

# 國立交通大學

## 資訊科學系

### 碩士論文

人體區域慣性感測網路之多螢幕計算物理電子遊戲與  
重力感測問題



A Multi-Screen Cyber-Physical Game Based on Body-Area Inertial Sensor  
Networks and Its Gravity Estimation Problem

研究生：張元澤

指導教授：曾煜棋 教授

中華民國一百年六月

人體區域慣性感測網路之多螢幕計算物理電子遊戲與  
重力感測問題

A Multi-Screen Cyber-Physical Game Based on Body-Area Inertial  
Sensor Networks and Its Gravity Estimation Problem

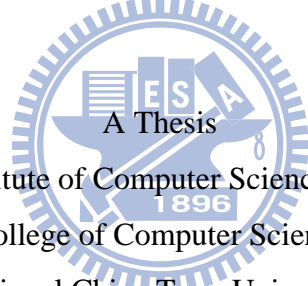
研究生：張元澤

Student：Yuan-Tse Chang

指導教授：曾煜棋

Advisor：Prof. Yu-Chee Tseng

國立交通大學  
資訊科學與工程研究所  
碩士論文



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

Master

in

Computer Science

June 2010

Hsinchu, Taiwan

中華民國一百年六月

# 人體區域慣性感測網路之多螢幕計算物理電子遊戲 與重力感測問題

學生：張元澤

指導教授：曾煜棋 教授

國立交通大學資訊科學與工程學系(研究所)碩士班

## 摘 要

近年來越來越多人關注如何透過人體區域慣性感測器來感測人體的動作，並將其應用於計算物理電子遊戲中。當計算物理電子遊戲越來越普及後，探討如何增加與改善計算物理電子遊戲的物理輸入（例如：慣性感測器）已經成為一個趨勢。基於這樣的趨勢，我們開發一個部屬於人體上之人體區域慣性感測網路來感測人體的姿勢及多螢幕的計算物理電子遊戲，讓玩家可以更真實的體驗遊戲。我們設計一套多螢幕遊戲引擎。其中包含人體區域慣性感測網路平台，計算物理遊戲控制器及一組遊戲引擎。我們的遊戲可以讓玩家與虛擬物件互動。

人體追蹤系統是基於加速度計。而其中一個基本的議題為，不管人體是否在移動，重力加速度的方向是否可以準確地得到。假設一個人體的剛體上部屬多個加速度感測器，近期的研究透過資料融合的法來預測該剛體上的重力加速度方向。然而，如何將加速度感測器部屬在最佳的位置來減少預測誤差仍然是一個開放問題。在這篇論文中，我們將部屬最佳化的問題公式化，展示部屬方式的方針及兩種部署策略 virtual-force-based (VF-based) method 及 Metropolis-based。

關鍵字：計算物理電子遊戲、慣性感測器、加速度計、多螢幕、重力加速度

# A Multi-Screen Cyber-Physical Game Based on Body-Area Inertial Sensor Networks and Its Gravity Estimation Problem

Student: Yuan-Tse Chang

Advisor: Prof. Yu-Chee Tseng

Department of Computer Science  
National Chiao Tung University, Taiwan

## Abstract

Deploying body-area inertial sensor networks on human bodies to capture motions has attracted a lot of interests recently, especially in cyber-physical video games and context aware applications. While video games on the cyber world have been quite popular, enhancing them with more physical inputs, such as those from inertial sensors, is becoming a new trend. Following this trend, we develop a video game integrated with body-area inertial sensor networks deployed on players as inputs and with multiple game screens to broaden players' views and provide more realistic interaction experiences. Our design simulates a multi-screen game engine by controlling a set of game engines simultaneously. A prototype with a body-area inertial sensor network platform, a cyber-physical game controller, and a set of game engines is demonstrated. The demonstrated game also addresses the interaction between virtual objects and players.

Tracking human posture system based on accelerometer. One fundamental issue in such scenarios is how to calculate the gravity, no matter when human body parts are moving or not. 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 on that rigid part. However, how to find the optimal deployment of sensors that minimizes the estimation error of the gravity is still an open problem. In this paper, we formulate the deployment optimization problem, present deployment guidelines, and propose some heuristics, including a virtual-force-based (VF-based) method and a Metropolis-based search method. Experimental results are presented to verify our results.

**Keywords:** cyber-physical video game, inertial sensor, accelerometer, multi-screen, gravity.



## 致 謝

碩士生涯轉眼即逝，在這兩年裡，無論是課業上或是生活上都有不少收穫。在這兩年學習的過程中，特別感謝曾煜棋教授不辭辛勞的指導，感謝吳鈞豪學長在每周的面談及撰寫論文時給予我相當多寶貴的意見。感謝 HSCC 實驗室的同學、學長姐、學弟妹以及曾經幫助過我的人。謝謝你們這兩年來的幫忙跟照顧。最後感謝父母親及家人在我求學過程中給予我各方面的協助，感謝爸爸在口試前一晚上陪我做最後的演練。

### 台灣閩南語(漢字)

碩士生涯一目矚仔就過啊，佇這兩年內底，無論是學業上抑是生活上攏學了袂少。佇這兩年學習的過程中，特別感謝曾煜棋教授的教示佢指導，感謝吳鈞豪前輩佇逐禮拜的面會中佢寫論文時予我真濟寶貴的意見。感謝 HSCC 實驗室的同窗、先輩、後輩佢捌共我幫助過的人，多謝恁佇這兩年的幫忙佢照顧。最後感謝序大人佢親人佇我讀冊過程中予我各方面的協助，感謝阿爸佇口試前一暗陪我做最後的演練。

### 台灣閩南語(羅馬字拼音)

Sik-sū sing-gâi tsit-bák-nih-á tō kuè-a, tī tse n̄ng-nî lâi-té, hô-lūn-sī hák-giap-siōng iah-sī sing-uah-siōng long ôh-liáu buē-tsió. Tī tse n̄ng-nî hák-sip ê kuè-tîng-tiong, tik-piát kám-siā 曾煜棋 kàu-siū ê kà-sī kah tsí-tō, kám-siā 吳鈞豪 tsian-puè tī tak lé-pài ê biān-huē tiong kah siá lūn-bûn sí hōo guá tsin tsē pó-kuì ê ì-kiàn. Kám-siā HSCC sít-giām-sik ê tông-tshong, sian-puè, hiō-puè kah kā guá pang-tsōo kuè ê lāng, to-siā lín tī tse n̄ng-nî ê pang-bāng kah tsiau-kòo. Tsuè-āu kám-siā sī-tuā-lāng kah tshin-lāng tī guá thák-tsheh kuè-tîng-tiong hōo guá kok hong-bīn ê hiáp-tsōo. Kám-siā a-pah tī kháu-tshì tsing tsit àm puê guá tsò tsuè-āu ê ián-liān.

### 台灣客家語(漢字)

碩士生活過个當遽，在這兩年間，無論係學業還係生活上都學了恁多。在這兩年學習的過程間，感謝曾煜棋教授个指教，感謝吳鈞豪前輩在每擺个面談同寫論文時分 佢恁多建議。感謝HSCC實驗室个同窗、前輩、後輩同幫助過 佢个人，感謝大家這兩年个照顧。最尾感謝爺哀同親人在 佢讀書个過程間分 佢各方面个協助，感謝阿爸在口試前暗晡陪 佢做最尾个練習。

### 台灣客家語(拼音-四縣腔)

Sag sii sen fad go ge dong' giag', cai ia` liong` ngien` gien', mo' lun hog ngiab song han' he sen fad song du hog liau` an` do'. Cai ia` liong` ngien' hog xib ge go cang' gien', gam` qia 曾煜棋 gau su ge zii` gau, gam` qia 吳鈞豪 qien' bi cai mien tam' tung' xia` lun vun' sii' bun' ngai' an` do' gien ngi. Gam` qia HSCC siid ngiam siid` tung' hog, qien' bi, heu bi tung' bong' cu go ngai' ge ngin', gam` qia tai ga' ia` liong` ngien' gien' ge zeu gu. Zui mi' gam` qia ia' oi' tung' qin' ngin' cai ngai' tug su' ge go cang' gien' bun' ngai' gog' fong' mien ge hiab cu, gam` qia a' ba' cai kieu` sii qien' am bu' pi' ngai' zo zui mi' ge lien xib.

# Contents

摘要	i
Abstract	ii
誌謝	iii
Contents	v
List of Figures	vi
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>2</b>
<b>3 Multi-Screen Cyber-Physical Game</b>	<b>3</b>
3.1 How to play this game	3
3.2 System Architecture	4
3.2.1 Body-Area Inertial Sensor Network	5
3.2.2 Cyber-Physical Game Controller	7
3.2.3 Game Engine	9
3.3 Demonstration	11
<b>4 Gravity Estimation Problem</b>	<b>13</b>
4.1 Data fusion	14
4.2 Observation	16
4.3 Optimization Heuristics	17
4.3.1 A Virtual-Force-Based Method	17
4.3.2 A Metropolis-Based Method	19
<b>5 Simulation and Experimental Results</b>	<b>20</b>
5.1 Simulation Setup	21
5.2 Comparison of $\sigma_e^2(P)$	21
5.3 Effects of More Sensors and Execution Time	22
5.4 Actual Gravity Estimation	25
<b>6 Conclusions</b>	<b>27</b>

# List of Figures

1	An overview of our Multi-Screen Cyber-Physical Game. . . . .	4
2	System architecture. . . . .	5
3	Software architecture. . . . .	6
4	Inertial sensor . . . . .	8
5	Human body model . . . . .	9
6	Sensor reading to orientation . . . . .	10
7	Unity development screen . . . . .	11
8	Game screen with mouse move . . . . .	11
9	360° panorama view . . . . .	12
10	Player . . . . .	12
11	Snapshots of the game . . . . .	12
12	Virtual character and camera settings in Fig. 11 . . . . .	13
13	Gravity vector is incorrect when move. . . . .	13
14	The kinematical model for a rigid body . . . . .	15
15	The radius of a sphere effects error variances. . . . .	17
16	An example of iterative projection on three consecutive polygons. . . . .	18
17	An example of running the VF-based method by game engine Unity. . . . .	19
18	Adjacent grid points of a current location $p_i$ . . . . .	19
19	Different geometries . . . . .	22
20	Comparison of different geometries, origin at geometry center . . . . .	22
21	Comparison of different geometries, origin at geometry edge . . . . .	22
22	Comparison of different geometries, origin is away from the geometry edge . . . . .	22
23	Effects of $m$ sensors. . . . .	23
24	Comparison of running times. . . . .	24
25	Three view of the box. . . . .	25
26	Rotating the box up and down. . . . .	25
27	Sensors' locations on the box in Fig. 26. . . . .	26
28	Comparison of length waveforms of measured gravity vectors. . . . .	26
29	Comparison of five deployments. . . . .	27
30	Comparison of standard errors of one sensor deployments. . . . .	28

# 1 Introduction

*Cyber-physical systems* (CPS) have attracted a lot of attention recently. Cyber system also grows very fast in the past recent years, due to the advance of mobile computing technologies, various wireless communication and networking technologies (such as WiFi, ZigBee, Bluetooth, etc.). On the other hand, the development of *Micro Electro Mechanical Systems* (MEMS) can enhance cyber systems with more physical inputs and actuators. Traditionally, the human interface in cyber video game is keyboards and joystick. Now more and more cyber video game use more kindly human inter face to attract players attention, such as body motions captured by inertial sensors, camera, 3D depth sensor. The Nintendo Wii [1] is one example with MEMS-based inertial sensors. XBOX360 Kinect [2] use camera, 3D depth sensor to capture player's action. Playstation Move [3] use a stick with 3-Axis Accelerometer, 3-Axis gyro , 3-Axis Magnetometer, and the stick also have a light ball on it, so the camera can capture the stick's position. Reference [4] shows that by using MEMS-based inertial sensors and a wired network, body motions can be reconstructed through computer animation with little distortion. 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 [5,6].

The above advances motivate us to develop a multi-screen cyber-physical video game with physical inputs from players equipped with body-area inertial sensor network (BISN). A BISN consists of multiple accelerometers, magnetometers, and gyroscopes connected by wired/wireless links. An important usage in a BISN is to track human motions through these inertial sensors. Its applications include cyber-physical video game [7], robotic balancing [8], localization [9], sports training [10], medical care [11], and computer graphics [4].

A lot of works have studied how to track human postures. Basically, human motions can be tracked by estimating rotations of joints, via accelerometers, electric compasses, and/or gyroscopes [12]. Accelerometers sense the directions of the gravity, compasses sense the directions of the North, and gyroscopes estimate angular velocities. However, since a human body may continuously move, this is not an easy task.

In this paper we consider 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. We develop a video game integrated

with body-area inertial sensor networks deployed on players as inputs and with multiple game screens to broaden players' views and provide more realistic interaction experiences.

We also consider a fundamental issue in a BISN, the gravity measurement problem. Regarding a human body as multiple rigid parts connected by joints, we study the deployment of accelerometers on one rigid part and the estimation of the gravity on that rigid part. The acceleration perceived by an accelerometer is the gravity (which is the same for all accelerometers on the rigid part) plus its own acceleration seen by an outside observer. Since the orientation of the rigid part is unknown, how to measure the gravity is a challenging problem. Given a sensor deployment, [8] shows a data fusion scheme to extract the gravity. Based on [8] we try to find the best locations of accelerometers that minimize the estimation error of the gravity. We present guidelines that can be applied to arbitrary geometries, which state that sensors should be evenly distributed over the surface of the rigid part such that the distances between any pairs of them are maximized. We show that this guideline leads to the optimal solutions for some special cases and near-optimal solutions in general. Then we propose two heuristics, called *virtual-force-based (VF-based)* and *Metropolis-based* method. Experimental results show that both methods perform quite well, even for complicated geometries.

The rest of paper is organized as follows. Section 3 describes about the Multi-Screen Cyber-Physical Game. Section 4 formulates the gravity estimation problem. Experimental results are given in Section 5, and conclusions are drawn in Section 6.

## 2 Related Work

In Cyber-physical systems, which are the integrations of computational and physical processes. [13] discuss current trends in the development and use of high-confidence medical cyber-physical systems (MCPS). These trends, including increased reliance on software to deliver new functionality, wider use of network connectivity in MCPS, and demand for continuous patient monitoring. [14] explore the temporal and spatial properties of events, define a novel CPS architecture, and develop a layered spatiotemporal event model for CPS. [15] present AnySense, a network architecture that supports video communication between 3G phones and Internet hosts in cyber-physical systems. AnySense implements transcoding of video streams between the Internet and circuit-switched 3G cellular net-

works, and is transparent to 3G service providers. [16] look at modern buildings entirely as a cyber-physical energy system and examine the opportunities presented by the joint optimization of energy use by its occupants and information processing equipment. The security issue be mention in [17] [18].

Basically, human motions can be tracked by estimating rotations of joints, via accelerometers, electric compasses, and/or gyroscopes [12]. When structures of articulated objects are unavailable, rotations of a single node may be tracked by fusing sensing data of different modalities [9] [12] [19]. With knowledge of human body structures, [20] [21] detect incorrect estimations of rotations that are outside of reachable regions of joints, and [22] compensates linear accelerations under body movements. For human location estimation in areas of static magnetic disturbances, [23] uses four magnetometers forming an orthogonal trihedron to gain more perception of magnetic fields.

### 3 Multi-Screen Cyber-Physical Game

In multi-screen cyber-physical game, we put BISN on human bodies to capture motions. With BISN on human bodies, the player will have more kindly human interface than the traditional one, such as keyboard or mouse. In the game, player can use its own human posture to be the game input information. Besides, we also develop a Multi-screen video game with a 360° panorama view. The rest of this section is organized as follows. Subsection 3.1 describe about how to play our game. Subsection 3.2.3 briefly introduce our system from bottom to up. Subsection 3.3 demonstrate a prototype of our game system.

#### 3.1 How to play this game

The overview of our system architecture is shown in Fig. 1. The player is surrounded by four screens. Each screen is set individually at North, East, West and South. Besides, in the game scene, each screen always captures a constant direction view. For example, the North screen always captures the North view of the game scene; the South screen always captures the South view of the game scene, and so on. In the middle of these four screen, we consider a single-player game; the player act as same as the virtual character which is in the game scene. For example, when the player raises his hand, the virtual character in

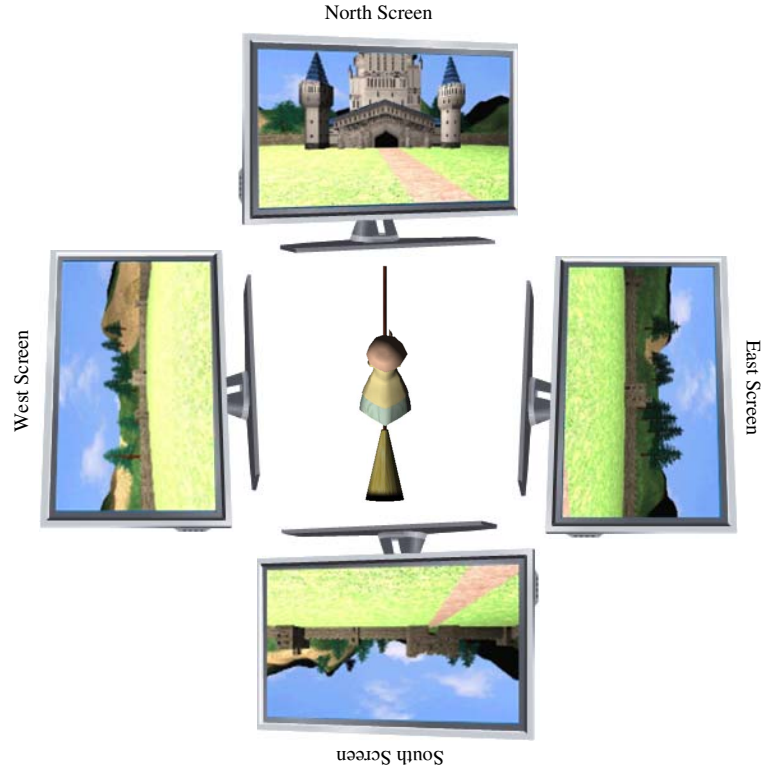


Figure 1: An overview of our Multi-Screen Cyber-Physical Game.

the game also raises his hand.

We adopt the Quidditch [24] sports in "Harry Potter" as our game scenario. In the game scene, the player flies around the practice field and try to hit the ball inside the game. In our game scenario, the virtual character in the game scene flies in a constant speed. The player use flying broom to control the flying direction and use bat to hit the ball. For example, when the player's flying broom is directed to the North screen, the virtual character in the game scene fly to the North. For capture human body and game equipment (flying broom and the bat), we put four wireless sensors on forearm, upper arm, flying broom and bat. Each sensor consist accelerometer and magnetometer. Accelerometer senses the direction of the gravity; magnetometer senses the direction of the North.

### 3.2 System Architecture

Our system architecture is shown in Fig. 2. In the middle of the room, we consider a single-player game; the BISN is deployed on the player to capture its motions. The sensing data are reported to the sink by wireless transmission. The sink is connected to



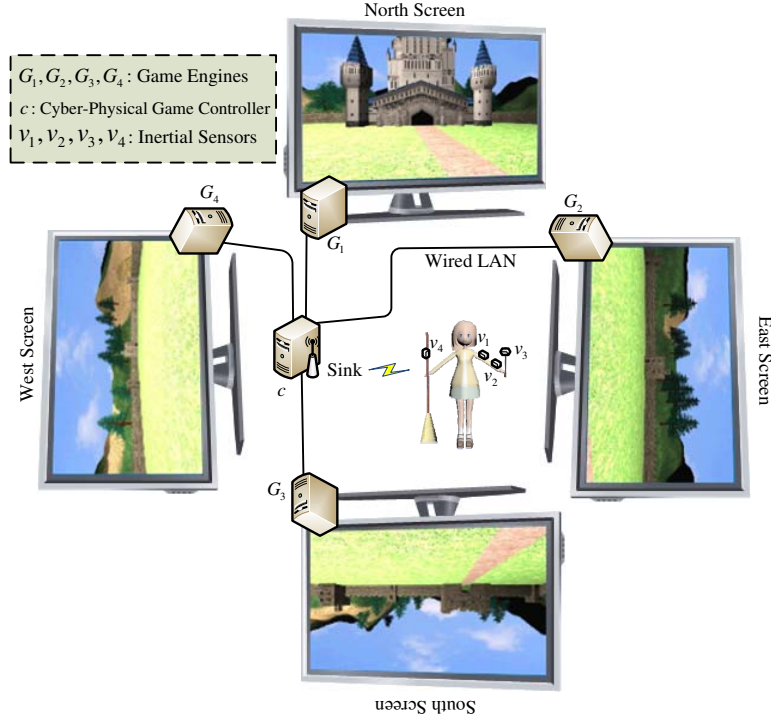


Figure 2: System architecture.

the cyber-physical game controller by USB, and the collected sensing data are processed by cyber-physical game controller and feed it four game engines  $G_1$ ,  $G_2$ ,  $G_3$ , and  $G_4$  via wired LAN interfaces. We place four game engines together with their screens near the walls of the room. Each game engine are set to four camera angles (east, west, north, and south) and they together provide a  $360^\circ$  panorama view of the game scene. The software architecture is shown in Fig. 3. Below, we briefly describe each component.

### 3.2.1 Body-Area Inertial Sensor Network

In the wireless sensing platform, the sink and the nodes worn on the player form a BISN. 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 microcontroller, and a wireless transmission module. In our work, each node is equipped with a tri-axial accelerometer and a tri-axial electronic compass. The tri-axial accelerometer senses the gravity in  $x$ ,  $y$ ,  $z$ -axes, while the tri-axial compass senses the earth magnetic force in  $x$ ,  $y$ ,  $z$ -axes. Note that gravity acceleration dominates the accelerations sensed by an accelerometer. The data are reported to the sink through the "Wireless Module", where we runs a polling MAC to avoid collisions and to improve throughput. For the BISN platform, we adopt Jennic

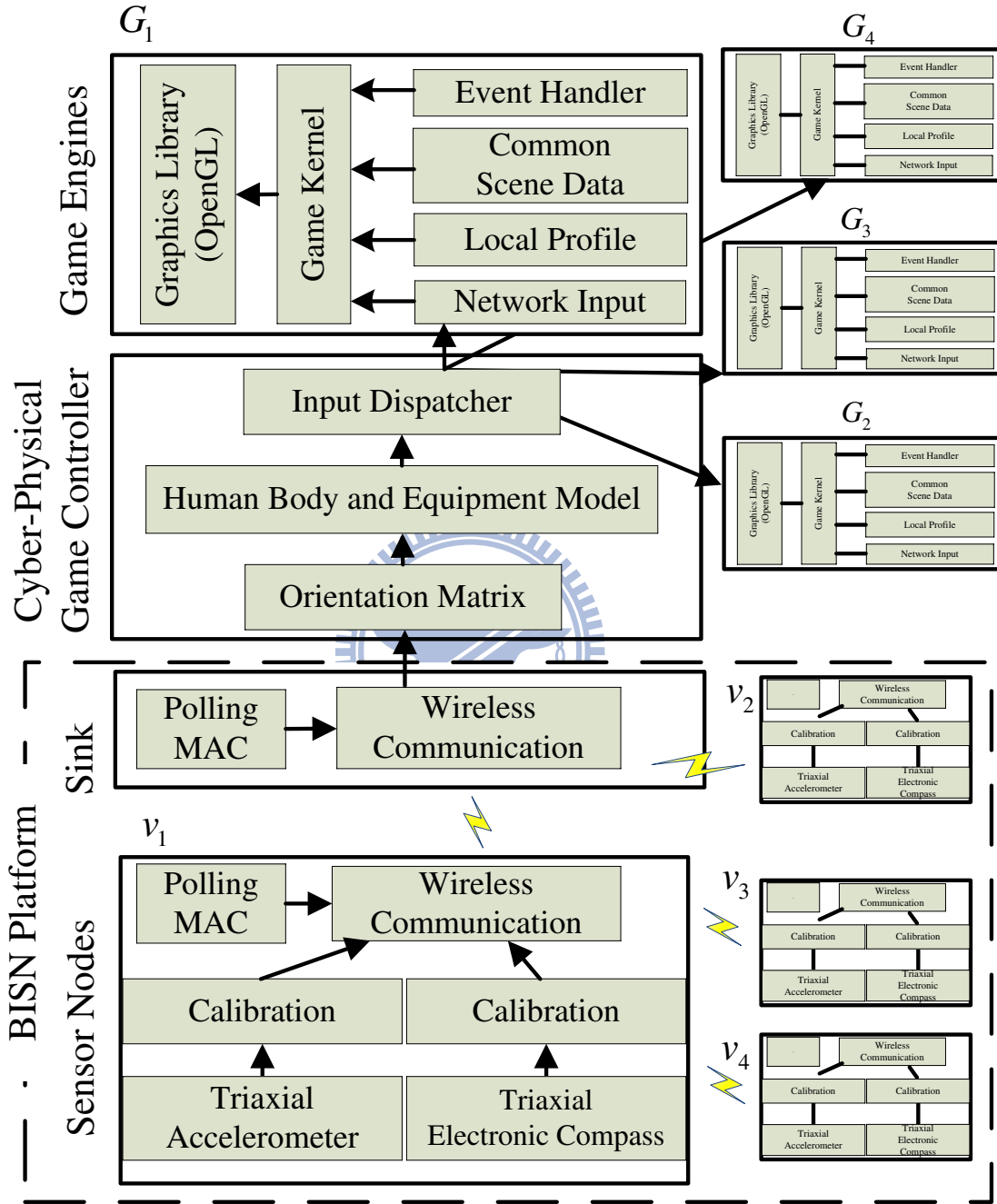


Figure 3: Software architecture.

JN5139 [25] and OS5000 IMU sensor [26].

**Nodes** Each node collects the sensing data from OS5000 through the Jennic UART port. After receive the data from the UART port, the data will store at the RAM temporary and wait for the sink to pull it. When the sink start to pull, Jennic will read the data from RAM and send it to the sink.

**OS5000 sensor** We use OS5000 to be our body sensor. The OS5000 family is a law cost compasses within a tiny 1 inch square footprint. OS5000 combines 3-axis magnetic with 3-axis accelerometers providing OEM users a precise tilt compensated heading, pitch and roll information suitable for a wide range of applications. Compass units offer an ASCII interface that includes both hard-iron and soft iron compensation and simple, user-configurable data formatting. In our prototype the sampling rate is set to 20Hz, a common value for motion capturing and we use 3-axis magnetic and 3-axis accelerometers to be the sensing reading, we don't use pitch and roll information because the pitch may not correct when pitch angle is over 70°.

**Sink** Sink is one of the most important devices in the BISN. It collects all packets from other nodes and sends them to the cyber-physical game controller from RS-232. When user wants to send commands to any device, it needs to transmit through sink.

**Jennic** We use Jennic JN5139 to develop BISN (Fig. 4). It's a experimental tools using in sensor network researching. Each Jennic JN5139 consists of a micro-controller and an IEEE 802.15.4 wireless transmission module and we also use Jennic JN5139 as the sink. Our body inertial sensor is based on the Jennic JN5139 with 16MHz 32-bit RISC CPU, 96kB RAM and 192kB ROM. The Jennic JN5139 includes 4-input 12-bit ADC, 2 11-bit DAC, 2 comparators, two application timer/counters, three system timers, two UARTs, 21 GPIO, 5 SPI port to select and 2-wire serial interface.

### 3.2.2 Cyber-Physical Game Controller

The first part of this module converts collected sensing data into inputs of single-screen game engines. In our demonstration, the inputs include the direction of the broomstick and the player's gestures. Given the sensing data, the "Orientation Matrix" block cal-

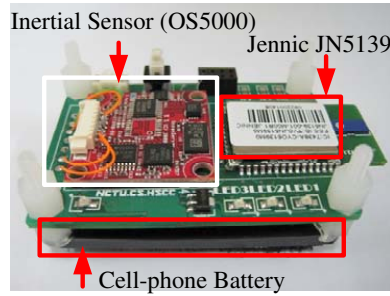


Figure 4: Inertial sensor

culates the absolute orientations of nodes with respect to the earth coordinate. The direction of broomstick can be obtained directly from its orientation matrices, and the player’s gestures are calculated from orientation matrices and a human model. The second part of this module dispatches the inputs to each single-screen game engine via a wired LAN. Note that traditional game engines and developing tools do not support multiple screens. By using this module to control the single-screen game engines, we can simulate a multi-screen game engine.

We run these game engines with the same game data, except that their cameras may have different settings. For example, the cameras may look at different directions to form a 360° panorama view of the game scene. Note that by feeding game engines with the same inputs, their behavior will be the same. This architectural design makes it simple to implement, and it has the potential to reuse existing games without modification.

**Input Dispatcher** 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. In our prototype, input dispatcher sends the proper parameters such as articulation angles, game objection position to the game engines via wired LAN interfaces.

**Human Body Model** We regard a human body as multiple movable parts (Fig. 5), each being a rigid body connected to another part by a rotational joint [27, 28]. Each rigid body has its own coordinate. For capture the rigid body’s tilt, the sensor is set at the rigid body. The sensor rendering the rigid bodies tilt information and we can use the tilt information to know the rigid bodies tilt in the earth coordinate through orientation matrix. In our prototype the player wears four inertial sensor nodes, one on

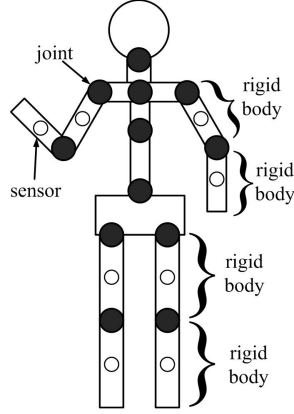


Figure 5: Human body model

the broomstick, one on forearm, one on upper arm, and one on the club.

**Orientation Matrix** As Fig. 6 sensors are putted on forearm and upper arm. Each sensor receives the 3-axis magnetic and 3-axis accelerometer from the OS5000. When the sensor is at rest, the 3-axis accelerometer reading should consist of only gravity, and the 3-axis magnetic reading consist of magnetic which direct to the North Magnetic Pole. Due to the magnetic reading is not horizontal to the horizon; we use Gram-Schmidt process [29] to let the magnetic reading project to the horizon, so that we can know the North direction. With the North direction and gravity direction is known, we can easily find the East direction, and the "Orientation Matrix" [27] represents the absolute orientations of all sensor nodes with respect to the earth coordinate can be found. For example, the gravity always directs to the ground. So if we know the gravity, the up direction is defined. But the accelerometer's readings not only sense the gravity but also sense other acceleration. With the wrong acceleration reading, the Orientation Matrix may represent the wrong orientation. We define this problem as a Gravity Estimation problem. And more detail about gravity estimation problem will discuss at 4.

### 3.2.3 Game Engine

We adopt Unity [30] as our game engine. Unity is a game development tool which features a fully integrated editor and physics engine for rapid 3D game prototyping. Coding is an easy job in Unity. In Unity we can do many works with only use script language. Besides, any object in Unity is fully objective.

In Unity, each game level is saved to a scene (Fig. 7) and each game level switches

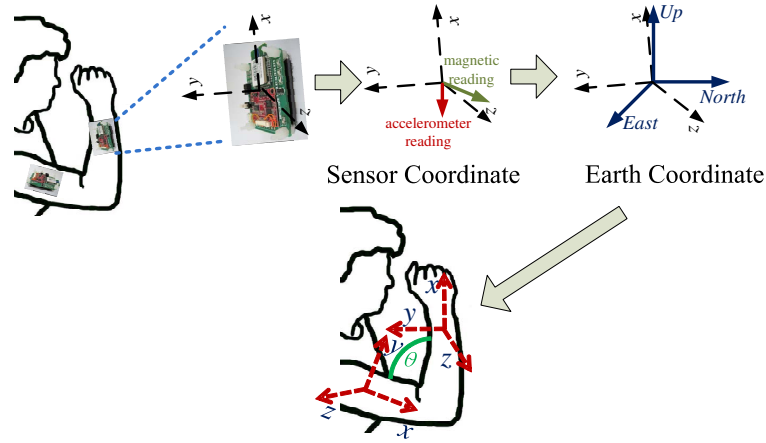


Figure 6: Sensor reading to orientation

through script. The "scene" is the "Fix part" in the game, it won't be changed during the executing, such as lighting, game scene, decorate game object, background sound will be set to scene. In the multi-screen video game "Camera" is an important part. Traditionally, the camera is mount to a mouse view, the screen view is moved when the player move the mouse (Fig. 8). In multi-screen video game, the camera mounts to the player and the camera always capture to a static direction. For example, if we have four screens in the room, each cameras capture view is  $90^\circ$  (Fig. 9); if we have six screens, each cameras capture view is  $60^\circ$ . In our prototype we use four screens.

Each game engine represents a view port to the game scene and each game engine has a "Game Kernel", which takes inputs from four modules. All game engines use the same game data except for the settings of cameras. Upon receiving inputs from the cyber-physical game controller, they update game status and settings of the cameras in the same way. For example, when the multi-screen controller broadcasts "moving east" to all game engines, each game engine moves its camera one unit to the east.

The "Common Scene Data" module describes the virtual world. 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 "Local Profile" that takes pictures for its virtual world and feeds the captured data to

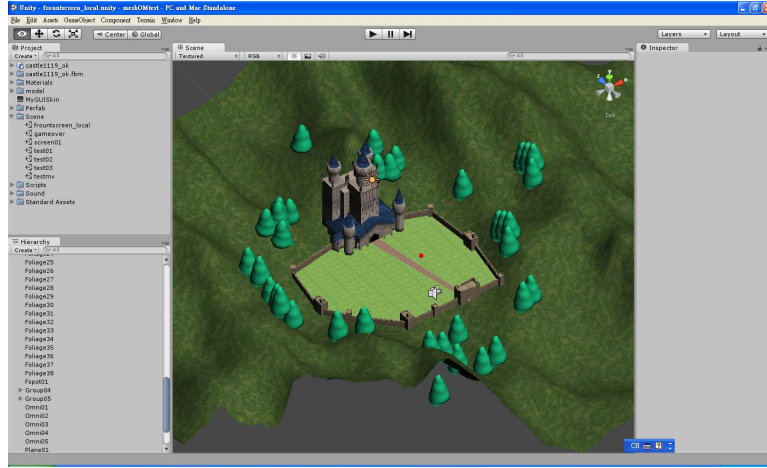


Figure 7: Unity development screen

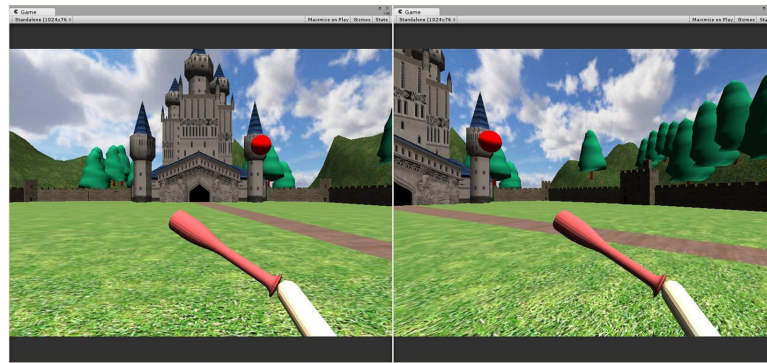


Figure 8: Game screen with mouse move

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.

### 3.3 Demonstration

We adopt the Quidditch sports in Harry Potter [24] as the game scenario to demonstrate the multi-screen cyber-physical game engine. The player Fig. 11 rides a flying broom in a practice field, and she practices to fly to the ball by her club. She flies at a constant speed and controls the broomstick to change her flying direction.

Fig. 11 shows some snapshots of the game. The player flies a short distance to the north in Fig. 11 (a), and then she turns to the west in Fig. 11 (b). We adopt Unity as our game engine, which features a fully integrated editor and a physics engine for rapid



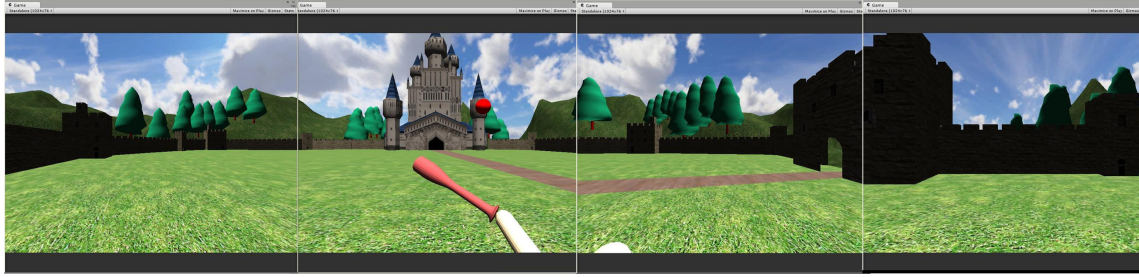


Figure 9: 360° panorama view



Figure 10: Player

3D game prototyping. Fig. 12 (a) and Fig. 12 (b) show the corresponding settings of the west and the north game engines in Fig. 11 (a), respectively. The virtual character, which is controlled by the player, looks to the north in both game engines. The camera of Fig. 12 (a), whose viewing volume is illustrated by white lines, looks to the west, while the camera of Fig. 12 (b) looks to the north. Fig. 12 (c) and Fig. 12 (d) correspond to the west and the north screens in Fig. 11 (b), respectively, where both of the player and the virtual character look to the west, but the directions of the cameras remain unchanged.

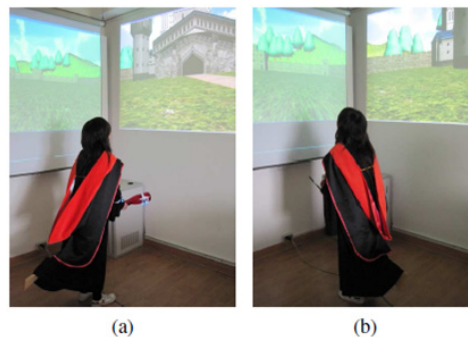


Figure 11: Snapshots of the game



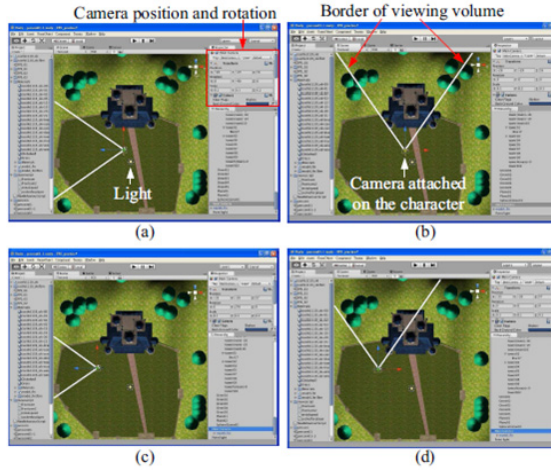


Figure 12: Virtual character and camera settings in Fig. 11

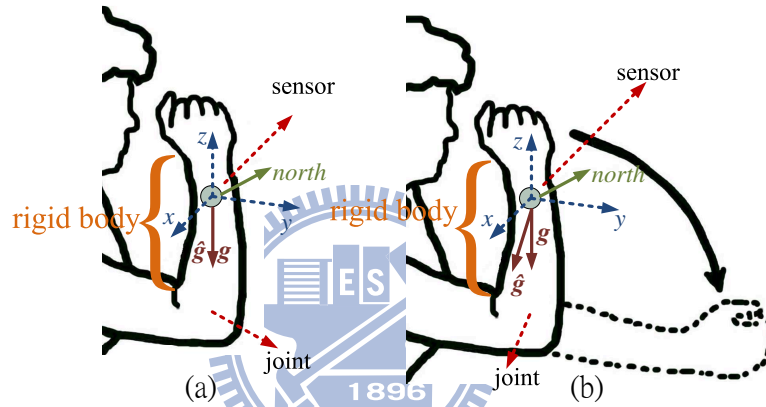


Figure 13: Gravity vector is incorrect when move.

## 4 Gravity Estimation Problem

In cyber-physical systems based on IMU sensor networks (where sensors typically include accelerometers, magnetometers, and gyro), human motions are captured by sensor nodes deployed at movable parts (such as limbs) and used as physical inputs and feedbacks to the systems. Tracking human postures is done by estimating rotations of joints, which are typically derived from sensing data of accelerometers, electric compasses, and gyroscopes [12]. Accelerometers sense the directions of the gravity, compasses sense the directions of the North, and gyroscopes help to stabilize the estimation. Traditionally, each rigid body put an accelerometer to sense the gravity. As Fig. 13, assume we put an internal sensor on forearm, Fig. 13 (a) is static, Fig. 13 (b) is in move, let  $\hat{g}$  be the measurement which is measure from the accelerometers and let  $g$  be the gravity, when the rigid body is rotating, the reading of accelerometer may consists gravity plus it's on acceleration seen

by an outside observer. But in accelerometer-based tilt estimation, we only interesting in gravity.

Given a sensor deployment, [8] shows a data fusion scheme to extract the gravity based on multi-accelerometer. However, improper deployments may significantly increase estimation errors, so there is room to further optimize the deployments. At the rest of this section, we consider how to find an optimal deployment to improving the estimation accuracy of accelerometers, and thus limit our discussions to only accelerometers.

## 4.1 Data fusion

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. Clearly, the gravity as being measured by each sensor is relative to its placement and angle. We regard a human body as multiple movable parts, each being a rigid body connected to another part by a rotational joint [27, 28]. Accelerometers are placed on a human body for posture tracking. The concept is shown in Fig. 5.

To formulate the gravity measurement problem, we consider one rigid body and the sensors on it, as illustrated in Fig. 14. 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  $x_\theta$ ,  $y_\theta$ , and  $z_\theta$  as its axes. The gravity  $g$  with respect to this  $\theta$ -coordinate is thus  $-1$  Guass along the  $z_\theta$  axis. Let the joint of the rigid body be at location  $r(t)$  at time  $t$ . For the rigid body, we assume a part-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, for any point on the part, its coordinate with respect to the  $r$ -coordinate remains unchanged no matter how the body moves. For any point at location  $p$  with respect to the  $r$ -coordinate, its location with respect to the  $\theta$ -coordinate changes over time  $t$  and can be written as  $p'(t) = r(t) + R(t)p$ , where  $R(t)$  is the  $3 \times 3$  rotation matrix translating from  $(x_r, y_r, z_r)$  to  $(x_\theta, y_\theta, z_\theta)$  [27].

Let  $s_1, s_2, \dots, s_m$  be  $m$  accelerometers deployed on the rigid body and  $p_1, p_2, \dots, p_m$  be their locations with respect to the  $r$ -coordinate, respectively. Without loss of generality, we assume that these accelerometers are properly placed in the sense that their  $x$ -,  $y$ -, and  $z$ -axes are perfectly 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

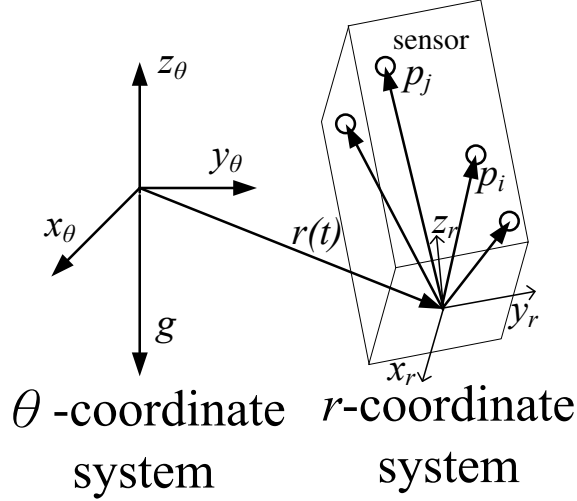


Figure 14: The kinematical model for a rigid body

the translation). This implies that the gravity being observed in the  $r$ -coordinate is the same as that being observed by any sensor.

Now, consider any sensor  $s_i$ ,  $i = 1, 2, \dots, m$ . Its location in the  $\theta$ -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 in the  $\theta$ -coordinate:

$$\ddot{p}'_i(t) = \ddot{r}(t) + \ddot{R}(t)p_i. \quad (1)$$

Since  $s_i$  is aligned to the  $r$ -coordinate and noise should be included, the actual reading  $a_i(t)$  of  $s_i$  should be  $a_i(t) = R^T(t)(-\ddot{p}'_i(t) + g) + n_i(t)$ . Note that both  $a_i(t)$  and  $n_i(t)$  are  $3 \times 1$  vectors, and each element of  $n_i(t)$  has a zero mean with a standard deviation  $\sigma_n$ . (For example, when the rigid body is at rest, the reading of  $s_i$  should consist of only gravity, giving  $a_i(t) = R^T(t)g + n_i(t)$ ; when it falls freely, the reading should be  $a_i(t) = n_i(t)$ .)

Below, assuming a fixed  $t$ , we will omit time information in our formulation. Plugging Eq. (1) into  $a_i$ , we have

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

Putting these  $m$  equations together, we have the equality:

$$A = QP + N, \quad (2)$$

where

$$A = \begin{bmatrix} a_1 & \dots & a_m \end{bmatrix} \in \mathfrak{R}^{3 \times m},$$

$$Q = \begin{bmatrix} R^T (-\ddot{r} + g), & -R^T \ddot{R} \end{bmatrix} \in \mathfrak{R}^{3 \times 4},$$

$$P = \begin{bmatrix} 1 & \dots & 1 \\ p_1 & \dots & p_m \end{bmatrix} \in \mathfrak{R}^{4 \times m},$$

$$N = \begin{bmatrix} n_1 & \dots & n_m \end{bmatrix} \in \mathfrak{R}^{3 \times m}.$$

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

Since  $\ddot{r} \ll g$  in  $Q$ , which is common for human motion,  $Q$ 's first column vector  $R^T(-\ddot{r} + g) \approx R^T g$ . By estimating  $Q$ , we can determine  $R^T g$ . Let  $\hat{Q}$  be an estimation of  $Q$ . Following [8], 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  $m = 4$  [31]. This implies that to find  $\hat{Q}$  we need at least four sensors. Since  $\hat{Q} - Q$  is a zero-mean vector, one needs to measure how  $\hat{Q}$  is close to  $Q$ . In [8], an *error variance*, which solely depends on the deployment matrix  $P$ , is defined:

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

where  $\rho_k(P)$  is the  $k$ th 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 part, the goal is to find the deployment matrix  $P$  such that the error variance  $\sigma_e^2(P)$  is minimized. Note that since the  $3\sigma_n^2$  in Eq. (3) is a constant, we only need to focus on the summation part.

## 4.2 Observation

It is natural to ask how the size of a rigid part affects the estimation errors. To observe it, we vary the radius of a sphere from 1 to 10 and measure the error variance of some fixed deployment. That is, if the location of sensor  $s_1$  is  $p_1$  on a unit sphere, it becomes  $10p_1$  on a sphere whose radius is 10. The results are shown in Fig. 15 for two deployments. We can see that the error variances decrease as the radius grows. The slope is quite steep at the beginning, implying that small objects are more vulnerable to sensor deployments.

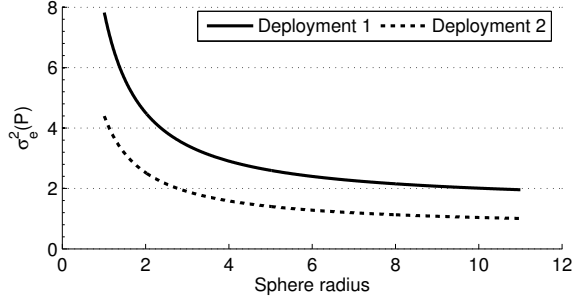


Figure 15: The radius of a sphere effects error variances.

### 4.3 Optimization Heuristics

As Fig. 15, intuitive deployments may result in poor performance. 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 some observations and guidelines. Then we realize them by a virtual-force-based (VF-based) method and a Metropolis-based search method.

#### 4.3.1 A Virtual-Force-Based Method

We present a virtual-force-based (VF-based) deployment method that can be applied to rigid bodies of arbitrary shapes. It approaches the gravity measurement problem by following the LD guideline and distributing sensor nodes according to the electronic particles distribution principle: *the nearer the particles, the stronger the repelling forces between them*. This algorithm works as follows.

1. First, randomly place sensors  $s_1, s_2, \dots, s_m$  on the surface of the rigid body. Let  $p_1, p_2, \dots, p_m$  be their locations, respectively.
2. For each pair of sensors  $s_i$  and  $s_j$ , the repelling force contributed by  $s_j$  on  $s_i$  is defined by a  $3 \times 1$  vector

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

where  $\beta$  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 for a period of  $T$  pushed by force  $f_i$ . Assuming that each sensor has a unit mass, we model the displacement by  $f_i T$ .

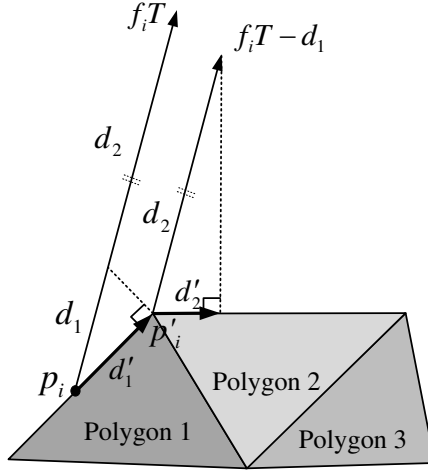


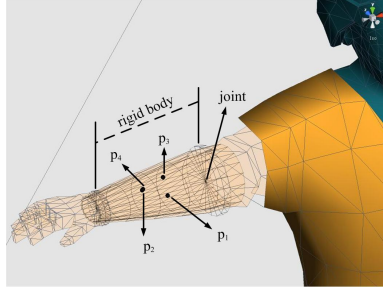
Figure 16: An example of iterative projection on three consecutive polygons.

So the new location of  $s_i$  is  $p_i + f_i T$ . Since this location may not be on the surface, a projection onto the surface is needed (we will discuss how to conduct the projection below).

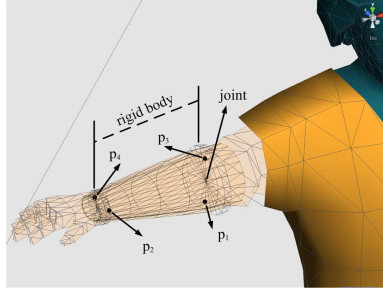
4. Let  $p_i^{(old)}$  and  $p_i^{(new)}$  be the previous and next locations, respectively, of  $s_i$ . We test whether  $\sum_{i=1}^m \|p_i^{(new)} - p_i^{(old)}\| \leq \sigma_{th}$  or the allowed number of iterations is reached. If so, we terminate this algorithm; otherwise, go back to step 2.

The above algorithm follows the virtual-force discipline [32] to maximize the inter-distances among sensors and thus our LD guideline. Note that since the final locations depend on the initial locations, to avoid finding only a local minimum, we may repeat the algorithm several times with randomized initial locations.

In the above step 3, how to conduct projection needs further explanation. To be computationally feasible, we approximate a surface by polygons as usually done in computer graphics. Fig. 16 shows how this works. Sensor  $s_i$  is at location  $p_i$  and is pushed by a displacement  $f_i T$ . The projection is conducted in a polygon-by-polygon fashion. First,  $p_i + f_i T$  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  $f_i T$  has been consumed and  $s_i$  has been moved a projected displacement  $d'_1$  to the edge of polygon 1, i.e.,  $p'_i$ . A remaining displacement of  $f_i T - d_1$  need to be applied to  $s_i$ , which is now at  $p'_i$ . Similarly,  $p'_i + f_i T$  is project to polygon 2, we regard that a partial displacement  $d_2$  out of  $f_i T - d_1$  has been consumed and  $s_i$  has been moved a projected displacement  $d'_2$  to the edge of polygon 2.



(a) initial deployment



(b) final deployment

Figure 17: An example of running the VF-based method by game engine Unity.

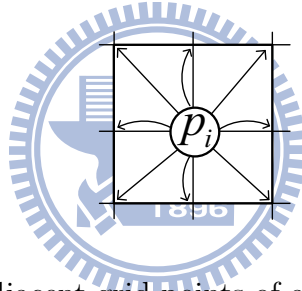


Figure 18: Adjacent grid points of a current location  $p_i$ .

It is worth pointing out that the above iterative process is indeed supported by many 3D game engines, such as Unity [30]. Fig. 17 shows how we configure a 3D object (an arm) by polygons in Unity. Fig. 17(a) shows the initial locations of four sensors on an arm, while Fig. 17(b) shows the final locations found by Unity.

#### 4.3.2 A Metropolis-Based Method

The Metropolis-based method follows a probabilistic search to get rid of local optimal solutions. Its cost, however, is larger 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. Without

loss of generality, we assume that each grid point has eight adjacent grid points (refer to Fig. 18), which may locate on adjacent faces of the surface area. 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. Begin with an 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  $\sigma_e^2(P)$ ) seen so far.

We design the transition probability  $q_{ij}$  according to the Metropolis' theorem [33]. 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

$$\pi(P) = \frac{1/\sigma_e^2(P)}{C}, \quad (4)$$

where  $C = \sum_P 1/\sigma_e^2(P)$ . Note that  $C$  normalizes the distribution. By Metropolis' theorem, this transition probability should be defined as

$$q_{ij} = \begin{cases} \frac{1}{8m} \min\{1, \frac{\pi(P_j)}{\pi(P_i)}\} & \text{if } P_j \in B(P_i), \\ 0 & \text{if } P_j \notin B(P_i), \\ 1 - \sum_{P_j \neq P_i} q_{ij} & \text{if } P_j = P_i. \end{cases} \quad (5)$$

Note that the actual value of  $C$  is immaterial in Eq. (5). Compared to the VF-based method, this algorithm requires much longer time to reach its stationary distribution, but it may output a better solution.

## 5 Simulation and Experimental Results

We have conducted simulations and real experiments to verify performance of the proposed algorithms in terms of  $\sigma_e^2(P)$  and the actual observed gravity.



## 5.1 Simulation Setup

We conduct simulations on five rigid body geometries as shown in Fig. 19. Geometry (a) is a cube with dimensions 4, 4 and 4. Geometry (b) is a rectangular box with dimensions 15 (length), 15 (width), and 30 (height). Geometry (c) is a Triangular Prism, of which the height is 30 and the bases are regular triangles with the length of each edge being 15. Geometry (d) is a cylinder, of which the height is 30 and the circle has a radius of 7.5. Geometry (e) is the graphical model of a human arm in Fig. 7. Since the locations of joints matter, for each geometry, we simulate the possible location of its joint. One is at the geometric center of the rigid body, shown by a circle  $\circ$  in Fig. 19. Another is at one surface of the rigid body, shown by a  $\times$  in Fig. 19. This case is more common in practice. The other is located outside the rigid body, which is not shown in Fig. 19. Since no previous method exists, we compare our VF-based method and our Metropolis-based method, by  $\sigma_e^2(P)$  and execution time.

## 5.2 Comparison of $\sigma_e^2(P)$

First, we consider the deployment of  $m = 4$  sensors. For the VF-based method, sensors are initially placed inside the geometries at random and then moved according to the LD guideline. The algorithm terminates when the total moving distance of sensors is smaller than  $\sigma_{th}$ , which is set close to zero in the simulation. The final locations of sensors are used as the output of the method. For the Metropolis-based method, the surface of each geometry is properly partitioned into grid points. Initially, sensors are placed at the grid points at random. The algorithm moves sensors according to transitional probabilities and terminates in 200,000 iterations. The best locations seen so far are used as the output of the method. To avoid finding only a local minimum, we repeat both methods ten times with randomized initial locations.

We show the error variance  $\sigma_e^2(P)$  achieved by each method. Since the location of the joints of rigid bodies matters, we simulate three possible situations. Fig. 20 shows the results when joints are at the geometric centers of the rigid bodies (indicated by  $\circ$  in Fig. 19), and Fig. 21 shows the results when joints are on the surface of the rigid bodies (indicated by  $\times$  in Fig. 19), and Fig. 22 shows the results when joints are outside the rigid bodies. Each column of the figures shows the distribution of the  $\sigma_e^2(P)$ s output by ten different executions of each method for one rigid body geometry in Fig. 19. It happens

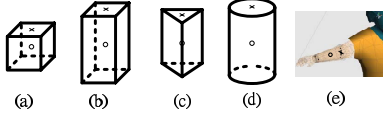


Figure 19: Different geometries

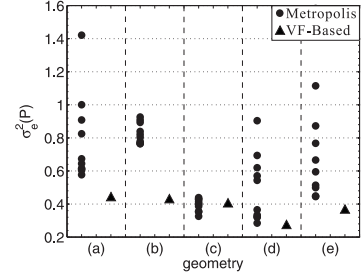


Figure 20: Comparison of different geometries, origin at geometry center

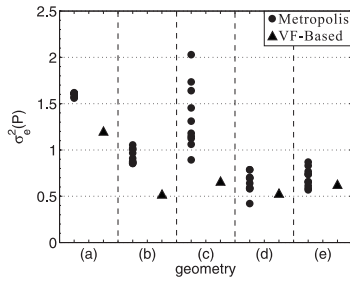


Figure 21: Comparison of different geometries, origin at geometry edge

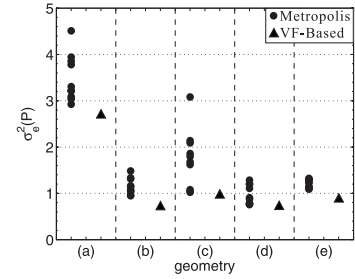
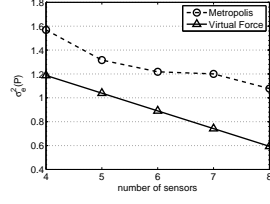


Figure 22: Comparison of different geometries, origin is away from the geometry edge

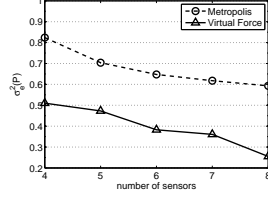
that ten executions of the VF-based method give almost the same  $\sigma_e^2(P)$  although the actual location of sensors may not be the same. As can be seen from Fig. 19, the VF-based method significantly outperforms the Metropolis-based method. For most cases, the VF-based method is able to give lower  $\sigma_e^2(P)$ s than the Metropolis-based method, and for the rest cases, the output of the VF-based method is comparable to that of the Metropolis-based method. While Metropolis-based method requires more trials to obtain better solutions, the VF-based method is able to find good solutions in several executions and less dependent on luck.

### 5.3 Effects of More Sensors and Execution Time

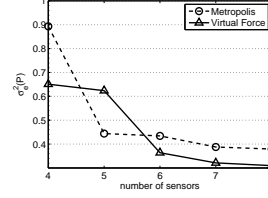
The number  $m$  of accelerometers to be deployed on a rigid part is a trade-off between accuracy and cost. In addition to the  $m = 4$  case, we consider different number of sensors to be deployed on the geometries in Fig. 19. The joints are at the surfaces of rigid bodies (indicated by  $\times$  in Fig. 19). For each value of  $m$ , we repeat the VF-based and the



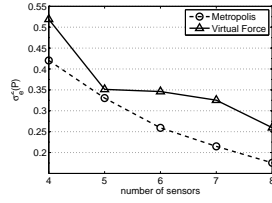
(a)



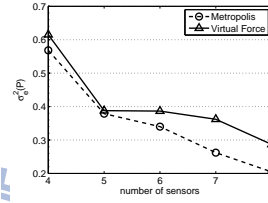
(b)



(c)



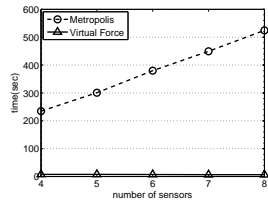
(d)



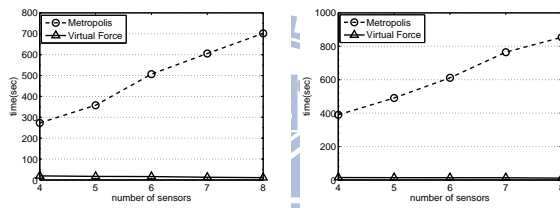
(e)

Figure 23: Effects of  $m$  sensors.

Metropolis-based method ten times and compare them by their optimal  $\sigma_e^2(P)$ s. Note that comparing the  $\sigma_e^2(P)$ s is more practical. For each geometry in Fig. 19, we show the comparison of both methods by  $\sigma_e^2(P)$ s in Fig. 23 and by the execution time in Fig. 24. The results of  $m = 4$  in Fig. 23 is also shown in Fig. 21. For both methods, deploying more sensors significantly reduces the error variance  $\sigma_e^2(P)$ . The relative accuracies between the two methods are roughly the same as  $m$  is increased. The execution time of the VF-based method increases quite slowly as more sensors are used, whereas the execution time of Metropolis-based method is quite sensitive to  $m$ . The Metropolis-based method takes at least 200 seconds to reach mathematical stationary distribution  $\pi(P)$ , whereas the VF-based method reduces the execution time by more than 1/8. The results demonstrate that the proposed LD guideline is both effective and efficient for deploying accelerometers on arbitrary geometries.

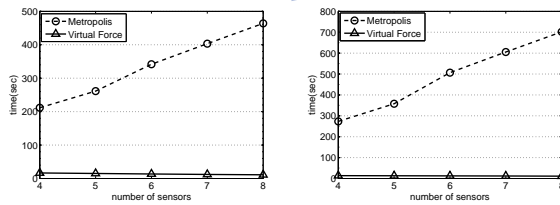


(a)



(b)

(c)



(d)

(e)

Figure 24: Comparison of running times.

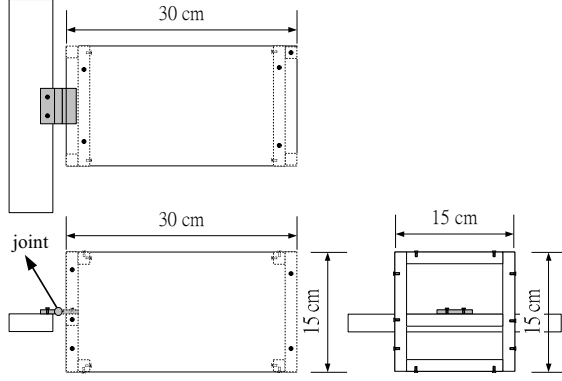


Figure 25: Three view of the box.



Figure 26: Rotating the box up and down.

## 5.4 Actual Gravity Estimation

In addition to simulations, we conduct real experiments to test the accuracy of the measured gravity. We make a rectangular box, deploy accelerometers on its surface, and then rotate the box physically to test the measured gravity. As shown in Fig. 26, we consider a rigid body of a rectangular box fixed by a joint on a table. The three views of the box and the joint are shown in Fig. 25, where the joint has one rotation degree of freedom. We deploy eight accelerometers at the degree of freedom. We deploy eight accelerometers at the vertices of the box. Their coordinates are shown in Fig. 26. Note that the coordinate of the joint is defined to be  $(0, 0, 0)$ . We consider two motions, one consisting of fast rotations repeatedly from Fig. 26(a) to Fig. 26(b), and the other consisting of slow rotations. The sensing data is stored at each sensor node and reported to a sink in a reliable way. We compute the measured gravity offline by a PC.

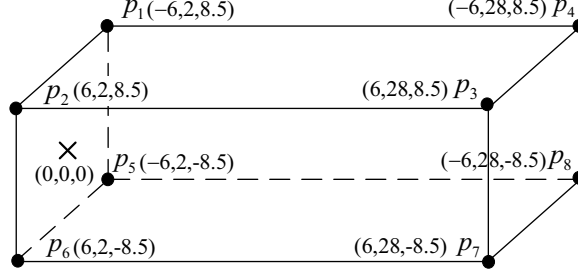


Figure 27: Sensors' locations on the box in Fig. 26.

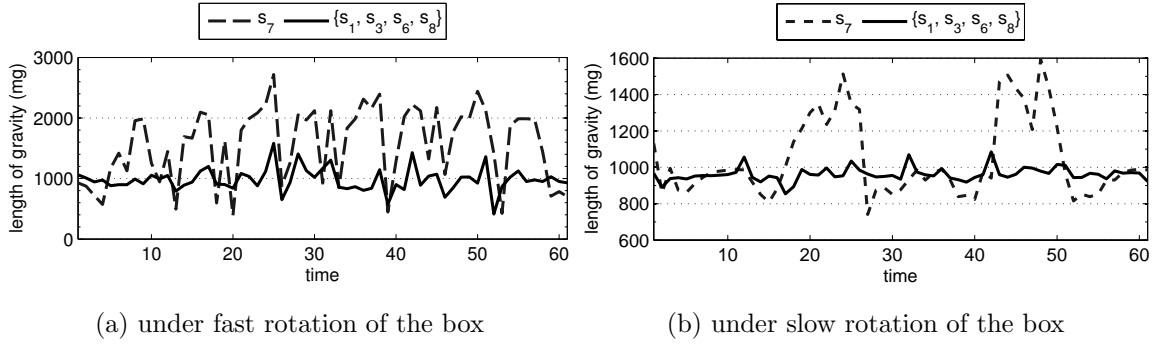


Figure 28: Comparison of length waveforms of measured gravity vectors.

We first compare the accuracy of using four sensors ( $m = 4$ ) with only one sensor ( $m = 1$ ). The four sensors are at  $p_1$ ,  $p_3$ ,  $p_6$ , and  $p_8$ , the optimal locations obtained by the VF-based method. The estimated gravity  $\hat{g}(t)$  at time  $t$  is the first column of its  $\hat{Q}(t)$ . We compare it with the raw sensing data of  $s_7$ . Since the actual gravity  $R^T(t)g$  is unknown, we compare by the lengths of measured gravities, taking advantage of the fact that  $\|R^T(t)g\|$  is always 1000 mg (milliguass), no matter how we rotate the box. Fig. 28 shows the length waveforms of measured gravity vectors under fast rotation (Fig. 28 (a)) and slow rotation (Fig. 28 (b)) of the box. Clearly, the gravity measured measurement by a single accelerometer is significantly disturbed by motion and noise, whereas the gravity measured by four sensors are much more accurate. This verifies the need of our multi-sensor approach.

Now, we compare different deployment by their measurement accuracy to verify the need of proper deployments. The accuracy of the measured gravity is computed as the standard deviation of measurement errors  $\|\hat{g}(t)\| - \|R^T(t)g\|$ , which intuitively is the average distance between a length waveform similar to Fig. 28 and a flat line of 1000 mg. We call such a metric the standard error of a deployment. Below, we compare five deployments.

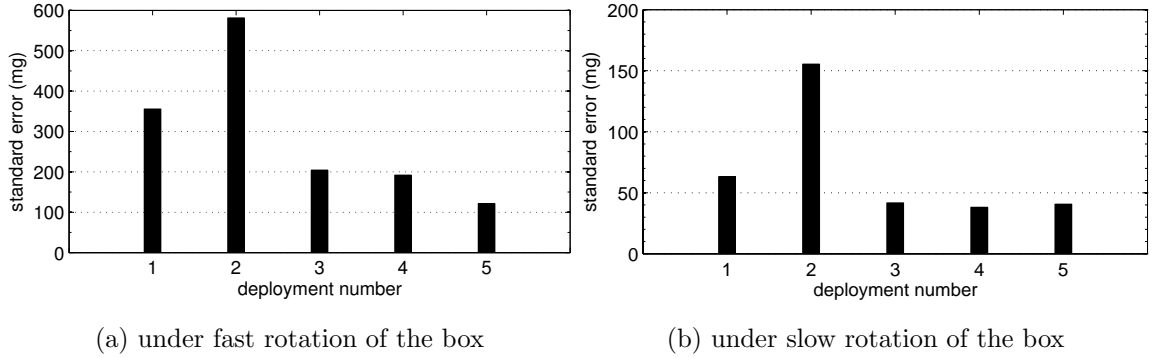


Figure 29: Comparison of five deployments.

1. Using one sensor at the position  $p_6$ .
2. Using four sensors at the positions  $p_3$ ,  $p_4$ ,  $p_5$ , and  $p_7$ .
3. Using four sensors at the positions  $p_1$ ,  $p_2$ ,  $p_6$ , and  $p_8$ .
4. Using four sensors at the positions  $p_1$ ,  $p_3$ ,  $p_6$ , and  $p_8$ .
5. Using all eight sensors.

The results are shown in Fig. 29. Deployment 1 (where  $m = 1$ ) is a reference for comparing multi-sensor deployments. We show the standard errors of all one-sensor deployments in Fig. 30. Clearly, when  $m = 1$ , placing the sensor at  $p_6$  would be a good choice. Deployment 2 shows a situation where an improper four-sensor deployment can be worse than a proper one-sensor deployment. Deployment 3 shows the results obtained by time-consuming trials and errors for  $m = 4$ . Deployment 4 and 5 are the results obtained by the VF-based method for  $m = 4$  and  $m = 8$ , respectively. The results verify the effectiveness of our VF-based method and show that it is able to find near-optimal solutions systematically, which is important for scalability.

## 6 Conclusions

In this paper, we develop a video game integrated with body-area inertial sensor networks, the player can interact with the video game simpler and play the video game with a 360° panorama view of the game scene. In the prototype of our Multi-Screen Cyber-Physical Video Game, we found a basic BISN problem, gravity estimation problem. We have investigated how to perceive gravity by deploying multiple accelerometers on a rigid part.

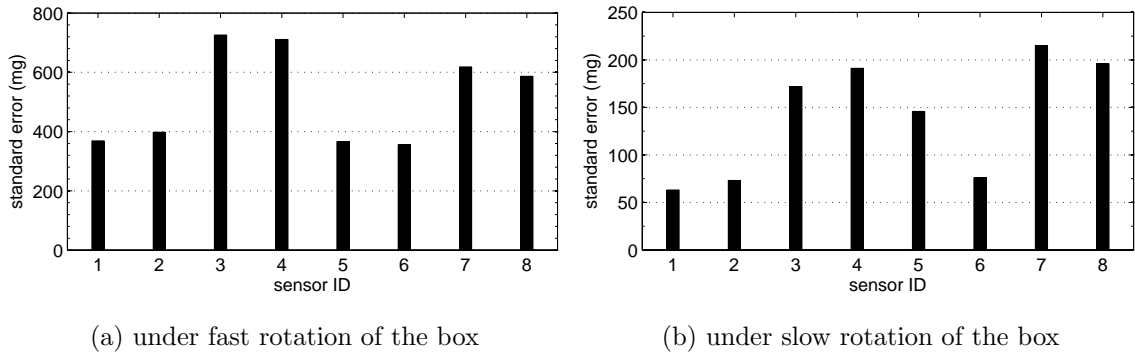



Figure 30: Comparison of standard errors of one sensor deployments.

We have modeled the readings of an accelerometer in terms of the gravity and the sensor’s own acceleration seen by an outside observer. Using the pseudoinverse scheme to extract the gravity from these readings, we have formulated the deployment optimization problem as one to find a deployment that minimizes the measurement errors. We have proposed deployment guidelines and two heuristics. Experiment results also verify the effectiveness of our methods.

## References

- 
- [1] Nintendo, “Wii,” <http://wii.com>, 2008.
  - [2] Microsoft, “XBOX360 Kinect,” <http://www.xbox.com/zh-TW/kinect>, 2011.
  - [3] SONY, “Playstation Move,” <http://asia.playstation.com/move/tw/>, 2010.
  - [4] D. Vlastic, 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.
  - [5] Panasonic, “KXP84 SERIES – Triaxial Accelerometer,” <http://www.kionix.com>, 2006.
  - [6] H. Kim and D. W. Fellner, “Interaction with hand gesture for a back-projection wall,” in *Computer Graphics International Conference*, 2004.
  - [7] 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.



- [8] 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.
- [9] 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.
- [10] 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.
- [11] C. M. Sadler and M. Martonosi, “Data compression algorithms for energy-constrained devices in delay tolerant networks,” in *Proc. of ACM Int’l Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.
- [12] 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.
- [13] I. Lee and O. Sokolsky, “Medical cyber physical systems.”
- [14] S. G. Ying Tan, Mehmet C. Vuran, “Spatio-temporal event model for cyber-physical systems,” in *29*.
- [15] G. X. . W. J. . Y. D. . P. T. . M. S. . X. Liu, “Toward ubiquitous video-based cyber-physical systems.”
- [16] Y. Kleissl, J. ; Agarwal, “Cyber-physical energy systems: Focus on smart buildings.”
- [17] R. Akella and B. M. McMillin, “Model-checking bndc properties in cyber-physical systems,” in *33*.
- [18] S. . H. L. . C. K. Wang, E.K. ; Yunming Ye ; Xiaofei Xu ; Yiu, “Security issues and challenges for cyber physical system.”

- [19] 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.
- [20] 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.
- [21] Z. Zhang, “Ubiquitous human motion capture using wearable micro-sensors,” in *Proc. of Int’l Conf. on Pervasive Comput. and Commun. (PerCom)*, 2009.
- [22] 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.
- [23] D. Vissière, A. Martin, , and N. Petit, “Using distributed magnetometers to increase IMU-based velocity estimation in perturbed areas,” in *Proc. of IEEE Int’l Conf. on Decision and Control*, 2007.
- [24] J. K. Rowling, “Quidditch,” <http://en.wikipedia.org/wiki/Quidditch>.
- [25] Jennic, “JN5139,” <http://www.jennic.com>.
- [26] OceanServer, “OS5000 Family – Triaxial Accelerometer,” <http://www.ocean-server.com>, 2008.
- [27] J. Carig, *Introduction to Robotics*. Prentice Hall, New Jersey, 2005.
- [28] 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.
- [29] Gramschmidt process. Wikipedia. [Online]. Available: [http://en.wikipedia.org/wiki/Gram-Schmidt\\_process](http://en.wikipedia.org/wiki/Gram-Schmidt_process)
- [30] Unity – 3D Game Engine. Unity. [Online]. Available: <http://unity3d.com>
- [31] 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.

- [32] G. Wang, G. Cao, and T. F. La Porta, “Movement-assisted sensor deployment,” vol. 5, no. 6, pp. 640–652, 2006.
- [33] S. Chib and E. Greenberg, “Understanding the metropolis-hastings algorithm,” *American Statistician*, vol. 49, no. 4, pp. 327–335, 1995.

