國 立 交 通 大 學

資訊科學與工程研究所

博 士 論 文

無線慣性感測網路中的
人體動作追蹤及其感測資料壓縮問題

Human Motion Tracking and Its Data Compression in Body-Area Inertial Sensor Networks

學　　生：吳鈞豪

指導教授：曾煜棋　教授

中 華 民 國 一 百 零 一 年 五 月

無線慣性感測網路中的人體動作追蹤及其感測資料壓縮問題
Human Motion Tracking and Its Data Compression in Body-Area Inertial Sensor Networks

學　　生：吳鈞豪　　Student: Chun-Hao Wu

指導教授：曾煜棋　　Advisor: Yu-Chee Tseng

國立交通大學
資訊科學與工程研究所
博士論文

A Dissertation
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in

Computer Science

May 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年五月

無線慣性感測網路中的人體動作追蹤及其感測資料壓縮問題

學生：吳鈞豪　　　　指導教授：曾煜棋

國立交通大學資訊科學與工程研究所博士班

摘　　要

感知科技和無線網路的進步，促成了「無線慣性感測網路」的發展。其可藉由在人體上，穿戴無線傳輸的慣性感測器，捕捉肢體的動作，並可應用在包含醫療照顧、電子遊戲和情緒運算上。對於建置高無線傳輸效率、高動作捕捉精度的感測平台所需要的技術，我們在此進行了三項基礎性的研究。

第一項研究，探討利用資料壓縮，來克服無線慣性感測網路中的感測資料收集議題。我們觀察到，雖然相鄰的感測節點可能激烈地競爭頻寬，但肢體移動時，其感測資料通常含有些許重覆，甚至是強烈的時、空相關性。我們爲無線慣性感測網路，特別設計壓縮演算法，以適用於其感測節點可監聽彼此傳輸的特性。爲了有效利用監聽的機制，我們將無線慣性感測網路上的資料壓縮問題，建模爲在監聽圖上的組合最佳化問題，證明其計算複雜度，並展示有效的計算方法。我們亦探討如何設計支援此壓縮模型的無線媒體存取層協定。實驗回報了以皮拉提斯的醫療復建動作進行的案例分析。結果顯示，我們的解決方法，可比先前研究節省百分之七十以上的傳輸資料量。

不同於第一項研究中，每個節點僅可容許監聽至多 $\kappa = 1$ 個其它節點的傳輸，在第二項研究裡，我們進一步考慮「複數空間相關性」，延伸 $\kappa = 1$ 到 $\kappa > 1$，並建構部分排序性的「有向無環圖」來表示感測節點間的壓縮相依性。相較於 $\kappa = 1$ 時，可在多項式時間內找到最小成本生成樹，我們證明即使 $\kappa = 2$，尋找最小成本的有向無環圖也是具 NP 完備性的。之後我們提出有效率的經驗性演算法，並用眞實感測資料驗證其效能。

除了資料收集，在第三項研究中，我們亦感興於在人體上佈置多個加速度計，以追蹤人體的動作。其中一個重要議題是如何計算重力的方向。這是很有挑戰的問題，尤其是當肢體在持續移動時。假設已將多個加速度計擺置於人體的一個剛性肢節上，一篇近期的論文提出一個資料融合的方法，可量測此剛體座標上重力的方向。然而，它並未探討，如何找到最佳的擺放位置，以達到最小的量測誤差。在此，我們定義此擺置最佳化問題，並提出兩個經驗性演算法，名之爲「基於梅式取樣之擺置法」與「最大間距法」。模擬與眞實試驗的結果，亦顯示我們的方法，在多種幾何形狀的剛體上，都能有效地找出近似解。

關鍵字：加速度計、無線慣性感測網路、行動運算、資料壓縮、擺置最佳化、人體動作追蹤

Human Motion Tracking and Its Data Compression in Body-Area Inertial Sensor Networks

Student: Chun-Hao Wu Advisor: Prof. Yu-Chee Tseng

Department of Computer Science, National Chiao Tung University

ABSTRACT

The advance of sensing technology and wireless communication has boosted *body-area inertial sensor networks (BISNs)*, in which wireless wearable inertial sensor nodes are deployed on a human body to monitor its motion. Applications include medical care, pervasive video games, and affective computing. We conduct fundamental research into the technologies required to create an efficient wireless communication BISN that maximizes motion tracking accuracy and data collection efficiency.

The first work addresses data collection issues in BISNs by data compression. We observe that, when body parts move, although sensor nodes in vicinity may compete strongly with each other, the transmitted data usually exists some levels of redundancy and even strong temporal and spatial correlations. Our scheme is specifically designed for BISNs, where nodes are likely fully connected and overhearing among sensor nodes is possible. We model the data compression problem for BISNs, where overhearing should be efficiently utilized, as a combinatorial optimization problem on overhearing graphs. We show its computational complexity and present efficient algorithms. We also discuss the design of the underlying MAC protocol to support our compression model. An experimental case study in Pilates exercises for patient rehabilitation is reported. The results show that our schemes reduce more than 70% of overall transmitted data compared with existing approaches.

Based on the first work, where a node is allowed to overhear at most $\kappa = 1$ node's transmission, in the second work, we consider *multi-spatial correlations* by extending $\kappa = 1$ to $\kappa > 1$ and constructing a partial-ordering *directed acyclic graph (DAG)* to represent the compression dependencies among sensor nodes. While a minimum-cost tree for $\kappa = 1$ can be found in polynomial time, we show that finding a minimum-cost DAG is NP-hard even for $\kappa = 2$. We then propose an efficient heuristic and verify its performance by real sensing data.

In addition to data collection, in the third work, we are interested in tracking human postures by deploying accelerometers on a human body. One fundamental issue in such scenarios is how to calculate the gravity. This is very challenging especially when the human body parts keep on moving. Assuming multiple accelerometers being deployed on a rigid part of a human body, a recent work proposes a data fusion method to estimate the gravity vector on that rigid part. However, how to find the optimal deployment of sensors that minimizes the estimation error of the gravity vector is not addressed. In this work, we formulate the deployment optimization problem and propose two heuristics, called *Metropolis-based* method and *largest-inter-distance-based (LID-based)* method. Simulation and real experimental results show that our schemes are quite effective in finding near-optimal solutions for a variety of rigid body geometries.

Keywords: accelerometer, body-area inertial sensor network, deployment optimization, human motion tracking, mobile computing, spatial-temporal data compression.

誌　　　　謝

本文從構思到完稿，花費了許多時間與心力，並得到不少人的幫忙才得以完成。在此要向所有幫助過的人，無論是直接或間接，傳達我的感謝之意。

尤其得特別感謝指導教授，曾煜棋教授，不遺餘力地協助我完成本研究，若本文的任何的貢獻，都得歸功於他的指導；研究上徬徨迷惘的時候，也給予我許多支持與開導。再來要感謝論文口試的委員們：陳志成教授（資工系）、溫□岸教授（電工系）、王友群教授（中山大學資工系）、金仲達教授（清華大學資工系）、朱浩華教授（台灣大學資工系）、鄭智湧教授（海洋大學電機系）與胡竹生教授（電機系）。感謝委員們專業的批評與指教，讓本文更臻完善。

此外，本文許多的技術知識，得感謝教導我專業科目的老師們。在計算的理論上，得特別感謝蔡錫鈞老師與黃廷祿老師，帶領我開拓理論的視野。在通訊與訊號處理上，得特別感謝高銘盛老師、杭學鳴老師的殷殷教誨。本文的實作與系統設計，得特別感謝教導我微處理機技術的曹孝櫟老師，同時也是大學時的專題指導老師，給予過許多的建議與幫助。

還要感謝一起成長、互相扶持的實驗室同學與助理，以及一起執行計畫的同仁們，不僅陪我渡過這段時光，與你們研討論文、交流心得，也讓我獲益良多。

最後，感謝我親愛的家人、朋友，以及寵物們。

謹此致謝。

吳鈞豪

中華民國一百零一年五月

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Backgrounds and Motivation

The advance of *micro-electro-mechanical systems (MEMS)* and wireless communication has boosted *body-area sensor networks (BSNs)*, in which miniature, integrated wireless sensor nodes are deployed on a human body to monitor various physical and biological signs, such as motion, heartbeat rate, and blood pressure [1]. An important application of BSNs is to track human body motions by deploying inertial sensors on movable body parts, forming a *body-area inertial sensor network (BISN)*.

An example of a human motion tracking system based on a BISN is shown in Fig. 1.1. The system senses a user's motion, analyzes it, understands the intention of the motion, and provides proper feedback to the user. The BISN, the front-end of the system, includes wireless inertial sensor nodes deployed on the user's body and a wireless sink node that collects sensing data from the network. The application system analyzes the collected sensing data for various purposes and is usually connected to peer systems and servers on the Internet for remote social interaction and monitoring.

The use of BISNs has many advantages over traditional camera-based motion capturing systems. It is much cheaper, immune to the line-of-sight problems in camera-based systems, easier to track the motion of multiple users, and easier to protect user privacy. Besides, it can be easily integrated with other types of sensors to provide more body information. Hence, BISNs have become a promising technology for a variety of applications, including pervasive video games [2, 3], medical care [4, 5], sports training [6], affective computing [3], and human-computer interface [7].

To boost this emerging technique, many fundamental issues need to be addressed. We list some of them as follows.

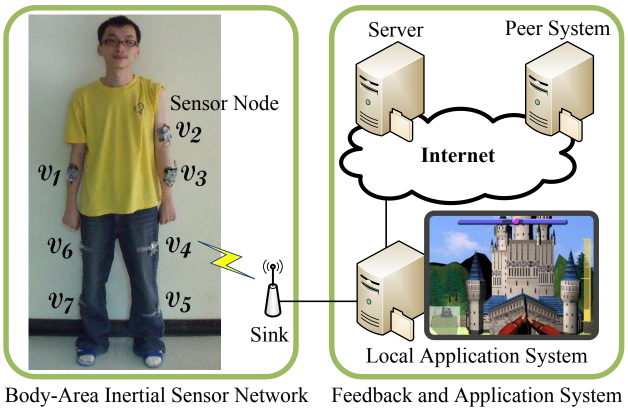


Figure 1.1: An example of a BISN-based human motion tracking system.

- **Data collection.** Applications may require real-time and high-resolution sensing data for various purposes, such as visualizing motions, recognizing body conditions, and diagnosing motion disorders. These requirements imply shorter transmission delays and higher sampling rates for a BISN. In addition, due to the size of a human body, nodes are typically dense and almost fully connected, leading to severe contention and collision among their transmissions [8,9]. Since wireless bandwidths are limited, how to efficiently utilize them is very important.

- **Motion tracking.** The fidelity of motion tracking has critical impacts on user experience and application performance. However, due to the complexity and uncertainty of human motion, improving it is a hard task. Issues and challenges include sensor calibration, sensing data fusion, sensor deployment, and the design of tracking algorithms.

- **Wearable sensor design.** The ease of wearing is one of the major concerns in adopting wearable sensors. Besides making sensors as small and light as possible, some research has considered how to weave them into clothes. It is also a challenging task to integrate sensors of multiple modalities into a single system on chip (SoC). In addition to challenges in hardware manufacturing, many issues arise when sensors are miniaturized. For example, the greatly constrained battery power may reduce network lifetime, the simplicity of miniaturized hardware may imply little computing power, and a shortened antenna may have a much shorter transmission range.

- **Energy saving and supply.** It is desirable to prolong the network lifetime as much as possible. Recharging batteries is typically quite inconvenient or even impossible when sensors are implanted in human bodies. Therefore, how to save energy is a critical issue

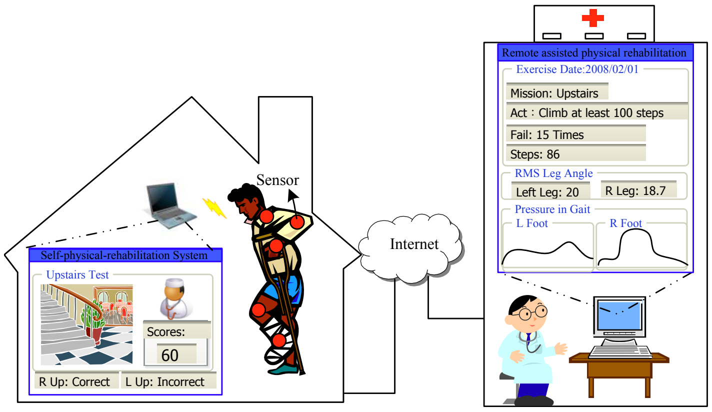


Figure 1.2: System architecture.

in the design of BSNs, from hardware to software. On the other hand, some research considers how to scavenge energy from the environment to replenish battery capacity.

Putting all these together, in the following, we build some prototypes to demonstrate novel application scenarios as well as to motivate our research.

1.1.1 Prototyping Experience: Home Rehabilitation

The BISN technology brings new light to medical care and telerehabilitation. As the aging population has increased significantly in the past decades, long-term health care has posed an increasing load for therapists and become an important challenge in the society. Traditionally, physical rehabilitation programs are instructed and monitored by therapists in the hospital, and thus consume severe medical resources. Furthermore, since the rehabilitation process is typically lengthy and boring, many patients may be reluctant to take it, thus further reducing its effectiveness.

In this prototyping, we demonstrate a system that uses intelligent sensors to capture human motion for home rehabilitation. Fig. 1.2 shows our system architecture. Here we consider a set of Pilates exercises for low back pain relief (refer to Fig. 1.3) as suggested by the therapists in Wan-Fang Hospital, Taipei. For each exercise, the graphical user interface will instruct the patient to wear sensors at appropriate body parts.

Fig. 1.4 shows the software architecture. The system consists of three parts: a wireless sensing platform, a motion analysis module, and a game interface. A patient needs to wear sensors on specified body parts. The motion analysis module evaluates the quality of a patient's movements and gives feedback through the game interface.

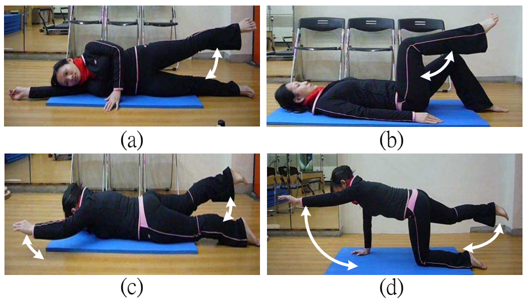


Figure 1.3: Pilates exercises for low back pain relief.

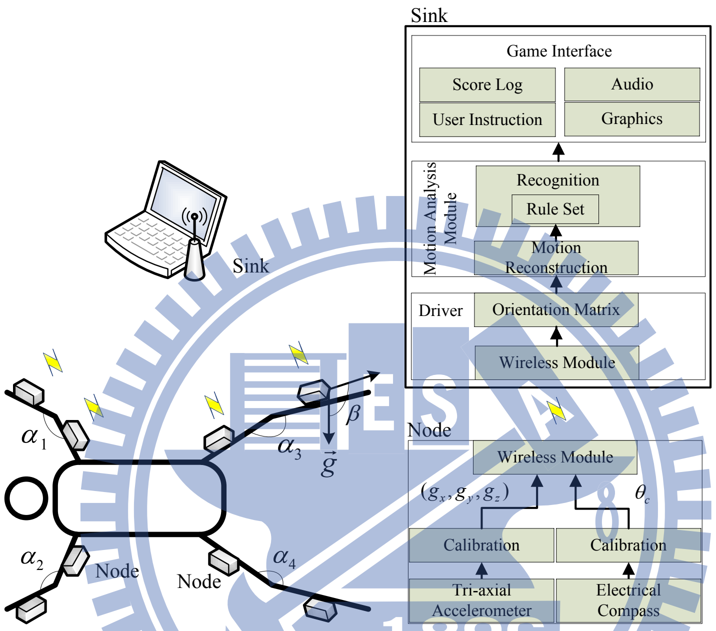
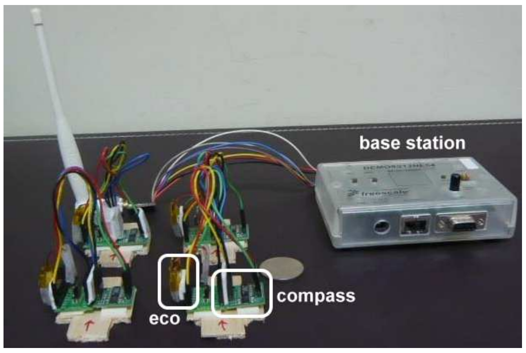


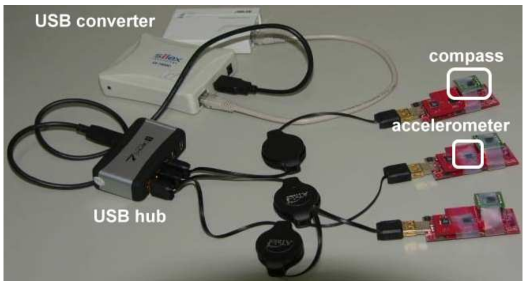
Figure 1.4: Software architecture.

Wireless sensing platform. We have tested two platforms: Octopus II and ECO systems (refer to Fig. 1.5). Octopus II is modified from the Tmote Sky system [10] and runs TinyOS, while ECO [11] is quite tiny and is extremely resource-limited. Both platforms use the TDCM3 electrical compass. Octopus II uses the FreeScale MMA7260QT accelerometer and ECO uses the Hitachi-Metal H34C accelerometer. These triaxial accelerometers can capture the gravity in x, y, z-axes. Note that a compass is needed since the rotation angle about the gravity cannot be inferred from the accelerometer along. Octopus II uses a TDMA protocol provided by TinyOS to increase data rate. On the other hand, ECO adopts a simple polling protocol. On the sink side, the accelerometer and compass readings from each node are further combined into orientation matrices in the world coordinate.

Motion analysis module. This module takes the accelerometer readings and the orientation matrices as input to reconstruct the patient's motions and gives scores. One major parameter of



(a) ECO system



(b) OctopusII system

Figure 1.5: Our wireless sensing platforms.

motions is the joint angle information ($\alpha_1, \ldots, \alpha_4$ in Fig. 1.4). Another is the angle between each sensor node and the gravity direction (β in Fig. 1.4). Our scoring mechanism analyzes these parameters to judge if the patient's performance is acceptable or not. For instance, the patient is considered to perform exercise (a) in Fig. 1.3 well if (1) the joint angle between two legs is around 45°, and (2) the movement of his pelvis is within a small range. Item (1) is implemented by restricting the joint angle within $45^\circ \pm 10^\circ$, and item (2) by restricting β, the angle between the frontal plane and the gravity direction, within a small range.

Game interface. This layer interacts with the user and stores the results of each exercise. The game has multiple levels, each as one Pilates exercise. The game will create some events, and the patient should react by performing the specified Pilates exercise. The game interface queries the motion analysis module to check if the patient's performance is acceptable, and if so, some rewards are given. The patient should get enough rewards to pass a level. Most of the hints are shown on the screen, but if the patient cannot watch the screen in some Pilates exercises, audio hints will be given instead.

In our demonstration, we show exercises (a) and (b) using Octopus II and ECO, respectively. In exercise (a), the game is to open a treasure box. The player should raise his leg to about 45° to open the lid and collect at least five treasures to pass this level. In exercise (b), the player should keep his knee joint angle to about 90° and raise two legs up simultaneously.

1.1.2 Prototyping Experience: Multi-Screen Cyber-Physical Video Game

Cyber-physical systems (CPS) have drawn a lot of attention recently. In the past two decades, cyber systems were one of the fastest growing areas due to the advance of mobile computing technologies (such as portable computers and smart phones) and the standardization of vari-

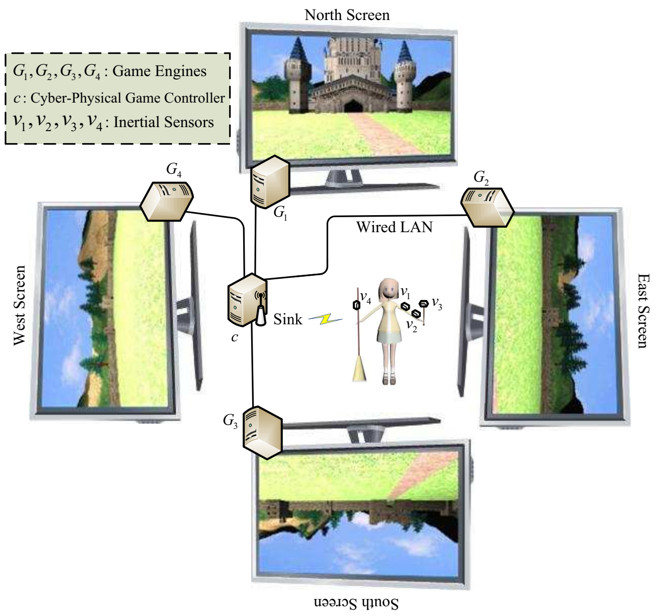

North Screen
G_1, G_2, G_3, G_4 : Game Engines
c : Cyber-Physical Game Controller
v_1, v_2, v_3, v_4 : Inertial Sensors
G_4
G_1
G_2
Wired LAN
West Screen
East Screen
Sink
c
v_4
v_1
v_3
v_2
G_3
South Screen

Figure 1.6: System architecture.

ous wireless communication and networking technologies (such as ZigBee, WiFi, Bluetooth, WiMAX, 2G/3G/3.5G, LTE, etc.), which are tightly integrated under the Internet environment. On the other hand, a major advance in physical systems is the development of *micro-electro-mechanical systems (MEMS)*, which can greatly enrich cyber systems with more physical inputs and actuators.

We are interested in CPS for cyber-physical video games. Traditionally, cyber video games are supported by game engines that take players' inputs from keyboards and joysticks. Enhancing such systems with more physical inputs, such as body motions captured by inertial sensors, has attracted a lot of attention recently. We call such systems *cyber-physical* games. The Nintendo Wii [2] is one example with MEMS-based inertial sensors. Reference [7] shows that by using MEMS-based inertial sensors and a wired network, body motions can be reconstructed through computer animation with little distortion. Wireless-based *body-area inertial sensor networks (BISNs)* have been studied in [12]. On the other hand, new display technologies, such as LCD, plasma, and back-projection displays, can further enrich cyber-physical games with better visual effects [13].

The above advances motivate us to develop a multi-screen cyber-physical video game with physical inputs from players equipped with BISNs. We demonstrate our prototype of the BISN platform, the Cyber-Physical Game Controller, and the Game Engines.

System description. Our system architecture is shown in Fig. 1.6. We consider a single-

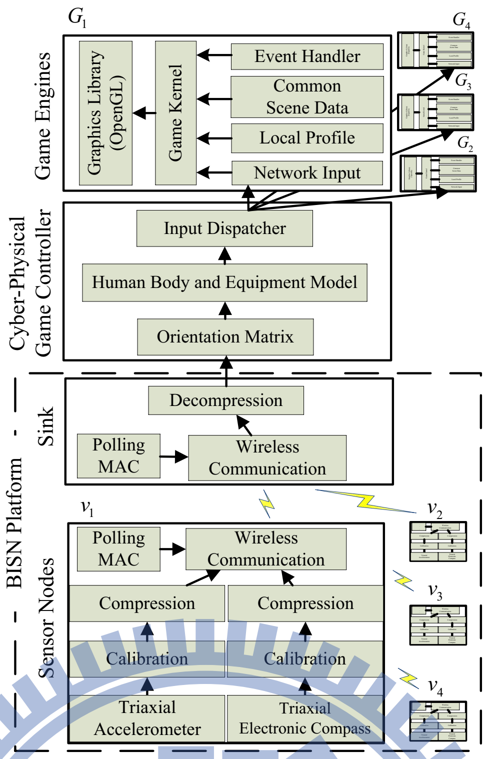


Figure 1.7: Software architecture.

player game. A BISN is deployed on the player. The sensing data is collected by the sink node attached to the Cyber-Physical Game Controller, which will process the sensing data and feed it to four game engines G_1, G_2, G_3, and G_4 via wired LAN interfaces. The four game engines are set to four camera angles (east, west, north, and south) and they together provide a $360°$ panorama view of the game scene. The software architecture is shown in Fig. 1.7. Below, we describe each component.

BISN platform. The platform consists of one sink node and four inertial sensor nodes v_1, v_2, v_3, and v_4. Each node consists of some inertial sensors, a micro-controller, and a wireless module. In our current implementation, each node is equipped with a triaxial accelerometer and a triaxial electronic compass. The triaxial accelerometer senses the gravity in x, y, and z-axes, while the triaxial compass senses the earth magnetic force in x, y, and z-axes. The "Calibration" module removes noises and converts sensing signal into meaningful data. Since sensing data is streaming data, the "Compression" module executes a differential coding to reduce data transmission. The compressed data is reported to the sink through the "Wireless Communication" module. The sink node runs a "Polling MAC" to avoid collisions and to improve throughput.

Cyber-physical game controller. This component has two main functions: 1) to convert

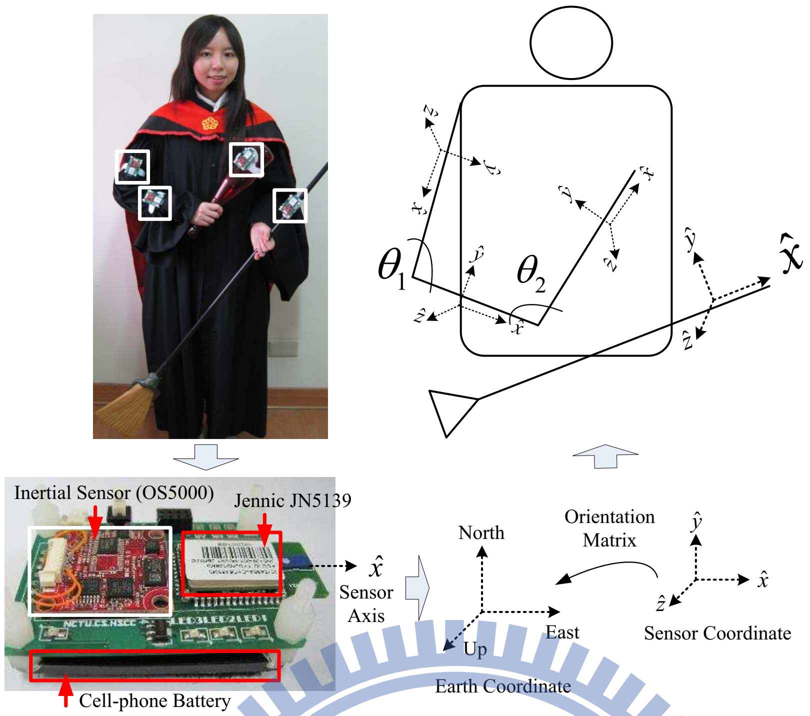


Figure 1.8: Human body and equipment model.

sensing data into game inputs (by "Orientation Matrix" and "Human Body and Equipment Model") and 2) to dispatch game inputs to each game engine (by "Input Dispatcher"). Fig. 1.8 illustrates the conversion process. The player wears four inertial sensor nodes, one on the broomstick, one on forearm, one on upper arm, and one on the club. From the sensing inputs, the "Orientation Matrix" represents the absolute orientations of all sensor nodes with respect to the earth coordinate. The "Human Body and Equipment Model" derives the direction of the broomstick (by its $\hat{x}$ axis) and two articulation angles θ_1 and θ_2 (by orientation matrices and limb lengths). These parameters are then fed to the game engines via wired LAN interfaces. Typical game engines do not support multiple screens. Our game controller tries to simulate a multi-screen game engine by sending proper data, via the "Input Dispatcher" to four game engines, as explained below.

Game engines. Each game engine has a "Game Kernel", which takes inputs from four modules. The "Common Scene Data" module describes the virtual world. It is the same for all game engines. The "Network Input" module receives game inputs from the "Cyber-Physical Game Controller" and updates the virtual world in the game. The "Event Handler" module processes the interactions, especially collisions, between virtual objects and the player. For example, when the player hits a ball, it bounces away, and the "Event Handler" increases the player's score. Each game engine contains a camera, whose direction is specified by the "Lo-

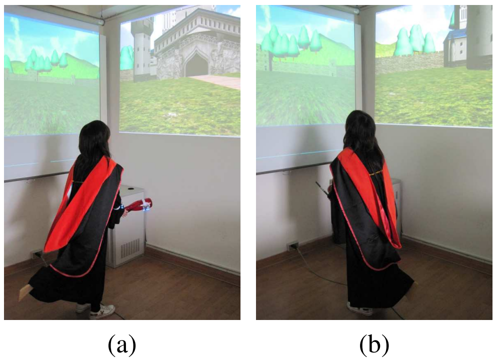


Figure 1.9: Snapshots of the game.

cal Profile", that takes pictures for its virtual world and feeds the captured data to the "Game Kernel", which calls the "Graphics Library" to display the results. Since we have four game engines with four cameras facing to east, west, north, and south, a 360° panorama view of the virtual world is provided to support better visual effect to the player.

Demonstration. We adopt the Quidditch sports in "Harry Potter" as our game scenario. The player rides a flying broom in a practice field, and she tries to attack approaching balls by her club. She flies at a constant speed and controls her broomstick to change directions.

For the BISN platform, we adopt Jennic JN5139 [14] and OS5000 sensor [15] (refer to Fig. 1.8). Each Jennic JN5139 consists of a micro-controller and a IEEE 802.15.4 wireless transmission module, and each OS5000 has a triaxial accelerometer and a triaxial electronic compass. The sampling rate is set to 20Hz, a common value for motion capturing [16]. The average packet loss rate of our polling MAC is about 0.16%, and the average delay for collecting a complete set of sensing data from a node is around 83 ms.

Fig. 1.9 shows some snapshots of the game. The player flies a short distance to the north in Fig. 1.9(a), and then she turns to the west in Fig. 1.9(b). We adopt Unity [17] as our game engine, which features a fully integrated editor and a physics engine for rapid 3D game prototyping. Fig. 1.10(a) and Fig. 1.10(b) show the corresponding settings of the west and the north game engines in Fig. 1.9(a), respectively. The virtual character, who is controlled by the player, looks to the north in both game engines. The camera of Fig. 1.10(a), whose viewing volume is illustrated by white lines, looks to the west, while the camera of Fig. 1.10(b) looks to the north. Fig. 1.10(c) and Fig. 1.10(d) correspond to the west and the north screens in Fig. 1.9(b), respectively, where both of the player and the virtual character look to the west, but the directions of the cameras remain unchanged.

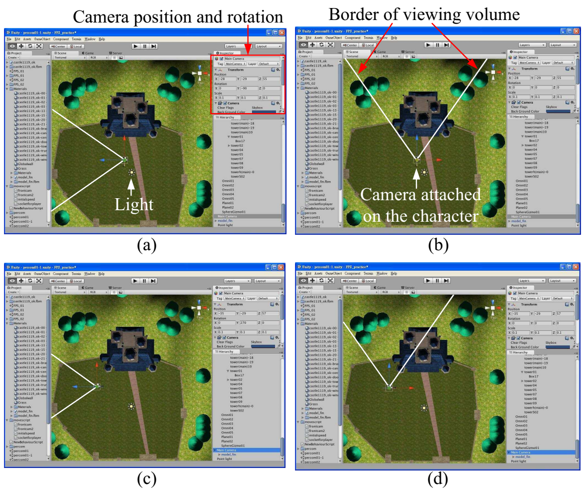

Camera position and rotation
Border of viewing volume
Light
Camera attached on the character
(a)
(b)
(c)
(d)

Figure 1.10: Virtual character and camera settings in Fig. 1.9.

1.2 Issues Addressed in this Dissertation

We conduct fundamental research into the technologies required to create an efficient wireless communication BISN that maximizes tracking accuracy and spectrum efficiency. Specifically, we conduct three pieces of research. The first work addresses the data collection issue by data compression by exploiting temporal-spatial correlations of sensing data. Based on the first work, the second work considers multi-spatial correlations of motion data to better exploit possible spatial correlations. In the third work, we address a fundamental issue in motion tracking, the gravity measurement problem. Below, we briefly describe each piece of our research.

Concerning that human bodies are relatively small and wireless packets are subject to more serious contention and collision, the first work of this dissertation addresses the data compression problem in a BSN. We observe that, when body parts move, although sensor nodes in vicinity may compete strongly with each other, the transmitted data usually exists some levels of redundancy and even strong temporal and spatial correlations. Unlike traditional data compression approaches for large-scale and multi-hop sensor networks, our scheme is specifically designed for BSNs, where nodes are likely fully connected and overhearing among sensor nodes is possible. We model the spatial-temporal data compression problem, where overhearing should be efficiently utilized, as a combinatorial optimization problem on graphs, showing its hardness, and presenting efficient algorithms to approximate optimal solutions. We also discuss the design of the underlying MAC protocol to support our compression model. An experimental case study in Pilates exercises for patient rehabilitation is reported. The results show that our schemes reduce more than 70% of overall transmitted data compared with existing approaches.

The first work has shown how to compress motion data by allowing a node to overhear at most $\kappa = 1$ node's transmission and exploit the correlation with its own data for data compression. In the second work, we consider *multi-spatial correlations* by extending $\kappa = 1$ to $\kappa > 1$ and constructing a partial-ordering *directed acyclic graph (DAG)* to represent the compression dependencies among sensor nodes. While a minimum-cost tree for $\kappa = 1$ can be found in polynomial time, we show that finding a minimum-cost DAG is NP-hard even for $\kappa = 2$. We then propose an efficient heuristic and verify its performance by real sensing data.

In addition to data collection, in the third work, we are interested in tracking human postures by deploying accelerometers on a human body. One fundamental issue in such scenarios is how to calculate the gravity. This is very challenging especially when the human body parts keep on moving. Fortunately, it is likely that there is a point of the body that touches the ground in most cases. This allows sensors to collaboratively calculate the gravity vector. Assuming multiple accelerometers being deployed on a rigid part of a human body, a recent work proposes a data fusion method to estimate the gravity vector on that rigid part. However, how to find the optimal deployment of sensors that minimizes the estimation error of the gravity vector is not addressed. In this work, we formulate the deployment optimization problem and propose two heuristics, called *Metropolis-based* method and *largest-inter-distance-based (LID-based)* method. Simulation and real experimental results show that our schemes are quite effective in finding near-optimal solutions for a variety of rigid body geometries.

To summarize, there are four significant innovations that resulted from this research:

1. Greatly improved spectrum efficiency of the wireless communications in a BISN by using spatial-temporal data compression on sensing data. This is accomplished by exploiting the spatial-temporal features inherent in human body motions and creating a data compression methodology that resulted in an energy-efficient, low-delay communication protocol that is suitable for real-time, accurate human motion capturing.

2. The theoretical findings that show the sharp increase in the computational complexity of exploiting spatial correlations. The optimal data compression scheme can be found in polynomial time when each node is restricted to overhear up to one other node; otherwise, allowing overhearing more nodes makes finding an optimal data compression scheme a NP-hard problem. We propose efficient heuristics to solve the NP-hard problem. The hardness results also apply to other types of WSNs.

3. The inertial sensor deployment scheme that maximizes tracking accuracy. Intuitively speaking, we place sensors on different locations of a target body part to collaboratively capture more information about its motions. We model the kinematics of sensors' readings and propose algorithms to find optimal deployments that capture the maximal amount of information, which can be fused to improve tracking accuracy.

4. The development of innovative applications using the motion tracking system to demonstrate its capabilities and to serve as a proof of concept for the technology in a variety of consumer usage models.

   - A cyber-physical video game where the user is able to control the game in a body-intuitive way, and is able to control the panoramic view of the game scenes in a sensor-controlled way.
   - A remote physical rehabilitation system that enables patients to avoid having to wait for rehabilitation appointments in the hospital, eliminating travel time, and reducing demands for medical resources.

1.3 Organization of this Dissertation

The rest of dissertation is organized as follows. Chapter 2 addresses data collection issues by exploiting data compression by taking advantage of the spatial-temporal features of sensing data when sensors are deployed on a human body and the possibility that sensors can overhear each others' transmissions in a BISN. Based on Chapter 2, Chapter 3 consider multi-spatial correlations of motion data for data compression. Besides data collection, Chapter 4 addresses a fundamental issue in human motion tracking, the gravity measurement problem. Chapter 5 concludes this dissertation.

Chapter 2

Data Compression by Temporal and Spatial Correlations in a BISN

2.1 Introduction

In Section 1.1.1, we have demonstrated applying BISNs to physical rehabilitation, where patients have to frequently practice certain fixed-pattern exercises repeatedly [5]. Such a scenario can be shown as in Fig. 2.1, where a BISN is deployed on a patient to capture her body motions. Via wireless links, the sink is able to collect sensing data, the data analyzer is able to analyze and visualize her motions, and the application system is able to give comments and send feedbacks to therapists for further diagnosis.

Such applications may require real-time and high-resolution sensing data for various purposes, such as visualizing motions, recognizing body conditions, and diagnosing diseases. These requirements imply shorter transmission delays and higher sampling rates for the BISN. In addition, due to the size of a human body, nodes are typically dense and almost fully connected, leading to severe contention and collision among their transmissions [8,9]. Since wireless bandwidths are limited, how to efficiently utilize them is very important.

We attack these issues by exploiting data compression by taking advantage of the special

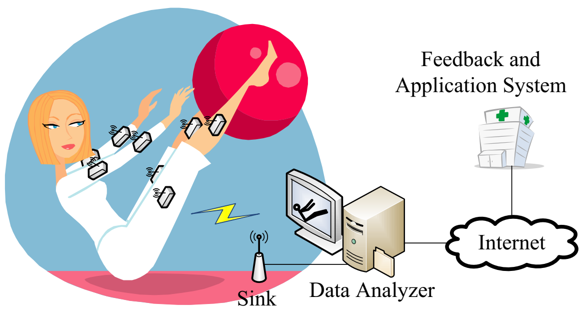


Figure 2.1: A BISN architecture for motion recognition.

features of sensing data when sensors are deployed on a human body and the possibility that sensors can overhear each others' transmissions in a BISN. We observe that when human body parts move, it usually exhibits strong temporal and spatial correlations among sensing data. By *temporal correlation*, body signs sensed by a single node typically change smoothly and slowly. By *spatial correlation*, body signs measured on different nodes are typically correlated because body components are connected and they normally move with certain rhymes. On the other hand, since nodes in a BISN are typically fully connected, spatial correlations can be efficiently exploited through overhearing, a nature of wireless communication, among nodes to minimize the total amount of transmissions. Although data compression schemes that exploit temporal and spatial correlations in wireless sensor networks (WSNs) have been widely discussed [18, 19], such schemes are usually designed for large-scale networks with multi-hop communications and are thus not suitable for BISNs, which are likely fully connected, providing a room for customized data compression of body motion data.

Since wearable devices are typically lightweight and have limited computation power and resources, the corresponding sensing data processing, compression, and MAC protocol should be lightweight, too. In particular, the following guidelines are important in designing data compression schemes for BISNs. First, the general form to model the temporal-spatial correlations of human body parts should not be too complicated, especially when considering the correlation among multiple sensors on different nodes. Second, we have to take the data gathering process and the distributed nature of a BISN into consideration. For example, the rich connectivity of a BISN normally incurs higher contention and collision among transmissions.

In this chapter, we propose a novel data compression method for BISNs based on overhearing. We exploit temporal and spatial correlations among sensing data to facilitate data compression and transmissions. Each node samples its data, overhears others' transmissions, compresses its data, and then transmits. Temporal and spatial correlations are modeled by differential coding and linear regression, respectively. Note that our data compression is lossless. In our method, an offline phase is conducted to learn these correlations and various system parameters. Then, nodes are sorted into a partial ordering tree, which specifies the preferred transmission order of nodes (level-1 nodes transmit first, followed by level-2 nodes, etc.). During the online phase, each node overhears others' transmissions, decodes some of them, encodes its own sample according to the learned correlations, and transmits the encoded result. It is to be noted that under our modeling, the compression ratio is indeed affected by nodes' transmission order. Our scheme can also tolerate that the actual transmission deviates from the ideally planned order, at the cost of a higher compression ratio. We show how to construct an optimal partial-ordering tree, called *Minimum-Cost Tree (MT)*, such that the total data transmitted on the air is minimal statistically. The MT scheme, however, may need to decode and encode a long sequence of overheard data. An alternative is a near-optimal *Minimum-Cost, Depth-Bounded Tree (MDT)*, which limits the number of transmissions to be overheard by each node. The MDT scheme is more practical, but unfortunately, it has been proven that optimizing tree cost given any bound on its depth is NP-hard.

To summarize, major contributions of this chapter are three-fold. First, a distributed data compression method suitable for BISNs is proposed to exploit temporal and spatial correlations of sensing data. Our method considers the correlation among multiple sensors on different nodes. The computation complexity and communication overhead are kept as low as possible. Second and more importantly, we utilize overhearing, a nature of wireless communication, and investigate the prediction directions in spatial correlations. We show how to construct an optimal partial-ordering tree, based on which nodes can sent their transmission sequences and overhear others' transmissions in a cooperative way so as to minimize the total amount of transmissions. To the best of our knowledge, applying overhearing in data compression is never explored before. We also propose a more practical minimum-cost, depth-bounded tree, which turns out to be NP-hard. These techniques may be applied to other distributed source coding, too. Third, to verify our results, an experimental case study in Pilates exercises for patient rehabilitation is reported.

The rest of chapter is organized as follows. Section 2.2 reviews some related works. Section 2.3 describes our BISN system architecture. Section 2.4 gives our data compression models and algorithms, including some notes on practical issues. Experimental results and our prototypes are shown in Section 2.5. Section 2.6 concludes this chapter.

2.2 Related Works

BSN is an emerging technique. Its bandwidth utilization issues have started to receive attention. Assuming a fully connected BSN, a QoS framework similar to IEEE 802.16 is proposed for BSNs in [20] to grant connections and allocate bandwidths based on applications' requests. Several lossy data compression methods have been proposed. In [21], sensors with higher priorities are allowed to transmit their sensing data with higher precision. If the data sources are sparse, e.g., in the case of PPG signals, compressed sensing techniques can be applied to reduce the required sampling rates [22]. For motion capturing, [12] has tested the performance of several existing lossless and lossy compression algorithms on motions that exhibit repetitive patterns, such as running. However, the compression is performed by each individual sensor and does not consider the correlations among sensors. We extend their work to consider correlations among sensors (both on the same node and on different nodes). Although our data compression model is lossless, any lossy transform (e.g., FFT and wavelet transform mentioned in [12]) can be applied to the data source before passing through our data compression model. If the application is interested only in certain events, rather than the sensing values, distributed motion classification/annotation would be most efficient [23, 24].

On the other hand, for large-scale WSNs, data compression has been intensively studied. We classify these solutions as follows.

- **Cross-layer solutions.** The impact of joint routing and compression is studied in [25]. Reference [26] studies the feasibility of sublinear scaling laws under different cooperation schemes for spatially correlated data.

- **Data aggregation.** Data compression may be performed by relay nodes or cluster heads via data aggregation. TAG [27] organizes a network into a tree and proposes SQL-like semantics to aggregate streaming data into histograms for low-power aggregation. Multi-resolution spatial and temporal coding is addressed in [28].

- **Data acquisition.** As acquiring sensing data from a large-scale WSN to a user's hand-held device may involve long delays, non-uniform energy consumptions, and network impairments, model-driven data acquisition models are proposed in [18, 19, 29]. Although these methods are suitable for infrequent queries and slow-changing environments, they are not practical for motion capturing, as motion data usually changes more rapidly.

2.3 BISN System Architecture

We consider a BISN deployed on a human body for motion recognition. An example is the Pilates exercises for patient rehabilitation. The system architecture is shown in Fig. 2.1. Wearable sensor nodes are placed at main movable body parts. These nodes will periodically report their readings to a sink node, which is connected to the data analyzer. The data analyzer will try

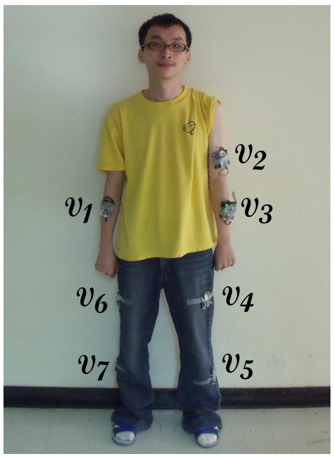


Figure 2.2: An example of placing sensor nodes on a human body.

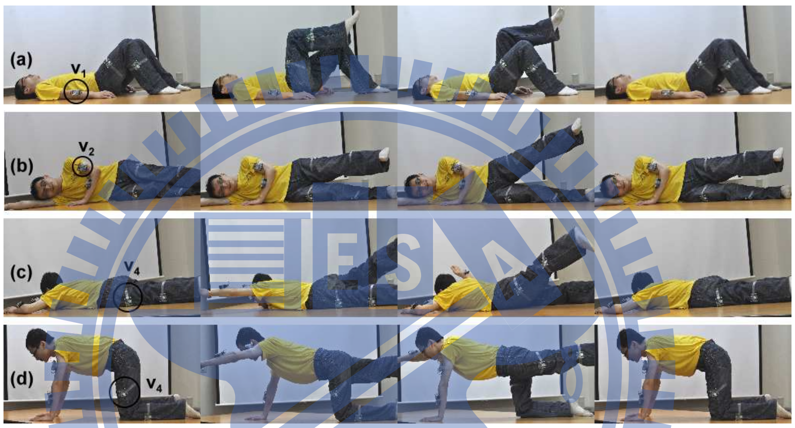


Figure 2.3: Examples of Pilates exercises.

to understand how well the person conducts some motions. The application server can further send feedbacks to the person and reconstruct, store, and replay these motions.

As an example, to recognize Pilates exercises for lower back pain relief [30, 31], nodes can be placed at main body parts as shown in Fig. 2.2. Some Pilates exercises are shown in Fig. 2.3. In the upper leg lifting exercise (b), one has to lie on his side and put legs and feet together in parallel. Then, lift the top leg up from $0°$ to about $45°$, stay for about 5 seconds, and then return to the start. The exercise needs to be repeated several times.

Due to the size of a human body, nodes in a BISN are typically fully connected. Therefore, when a node is transmitting, the other nodes can overhear its transmission. In addition, their data could be highly correlated. By utilizing the overheard information, one may conduct compression based on sensor data's spatial and temporal correlations. We show that by properly arranging nodes' transmission order, not only the competition and interference among nodes can be relieved, but also significant compression can be done. In this work, we place a triaxial accelerometer in each sensor node. The sampling rate on each axis is set to 20 Hz, a common value for body motion measurement [16]. (However, the applicability of our work is not limited to these assumptions.) Sensor readings are calibrated and compressed locally, but the computation should be kept as simple as possible.

2.4 Design of Data Compression by Overhearing

We first present our basic idea in Section 2.4.1. Then we present our data compression model and two algorithms to sort nodes' overhearing sequences in Section 2.4.2 and Section 2.4.3, respectively. Although our model does not require to change the underlying MAC protocol, we discuss the corresponding design issues in Section 2.4.4. Finally, in Section 2.4.5, we discuss how to retrain the system where there are deviations between the training parameters and the actual behaviors.

2.4.1 Basic Idea

We consider a BISN with n nodes $v_1, v_2, \ldots, v_n$, each equipped with m sensors. Note that in the case of triaxial accelerometer, each axis is regarded as one separate sensor. We will design a compression scheme to exploit the *temporal* and *spatial correlations* of sensor readings. The goal is to automatically learn these correlations to facilitate compression. Our system has an *offline phase* and an *online phase*. During the offline phase, we will collect the raw outputs of the jth sensor of v_i into a column vector $R_i^{(j)}$, where the tth sampling result is stored at the tth entry in the vector and is written as $R_i^{(j)}[t]$. Note that since the same Pilates exercise should be repeated multiple times by the same person to make the training results more representative, $R_i^{(j)}$ should contain the result of several repetitions of the same Pilates exercise. Then $R_i^{(j)}$ is calibrated into a column vector $X_i^{(j)}$ of the same size carrying meaningful data. Note that we do not specify the size of vector $R_i^{(j)}$. A larger vector means more statistics, leading to higher accuracy in our prediction of correlations.

Temporal correlation hereby means the similarity of sensing values over the time axis, i.e., $\Delta X_i^{(j)}[t] = X_i^{(j)}[t] - X_i^{(j)}[t-1]$ is very likely to fall within a small range relative to that of the original value $X_i^{(j)}[t]$. Previous researches [12,16] also show that body movements are normally centered at very slow frequencies and do not change rapidly in nearby samples.

Spatial correlation hereby means that human motions in nearby body parts show inherent rhythm, making it a good source for data compression. Specifically, one may easily predict

$\Delta X_i^{(1)}$, $\Delta X_i^{(2)}$, ..., $\Delta X_i^{(m)}$ given $\Delta X_k^{(1)}$, $\Delta X_k^{(2)}$, ..., $\Delta X_k^{(m)}$, if nodes v_i and v_k are attached to correlated body parts. To achieve low-complexity compression, we would investigate the possibility of establishing the following linear correlation:

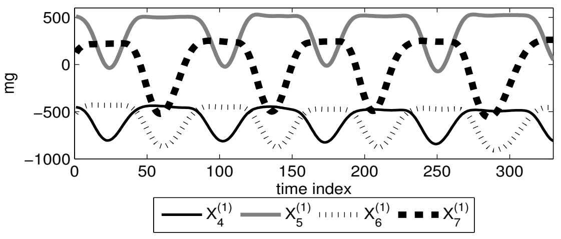
$$\begin{aligned}\Delta X_i^{(j)} = \alpha_{(i,j)|k}\mathbf{1} + \beta_{(i,j)|(k,1)}\Delta X_k^{(1)} + \\ \beta_{(i,j)|(k,2)}\Delta X_k^{(2)} + \cdots + \beta_{(i,j)|(k,m)}\Delta X_k^{(m)} + \\ \epsilon_{(i,j)|k},\end{aligned} \tag{2.1}$$

where $\alpha_{(i,j)|k}$ and $\beta_{(i,j)|(k,1)}, \ldots, \beta_{(i,j)|(k,m)}$ are scalars, $\mathbf{1}$ is a column vector containing all 1s, and $\epsilon_{(i,j)|k}$ is a column vector to compensate the prediction errors. If the outputs of some sensor of v_k have little correlation with $\Delta X_i^{(j)}$, its corresponding β should approach zero. With properly selected coefficients α and β, vector $\epsilon_{(i,j)|k}$ would contain all very small values, making it much less costly to represent $\epsilon_{(i,j)|k}$ than $\Delta X_i^{(j)}$. The number of bits required to represent $\epsilon_{(i,1)|k}$, $\epsilon_{(i,2)|k}$, ..., $\epsilon_{(i,m)|k}$ is used as the cost to compress sensing data of v_i when the data of v_k is known. Intuitively, we try to recover v_i's data from v_k's data. In subsequent sections, we will show how to find the optimal predictor v_k of each node v_i (by building a partial ordering tree) so as to minimize the overall cost.

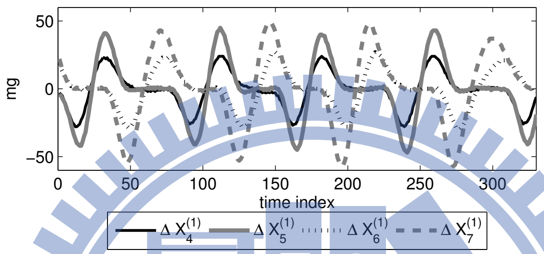
On the other hand, sensors of the same node v_i (such as $\Delta X_i^{(j)}$ and $\Delta X_i^{(k)}$) may also exhibit strong spatial correlations, especially among the three axes of a triaxial accelerometer. We call such spatial correlations *intranode spatial correlations*, and those among sensors of different nodes *internode spatial correlations*. So Eq. (2.1) is generalized as follows to include intranode spatial correlations:

$$\begin{aligned}\Delta X_i^{(j)} = \alpha_{(i,j)|k}\mathbf{1} + \sum_{p=1}^{m} \beta_{(i,j)|(k,p)}\Delta X_k^{(p)} + \\ \sum_{q \in L_{(i,j)|k}} \beta_{(i,j)|(i,q)}\Delta X_i^{(q)} + \epsilon_{(i,j)|k},\end{aligned} \tag{2.2}$$

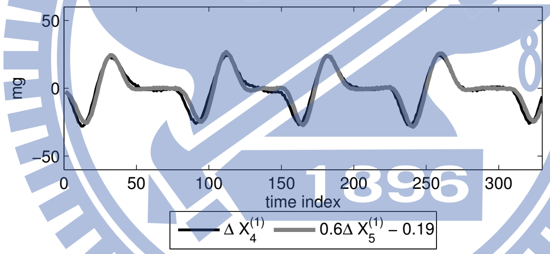
where we use all sensors of v_k and a set $L_{(i,j)|k}$ of sensors of v_i to predict $\Delta X_i^{(j)}$. Note that to conduct data compression, we must avoid circular dependency among sensors of the same v_i. $L_{(i,j)|k}$ is to specify the set of sensors of v_i whose sensing data are encoded before that of sensor j. Later on, we will show how to determine the set $L_{(i,j)|k}$. By learning the intranode spatial correlations in Eq. (2.2), the resulting error vector $\epsilon_{(i,j)|k}$ of Eq. (2.2) is expected to be even smaller than that of Eq. (2.1), making compressing the former more efficient than compressing the latter.



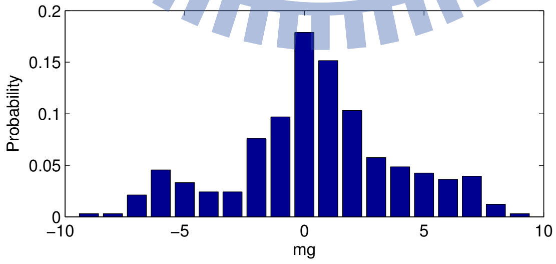
(a)



(b)



(c)



(d)

Figure 2.4: Validation of our temporal and spatial correlation modeling. (a) Calibrated data $X_i^{(1)}$ of exercise Fig. 2.3(a). (b) The difference vector $\Delta X_i^{(1)}$. (c) Predicting $\Delta X_4^{(1)}$ by $\Delta X_5^{(1)}$. (d) Probability distribution of the prediction error vector $\epsilon_{(4,1)|5}$.

Our experiences show that the above modeling is quite effective in recognizing Pilates exercises. Consider the sensor deployment in Fig. 2.2 and the exercise in Fig. 2.3(a) and assume that each node has only one x-axis accelerometer. Fig. 2.4(a) shows the calibrated data $X_4^{(1)}, \ldots, X_7^{(1)}$. We see that $X_4^{(1)}$ and $X_5^{(1)}$ exhibit strong spatial correlations, and so are $X_6^{(1)}$ and $X_7^{(1)}$. By taking the difference operation on $X_i^{(1)}$, Fig. 2.4(b) shows that $\Delta X_i^{(1)}$ has a much smaller range, and the aforementioned spatial correlations still exist. So we can apply Eq. (2.2) to predict $\Delta X_4^{(1)}$ by $\Delta X_5^{(1)}$, as shown in Fig. 2.4(c). Fig. 2.4(d) shows the corresponding probability distribution of the contents of the error vector $\epsilon_{(4,1)|5}$. As can be seen, the distribution concentrates around 0mg.

By overhearing among sensor nodes, we will investigate the feasibility of learning these correlations in the offline phase and exploring real-time data compression in a BISN during the online phase. We will also address the issue of the supporting MAC protocol, especially the design of packet collision, in Section 2.4.4.

2.4.2 Data Compression Model

Data flows of our offline and online phases are shown in Fig. 2.5. Temporal correlation will be exploited on individual sensors by a simple differential coding, while spatial correlation will be exploited across sensors and will be learned from the offline phase. The learned results consist of some coefficients and some transmission and encoding orders, which are sent to the sensor nodes for them to conduct online data compression. The *transmission order*, which is represented by a spanning tree, specifies the partial ordering of nodes' transmissions and the target nodes that a node should overhear. To simplify the notations, we will abbreviate $R_i^{(1)}, \ldots, R_i^{(m)}$ by $R_i^{(1:m)}$, $\Delta X_i^{(1)}, \ldots, \Delta X_i^{(m)}$ by $\Delta X_i^{(1:m)}$, $\beta_{(j,q)|(i,1)}, \ldots, \beta_{(j,q)|(i,m)}$ by $\beta_{(j,q)|(i,1:m)}$, $\epsilon_{(i,1)|j}, \ldots, \epsilon_{(i,m)|j}$ by $\epsilon_{(i,1:m)|j}$, $\beta_{(j,q)|(j,l)}$ for all $l \in L_{(j,q)|i}$ by $\beta_{(j,q)|(j,L_{(j,q)|i})}$, and so on.

Offline phase. In the offline phase, for each type of motion, training data are collected from the BISN in the "Data Collection" block and then sent to an external data analyzer, where the rest of the blocks will be executed.

Data collection. We will collect from nodes $v_1, v_2, \ldots, v_n$ column vectors $R_1^{(1:m)}$,$R_2^{(1:m)}$, $\ldots$, $R_n^{(1:m)}$. In the collection process, each node v_i puts the tth sampling result of its qth sensor into $R_i^{(q)}[t]$. $R_i^{(1:m)}$ should be stored in v_i and then forwarded to the sink in a reliable way to avoid the impact of network impairments. The same Pilates exercise should be repeated

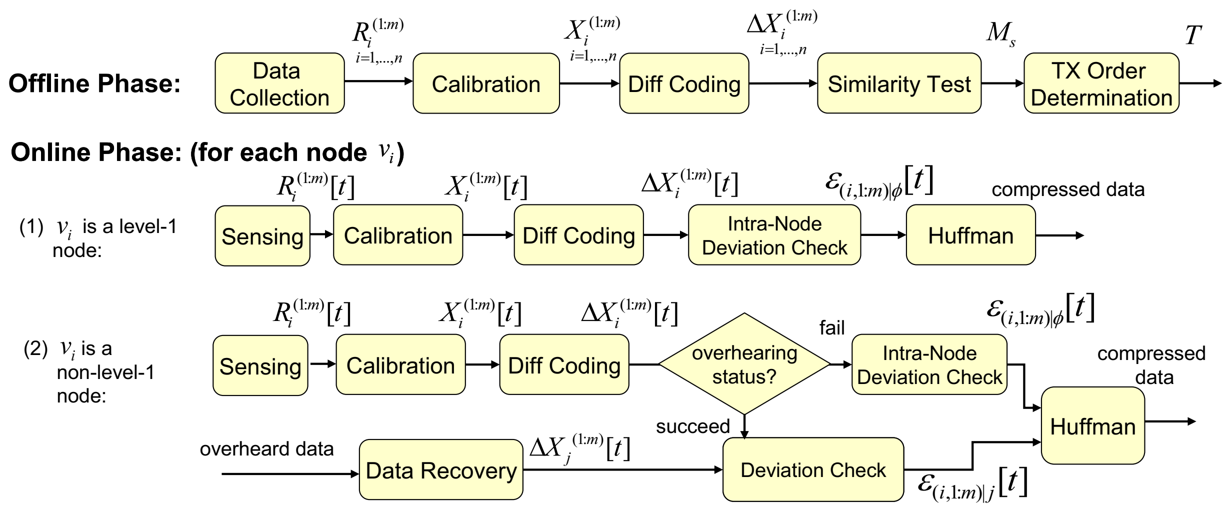


Figure 2.5: Offline and online phases of our data compression model.

multiple times by the same person to make the training results more representative, i.e., $R_i^{(j)}$ should contain the results of several repetitions of the same Pilates exercise. To make the learned results effective, it is assumed that the same Pilates exercise is performed by the same person in the online phase.

Calibration. Each $R_i^{(j)}$, $i = 1 \dots n$, $j = 1 \dots m$, is converted into a meaningful vector $X_i^{(j)}$. For example, the raw outputs of an accelerometer may be converted from voltages into physical units (e.g., Guass). Then we may apply various data filtering techniques to refine the data. For human motion, we propose to use a low-pass filter to remove noise. We adopt the FIR filter based on Kaiser window function [32]. A low-pass filter will zero out the signal components whose frequencies are higher than a predefined stopband frequency. As the energy of the filtered signal components become larger, the resulting signal waveform will become smoother. To automatically determine the stopband, we apply a fast Fourier transform to $R_i^{(j)}$ such that the removed energy is equal to γ percent of the total energy. We will test different values of γ in Section 2.5. The filtered data are rounded to the nearest integers and recorded in $X_i^{(j)}$.

Diff coding. To exploit temporal correlation, each $X_i^{(j)}$ is converted into a differential column vector $\Delta X_i^{(j)}$ such that $\Delta X_i^{(j)}[t] = X_i^{(j)}[t] - X_i^{(j)}[t-1]$.

Similarity test. Our goal is to identify the correlation between the outputs of each pair of v_i and v_j. Specifically, an $n \times n$ similarity matrix M_s will be constructed. $M_s[i][j]$ is to measure how well we can predict the data of v_j given the data of v_i. For ease of explanation, for now we assume that $L_{(j,1:m)|i}$ is given (we will explain how to find it later). To compute $M_s[i][j]$, we apply Eq. (2.2) and try to build the following equality for each $q = 1, 2, \ldots, m$, where $t = 1, 2, \ldots$:

$$\Delta X_j^{(q)}[t] = \alpha_{(j,q)|i} + \sum_{p=1}^{m} \beta_{(j,q)|(i,p)} \Delta X_i^{(p)}[t] + \\ \sum_{l \in L_{(j,q)|i}} \beta_{(j,q)|(j,l)} \Delta X_j^{(l)}[t].$$

Then we can compute $\alpha_{(j,q)|i}$, $\beta_{(j,q)|(i,1:m)}$ and $\beta_{(j,q)|(j,L_{(j,q)|i})}$ for each q by linear regression analysis [33]. Note that linear regression only tries to use the coefficients on the right-hand side to approximate $\Delta X_j^{(q)}[t]$, so we let the difference $\epsilon_{(j,q)|i}[t] = \Delta X_j^{(q)}[t] - \alpha_{(j,q)|i} - \sum_{p=1}^{m} \beta_{(j,q)|(i,p)} \Delta X_i^{(p)}[t] - \sum_{l \in L_{(j,q)|i}} \beta_{(j,q)|(j,l)} \Delta X_j^{(l)}[t]$. Intuitively, if $\Delta X_i^{(1:m)}$ is obtained by overhearing, we only need to transmit the difference vector $\epsilon_{(j,q)|i}$ to obtain $\Delta X_j^{(q)}$. Here we use the notation $\epsilon_{(j,q)|\phi}$ to represent that v_i's transmission is not based on any overhearing. Let the offline and online distributions of $\epsilon_{(j,q)|i}$ be $f_{(j,q)|i}$ and $g_{(j,q)|i}$, respectively. Assuming that the Huffman encoding [34] is applied, we will use the distribution of $\epsilon_{(j,q)|i}$ measured during the offline phase to transmit the data measured during the online phase. So we let $M_s[i][j]$ be the total of entropies, $\sum_{q=1}^{m} H(f_{(j,q)|i})$. To compute this term, we also need to determine the encoding sequence of the m sensors of v_j, i.e., $L_{(j,q)|i}$, $q = 1 \ldots m$. We develop a greedy heuristic as follows. We work in the backward direction starting from the last sensor. Specifically, for $q = 1 \ldots m$, we compute $H(f_{(j,q)|i})$, assuming that $L_{(j,q)|i}$ contains all other sensors $\neq q$. Then the sensor q with the minimum entropy is selected as the last one in the sequence. We then exclude this sensor and repeat the the same process for the rest of the $m - 1$ sensors. After determining the encoding sequence, $M_s[i][j]$ can be obtained.

Note that when $i = j$, $M_s[i][i]$ is a special case, which means that v_i is the first node to transmit and cannot overhear any information. In this case, we will ignore the factor of internode correlation in Eq. (2.2) and only consider intranode correlation. So we reduce Eq. (2.2) to the following:

$$\Delta X_i^{(q)} = \alpha_{(i,q)} \mathbf{1} + \sum_{l \in L_{(i,q)}} \beta_{(i,q)|(i,l)} \Delta X_i^{(l)} + \epsilon_{(i,q)|\phi}. \tag{2.3}$$

With a similar greedy process, we can compute $M_s[i][i]$. So the whole matrix M_s is obtained.

TX order determination. Based on M_s, the main goal of this block is to find a proper order of nodes' transmission time based on the correlations of their data to achieve the best compression ratio. The ordering instructs a node v_i to sequentially overhear the transmissions of some earlier nodes, so as to compute v_i's error vector $\epsilon_{(i,1:m)|j}$ with respect to v_j's data. In the offline phase, assuming ideal error-free transmissions, we model the ordering problem as one of finding a spanning tree T along a directed weighted complete graph $G = (\{u\} \cup V, E)$, where u is a virtual node, $V = \{v_1, v_2, \dots, v_n\}$, and E contains all possible edges between nodes. The directed edge $\langle v_i, v_j \rangle \in E$ means "predicting $\Delta X_j^{(1:m)}$ from $\Delta X_i^{(1:m)}$" and the directed edge $\langle u, v_i \rangle \in E$ means "sending $\Delta X_i^{(1:m)}$ to the sink directly without overhearing". So the weight of $\langle v_i, v_j \rangle$ is $w(\langle v_i, v_j \rangle) = M_s[i][j]$ and the weight of $\langle u, v_i \rangle$ is $w(\langle u, v_i \rangle) = M_s[i][i]$. Since virtual node u will not overhear any node, we let $w(\langle v_i, u \rangle) = \infty$ for all v_i. Note that T is an outgoing directed tree and is always rooted at u. In Section 2.4.3, we will propose two solutions to construct T.

Online phase. The online phase is performed by each node in a distributed manner (refer to Fig. 2.5). Consider the transmission of the tth sampling result of v_i. There are two cases: v_i is a level-1 node and v_i is a non-level-1 node. In both cases, it will go through the "Sensing" block to collect the raw data and the "Calibration" and the "Diff Coding" blocks to obtain $\Delta X_i^{(1:m)}[t]$. If v_i is a level-1 node in T, $\Delta X_i^{(1:m)}[t]$ will go through the "Intranode Deviation Check" block and the resulting $\epsilon_{(i,1:m)|\phi}[t]$ will be compressed by the "Huffman" block and then transmitted without overhearing. If v_i is a non-level-1 node, then it has to overhear all precedent nodes along the path from the root u to itself in T. If all these data are overheard successfully, the sequence of blocks "Data Recovery", "Deviation Check", and "Huffman" will be activated, as shown in the lower path of Fig. 2.5. However, it is possible that v_i may miss some expected overhearing targets for some reasons. In this case, v_i should transmit without going through overhearing; it activates the "Intranode Deviation Check" block and the "Huffman" block, as shown in the upper path of Fig. 2.5.

Intranode deviation check. In the above discussion, v_i has to transmit without overhearing. From $\Delta X_i^{(1:m)}[t]$, v_i follows Eq. (2.3) to calculate $\epsilon_{(i,q)|\phi}[t] = \Delta X_i^{(q)}[t] - \alpha_{(i,q)} - \sum_{l \in L_{(i,q)}} \beta_{(i,q)|(i,l)} \Delta X_i^{(l)}[t]$ for $q = 1, \dots, m$ to obtain $\epsilon_{(i,1:m)|\phi}[t]$.

Data recovery. Let $u \to v_{j_1} \to v_{j_2} \to \cdots \to v_{j_p} \to v_i$ be the path in tree T from the root u to v_i. Node v_i has to overhear the transmissions by $v_{j_1}, v_{j_2}, \dots, v_{j_p}$ and recover the original $\epsilon_{(j_1,1:m)|\phi}[t]$, $\epsilon_{(j_2,1:m)|j_1}[t]$, $\epsilon_{(j_3,1:m)|j_2}[t], \dots, \epsilon_{(j_p,1:m)|j_{p-1}}[t]$ through Huffman decoding. If all the necessary data are overheard successfully, the lower path of Fig. 2.5 is executed. Otherwise, it follows the upper path of Fig. 2.5 (i.e., this block and the subsequent "Deviation Check" block are ignored). In the former case, v_i first applies $\Delta X_{j_1}^{(q)} = \alpha_{(j_1,q)}\mathbf{1} + \sum_{l \in L_{(j_1,q)}} \beta_{(j_1,q)|(j_1,l)} \Delta X_{j_1}^{(l)} + \epsilon_{(j_1,q)|\phi}$ for $q = 1, \dots, m$ to obtain $\Delta X_{j_1}^{(1:m)}$. Then it follows the encoding sequence and repeatedly applies $\Delta X_{j_{k+1}}^{(q)}[t] = \alpha_{(j_{k+1},q)|j_k} + \sum_{r=1}^{m} \beta_{(j_{k+1},q)|(j_k,r)} \Delta X_{j_k}^{(r)}[t] + \sum_{l \in L_{(j_{k+1},q)}} \beta_{(j_{k+1},q)|(j_{k+1},l)} \Delta X_{j_{k+1}}^{(l)}[t] + \epsilon_{(j_{k+1},q)|j_k}[t]$ for each of its own sensor q, and for $k = 1, \ldots, p-1$, until $\Delta X_{j_p}^{(1:m)}[t]$ are obtained.

Deviation check. From $\Delta X_i^{(1:m)}[t]$ and $\Delta X_{j_p}^{(1:m)}[t]$ (obtained by the "Data Recovery" block), v_i calculates $\epsilon_{(i,q)|j_p}[t] = \Delta X_i^{(q)}[t] - \alpha_{(i,q)|j_p} - \sum_{r=1}^{m} \beta_{(i,q)|(j_p,r)} \Delta X_{j_p}^{(r)}[t] - \sum_{l \in L_{(i,q)|j_p}} \beta_{(i,q)|(i,l)} \Delta X_i^{(l)}[t]$ for $q = 1, \ldots, m$ to obtain $\epsilon_{(i,1:m)|j_p}[t]$.

Huffman. If the lower path of Fig. 2.5 is taken, v_i follows its encoding sequence to encode $\epsilon_{i^{(q)}|j_p}[t]$ for each $q = 1 \ldots m$ by Huffman encoding, puts the encoding results together, and transmits the compressed data. Note that the average compressed data size should be pretty close to the total entropies of $\epsilon_{(i,1:m)|j_p}$ computed in the offline phase if the exercises are conducted similarly to the offline phase. If the upper path of Fig. 2.5 is taken, v_i also encodes $\Delta X_i^{(q)}[t]$ for each $q = 1 \ldots m$ by Huffman encoding, puts the encoding results together, and transmits the compressed data.

2.4.3 Algorithms for Constructing the TX Order Tree T

Recall that in the "TX Order Determination" block in the offline phase, we are given a directed weighted complete graph G, in which each edge represents the corresponding message volume when a node's transmission is based on overhearing another node's transmission. Our goal is to form a TX order tree T from G for the best transmission efficiency. We will propose two solutions. The first one, called *Minimum-Cost Tree (MT)*, can achieve optimal compression ratio, but may require a node to overhear and decompress the transmissions of multiple nodes before it can transmit its own data. The second one, called *Minimum-Cost, Depth-Bounded Tree (MDT)*, allows a node to bound the number of nodes that it has to overhear and is thus more practical for wireless environments with non-negligible transmission errors. Note that given a T and a path $u \rightarrow v_i \rightarrow v_j \rightarrow v_k \rightarrow \ldots$ in T, the underlying meanings is: "v_i transmits to the sink directly", "v_j overhears v_i's transmission, recovers v_i's original data, and compresses its own data", "v_k overhears v_i's transmission, recovers v_i's original data, overhears v_j's transmission, recovers v_j's original data, and compress its own data", etc. For v_k, overhearing only v_j's transmission is not sufficient because v_j's data is compressed data and can be decompressed only if v_i's original data is known. This chain effect is not preferred, especially for BISNs.

The MT scheme works by finding an outgoing tree T rooted at u from G with the smallest total edge weight. We adopt an efficient polynomial-time algorithm [35, 36] to find T, which works similarly to Kruskal's algorithm for undirected graphs. The algorithm greedily selects incoming edges with minimum weights and breaks cycles, if any.

1. We will pick a set S of edges from G to form the edges of T. Edges entering u are first removed from G. For all incoming edges of each v_i, select the one with the smallest weight into S. Now, S contains n edges. For each $v_i \in V$, we denote by $\text{prev}(v_i)$ the (only) node such that $\langle \text{prev}(v_i), v_i \rangle \in S$.

2. If S contains no cycles, then $T = (\{u\} \cup V, S)$ is an optimal tree and the algorithm terminates. Note that T is connected because it is acyclic and every node except the root u has an incoming edge. Otherwise, continue to the following steps to break the cycles. (Note that there are many polynomial time algorithms to identify cycles in a graph.)

3. Find any cycle C in S. For each edge $\langle v_i, v_j \rangle \in E$ such that $v_i \notin C$ and $v_j \in C$, we re-weight it as follows:

$$w'(\langle v_i, v_j \rangle) = w(\langle v_i, v_j \rangle) - w(\langle \text{prev}(v_j), v_j \rangle).$$

Note that $w'(\langle v_i, v_j \rangle) \geq 0$ because $\langle \text{prev}(v_j), v_j \rangle$ has the smallest weight among all v_j's incoming edges in E (see step 1).

4. From all re-weighted edges in step 3, let $\langle v_i, v_j \rangle$ be the one with the smallest weight $w'(\langle v_i, v_j \rangle)$. Break the cycle by deleting $\langle \text{prev}(v_j), v_j \rangle$ from S and adding $\langle v_i, v_j \rangle$ to S.

5. After breaking the cycle, edges in C do not need to be considered any more. So we adjust G by contracting the nodes in C into a pseudo-node v_k, by removing all nodes and edges in C from G and adding a v_k to G. For each $v_i \notin C$, we let weight $w(\langle v_k, v_i \rangle) = \min\{w(\langle v_j, v_i \rangle), v_j \in C\}$ and weight $w(\langle v_i, v_k \rangle) = \min\{w'(\langle v_i, v_j \rangle), v_j \in C\}$, where w' means the new weight in step 3.

6. Go to step 2 with the contracted graph G.

It is proved in [35, 36] that the T found above is the minimum outgoing tree. For example, consider the network in Fig. 2.6(a). The solid arrows are selected into S in step 2. There is one cycle containing nodes v_1, v_2, and v_3. The cycle is contracted into a pseudo-node v_5 in Fig. 2.6(b). We delete $\langle v_1, v_2 \rangle$ from S and add $\langle u, v_2 \rangle$ to S. Weights of edges to and from v_5 are recalculated. After one iteration, tree T is found.

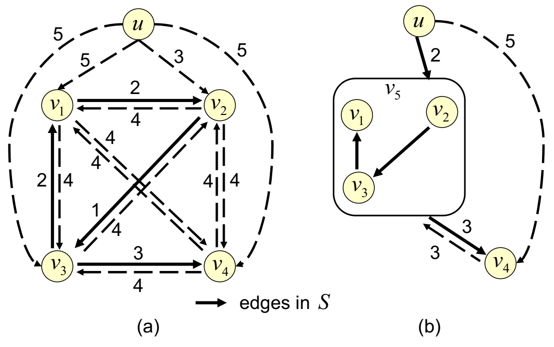


Figure 2.6: An example of the MT method.

The second MDT scheme enforces that the depth of T be bounded by a constant δ. For example, if we set $\delta = 2$, then a level-1 node can transmit anytime and a level-2 node only needs to overhear its previous node's transmission. Limiting T's depth would be more practical for BISNs. However, it turns out that optimizing this "bicriteria lowest-cost, bounded-depth" problem is NP-hard [37]. It has been proved that approximating this problem cannot be achieved within a log factor and an approximation algorithm is proposed in [37], which can bound the cost within a factor of $O(logn)$ of the optimal solution. Several heuristics for finding bounded-diameter minimum spanning trees in an undirected graph have been proposed (refer to [38] and the references therein). Although their goals are similar to our MDT problem in that bounding the depth of the tree to δ can be enforced by bounding its diameter to 2δ, they are only applicable to undirected graphs. So we will adopt the solution in [37] in our simulation studies. It remains an open issue to find better heuristics for the MDT problem.

2.4.4 Design Issues of the Underlying MAC Protocol

The above overhearing and transmitting activities can rely on any general wireless MAC protocols to achieve our goal of data compression. The only requirement would be to support assigning priorities to nodes so as to utilize the partial ordering tree T. For example, one may use a polling-based protocol, a prioritized CSMA, or even a polling-based protocol on top of an existing underlying MAC protocol. However, if modifying the underlying MAC is possible, better performance may be achieved. Below, we comment on the desired changes on existing MAC protocols to facilitate our overhearing behavior. Our discussions will consider both polling-based and CSMA-based protocols.

- *Backoff window.* Recall the partial ordering tree T. Nodes should transmit their sensing data according to their levels in T. If a polling-based MAC is adopted, no change is needed; the sink node can simply poll nodes level-by-level. If a CSMA-based MAC is adopted, then we can assign nodes' backoff timers based on their levels. A lower-level node should adopt a smaller backoff window, while a higher-level one should adopt a larger backoff window. (This is similar to the design of IEEE 802.11e, which uses differential backoff windows.)

- *Packet loss behavior.* Next, we consider the packet loss issue. Packet loss may happen to the sink as well as a node which intends to overhear the packet. For loss at the sink node, if a polling-based MAC is used, it can simply re-poll the sender; if a CSMA-based MAC is used, the sender will automatically retransmit following the protocol's definition. For loss at a node intending to overhear the packet, no matter which kind of MAC is adopted, when the opportunity for it to transmit appears before it already overhears all needed packets that it intends to overhear, it runs into a dilemma of whether to transmit or not. However, a node missing the packet that it intends to overhear should not request for retransmission because this would further complicate the problem (e.g., reference [39] also suggests avoiding retransmitting lost packets because doing so may violate the delay requirement). There are two options. One way is to repeat the backoff process, expecting that all needed packets that it intends to overhear would arrive correctly later on. The other way is to still transmit but assume that it is a level-1 node (i.e., without compression by overhearing). We would recommend using the second option because the former approach may encounter a cascaded effect and even a dilemma where a packet may have been correctly received by the sink but simply missed by the node. Also note that our data collection is lossless. Missing intended overhearing packets at a node only increases the compression ratio, but the accuracy of collected data is unaffected.

- *Data aggregation.* In the above discussion, to keep our presentation simple, we have focused on each piece of data individually and discussed its compression. To improve bandwidth utilization, it is better to pack several (compressed) sensing data into one packet instead of sending multiple small packets. This would depend on the maximum payload size agreed on the underlying MAC protocol. Since our overhearing scheme can further reduce the size of transmitted data, any data aggregation scheme can directly benefit from our scheme. (Note that since we adopt Huffman coding, which is a prefix-free coding, no delimiter character needs to be inserted between two pieces of encoded data.)

2.4.5 Retraining the System

So far, we have assumed that the training data in the offline phases can precisely reflect the behaviors during the online phase. In practice, there may exist some degree of mismatch in the predicted correlations, thus causing inefficiency. Specifically, there could be two major sources of such mismatch. One is that the trained Huffman codes may not match the online probability distribution of the ϵ functions. The other is that the trained αs and βs may drift from the actual coefficients during the online phase. Fortunately, these problems are solvable because our data compression model is lossless, implying that the online user motions are always recoverable, as long as the communication channel remains working. So long as the user's motions are repetitive, retraining the system by the data analyzer is always possible.

When the learned probability distributions of the ϵ functions do not match with the online distributions, the amount of additional bits in the online transmission is known as the relative entropy [34]. We may use adaptive Huffman coding [40] to measure current distributions, adjust the Huffman trees accordingly, and forward the results to the sensor nodes.

Sometimes the online spatial correlations (αs and βs) may not match with the learned results for many reasons. For example, the user may simply perform a different motion, or the system is worn by a different person. (Note that this is less serious in the case of Pilates exercises because they are a set of standard and repetitive motions.) To make our scheme adaptive, the data analyzer can keep a window of past sensing data in its database, periodically compute new αs and βs, and update them to sensor nodes whenever mismatch is detected.

2.5 Experiment Results: A Case Study in Pilates Exercises

To verify the effectiveness of the proposed compression method, we have conducted some experiments on Pilates exercises. We collect sensing data, split them into a training data set and a test data set, build the offline data compression model by the training data set, and then analyze performance by the test data set.

2.5.1 Experiment Setup and Data Collection

We conduct experiments on the four Pilates exercises as shown in Fig. 2.3. We place $n = 7$ nodes on appropriate body parts to monitor human motions (refer to Fig. 2.2). Each node is a Jennic JN5139 single-chip microprocessor [14] with a 16-MIPS RISC CPU, 192KB ROM, 96KB RAM, a ZigBee-compliant module, and an OS5000 triaxial accelerometer [15]. Note that we regard a triaxial accelerometer as $m = 3$ sensors. We index the x-, y-, and z-axis accelerometers as sensors 1, 2, and 3, respectively. Each piece of raw data in each axis is 16 bits, and the sampling rate is set to 20 Hz. To fairly compare different methods, we collect sensing data from the BISN, calibrate it on a PC, and also test different methods on the same set of calibrated data on the PC. Each exercise is repeated 20 times by the same person, and the sensing results, which consist of thousands of sensing values, are stored in nodes and then forwarded to the sink in a reliable way to avoid the impact of network impairments. Two-third of the calibrated sensing data are used as the training data set, and the rest one-third as the test data set. We use *compression ratio* (size of the compressed data divided by that of the uncompressed data) as our main performance metric.

For each exercise, we compare three data compression methods.

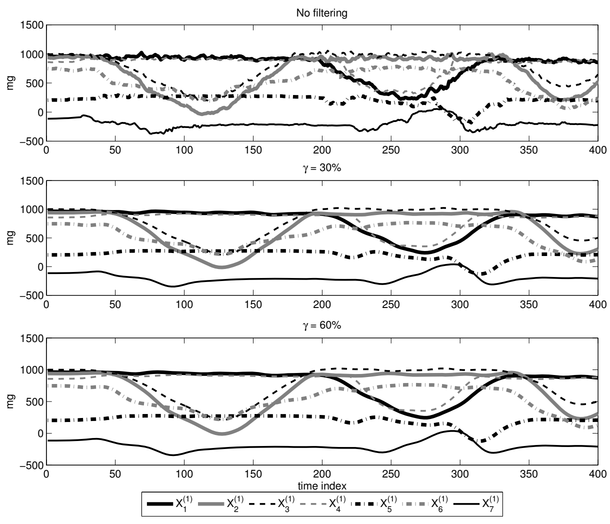
- **Zlib.** This lossless method is used in [12] to compress motion data. It first accumulates pieces of sensing data in a fix-sized buffer, compresses the buffer by the zlib compression algorithm once it is full, and transmits the compressed buffer directly to the sink. So it compresses data in a block-by-block fashion. Note that it has no sense of overhearing in its design.

- **MT.** This is our MT method.

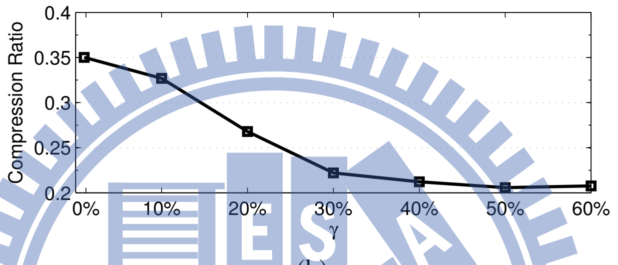
- **MDT.** This is our MDT method with depth bounded by $\delta = 2$. The MDT tree is found by the approximation method in [37].

- **Diff.** To see the improvement made by exploiting spatial correlations, this method utilizes only differential coding, intranode correlations among sensors, and Huffman encoding. Technically, it is our MDT method with depth bounded by $\delta = 1$.

To make comparison, we also implement a method with no data compression, called *uncompressed* method. Note that we realize the "Huffman" block based on the implementations recommended by [41,42]. The other blocks, from our experiences, only take a few multiplications and additions to complete. This is also true for the FIR filter in the "Calibration" block. So the feasibility of our scheme has been verified by our prototyping. The performance part done on a PC is mainly for comparing data compression ratio and energy consumption of different methods (Since we do not have the implementations of other methods on our sensor platform).



(a)



(b)

Figure 2.7: The effect of γ. (a) waveforms of $X_i^{(1)}$ before and after the low-pass filter. (b) compression ratio vs γ.

This PC-based simulation can correctly capture the intended metrics, except network behaviors (which will be modeled by some network simulations).

2.5.2 Effects of γ

In the "Calibration" block, the parameter γ of the low-pass filter tries to remove noises while retain information of interest. Note that its purpose is to smooth out the raw data; it does not change the size of a piece of sensing data. Fig. 2.7(a) shows the low-pass-filtered $X_i^{(1)}$ in the x-axis of exercise Fig. 2.3(d) for $\gamma = 0\%$, 30%, and 60% (the results for y-axis and z-axis are similar and are not shown here). Fig. 2.7(b) shows the compression ratios achieved by different γs when the MT method is applied. Considering both accuracy and compression ratio, we will set $\gamma = 30\%$ in the rest of our experiments.

2.5.3 Compression Ratio under Ideal Channel Condition

First, we consider an ideal wireless channel. That is, the effect of network impairments, such as transmission errors, is ignored here. For MT, MDT, and Diff methods, we build the data compression models by the training data set and measure their online compression ratios by the test data set. For each method, the offline phase outputs a transmission scheduling tree T, the corresponding coefficients αs and βs, and the Huffman codewords for each sensor. Note that all nodes in the Diff method are level-1 nodes, so there is no overhearing. These results are applied to the online phase to compress the test data set. The "Huffman" block adopts the adaptive Huffman coding. For the Zlib method, we buffer the test data in every four seconds in a block and then conduct compression (note that in Zlib, the block size is a trade-off between compression ratio and delay [12]). Note that this is not an issue for our methods because our methods can perform compression on each piece of sensing data $\Delta X_i^{(1:m)}[t]$.

Fig. 2.8 shows the compression ratios achieved by individual nodes. Fig. 2.9(a) compares the overall compression ratios after summing up the data transmitted by all nodes. The corresponding partial-ordering trees are shown in Fig. 2.9(b). Our MT, MDT, and Diff outperform Zlib significantly in all nodes and all exercises. Note that nodes that move more frequently exhibit higher compression ratios. Also note that Diff can be regarded as a special case of our scheme which only tries to exploit the correlation of intranode data. So these performance curves show the importance of exploiting the correlation of internode data. For most nodes, MT outperforms MDT by small gaps (for very few nodes, MDT may outperform MT because MT tries to minimize the average of compressed data sizes). From Fig. 2.9(b), we also see that the number of level-1 nodes of a partial-ordering tree has major impact on the compression ratio of a scheme (Diff can be regarded as consisting of only level-1 nodes). This observation again proves the importance of exploiting spatial correlations. We believe that this is even more important when the network becomes larger. While the overall performance of MT is always better than MDT, MDT is much simpler and is more robust against network impairments, as to be shown later on.

2.5.4 Compression Ratio under Non-Ideal Channel Condition

Since MT and MDT rely on overhearing, we intend to evaluate how packet loss may degrade the performance of our methods (note that when a node fails to overhear the packets that it expects to receive, it will simply compress its data based on temporal and intranode spatial

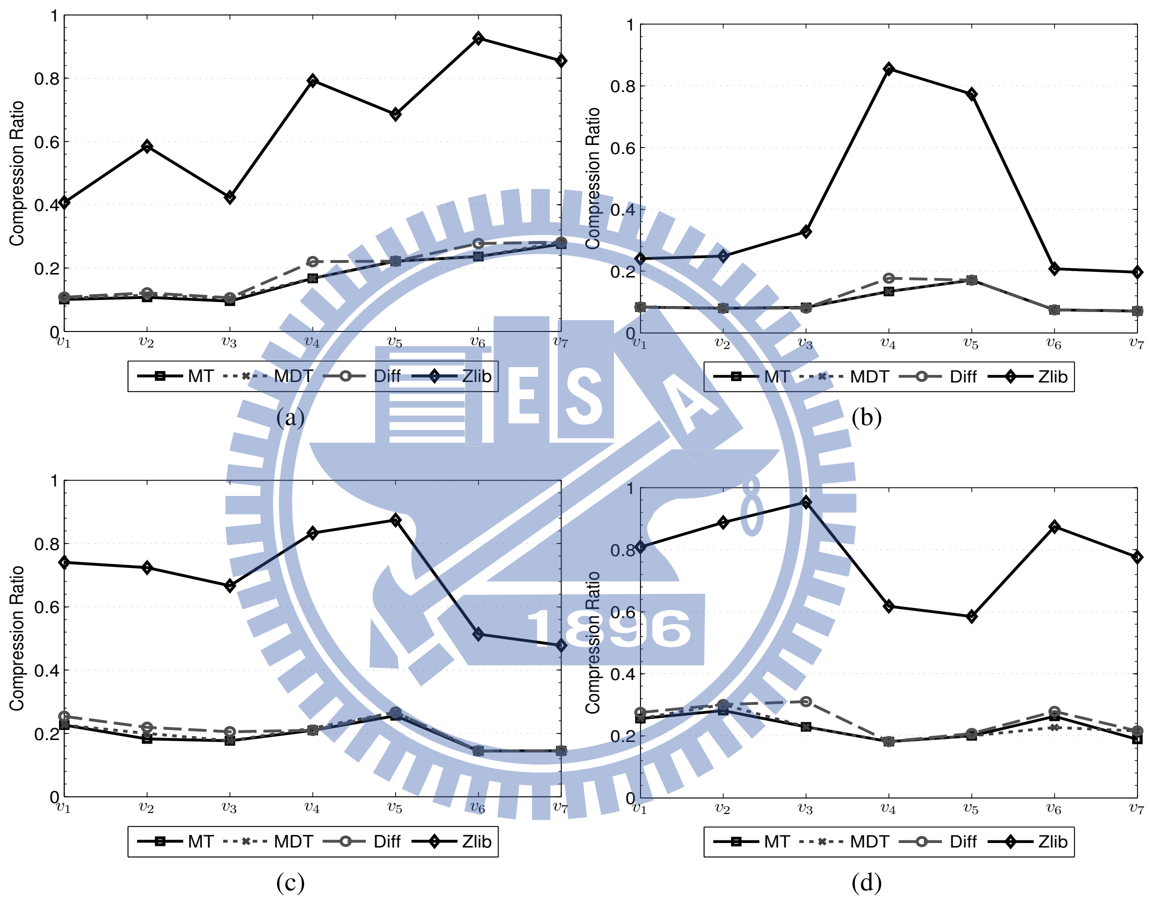
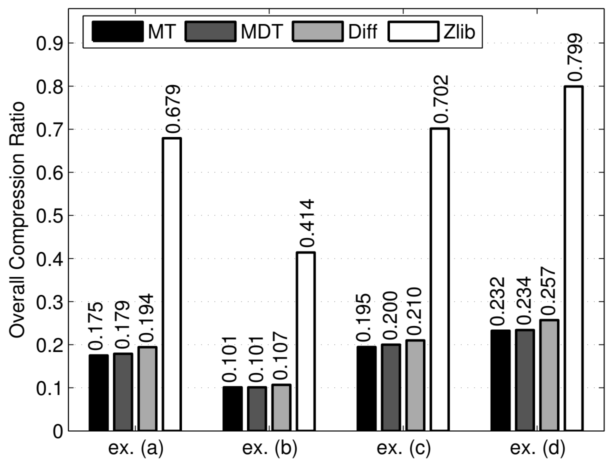
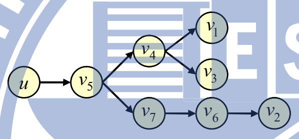
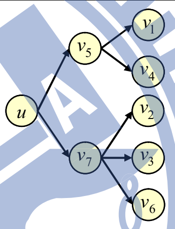
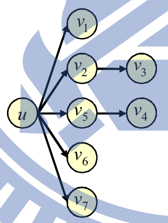
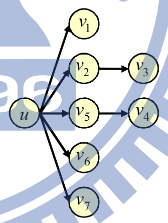
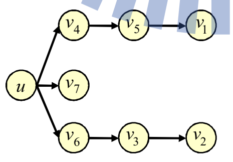
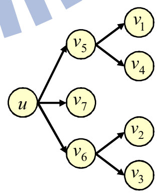
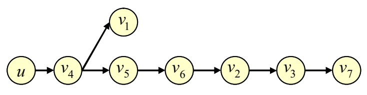
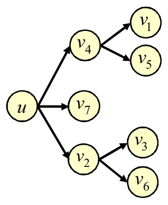


Figure 2.8: Comparison of compression ratios of individual nodes (v_1, v_2, ..., v_7) under an ideal wireless channel. (a) exercise Fig. 2.3(a). (b) exercise Fig. 2.3(b). (c) exercise Fig. 2.3(c). (d) exercise Fig. 2.3(d).



(a)

exer cise	MT	MDT
(a)		
(b)		
(c)		
(d)		

(b)

Figure 2.9: Comparison of overall compression ratios and the partial-ordering trees being used. (a) overall compression ratio. (b) partial-ordering trees.

correlations). The packet loss probabilities of wireless links typically vary over time and space, depending on the environment [43]. We simulate the network behavior by MATLAB because the probability of packet loss in a real test-bed is not controllable. In our simulation, we set a packet loss probability p_l for each wireless link (i.e., the network is fully connected, but each link has a loss probability of p_l) and vary p_l for the whole network to observe the compression performance. We believe that using the same p_l for all links suffices because our BISN is not large (with eight nodes) and varying p_l for individual links may bias the observed results.

We use a simple polling MAC, which works as follows.

1. The sink node sorts nodes of the partial-ordering scheduling tree T (shown in Fig. 2.9(b)) by a BFS traversal. The result is the polling order.

2. In each round, the sink node polls nodes according to the above polling order. It asks each polled node v_i to either report $\epsilon_{(i,1:m)|\phi}$ or $\epsilon_{(i,1:m)|j_p}$ depending on whether the sink node could recover the received data.

3. Upon receiving the polling request, the polled node broadcasts its reply packet, which contains 10 consecutive pieces of sensing data. All other nodes, including the sink node, have a probability of $(1 - p_l)$ to correctly receive the packet.

4. The sink node retransmits its polling request at most twice in case of packet loss, and then polls the next node.

Note that in the above MAC protocol, the event of packet loss is solely based on each node's own observation of success/failure (and thus has the same loss probability of p_l). We run the simulation 1000 times and calculate the overall compression ratio over all nodes. The polling message sent by the sink node is a one-byte packet consisting of the node ID of the polled node and the expected type of reply ($\epsilon_{(i,1:m)|\phi}$ or $\epsilon_{(i,1:m)|j_p}$). The reply packet (for MT and MDT) contains a 9-byte MAC header and a variable-size payload (say, $\epsilon_{(i,1:m)|j_p}[t], \epsilon_{(i,1:m)|j_p}[t+1], \ldots, \epsilon_{(i,1:m)|j_p}[t+9]$). To eliminate the impact of packet loss, the MAC header always includes the first piece of the original sensing data, i.e., $X_i^{(1:m)}[t]$. For the uncompressed method, the reply packet contains a 2-byte MAC header and a fix-size payload (say, $X_i^{(1:m)}[t], X_i^{(1:m)}[t+1], \ldots, X_i^{(1:m)}[t+9]$).

Fig. 2.10 shows the overall compression ratios and the corresponding overhearing percentages for each exercise under different methods, where the overhearing percentage is defined as

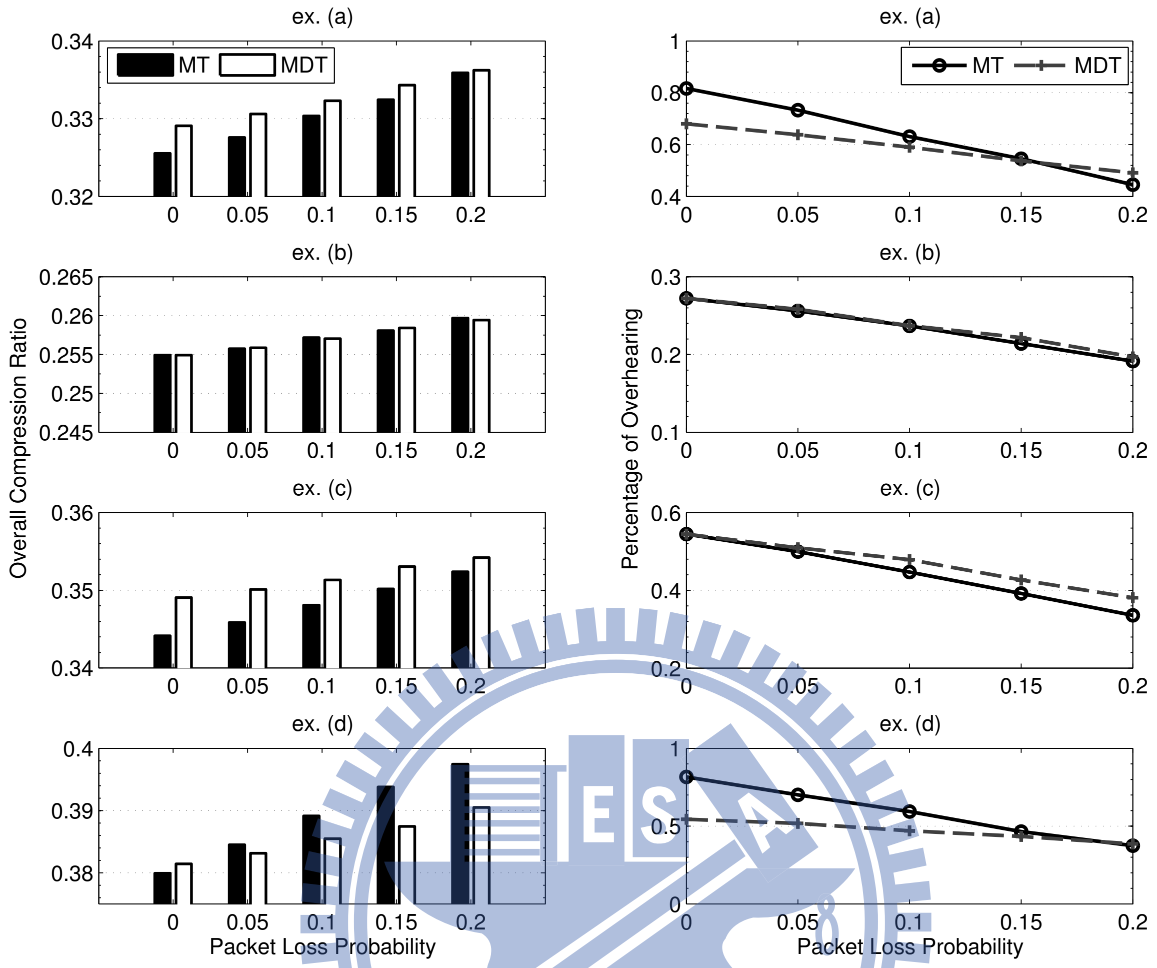


Figure 2.10: The effect of wireless packet loss.

the number of reply packets that send $\epsilon_{(i,1:m)|j_p}$ divided by the total number of reply packets. Intuitively, a higher percentage of overhearing will help reduce the overall compression ratio. As we can see, as p_l increases, both MT's and MDT's compression ratios will increase, but MDT will gradually outperform MT as p_l increases.

2.5.5 Energy Cost

As energy consumption is an important factor in BISNs, we conduct experiments to observe how our methods perform with respect to this. We adopt the polling MAC protocol in Section 2.5.4. Table 2.1 summarizes the parameters used in this evaluation. The round length of the polling MAC will depend on the sampling rate. In the beginning of each round, each node has to wake up. The sink node then polls nodes according to the polling order. In the case of packet loss from the polled node, the sink can re-poll at most two more times. Loss is determined by the polling timeout parameter. Each node has to keep on listening to the channel until it receives a polling request. After being polled and transmitting its data, the node can turn off its transceiver until the beginning of the next round. Note that after transmission, a node has to wait an additional polling timeout interval before turning off its transceiver to make sure that the sink will not re-poll it.

Table 2.1: Energy-Related Parameters Used in Our Evaluation

Parameter	Value
Round length	250 ms
Polling timeout	10 ms
Polling retry limit	2
Overhead of uncompressed MAC	2 bytes
MAC overhead of MT (MDT)	9 bytes
PHY overhead	16 bytes
RF reception power	35.5 mW
RF transmission power	35.5 mW
RF idle power	10 mW
Bit time	4 μs
RX/TX switching time	10 μs

The power consumption of a sensor node mainly comes from its computing module, sensor module, and wireless module. Here, we are mainly interested in the power consumption of the wireless module because the other factors are irrelevant to our work. The parameters in Table 2.1 are mainly based on those in [44, 45], which consider IEEE 802.15.4-compatible transceivers operating in 2.4 GHz with a data rate of 250 kbps. A node can switch among three modes: reception, transmission, and idle modes. When being turned off, a wireless module consumes no power. The bit time is the reverse of the data rate. The RX/TX switching time specifies the time interval required for a reception followed by an immediate transmission to account the nodal processing time.

We mainly observe the average energy consumption per node per polling round. Assuming the same initial battery energies for all nodes, it follows that the node that exhausts the maximal average energy per round dies first. In Fig. 2.11, we run our simulation 1000 rounds for each exercise and show the maximal per-node, per-round energy consumption among all nodes under different packet loss probabilities. As we can see, both MT and MDT save over 40% of energy as opposed to the uncompressed case.

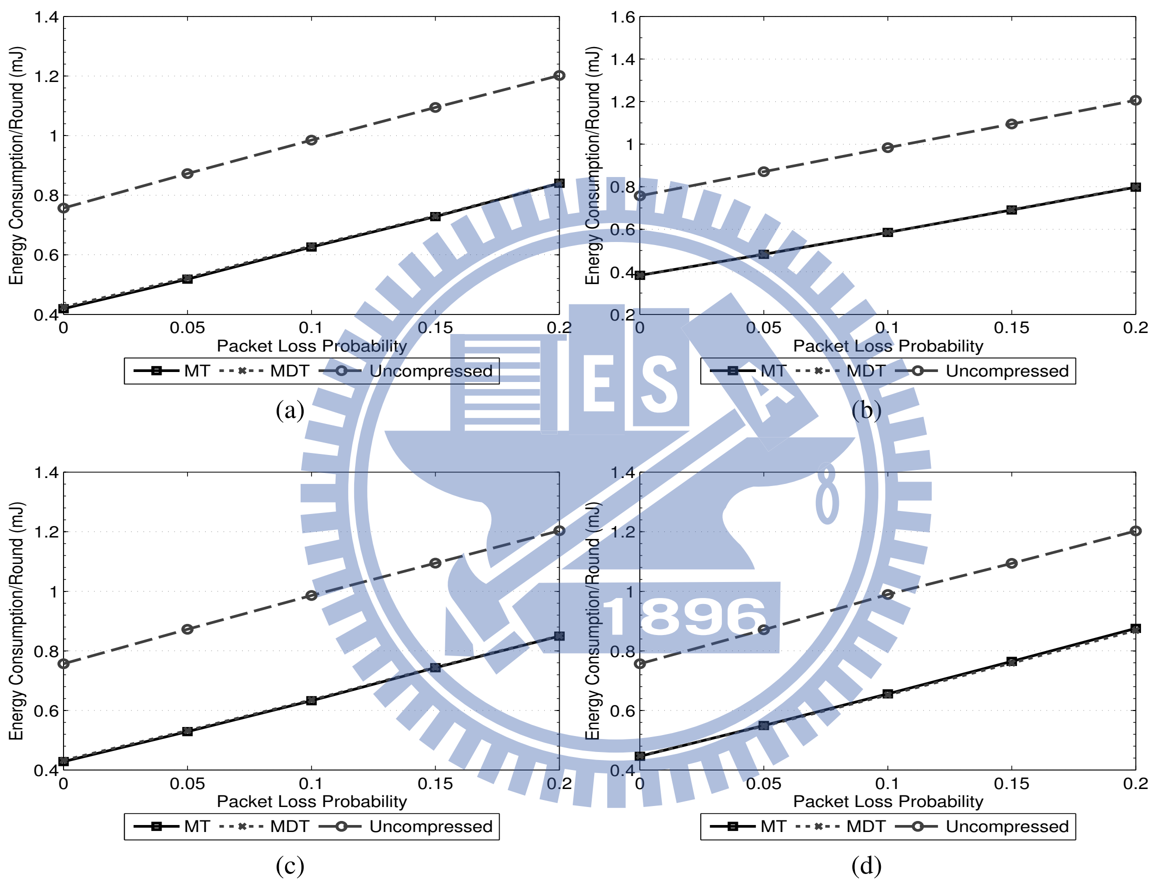


Figure 2.11: Comparison of the maximal average per-node energy consumption among all nodes for each exercise. (a) exercise Fig. 2.3(a). (b) exercise Fig. 2.3(b). (c) exercise Fig. 2.3(c). (d) exercise Fig. 2.3(d).

2.5.6 Scalability Issue

In addition, given a fixed channel bandwidth, the overall compression ratio also reflects how many sensor nodes may co-exist in a BISN. So this also implies the scalability of a method. For example, in Fig. 2.10, the compression ratios of both MT and MDT are upper-bounded by 0.4. This implies that our methods can roughly tolerate a BISN that is 2.5 times that of the uncompressed method. Of course, the impact of contention and collision among nodes are not taken into consideration. Since our work is not targeted at MAC design, we will leave this issue for future study.

2.5.7 Effect of System Retraining

In the previous experiments, we have assumed that the training and testing data are collected from the same person. In reality, a set of training data may be used to serve a different person who performs exercises in a different fashion. In this case, the system needs to be retrained, as discussed in Section 2.4.5, when we find deviations between the training parameters and the actual behaviors. Fortunately, our data compression is lossless, which means that any deviation can be accurately captured. The cost is a higher compression ratio before the retraining is done. We evaluate this problem below.

We conduct this experiment with two users. A window-based method is adopted to keep the most recent sensing data. Before the network starts, the initial data compression model is built by the training data of one user. Recall that a model includes coefficients αs and βs in Eq. (2.2), Huffman codewords, and partial ordering trees. Then the second user conducts exercises on the system. Upon receiving new sensing data, the data analyzer inserts it at the end of the sliding window and removes the same amount of old data from the window. In this evaluation, a sliding window which can keep 20 seconds of sensing data is used. Originally, it contains sensing data from the first person. Testing data is divided into blocks of length of five seconds. These blocks will gradually replace all data of the first user. In Fig. 2.12, we evaluate the overall compression ratio under ideal channels before the ith block is moved into the sliding window. To make comparison, we also show overall compression ratios where the initial data compression models are trained by the actual user who performs the online testing (i.e., the second person). We can see that there is a higher compression ratio before retraining starts. The gap diminishes as the second person's data gradually replaces the first person's in the sliding window. After the fifth block, the sliding window is completely filled with the actual

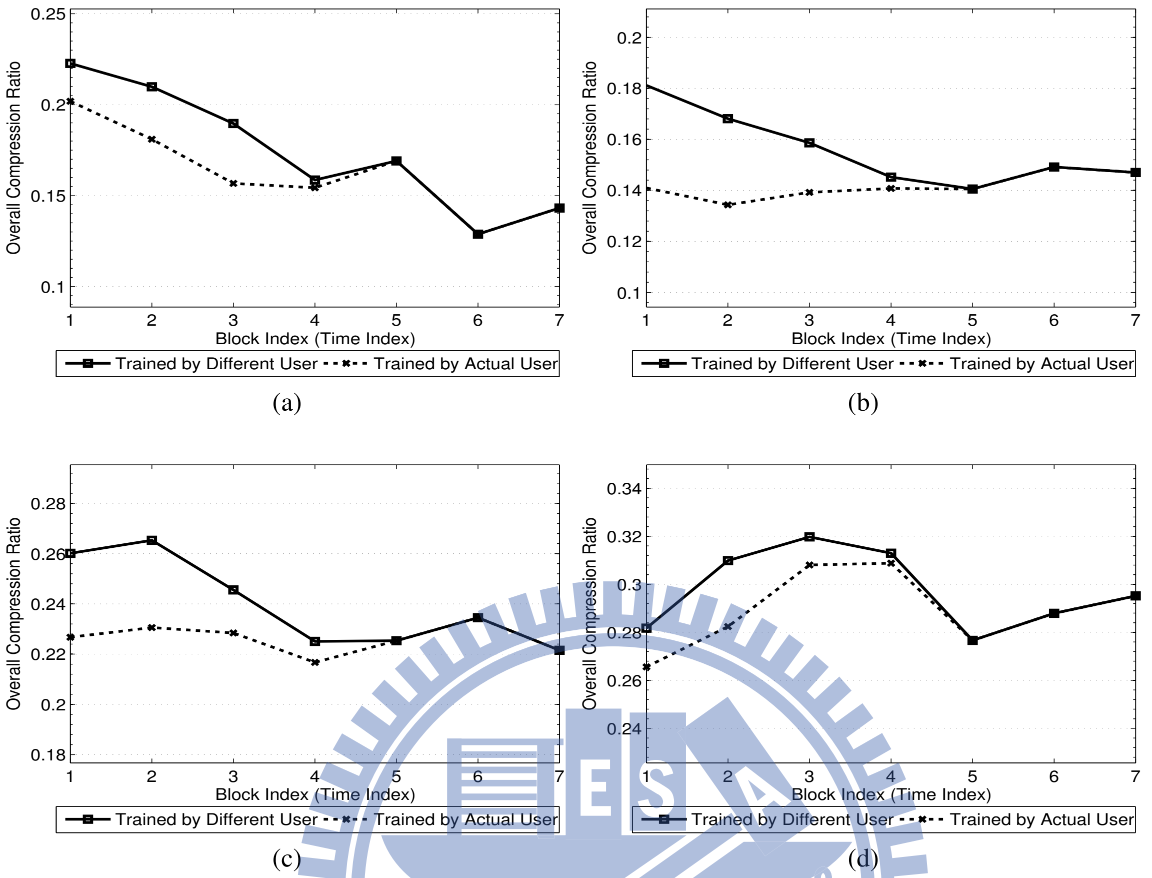


Figure 2.12: The effect of system retraining on overall compression ratio. (a) exercise Fig. 2.3(a). (b) exercise Fig. 2.3(b). (c) exercise Fig. 2.3(c). (d) exercise Fig. 2.3(d).

user's data, so there is no deviation between the training parameters and the actual behaviors. We can see that our data compression model can be accurately retrained within a short delay.

2.6 Conclusions

In this chapter, we have introduced a novel data compression framework to attack the multi-sensor, multi-node compression problem in BISNs. We exploit the temporal and spatial correlations among sensing data from multiple sensor nodes by differential coding and overhearing among sensor nodes via linear regression. We also formulated the transmission-ordering problem to determine nodes' overhearing sequence to better exploit spatial correlations and thus minimize the total amount of transmissions. We proposed two scheduling methods, which differ in the longest sequence of nodes that a node needs to overhear. Our experiments on capturing Pilates exercises demonstrated the effectiveness of our approach. Our approach only requires a sensor node to overhear at most one other node. One future direction that deserves further research is to exploit the correlation between a node and a set of other nodes.

Chapter 3

Exploiting Multi-Spatial Correlations of Motion Data for Data Compression

3.1 Introduction

In Chapter 2, we have shown how to compress motion data by considering *single-spatial correlations* of motion data, allowing a node to overhear at most $\kappa = 1$ node's transmission and to exploit the correlation with its own data for data compression. It is based on the following two observations. First, since a BISN is likely fully connected, overhearing among nodes is possible. This opens a space to relieve collision and contention among transmissions. Second, since human body motions have inherent rhymes, it typically results in strong spatial correlations among measurements of different nodes. This opens another space to compress sensing data when overhearing is possible.

In this chapter, we extend Chapter 2 by considering *multi-spatial correlations* of motion data in a BISN. That is, we further allow a node to overhear $\kappa > 1$ nodes' transmissions. By overhearing multiple nodes, the spatial correlations of motion data can be better exploited, resulting in better compression ratios. However, such an extension is nontrivial. To solve the overhearing dependencies among nodes, we formulate the problem as one of finding an optimal DAG that minimizes the total amount of transmissions in a BISN. While a minimum-cost tree for $\kappa = 1$ can be found in polynomial time, we show that finding a minimum-cost DAG is NP-hard even for $\kappa = 2$. We then propose an efficient heuristic. Experimental results with real BISN motion data are presented, which show more than 50% improvement over Chapter 2.

The rest of chapter is organized as follows. Section 3.2 formulates the multi-spatial data compression problem, shows its NP-hardness, and proposes a heuristic. Experimental results are shown in Section 3.3. Section 3.4 concludes this chapter.

3.2 Multi-Spatial Data Compression

3.2.1 Problem Definition

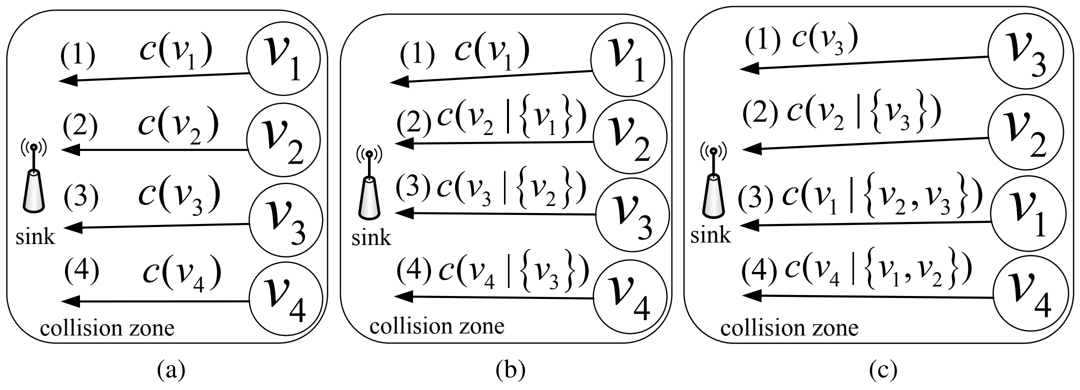
We consider a BISN with n sensor nodes $v_1, v_2, \ldots, v_n$ deployed on a human body. Each node has an inertial sensor. Via wireless links, nodes periodically report their readings to a sink in a round-by-round fashion. Since a human body is relatively small, nodes are assumed to be mostly fully connected and thus overhearing is possible.

In order to reduce transmission, a node may conduct data compression. Let us consider three models. In Fig. 3.1(a), each node simply compresses its own data individually. In Fig. 3.1(b), a node may compress its data based on its spatial correlation with another node. In Fig. 3.1(c), the compression may be based on correlations with multiple nodes. Chapter 2 adopts the model in Fig. 3.1(b) and uses an offline phase to learn the spatial correlations among sensors. A case study in Pilates exercises shows significant compression effect using the model in Fig. 3.1(b).

In this work, we extend Chapter 2 by considering the *multi-spatial data compression (κ-MDC) problem*. A node v_i may compress its data based on overhearing the data of a set S_i of nodes. The average size of the compressed data sent by v_i is denoted by $c(v_i \mid S_i)$. Note that when no spatial correlation is applied, $S_i = \emptyset$. Our goal is to find a proper set S_i for each v_i such that $\sum_{i=1}^{n} c(v_i \mid S_i)$ is minimized. To be more practical, we enforce $|S_i| \leq \kappa$ for a constant κ.

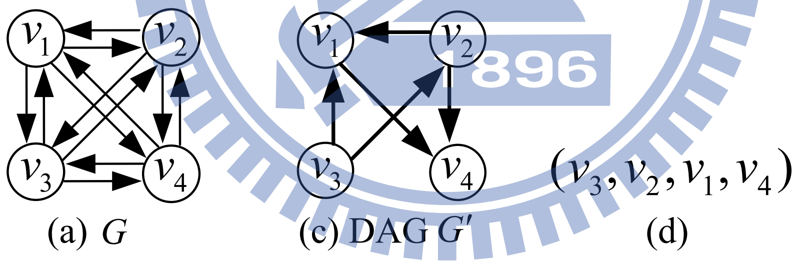
The problem can be formulated as follows. We model the links of the BISN by a complete directed graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$. The operation of letting v_i overhear $S_i \subseteq V \setminus \{v_i\}$ is formulated as finding a subset $E_i \subset E$ such that $\langle v_j, v_i \rangle \in E_i$ iff $v_j \in S_i$. So there are $\binom{n-1}{0} + \binom{n-1}{1} + \cdots + \binom{n-1}{\kappa}$ combinations for the selection of S_i. The corresponding transmission cost is $c(v_i \mid S_i)$. It follows that any DAG $G' = (V, E' \subset E)$ found from G represents an acyclic overhearing relation among nodes and has a total cost of $\sum_{i=1}^{n} c(v_i \mid S_i)$. The concept is shown in Fig. 3.2. Our goal is to find the minimum-cost DAG. From G', the transmission sequence can be determined by a topological sort on G' [46].

Note that $c(v_i \mid S_i)$ may be represented by conditional entropies and should be learned in advance from offline training. Also note that in practice v_i may fail to overhear some nodes in S_i. In this case, v_i may either transmit only $c(v_i)$ or suboptimal $c(v_i \mid S_i')$ such that the transmissions of nodes in S_i' have been overheard by v_i. That is, the DAG G' represents the best strategy, but each node can dynamically change its compression strategy at a higher cost when



(1) $c(v_1)$
(2) $c(v_2)$
(3) $c(v_3)$
(4) $c(v_4)$
sink
collision zone
(a)
(1) $c(v_1)$
(2) $c(v_2 \mid \{v_1\})$
(3) $c(v_3 \mid \{v_2\})$
(4) $c(v_4 \mid \{v_3\})$
sink
collision zone
(b)
(1) $c(v_3)$
(2) $c(v_2 \mid \{v_3\})$
(3) $c(v_1 \mid \{v_2, v_3\})$
(4) $c(v_4 \mid \{v_1, v_2\})$
sink
collision zone
(c)

Figure 3.1: Data compression models. $c(v_i \mid S_i)$ means the compressed size of v_i's data after v_i overhears S_i. (a) individual. (b) single-spatial. (c) multi-spatial.



$\lvert S_i \rvert = 0$	$\lvert S_i \rvert = 1$	$\lvert S_i \rvert = 2$
$c(v_1)$	$c(v_1 \mid \{v_j\}), j = 2,3,4$	$c(v_1 \mid \{v_j, v_k\}), j,k = 2,3,4$
$c(v_2)$	$c(v_2 \mid \{v_j\}), j = 1,3,4$	$c(v_2 \mid \{v_j, v_k\}), j,k = 1,3,4$
$c(v_3)$	$c(v_3 \mid \{v_j\}), j = 1,2,4$	$c(v_3 \mid \{v_j, v_k\}), j,k = 1,2,4$
$c(v_4)$	$c(v_4 \mid \{v_j\}), j = 1,2,3$	$c(v_4 \mid \{v_j, v_k\}), j,k = 1,2,3$

(b)

Figure 3.2: An example of 2-MDC: (a) G, (b) all dependency relations, (c) a possible DAG G', and (d) a topological sort of G'.

needed.

3.2.2 Complexity Analysis

In [47], it is shown that finding a minimum-cost DAG for $\kappa = 1$ can be solved efficiently in polynomial time. Below, we show that even for the simplest generalization of $\kappa = 2$, finding an optimal DAG is NP-hard. The proof is based on a variant of the *feedback arc set (FAS)* problem, a well-known NP-hard problem [48].

Definition 3.2.1. Given a digraph $G_f = (V_f, E_f)$, the FAS problem is to find a feedback arc set $E'_f \subseteq E_f$ with the minimal size $|E'_f|$ such that $G'_f = (V_f, E_f \setminus E'_f)$ is a DAG.

Lemma 3.2.2. *The FAS problem remains NP-hard for in-degree-2 digraphs, where each node has at most two incoming arcs.*

Proof: It is known that the FAS problem remains NP-hard for digraphs in which the total in-degree and out-degree of each node is no more than three [49]. By noting that in such digraphs, no node with three incoming arcs can be in a cycle, the lemma follows directly. ■

Theorem 3.2.3. *The κ-MDC problem is NP-hard for $\kappa \geq 2$.*

Proof: We will show the case of $\kappa = 2$, and other cases can be shown similarly. The proof is based on reduction from Lemma 3.2.2. Given an in-degree-2 digraph $G = (V, E)$ of the FAS problem, we construct an instance of the κ-MDC problem $G' = (V, E' \supseteq E)$, a complete digraph whose cost $c(\cdot \,|\, \cdot)$ is described later. Note that both problems ask for finding DAGs, the former by removing arcs from G and the latter by selecting arcs from G'. Hence, the cost of removing an arc from G is translated to the additional cost when this arc is not selected from G'.

Below, we show how to assign $c(v_i \,|\, S)$ for any v_i and S. For each v_i that has no incoming arc in G, we let $c(v_i) = 0$ and all other $c(v_i \,|\, S) = \infty$ (or a value large enough). For each v_i that has only one incoming arc $\langle v_j, v_i \rangle$ in G, we let $c(v_i) = 1$, $c(v_i \,|\, \{v_j\}) = 0$, and all other $c(v_i \,|\, S) = \infty$. Note that an additional cost is incurred when $\langle v_j, v_i \rangle$ is not selected from G'. Similarly, for each v_i that has two incoming arcs $\langle v_j, v_i \rangle$ and $\langle v_k, v_i \rangle$ in G, we let $c(v_i) = 2$, $c(v_i \,|\, \{v_j\}) = c(v_i \,|\, \{v_k\}) = 1$, $c(v_i \,|\, \{v_j, v_k\}) = 0$, and all other $c(v_i \,|\, S) = \infty$. The additional cost depends on how many of $\langle v_j, v_i \rangle$ and $\langle v_k, v_i \rangle$ are not selected from G'.

By this construction, the total cost of a DAG found from G' is the number of arcs to be removed from G. Hence, once the minimum-cost DAG is found from G', the optimum FAS of G is found. ■

3.2.3 A Greedy Cycle Breaker Heuristic

To the best of our knowledge, there is no known approximation for the κ-MDC problem. We thus propose a heuristic called *greedy cycle breaker (GCB)*, which is inspired by the FAS problem. Given a complete digraph $G = (V, E)$, it works by selecting an optimal subgraph and then breaking cycles greedily.

1. For each v_i, find the set $S_i^* \subseteq V \setminus \{v_i\}$ such that $|S_i^*| \leq \kappa$ and $c(v_i \,|\, S_i^*)$ is minimized. Then, for each $v_j \in S_i^*$, add arc $\langle v_j, v_i \rangle$ to set E'. This forms an optimal subgraph $G' = (V, E')$.

2. If the above G' contains no cycle, then G' is the minimum-cost DAG and the algorithm terminates. (There are many polynomial time algorithms to identify cycles in a graph.) Otherwise, continue to step 3.

3. Find a set $U \subseteq V$ such that for each $v_i \in U$, $S_i^* \subseteq U$ and the overhearing relations among U is a DAG. Note that U may be empty. Below, we will add nodes to U iteratively.

4. For each $v_i \in V \setminus U$, compute a suboptimal set $U_i^* \subseteq U$ such that the additional cost $c(v_i \,|\, U_i^*) - c(v_i \,|\, S_i^*)$ is minimal. Among all these nodes, choose the one, say v_i, such that the additional cost is minimal. Then, we add v_i to U and set its overhearing set to be U_i^*.

5. Repeat step 4 until $U = V$.

In our experience, such a greedy method is quite effective for motion data. Experimental results in Section 3.3 show that GCB is quite close to the optimum in most cases.

3.3 Experimental Results

We conduct experiments on real BISN motion data to verify the performance of GCB as well as the effects brought by multi-spatial correlations.

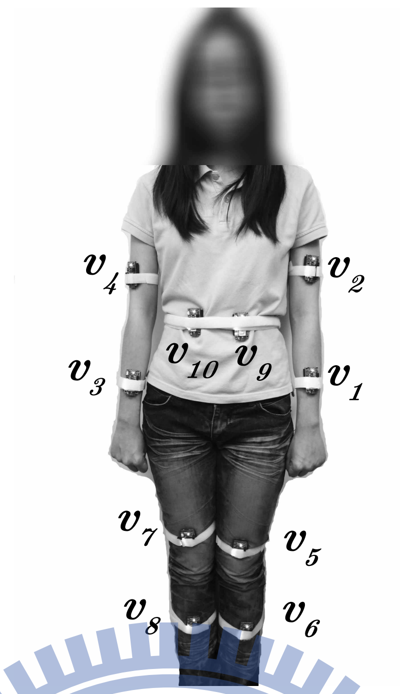


Figure 3.3: The sensor locations in our experiments.

3.3.1 Environment

We place $n = 10$ nodes on a human body to capture motions. Two nodes are placed on each limb, and the rest two are placed around the waist (refer to Fig. 3.3). Each node is a Jennic JN5139 single-chip microprocessor [14] with a 16-MIPS RISC CPU, 192KB ROM, 96KB RAM, a ZigBee-compliant module, and an triaxial accelerometer [50]. Each piece of raw data in each axis is 16 bits, and the sampling rate is set to 20 Hz. For ease of presentation, we use only one axis. The results can be straightforwardly extended to multiple axes.

We collect three sets of motion data: (a) walking, (b) walking upstairs, and (c) running. For each motion, we compare three data compression methods.

- **1-MDC.** This is the single-spatial compression method in Chapter 2.
- **2-GCB.** This is our GCB method with $\kappa = 2$.
- **2-OPT.** This is an exhaustive search for the optimum results of 2-MDC.

3.3.2 Compression under Ideal Channels

First, we consider ideal wireless channels as follows. We assume that nodes can overhear all intended packets without loss. Toward this end, we collect all nodes' raw data first and use two-third of them for learning inter-node correlations and the rest for testing. From the learning

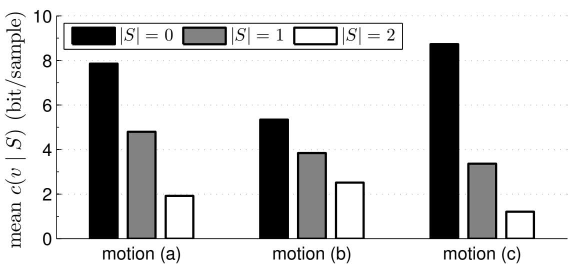


Figure 3.4: The distributions of $c(v \mid S)$ by $|S|$.

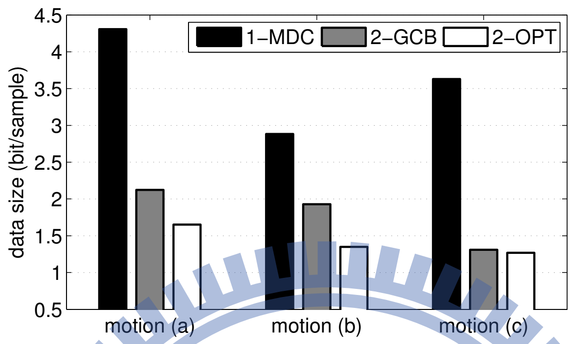


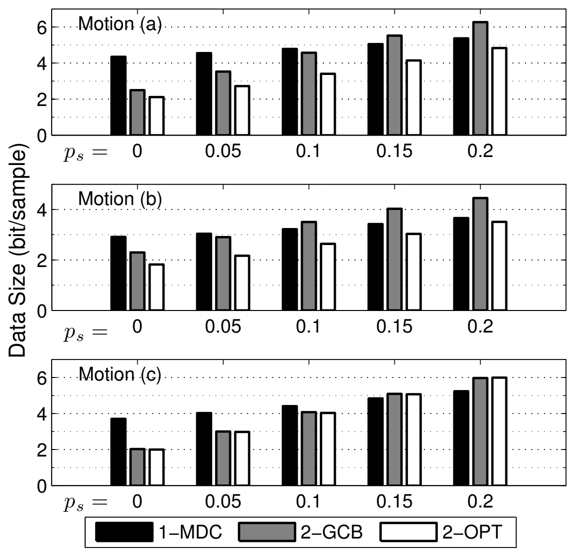
Figure 3.5: Comparison of compressed data sizes under ideal channels.

data set, we obtain the cost functions $c(v_i \mid S_i)$ through Huffman compression for all possible S_is. Its distribution is shown in Fig. 3.4. The results verify that multi-spatial correlations are worth exploiting.

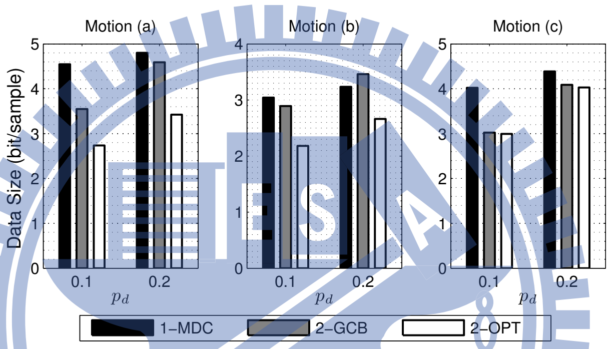
Based on these $c(v_i \mid S_i)$s, we then develop different compression schemes. Then, we use the testing data set and run a simulation program to make comparison. The results are shown in Fig. 3.5. For motion (a), each piece of compressed data without overhearing is about 8 bits (refer to Fig. 3.4). The 1-MDC and 2-GCB methods reduce the data sizes to 4.3 bits and 2.1 bits, respectively. The optimum size by 2-OPT is 1.7 bits. The trends for other motions are similar. This verifies the effectiveness of GCB.

3.3.3 Compression under Lossy Channels

From the same learning and testing data sets, we further simulate environments with lossy channels as follows. Let $p_{ij}(t)$ be the probability that a packet sent by v_i at time t may be lost at v_j. We consider both static and dynamic channels. For the former, we consider a fixed value p_s for all $p_{ij}(t)$s. For the latter, we let each $p_{ij}(t)$ be randomly drawn from $[0, p_d]$, where p_d is a constant. When a node v_i loses any intended packet, it simply compresses by itself (thus the cost is $c(v_i)$ bits). The results are shown in Fig. 3.6. When $p_s = 0$ or $p_d = 0$, it represents the ideal channels. Evidently, a higher p_s or p_d would impair our schemes since there is a less chance



(a)



(b)

Figure 3.6: Comparison of compressed sizes under lossy channels. (a) static channels. (b) dynamic channels.

for overhearing. As the loss probability increases, the portion of data that is compressed by spatial correlations also decreases (refer to Fig. 3.7 for static channels). One possible solution deserving investigation is to adaptively fall back to 1-MDC when the channels are found lossy.

3.4 Conclusions

In this chapter, we have demonstrated how to exploit multi-spatial correlations in motion data compression. We model the problem in a BISN and prove its relation with NP-hardness. We then propose a heuristic and verify its effectiveness through real experimental data under different channel models.

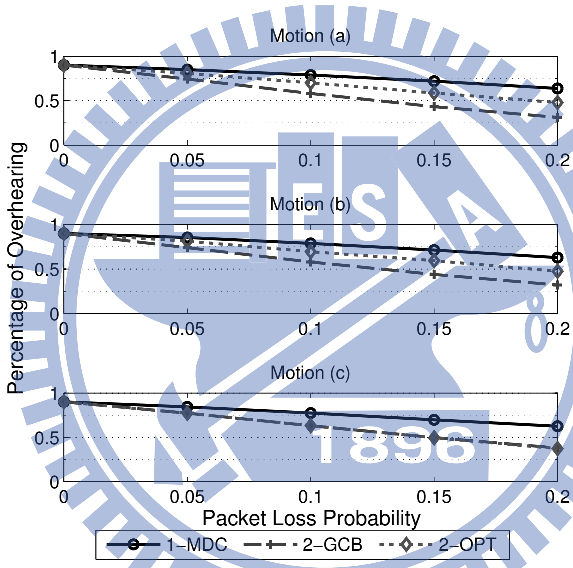


Figure 3.7: Percentage of overhearing under static lossy channels.

Chapter 4

Inertial Sensor Deployment Methods for Gravity Measurement

4.1 Introduction

A *body-area inertial sensor network (BISN)* consists of multiple accelerometers, magnetometers, and gyroscopes connected by wired/wireless links. An important usage of a BISN is to track human motions through these inertial sensors. Its applications include video games [51], robotic balancing [52], localization [53], sports training [6], medical care [47,54], and computer graphics [7].

Regarding a human body as multiple rigid parts connected by joints, many researchers consider a full-body motion as the combination of the motions of multiple rigid parts through the human body skeleton structure. Basically, the motion of each rigid part is measured by some inertial sensors deployed on a human body [55]. Many gesture applications have been developed [4, 54, 56].

One fundamental issue in BISNs is the *gravity measurement problem*. We want to measure the gravity vector with respect to a rigid part of a human body, no matter it is moving or not. A trivial case is when an accelerometer is mounted on a car driving on a flat plane as in Fig. 4.1(a). The gravity vector $\vec{g}$ with respect to the accelerometer is always the same. However, as shown in Fig. 4.1(b), for a human body, the orientation of each rigid part keeps on changing. Therefore, for accelerometers, determining the gravity vector of a rigid part is a fundamental issue to be solved before trying to track human motions.

To solve the gravity measurement problem, [52] considers a rigid part of a human body as being held by a joint which is at a constant velocity with respect to the Earth coordinate. Assuming multiple accelerometers deployed at known locations on the rigid part, it proposes

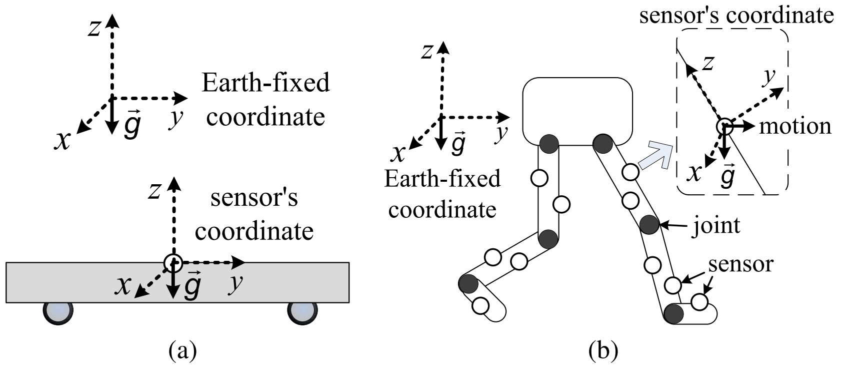


Figure 4.1: Gravity measurement in (a) a car and (b) a human body, where g is the gravity vector.

a data fusion method to calculate the gravity vector on the rigid part. However, it only partly addresses the error issue, which is common to sensor reading. Also, the locations of these sensors may highly impact the potential value of errors. In this paper, based on the model in [52], we further formulate the *deployment optimization problem* as one of finding the best locations of accelerometers on a rigid part that minimize the estimation error of the gravity vector. The problem is difficult even for simple geometries of rigid parts. We propose two heuristics, called *Metropolis-based* method and *largest-inter-distance-based (LID-based)* method. The former follows a probabilistic search principle that would eventually reach near-optimal solutions. The latter is based on a *"largest inter-distance (LID)" guideline* obtained from our observations, which states that sensors should be evenly distributed over the surface of the rigid part. We show that this guideline leads to an optimal solution for some special cases and near-optimal solutions in general. Experimental results show that both methods perform quite well, even for complicated geometries.

We remark that although the LID guideline, which tries to push accelerometers away from each other on one rigid part, has been widely adopted in other fields, its validity on our sensing model is not yet verified. Our analysis starts from the kinematical model of rigid parts and we demonstrate its effectiveness simulation- and experiment-wise.

Our results are applicable to any geometrical model illustrated in Fig. 4.2, where a rigid part is held by a joint that moves at a constant velocity v with respect to an Earth-fixed coordinate. The rigid part may rotate at any angular velocity ϕ about the joint. Both v and ϕ are unknown to our system. This allows us to apply it to many physical rehabilitation, robotic balancing, and gait analysis scenarios.

The rest of paper is organized as follows. Section 4.2 reviews related works. Section 4.3

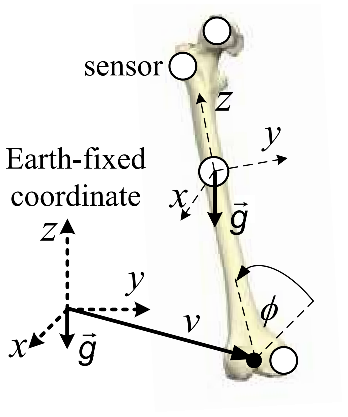


Figure 4.2: The geometrical model of our work, where v and ϕ are unknown to our system and g is to be calculated.

formulates the gravity measurement and the deployment optimization problems. Section 4.4 presents our heuristics. Performance studies are discussed in Section 4.5. Conclusions are drawn in Section 4.6.

4.2 Related Works

The deployment of sensors directly determines the network topology and influences communications. This issue has long been studied in wireless sensor networks. A comprehensive survey can be found in [57]. Many deployment algorithms have been proposed for isotropic sensors with unit-disc sensing coverage [58–61]. Directional sensors, such as cameras, have been studied in [62, 63]. The irregularity of sensing quality has been studied in [64]. However, none of these studies is suitable for accelerometers when they are deployed on a human body for motion sensing.

Comprehensive surveys on BISNs can be found in [1, 65], and surveys on human motion tracking can be found in [66, 67]. Traditionally, one accelerometer is deployed on one rigid part of a human body to measure its orientation, and the results from multiple parts are combined through a skeleton model to reconstruct full-body motions [53, 55, 68]. With additional motion constraints, [69–72] further avoid infeasible orientations. However, these approaches rely on empirical methods to eliminate such factors. To solve this issue, [52] assumes multiple accelerometers being deployed on one rigid part and proposes a kinematical method that optimally measures the gravity. However, how to optimize the locations of accelerometers so as to remove as much sensor reading errors as possible is left by as an open problem.

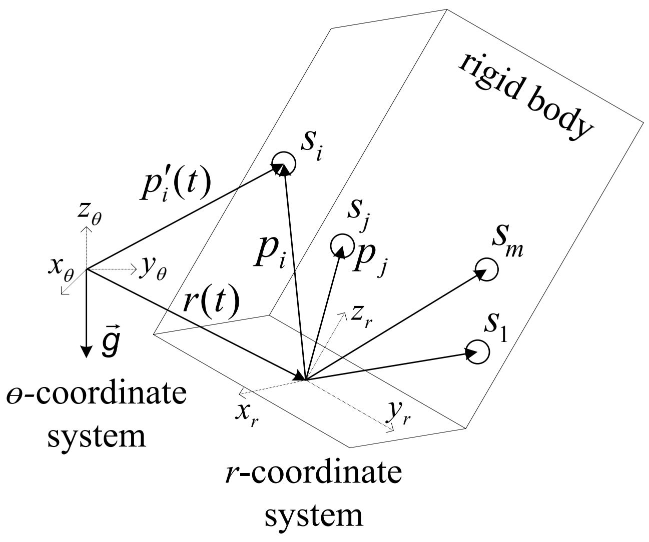


Figure 4.3: The kinematical model for one rigid body.

4.3 Gravity Measurement and Deployment Optimization Problems

We are interested in tracking human postures by deploying accelerometers on a human body. One fundamental issue in such scenarios is how to calculate the gravity, no matter when the body parts are moving or not [55]. A human body can be regarded as multiple movable *parts*, each being a *rigid body*, connected to another part by a *rotational joint* [73, 74]. Multiple accelerometers are placed on each rigid body for gravity measurement.

To formulate the *gravity measurement problem*, we consider one rigid body deployed with multiple sensors as in Fig. 4.3. We assume that there is an Earth-fixed coordinate system, representing views of a fixed observer, with $\theta = (0, 0, 0)$ as its origin and x_θ, y_θ, and z_θ as its axes. The gravity $\vec{g}$ with respect to this θ-coordinate is thus 1 g (or 9.8 m/sec^2) along the $-z_\theta$ axis. Let the joint of the rigid body be at location $r(t)$ with respect to the θ-coordinate at time t. For the rigid body, we assume that there is a fixed coordinate with respect to the joint with $r(t)$ as its origin and x_r, y_r, and z_r as its axes. Therefore, each point on the rigid body has a fixed coordinate with respect to the r-coordinate, no matter the rigid body moves or not. For any point at location p with respect to the r-coordinate, its location with respect to the θ-coordinate changes over time t and can be written as $p'(t) = r(t) + R(t)p$, where $R(t)$ is the 3×3 *rotation matrix* that translates (x_r, y_r, z_r) to $(x_\theta, y_\theta, z_\theta)$ [73].

Let s_1, s_2, $\ldots$, s_m be m accelerometers deployed on the rigid body and p_1, p_2, $\ldots$, p_m be their locations at the r-coordinate, respectively. Without loss of generality, we assume that these accelerometers are properly pre-calibrated in the sense that their x-, y-, and z-axes are aligned to the x_r, y_r, and z_r axes of the r-coordinate, respectively (otherwise, a rotation matrix from the r-coordinate to each sensor's coordinate would do the translation). This implies that the gravity being observed by the joint is the same as that being observed by any sensor.

Now, consider any sensor s_i, $i = 1, 2, \ldots, m$. Its location with respect to the θ-coordinate at time t is $p'_i(t) = r(t) + R(t)p_i$ (note that p_i is time-invariant). Taking the second derivative of $p'_i(t)$, we have its acceleration with respect to the θ-coordinate:

$$\ddot{p}'_i(t) = \ddot{r}(t) + \ddot{R}(t)p_i. \tag{4.1}$$

Since s_i is calibrated to the r-coordinate, its actual reading $a_i(t)$ should be $a_i(t) = R^T(t)\left(-\ddot{p}'_i(t) + \vec{g}\right) + n_i(t)$, where $R^T(t)$ is the inverse/transpose of $R(t)$ (the inverse of a rotation matrix equals its transpose) and $n_i(t)$ is the noise. Note that both $a_i(t)$ and $n_i(t)$ are 3×1 vectors with respect to the r-coordinate, and each element of $n_i(t)$ has a zero mean with a standard deviation σ_n. (For example, when the rigid body is at rest, $\ddot{p}'_i(t) = 0$ and $a_i(t) = R^T(t)g + n_i(t)$; when it falls freely, $a_i(t) = n_i(t)$.)

Below, assuming a fixed t, we will omit time information in our formulation. Plugging Eq. (4.1) into a_i, we have

$$\begin{aligned} a_i &= R^T\left(-\ddot{r} - \ddot{R}p_i + \vec{g}\right) + n_i \\ &= \left[R^T\left(-\ddot{r} + \vec{g}\right), \quad -R^T\ddot{R}\right]\begin{bmatrix}1\\ p_i\end{bmatrix} + n_i. \end{aligned}$$

Putting m equations for all s_i's together, we have the equality:

$$A = QP + N, \tag{4.2}$$

where

$$\begin{aligned} A &= \begin{bmatrix}a_1 & \ldots & a_m\end{bmatrix} \in \Re^{3\times m}, \\ Q &= \left[R^T\left(-\ddot{r} + g\right), \quad -R^T\ddot{R}\right] \in \Re^{3\times 4}, \\ P &= \begin{bmatrix}1 & \ldots & 1\\ p_1 & \ldots & p_m\end{bmatrix} \in \Re^{4\times m}, \\ N &= \begin{bmatrix}n_1 & \ldots & n_m\end{bmatrix} \in \Re^{3\times m}. \end{aligned}$$

Here, A and P are known and Q is to be determined. P is called the *deployment matrix* of sensors $s_1, s_2, \ldots, s_m$.

Assuming that the joint moves at a constant velocity with respect to θ, we can plug $\ddot{r} = 0$ in Eq. (4.2). Hence, Q's first column vector $R^T(-\ddot{r} + g) = R^T g$. By estimating Q, we can determine $R^T g$. Let $\hat{Q}$ be an estimation of Q. Following [52], a $\hat{Q}$ that makes $\hat{Q} - Q$ as small as possible can be found by $\hat{Q} = AP^+$, where P^+ is the Moore-Penrose pseudoinverse of P when $m > 4$ and $P^+ = P^{-1}$ (the inverse of P) when $m = 4$ [75]. This implies that to find $\hat{Q}$ we need at least four sensors. Since $\hat{Q} - Q$ is zero-mean, to measure how $\hat{Q}$ is close to Q, [52] defines an *error variance*, which solely depends on the deployment matrix P:

$$\sigma_e^2(P) = 3\sigma_n^2 \sum_{k=1}^{4} \frac{1}{\rho_k^2(P)}, \tag{4.3}$$

where $\rho_k(P)$ is the kth largest singular value of P. It is claimed that a smaller $\sigma_e^2(P)$ implies a more accurate $\hat{Q}$. Therefore, we formulate the *deployment optimization problem* as follows: given a rigid body and m accelerometers to be deployed on the surface of the rigid body, the goal is to find a deployment matrix P such that the error variance $\sigma_e^2(P)$ is minimized. Note that since the $3\sigma_n^2$ in Eq. (4.3) is a constant, we only need to focus on the summation part.

4.4 Optimization Heuristics

The above formulation has related the sensor deployment problem to one of finding a P that minimizes $\sigma_e^2(P)$. The problem is difficult even for simple geometries. Below, we present two heuristics, called *Metropolis-based* and *largest-inter-distance-based (LID-based)* methods. The former is based on probabilistic optimization, and the latter is based on some observations and guidelines.

4.4.1 Metropolis-Based Method

We present a Metropolis-based deployment method that can be applied to rigid bodies of arbitrary shapes. It follows a probabilistic search to get rid of local optimal solutions. It, however, has longer search time. This is acceptable for our sensor deployment problem since once a good solution is found, it can be used repeatedly.

In this method, we partition the surface of the rigid body into mesh-like grid points. We say that two deployment matrices P_i and P_j are *neighbors* if they differ by exactly one sensor's location and this sensor's locations in P_i and P_j are neighboring grids. We define the eight nearest grid points of a grid point as the latter's neighbors. Therefore, each P_i has up to $8m$ neighbors. We denote by $B(P_i)$ the set of P_i's neighbors. The algorithm works as follows:

1. Select any arbitrary deployment matrix P_i.
2. From P_i, we choose one of its neighbors, say P_j, as the next deployment with probability q_{ij} (discussed below). Note that $\sum_{\forall j} q_{ij} = 1$.
3. Repeat step 2 for a predefined number of times, and output the best deployment (with the smallest $\sigma_e^2(P)$) along the above search path.

We design the transition probability q_{ij} according to the Metropolis' theorem [76]. The resulting q_{ij}s should ensure: if we run the above search process long enough, each deployment P should be visited by a mathematically stationary distribution $\pi(P)$ such that

$$\pi(P) = \frac{(1/\sigma_e^2(P))^\lambda}{C}, \tag{4.4}$$

where the parameter $\lambda > 3$ and $C = \sum_P (1/\sigma_e^2(P))^\lambda$. Note that C normalizes the distribution, and a higher λ ensures that a better deployment will be allocated a higher visiting probability. By Metropolis' theorem, this transition probability should be defined as

$$q_{ij} = \begin{cases} \frac{1}{8m}\min\{1, \frac{\pi(P_j)}{\pi(P_i)}\} & \text{if } P_j \in B(P_i), \\ 0 & \text{if } P_j \notin B(P_i), \\ 1 - \sum_{P_j \neq P_i} q_{ij} & \text{if } P_j = P_i. \end{cases} \tag{4.5}$$

Note that the actual value of C is immaterial in Eq. (4.5).

We argue that, though simple, this algorithm is highly probable to obtain a near-optimal solution. This is because Eq. (4.4) is devised similar to a (continuous) Pareto distribution with a density function $f(x) = C'/x^\lambda$ defined over $a \leq x$, where C' and a are constants [77]. The Pareto distribution has a mean of $(\lambda - 1)a/(\lambda - 2)$, which is very close to the minimal value a for a large λ. Through this similarity, we can expect the mean $\sum_P \sigma_e^2(P)\pi(P)$ to be very close to the minimum σ_e^2. Note that the variance of a Pareto distribution is also very tight for a large λ. Hence, once the search reaches the stationary distribution, the output deployment is highly probable to be a near-optimal solution.

4.4.2 Largest-Inter-Distance-Based Method

While the Metropolis-based method takes a long time to search near-optimal solutions, this method is based on some observations and a guideline, making it both efficient and effective in most cases. To minimize the term $\sum_{k=1}^4 1/\rho_k^2(P)$ in Eq. (4.3), we propose the following "largest inter-distance (LID)" guideline: *Sensors should be evenly distributed over the surface*

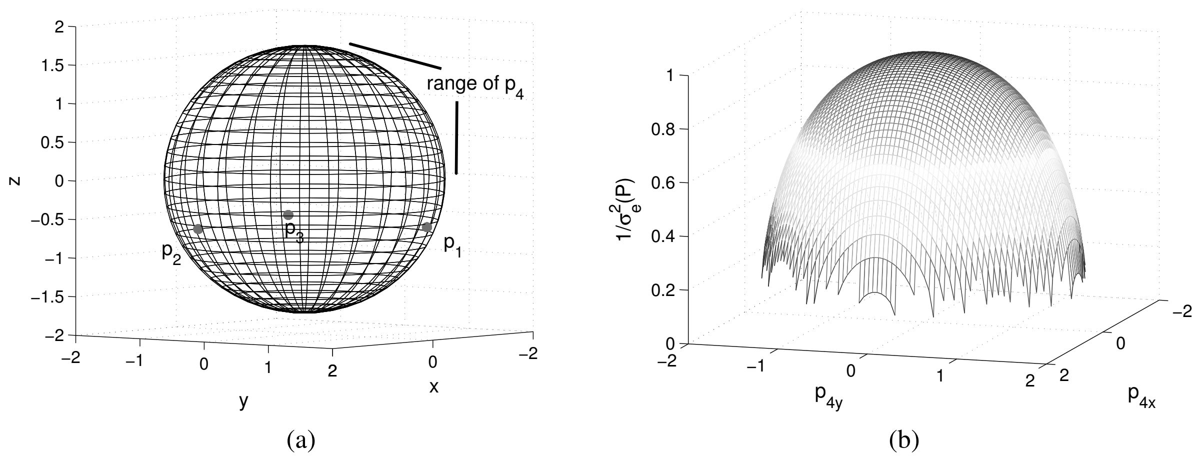


Figure 4.4: (a) a sphere rigid body attached by three fixed sensors and one moving sensor and (b) the reciprocals $1/\sigma_e^2(P)$ by moving point p_4.

area such that the distance between each pair of them is maximized. The guideline is drawn from observations on some special cases where optimal deployments can be found. Below, we discuss these special cases and then present the LID-based method.

To show the optimality of a deployment, we present the following lemma.

Lemma 4.4.1. *The error variance $\sigma_e^2(P)$ of a $4 \times m$ deployment matrix $P = (P_{ij})$ is bounded by*

$$\sigma_e^2(P) \geq 16/|P|^2, \tag{4.6}$$

where $|P| = \sqrt{\sum_{i,j} P_{ij}^2}$. The equality holds iff $\rho_1(P) = \rho_2(P) = \rho_3(P) = \rho_4(P)$.

Proof: The harmonic mean of $\rho_k^2(P)$, $k = 1 \ldots 4$, can be expressed by

$$\frac{4}{1/\rho_1^2(P) + 1/\rho_2^2(P) + 1/\rho_3^2(P) + 1/\rho_4^2(P)} = \frac{4}{\sigma_e^2(P)}.$$

By properties of the Frobenius matrix norm [78], we have $\rho_1^2(P) + \rho_2^2(P) + \rho_3^2(P) + \rho_4^2(P) = |P|^2$, so the arithmetic mean of $\rho_k^2(P)$s is $|P|^2/4$. The lemma is a direct consequence of the fact that the arithmetic mean is always larger than the harmonic mean. Eq. (4.6) holds when $\rho_i^2(P) = \rho_j^2(P)$ for all i and j [79], or simply $\rho_i(P) = \rho_j(P)$ because $\rho_i(P) \geq 0$ for all i. ■

Lemma 4.4.1 also suggests that when the joint of a rigid body is at its geometric center, a near-optimal deployment may be obtained according to the LID guideline. To see this, consider a spherical rigid body with its joint at its center. Since each point on the sphere is equidistant to the joint, the $|P|$ has a fixed value. In Fig. 4.4(a), we deploy three fixed sensors at $p_1 = (1/\sqrt{3} - 1, 1 + 1/\sqrt{3}, -\sqrt{3}/3)$, $p_2 = (1 + 1/\sqrt{3}, 1/\sqrt{3} - 1, -\sqrt{3}/3)$, and

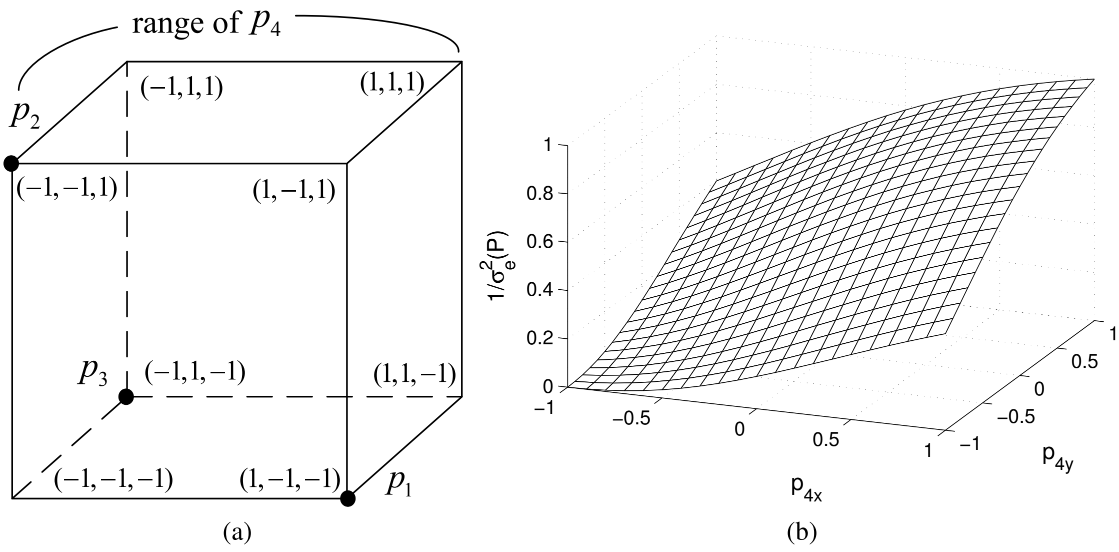


Figure 4.5: (a) a cube rigid body attached by three fixed sensors and one moving sensor and (b) the reciprocals $1/\sigma_e^2(P)$ by moving point p_4.

$p_3 = (-2/\sqrt{3}, -2/\sqrt{3}, -\sqrt{3}/3)$, and move the fourth sensor p_4 around on the upper part of the sphere. Let the coordinate of p_4 be (p_{4x}, p_{4y}, p_{4z}). Fig. 4.4(b) shows the reciprocal $1/\sigma_e^2(P)$ with respect to (p_{4x}, p_{4y}). Evidently, a larger $1/\sigma_e^2(P)$, or a smaller $\sigma_e^2(P)$, can be obtained by moving p_4 farther away from other sensors. The maximal value of $1/\sigma_e^2(P)$ appears at $p_4 = (0, 0, \sqrt{3})$. Under this configuration, we actually have $\rho_1(P) = \rho_2(P) = \rho_3(P) = \rho_4(P) = 2$. Since $|P|$ is fixed for all Ps, the optimality directly follows from Lemma 4.4.1. Basically, following the LID guideline to move four sensors around a sphere will result in forming a regular tetrahedron, whose faces are regular triangles. It can be shown that such deployments are optimal (see the proof in Section 4.7.1). We summarize the results with a theorem.

Theorem 4.4.2. *Given four sensors to be deployed on a sphere rigid body whose joint is at its geometric center, a regular tetrahedron deployment gives the minimum* $\sigma_e^2(P)$.

The LID guideline can also be applied to other geometries. In Fig. 4.5(a), we consider four sensors to be deployed on a cube whose joint is at its geometric center. We fix three sensors at $p_1 = (1, -1, -1)$, $p_2 = (-1, -1, 1)$, and $p_3 = (-1, 1, -1)$ and move $p_4 = (p_{4x}, p_{4y}, 1)$ around the top surface where $-1 \le p_{4x} \le 1$ and $-1 \le p_{4y} \le 1$. Fig. 4.5(b) shows the reciprocal $1/\sigma_e^2(P)$ with respect to (p_{4x}, p_{4y}). As can be seen, when p_1, p_2, p_3, and p_4 follows the LID guideline and results in forming a regular tetrahedron, the deployment is optimal (see the proof in Section 4.7.2).

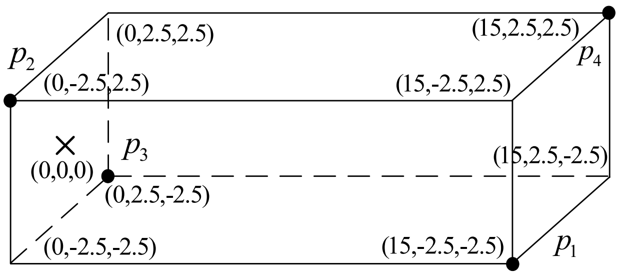


Figure 4.6: A rectangular box attached by four sensors with its joint at $\times$.

Theorem 4.4.3. *Given four sensors to be deployed on a cube rigid body whose joint is at its geometric center, a regular tetrahedron deployment at the vertices gives the minimum $\sigma_e^2(P)$.*

For the case of a rectangular box with the joint placed at its geometric center, we have experimented many cases with four sensors and found that a tetrahedron-like deployment (such as the one in Fig. 4.6) always gives the minimum $\sigma_e^2(P)$. However, we can not derive a formal proof of its optimality yet. On the other hand, when the joint is not at its geometric center, we do find a counterexample as follows. In the case of Fig. 4.6, where the joint is at the left surface (marked by a $\times$), the $\sigma_e^2(P)$ is 0.5844. With the same joint position, when $p_1 = (0, 2.5, 1.6)$, $p_2 = (0, -0.7, -2.46)$, $p_3 = (1.56, -2.5, 2.48)$, and $p_4 = (12.84, 2.5, -2.48)$, we find a smaller $\sigma_e^2(P) = 0.5347$. Hence, the LID guideline is able to give optimal solutions in some special cases and near-optimal solutions in general.

Now, we present an algorithm to implement the LID guideline for rigid bodies of arbitrary shapes. It distributes sensor nodes by introducing virtual forces that mimics the electronic particles distribution principle: *the nearer the particles, the stronger the repelling forces between them.* This algorithm works as follows.

1. Initially, randomly place sensors $s_1, s_2, \ldots, s_m$ on the surface of the rigid body. Let p_1, $p_2, \ldots, p_m$ be their locations, respectively.

2. For each pair of s_i and s_j, the repelling force contributed by s_j on s_i is defined by a 3×1 vector

$$f_{ji} = \left(\frac{1}{\|p_i - p_j\|} + \beta \right) \frac{p_i - p_j}{\|p_i - p_j\|},$$

where β is a constant speed to expedite the iterative process. In the exceptional case of $p_i = p_j$, we set f_{ji} to a unit vector of any direction. The total force contributed by all other sensors on s_i is $f_i = \sum_{j \neq i} f_{ji}$.

3. For each sensor s_i, we let it move a displacement vector of kf_i, where k is a scaling constant. So the new location of s_i is $p_i + kf_i$. Since this location may not be on the surface of the rigid body, a projection operation is needed (we will discuss this below).

4. Let $p_i^{(old)}$ and $p_i^{(new)}$ be the previous and next locations of s_i, respectively. We test whether $\sum_{i=1}^{m} \|p_i^{(new)} - p_i^{(old)}\| \leq \sigma_{th}$, where σ_{th} is a threshold value, or the allowed number of iterations is reached. If so, we terminate this algorithm; otherwise, go back to step 2.

The above algorithm follows the virtual-force discipline [80] to maximize the inter-distances among sensors. Note that since the final locations depend on the initial locations, to avoid falling into a local minimum, we may repeat the algorithm several times with randomized initial locations.

Below, we explain the projection operation in step 3. To be computationally feasible, we approximate a surface by polygons as usually done in computer graphics. Fig. 4.7 shows an example. Sensor s_i is at location p_i and is pushed by a displacement kf_i. The projection is conducted in a polygon-by-polygon manner. First, $p_i + kf_i$ is projected to polygon 1. Since the projected point is beyond the range of polygon 1, we regard that a partial displacement d_1 out of kf_i has been consumed and s_i has been moved to p'_i, which is at the edge of polygon 1 (see Fig. 4.7). A remaining displacement of $kf_i - d_1$ needs to be applied to p'_i. Similarly, $p'_i + (kf_i - d_1)$ is projected to polygon 2. Since the projected point is also beyond the range of polygon 2, we regard that a partial displacement d_2 out of $kf_i - d_1$ has been consumed and s_i has been moved to p''_i, which is at the edge of polygon 2 (see Fig. 4.7). A remaining displacement of $kf_i - d_1 - d_2$ needs to be applied to p''_i. Finally, $p''_i + (kf_i - d_1 - d_2)$ is projected to polygon 3. Since the projected point is within the range of polygon 3, the final location of s_i is p'''_i on polygon 3. Note that the above iterative process is indeed supported by many 3D game engines, such as Unity [17]. Fig. 4.8 shows how we configure an arm by polygons in Unity and run our LID-based method.

4.5 Simulation and Experimental Results

We have conducted simulations and real experiments to verify our schemes.

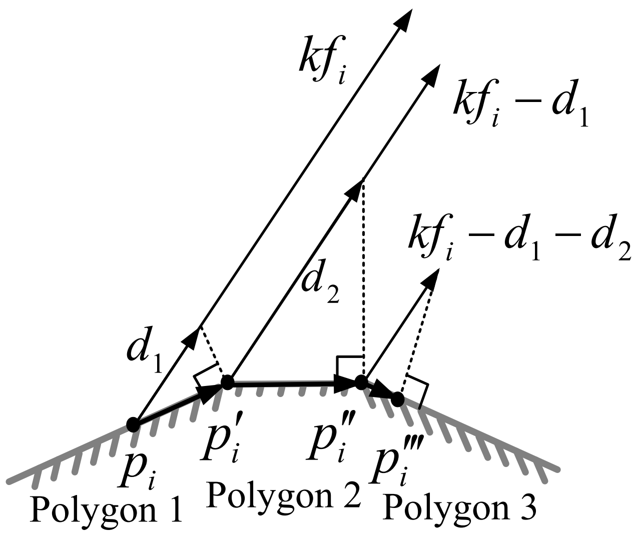


Figure 4.7: An example of iterative projection on three consecutive polygons.

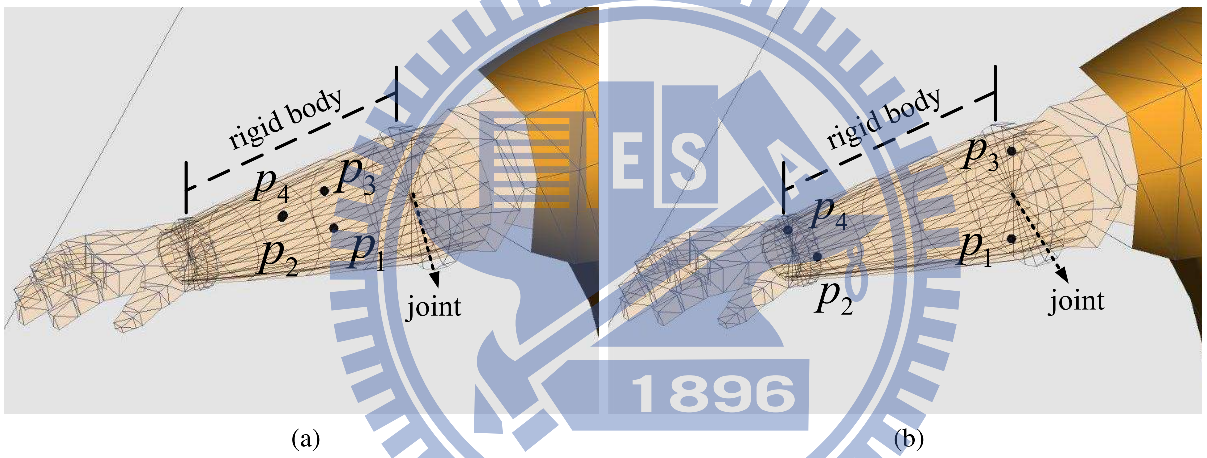


Figure 4.8: An example of running the LID-based method by Unity. (a) initial deployment. (b) final deployment.

4.5.1 Simulation Setup

We consider five rigid body geometries in Fig. 4.9. For each geometry, we simulate its joint at three possible positions, one at its geometric center (shown by $\circ$), one at the center of a face (shown by $\times$), and one at a corner (shown by $\triangle$). The combinations of geometries and joints give a total of 15 rigid bodies to be evaluated.

For each rigid body, we compare the following three deployment methods.

- **Random Deployment.** This method distributes sensors at random positions. It represents previous approaches, which place sensors without any optimization [52].

- **Metropolis-Based Deployment.** This is our Metropolis-based deployment method. In

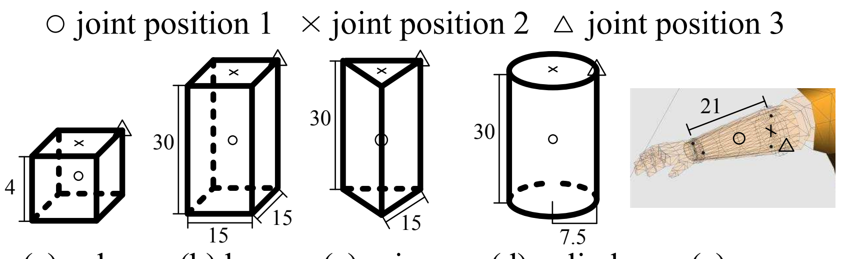


Figure 4.9: Rigid body geometries, each with its joint placed at one of three possible positions.

our simulations, we give each run 10,000 iterations and set the grid sizes of geometries (a) to (e) in Fig. 4.9 to 0.1, 0.4, 0.4, 0.2, and 0.15, respectively.

- **LID-Based Deployment.** This is our LID-based deployment method. In our simulations, we set $\beta = 0.5$, $k = 1$, and $\sigma_{th} = 0.0001$. The maximum iteration number of a run is set to 2,000.

To simulate the estimation of gravity vectors, for each deployment matrix P, we follow Eq. (4.2) to calculate $\hat{Q}$. Without any loss of generality, R in Eq. (4.2) is set to the identity matrix, so the real gravity vector $R^T\vec{g}$ (which is hidden in Q) becomes $\vec{g}$. The $\ddot{R}$ is arbitrarily set to the identity matrix because it is filtered out by our estimation scheme. The noise level σ_n is set to 50 mg (note that the magnitude of $\vec{g}$ is 1000 mg). The estimated gravity vector $\hat{g}$ is the first column of $\hat{Q}$. Since $\vec{g}$ has a constant magnitude, we compare the angle α between $\hat{g}$ and $\vec{g}$.

4.5.2 Comparison of Accuracy and Execution Time

First, we consider using $m = 4$ sensors only. For each rigid body, each method is run 200 times (each with its own randomized initial sensor positions) to constitute an averaged simulation result. We compare methods by α, σ_e^2, and the execution time.

The results of α and σ_e^2 are shown in Fig. 4.10 and Fig. 4.11, respectively. The unit of σ_e^2 is ignored since it is merely used for comparison. Comparing Fig. 4.10 and Fig. 4.11, we see that optimizing α can be done through σ_e^2. The random deployment method always gives large α and σ_e^2. Our proposed methods obtain significant improvements and also show tight confidence intervals. The LID-based method outperforms the Metropolis-based method in most cases, except a few cases in Fig. 4.11(c) for the triangular prism. One possible reason for the exceptions may be that it does not consider the joint position. Since both proposed methods give very small σ_e^2, we can believe that their results are near-optimal.

Fig. 4.12 compares the execution time measured on a PC with an Intel i7 CPU. We report the cases when joints are at the geometric centers. The random method terminates almost immediately after execution, but it gives poor results. The Metropolis-based method trades notably longer execution time for better results. Evidently, the LID-based method is both effective and efficient.

4.5.3 Effects of the Number of Sensors

The previous results only consider $m = 4$. Here we vary m to observe the effect. For each value of m, we repeat the previous simulations for our proposed methods. The random deployment method is ignored (for better presentation). Fig. 4.13 and Fig. 4.14 show the mean α and the mean σ_e^2, respectively, where we denote by "J1" the joint position 1 (refer to Fig. 4.9), by "J2" the joint position 2, and so on. The results show that deploying more sensors does reduce the errors. The LID-based method is still preferred. Note that in Fig. 4.13(c), for the joint position 2, the LID-based method can outperform the Metropolis-based method by using $m = 5$ sensors. Fig. 4.15 compares the execution time. The LID-based method is not sensitive to the value of m, whereas the Metropolis-based method has linearly increasing execution time as m increases.

4.5.4 Real Gravity Measurement

In addition to simulations, we also conducted real experiments to measure the gravity vector. We constructed a rectangular box, fixed the box by a hinge joint, deployed eight lab-made accelerometers on its surface, and then rotated the box to measure the gravity. Fig. 4.16 shows our experiment scenario and how we rotated the box around the beam. Here, the joint is set to be $(0, 0, 0)$ of the r-coordinate, whose three axes are denoted by x_r, y_r, and z_r, respectively. Note that the x_r axis is along the beam. We consider two motions, one consisting of a slow rotation between "up", "down", and "up" states and the other consisting of repeated fast rotations between "up" and "down" states. The sampling rate was set to 10 Hz. The sensing data was stored at each sensor node and reported to a sink in a reliable way. Then, we computed offline by a PC the estimated gravity vector $\hat{g}(t)$ with respect to the r-coordinate.

Although the real gravity vector $R^T\vec{g}(t)$ is desired for comparison, it was unavailable in our experiments. Therefore, we verify the accuracy of $\hat{g}(t)$ by the appearance of its waveform, which should correspond to how we rotated the box. Let $\alpha_x(t)$ be the angle between $\hat{g}(t)$ and x_r, $\alpha_y(t)$ be between $\hat{g}(t)$ and y_r, and $\alpha_z(t)$ be between $\hat{g}(t)$ and z_r. Fig. 4.17(a) shows the results of the slow rotation. We can see that all angles are measured quite accurately. The $\alpha_x(t)$ is nearly constant, reflecting that fact that the rotation was about the x_r-axis. The values of $\alpha_y(t)$ and $\alpha_z(t)$ also match the sequence of “up”, “down”, and “up” states. Note that the curves cannot be perfectly smooth because the rotation was manually conducted. Fig. 4.17(b) shows the results of the fast rotations. The $\alpha_x(t)$ is slightly fluctuating since the beam was unexpectedly shaken during the fast rotations. However, $\alpha_y(t)$ and $\alpha_z(t)$ still match the repeated rotations between “up” and “down” states. The experimental results show that our schemes also work well in real situations.

4.6 Conclusions

In this paper, we have investigated the gravity measurement problem for rigid bodies. We have modeled the sensor deployment problem by considering the locations of sensors on one rigid body to minimize the measurement errors. The model is quite suitable for BISNs. Based on our formulation, we proposed two heuristics to find near-optimal deployments for arbitrary rigid bodies. One is based on a probabilistic optimization technique that eventually finds near-optimal solutions. The other is based on a “largest inter-distance” guideline, which leads to optimal solutions in some special cases. Experimental results showed that our methods are quite effective and significantly outperform previous approaches.

4.7 Appendices

4.7.1 Proof of Theorem 4.4.2

In Section 4.4.2, we show that Theorem 4.4.2 holds for a sphere of radius $\sqrt{3}$. Here, we show that Theorem 4.4.2 also holds for spheres of any radii. Specifically, let the position of a sensor s_i, $i = 1, \ldots, 4$, be $p_i = (p_{ix}, p_{iy}, p_{iz})$. Deploying s_i on a sphere of radius d implies that $p_{ix}^2 + p_{iy}^2 + p_{iz}^2 = d^2$. We show that when p_is form a regular tetrahedron, the deployment is optimal.

A deployment matrix P is a 4×4 matrix, where each column is a four dimensional vector. Although the geometrical interpretation of singular value decomposition (SVD) of P plays an important role in understanding Theorem 4.4.2, it is hard to show the SVD process for four dimensional vectors. Noting that the LID guideline can be applied under dimension reduction of P, instead of considering a 3D sphere and its four-dimensional P directly, we will first consider deploying $m = 3$ sensors on a 2D circle, analyze properties of its three-dimensional P (a 3×3 matrix) geometrically, and show that following the LID guideline, a regular triangle deployment is optimal. Then, the analysis will be extended to a four-dimensional P in a straightforward manner to complete the proof.

In the 2D reduction, three sensors are deployed on a circle of radius d. Each s_i's location $p_i = (p_{ix}, p_{iy})$ satisfies $p_{ix}^2 + p_{iy}^2 = d^2$. The deployment matrix becomes

$$P = \begin{bmatrix} 1 & 1 & 1 \\ p_{1x} & p_{2x} & p_{3x} \\ p_{1y} & p_{2y} & p_{3y} \end{bmatrix}. \tag{4.7}$$

Similarly, $|P|$ has a fixed value. Below, we apply Lemma 4.4.1 to Eq. (4.7) to show that when p_is form a regular triangle (the counterpart of a regular tetrahedron), the deployment is optimal.

Each column of P in Eq. (4.7) is a vector that points to a circle on the $x = 1$ plane (refer to Fig. 4.18(b)). It turns out that when we consider a circle of any radius d, the equalities $\rho_1(P) = \rho_2(P) = \rho_3(P)$ in Lemma 4.4.1 may not exist. To overcome such difficulties, we extend P in Eq. (4.7) to

$$\tilde{P} = \begin{bmatrix} \delta & \delta & \delta \\ p_{1x} & p_{2x} & p_{3x} \\ p_{1y} & p_{2y} & p_{3y} \end{bmatrix},$$

where $\delta > 0$. Note that in the 3D space of $\tilde{P}$, the circle is translated from the $x = 1$ plane to the $x = \delta$ plane.

Lemma 4.7.1. *The equalities $\rho_1(\tilde{P}) = \rho_2(\tilde{P}) = \rho_3(\tilde{P})$ hold if and only if $\|p_1 - p_2\| = \|p_2 - p_3\| = \|p_3 - p_1\|$ and $d = \sqrt{2}\delta$.*

Proof: Let the SVD of $\tilde{P}$ be $\tilde{P} = R_2 \Xi R_1$, where R_1 and R_2 are rotation matrices, and Ξ is the diagonal matrix whose diagonal entries are $\rho_1(\tilde{P})$, $\rho_2(\tilde{P})$, and $\rho_3(\tilde{P})$. As shown in Fig. 4.18, the three matrices are applied to the base vectors $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, and $e_3 = (0, 0, 1)$ to carry out the transformation. If $\rho_1(\tilde{P}) = \rho_2(\tilde{P}) = \rho_3(\tilde{P})$, we have $\tilde{P} = R_2 \Xi R_1 = \rho_1(\tilde{P}) R_2 R_1$. That is, after multiplying $\tilde{P}$, the base vectors remain orthogonal to each other and each e_i is scaled by $\rho_1(\tilde{P})$. At the $x = \delta$ plane, setting radius $d = \sqrt{2}\delta$ preserves the orthogonality, which also implies that p_1, p_2, and p_3 form a regular triangle. The "only if" part can be shown similarly. ■

In moving from the $x = \delta$ plane back to the $x = 1$ plane, the following property holds.

Lemma 4.7.2. *The product of singular values $\rho_1(P)\rho_2(P)\rho_3(P)$ equals $\rho_1(\tilde{P})\rho_2(\tilde{P})\rho_3(\tilde{P})/\delta$.*

Proof: By properties of SVD, the squares of singular values of a matrix P, namely $\rho_k^2(P)$s, are the eigenvalues of P^TP. Since the determinant of a square matrix equals the product of its eigenvalues, we have

$$\det(P^TP) = \rho_1^2(P)\rho_2^2(P)\rho_3^2(P).$$

We may decompose $\tilde{P}$ as

$$\tilde{P} = \begin{bmatrix} \delta & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} P = \Delta P.$$

By properties of determinants, we also have

$$\begin{aligned} \det(\tilde{P}^T\tilde{P}) &= \det\left[(\Delta P)^T(\Delta P)\right] \\ &= \det(\Delta P)\det(\Delta P) \\ &= \det(\Delta)^2\det(P^TP) \\ &= \delta^2\det(P^TP). \end{aligned}$$

Since singular values are larger than zero, we have the desired result. ■

Because the arithmetic mean $\sum_{k=1}^3 \rho_k^2(\tilde{P})/3$ is fixed for all deployments, from Lemma 4.7.1 and the power mean inequality [79], we know that an extended $\tilde{P}$ where sensors form a regular triangle has the largest geometric mean $\sqrt{3}\rho_1^2(\tilde{P})\rho_2^2(\tilde{P})\rho_3^2(\tilde{P})$ and the largest harmonic mean $3/\sum_{k=1}^3 \rho_k^{-2}(\tilde{P})$, and they are equal to the arithmetic mean. Hence, by Lemma 4.7.2, we also know that the original P has the largest geometric mean. Now that the arithmetic mean is fixed for all deployments, we can expect that P has the largest harmonic mean, supporting our claim.

We now extend the above discussion to the original 4×4 deployment matrix. We show that a regular tetrahedron, where every tuple of three points p_i, p_j, and p_k forms a regular triangle, is an optimal deployment on a sphere. The extension can be done in a straightforward manner. Lemma 4.7.1 and Lemma 4.7.2 are extended as follows.

Lemma 4.7.3. *The equalities $\rho_1(\tilde{P}) = \rho_2(\tilde{P}) = \rho_3(\tilde{P}) = \rho_4(\tilde{P})$ hold if and only if $d = \sqrt{3}\delta$ and p_1, p_2, p_3, and p_4 form a regular tetrahedron.*

Lemma 4.7.4. *The following equality holds for a deployment matrix P and its extension $\tilde{P}$: $\rho_1(P)\rho_2(P)\rho_3(P)\rho_4(P) = \rho_1(\tilde{P})\rho_2(\tilde{P})\rho_3(\tilde{P})\rho_4(\tilde{P})/\delta$.*

Again, since the arithmetic mean of $\rho_k^2(P)$s is fixed for all deployments, we can expect a tetrahedron, which has the largest geometric mean, to also have the largest harmonic mean. We show it by an example in Fig. 4.4. Since the harmonic mean is the reciprocal of error variance $\sigma_e(P)$, we conclude that a regular tetrahedron is optimal.

4.7.2 Proof of Theorem 4.4.3

First, we show that Theorem 4.4.3 holds for the cube in Fig. 4.5(a). The four sensor locations p_1, p_2, p_3, and p_4 in Fig. 4.5(a) form a regular tetrahedron, and the deployment matrix P has the maximal norm $|P|$. Because a cube can be circumscribed by an outer sphere, the regular tetrahedron in Fig. 4.5(a) is indeed a rotation of the regular tetrahedron in Fig. 4.4(a). From Theorem 4.4.2, we know that a regular tetrahedron deployment is optimal for a fixed $|P|$. Then, the optimality ensues from Lemma 4.4.1. Similarly, Theorem 4.4.3 also holds for cubes of other edge lengths.

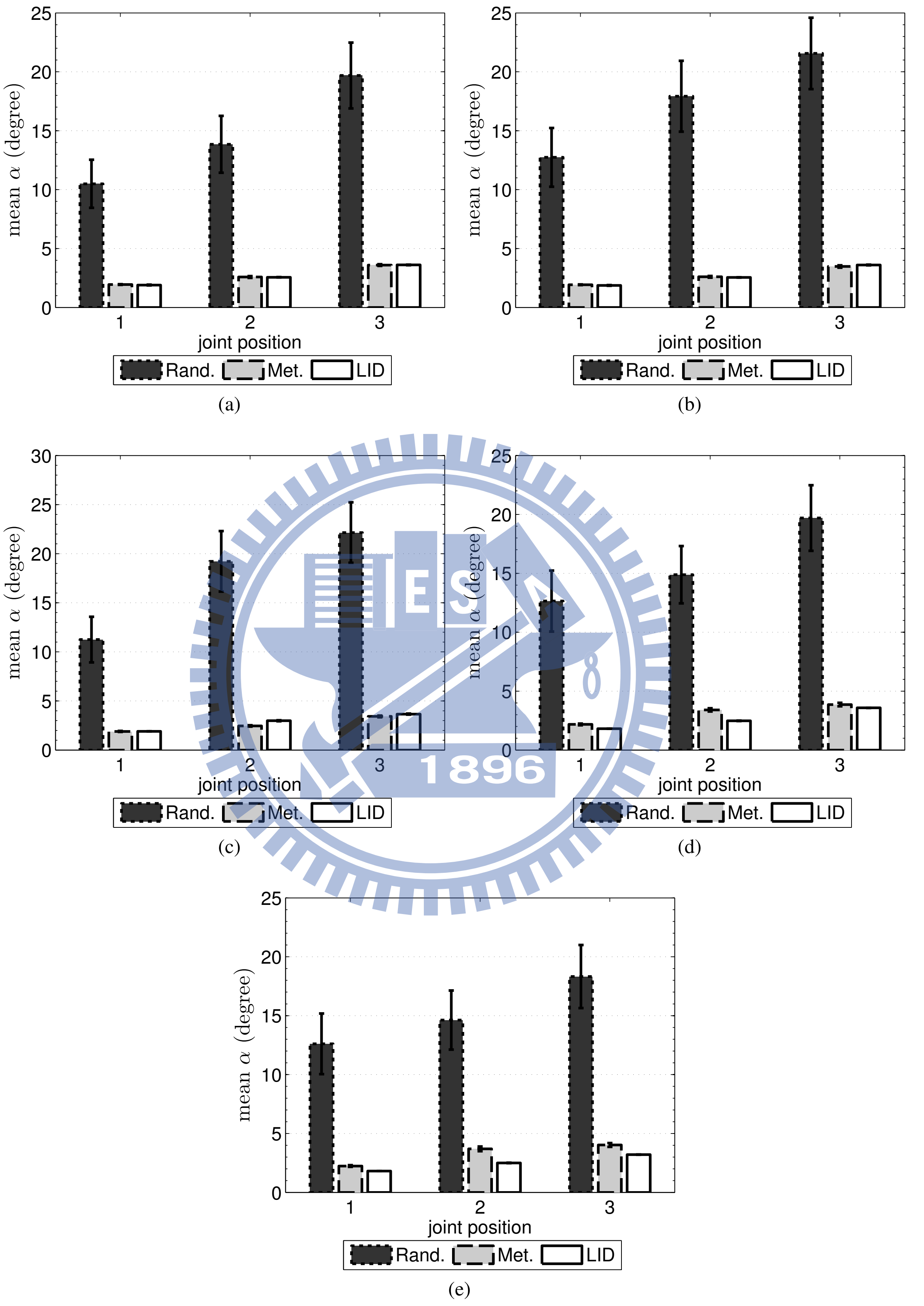
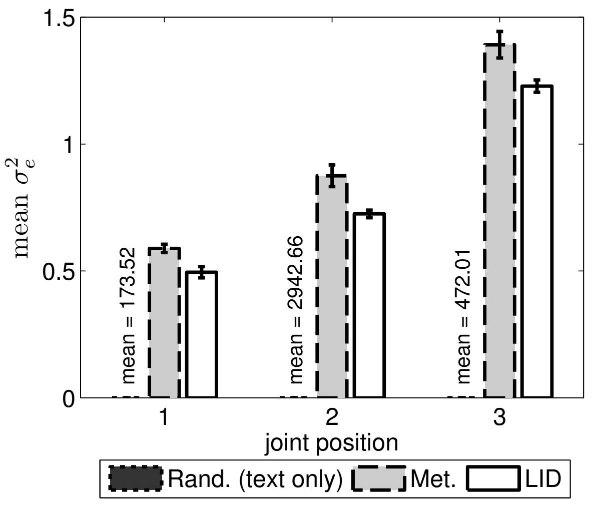
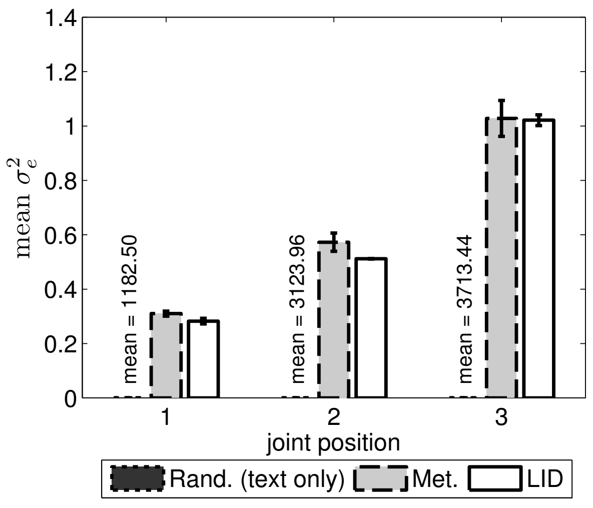


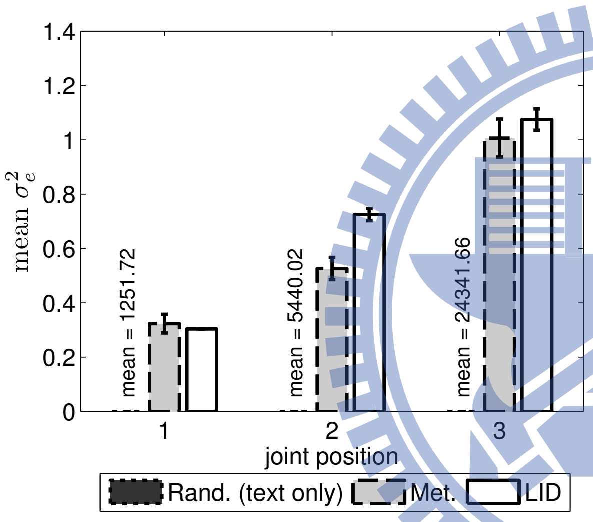
Figure 4.10: Comparison of α, with 95% confidence intervals shown on the bars. (a) geometry (a). (b) geometry (b). (c) geometry (c). (d) geometry (d). (e) geometry (e).



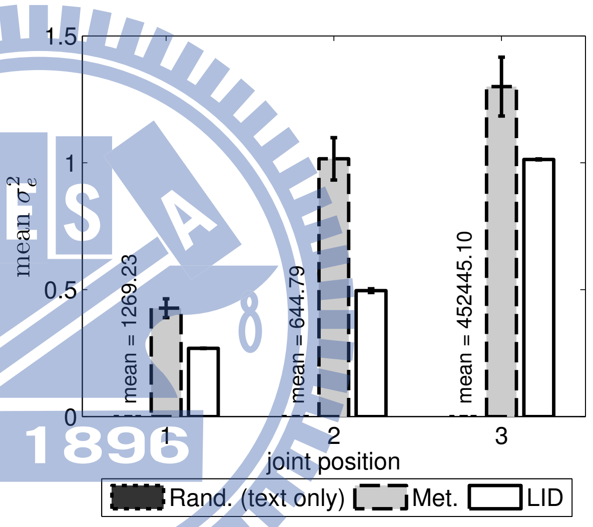
(a)



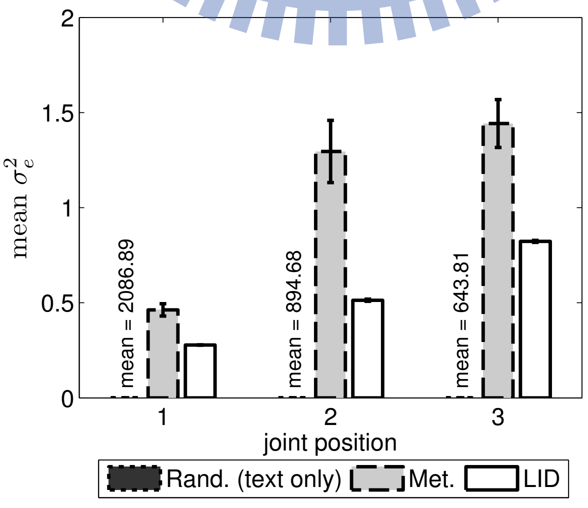
(b)



(c)



(d)



(e)

Figure 4.11: Comparison of σ_e^2, with 95% confidence intervals shown on the bars. (a) geometry (a). (b) geometry (b). (c) geometry (c). (d) geometry (d). (e) geometry (e).

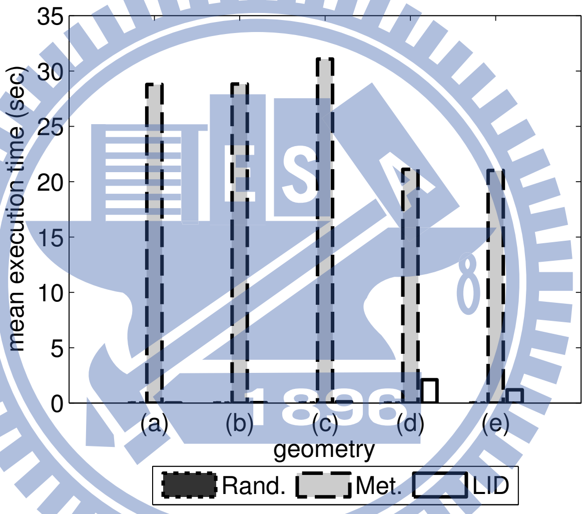


Figure 4.12: Comparison of execution time.

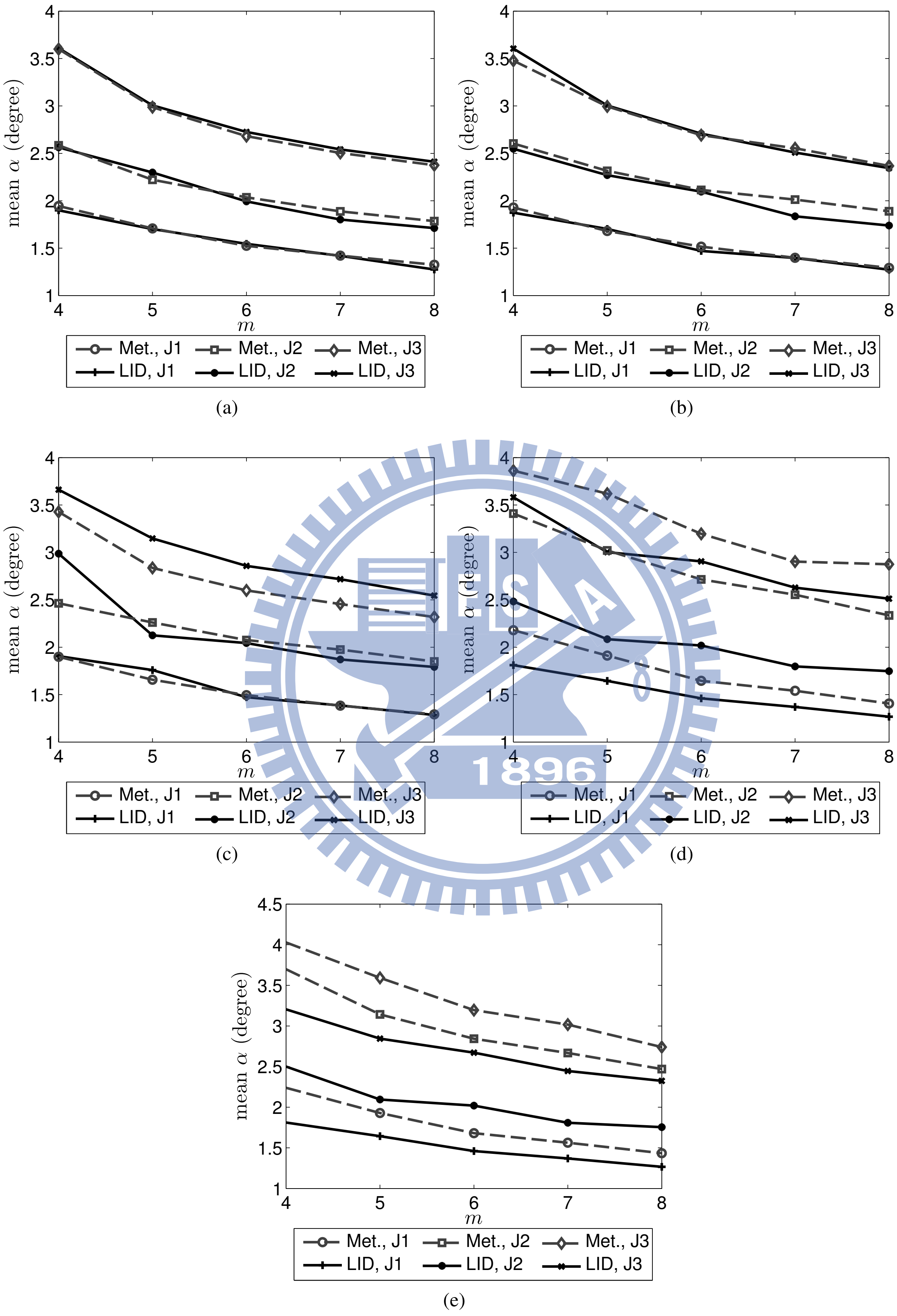


Figure 4.13: Effects of m on α. (a) geometry (a). (b) geometry (b). (c) geometry (c). (d) geometry (d). (e) geometry (e).

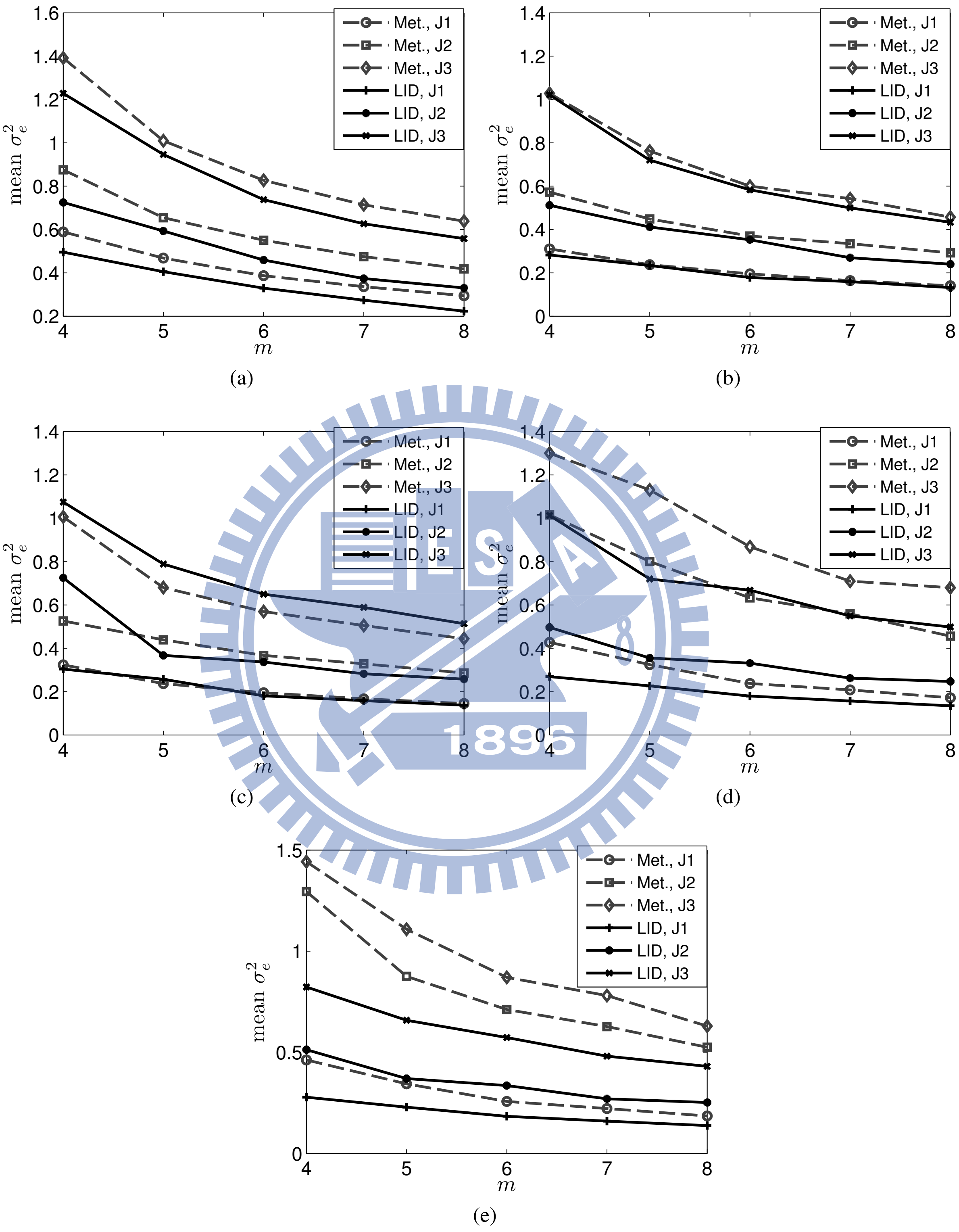


Figure 4.14: Effects of m on $\sigma_e^2(P)$. (a) geometry (a). (b) geometry (b). (c) geometry (c). (d) geometry (d). (e) geometry (e).

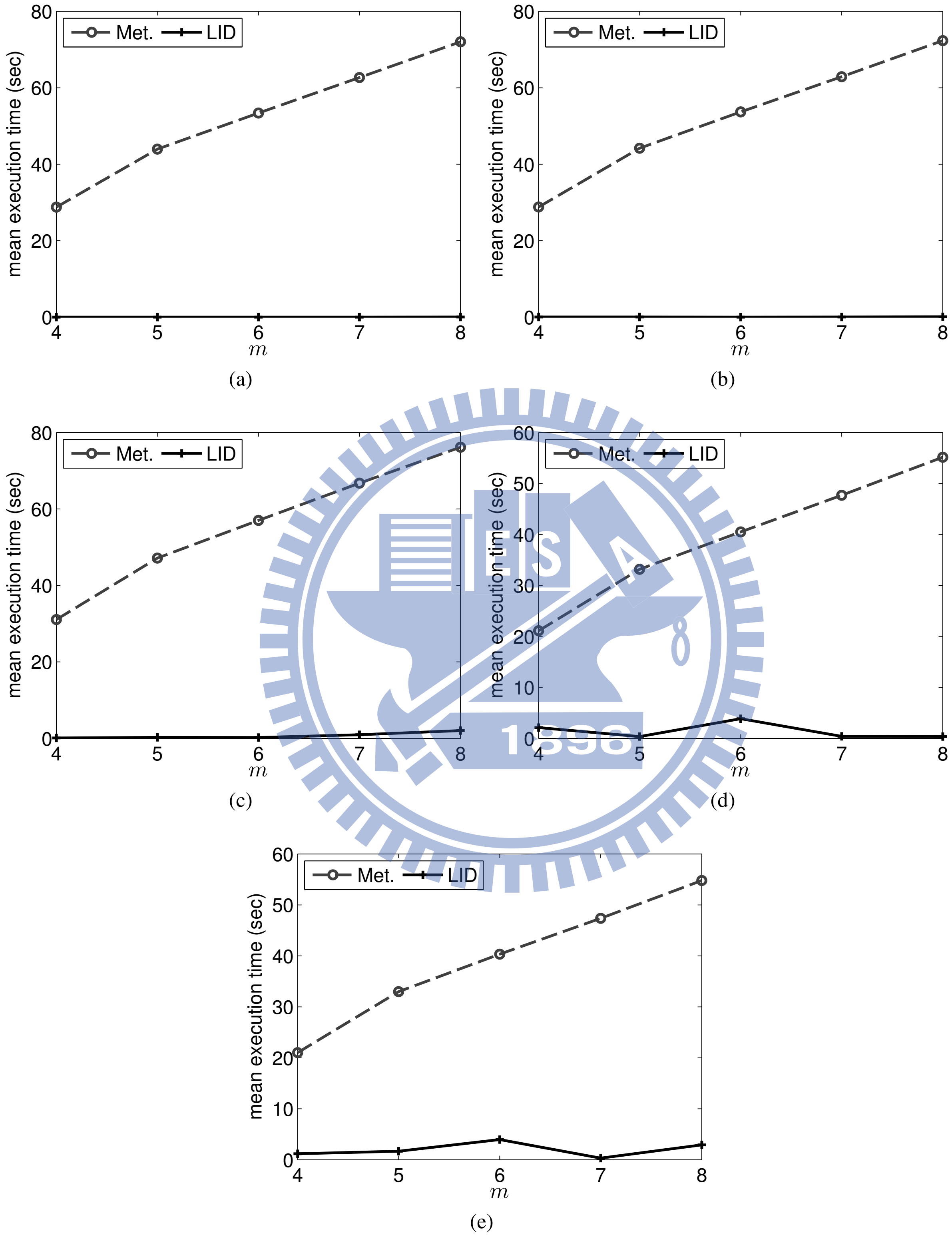
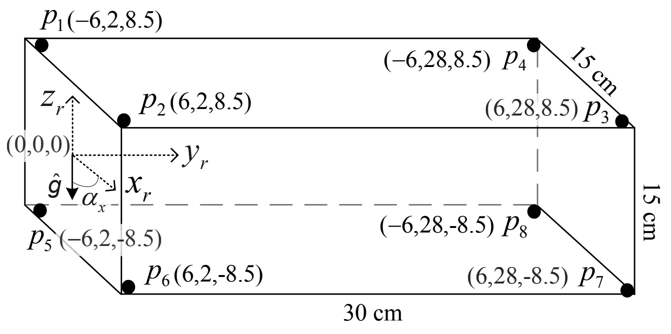
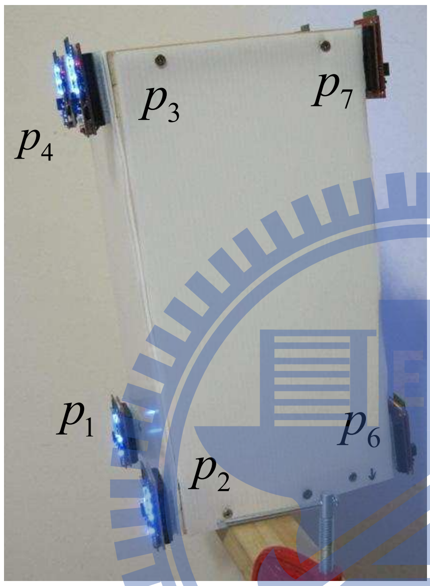


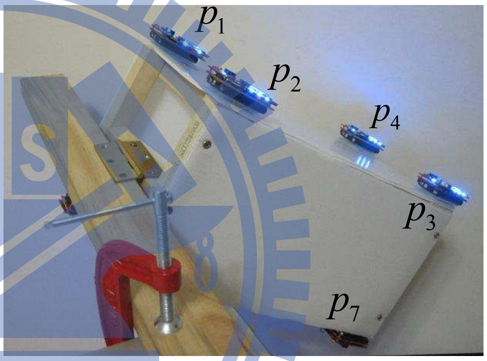
Figure 4.15: Effects of m on execution time. (a) geometry (a). (b) geometry (b). (c) geometry (c). (d) geometry (d). (e) geometry (e).



(a)

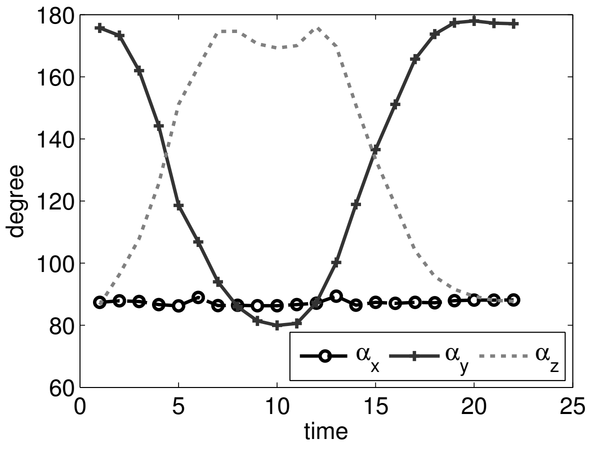


(b)

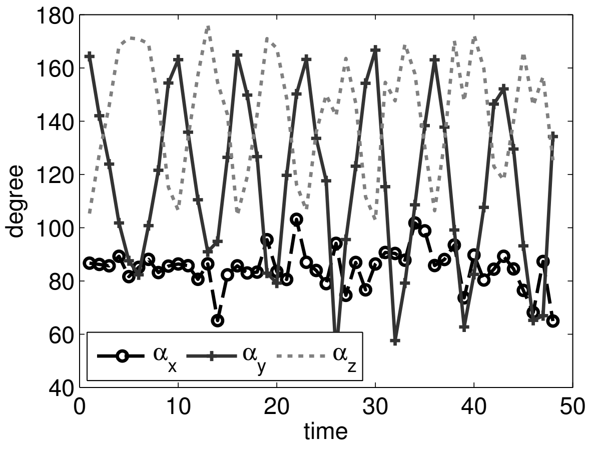


(c)

Figure 4.16: Our experimental scenario: (a) geometry model, (b) state "up", and (c) state "down".



(a)



(b)

Figure 4.17: The angles between $\hat{g}(t)$ and the axes. (a) a slow rotation. (b) fast rotations.

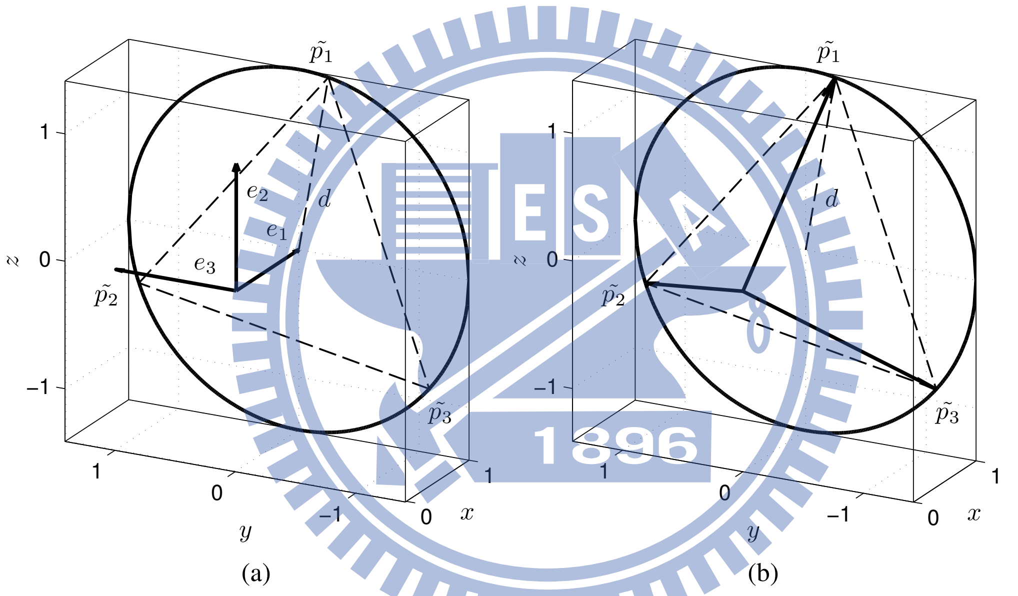


Figure 4.18: The SVD process for a regular triangle.

Chapter 5

Conclusions

In this dissertation, we have proposed a BISN-based human motion capturing platform that enhances both motion tracking accuracy and data collection efficiency. We have proposed a spatial-temporal data compression scheme to address the data collection issues in BISNs. Our data compression scheme exploits the possibility of overhearing among nodes in a BISN. We have extended our data compression scheme to allow exploiting multi-spatial correlations of motion data. For motion tracking accuracy, we have proposed inertial sensor deployment schemes to optimize sensors' collaboration on motion tracking. We have also demonstrated home rehabilitation systems and cyber-physical video games based on this platform.

Bibliography

[1] G.-Z. Yang, Ed., *Body Sensor Networks*. Springer, 2006.

[2] Nintendo, “Wii,” http://wii.com, 2011.

[3] S. Kang, J. Lee, H. Jang, Y. Lee, S. Park, and J. Song, “A scalable and energy-efficient context monitoring framework for mobile personal sensor networks,” *IEEE Trans. Mobile Comput.*, vol. 9, no. 5, pp. 686–702, 2010.

[4] J. Brutovsky and D. Novak, “Low-cost motivated rehabilitation system for post-operation exercises,” in *Proc. of Int’l Conf. on Engineering in Medicine and Biology Society (EMBS)*, 2006.

[5] J. M. Winters, Y. Wang, and J. M. Winters, “Wearable sensors and telerehabilitation: Integrating intelligent telerehabilitation assistants with a model for optimizing home therapy,” *IEEE Engineering in Medicine and Biology Magazine*, vol. 22, pp. 56–65, 2003.

[6] D. T. W. Fong, J. C. Y. Wong, A. H. F. Lam, R. H. W. Lam, and W. J. Li, “A wireless motion sensing system using ADXL MEMS accelerometers for sports science applications,” in *Proc. of World Congress on Intelligent Control and Automation*, 2004.

[7] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović, “Practical motion capture in everyday surroundings,” *ACM Trans. on Graphics*, vol. 26, no. 3, p. 35, 2007.

[8] D. Cavalcanti, R. Schmitt, and A. Soomro, “Performance analysis of 802.15.4 and 802.11e for body sensor network applications,” in *Proc. of Int’l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2007.

[9] B. de Silva, A. Natarajan, and M. Motani, “Inter-user interference in body sensor networks: Preliminary investigation and an infrastructure-based solution,” in *Proc. of Int’l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2009.

[10] Moteiv, “Tmote Sky Zigbee-based WSN platform,” http://www.moteiv.com, 2007.

[11] C. Park and P. H. Chou, “Eco: ultra-wearable and expandable wireless sensor platform,” in *Proc. of Int’l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2006.

[12] L. Cheng, S. Hailes, Z. Cheng, F.-Y. Fan, D. Hang, and Y. Yang, “Compressing inertial motion data in wireless sensing systems – an initial experiment,” in *Proc. of Int’l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2008.

[13] Panasonic, “Life Wall — 150-inch interactive TV wall,” http://www.panasonic.com/cesshow, 2008.

[14] Jennic, “JN5139 — low power, low cost wireless microcontroller,” http://www.jennic.com, 2011.

[15] OceanServer, “OS5000 family – triaxial accelerometer and electronic compass,” http://www.ocean-server.com, 2008.

[16] C. V. C. Bouten, K. T. M. Koekkoek, M. Verduin, R. Kodde, and J. D. Janssen, "A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity," *IEEE Trans. Biomed. Eng.*, vol. 44, no. 3, pp. 136–147, 1997.

[17] Unity, "Unity — 3D game engine," http://unity3d.com, 2011.

[18] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proc. of Int'l Conference on Very Large Data Bases (VLDB)*, 2004.

[19] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proc. of Int'l Conference on Data Engineering (ICDE)*, 2006.

[20] G. Zhou, J. Lu, C.-Y. Wan, M. Yarvis, and J. Stankovic, "BodyQoS: Adaptive and radio-agnostic QoS for body sensor networks," in *Proc. of IEEE INFOCOM*, 2008.

[21] D. Jea, W. Wu, W. J. Kaiser, and M. B. Srivastava, "Approximate data collection using resolution control based on context," in *Proc. of Int'l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2007.

[22] P. K. Baheti and H. Garudadri, "An ultra low power pulse oximeter sensor based on compressed sensing," in *Proc. of Int'l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2009.

[23] H. Ghasemzadeh, E. Guenterberg, and R. Jafari, "Energy-efficient information-driven coverage for physical movement monitoring in body sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 1, pp. 58–69, 2009.

[24] E. Guenterberg, H. Ghasemzadeh, and R. Jafari, "A distributed hidden markov model for fine-grained annotation in body sensor networks," in *Proc. of Int'l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2009.

[25] A. Scaglione and S. D. Servetto, "On the interdependence of routing and data compression in multi-hop sensor networks," in *Proc. of ACM Int'l Conference on Mobile Computing and Networking (MobiCom)*, 2002.

[26] T. ElBatt, "On the trade-offs of cooperative data compression in wireless sensor networks with spatial correlations," *IEEE Trans. Wireless Commun.*, vol. 8, no. 5, pp. 2546–2557, 2009.

[27] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 131–146, 2002.

[28] Y.-C. Wang, Y.-Y. Hsieh, and Y.-C. Tseng, "Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications," *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 827–838, June 2009.

[29] A. Silberstein, G. Puggioni, A. Gelfand, K. Munagala, and J. Yang, "Suppression and failures in sensor networks: A bayesian approach," in *Proc. of Int'l Conference on Very Large Data Bases (VLDB)*, 2007.

[30] R. Rydeard, A. Leger, and D. Smith, "Pilates-based therapeutic exercise: Effect on subjects with nonspecific chronic low back pain and functional disability: A randomized controlled trial," *Journal of Orthopaedic & Sports Physical Therapy (JOSPT)*, vol. 36, no. 7, pp. 472–484, 2006.

[31] I. Garci'a, S. de Barros, and M. Saldanha, "Isokinetic evaluation of the musculature involved in trunk flexion and extension: Pilates method effect," *Revista Brasileira de Medicina do Esporte*, vol. 10, no. 6, pp. 491–493, 2004.

[32] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Prentice-Hall, 1989.

[33] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 4th ed. Wiley-Interscience, New York, 2006.

[34] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, New York, 1991.

[35] Y. J. Chu and T. H. Liu, "On the shortest arborescence of a directed graph," *Science Sinica*, vol. 14, pp. 1396–1400, 1965.

[36] J. Edmonds, "Optimum branchings," *J. Research of the National Bureau of Standards*, vol. 71B, pp. 233–240, 1967.

[37] J. Naor and B. Schieber, "Improved approximations for shallow-light spanning trees," in *Proc. of IEEE Symposium on Foundations of Computer Science (SFCS)*, 1997.

[38] T. T. H. Binh, R. I. McKay, N. X. Hoai, and N. D. Nghia, "New heuristic and hybrid genetic algorithm for solving the bounded diameter minimum spanning tree problem," in *Proc. of ACM conference on Genetic and evolutionary computation (GECCO)*, 2009.

[39] M. Keally, G. Zhou, and G. Xing, "Watchdog: Confident event detection in heterogeneous sensor networks," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2010.

[40] D. E. Knuth, "Dynamic Huffman coding," *J. Algorithms*, vol. 6, no. 2, pp. 163–180, 1985.

[41] H.-A. Pham, V.-H. Bui, and A.-V. Dinh-Duc, "An adaptive, memory-efficient and fast algorithm for Huffman decoding and its implementation," in *Proc. of Int'l Conference on Interaction Sciences (ICIS)*, 2009.

[42] R. Hashemian, "Design and hardware implementation of a memory efficient Huffman decoding," *IEEE Trans. Consum. Electron.*, vol. 40, no. 3, pp. 345–352, 1994.

[43] A. Natarajan, B. de Silva, K.-K. Yap, and M. Motani, "Link layer behavior of body area networks at 2.4 ghz," in *Proc. of ACM Int'l Conference on Mobile Computing and Networking (MobiCom)*, 2009.

[44] J. Ammer and J. Rabacy, "The energy-per-useful-bit metric for evaluating and optimizing sensor network physical layers," in *Proc. of IEEE Sensor and Ad Hoc Communications and Networks Conference (SECON)*, 2006.

[45] A. Y. Wang and C. G. Sodini, "A simple energy model for wireless microsensor transceivers," in *Proc. of IEEE Global Telecommunications Conference (Globecom)*, 2004.

[46] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2009.

[47] C.-H. Wu and Y.-C. Tseng, "Data compression by temporal and spatial correlations in a body-area sensor network: A case study in pilates motion recognition," *IEEE Trans. Mobile Comput.*, vol. 10, no. 10, pp. 1459–1472, 2011.

[48] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[49] F. Gavril, "Some NP-complete problems on graphs," in *Proc. of Conf. on Information Sciences and Systems (CISS)*, 1977.

[50] Analog Devices, "ADXL345 triaxial accelerometer," http://www.analog.com, 2011.

[51] C.-H. Wu, Y.-T. Chang, and Y.-C. Tseng, "Multi-screen cyber-physical video game: An integration with body-area inertial sensor networks," in *Proc. of Int'l Conf. on Pervasive Comput. and Commun. (PerCom)*, 2010.

[52] S. Trimpe and R. D'Andrea, "Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom," in *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2010.

[53] M. Sippel, A. Abduhl-Majeed, W. Kuntz, and L. Reindl, "Enhancing accuracy of an indoor radar by the implementation of a quaternion- and unscented kalman filter- based lightweight, planar, strapdown IMU," in *Proc. of the European Navigation Conf. (ENC-GNSS)*, 2008.

[54] L. Atallah, G. G. Jones, R. Ali, J. J. Leong, B. Lo, and G.-Z. Yang, "Observing recovery from knee-replacement surgery by using wearable sensors," in *Proc. of Int'l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2011.

[55] J. K. Lee and E. J. Park, "A minimum-order kalman filter for ambulatory real-time human body orientation tracking," in *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2009.

[56] C.-C. Lo, C.-P. Chiu, Y.-C. Tseng, S.-A. Chang, and L.-C. Kuo, "A walking velocity update technique for pedestrian dead-reckoning applications," in *IEEE Symposium on Personal, Indoor, Mobile and Radio Commun. (PIMRC)*, 2011.

[57] J. Chen, E. Shen, and Y. Sun, "The deployment algorithms in wireless sensor networks: A survey," *Information Technology Journal*, vol. 8, no. 3, pp. 293–301, 2009.

[58] Y.-C. Wang and Y.-C. Tseng, "Distributed deployment schemes for mobile wireless sensor networks to ensure multi-level coverage," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1280–94, 2008.

[59] C.-F. Huang, L.-C. Lo, Y.-C. Tseng, and W.-T. Chen, "Decentralized energy-conserving and coverage-preserving protocols for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 2, pp. 182–187, 2006.

[60] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Trans. Sen. Netw.*, vol. 1, no. 1, pp. 36–72, 2005.

[61] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2003.

[62] D. Tao, H. Ma, and L. Liu, "Coverage-enhancing algorithm for directional sensor networks," in *Mobile Ad-hoc and Sensor Networks*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2006, vol. 4325, pp. 256–267.

[63] Y. Wang and G. Cao, "On full-view coverage in camera sensor networks," in *Proc. of IEEE INFOCOM*, 2011.

[64] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Robust sensor placements at informative and communication-efficient locations," *ACM Trans. Sen. Netw.*, vol. 7, no. 4, pp. 31:1–31:33, 2011.

[65] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. Leung, "Body area networks: A survey," *ACM/Springer Mobile Networks and Applications*, vol. 16, no. 2, pp. 171–193, 2011.

[66] G. Welch and E. Foxlin, "Motion tracking: No silver bullet, but a respectable arsenal," *IEEE Comput. Graph. Appl.*, vol. 22, no. 6, pp. 24–38, 2002.

[67] D. A. Rodrguez-Silva, F. Gil-Castieira, F. J. Gonzlez-Castao, R. J. Duro, F. Lpez-Pea, and J. Vales-Alonso, "Human motion tracking and gait analysis: a brief review of current sensing systems and integration with intelligent environments," in *IEEE Int'l Conf. on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*, 2008.

[68] B. Huyghe, J. Doutreloigne, and J. Vanfleteren, "3D orientation tracking based on unscented kalman filtering of accelerometer and magnetometer data," in *IEEE Sensors Applications Symposium (SAS)*, 2009.

[69] Y.-X. Lai, L. Shu, A. Vasilakos, J. J. P. C. Rodrigues, C.-F. Lai, and Y.-M. Huang, "3D adaptive reconstruction of human motion from multi-sensors," in *Int'l Workshop on Wireless Sensor, Actuator, and Robot Networks (WiSARN)*, 2011.

[70] A. D. Young, "Use of body model constraints to improve accuracy of inertial motion capture," in *Proc. of Int'l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2010.

[71] Z. Zhang, L. W. Wong, and J.-K. Wu, "3D upper limb motion modeling and estimation using wearable micro-sensors," in *Proc. of Int'l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2010.

[72] Z. Zhang, "Ubiquitous human motion capture using wearable micro-sensors," in *Proc. of Int'l Conf. on Pervasive Comput. and Commun. (PerCom)*, 2009.

[73] J. Carig, *Introduction to Robotics*. Prentice Hall, New Jersey, 2005.

[74] A. D. Young, "Comparison of orientation filter algorithms for realtime wireless inertial posture tracking," in *Proc. of Int'l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, 2009.

[75] D. R. Basu and A. Lazaridi, "Stochastic optimal control by pseudo-inverse," *The Review of Economics and Statistics*, vol. 65, no. 2, pp. 347–350, 1983.

[76] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *American Statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[77] Wikipedia, "Pareto distribution — wikipedia, the free encyclopedia," http://en.wikipedia.org/wiki/Pareto_distribution, 2011.

[78] ——, "Frobenius matrix norm — wikipedia, the free encyclopedia," http://en.wikipedia.org/wiki/Matrix_norm, 2011.

[79] ——, "Power mean inequality — wikipedia, the free encyclopedia," http://en.wikipedia.org/wiki/Power_mean, 2011.

[80] G. Wang, G. Cao, and T. F. La Porta, "Movement-assisted sensor deployment," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 640–652, 2006.

Curriculum Vitae

Chun-Hao Wu

Contact Information

Address Department of Computer Science, National Chiao Tung University, 1001 University Road, Hsinchu, 30010, Taiwan.

Email wchunhao@cs.nctu.edu.tw

Education

- Taipei Municipal Ckien Kuo High School, 2000.9-2003.6.
- B.S., Department of Computer Science, National Chiao Tung University, 2003.9-2007.6.
- Ph.D., Department of Computer Science, National Chiao Tung University, 2007.9-2012.5.

Research Interests

- Body Sensor Networks
- Distributed Computing
- Pervasive and Mobile Computing
- Wireless Sensor Networks

Research Projects

- Undergraduate project, "Design and Implementation of End-to-End WiMAX/802.16e Wireless Network Simulator, and Its Performance Evaluation", granted by NSC, 2006-2007.
- Ph.D. student investigator, "Bio-information and motion tracking system for physical rehabilitation", granted by I.I.I., 2008.
- Ph.D. student investigator, "Multi-Screen Cyber-Physical Video Game Based on BSNs", granted by Ministry of Economic Affairs, 2009.
- Ph.D. student investigator, "Hand motion based wireless multitouch input device", in association with Fontal Tech., granted by National Science Council, 2009-2010.

- Ph.D. student investigator, "Intelligent Living Applications in ZigBee-Based Wireless Sensor Networks", granted by Ministry of Economic Affairs, 2011-2012.

Publication Lists

Journal papers.

1. **C.-H. Wu**, Y.-C. Tseng, "Data Compression by Temporal and Spatial Correlations in a Body-Area Sensor Network: A Case Study in Pilates Motion Recognition," IEEE *Trans. Mobile Computing*, vol. 10, no. 10, 2011 (SCI, SCIE, EI).

2. **C.-H. Wu**, Y.-C. Tseng, "Exploiting Multi-Spatial Correlations of Motion Data in a Body Sensor Network," IEEE *Communications Letters*, 2012 (accepted).

3. **C.-H. Wu**, Y.-C. Tseng, "Deploying Sensors for Gravity Measurement in a Body-Area Inertial Sensor Network," IEEE *Trans. Information Technology in BioMedicine* (submitted).

Conference papers.

1. Y.-C. Tseng, **C.-H. Wu**, F.-J. Wu, C.-F. Huang, C.-T. King, C.-Y. Lin, J.-P. Sheu, C.-Y. Chen, C.-Y. Lo, C.-W. Yang, and C.-W. Deng, "A Wireless Human Motion Capturing System for Home Rehabilitation", *Mobile Data Management*, May, 2009, Taipei, Taiwan (Recipient of **Best Demo Award**).

2. **C.-H. Wu**, Y.-T. Chang, and Y.-C. Tseng, "Multi-Screen Cyber-Physical Video Game: An Integration with Body-Area Inertial Sensor Networks", *Int'l Conf. on Pervasive Computing and Communications (PerCom)*, March, 2010, Mannheim, Germany.

3. **C.-H. Wu**, Y.-T. Chang, and Y.-C. Tseng, "On Optimization of Accelerometers Deployment for Human Posture Tracking", *Int'l Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, May, 2011, Texas, USA.

4. J.-Z. Wang, **C.-H. Wu**, Y.-C. Tseng, "VirtualTouch: A Finger Glove to Simulate Touch Screen Commands," IEEE *Sensors*, 2012 (accepted).

Academic Activities

Reviewer contribution.

1. IEEE Trans. Systems, Man, and Cybernetics: Part B.

2. Ad Hoc Networks.

3. Int'l J. of Sensor Networks.

4. Int'l Conf. on Future Information Networks.

5. IEEE Trans. Parallel and Distributed Systems.

Technical program committee.

1. IEEE International Conference on Ubiquitous Intelligence and Computing, 2012.

2. IEEE Symp. on Business, Engineering and Industrial Applications, 2012.