selected, and sea temperature measurements at depths of 25m, 50m, 75m, 100m, 125m, 150m, 175m, 200m, 300m, and 500m are taken at each station. The measurements are collected every 10min from 8/21/2018 to 8/30/2018. The selected seven monitoring stations are gathered into one cluster. Thus, at each sampling moment, a data vector consisted of 70 variables is collected from the cluster. During the selected sampling period, 1440 samples have been received. The first 1000 data samples are used for model training. According to the characteristics and quantity of the data, the AE model with a single hidden layer is used. The mapping function used by the encoder is Sigmoid function, and the ReLu function is used as the mapping function of the decoder. The mini-batch gradient descent algorithm is used for model training, and the 1001∼1100 data samples are used to test the data outlier detection accuracy.

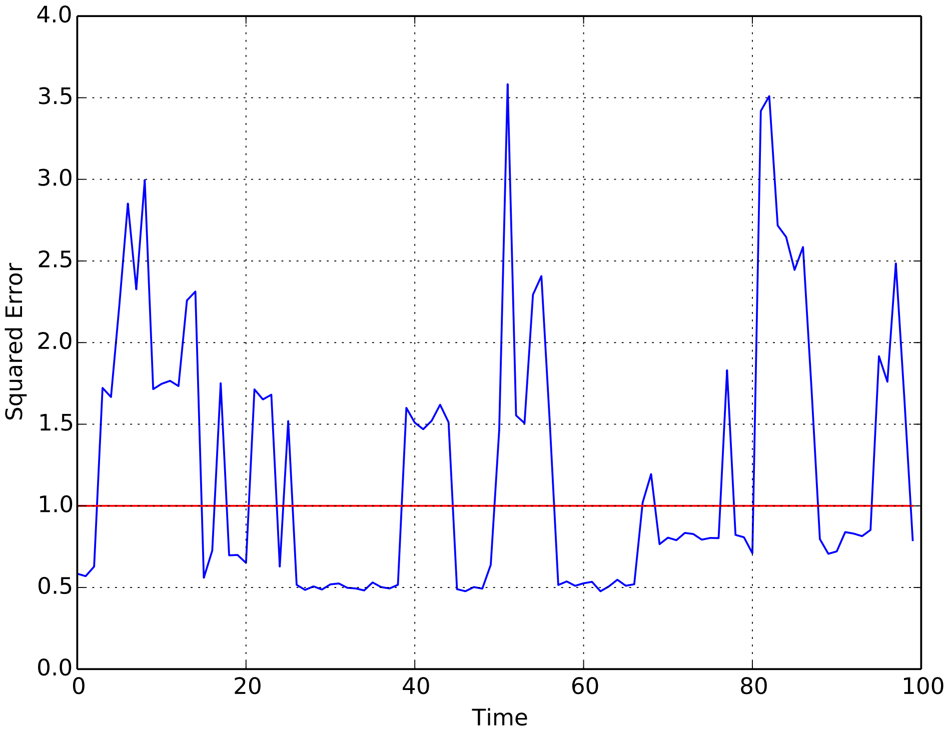


Figure 7.5: Squared error generated by data reconstruction at 100 selected sampling moments.

The squared error generated during the selected 100 sampling moments is shown in Fig.7.5. According to statistics, among the 100 selected data samples, there are 44 data samples with at least one data outlier occurred among the 70 variables. It can be observed from Fig.7.5 that the occurrence of data outlier can generate a large fluctuation in the squared error, and an appropriate threshold (the red line in Fig.7.5) can be used to accurately identify the data outlier.

## 7.4.2 Evaluation Metrics

For a preset threshold, the data outlier detection result is evaluated by the true positive rate (TPR) and the false positive rate (FPR). TPR is the ratio between the number of correctly detected outliers to the total number of outliers. FPR is the ratio between the number of normal data samples that are erroneously detected as outliers to the total number of normal data samples. Multiple thresholds can generate multiple sets of TPR and FPR. Based on the multiple sets of TPR and FPR, the receiver operating characteristic curve (ROC curve) can be drawn, where the horizontal axis is FPR and the vertical axis is TPR. The area under the ROC curve (area under the curve, AUC) is generally used to evaluate the data outlier detection algorithm.

The range of AUC is 0~1, where the closer the AUC is to 1, the closer the detection algorithm is to an ideal detector.

### 7.4.3 Simulation Results

Fig.7.6 shows the multiple ROC curves generated with different numbers of units in the single hidden layer. It can be seen from Fig.7.6 that the three ROC curves coincide and the AUC is 1, which indicates that an appropriate threshold can always be found to make TPR of the data outlier detection to 1 while FPR to 0, whatever the number of units in the hidden layer is. When the numbers of units in the hidden layer are different, the appropriate thresholds are also different. When the numbers of units are 70, 50 and 35, the appropriate values of the thresholds are 0.9, 1.4 and 2, respectively. This is because with the decrement in the number of units in the hidden layer (from 70 to 35), the extent of data compression done by the encoder increases, which leads to the decrease in the accuracy of data decoding and the rise in the squared error of data reconstruction. The overall increment in the error does not affect the accuracy of data outlier detection as shown in Fig.7.5 but leads to an increase in the appropriate threshold. Although the squared error of the data reconstruction increases, the amount of data uploading can be reduced. Thus, there exists a trade-off, which can be determined by the requirements of specific applications.

## 7.5 Chapter Summary

In this chapter, an architecture of the edge-cloud collaborative IoT monitoring system is proposed, which consists of the cloud platform, edge devices, and wireless sensor nodes. The functions of each component and the intercommunications are given in detail. Furthermore, an AE-based data outlier detection algorithm is proposed. The data outlier detection algorithm is evaluated by using the practical oceanic atmospheric data. The resulting ROC curves indicate that the data outlier detection algorithm is close to an ideal detector.

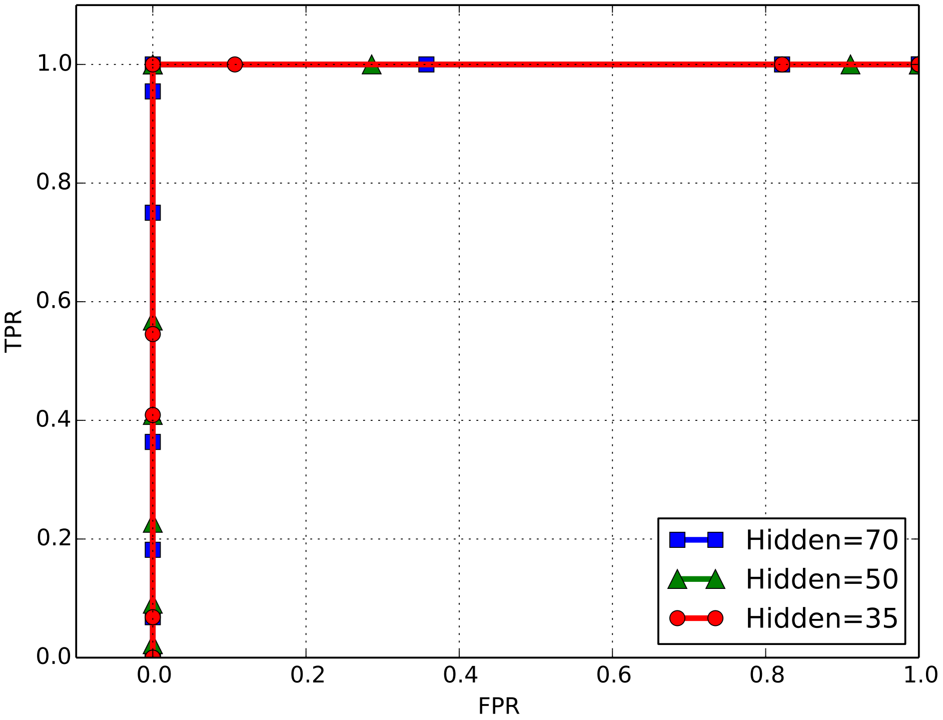


Figure 7.6: ROC curves of the data outlier detection with different numbers of units in the single hidden layer.

# Chapter 8

# Conclusion and Future Work

## 8.1 Conclusion

With the pervasive deployment and enlarging scale of IoT systems, the number of involving IoT devices increases in an explosive trend, which generates a massive amount of data. The huge amount of IoT devices and the correspondingly generated IoT data bring critical challenges to the IoT systems. The related issues addressed in the thesis are fallen into the following two major aspects: data processing of the massive amount of IoT data and topology management of the subnets self-organized by IoT end devices in the large-scale IoT systems. More specifically,

- *IoT Data Processing in Large-Scale IoT Systems*

The huge number of IoT end devices continuously generate a massive amount of data, which challenges the IoT systems in data processing and analysis. Providing the weak capabilities of IoT end devices, the IoT data needs to be uploaded to the remote data center, *e.g.*, cloud computing platform, for comprehensive data analytics and storage. However, the overwhelming amount of data imposes a heavy burden on the network bandwidth of the trunk link for data uploading, which may even result in system crashes. Furthermore, due to the complex environmental situations of the deployment fields and the low-cost feature of IoT end devices, the devices are vulnerable to different kinds of attacks and even inner malfunctions, which can finally lead to the abnormality in IoT data. The tainted IoT data can lead the data-driven IoT systems into unsafe conditions. Therefore, it is necessary to develop algorithms for real-time data processing in large-scale IoT systems.

- *Topology Management of Self-Organized Subnets in Large-Scale IoT Systems*

Due to the self-organized and dynamic features, topology management is among the most critical challenges in large-scale IoT systems. The physical topology of a large-scale IoT system indicates not only the logical connectivity statuses (*i.e.*, logical topology) of the self-organized subnets but also the physical locations of the IoT end devices. Therefore, awareness of physical topology in the cloud can facilitate the system with performance optimization. However, due to the features of random deployment and self-organization, the physical topology of a large-scale IoT system is extremely hard to control during the deployment stage. In addition, due to the low-cost feature of IoT end devices, especially the wireless sensor nodes, the devices are typically built with constrained resources and are vulnerable to malicious attacks. It is not uncommon to witness the malfunction and death of devices, which can finally change the connectivity statuses and system topology. Besides, associations and disassociations of the dynamic devices can also lead to the variation of topology. Thus, it is necessary to develop topology discovery schemes for the large-scale IoT systems in order to construct the physical topology in the cloud.

To overcome the issues mentioned above, a number of algorithms and schemes for data processing and topology discovery in the large-scale IoT systems have been developed. The contributions that have been made in this thesis and the conclusions drawn from these contributions are summarized as follows:

A comprehensive study of data analytics in IoT systems has been conducted in Chapter 2. The fundamentals of IoT data analytics were firstly elucidated. Afterwards, the system architectures that could support effective and efficient data analytics in IoT systems have been analyzed. Finally, the existing applications were investigated from the perspectives of system design and shortcomings of performance.

In Chapter 3, a cluster-based data analysis framework has been proposed using R-PCA, which could aggregate the redundant data and detect the outliers. More specifically, at a cluster head, spatially correlated sensor data collected from cluster members were aggregated by extracting the PCs, and potential data outliers were determined by the abnormal SPE score, which was defined as the square of residual value after extraction of PCs. With R-PCA, the parameters of the PCA model could be recursively updated to track the changes in IoT systems.

The cluster-based data analysis framework also relieved the computational and processing burdens on sensor nodes. Practical databases based simulations have confirmed that the proposed framework efficiently aggregated the correlated sensor data with high recovery accuracy. The data outlier detection accuracy was also improved by the proposed method as compared to the benchmark algorithms.

In Chapter 4, an edge computing enabled temporal IoT data reduction scheme has been proposed to reduce the total amount of IoT data uploaded to the cloud. More specifically, IoT data were firstly modeled as multivariate normal distribution by the cloud. Dual KFs with identical parameters were then deployed at both the edge and cloud ends. The same predictions were simultaneously triggered by the dual KFs at both ends. Only the measured IoT data out of the predicted range was further uploaded from edge to cloud. Otherwise, predicted values were used at both ends instead of measurements. A simple prototype IoT system has been developed for performance evaluation. Experimental results have indicated that the proposed scheme significantly reduced the number of packets uploaded to the cloud while guaranteed the data accuracy.

In Chapter 5, a cloud-orchestrated physical topology discovery scheme for large-scale IoT systems using UAVs has been proposed, in order to construct the physical topology in the cloud. More specifically, the large-scale monitoring area was firstly split into a number of subregions for UAV-enabled data collection. Within the subregions, parallel MHRW was developed to gather the information of wireless sensor nodes, including their IDs and neighbor tables. The collected information was then forwarded to the cloud through UAVs for the initial construction of logical topology. Afterwards, a network-wide 3D localization algorithm was further developed based on the constructed logical topology and multidimensional scaling method, termed as Topo-MDS, where the UAV equipped with a GPS chipset was served as a mobile anchor to locate the sensor nodes. Simulation results have indicated that the parallel MHRW improved both the efficiency and accuracy of logical topology discovery. Besides, the Topo-MDS algorithm dramatically improved the 3D localization accuracy, as compared to the existing algorithms in the literature.

In Chapter 6, a UAV enabled spatial data sampling scheme has been proposed using DAE neural network. More specifically, a UAV-enabled edge-cloud collaborative IoT system architecture was firstly developed for data processing in large-scale IoT monitoring systems, where UAV was utilized as a mobile edge computing device. Based on this system architecture, the UAV-enabled spatial data sampling scheme was further proposed, where the wireless sensor nodes of large-scale IoT systems were clustered by a newly developed bounded-size K-means clustering algorithm. A neural network model, *i.e.*, DAE, was applied to each cluster for data sampling and reconstruction, by exploiting the spatial correlation among data samples. Simulations have been conducted and the results indicated that the proposed scheme improved the data reconstruction accuracy under the same sampling ratio without introducing extra complexity, as compared to the compressive sensing based method.

Based on the system architecture proposed in Chapter 6, an AE neural network based data outlier detection algorithm has been developed in Chapter 7. By using AE, the spatial correlation of data could be fully utilized to improve the data outlier detection accuracy. Performance evaluation has been conducted based on the oceanic atmospheric data, where the numerical results indicated that the developed scheme could detect the data outliers accurately.

## 8.2 Future Work

The technical issues on IoT data processing and topology management in the large-scale IoT systems have been resolved in the thesis. Several other challenges still need to be investigated to further enhance the performance of IoT systems. Some of the future research directions are identified in this section, including collaborative artificial intelligence (AI), cost-efficient event management, and security and privacy protection.

### 8.2.1 Collaborative Artificial Intelligence

In developing the sensing, learning and decision-making capabilities in IoT systems, several technical issues have been met, including network optimization, resource allocation, and big IoT data analytics. AI technology is a promising solution to these issues. By using AI, the distributed system resources can be collaboratively and optimally allocated to provide timely responses to the demands of users and devices. However, due to the low-cost and distributed features of IoT systems, the IoT devices are built with limited resources while the whole system resources are scattered, which can be hardly used for the computation-intensive AI algorithms directly. Therefore, collaborative computing needs to be developed to implement the AI algorithms. The specific research issues that need to be addressed in collaborative computing are analyzed as follows.

- *Resource awareness of the IoT system components:* The objective of collaborative computing in the IoT systems is to meet the requirements of tasks while minimizing the consumption of system resources. Therefore, it is necessary to be aware of the available resources of all the system components, including the communication resources (*e.g.*, bandwidth and spectrum), computing resources and power supplies. With the awareness of available resources, the consumption can be optimally allocated among the components while assigning the tasks.

- *Task offloading:* Computational tasks need to be optimally allocated within the IoT systems, where the specific responsibilities of each IoT system component have to be determined to minimize the resource consumption and optimize the time of task completion. In terms of the real-time requirements, it is better to allocate the tasks to the edge devices, since edge devices are closer to the IoT end devices. However, the service of edge computing is generally supported by lightweight devices such as cloudlet servers. The tasks with a massive amount of IoT data and high computing complexity still have to be partially offloaded to the cloud end. Therefore, it is critical to developing certain decision-making strategies for task allocation so that the time and resources consumed by the combination of processing and communications can be minimized.

- *Quality of service (QoS) enhancement:* In terms of QoS enhancement, firstly, it is necessary to identify the constraints of the IoT systems that possibly lead to the shortcomings of performance, such as underlying network bandwidth, computing power, and cache size. Based on the findings, corresponding methods need to be further developed to minimize the response time and resource consumption and also enhance the reliability in the case of system failures.

### 8.2.2 Cost-Efficient Event Management

Edge-cloud collaborative IoT systems have already been applied to long-term event monitoring. The long-term event monitoring highly relies on seamless interactions with the real world through sensing devices [110]. Due to the complex and dynamic features of the monitoring environments, a large number of sensing devices need to be deployed to fully cover the large area and adapt to the instantaneous environmental changes.

The data measured and collected by the sensing devices are finally stored in the cloud. Based on the historical data, comprehensive data analytics can be conducted to extract the normal patterns of the systems and monitoring targets and predict future trends. The results of data analytics can be either kept in the cloud or sent back to the edge devices for event detection, where data that do not follow the normal pattern or trend are detected to identify the abnormal events. Definitely, event detection executed at the edge devices can improve the timeliness of responses.

However, long-term event monitoring and detection are resource-draining, where massive system resources are occupied and consumed. One potential solution is the data-driven event triggering technique, which enables the actions of communication or computing taken place only when a particular event or a series of events occur [111]. By using such an approach, the consumption of system resources can be dramatically reduced, since most of the devices can be scheduled to sleep when no event occurs. Some of the future research issues in this area are highlighted below.

- *Behavior modeling:* Based on the historical data stored in the cloud, behaviors of the monitoring targets can be learned. However, several concerns still need to be resolved, including how to define appropriate reference models, how to handle the unpredictable characteristics of systems, and how to train models with machine learning algorithms.

- *Event detection:* By the exploitation of behavior modeling, the normal patterns and trends can be identified. Thus, the sampled data that do not follow the normal patterns are detected to identify the abnormal events. Machine learning methods [102] and even neural network models [112] can be applied to the abnormal event detection.

- *Timing of event triggering:* Event triggering technique can reduce the consumption of system resources. However, it is a big issue to determine the timing of event triggering, since the events are dynamic, which may have recurrent patterns. In other words, how to identify the states of an event and adjust the system accordingly needs to be investigated.

- *On-demand resource allocation:* After an event is triggered, how to handle the events and allocate the system resources (*e.g.*, energy, bandwidth, and spectrum) according to the demands remains a big technical challenge.

### 8.2.3 Security and Privacy Protection

Although IoT technology has enabled several conventional systems into intelligent areas, how to provide real-time security and privacy protection remains a key technical concern [113]. IoT devices are heterogeneous and built with quite different capabilities. Some of the devices with weak capabilities are vulnerable to security threats, due to the transparent air interfaces of wireless communications and lack of protection mechanisms. Thus, malicious attacks can occur at any phase of IoT data processing, including data collection, communications, modeling, *etc.* IoT data are highly related to the privacy of users, since IoT systems register personal information and monitor the daily behaviors of users [114]. Therefore, it can be inferred that malicious attacks can unveil the privacy of users.

Furthermore, due to the unique characteristics of IoT systems, the traditional security and privacy protection mechanisms can hardly be applied to the IoT systems directly. From the perspective of IoT data characteristics, the intricate patterns and characteristics of IoT data are seldom considered in the traditional protection mechanisms. Moreover, most of the traditional protection mechanisms are based on the static databases, while in the IoT systems, data are dynamically changing due to the unstable system states. In terms of the edge-cloud collaborative system architecture, cloud platform as the remote data and control center can provide a global view of the system, which is generally used in the centralized security and privacy protection mechanisms for device authentication and access control. However, due to the heterogeneous and dynamic features of IoT end devices, the traditional centralized mechanisms enabled by the cloud platform are not efficient enough to authenticate the huge number of devices and authorize their access to IoT data. In addition to the IoT end devices, the distributed edge devices also bring security risks to the IoT systems. Unlike the cloud platform generally provided by trustworthy third parties, edge devices are from multiple unauthorized providers, which imposes an extra burden on the device authentication and access control.

Therefore, it is necessary to develop some new mechanisms to protect the security and privacy of the edge-cloud collaborative IoT systems. One potential solution is the utilization of the edge-cloud collaboration, where edge devices are secured by the cloud platform and function as proxies to protect the resource-constraint IoT end devices [35]. The other one is developing decentralized protection mechanisms, such as blockchain-based methods [115]. Based on the above discussions, the research directions that need to be seriously considered in the future are summarized as follows.

- *Security enhancement on collaborative computing:* Collaborative computing can facilitate IoT systems with performance enhancement in several aspects. However, there are still several security threats needed to be considered, such as how to protect the confidentiality and integrity of IoT data in the procedure of task offloading and how to prevent the privacy from being unveiled to the service provider when archive and analyze IoT data in the third-party cloud platform.

- *Privacy protection:* Since personal information is registered and daily behaviors are monitored, IoT data are highly related to the privacy of users. Therefore, privacy protection mechanisms such as differential privacy need to be further investigated.

- *Access control:* Although the cloud platform can provide centralized device authentication and access control, it is not efficient enough for the IoT systems with a huge amount of end devices and edge devices. Therefore, both edge-cloud collaborative and decentralized mechanisms need to be further studied to manage the access control of the huge number of heterogeneous IoT devices.

# Bibliography

[1] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[2] Rima Qureshi. Ericsson mobility report. *Ericsson, Stockholm, Sweden, Tech. Rep. EAB-14*, 28658, 2014.

[3] Luca Davoli, Laura Belli, Antonio Cilfone, and Gianluigi Ferrari. From micro to macro IoT: Challenges and solutions in the integration of IEEE 802.15. 4/802.11 and sub-GHz technologies. *IEEE Internet of Things Journal*, 5(2):784–793, 2017.

[4] Philip Russom et al. Big data analytics. *TDWI Best Practices Report, 4th quarter*, 19(4):1–34, 2011.

[5] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 2018.

[6] Min Chen, Shiwen Mao, Yin Zhang, and Victor CM Leung. Big data: related technologies, challenges and future prospects. 2014.

[7] INFORMS. Best definition of analytics. *[Online] Available: https://www.informs.org/About-INFORMS/News-Room/O.R.-and-Analytics-in-the-News/Best-definition-of-analytics*.

[8] Alex Bekker. 4 types of data analytics to improve decision-making. *[Online] Available: https://www.scnsoft.com/blog/4-types-of-data-analytics*, 2017.

[9] Tianqi Yu, Xianbin Wang, and Abdallah Shami. A novel fog computing enabled temporal data reduction scheme in IoT systems. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–5. IEEE, 2017.

[10] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[11] Shree Krishna Sharma and Xianbin Wang. Live data analytics with collaborative edge and cloud processing in wireless IoT networks. *IEEE Access*, 5(99):4621–4635, 2017.

[12] Paula Ta-Shma, Adnan Akbar, Guy Gerson-Golan, Guy Hadash, Francois Carrez, and Klaus Moessner. An ingestion and analytics architecture for IoT applied to smart city use cases. *IEEE Internet of Things Journal*, 5(2):765–774, 2018.

[13] Adnan Akbar, George Kousiouris, Haris Pervaiz, Juan Sancho, Paula Ta-Shma, Francois Carrez, and Klaus Moessner. Real-time probabilistic data fusion for large-scale IoT applications. *IEEE Access*, 6:10015–10027, 2018.

[14] Daniel Puschmann, Payam Barnaghi, and Rahim Tafazolli. Using LDA to uncover the underlying structures and relations in smart city data streams. *IEEE Systems Journal*, 12(2):1755–1766, 2018.

[15] Tejal Shah, Ali Yavari, Karan Mitra, Saguna Saguna, Prem Prakash Jayaraman, Fethi Rabhi, and Rajiv Ranjan. Remote health care cyber-physical system: quality of service (QoS) challenges and opportunities. *IET Cyber-Physical Systems: Theory & Applications*, 1(1):40–48, 2016.

[16] Diana C Yacchirema, David Sarabia-Jácome, Carlos E Palau, and Manuel Esteve. A smart system for sleep monitoring by integrating IoT with big data analytics. *IEEE Access*, 6:35988–36001, 2018.

[17] Enshaeifar Enshaeifar, Payam Barnaghi, Severin Skillman, Andreas Markides, Tarek Elsaleh, Sahr Thomas Acton, Ramin Nilforooshan, and Helen Rostill. The Internet of Things for dementia care. *IEEE Internet Computing*, 22(1):8–17, 2018.

[18] Katia Moskvitch. When machinery chats, connections industrial IoT. *Engineering & Technology*, 12(2):68–70, 2017.

[19] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. Industrial Internet of Things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, 2018.

[20] Johan Åkerberg, Mikael Gidlund, and Mats Björkman. Future research challenges in wireless sensor and actuator networks targeting industrial automation. In *IEEE International Conference on Industrial Informatics (INDIN)*, pages 410–415. IEEE, 2011.

[21] Ovidiu Vermesan, Peter Friess, Patrick Guillemin, Sergio Gusmeroli, Harald Sundmaeker, Alessandro Bassi, Ignacio Soler Jubert, Margaretha Mazura, Mark Harrison, Markus Eisenhauer, et al. Internet of Things strategic research roadmap. *Internet of Things-Global Technological and Societal Trends*, 1(2011):9–52, 2011.

[22] Luigi Atzori, Antonio Iera, and Giacomo Morabito. SIoT: Giving a social structure to the Internet of Things. *IEEE Communications Letters*, 15(11):1193–1195, 2011.

[23] Zia Ush Shamszaman and Muhammad Intizar Ali. Toward a smart society through semantic virtual-object enabled real-time management framework in the social Internet of Things. *IEEE Internet of Things Journal*, 5(4):2572–2579, 2018.

[24] Vishal Sharma, Ilsun You, and Ravinder Kumar. ISMA: Intelligent sensing model for anomalies detection in cross platform OSNs with a case study on IoT. *IEEE Access*, 5:3284–3301, 2017.

[25] Tianqi Yu, Xianbin Wang, Jiong Jin, and Kenneth McIsaac. Cloud-orchestrated physical topology discovery of large-scale IoT systems using UAVs. *IEEE Transactions on Industrial Informatics*, 14(5):2261–2270, 2018.

[26] Shifeng Fang, Li Da Xu, Yunqiang Zhu, Jiaerheng Ahati, Huan Pei, Jianwu Yan, Zhihui Liu, et al. An integrated system for regional environmental monitoring and management based on Internet of Things. *IEEE Transactions on Industrial Informatics*, 10(2):1596–1605, 2014.

[27] Tianqi Yu, Xianbin Wang, and Abdallah Shami. UAV-enabled spatial data sampling in large-scale IoT systems using denoising autoencoder neural network. *IEEE Internet of Things Journal*, 2018.

[28] Partha Pratim Ray, Mithun Mukherjee, and Lei Shu. Internet of Things for disaster management: State-of-the-art and prospects. *IEEE Access*, 5:18818–18835, 2017.

[29] Marcus Handte, Stefan Foell, Stephan Wagner, Gerd Kortuem, and Pedro José Marrón. An Internet-of-Things enabled connected navigation system for urban bus riders. *IEEE Internet of Things Journal*, 3(5):735–744, 2016.

[30] Daniel Minoli, Kazem Sohraby, and Benedict Occhiogrosso. IoT considerations, requirements, and architectures for smart buildings–energy optimization and next generation building management systems. *IEEE Internet of Things Journal*, 2017.

[31] Guobin Xu, Wei Yu, David Griffith, Nada Golmie, and Paul Moulema. Toward integrating distributed energy resources and storage devices in smart grid. *IEEE Internet of Things Journal*, 4(1):192–204, 2017.

[32] Xiangmao Chang, Jun Huang, Shucheng Liu, Guoliang Xing, Hongwei Zhang, Jianping Wang, Liusheng Huang, and Yi Zhuang. Accuracy-aware interference modeling and measurement in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 15(2):278–291, 2016.

[33] Yang Zhang, Nirvana Meratnia, and Paul Havinga. Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials*, 12(2):159–170, 2010.

[34] John A Stankovic. Research directions for the Internet of Things. *IEEE Internet of Things Journal*, 1(1):3–9, 2014.

[35] Mung Chiang and Tao Zhang. Fog and IoT: An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6):854–864, 2016.

[36] Mianxiong Dong, Kaoru Ota, and Anfeng Liu. RMER: Reliable and energy efficient data collection for large-scale wireless sensor networks. *IEEE Internet of Things Journal*, 3(4):511–519, 2016.

[37] Mihaela I Chidean, Eduardo Morgado, Margarita Sanromán-Junquera, Julio Ramiro-Bargueno, Javier Ramos, and Antonio J Caamaño. Energy efficiency and quality of data reconstruction through data-coupled clustering for self-organized large-scale WSNs. *IEEE Sensors Journal*, 16(12):5010–5020, 2016.

[38] Feng Wang and Jiangchuan Liu. Networked wireless sensor data collection: Issues, challenges, and approaches. *IEEE Communications Surveys & Tutorials*, 13(4):673–687, 2011.

[39] Knut Ovsthus, Lars M Kristensen, et al. An industrial perspective on wireless sensor networks–a survey of requirements, protocols, and challenges. *IEEE Communications Surveys & Tutorials*, 16(3):1391–1412, 2014.

[40] Tianqi Yu, Xianbin Wang, and Abdallah Shami. A novel R-PCA based multivariate fault-tolerant data aggregation algorithm in WSNs. In *IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2016.

[41] NDBC-TAO. *[Online] Available: http://tao.ndbc.noaa.gov/tao/data_download/search_map.shtml*, 2016.

[42] Madden Samuel. Intel lab data. *Massachusetts Avenue: University of Massachusetts Institute of Technology*, pages 12–31, 2012.

[43] Jinran Chen, Shubha Kher, and Arun Somani. Distributed fault detection of wireless sensor networks. In *Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*, pages 65–72. ACM, 2006.

[44] Sutharshan Rajasegarar, Christopher Leckie, Marimuthu Palaniswami, and James C Bezdek. Quarter sphere based distributed anomaly detection in wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, volume 7, pages 3864–3869, 2007.

[45] Zhang Yang, Nirvana Meratnia, and Paul Havinga. An online outlier detection technique for wireless sensor networks using unsupervised quarter-sphere support vector machine. In *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 151–156. IEEE, 2008.

[46] Vassilis Chatzigiannakis and Symeon Papavassiliou. Diagnosing anomalies and identifying faulty nodes in sensor networks. *IEEE Sensors Journal*, 7(5):637–645, 2007.

[47] Rui Zhang, Ping Ji, Dinkar Mylaraswamy, Mani Srivastava, and Sadaf Zahedi. Co-operative sensor anomaly detection using global information. *Tsinghua Science and Technology*, 18(3):209–219, 2013.

[48] Shing-Chow Chan, HC Wu, and Kai Man Tsui. Robust recursive eigendecomposition and subspace-based algorithms with application to fault detection in wireless sensor networks. *IEEE Transactions on Instrumentation and Measurement*, 61(6):1703–1718, 2012.

[49] Deniz Erdogmus, Yadunandana N Rao, Hemanth Peddaneni, Anant Hegde, and Jose C Principe. Recursive principal components analysis using eigenvector matrix perturbation. *EURASIP Journal on Applied Signal Processing*, 2004:2034–2041, 2004.

[50] Yan Sun, Hong Luo, and Sajal K Das. A trust-based framework for fault-tolerant data aggregation in wireless multimedia sensor networks. *IEEE Transactions on Dependable and Secure Computing*, 9(6):785–797, 2012.

[51] Liu Xiang, Jun Luo, and Catherine Rosenberg. Compressed data aggregation: Energy-efficient and high-fidelity data collection. *IEEE/ACM Transactions on Networking (TON)*, 21(6):1722–1735, 2013.

[52] Xiao-Yang Liu, Yanmin Zhu, Linghe Kong, Cong Liu, Yu Gu, Athanasios V Vasilakos, and Min-You Wu. CDC: Compressive data collection for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(8):2188–2197, 2015.

[53] Amirmohammad Rooshenas, Hamid R Rabiee, Ali Movaghar, and M Yousof Naderi. Reducing the data transmission in wireless sensor networks using the principal component analysis. In *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 133–138. IEEE, 2010.

[54] Christos Anagnostopoulos and Stathes Hadjiefthymiades. Context compression: Using principal component analysis for efficient wireless communications. In *IEEE International Conference on Mobile Data Management (MDM)*, volume 1, pages 206–215. IEEE, 2011.

[55] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[56] Sanjiv K Bhatia. Adaptive K-means clustering. In *FLAIRS Conference*, pages 695–699, 2004.

[57] Yingxin Xie, Xiangguang Chen, and Jun Zhao. Adaptive and online fault detection using RPCA algorithm in wireless sensor network nodes. In *International Conference on Intelligent System Design and Engineering Application (ISDEA)*, pages 1371–1374. IEEE, 2012.

[58] J Martínez-Carranza, Francisco Soto-Eguibar, and Héctor Moya-Cessa. Alternative analysis to perturbation theory in quantum mechanics. *The European Physical Journal D*, 66(1):1–6, 2012.

[59] Peng Jiang. A new method for node fault detection in wireless sensor networks. *Sensors*, 9(2):1282–1294, 2009.

[60] Valenzuela-Valds Juan F., Lpez Miguel, Angel, Padilla Pablo, Padilla Jos, L, and Minguillon Jesus. Human neuro-activity for securing body area networks: Application of brain-computer interfaces to people-centric Internet of Things. *IEEE Communications Magazine*, 55:62 – 67, 2017.

[61] Feng Shuo, Setoodeh Peyman, and Haykin Simon. Smart home: Cognitive interactive people-centric Internet of Things. *IEEE Communications Magazine*, 55:34 – 39, 2017.

[62] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 169–186. Springer, 2014.

[63] Farhoud Hosseinpour, Juha Plosila, and Hannu Tenhunen. An approach for smart management of big data in the fog computing context. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 468–471. IEEE, 2016.

[64] Farahd Mehdipour, Bahman Javadi, and Aniket Mahanti. Fog-engine: Towards big data analytics in the fog. In *IEEE International Conference on Dependable, Autonomic and Secure Computing, Pervasive Intelligence and Computing, Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 640–646. IEEE, 2016.

[65] Rongxing Lu, Kevin Heung, Arash Lashkari, and Ali Ghorbani. A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT. *IEEE Access*, 2017.

[66] Kanti V Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, pages 519–530, 1970.

[67] Peter Bodik, Wei Hong, Carlos Guestrin, Sam Madden, Mark Paskin, and Romain Thibaux. Intel lab data. *Online dataset*, 2004.

[68] Eben Upton and Gareth Halfacree. *Raspberry Pi user guide*. John Wiley & Sons, 2014.

[69] Google BigQuery IoT platform data. *[Online] Available: https://bigquery.cloud.google.com/dataset/iot-platform-1385 iot_platform_data*, 2016.

[70] GreenOrbs Project. *[Online]. Available:http://www.greenorbs.org/.*

[71] NDBC-TAO Project. *[Online]. Available:http://tao.ndbc.noaa.gov/.*

[72] Zhengguo Sheng, Chinmaya Mahapatra, Chunsheng Zhu, and Victor CM Leung. Recent advances in industrial wireless sensor networks toward efficient management in IoT. *IEEE Access*, 3:622–637, 2015.

[73] Li Da Xu, Wu He, and Shancang Li. Internet of Things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.

[74] Dulanjalie C Dhanapala and Anura P Jayasumana. Topology preserving maps extracting layout maps of wireless sensor networks from virtual coordinates. *IEEE/ACM Transactions on Networking (TON)*, 22(3):784–797, 2014.

[75] Ahmed Douik, Salah A Aly, Tareq Y Al-Naffouri, and Mohamed-Slim Alouini. Robust node estimation and topology discovery algorithm in large-scale wireless sensor networks. *arXiv preprint arXiv:1508.04921*, 2015.

[76] Luis Ramo Pinto, André Moreira, Luis Almeida, and Anthony Rowe. Characterizing multihop aerial networks of COTS multirotors. *IEEE Transactions on Industrial Informatics*, 13(2):898–906, 2017.

[77] Leandro A Villas, Daniel L Guidoni, and Jo Ueyama. 3D localization in wireless sensor networks using unmanned aerial vehicle. In *IEEE International Symposium on Network Computing and Applications (NCA)*, pages 135–142. IEEE, 2013.

[78] Eunchan Kim, Sangho Lee, Chungsan Kim, and Kiseon Kim. Mobile beacon-based 3D-localization with multidimensional scaling in large sensor networks. *IEEE Communications Letters*, 14(7):647–649, 2010.

[79] Byungrak Son, Yong-sork Her, and Jung-Gyu Kim. A design and implementation of forest-fires surveillance system based on wireless sensor networks for South Korea mountains. *International Journal of Computer Science and Network Security (IJCSNS)*, 6(9):124–130, 2006.

[80] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *IEEE Transactions on Industrial Informatics*, 9(1):132–141, 2012.

[81] Apratim Shaw and Kamran Mohseni. A fluid dynamic based coordination of a wireless sensor network of unmanned aerial vehicles: 3-D simulation and wireless communication characterization. *IEEE Sensors Journal*, 11(3):722–736, 2010.

[82] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Mérouane Debbah. Mobile unmanned aerial vehicles (UAVs) for energy-efficient Internet of Things communications. *IEEE Transactions on Wireless Communications*, 16(11):7574–7589, 2017.

[83] Minas Gjoka, Maciej Kurant, Carter T Butts, and Athina Markopoulou. Walking in Facebook: A case study of unbiased sampling of OSNs. In *IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[84] Zhang-Xin Chen, He-Wen Wei, Qun Wan, Shang-Fu Ye, and Wan-Lin Yang. A supplement to multidimensional scaling framework for mobile location: A unified view. *IEEE Transactions on Signal Processing*, 57(5):2030–2034, 2009.

[85] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5):698–700, 1987.

[86] IPW Group et al. Part 15.4: Low-Rate wireless personal area networks (LR-WPANs). Amendment 1: MAC sublayer. *IEEE Standard for Local and Metropolitan Area Netwwork IEEE Std 802.15. 4e-2012*, 2012.

[87] MEMSIC MPR2400CB. 2.4GHz MICAz Processor Board, datasheet (2013).

[88] DJI Spark datasheet. *[Online] Available: https://dl.djicdn.com/ downloads/Spark/20170621/Spark+User+Manual+V1.2.pdf*, 2017.

[89] Sinan Kurt and Bulent Tavli. Path-loss modeling for wireless sensor networks: A review of models and comparative evaluations. *IEEE Antennas and Propagation Magazine*, 59(1):18–37, 2017.

[90] Tsenka Stoyanova, Fotis Kerasiotis, Aggeliki Prayati, and George Papadopoulos. A practical RF propagation model for wireless network sensors. In *International Conference on Sensor Technologies and Applications (SENSORCOMM)*, pages 194–199. IEEE, 2009.

[91] Colin Cooper, Tomasz Radzik, and Yiannis Siantos. Estimating network parameters using random walks. *Social Network Analysis and Mining*, 4(1):168, 2014.

[92] Tianqi Yu, Xianbin Wang, and Abdallah Shami. Physical topology discovery scheme for wireless sensor networks using random walk process. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–5. IEEE, 2016.

[93] Liqiang Zhang, Qiang Cheng, Yingge Wang, and Sherali Zeadally. A novel distributed sensor positioning system using the dual of target tracking. *IEEE Transactions on Computers*, 57(2):246–260, 2008.

[94] Gang Xu, Edith C-H Ngai, and Jiangchuan Liu. Ubiquitous transmission of multimedia sensor data in Internet of Things. *IEEE Internet of Things Journal*, 5(1):403–414, 2017.

[95] Hongming Cai, Boyi Xu, Lihong Jiang, and Athanasios V Vasilakos. IoT-based big data storage systems in cloud computing: Perspectives and challenges. *IEEE Internet of Things Journal*, 4(1):75–87, 2017.

[96] Furqan Alam, Rashid Mehmood, Iyad Katib, Nasser N Albogami, and Aiiad Albeshri. Data fusion and IoT for smart ubiquitous environments: A survey. *IEEE Access*, 5:9533–9554, 2017.

[97] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pages 1096–1103, 2008.

[98] Chong Luo, Feng Wu, Jun Sun, and Chang Wen Chen. Compressive data gathering for large-scale wireless sensor networks. In *International Conference on Mobile Computing and Networking*, pages 145–156. ACM, 2009.

[99] Giorgio Quer, Riccardo Masiero, Gianluigi Pillonetto, Michele Rossi, and Michele Zorzi. Sensing, compression, and recovery for WSNs: Sparse signal modeling and monitoring framework. *IEEE Transactions on Wireless Communications*, 11(10):3447–3461, 2012.

[100] Xiangling Li, Xiaofeng Tao, and Guoqiang Mao. Unbalanced expander based compressive data gathering in clustered wireless sensor networks. *IEEE Access*, 5:7553–7566, 2017.

[101] Mihaela I Chidean, Eduardo Morgado, Eduardo del Arco, Julio Ramiro-Bargueno, and Antonio J Caamaño. Scalable data-coupled clustering for large scale WSN. *IEEE Transactions on Wireless Communications*, 14(9):4681–4694, 2015.

[102] Tianqi Yu, Xianbin Wang, and Abdallah Shami. Recursive principal component analysis-based data outlier detection and sensor data aggregation in IoT systems. *IEEE Internet of Things Journal*, 4(6):2207–2216, 2017.

[103] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. Rate-distortion balanced data compression for wireless sensor networks. *IEEE Sensors Journal*, 16(12):5072–5083, 2016.

[104] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning, lecture 6a, overview of mini-batch gradient descent. *[Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.*, 2012.

[105] LAN/MAN Standards Committee. Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). *IEEE Computer Society [online]. Available: http://profsite.um.ac.ir/~hyaghmae/ACN/WSNMAC1.pdf.*, 2003.

[106] Rick Chartrand and Wotao Yin. Iteratively reweighted algorithms for compressive sensing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3869–3872. Las Vegas, NV, USA, 2008.

[107] Thomas P. Minka. A comparison of numerical optimizers for logistic regression. *[Online]. Available: https://tminka.github.io/papers/logreg/minka-logreg.pdf*, 2007.

[108] Yunhao Liu, Yuan He, Mo Li, Jiliang Wang, Kebin Liu, and Xiangyang Li. Does wireless sensor network scale? A measurement study on GreenOrbs. *IEEE Transactions on Parallel and Distributed Systems*, 24(10):1983–1993, 2013.

[109] Landry Bernard, Kevin Kern, Jing Zhou, and Chung-Chu Teng. Refreshed data system for tropical atmosphere ocean (tao) buoy array. In *OCEANS 2008-MTS/IEEE Kobe Techno-Ocean*, pages 1–7. IEEE, 2008.

[110] Shuai Zhao, Le Yu, and Bo Cheng. An event-driven service provisioning mechanism for IoT (Internet of Things) system interaction. *IEEE Access*, 4:5038–5051, 2016.

[111] Panayiotis Kolios, Christos Panayiotou, Georgios Ellinas, and Marios Polycarpou. Data-driven event triggering for IoT applications. *IEEE Internet of Things Journal*, 3(6):1146–1158, 2016.

[112] Tianqi Yu, Yongxu Zhu, and Xianbin Wang. Autoencoder neural network-based abnormal data detection in edge computing enabled large-scale IoT systems. *Chinese Journal on Internet of Things*, 2(4):14–21, 2018.

[113] Muhammad Habib ur Rehman, Ejaz Ahmed, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Muhammad Imran, and Shafiq Ahmad. Big data analytics in industrial IoT using a concentric computing model. *IEEE Communications Magazine*, 56(2):37–43, 2018.

[114] Pawani Porambage, Mika Ylianttila, Corinna Schmitt, Pardeep Kumar, Andrei Gurtov, and Athanasios V Vasilakos. The quest for privacy in the Internet of Things. *IEEE Cloud Computing*, (2):36–45, 2016.

[115] Oscar Novo. Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE Internet of Things Journal*, 5(2):1184–1195, 2018.

# Curriculum Vitae

**Name:** Tianqi Yu

**Post-Secondary Education and Degrees:**

2015 - Present, Ph.D.
Electrical and Computer Engineering
The University of Western Ontario
London, Ontario, Canada

2013 - 2015, M.E.Sc
Electrical and Computer Engineering
The University of Western Ontario
London, Ontario, Canada

2009 - 2013, B.Eng (Hons)
Communication Engineering
Wuhan University
Wuhan, Hubei, China

**Honours and Awards:**

Graduate Student Research Award, ECE UWO 2019
Graduate Student Award for Excellence in Research, ECE UWO 2018
Student Travel Grant, IEEE GLOBECOM 2017
Outstanding Presentation, NSERC CREATE Symposium 2017
NSERC CREATE Scholar, 2013-2015
Best Student Paper Award, IEEE VTC 2015 Spring

**Related Work Experience:**

Teaching Assistant
The University of Western Ontario
2013 - Present

Research Assistant
The University of Western Ontario
2013-Present

**Publications:**

**Journal**:

[1] T.Yu, X.Wang and A.Shami. “UAV-Enabled Spatial Data Sampling in Large-Scale IoT Systems Using Denoising Autoencoder Neural Network”. IEEE Internet of Things Journal, 2019.

[2] T.Yu, Y.Zhu and X.Wang. “Autoencoder Neural Network-based Abnormal Data Detection in Edge Computing Enabled Large-Scale IoT Systems”. Chinese Journal on Internet of Things, 2018.

[3] T.Yu, X.Wang, J. Jin, K. McIsaac. “Cloud-Orchestrated Physical Topology Discovery of Large-Scale IoT Systems Using UAVs”. IEEE Transactions on Industrial Informatics, 2018.

[4] T.Yu, X.Wang and A.Shami. “Recursive Principal Component Analysis based Data Outlier Detection and Sensor Data Aggregation in IoT Systems”. IEEE Internet of Things Journal, 2017.

**Book Chapter**:

[1] T.Yu, X.Wang. “Real-time Data Analytics in Internet of Things Systems”. in Handbook of Real-Time Computing, Springer (*accepted*).

[2] T.Yu, X.Wang and Y.Zhu. “Blockchain Technology for IoT Systems: Principle, Applications and Challenges”. in 5G-enabled Internet of Things, Taylor Francis LLC, CRC Press, 2019.

**Conference**:

[1] T.Yu, X.Wang and A.Shami. “A Novel Fog Computing Enabled Temporal Data Reduction Scheme in IoT Systems”. IEEE Global Communications Conference (GLOBECOM), Singapore, 2017. (Student Travel Grant)

[2] T.Yu, X.Wang and A.Shami. “Physical Topology Discovery Scheme for Wireless Sensor Networks Using Random Walk Process”. IEEE Global Communications Conference (GLOBECOM), Washington DC, USA, 2016.

[3] T.Yu, X.Wang and A.Shami. “A Novel R-PCA based Multivariate Fault-Tolerant Data Aggregation Algorithm in WSNs”. IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 2016.

[4] T.Yu, A. Akhtar, X.Wang and A.Shami. “Energy-Efficient Scheduling Mechanism for Indoor Wireless Sensor Networks”. IEEE Vehicular Technology Conference (VTC Spring), Glasgow, UK, 2015. (Best Student Paper Award)

[5] T.Yu, A. Akhtar, A.Shami and X.Wang. “Temporal and Spatial Correlation based Distributed Fault Detection in Wireless Sensor Networks”. IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Halifax, Canada, 2015.