Embedded and visual programming for SmartSuit motion capture system

Nguyen, Kim Doang

2009

Nguyen, K. D. (2009). Embedded and visual programming for SmartSuit motion capture system. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/20927>

<https://doi.org/10.32657/10356/20927>

Embedded and Visual Programming for SmartSuit Motion Capture System

SUBMITTED

BY

NGUYEN KIM DOANG

(G0702312L)

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

A thesis research project report
presented to the
Nanyang Technological University
in partial fulfillment of the
requirements for the
Degree of Master of Engineering
Nanyang Technological University

Year (2009)

Acknowledgement

I would like to express my sincere thanks to my supervisor, Professor Chen I-Ming, for his patient and kind training, guidance and motivation by his plentiful knowledge and experiences. His expertise in technical and research background of this research has immensely contributed to the completion of this report.

I am also deeply grateful to Professor Yeo Song Huat and Professor Duh Been Lirn. Their ever encouraging support, guidance, patience and valuable advices effectively orient my research to right way.

I honestly appreciate the aids of Lim Kwang Yong, Dong Wei and Goh Young Koon, Dr. Luo Zhiqiang, Dr. Lim Chee Kian, Li Kang, Ni Wei, Gu Chao in the beneficial studies and research work on the optical linear encoder, IMU and mechanical design. Their roles are important in guiding me in the fulfillment of this research project.

I would like to thank: Mr Lim Eng Cheng, Technician-in-charge of Robotics Research Centre, for his coordination of the available resources, Ms Agnes Tan for her guidance, support and valuable advice; Mr You Kim San and Miss Toh Yen Mei for their valuable advices and assistance in the test process.

Finally, to friends who have contributed in one way or another, thank you for your continuous support and endless encouragement during this time.

Table of Content

ACKNOWLEDGEMENT.....	II
TABLE OF CONTENT.....	III
LIST OF FIGURES	VII
ABSTRACT.....	XI
CHAPTER 1 INTRODUCTION.....	1
1.1 MOTION CAPTURE SYSTEMS.....	1
1.2 APPLICATIONS	2
1.3 CLASSIFICATION	3
1.4 MOTION CAPTURE DATA FLOW	7
1.5 IDEAL MOTION TRACKING SYSTEM	9
1.6 MOTIVATION.....	10
1.7 OBJECTIVES AND SCOPE OF THE THESIS	12
1.8 REPORT OUTLINE.....	15
CHAPTER 2 BACKGROUND ON SMARTSUIT SENSING MODULE.....	16
2.1 OPTICAL LINEAR ENCODER	16
2.1.1 Working principle	16
2.1.2 Mapping from linear information to joint angle.....	17
2.2 SENSOR ELECTRONICS	19
2.3 MECHANICAL DESIGN.....	20
2.4 PLACEMENT OF THE SENSING MODULES ON HUMAN ARM.....	21
2.4.1 Shoulder joint.....	22
2.4.2 Elbow joint.....	22
2.4.3 Wrist joint.....	22
2.5 SENSOR PACKAGE DESIGN	23
2.5.1 Wrist module	24
2.5.2 Elbow module.....	25

2.5.3	<i>Shoulder module</i>	25
2.6	SUMMARY	26
CHAPTER 3 DESIGN OF MOTION SENSOR NETWORK		27
3.1	SELECTION OF SENSOR NETWORK TOPOLOGY	27
3.1.1	<i>Point-to-point topology</i>	29
3.1.2	<i>Ring topology</i>	30
3.1.3	<i>Star topology</i>	31
3.1.4	<i>Mesh topology</i>	32
3.1.5	<i>Hierarchical topology</i>	33
3.1.6	<i>Bus topology</i>	34
3.1.7	<i>Discussion on the selection of the network topology</i>	36
3.2	SMARTSUIT NETWORK ARCHITECTURE	39
3.3	ACCELEROMETER	43
3.4	FIRMWARE PROGRAMMING.....	46
3.4.1	<i>Node process</i>	48
3.4.2	<i>Data Concentrator process</i>	48
3.5	SUMMARY	50
CHAPTER 4 MOTION PROCESSING SOFTWARE FOR SMARTSUIT		51
4.1	MOTION DATA PROCESSING UNIT	53
4.1.1	<i>Serial ports</i>	53
4.1.2	<i>Protocol</i>	54
4.1.3	<i>Programming</i>	55
4.1.4	<i>Conversion from raw data to angles</i>	56
4.2	CALIBRATION.....	57
4.3	HUMAN ARM KINEMATIC MODEL	60
4.3.1	<i>Establishment of D-H coordinates and parameters for the human arm</i>	62
4.3.2	<i>Joint angle mapping from sensor data</i>	64
4.4	HUMAN MOTION DATA STRUCTURE – A HIERARCHICAL MODEL	68
4.5	GRAPHICS DISPLAY	72
4.5.1	<i>Rendering graphics using OpenGL</i>	72

4.5.2	<i>Interacting with modeling softwares.....</i>	74
4.6	MOTION CAPTURE DATA FORMATS	75
4.6.1	<i>BioVision Hierarchy (BVA/BVH)</i>	77
4.6.2	<i>Acclaim ASF/AMC</i>	79
4.7	OPERATING FLOW	81
4.8	SUMMARY	86
CHAPTER 5 VALIDATION OF SMARTSUIT AS A SENSING MODULE.....		87
5.1	REPEATABILITY AND ACCURACY TEST OF OLE	87
5.1.1	<i>Set-Up of PowerCube Comparison Experiment</i>	87
5.1.2	<i>Results from rigid link experiment.....</i>	88
5.2	IN-SITU TEST OF OLE SENSORS.....	89
5.2.1	<i>Set-up of Comparison Experiment.....</i>	89
5.2.2	<i>Results of Comparison Experiment.....</i>	92
5.3	SUMMARY	95
CHAPTER 6 VALIDATION OF SMARTSUIT AS A CAPTURE SYSTEM		96
6.1	EXPERIMENTAL SET-UP	97
6.2	VALIDATION PROCEDURES	97
6.3	ANALYSIS	99
6.3.1	<i>Repeatability testing.....</i>	100
6.3.2	<i>Reliability analysis.....</i>	101
6.4	RESULTS	103
6.4.1	<i>Repeatability</i>	103
6.4.2	<i>Reliability.....</i>	104
6.5	DISCUSSION	105
CHAPTER 7 CONCLUSION		106
1)	<i>Body sensor network for upper extremity motion capture.....</i>	107
2)	<i>Kinematics model.....</i>	108
3)	<i>SmartSuit motion processing software.....</i>	108
7.1	CONTRIBUTIONS OF THE THESIS	110

7.2 FUTURE WORKS.....	112
7.2.1 Extension of the body sensor network to entire human body.....	112
7.2.2 Applications development	113
REFERENCES.....	114
LIST OF PUBLICATIONS	117
APPENDIX A	A1
APPENDIX B	A6
APPENDIX C	A9
APPENDIX D	A11

List of Figures

Figure 1-1 Gypsy 6 developed by <i>Animazoo Ltd</i> [7]	3
Figure 1-2 PHANTOM developed by <i>SensAble Technologies, Inc.</i> [8]	4
Figure 1-3 HiBall Optical Sensor and the HiBall Ceiling Beacon Arrays [10]	4
Figure 1-4 Eight-camera Vicon MX40 motion capture system [12]	5
Figure 1-5 IGS-190 invented by <i>Animazoo Ltd</i> [7]	6
Figure 1-6 ShapeWrape created by Measurand [16]	7
Figure 1-7 Data flow for motion capture systems	8
Figure 2-1 Channel A and B 90° out of phase to indicate forward or backward traverse and block diagram of linear encoder sensor [18]	17
Figure 2-2 OLE working principle - conversion of displacement to angle	18
Figure 2-3 Example of the data processing on the actual body motion	18
Figure 2-4 Block diagram of the sensing module	20
Figure 2-5 Components of the sensing module prototype	20
Figure 2-6 (a) CAD drawing of the base structure; (b) Actual base structure	21
Figure 2-7 (a) CAD drawing of the slider; (b) Actual slider	21
Figure 2-8 Placement of the OLEs on shoulder, elbow and wrist joints	23
Figure 2-9 (a) The wrist module: The tip of strip is locked by a PVC Connector which is shown at the down side of this figure. The sensor, the connector and the tip will be covered by the fabric. (b) Plane view of the wrist suit on the human arm.	24
Figure 2-10 Elbow module: (a) the prototype of the elbow module; (b) the module worn on the elbow	25
Figure 2-11 (a) Should module prototype; (b) Side view of the module worn on the shoulder	25
Figure 3-1 Layout of the 15 sensor nodes and the Data Concentrator on a human body.	28
Figure 3-2 (a) Permanent point-to-point topology; (b) Permanent line topology; (c) Switched point-to-point topology; (d) Switched line topology	30
Figure 3-3 Ring topology	31
Figure 3-4 Star topology	31

Figure 3-5 Two types of mesh network topology: (a) Full connected mesh; (b) Partially connected mesh.....	33
Figure 3-6 Hierarchical topology.....	34
Figure 3-7 Data flow of bus topology (a), partially connected mesh (b) and hierarchical topology (c).....	35
Figure 3-8 (a) Linear bus topology; (b) Distributed bus topology.....	36
Figure 3-9 The system of 15 sensor nodes is connected by distributed bus topology.....	37
Figure 3-10 Motion capture sensor network block diagram.....	39
Figure 3-11 Data Concentrator	40
Figure 3-12 Actual SmartSuit sensor network.....	41
Figure 3-13 block diagram for wireless communication	42
Figure 3-14 Pitch (θ) and roll (φ) orientation of an accelerometer.....	43
Figure 3-15 Coordinate frames (body frame and world frame).....	45
Figure 3-16 Flow chart of the firmware for a Data Concentrator and three sensor nodes for arm motion capture.....	47
Figure 4-1 The Software architecture for human motion capture system	52
Figure 4-2 Flow chart of the procedure for acquiring data through serial port	55
Figure 4-3 The user is posing a calibration gesture for the OLE at the elbow joint; The white board is marked with the angle that the forearm is supposed to pose, while the upper arm is kept vertical.....	58
Figure 4-4 Sample calibration curve for capturing elbow joint motions	59
Figure 4-5 Comparison between calibrated data and data without calibration.....	60
Figure 4-6 Denavit-Hartenberg coordinate system for human arm	61
Figure 4-7 Position of the sensing modules on the human arm.....	61
Figure 4-8 Biometric parameters	64
Figure 4-9 Absolute angles θ_{w0} and θ_{w1} are given by the accelerometer mounted on the user's upper arm.....	64
Figure 4-10 Elbow joint angle between the upper and forearm, θ_3 , provided by the OLE of node 2	65
Figure 4-11 absolute angles θ_{w4} and θ_{w6} are tracked by the accelerometer attached on the hand.....	65

Figure 4-12 Range of motion for human shoulder [23]	67
Figure 4-13 Range of motion for human elbow [23]	67
Figure 4-14 Range of motion for human wrist [23]	67
Figure 4-15 Human link model (refer to Figure 4-16 for the complete hierarchical link model)	68
Figure 4-16 Hierarchical human link model	70
Figure 4-17 Hierarchical human joint model	71
Figure 4-18 Algorithm of rendering virtual human model using OpenGL	73
Figure 4-19 Resulting virtual avatar rendered using the proposed algorithm	74
Figure 4-20 Sample segment of the BVH file	78
Figure 4-21 Sample segment of *.ASF and *.AMC	79
Figure 4-22 GUI of the Motion Capture Viewer	81
Figure 4-23 Flowchart for the software's main operating flow	82
Figure 4-24 SmartSuit graphical user interface	85
Figure 4-25 (a) Data gauges; (b) Data graphs	85
Figure 5-1 Set-Up of PowerCube Comparison Experiment	88
Figure 5-2 Graph showing PowerCube Reading against Sensor Reading	88
Figure 5-3 Set-Up of Comparison Experiment for BIOPAC Goniometer and OLE	90
Figure 5-4 Graph showing Goniometer Reading against Sensor Reading	91
Figure 5-5 Flowchart of Experimental Procedure	92
Figure 5-6 Slow Flexion Motion	93
Figure 5-7 Fast flexion motion.	94
Figure 6-1 Experiment set-up for repeatability and reliability tests for arm motion capture	96
Figure 6-2 Arm movement testing procedure:	98
Figure 6-3 Arm movement testing procedure (continued):	99
Figure 6-4 Sample data from each joint for a single data block (one subject) collected during the performance of the arm-reaching task. Angles (the vertical dimension) are in degrees. Time (the horizontal dimension) is in seconds. When the arm lies horizontally, the degree of the shoulder joint is 0 degree. When the upper arm, forearm and hand lie in	

one line, the angles of both elbow and wrist are 0 degrees. Wrist roll is the angle of the hand with respect to the forearm.....	102
Figure 6-5. Individual data block averages for repeatability. Each value is the average of 10 cycles of the arm-reaching task for each sensor per subject.	103
Figure 6-6 Average range and standard deviation (SD) for each subject.	104
Figure A. 1 Actual schematic of a serial port	A1
Figure A. 2 Actual EHSII wireless node	A5
Figure A. 3 Detailed block diagram of the EHSII wireless node	A5

Abstract

This thesis presents a novel sensing technology using optical linear encoders (OLE) to capture human motion. A CAN bus architecture is proposed to connect the OLE-based sensor nodes, each of which consists of a tri-axis accelerometer and an OLE. The network of three SmartSuit sensing modules is able to capture full motion of human arm with a 7-DOF kinematic model. Firmware programs are developed and embedded into wearable sensor nodes to implement the architecture. In addition, the programming framework for motion capture and processing is introduced. Based on the framework, the motion capture software is developed with all necessary features. The software interfaces with the wearable sensor hardware via serial communication. The motion data are processed and stored in hierarchical models. The software's graphics display unit regenerates the body motion using either OpenGL rendering method or modeling softwares. Moreover, the software can import and export standard motion data formats, facilitating our OLE-based motion capture system to communicate with various platforms. Experiments were carried out to compare the performance of the OLE sensing module with BIOPAC Goniometer. The results show that the OLE's performance is comparable to that of those expensive systems, and also validate the sensor network architecture, firmware and the SmartSuit software. Furthermore, a statistical study is conducted to confirm the repeatability and reliability of the new OLE sensing module and the wearable sensor network. The results demonstrate that the new sensor system has strong potential to be used as a low-cost tool for motion capture, and arm function evaluation for short-term as well as long-term monitoring.

Keywords: Motion capture, wearable sensors, sensor networks, software architecture.

Chapter 1 Introduction

1.1 Motion Capture Systems

As computers are becoming more widespread in the society, there is the need for more advanced interfaces. This has led to a new research field known as Human-Computer Interaction. One problem with current interfaces is that the communication is mainly done via devices non-intuitive to humans, e.g. mice and keyboards. The communication is on the terms of the computer rather than natural terms such as speech and body languages. Since it is hard for human to learn the language of computers, the solution must be to develop computer systems which are able to communicate with human beings whose knowledge about computer languages is humble. This means computers should be equipped with devices providing the ability to interact in a human-centered manner. These devices could be: sensory or vision systems to enable them to see, microphones to enable them to hear, loudspeaker to enable them to speak, etc. Technologies providing speech and hearing abilities have been leaving labs and entering the market. The observing ability, however, is still lacking due to the high complexity involved. This has led to a relatively new research field, known as human motion capture, which equips users the ability to study methods for accurately describing human limb movement, postures and gestures in spatial and temporal manner.

Motion capture is the process of capturing human body motion, via an external sensing device. However, a closer investigation reveals that this is not entirely true. Many external devices, e.g. a computer mouse, a steering wheel in a car, are usually not considered as motion capture devices, and their main tasks are not considered to be motion capture, although they do capture human motion. Generally, a device, which is associated with recording or storing of human motion information, can be considered a motion capture device. The development in motion capture technologies comes from two sources: research and industry. Initially motion capture applications were designed to measure, and thereby help understand, the motion patterns of human beings and animals. Those studies were solely driven by research. Later medical advances as well as entertainment applications set up the active market of motion capture products.

1.2 Applications

Human motion capture has a wide range of applications. In medicine, computers can use motion capture data to analyze patients' movements for rehabilitative purposes [1], such as examining the recovery of the patients from strokes or accidents. Motion analysis can help sportsmen locate their weaknesses, and improve their performances. The film industry is also a pioneer in application of motion capture since the arising of realistic computer graphics. The capture and re-targeting of the movements of actors to animated characters is a very important application, which is used not only in movies, but also in video games and in live broadcasts. Alternatively, computer-generated human figures need realistic animation, which is rather difficult to achieve manually using merely modeling programs. By utilizing recent developments in statistical learning approaches, it is possible to model and generate stylized motions [2]. However, these techniques still necessitate large amounts of training data, which are usually acquired through motion capture. The automatic tracking of human motions is significant for surveillance and security. Computerized systems can detect suspicious patterns of behaviors [3] and trigger alerts. Computers could become virtual teachers in activities such as dancing, yoga or sign languages, capable of both instructing students and correcting their errors.

Finally, motion capture finds exciting applications in smart offices or households [4]-[6], where computers try to understand the intentions of human beings. Movements and gestures are essential means of communication, and recognizing them is a milestone towards “human-aware” buildings. More generally, gestures become an essential way to interact with computers in a more natural and expressive manner than current computer-centered devices. The entire architecture of computer interfaces may be defined by gesture-based interactions. The interfaces, combined with speech, could allow users to interact freely with virtual objects. Video games are an obvious example of application that would greatly benefit from body tracking to enhance the immersion of the player. Likewise, tracking motions can be used to control realistic avatars in virtual environment. Social interactions would then be possible without the barrier of distance.

1.3 Classification

A wide range of motion-tracking technologies have been developed for human motion capture in entertainment and biomedical applications, including mechanical capture system, optical tracking systems, acoustic, inertial/magnetic tracking systems, hybrid and markerless tracking system. Most are dependent on an artificially generated source, thus susceptible to interference and noise, and require restrictive workspace.

Figure 1-1 Gypsy 6 developed by Animazoo Ltd [7]

Mechanical capture systems can be placed in two separate categories, **body-based** and **ground-based**. **Body-based** systems, such as *Gypsy6* from *Animazoo Ltd* (Figure 1-1), use an **exoskeleton** that is attached to the articulated structure to be tracked. Rigid or flexible goniometers are utilized to measure the joint angle of user's body structure. The attachment of the body-based mechanism and the positioning of the goniometers raise some critical concerns. The exoskeleton may move relatively to the user's body during its motion due to body skin movements. In addition, the alignment of the goniometer with body joints is seriously difficult. These problems can dramatically reduce the accuracy of exoskeletons in motion capture, and make its performance vary from person to person. **Ground-based** systems attach one end of a boom or shaft to a tracked object and typically have six degrees of freedom (DOF). Their advantage is the ability to provide haptic feedback. For instance, *PHANTOM* from *SensAble Technologies, Inc.*, shown in Figure 1-2, allows users to explore applications requiring force feedback. The 2-DOF joints in the device have a motion range of

lower arm movement about the elbow. The 3-DOF joint has a motion range of full arm movement about the shoulder.

Figure 1-2 PHANTOM developed by SensAble Technologies, Inc. [8]

Optical tracking systems can also be categorized into two types, **Pattern recognition** and **Image processing**. **Pattern recognition** systems sense a pattern of lights and use this data to regenerate position and/or orientation. This kind of capture system can be “outside-in” when the sensors are attached in the ground and the emitters are on the moving objects; or “inside-out” when sensors are installed on the objects and the emitters are fixed. **HiBall-3100 Tracker** uses this technology to record motions. It includes two main components: HiBall Optical Sensor and the HiBall Ceiling Beacon Arrays (Figure 1-3). The HiBall Optical Sensor has 6 lenses and photodiodes in which each photodiode is able to ‘view’ infrared LEDs in the Beacon Arrays mounted on the ceiling, through several of the 6 lenses. By positioning the sensors on a mobile object, inside-out tracking, the system is capable of capturing the object’s motions [9].

Figure 1-3 HiBall Optical Sensor and the HiBall Ceiling Beacon Arrays [10]

Image processing motion capture systems track motion of objects by using several cameras to track predefined points on mobile objects within a constrained volume [11]. The tracked

points may be marked actively or passively. Two major companies, *Qualisys* and *Vicon*, use this camera-marker principle of motion tracking. In their systems, common tracking points on the tracked objects in each 2D image are correlated utilizing stereometric technique. This data and a model relating each of the images and camera parameters are then used to regenerate the user motion. The markers can be either active (light emitting) or passive (reflective). In an active optical tracking system (*Qualisys*), the camera record the infrared light emitted by the LED's attached to the user. Passive systems (*Vicon*) consist of infrared LEDs installed around camera lens and infrared-pass filters located above the camera lens. The camera measure the light reflected from the markers to produce the object's motion. Figure 1-4 shows a capture session using a camera-marker system.

Figure 1-4 Eight-camera Vicon MX40 motion capture system [12]

Magnetic tracking systems record both position and orientation by using sets of small perpendicularly mounted coils to sense sequentially generated low-frequency magnetic fields. Position is measured by the changes in net strength across the coils, which is proportional to the distance from the field transmitter. The sequentially emitted fields induce currents in each of the perpendicular coils, allowing measurement of orientation. The emergence of MEMS technology allows inventors in motion capture to manufacture tiny inertial/magnetic sensors. The low-cost and small-size sensors make it possible to produce well designed suits which are embedded with the whole motion capture system. Attachment of such sensor modules to each of the major limb segments of a human allows us to independently determine the orientation of each segment relative to a world reference frame.

The human model is constructed from multiple independently oriented limb segments constrained by their attachment to each other [13]. In *Moven* system from *Xsens Technologies*, 16 inertial motion trackers give absolute orientation estimates which can also be used to calculate the 3D linear accelerations in the world coordinate in turn providing translation estimates of the body segments. *IGS-190* from *Animazoo Ltd* may be another good example of this motion capture technology (Figure 1-5). It consists of 18 inertial sensors attached to a flexible lycra suit to record motions of the user.

Figure 1-5 IGS-190 invented by *Animazoo Ltd* [7]

Ultrasonic tracking systems can determine position through either time-of-flight and triangulation or phase-coherence [14]. Users carry an ultrasonic microphone or tracker that is connected to a radio transmitter. The microphone acquires ultrasonic pulses sent from four speakers located along the edges of the performance space and radios them back to a computer. Based on the time between each pulse being sent and its arrival at the microphone, the computer can calculate the microphone's 3D position.

Fiber-optic systems, such as *ShapeWrap* from *Measurand Inc.*, compare light between two ends of an optical fiber to estimate angles [15]. *ShapeWrap* is composed of four fiber-optic sensor strips and three inertial/magnetic sensors, as depicted in Figure 1-6. The four sensor strips are positioned along the two arm and the two legs to measure the translation as well as orientation of the limbs' links. The three inertial sensors are attached to the head, thorax and pelvis to record the real-time orientation of these points. Those sensors together with a human body model make it possible to reproduce the motions of the user in real-time for analysis. Nevertheless, similar to exoskeleton, this system is hard to wear, and require tedious calibration to obtain acceptable results.

Figure 1-6 ShapeWrape created by Measurand [16]

1.4 Motion Capture Data Flow

Most motion capture systems' pipelines have the similar data flow, which can be depicted in Figure 1-7. Basically data flow from the end hardware which directly records human motion through medium components, then into the motion capture software inside the host computer.

Sensor nodes are the first part which records the raw motion data. These components are normally closest to the human body. For instances, in the optical motion capture system, each marker-camera set is a sensor node, or in the exoskeleton device each goniometer is a sensor node, directly measuring the subject motion.

Figure 1-7 Data flow for motion capture systems

The raw data is collected by signal processing units, which are often integrated with the sensor nodes. For example, in ShapeWrap motion capture system of Measurand, each fiber-optic sensor comes with a processing unit at the end of the tape. In the optical tracking system, the signal processors are integrated in the camera units.

All data will be fed into a Data Concentrator (could be many of the data are bulky), whose functions mainly are to collect all the motion data and interface with the host PC. The communication gateway can be serial ports, USB, LAN cable, Bluetooth, wireless antenna etc.

Every motion capture system comes with its processing software, e.g. ShapeRecorder of Measurand, Motion Tracker Manager of Xsens, and BodyBuilder of Vicon. The software

receives data from the Data Concentrator through computer communication ports. The motion capture software is the main tool for user to interact with the hardware system. The software provides the ability for system calibration, homing and data filtering. The software also compiles the kinematics information, demonstrates the captured motion, stores data for further analysis, or performs some post-processing task such as motion editing, adapting, matching, and other applications.

Another source of data is from subject model acquisition. In this process, human body characteristics are identified. For example, body size, bone lengths, joint diameters are all measured. These parameters are very important to regenerate human motions. The alternatives of the accompanying motion capture softwares are those graphic or animation softwares, which support motion capture feature. Some dominant ones are MotionBuilder, Maya, 3ds Max, and RenderMan. In this project, the data flow will be constructed in the similar way as most of the current motion capture systems follow.

1.5 Ideal Motion Tracking System

Through the review of the motion sensory technologies, it is noted that most motion tracking systems offer relatively impressive performance under some circumstances but exhibits limitations in other applications. Users often long for a system that overcomes the shortcomings related to their particular circumstances. Typical desires are reduced infrastructure, improved robustness, and reduced latency. An ideal motion tracking system, “tracker-on-a-chip” (ToC), was conceptually proposed by Welch and Foxlin [17]. For example, a ToC may integrate an accelerometer, a magnetometer, a gyroscope and a microcontroller all together in one tiny chip, which may be the next generation of an Inertial/Magnetic Unit (IMU). A ToC is the basic element, a set of which forms the ideal motion capture system. According to Welch and Foxlin, the ideal motion capture device, or the ToC, would satisfy everyone if its characteristics are as follows:

- (1) **Tiny** - the size of an 8-pin DIP (dual in-line package) or even a transistor;
- (2) **Self-contained** - with no other parts to be mounted in the environment or on the user;
- (3) **Complete** - tracking all six degrees of freedom (position and orientation);
- (4) **Accurate** - with resolution better than 1 mm in position and 0.1 degree in orientation;

- (5) **Fast** - running at 1,000 Hz with latency less than 1 ms, no matter how many ToCs are deployed;
- (6) **Immune to occlusions** - needing no clear line of sight to anything else;
- (7) **Robust** - resisting performance degradation from light, sound, heat, magnetic fields, radio waves, and other ToCs in the environment;
- (8) **Tenacious** - tracking its target no matter how far or fast it goes;
- (9) **Wireless** - running without wires for three years on a coin-size battery; and
- (10) **Cheap** - costing \$1 each in quantity.

Every tracker system which is available today falls short on at least seven of these 10 characteristics. This number of criteria that current motion capture systems do not satisfy is unlikely to shrink much in the foreseeable future. But still some of the criteria are well fulfilled as researchers and developers have pragmatically and cleverly exploited every available physical principle to achieve impressive results for specific applications.

1.6 Motivation

Highly articulated biomechatronic systems, such as humanoid robots, exoskeletons, and medical assistive devices, are gaining high popularity in recent years because of their potential applications in human-centered environment for health care, service, security, and entertainment purposes. However, in most current motion tracking systems, inflexibly confine working spaces are requisite. This hinders the widespread latitude of using motion capture in low-end (low-cost) applications. In addition, all current motion capture systems are very far from the ideal ToC. Moreover, there are very few sensing technologies, while algorithms that edit, transform, interpolate, and recompose motion data are abundant [17]. The criteria and the specifications of the desired motion capture system are summarized in Table 1.1. The first column lists the criteria to appraise a motion capture system. The second column summarizes the best specifications that current commercial motion capture systems have with respect to the criteria. The third column presents the goals of our design for each of the specification enumerated in the first column. The last column shows the ideal motion tracking device proposed by [17].

TABLE 1.1
COMPARISON OF IDEAL TOC CHARACTERISTICS

Characteristics	Current commercial Systems	Our system	Ideal (ToC)
Small-sized	15 mm x 15mm x 15 mm smallest	10 mm x 15 mm x varying length	8-pin DIP
Self-contained	Require peripherals	No other parts	No other parts
Complete	6 DOF	6 DOF	6 DOF
Accurate	5°, typically	3°	0.1°, 1 mm
Fast	200 Hz, maximum	100 Hz	1000 Hz
Immune to occlusions	No, typically	Yes	Yes
Robust	Degradation due to light, sound, etc.	No degradation	No degradation
Tenacious	60 m, typically	50 m	Unlimited
Wireless	Hours on batteries	Days on battery	Years on battery.
Cost (USD)	\$50K-200K/system	<\$50 each	\$1 each

A new sensing technique, termed Optical Linear Encoder (OLE), is proposed in the SmartSuit project to achieve these goals. The association of the OLE with an accelerometer creates an integrated sensing module, which is one step closer to the ideal one. The integrated sensing module captures the linear information (measured by an OLE) and orientations (tracked by a MEMS-based accelerometer) of the human body simultaneously.

The integrated sensing node has its own field of applications, for example functional assessment of individual joint motions in rehabilitation. However, there is a crucial need for connecting the sensing modules into a sensor network for wider range of applications, such as capturing human limb motions, or even entire body motions. To achieve this, the embedded firmware, which is a fixed and small program downloaded into the sensor chips for controlling connected devices, is required. The firmware architecture is liable for handling data acquisition from the OLEs and accelerometers in the sensor network, transmitting motion data within the networks, and communicating with other application devices, such as personal computers.

Furthermore, as the resource and computational power of the embedded microcontroller of the sensor nodes are limited, most of the data processing could not be implemented in the hardware. Therefore, with the collected raw data from the sensor hardware, motion processing software is essential to bridge the complete the motion capture system, and bridge the hardware with applications. Raw data provided from the integrated sensing module are mere encoded numbers. We need to calibrate the sensor, and convert the numbers into meaningful information, for examples converting the OLE signal into joint angles, converting the acceleration from accelerometer into orientations. However, the converted data still does not reflect the actual motion of the user. Hence, a kinematic model is required to construct human motion data structure from sensor data. Accelerometers possess singularities which disable them to capture motion of human arms and entire body. Due to the intrinsic design problem, accelerometers fail to capture any rotation about gravitational axis. When the local axes are nearly parallel to the gravitational axis, sensor noises are intensified. This motivates the development of a sensory kinematic model to fix the accelerometers' problems in combination with the OLE.

1.7 Objectives and Scope of the Thesis

The objective of this project is to design a body sensor network and embedded firmware for connecting the SmartSuit sensing modules to capture human motion, and to develop a software framework for reconstructing human motion as well as bridging the motion capture system to application fields.

The scope of this thesis is as follows:

- 1) *Design the sensor network architecture to connect the sensing modules:* A optimal method of linking the sensors needs to be investigated. The network architecture must handle the data acquisition and transmission properly to ensure the real-time operation of the motion capture system with the least power consumption. The design should keep the size of the sensor nodes compact. In addition, the sensor network must be comfortably wearable on human body. The sensor network of three invented sensing modules and one Data Concentrator will be established to capture the motion of human arm. The communication within the sensor network is performed via CAN bus, while the communication between the network and the computer is performed via three alternatives, which are RS232, USB cable and Radio Frequency (RF) transmission.
- 2) *Develop the embedded firmware for the sensor network:* The firmware is written for controlling all the electronics devices in the network, implementing communication protocols and designed network architecture. The firmware should synchronize the communication among the network, and optimize the speed of transmission. The firmware is written in the C programming language, and is compiled for Microchip's microcontroller. It is written and embedded into three nodes and the Data Concentrator, performing the ultimate task of capturing motion of human upper extremities.
- 3) *Propose the kinematic model:* The kinematic model maps the sensor data, which are linear encoder readings from the OLEs and acceleration from the accelerometers, into human joint angles and motion. The model would also solve the accelerometers' singularities with the compensation data from the OLEs. The kinematic model is investigated for human arm only because the sensor network is limited for arm motion capture. However the model can be easily extended to entire human body with the same modeling method.
- 4) *Build the motion processing software for SmartSuit:* The software is programmed to provide the tool for the computer to interface with the motion capture hardware. The software communicate with SmartSuit via communication ports of the host computer, collect the data then feed them into the kinematic model to reproduce the motion human motion. It plays the role of a Graphical User Interface for users to view and analyze recorded human motion. The motion capture software, written in the C++ programming language and

compiled by Microsoft Visual Studio, graphically displays the recorded motion. The software will be designed with the aim to provide a virtual-reality based software for rehabilitation applications. The software is implemented with only crucial modules, such as interfacing with hardware, calibration, kinematic model, graphic display, and interfacing with other platforms through standard motion data formats.

5) *Validate the sensor network architecture, the firmware, and the software with experiments:* The experiments include the tests of the SmartSuit's performance in comparison with other commercial motion capture systems, and the test for SmartSuit's reliability and repeatability. The experiments are carried out for human arm motions to validate the SmartSuit sensor network as a goniometry device and as a complete motion capture system.

1.8 Report Outline

This report consists of seven chapters. Chapter 1 provides the introduction, literature reviews, motivations, as well as the main objectives of the project.

Chapter 2 illustrates innovation of an inexpensive, accurate and wearable motion sensor unit, which combines a Linear Encoder and an accelerometer. This chapter provides the background work for the thesis project. All sensing module hardware, sensor packaging and placement are crucial to the system integration.

Chapter 3 discusses the construction of a body sensor network and related issues, such as network design and architecture, and embedded firmware of the sensor network based on SmartSuite sensing modules.

Chapter 4 proposes the PC software architecture, and discusses the development of the motion capture software based on the suggested software framework. The chapter also explains standard data format used in the motion processing software.

Chapter 5 is devoted to the experimental validation and comparison test of the OLE in comparison with other commercial motion-capture systems.

Chapter 6 reports the experimental validation and results of the repeatability tests and reliability tests on SmartSuit. The tests were carried out when users wear multiple SmartSuit sensing modules forming a network to perform motion routines emulating a real rehabilitation therapy session.

Chapter 7 concludes the project with a summary of what that have been achieved, contributions of the works, and possible research directions in the future.

Chapter 2 Background on SmartSuit Sensing Module

The purpose of this chapter is to provide the background knowledge about the core technology invented in the initial phase of the SmartSuit project. SmartSuit's sensing module is designed to be a motion capture device, composing of an OLE and an accelerometer. This chapter includes the discussion on architecture, design, working principle, placement and packaging of the sensing module.

2.1 Optical Linear Encoder

Optical linear encoder is the fundamental element of the invented sensing technique. The linear motion of the encoder head along the base structure is recorded by the optical signals reflected from an engraved strip. The travelled distance of the encoder is then converted to joint angle using a mapping model with calibration factors.

2.1.1 Working principle

The linear encoder is a miniature optical sensor that outputs a pulse as it senses an interruption to its reflective path. It has both an infra-red emitter and a receiver built in a single package. The infra-red light from the emitter is reflected off a reflective linear code strip, engraved with parallel lines 0.1mm apart.

The reflected light is captured by a receiver. If the light is indeed reflected (i.e. not interrupted by the micro-lines engraved on the plastic substrate attached to the base reflective material), the signal processing circuitry will output two electrical pulses (i.e. channel A and B) that are 90° out of phase. The two electrical pulses that are out of phase indicate the travel direction of the sensor with respect to the linear encoder (Figure 2-1). The pulses represent the number of interruptions to the sensors' reflection. This linear code strip, thus provide a measurement device indicating the distance traveled as the encoder moves over the linear code strip. Each pulse that is read from the linear encoder represents a distance of 0.1mm.

Figure 2-1 Channel A and B 90° out of phase to indicate forward or backward traverse and block diagram of linear encoder sensor [18]

2.1.2 Mapping from linear information to joint angle

In order to obtain the joint angle of a human joint, e.g. elbow, the linear encoder is attached to a wire that is fixed to one side of the joint, i.e. forearm, while the linear encoder assembly is fixed to the upper arm. The linear encoder is free to slide along its longitudinal axis, which is coincident with the single free axis of the wire. When used, the linear encoder is attached to the body skin. The sensor is wrapped around the joint. Because the center of rotation of the joint is within the skeleton structure, not on the surface of the limb segment, the offset from the sensor placement to the joint rotation center will enable us to interpret the joint displacement by the linear displacement of the sensor reading as shown in Figure 2-2. Assume that the stretch caused by the elbow bend produces a linear displacement of D (provided by the encoder readings). Also, we assume that the radius of the offset disk formed due to the sensor placement is R . The sensor displacement due to the joint bending along the disk, Δx , is therefore equal to D . Hence we have:

$$D = \Delta x \quad (1)$$

We denote α to be the rotation angle of the joint, and R to be the radius of the offset disk. Because Δx is along the circular arc of the offset disk, and due to (1) the following relationship among arc length, subtended angle and arc radius holds:

$$\Delta x = \frac{\alpha}{360^\circ} \cdot 2\pi R$$

Therefore,

$$D = \frac{\alpha}{360^\circ} \cdot 2\pi R \quad (2)$$

Thus the joint angle can be estimated from the linear displacement of the OLE using equation (3) as follows:

$$\alpha = \frac{D}{R} \cdot \frac{360^\circ}{2\pi} \quad (3)$$

The radius of the offset disk, also known as the radius of the elbow joint, R , is assumed to be constant.

Figure 2-2 OLE working principle - conversion of displacement to angle

For example with the Figure 2-3 below, for the elbow performing some motion, assume the rotation radius to be 3cm (can be identified during calibration); the slider moves 2 cm. According to equation (3), the movement angle would be 38.2°.

Figure 2-3 Example of the data processing on the actual body motion

2.2 Sensor Electronics

The wearable sensing module is self-contained, consisting of an OLE, an accelerometer, a Digital Signal Controller, and a CAN controller. The OLE measures the angle of the joint it wraps around by counting the number of active signals from a code-strip. The accelerometer gives the orientations of the link on which the sensing module is mounted by the relative accelerations along its local coordinate system with respect to the gravitational acceleration. The CAN controller collects all OLE and accelerometer data and queue them to CAN bus when the sensing module is connected to a body sensor network. CAN controller is also in charge of message filtering and synchronization. The electrical block diagram and the actual component circuit of the sensing module are depicted in Figure 2-4 and Figure 2-5 respectively.

The DSC employed here is Microchip's dsPIC33FJ12MC202, a 16-bit digital signal processor based on RISC architecture. In the prototype, it communicates with the accelerometer via the SPI protocol. The DSC uses a 16-bit Quadrature Encoder Interface (QEI) peripheral that has a programmable digital noise filter on input pulses to capture pulses from the linear encoder. After the signal processing, the DSC packages orientation information with a tracker identification number, and then transmits the sets of information to the CAN controller, MCP2515, enabling the module to communicate with other sensors when networked. The Avago™ AEDR-8400-132 miniature linear encoder is a 3VDC quadrature linear encoder with an accuracy of 0.1 mm. The sensor measures only 3mm x 3.28mm x 1.26mm in size. The linear encoder communicates to the DSC's QEI. The LIS3LV02DQ accelerometer from STMicroelectronics is a tri-axis low-g MEMS-based capacitive accelerometer. The LIS3LV02DQ measures 7mm x 7mm x 1.3mm, weights about 0.2 gram, and has an effective sensing range from -2g to +2g, in which $1g = 9.81m/s^2$. It includes a sensing element capable of measuring linear acceleration signals over a bandwidth of 640 Hz, and an IC interface able to send out the data in 12/16 bit data representation through an I2C/SPI serial interface.

Figure 2-4 Block diagram of the sensing module

Figure 2-5 Components of the sensing module prototype

2.3 Mechanical Design

The sensing unit is self-contained in that it embeds all the components in one single package of $12 \text{ mm} \times 22 \text{ mm} \times \text{varying length}$. The length of the sensing module depends on the joint angle to be measured, for example with elbow joint the module's length is designed to be 60mm. A miniature linear encoder is made to slide over a Delrin base structure as shown in Figure 2-6. The encoder is attached to a flexible stainless steel wire that has a diameter of 1mm. It is guided by a tube to restrict it to one degree of freedom. The linear code strip is adhered to the base structure. The base structure's function is to allow the encoder to traverse above the linear code strip while maintaining a constant gap of about 0.4 mm between the code strip and linear encoder.

Figure 2-6 (a) CAD drawing of the base structure; (b) Actual base structure

The base structure is cut with three grooves to guide the motion of the slider. The slider, which bears the transmitter and receiver of the encoder, is designed with six pins fitting onto the three grooves of the base structure (Figure 2-7). The pin-groove design makes sure that the contacts between the slider and the casing are at six contact points, therefore maintains the smooth and stable motion of the slider onto the base structure.

Figure 2-7 (a) CAD drawing of the slider; (b) Actual slider

2.4 Placement of the sensing modules on human arm

An extensive study on biomechanics and structure of human arm was conducted to systematically determine the placement of OLEs on joints to record arm motions most effectively [19].

2.4.1 Shoulder joint

The shoulder joint is where the humerus attaches to the scapula, made up of three bones: the clavicle (collarbone), the scapula (shoulder blade), and the humerus (upper arm bone) as well as associated muscles, ligaments and tendons. There are three joints of the shoulder: glenohumeral, acromioclavicular, and sternoclavicular joints. The glenohumeral joint is the major joint of the shoulder, which is a ball-and-socket joint that allows the arm to rotate in a circular fashion or to hinge out and up away from the body and around it. Being aware of this, the sensor can be fixed on the upper arm while the fixed end is placed at the top of acromial arch as shown in Figure 2-8 (left). When the arm moves horizontally around the body, the sensor will move relatively to the fixed end, and, therefore, give the data indicating the angle of rotation (the other two rotations are tracked by the accelerometer of the same sensor node).

2.4.2 Elbow joint

The elbow-joint is a ginglymus or hinge joint which is a one-DOF joint, formed by three bones: the humerus of the upper arm, and the paired radius and ulna of the forearm. Placement of the sensor for the elbow is rather simple: the fixed cable end-point is placed as reference point on the upper arm while OLE will be placed on the forearm, as depicted in Figure 2-8 (middle). The pulling wire is ensured to pass through the olecranon so that the OLE responds most sensitively to the elbow rotation.

2.4.3 Wrist joint

The wrist is a region and is a complex of joints formed by double row of small short bones, carpals, twisted to form a malleable hinge. The wrist-joint is a condyloid articulation allowing 3-DOF motions. The narrow-range rotating motion of wrist poses as a problem for some of the motion capture systems especially for those optical (visual) and ultrasound sensors as they are unable to differentiate between the positions of the two markers which are closely placed together hence causing confusion to the data captured when a rotation motion was executed during the capturing. To capture the motion of wrist flexion and hyperextension (other rotations are given by the accelerometer), the OLE is attached on the back of forearm, and the fixed cable end-point is on the back of hand, Figure 2-8 (right). The encoder will travel backward (toward the elbow) when the wrist is in hyperextension and

forward (toward the wrist) when the wrist is in flexion. The travelled distance of the encoder is then converted into the rotation angle of the wrist.

Figure 2-8 Placement of the OLEs on shoulder, elbow and wrist joints

2.5 Sensor Package design

When designing the sensor package on the human arm, the following criteria are required:

1. The sensors are located along the planes proposed in the above section.
2. The sensor unit with the casing and the tip of the pulling cable should be fixed at two different links of a joint so that the OLE can accurately measure the joint movement.
3. The sensor package ensures a suitable comfort level without sliding on user's skin.
4. The package is easy to wear by one person and fit to human arms with different sizes.
5. The sensor package should not constrain the nature movement of the human arm.

Based on these criteria, we designed a modular-based fabric suit to package sensors in this study. The suit includes three modules, the shoulder, elbow and wrist suits.

Figure 2-9 (a) The wrist module: The tip of strip is locked by a PVC Connector which is shown at the down side of this figure. The sensor, the connector and the tip will be covered by the fabric.
(b) Plane view of the wrist suit on the human arm.

2.5.1 Wrist module

As seen in Figure 2-9 (a), the wrist module holds two sensors. The two sensors are mounted on a Velcro strip to ease replacement and adjustment of the position of the sensors. The guiding strip is then covered by a layer of extension of the fabric to secure the path of motion of the guiding strip and also for aesthetic purposes. The fixed end used to secure the tip of the guiding strip is also covered by another piece of fabric of the same textile. The tip of the guiding strip is locked by a 2-Way PVC connector. This connector only consists of a small metal connector. It is compact in size, and also directly attached to a particular spot on the suit to reduce the overall size of the suit. The wrist module is placed on the human wrist, as shown in Figure 2-9 (b). A first wrap of the extension of the textile is meant to secure the position of the module. A second wrap around the sensors is used to hide them visually and for aesthetic purposes.

2.5.2 Elbow module

As seen in Figure 2-10, the elbow module also holds one sensor at one side and locks the tip of the guiding strip at another side. At both side, the Velcro straps are used to fix the module on the upper arm and the fore-arm. Furthermore, individual housing is used to cover the sensor. To overcome the problem at the area of elbow joint, the textile is extended and featured a securing method to maintain contact with human skin.

Figure 2-10 Elbow module: (a) the prototype of the elbow module; (b) the module worn on the elbow

2.5.3 Shoulder module

As seen in Figure 2-11, the shoulder module composes of one sensor and the concentrator used in the CAN bus. The entire sensor is covered by fabric for aesthetic purpose. A pouch is located at near the fixed end of the guiding strip to house the battery pack.

Figure 2-11 (a) Should module prototype; (b) Side view of the module worn on the shoulder

2.6 Summary

SmartSuit's sensing module can be used as a body motion tracking device, which is a combination of an OLE and an accelerometer. The new method for sensing body joint motion uses optical linear encoder (OLE) to convert the encoder head's travelled distance into the joint angle. The OLE acts as a goniometry device measuring joint angles. The accelerometer provides the orientations of the limb segment it is attached to. Engineering design of the sensor assembly, i.e. electronic architecture and mechanical design of the sensing module are described here. The working principle of OLE's sensor is introduced. The design for packaging and scheme for placing each sensing module on human arm is proposed and prototyped. The similar sensor packaging and placement would work for other body parts, such as legs. The following chapters will describe the embedded programming and PC programming for SmartSuit sensors for complete system integration.

Chapter 3 Design of Motion Sensor Network

As introduced in the last Chapter, the wearable sensing node has excellent capability of capturing joint angles. The method of placement on human arm and sensor package were also discussed. For wider range of applications, such as capturing human limb motions, or the entire body motions, there is a need for setting up a sensor network of the wearable sensing modules. To achieve this, the embedded firmware for programming the sensor communication is required. The firmware is responsible for handling the polling of data from the OLE and accelerometer of every node in the sensor network, transmitting motion data within the networks, and communicating with the host computer. This chapter will first present a review on network topologies, evaluate and propose an appropriate network topology for SmartSuit. The SmartSuit network architecture will be discussed in details.

3.1 Selection of Sensor Network Topology

In connecting the sensing modules into a sensor network, proper network topology to link the sensors needs to be studied. Sensor network topologies, the manner of arranging sensor nodes in a network, are classified into six main types: point-to-point, bus, star, ring, mesh, and hierarchy [20]. Each of the network topologies will be discussed.

The number of sensors in the network is determined based on the joints' mobility. Each upper limb needs three SmartSuit sensing modules: one for capturing three DOFs of the shoulder (the accelerometer provides two DOFs, and the OLE provides one DOF), one for capturing the one DOF elbow motion (only the OLE is used at this joint), and one for capturing three DOFs wrist motion (two DOFs from the accelerometer, and one DOF from the OLE). Similarly for each lower extremity, three SmartSuit sensing modules are needed: one for capturing three DOFs of the hip joint, one for capturing the one DOF knee motion, and one for capturing three DOFs ankle motion. To track the motion of head and body trunk (upper trunk and lower trunk), Inertial/Magnetic Units (IMUs), which can provide three rotations of the body it is attached to, are requisite. So in total, 15 sensor nodes, including 12 SmartSuit sensing modules to capture limbs motion and three IMUs to capture head and trunk motion, are needed for full-body motion tracking. Figure 3-1 shows the conceptual layout of the sensors on the body.

Figure 3-1 Layout of the 15 sensor nodes and the Data Concentrator on a human body

The Data Concentrator acts as a gateway for collecting the motion data from sensor nodes and transmitting them to the host computer. For wearability, the desired implementation method is through wireless protocols. However, the complexity of the network communication increases with the number of wireless nodes. The Data Concentrator needs to have sufficient capacity and bandwidth to handle all the sensor nodes' communications. For a wearable system, such Data Concentrator design may not be suitable due to power consumption and complexity concerns. Also, because of the wireless communication, each sensor node needs a battery and a transmission unit on board. This will significantly augment the size of the sensor nodes. In addition the transmission speed of available wireless protocols is still significantly limited. Wireless network signals also suffer from the

interferences of ambience and of its own network components. Therefore, the implementation method is decided to be through wires. The problem of wired network is that the wires may hinder the system's wearability and user motion. Hence, careful design of the network topology is required.

The requirements for the design problem are as follows:

- When worn on the human body, the sensor network must maintain the comfortable feature of a wearable system, and must not hinder the motion of the user.
- Data from the 15 sensor nodes are transmitted to the Data Concentrator in the fastest and most direct manner to maintain the real-time operation of the motion capture system.
- Minimum signal processing and communication algorithms are required.
- The network must be reconfigurable. This means that it can be used for tracking the motion of single joint or single limb segment, and multiple joints or limb segments as well when needed.

The problem statement is therefore: **Given the 15 sensor nodes for capturing motion of human body, what is the best network topology for SmartSuite system to satisfy the above design requirements?**

3.1.1 Point-to-point topology

Point-to-point network topology is the most basic network structure to provide the connection between two end-nodes. There are two types of point-to-point topology, which are permanent and switched. In permanent topology (Figure 3-2 (a)), the link is dedicated to connect the nodes. The advantage of this type is that the communication between the two end-nodes is always guaranteed. On the other hand, in switched topology (Figure 3-2 (c)) the link is controlled to change the branch. The advantage of this method is that the network can be flexible and dynamic because such a connection can be dropped when not needed. A normal telephone is one typical example of this type: the connection is only set up when a

user dials a specific number, and switched to another node when the user dials a different number. When a set of this type is linked together linearly, we have a line topology.

Figure 3-2 (a) Permanent point-to-point topology; (b) Permanent line topology; (c) Switched point-to-point topology; (d) Switched line topology

If the point-to-point is used for SmartSuit, permanent method will be used since there is no need for switching connections. In this case, the 15 sensor nodes will be connected in a line structure, similar to the line topology shown in Figure 3-2 (b). Switched topology is used in line topology (Figure 3-2 (d)) only in the case that changing the line branch is needed. In line topology, the 14th sensor will acquire the data of the last node from the 15th node. The 13th sensor will acquire the data of the last two nodes from the 14th node. And the procedure keep going until the data of all 15 nodes are collected by a Data Concentrator and transmitted to the computer. Therefore, there will be too much waiting time in the operation, meaning for example the 13th node has to wait for the 14th node to finish the acquisition, and so on. In addition, data transmission consumes too much memory resource in the microcontrollers of the nodes. For instance the 14th node has to store the data of itself and the 15th node; the 13th node has to store the data of itself and the 14th node and the 15th node, etc. Hence when the number of nodes becomes a large number, the need for data storage will become significantly large. Another problem with this topology is that if any node does not work properly, the entire network fails to perform. Thus, this network topology is not efficient for our system.

3.1.2 Ring topology

The difference between ring topology and line topology is that in ring topology the last node is connected to the Data Concentrator as shown in Figure 3-3. Data are passed from one node to another without an end. The only superior feature of this type of topology is that the data concentrator can communicate directly with the first and the last sensor nodes, instead of

only the first node as in the line topology discussed previously. This topology has the disadvantages of the line topology mentioned. In SmartSuit, data need to travel from the sensor nodes to the Data Concentrator in the most direct manner and then to be transmitted to the host computer. Therefore, this topology is not suitable for SmartSuit.

Figure 3-3 Ring topology

3.1.3 Star topology

In star topology, all the nodes are connected to the Data Concentrator, or the central node, through either permanent or switched point-to-point method. Figure 3-4 illustrates the layout for a star network. The Data Concentrator acts as a station, to which data from the nodes are transmitted. The Data Concentrator can send the data of all the nodes at one time after collecting them. In this case, the data load at the Data Concentrator would be very heavy. Alternatively, the central node may arrange with the computer which node's data the computer wants at the specific time, and then send them according to the requests. Thus, proper handshaking and synchronization need to be done. A disadvantage of this topology is the network performance depends greatly on the processing capacity of the central station.

Figure 3-4 Star topology

If star topology is applied for SmartSuit sensor network with the Data Concentrator as the central node, data from each node will be transmitted directly to the central node without passing through any other node. The waiting time during operation can be minimized, and the failure of any peripheral node will not disable the entire network. The capacity of the network (number of sensor nodes) can be easily scaled without changing the network structure. Wired star connection allows data to be communicated to the Data Concentrator directly without interference from other nodes. However, physical wiring of 15-node star network poses a real challenge for a wearable system. Such arrangement may also hinder the body motion of users.

3.1.4 Mesh topology

A mesh topology in which all nodes are connected to each other via point-to-point links is called a fully connected mesh. This kind of mesh allows data from any node to be concurrently passed to all of the other nodes. Fully connected meshes are normally costly and intricate for practical network. Figure 3-5 (a) gives an example of a fully connected mesh network. On the other hand, the mesh topology in which some nodes are linked to several other nodes is called partially connected network. This type of mesh is generally simpler and more flexible than fully connected one, and eliminates redundant links to reduce unnecessary communication overhead and also the cost. Figure 3-5 (b) shows a partially connected mesh network. One advantage of the mesh network is that the sensor nodes can be connected, and able to communicate with one another. Without this feature, mesh topology becomes the star topology. Since there is no fixed general structure for mesh topology, this type has high flexibility for connecting networks. Routing of a mesh network is planned depending on the specific applications and requirements, and can be very complicated. For example, a mesh can be designed for least chance of failure, or for fastest data transmission, and so on. A mesh network basically configures the data to travel in the shortest path, or by taking the least number of hops. In SmartSuit motion capture system, the communication among the sensor nodes is not required, while the main communication is between sensor nodes and the Data Concentrator. The reason is that motion data from the sensing module are supposed to be transmitted directly to the Data Concentrator. A disadvantage of the mesh network is that the delay introduced by multiple hops increases with large number of sensor nodes. To relieve the problem, a distributed mesh network can be utilized by adding more Data Concentrators. Each Data Concentrator handles a mesh network of a small number of sensor

nodes. These Data Concentrator then either directly transmit the motion data to the PC (multiple communication lines), or send the data to a main station, which in turn transmits the data to the PC (single communication lines). Hence, the distributed mesh is a possible method to be considered.

**Figure 3-5 Two types of mesh network topology: (a) Full connected mesh;
(b) Partially connected mesh**

3.1.5 Hierarchical topology

A hierarchical network is a hierarchy in which the Data Concentrator is at the highest level, as the root node, and the sensor nodes are the descendent nodes at lower levels. The Data Concentrator and the descendent nodes will form a tree-like structure with all point-to-point links. Figure 3-6 shows a four-level hierarchical network. A hierarchical structure has at least three levels including the Data Concentrator. The number of links of a hierarchical network is equal to that of a star structure, which is the number of sensor minus one. Data are transferred via the links from a child node to the root node, Data Concentrator. For optimal performance, the Data Concentrator needs to schedule a searching algorithm to find the shortest path to send the request to a specific sensor node, and for the data to be transmitted back. For networks where a node needs to process the data of its child node before sending the data to its parent, this topology is very useful. Similar to the mesh topology, the hierarchical topology is not a good choice for SmartSuit because the highly ordered communications among levels of sensors in hierarchy are redundant for SmartSuit. The reason is that if data flow from a lower level a higher one (not directly to the Data

Concentrator), the controllers of the nodes at the higher level need to collect the data, store them and transmit to their parents, until reaching the Data Concentrator. This data flow suffers from a delay each time the data pass a node, and consumes unnecessary computation and memory resources.

Figure 3-6 Hierarchical topology

3.1.6 Bus topology

In a sensor network established by the bus topology, a long and shared communication line (or lines), or a backbone cable, is utilized to connect all the nodes as well as the Data Concentrator (Figure 3-8). This type of network structure requires much less cabling compared to the previously discussed topologies, because the only wire needed is the backbone cable. Addition of nodes to the network is very simple and can be performed even when the system is in operation. A bus network allows each sensor node to queue its data into the shared communication line, or bus. Problems may arise when more than one node try to pass the data to the Data Concentrator at the same time via the bus. In order to deal with this, each node in the network can be designed with a bus controller, which schedules some routine of collision handling or collision avoidance for communication on the bus. Figure 3-7 illustrates the difference between bus topology (Figure 3-7 (a)) and the others, such as partially mesh (Figure 3-7 (b)) and hierarchical network (Figure 3-7 (c)). In the figure the dotted arrows depict the data flow. In bus topology, data flow from each sensor node to the communication line and are picked up by the Data Concentrator. In other topologies, data flow into other nodes in order to reach the Data Concentrator. This means the processor of each node need to process the received data from other nodes and transmit them together with its own data to the next one.

Figure 3-7 Data flow of bus topology (a), partially connected mesh (b) and hierarchical topology (c). The dotted arrows depict the data flow

There are two types of bus topology: linear bus and distributed bus. The simpler one, linear bus (Figure 3-8 (a)), includes only one communication line with two end points which act as terminating resistance to absorb spare signals. For the more advanced structure, distributed bus (Figure 3-8 (b)), the communication line starts at the root, or head end, and forks into various sub-lines. This type of bus has multiple end points. Both linear and distributed bus topologies are suitable for SmartSuit because the daisy-chain connection of the bus network matches the intrinsic human body and limb structure. Motion data from the sensor nodes are transmitted directly to the Data Concentrator via communication lines, not passing through any other sensor node. If scheduled properly, signal jam can be avoided. The communication protocol and scheduling routine are also simple. Mature commercial interfaces, such as CAN bus, provide very high data transmission speed up to 1Mbit/s. In addition, this bus design is suitable for body sensor network as it requires much less wiring, and minimizes inconvenience to user motions.

Figure 3-8 (a) Linear bus topology; (b) Distributed bus topology

3.1.7 Discussion on the selection of the network topology

We can conclude from the discussion in the previous sections that the bus topology offers a good solution to the requirements of SmartSuit as outlined in section 3.1. This section will discuss the implementation of the body sensor network for SmartSuit with optimal results.

The most straight forward and efficient way is to utilize linear bus topology to connect the entire 15 sensor nodes and the Data Concentrator. However, there are practical constraints in the implementation. Therefore, to keep the sensor node small and light weight, the battery should not be installed at each node. It is desirable to supply power to the sensor nodes from a single point in the network, which is the Data Concentrator. In SmartSuit, if linear bus is implemented, the last sensor nodes may not receive sufficient power to properly operate due

to engineering constraint that limit the maximum number of nodes along a linear bus. It was found from the tests that the interferences in the network make the current drop gradually after each sensor node along the path. In the current configuration of the sensor nodes, the power supply from the Data Concentrator can only support up to four nodes. Moreover, wiring becomes an issue if the sensors are arranged on human body as shown in Figure 3-9. The reason is that the long linear communication line needs to connect the sensor nodes along the arms, the legs and then the body trunk to the head.

Figure 3-9 The system of 15 sensor nodes is connected by distributed bus topology

To overcome the problem, distributed bus network could be used. Human body itself has a distributed structure from the main body branching to limb and head. Figure 3-9 conceptually depicts the topology implemented on full SmartSuit system. The main communication line is

branched into four sub-lines by a bus router. The four bus lines connect sensors along right arm, left arm, right leg and left leg, and meet together at the bus router. The Data Concentrator is linked to the main line to collect motion data flowing on the bus lines. The drop of current can be minimized in the distributed network because each line has only three sensor nodes. This network design is also reconfigurable, allowing SmartSuit to be used in a modular form. For instance, in some applications, only motion of an arm is interested. The other three branches can be unplugged from the bus router without affecting the operation of the arm line. In this project, only one branch of the sensor network is established for track the motion of human upper extremities. This can be extended to other branches of the human body.

3.2 SmartSuit Network Architecture

This section discusses about the network architecture and how SmartSuit system works as a body motion sensor network. In this project a network to capture arm motions, consisting of three sensor node and one Data Concentrator, is established. Figure 3-10 illustrates the architecture of SmartSuit sensor network. CAN bus with well mature protocol and high speed of transmission (up to 1Mbps/s) is used to realize the proposed SmartSuit network structure.

Figure 3-10 Motion capture sensor network block diagram

The function of the **Data Concentrator** is to acquire motion data from sensor nodes and communicate with the application devices, such as a personal computer or Personal Digital Assistant (PDA). The Data Concentrator includes: (1) a CAN controller to connect itself with the sensor network via the bus communication line, schedule network transmission, and prevent data jam and collision; (2) a Microchip microcontroller to process the data and manipulate transmission, handle wireless protocol, as well as synchronization between the application device and sensing modules; and (3) an RF transmitter to send and receive

wireless signals with the application device. The bus line composes of four wires, two for signal (CANH for high bytes and CANL for low bytes) and two for power supply drawn from the Data Concentrator (source and ground). Power from batteries is moderated by a regulator to 3.3V, which is the standard source for all the electronic components of the board and the sensor nodes. The Data Concentrator also allows the sensor network to communicate with computers via a USB cable with a USB hub onboard. The Data Concentrator acts as a gateway, as well as the master which control the behaviors of the slaves (sensor nodes). In SmartSuit sensor network, CAN bus plays as a broadcast type of bus, in which all nodes can "listen to" all transmissions. The motion data is labeled with a unique identifier for each node instead of containing addresses of either transmitting or receiving node. There is no way to send a message to just a specific node; all nodes invariably pick up all traffic. Using a message filter, embedded in the nodes' firmware, each node responds only on a specific message. Figure 3-11 shows the Data Concentrator with the RF board plugged on top of it.

Figure 3-11 Data Concentrator

Figure 3-12 shows the picture of an actual SmartSuit network connected by CAN bus. All the nodes in the network have the common component structure. The Microchip dsPIC33F32MC204 micro-processor stores the program in memory, which constantly polls analog signals from the LIS3LV02DQ accelerometer and digital signals from the OLE via

Quadrature Encoder Interface (QEI), and then queue the combined message via SPI port to the Microchip MCP2515 CAN controller, which handles all transmitting, receiving and filtering of message packets of useful information, and allows its host node to communicate with other nodes and the Data Concentrator in the CAN bus network. The CAN controller passes the data to a CAN driver through the buffer couple TX and RX. The signals are converted to the CAN bus voltages ranging from 0 to 5 volts, with a shift of $\pm 12V$ by the CAN driver chip MCP2551.

Figure 3-12 Actual SmartSuit sensor network

Data acquisition is started as the application device, i.e. personal computer, sends a request to the data concentrator with structure:

`<type> <location> <nodeID>`

This request specifies what type of data is demanded, the location of the node and its ID. Data Concentrator in turn collect the respective information from the interested sensing module via CAN bus, and then send back to the application device, through a wireless RF transmitter, the message with the following structure:

<type> <location> <nodeID> <encoder> <X> <Y> <Z>

The reply includes the configuration information, and the respective encoder readings, as well as acceleration along x-, y-, and z- axes of the accelerometer. The entire transmitting process is synchronized by the standard CAN protocol.

A Radio Frequency (RF) station is designed to get the motion data from the sensor network. Figure 3-13 shows the block diagram for wireless communication from the sensor network to the application device (PC). This RF station, which is plugged into the computer via USB port, communicates directly with the RF transmitter of the Data Concentrator. The RF station runs with TinyOS operating system designed for wireless embedded sensor networks. The Data Concentrator initially searches for a path to the RF station at the computer. If the station is detected, the Data Concentrator will establish the communication using ZigBee protocol. Motion data are transmitted in a 28-bit data payload per package. A time stamp is included for each set of samples. Handshaking between the Data Concentrator and the RF station is controlled by the Media Access Control layer as specified by the IEEE 802.15.4 standard. The data acquisition process is controlled by the software with data processing, as well as graphical display units. Even though all experiments in this project were performed using a computer, the system is able to work with other mobile devices too.

Figure 3-13 block diagram for wireless communication

3.3 Accelerometer

The OLE and accelerometer are the two crucial sensing components which are integrated in each sensor node. OLE gives linear distance that the encoder has traveled, which can be converted into joint angles when worn on human body. The accelerometer measures the acceleration of the sensor node. This acceleration can be used to calculate the orientation of the sensor by projection of the gravitational acceleration vector onto the accelerometer's local coordinate system. In this section, the working principles of the accelerometer will be described.

An accelerometer captures its own acceleration relative to gravity. Thus the measured acceleration is equal to the difference between the inertial acceleration and the local gravitational acceleration. Here, the inertial acceleration describes change of velocity due to a localized force, applied only on a point or on a surface. The point transmits the localized force to all other particles inside the accelerated body. To obtain the inertial acceleration (due to motion alone), the gravity offset needs to be subtracted. As a result, an accelerometer at rest on the Earth's surface should indicate $1g$ ($\sim 9.81 \text{ m/s}^2$) downwards along the vertical axis perpendicular to the ground. When the accelerometer is at an arbitrary orientation with respect to the ground, it yields acceleration components along its own local coordinate axes. Thus by finding the gravitational acceleration vector in the local coordinate system of the accelerometer, the orientation of the accelerometer can be estimated [21].

Figure 3-14 Pitch (θ) and roll (ϕ) orientation of an accelerometer

We denote the pitch (θ) as the angle of local x-axis with respect to the ground. As illustrated in Figure 3-14 (b), if we rotate the accelerometer about the y-axis, the angle between the local x-axis and the global X_w -axis is the pitch. We also denote that the roll (φ) is the angle of local y-axis with respect to the ground. As illustrated in Figure 3-14 (c), this means if we rotate the accelerometer about its x-axis, the angle between the local y-axis and the global Y_w -axis is the roll.

In orientation computation using the accelerometer, the local coordinate axis refers to the coordinate axis with its origin at the center of the associated sensor. An additional global coordinate axis is required and refers to the fixed coordinate axis that is the horizontal plane normal to the earth's gravity. When the accelerometer is in static condition, the gravity vector can be derived from the three orthogonal accelerations (Figure 3-15):

$$a = \sqrt{a_x^2 + a_y^2 + a_z^2} \approx g, \quad (4)$$

Here, a_x , a_y , and a_z represents the acceleration components of the gravity vector measured in its local coordinate x-, y- and z-axis respectively. Let a_{xy} , a_{yz} , and a_{zx} be the projection of the gravitational acceleration vector ($\mathbf{g}$) to the local plane xy, yz, and zx respectively. We have:

$$a_{xy} = \sqrt{a_x^2 + a_y^2} \quad (5)$$

$$a_{yz} = \sqrt{a_y^2 + a_z^2} \quad (6)$$

$$a_{zx} = \sqrt{a_z^2 + a_x^2} \quad (7)$$

From Figure 3-15, we have: (1) the pitch (θ) is complement with the angle between vector $\mathbf{a}_x$ and vector $\mathbf{g}$; (2) the angle between vector $\mathbf{a}_x$ and vector $\mathbf{g}$ is complement (their sum is equal to 90°) with the angle between vector $\mathbf{a}_{yz}$ and vector $\mathbf{g}$ (α). Therefore, the pitch (θ) is equal to the angle between vector $\mathbf{a}_{yz}$ and vector $\mathbf{g}$ (α). Thus the expression to calculate the pitch is:

$$\theta = \alpha = \arctan\left(\frac{a_x}{a_{yz}}\right) \quad (8)$$

$$\rightarrow \theta = \arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \quad (9)$$

Similarly, we have: (1) the roll (φ) is complement with the angle between vector $\mathbf{a}_Y$ and vector $\mathbf{g}$; (2) the angle between vector $\mathbf{a}_Y$ and vector $\mathbf{g}$ is complement with the angle between vector $\mathbf{a}_{XZ}$ and vector $\mathbf{g}$ (β). Therefore the roll (φ) is equal to the angle between vector $\mathbf{a}_{XZ}$ and vector $\mathbf{g}$ (β). Thus we infer the expression to calculate the pitch:

$$\varphi = \beta = \arctan\left(\frac{a_y}{a_{zx}}\right) \quad (10)$$

$$\rightarrow \varphi = \arctan\left(\frac{a_y}{\sqrt{a_z^2 + a_x^2}}\right) \quad (11)$$

Figure 3-15 Coordinate frames (body frame and world frame)

The angle conversion from acceleration measurement can be carried out in the embedded firmware as well as the PC software. In this project, the conversion is done by the PC software to relieve the computation burden on the firmware. The singularities of the accelerometer as a tilt sensor are listed below. The singularities are compensated with the OLEs using a kinematic model presented in the next chapter.

- The accelerometer is unable to measure the heading orientation, which is the rotation about its local z-axis.
- The accelerometer is unable to measure the rotation about its world frame's Z_w -axis.
- From equation (9), when $a_y^2 + a_z^2$ is approximately zero, the pitch is undetermined. This means, a singularity occurs when a_y and a_z are close to zero, or the local yz-plane is perpendicular to the gravitational vector.
- Similarly, from equation (11), when $a_z^2 + a_x^2$ is approximately zero, the pitch is undetermined. This means, a singularity occurs when a_z and a_x are close to zero, or the local zx-plane is perpendicular to the gravitational vector.

3.4 Firmware Programming

Firmware for the wearable sensor network is designed and implemented to control behaviors of the network's components, such as data acquisition, transmission, protocol, encoding, and decoding. Each node of the network, including sensors and the Data Concentrator, has a microcontroller (Microchip's dsPIC33 family) which can run the embedded software. To capture the motion of the entire human body, 15 SmartSuit sensing modules, three IMUs and one Data Concentrator are required. However within the scope of this project, a prototype network of three SmartSuit sensing modules and one Data Concentrator will be developed to capture the motion of human upper extremities. The full network version for the entire body will be the future work. When the sensor network is in operation, there are five main embedded processes are running concurrently. Three of these control sensor nodes. And two processes control the acquisition and transmission in the Data Concentrator. The processes are designed to ensure that the motion data is transmitted from the sensors to the PC in reliable and fast manner. This section explains the main flow of the firmware controlling the sensor network, which is illustrated by the flowchart in Figure 3-16.

Figure 3-16 Flow chart of the firmware for a Data Concentrator and three sensor nodes for arm motion capture

3.4.1 Node process

All wearable sensor nodes in the CAN bus network run the same processes. The processes include reading data from the OLE and accelerometer and sending the data to CAN bus. The embedded program in the node has the following workflow. At first, initialization is performed to reset all parameters of Serial Peripheral Interface (SPI) bus, used for CAN bus transmission, and Quadrature Encoder Interface (QEI), used for communication with the OLE, and Inter-Integrated Circuit (I2C) bus, used for communication between chips, to the default configuration. This is followed by reading the linear motion data of OLE from POSCNT register, which is a special counter used to count the number of lines the encoder have travelled. The expression is $\text{Encoder_Position} = \text{POSCNT_b15} + \text{POSCNT}$. Accelerations (along x-, y-, and z-axis) of the accelerometer are also acquired from the registers reserved for the task, by the function `read_register(<memory location of the respective register>)`. The data is then stacked to SPI register ready for CAN bus communication, using the function `SPIByteWrite()`. Each node has a specific SPI register, which is fixed for all runs. The loop of reading and stacking data keep going as long as the node is in operation. By measuring the period of an indicating LED on the sensing module board using an oscilloscope, the actual time interval between two consecutive processes is approximately 10 ms. Hence, after every 10 ms, the sensor node queues its acquired motion data to the CAN bus.

3.4.2 Data Concentrator process

There are two parallel processes running in the Data Concentrator. The main process, which is in charge of polling motion data from the sensing nodes through CAN bus, begins with initialization to clear all flags and variables, and reset all the necessary components, such as oscillators, interrupts, CAN bus, UART (Universal asynchronous receiver/transmitter) used for serial communication, registers and so on. The procedure then requests CAN bus transmission by calling the function `ecanWriteMessage()` to write a message to ECAN (Enhanced CAN) message buffer 0 to 5. Specific CAN buffers are booked for keeping the motion data received from the sensing node, as listed below.

- CAN buffer located at 0x0006FFFF is reserved for sensing node 1.
- CAN buffer located at 0x000AFFFF is reserved for sensing node 2.

- CAN buffer located at 0x000EFFFF is reserved for sensing node 3.

Each CAN beffer location contain variables, which host sensing data:

- OLE reading is stored in `ecan1msgBuf[31][3]`.
- Acceleration along x-axis is stored in `ecan1msgBuf[31][4]`.
- Acceleration along y-axis is stored in `ecan1msgBuf[31][5]`.
- Acceleration along z-axis is stored in `ecan1msgBuf[31][6]`.

For example, if the acceleration is along the y-axis of the third sensor, then it is read from the variable `ecan1msgBuf[31][5]` of CAN buffer located at 0x000EFFFF. The location of CAN buffer is specified by the function `ecan1WriteTxMsgBufId(buf, txId, ide, remoteTrans)`. The data collected from each sensing node are decoded (by linking the low bytes and high bytes into a single string), and stored in memory for transmission to PC. A delay after each node's data are read ensures the complete package of data is received. The routine is carried out for every sensor from the first to the last. The loop iterates this polling task from the first node to the last node.

The other process, parallel to the main one, running in the Data Concentrator is indeed the interrupt `_U1RXInterrupt`. This hardware interrupt was employed to avoid wasting the processor's valuable time in polling loops, waiting for external events. It shares the resource of the microcontroller with the main process, and uses us the idle time in the main process to perform its own task. This interrupt handles the transmission task between the sensor network and the end application (PC). The procedure first processes the command received from the PC through the RF board. Following the convention, is the command indicates the character 'I' in its request string, the Data Concentrator will pick the motion data of the sensing node 1, which have been collected by the main process. And similarly, 'J' imply the request for sensing node 2's data, and 'K' indicates the request for sensing node 3's data. This request-reply manner reduces the data load for wireless transmission, which means only one node's data are transferred each time. The sensing data are arranged into a string, whose format is `E****X****Y****Z****`, in which E stands for encoder linear movement, X, Y and Z stand for acceleration along x-, y- and z-axis respectively. When the PC receives the message, the decoding would be straight forward. The data are queued for transmission by

the function `WriteUART1()`. The RF board will send the data to PC wirelessly through Zigbee protocol. This interrupt is triggered by the processor's clock. This is in an unconditional loop as long as the microcontroller in the Data Concentrator is running. By measuring the period of an indicating LED on the Data Concentrator board using an oscilloscope, the actual time interval between two consecutive overall processes, each of which includes the main process and the interrupt, is approximately 22 ms. Hence, after every 22 ms, the Data Concentrator sends the data to the PC if it receives the request.

3.5 Summary

In this chapter, the hardware and firmware of the wearable sensor system for SmartSuit are discussed in details. The chapter starts with a review of all the fundamental network topologies, including point-to-point, ring, star, mesh, hierarchical, and bus. Each topology is appraised for the task of connecting our sensing modules into a SmartSuit sensor network, while maintaining the fast data transmission and wearable features of a body sensor network. The bus topology is chosen because it offers a daisy-chain fashion for connecting all the sensor nodes. The distributed bus network is a perfect match to the distributed structure of human limbs and body. The data are transmitted from the sensor nodes to the Data Concentrator in the most direct manner. The wiring of the bus network does not hinder the free motion of users. The CAN bus, which is a mature bus topology offering high speed of transmission up to 1Mb/s, is implemented with three sensing modules and one Data Concentrator to capture the motion of human arm. The accelerometer, one of the two sensing elements of each node, integrated in each sensing module as a tilt sensor to provide two additional tilting angle of the module. The tilt sensor suffers from many singularities, which, however, are compensated with the OLE using a kinematic model presented in the next chapter. Finally the firmware embedded in the microcontroller of each node in the sensor network, capturing the motion of a human arm, is proposed and discussed. The firmware in each sensor node acquires motion data from the OLE and the accelerometer and transmit to the Data Concentrator via the CAN bus. The sensing module firmware has an operating period of 10 milliseconds. The firmware in the Data Concentrator collects the motion data from the CAN bus, and then send them to the PC. The Data Concentrator firmware has an operating period of 22 milliseconds.

Chapter 4 Motion Processing Software for SmartSuit

Software is one of the most essential parts of any motion capture system. The requirements for the software are:

- 1) The software interfaces with the hardware system to acquire the motion data;
- 2) The software processes the data to obtain the kinematic information (and dynamic information if needed);
- 3) The software regenerates the captured motions, stores them in computer memory, and displays them graphically;
- 4) The software can interface with other motion modeling software and humanoid robotics systems.

The motion processing software provides a terminal for users to interface with the human motion hardware system, and adapts the recorded data to different fields of application. The software architecture was proposed as a framework for motion capture software. Figure 4-1 illustrates the components of the software architecture, which serves as a programming guide for developing the SmartSuit motion processing software.

Viewed from the motion capture hardware side, the entrance to the software is the motion data processing unit, which converts the raw data recorded from the Data Concentrator to positions, angles, velocities, accelerations. The motion data processing unit uses the calibration procedure as references. The filtering scheme smoothens the motion data. The data coming out from the processing unit are then adapted to kinematic data and dynamic data. The kinematic and dynamic information is either fed in to the Virtual Avatar Controller unit to generate more natural motion data or the Data File Converter unit. For demonstration task, the data are used to render motions of the virtual human for graphic display. In the Graphic Display unit, user can also create the virtual environment for virtual reality or computer game study. For interaction with robotics systems, the motion data are transferred into Microsoft Robotics Studio, which provide programming tools compatible to a wide variety of robot hardware and many programming languages, to control physical agents. For

cross-platform applications, the Converter converts motion data into standard motion capture data files, e.g bvh/bva, c3d. Commercial modeling and animation programs, for instances Maya, 3ds Max, RenderMan, Blender, Poser can load the motion standard data files to control the characters. The data files are also recorded for storage sake.

Figure 4-1 The Software architecture for human motion capture system

This chapter is devoted to the PC software design for SmartSuit. First, the interfacing methodology between the software and the hardware is discussed. In this project, RS232, USB and wireless transmission by Zigbee protocol are used. Biometric data, such as bone length and structure and skin characteristics, alter from person to person. Hence calibration procedure is required to estimate scaling factors that amending OLE data. Then, human body modeling, including a 7-DOF kinematic model for arms and data structure for human motion,

is studied. In addition the interaction between the SmartSuit software with other modelling software is explained. The medium file used for SmartSuit is 3ds file format (*.3ds). Graphics display algorithm using OpenGL, which is an open and very popular graphic library, is also presented. Finally different motion capture data formats, which are used in various tracking systems, are explained.

4.1 Motion Data Processing Unit

The motion data processing unit is the entrance of the software, which acquires data flow from the motion capture system. The processing unit then converts the raw data, including encoder readings from the OLEs and acceleration from accelerometers explained in section 3.4, recorded from the Data Concentrator to positions, angles, and accelerations for further usage.

Serial communication is one of the most popular and efficient data transmission methods, which is widely used in industries as well as research environments. This section starts with a brief introduction to the basic serial communication technique. The protocols and crucial data conveying issues are also discussed. The section then narrows down to the methods used in the SmartSuit: RS232, USB and wireless transmission by Zigbee protocol; the conversion from accelerometer data to joint angles; and finally discusses about the programming aspects for this type of interfacing.

4.1.1 Serial ports

A serial port is a computer interface that transmits data one bit at a time (that's why it's "serial"). In common use, serial port refers to ports that use a particular asynchronous protocol, for examples RS232, USB, and wireless through wireless stations on PCs and many serial ports in embedded systems. Most serial ports are bidirectional, meaning they can both send and receive data. In PCs, applications access most serial ports as COM ports. Some USB devices function as virtual COM ports, which applications can access in the same way as physical serial ports. Microcontrollers in embedded systems can use serial ports to

communicate with other embedded systems and PCs. Three types of serial communications, through RS232, USB and wireless transmission, were chosen to be the main way to transmit data from the motion capture hardware system to PCs in this project.

4.1.2 Protocol

A protocol is a set of rules that defines how computers manage communications. Serial communications must implement a low-level communication protocol and may also implement a higher-level message protocol.

(a) Communication protocols

A communication protocol defines how the bits travel, including when a computer can transmit, the bit rate, and in what order the bits transmit. The UART typically handles the details of sending individual bits and storing received bits on the serial port. A communication protocol can include the use of status and control lines. These lines can indicate when a transmitter has data to send or when a receiver is able to accept new data. The process of exchanging this information is called flow control. Hardware flow control uses dedicated lines for the signals. Devices can also use software flow control to provide the same information by sending defined codes, typically in the same path used for data.

(b) Message protocols

Serial communications often exchange messages that consist of blocks of data with defined formats. A message protocol can specify what type of data a message contains and how information is structured within the message. The computers in a network need a way to detect which computer is the intended receiver of transmitted data. Networks typically assign an address to each computer and include the receiver's address in each message. To enable a receiving computer to detect the start and end of a message, a message can include codes to indicate these events or a header that stores the message length. A message can also include one or more bytes that the receiving computer uses in error checking.

Figure 4-2 Flow chart of the procedure for acquiring data through serial port

4.1.3 Programming

The proposed programming procedure for interfacing with hardware devices through serial ports is listed below. The interfacing algorithm is depicted by the flowchart in Figure 4-2. The algorithm, which works for all serial communication methods including RS232, USB and wireless transmission, allows the software to create a port for data to flow into the PC. The algorithm is explained step by step as follows.

- (i) Create the object “com port”, which can use the defined functions of the class CComPort. The com port number is specified in this step.

- (ii) Create a Handle which takes care of data transmission through the serial port
- (iii) Set the serial communication parameters, discussed previously.
- (iv) Configure the communications port according to specifications set in (iii) in a device-control block (DCB) structure.
- (v) Manage communication timeouts.
- (vi) Write the predefined code to let the Data Concentrator know that the computer is available to receive data.
- (vii) Set buffers' properties.
- (viii) Read the data sent from the Data Concentrator and put them into the buffers.
- (ix) Share data among the operating processes (filtering, rendering, etc)

The communication baud rate is set to 115200 bits/s. Because each package of motion data is an array of 20 character elements, each of which has the size of 8 bits, the transmission provides a sampling rate of 720 Hz. This sampling rate allows the system to operate in real-time.

4.1.4 Conversion from raw data to angles

As presented in Chapter 3, the data transmitted from the hardware is in the following form for each node:

*E****X****Y****Z*****

This string received is called raw data, in which E stands for encoder, X, Y, Z stands for accelerations along three local x-, y-, and z- axes. From the working principle of accelerometers explained in Chapter 3, the roll and pitch orientations of the respective sensor node are calculated by:

$$\theta = \arctan \left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}} \right), \quad (12)$$

$$\varphi = \arctan \left(\frac{a_y}{\sqrt{a_z^2 + a_x^2}} \right), \quad (13)$$

Here, θ , φ , a_x , a_y , and a_z are pitch, roll, and acceleration along the three local x-, y-, and z-axes, respectively.

On the other hand, the encoder raw data are processed through the calibration procedure, which is discussed in the next section, to get the joint angle.

4.2 Calibration

Biometric data, such as the bone length and structure and skin characteristics, alter from person to person. This variation affects the accuracy and reliability of the motion capture process. Therefore a scale factor (SF) is inserted into equation (3) to compensate for biometric uncertainty, which may disturb the tracking accuracy. Thus we have

$$\alpha = SF(D_c) \cdot \frac{D}{R} \cdot \frac{360^\circ}{2\pi} \quad (14)$$

Where $SF(D_c)$ is a function of calibrating joint travel distance of the encoder head D_c :

$$SF(D_c) = 2\pi \cdot \frac{\alpha_c}{360^\circ} \cdot \frac{R}{D_c} \quad (15)$$

The scale factor is obtained through a calibrating process:

- (1) Mount an OLE around the joint to be calibrated;

- (2) Joint poses predefined gestures $\alpha_c = 0^\circ$;
- (3) D_c , the distance travelled by the encoder, is recorded from the OLE readings;
- (4) $SF(D_c)$ is computed by equation (15), with known α_c and D_c ;
- (5) Repeat from (1) to (3) with $\alpha_c = 30^\circ, 60, 90$, and so on until reaching the end of range of motion for the joint (every joint has different range of motion)
- (6) A look-up table is formed by the pairs of SF and D_c . Thus when using (14) to calculate the actual joint angle, $SF(D_c)$ with respect to each value of $D_c = D$ is interpolated from the table;

Figure 4-3 The user is posing a calibration gesture for the OLE at the elbow joint; The white board is marked with the angle that the forearm is supposed to pose, while the upper arm is kept vertical.

Figure 4-3 illustrates a calibration experiment when the user is posing a calibration gesture for the OLE at the elbow joint. The white board is marked with the angle that the forearm is supposed to pose, while the upper arm is kept vertical. Table 4.1 is a sample calibration table

with elbow joint angle, encoder travelled distance, calculated scale factor, and joint angle without calibration. Figure 4-4 show the calibration curve for the Scale Factor.

Figure 4-5 illustrates the comparison between the calibrated data and the data without calibration. It can be observed that the scale factor bends the linear relationship between the encoder reading and the resulted joint angle (which is obtained with mechanical rigid links and joint, refer to the experiment in section 5.1) to a non-linear curve with larger gradient when the joint gets closer to the limit of the range of motion. The range of motion for human elbow joint is from 0° to 145° .

TABLE 4.1
SAMPLE CALIBRATION FOR THE OLE AT THE ELBOW JOINT

Calibrated joint angle (degree)	Encoder travel distance (cm)	Scale Factor	Joint angle without calibration (degree)
0	0	1	0
30	1.41	1	30
60	2.8	1.007	59.6
90	3.91	1.084	83
120	4.86	1.161	103.4
140	5.22	1.261	111.1

Figure 4-4 Sample calibration curve for capturing elbow joint motions

Figure 4-5 Comparison between calibrated data and data without calibration

4.3 Human Arm Kinematic Model

The accelerometers suffer from singularities as discussed in section 3.3. While other singularities can be eliminated by filtering algorithms, the accelerometers' inability to capture the heading angle and rotations about the vertical axis cannot be solved. Since gravity cannot be used to calculate changes in the heading of accelerometers, the yaw angle (or heading orientation) cannot be determined. Thus the accelerometers can measure only two tilting angles of a rigid body. Any rotation by the rigid body about the vertical axis cannot be captured. For example, when the arm moves horizontally, as depicted in Figure 4-6, from position P_1 to position P_2 , accelerometers alone are incapable of tracking the motion. By attaching OLEs around the body joints, including one at the shoulder, one at the elbow, and one at the wrist, the sensing singularity of the accelerometers can be compensated using a 7-DOF model of human arm based on Denavit-Hartenberg representation. This model map sensor data acquired from the SmartSuit sensor network into joint angles, and is used to regenerate the captured human motion. Figure 4-7 illustrates the position of the sensing modules attached on the human arm.

Figure 4-6 Denavit-Hartenberg coordinate system for human arm

Figure 4-7 Position of the sensing modules on the human arm

4.3.1 Establishment of D-H coordinates and parameters for the human arm

In this thesis, the human arm movements are assumed as articulated motions of a rigid link-joint mechanism. A 7-DOF articulate arm model is used to kinematically describe the movement of a human arm, excluding the fingers. The Denavit-Hartenberg coordinate frames can be established by the following procedure (Figure 4-6) [22]:

- (1) Establishing the world coordinate system: A Cartesian coordinate system $X_w Y_w Z_w$ is assigned to the pelvis of the human model with X_w pointing toward the front of the model, and Y_w pointing upward.
- (2) Let i be the index of the degree of freedom of each joint. For each i from 0 to 6, the following steps from (3) to (6) are performed:
- (3) Establishing the joint axis: Align the z_{i-1} axis with the axis of the i^{th} joint's motion.
- (4) Establishing the origin of the i^{th} coordinate system: The origin of the $x_i y_i z_i$ coordinate system is located at the intersection of the z_{i-1} axis and the common normal between the z_i and z_{i-1} axes
- (5) Establishing x_i axis: the x_i is normal to z_{i-1} axis
 - a) If z_i and z_{i-1} do not intersect, the x_i axis is along the common normal to z_i and z_{i-1} , and its direction is defined from z_{i-1} -axis toward the z_i -axis.
 - b) If two joint axes z_i and z_{i-1} are parallel and not intersecting, the direction is defined from z_{i-1} -axis to z_i -axis
 - c) If z_{i-1} -axis is collinear with z_i -axis, x_i -axis is chosen randomly in the plane perpendicular to the z -axes
 - d) If z_{i-1} and z_i intersect, x_i -axis is perpendicular to both z_{i-1} -axis and z_i -axis with any direction
- (6) Establish y_i -axis: y_i -axis is constructed according to right-hand rule, so that $x_i y_i z_i$ is a Cartesian orthogonal coordinate system

The Denavit-Hartenberg parameters are denoted as θ_i , d_i , a_i , and α_i , each set of which associate with each link of the human model. These parameters constitute a sufficient set to completely describe the kinematic configuration of each link.

- θ_i is the joint angle from the x_{i-1} axis to the x_i axis measured about z_{i-1} axis. Its positive direction is counter-clockwise.
- d_i is the distance from frame $i-1$'s origin to the intersection of z_{i-1} -axis with x_i -axis.

- a_i is the distance from the intersection of z_{i-1} -axis and x_i -axis to the i^{th} frame's origin.
- α_i is the offset angle from z_{i-1} -axis to z_i -axis measured about x_i -axis. Its positive direction is counter-clockwise.

Note from Figure 4-6 that frame 0, frame 1 and frame 2, representing the 3 DOFs of the shoulder complex, have the common origin at center of the shoulder joint; frame 3 is located at the center of elbow joint; and similar to shoulder, frame 4, frame 5 and frame 6 have the same origin at center of wrist joint to describe the wrist motion. The transformation matrix from frame $(i-1)$ to i is formulated, based on the coordinate arrangement in Figure 4-6, as

$$T_i^{i-1} = \begin{bmatrix} c\theta_i & -c\alpha_i \sin\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (16)$$

The specific values of these D-H parameters are listed in Table 4.2. This kinematic model allows us to compute the translation of any frame with respect to any other frame, as long as the parameters θ_i , d_i , a_i , and α_i are known, by the kinematic chain equation:

$$T_n^m = T_{m+1}^m \cdot T_{m+2}^{m+1} \dots T_n^{n-1} \quad (17)$$

Note that d_i and a_i , biometric factors (Figure 4-8), either link length or 0 (when the frames have the same origin); and that $d_3 = a_3 =$ upper arm length, and $d_4 = a_4 =$ forearm length. The angles between arm joint rotation axes: $\alpha_0 = \alpha_2 = \alpha_4 = 90^\circ$; $\alpha_1 = \alpha_5 = \alpha_6 = -90^\circ$; and $\alpha_3 = 0$.

TABLE 4.2
D-H PARAMETERS

Joint i	θ	α	d	a
0	θ_0	90	d_0	a_0
1	θ_1	-90	0	0
2	θ_2	90	0	0
3	θ_3	0	d_3	a_3
4	θ_4	90	d_4	a_4
5	θ_5	-90	0	0
6	θ_6	-90	0	0

Figure 4-8 Biometric parameters

4.3.2 Joint angle mapping from sensor data

All the joint angles, θ_0 , θ_1 , θ_2 , θ_3 , θ_4 , θ_5 , and θ_6 , need to be determined from the captured sensor data with SmartSuit sensing modules worn on the shoulder, elbow and wrist. At the shoulder, the absolute angles θ_{w0} and θ_{w1} (Figure 4-9) are given by the accelerometer mounted on the user's upper arm (of the 1st sensing unit, node 1). It is noted that θ_{w0} and θ_{w1} are the angles of the transformation matrixes from the world coordinate system to the frame 0, T_0^w , and frame 1, T_1^w , respectively. The OLE of the same node measures θ_2 , which is the angle of the transformation matrix from frame 1 to frame 2, T_2^1 .

Figure 4-9 Absolute angles θ_{w0} and θ_{w1} are given by the accelerometer mounted on the user's upper arm

The relative angle between the upper and forearm, provided by the OLE of node 2, indicates elbow joint angle θ_3 (Figure 4-10), the variable of the transformation matrix from frame 2 to frame 3, T_3^2 .

Figure 4-10 Elbow joint angle between the upper and forearm, θ_3 , provided by the OLE of node 2

The wrist rotation θ_5 , a singularity to the accelerometer, is recorded by the OLE of node 3. The angle θ_5 is the angle of the transformation matrix from frame 4 to frame 5, T_5^4 . The accelerometer attached on the hand tracks the absolute angles θ_{w4} and θ_{w6} (Figure 4-11) of the transformation matrixes from the world coordinate to frame 4, T_4^w , and to frame 6, T_6^w .

Figure 4-11 absolute angles θ_{w4} and θ_{w6} are tracked by the accelerometer attached on the hand

There remain θ_0 , θ_1 , θ_4 , and θ_6 to be calculated. The angle θ_0 is the rotation angle from the world coordinate system to the frame 0, so it is equal to θ_{w0} . Now we have:

$$T_1^w = T_0^w T_1^0 \quad (18)$$

Hence,

$$T_1^0 = [T_0^w]^{-1} T_1^w \quad (19)$$

Transformations T_0^w and T_1^w are known because θ_{w0} and θ_{w1} are known; T_1^0 would be determined by (19), and so is θ_1 . To find θ_4 , which is the variable of the matrix T_4^3 , we have

$$T_4^w = T_0^w T_1^0 T_2^1 T_3^2 T_4^3 \quad (20)$$

i.e.
$$T_4^3 = [T_3^2]^{-1} [T_2^1]^{-1} [T_1^0]^{-1} [T_0^w]^{-1} T_4^w \quad (21)$$

As T_0^w , T_1^0 , T_2^1 , T_3^2 and T_4^w are obtained from θ_{w0} , θ_1 , θ_2 , θ_3 and θ_{w4} , T_4^3 can be computed from (21), and so is θ_4 . Now the transformation from the world coordinate system to Frame 6 can be expressed as

$$T_6^w = T_0^w T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 \quad (22)$$

Hence

$$T_6^5 = [T_5^4]^{-1} [T_4^3]^{-1} [T_3^2]^{-1} [T_2^1]^{-1} [T_1^0]^{-1} [T_0^w]^{-1} T_6^w \quad (23)$$

Based on the result of T_4^3 , the transformation T_6^5 can be determined from (23) and θ_6 can also be computed using T_6^5 .

Thus the kinematic model allows all the joint angles θ_0 , θ_1 , θ_2 , θ_3 , θ_4 , θ_5 , and θ_6 to be calculated. The joint angles can be directly fed into the virtual human to regenerate the captured motion, or stored for further analysis. Note that to avoid multiple solutions from the trigonometric equations in calculating the joint angle θ_i , the range of motion of the human joint is used to set the limit for the joint angle. Figure 4-12, Figure 4-13, and Figure 4-14 illustrate the range of motion of the human shoulder, elbow and wrist, according to the study by Luttgens et al [23].

Figure 4-12 Range of motion for human shoulder [23]

Figure 4-13 Range of motion for human elbow [23]

Figure 4-14 Range of motion for human wrist [23]

4.4 Human Motion Data Structure – a Hierarchical Model

This section discusses the hierarchical model of human body used to store motion data and to regenerate motion captured from the SmartSuit. The software uses object oriented method to render the 3D human avatar. The main data structures for the objects in the program are also explained.

Figure 4-15 Human link model (refer to Figure 4-16 for the complete hierarchical link model)

a) Modeling the entire human body motion

The model to capture the human motion using SmartSuite is illustrated in Figure 4-15. Each arm is modeled by three links, namely the upper arm, the forearm, and the hand. The arms are attached to a rigid body trunk, which is the thorax by 3-DOF shoulder joints. Each leg is also modeled by three links, consisting of the upper leg, the lower leg, and the foot. Similar to the arms, the legs are attached to the trunk via the pelvis. The mobility of the model may be calculated utilizing Kutzbach's equation for spatial open chains:

$$F = 6N - \sum_{i=3}^5 i \cdot j_i \quad (24)$$

Where, N is the number of movable links; i is the class of the joint (the number of imposed constraints, $i=6-f$, where f is the number of DOF), j_i is the number of joints of class i .

From Figure 4-15, it can be seen that the human model has 16 movable links; 12 joints of 3rd class (with three DOFs); four joints of 5th class (with one DOF). Therefore,

$$F = 6 \cdot 16 - 3 \cdot 12 - 5 \cdot 4 = 40 \text{ (DOF)}.$$

b) Primitive Object:

The human model is then constructed by composing the primitive objects into a virtual human, which is shown in Figure 4-15. The primitive objects are the most basic objects, such as joints and links, which together form the whole body structure. The primitive objects represent graphical relationship which follows the connection of all body parts. The general data structure of a primitive object is as follows:

```
struct pObject{
    Shape ObjectShape;
    Size ObjectSize;
    string ObjectName;
    int ObjectID;
    Vector ObjectPos;
    Vector ObjectRot;
    Vector ObjectAcc;
    Matrix Coordinate;
    pObject *Next;
    pObject *Childs;
    pObject *Parent;
};
```

*ObjectShape: The shape of the object, including cylinder, sphere, diamond, ellipsoid

*ObjectSize: the dimensions of the object based on the shape

*ObjectName, the name of the object, e.g. LowArm, Hand, Trunk, Thigh, Head, etc

*ObjectID: the Identification number of the object

*ObjectPos: the current position of the object

*ObjectRot: the current rotation of the object

*Coordinate: the matrix specifying the local coordinate system attach on to the object

*Next: the pointer pointing to the next object

*Childs: the pointer pointing to the childs of the object in the hierarchical model

*Next: the pointer pointing to the parent of the object in the hierarchical model

Figure 4-16 Hierarchical human link model

Every body part that constitutes the virtual human model is defined in the above structure. The pointers (*Next, *Childs, *Parent) will combine all the links to form the hierarchical model of the virtual human as shown in Figure 4-16. The state of each link is described by the relative pose with respect to the coordinate frame of its parent. In the model, each node consists of transformation (rotation and translation), size (depending on node's shapes), and geometric shape information. The transformation defines a coordinate system with respect to the parent coordinate system. In the case of the root node, the transformation means a rotation and translation on the reference frame. On the contrary, other nodes have only the relative rotation for their parent. Each node also includes the information of the kinematic constraints such as D.O.F. constraints and joint angle limitation, e.g. elbow joint cannot rotate beyond 180° . Size of geometric shape for each link handles variety of body proportions for different human models.

Similar joint objects are linked together to form a hierarchy of joint models as depicted in Figure 4-17. The D.O.F for each joint is also shown in the model, which is the number embraced by the double brackets. The child nodes inherit the properties of the parents. By the hierarchical models, users can easily transverse the tree using searching algorithms to edit data in each node.

Figure 4-17 Hierarchical human joint model

4.5 Graphics Display

The motion information, after stored in the objects' data structures, is illustrate in a graphics display mode. A graphical human model is drawn in 3D virtual space to regenerate the captured motion of human. This portion discusses the programming issues of the graphics rendering, including two main techniques, such as using OpenGL and using graphics modeling software.

4.5.1 Rendering graphics using OpenGL

In order to render the virtual human to demonstrate the captured motion, we used OpenGL as the graphics library. OpenGL is chosen because it is the industry's most widely used, supported and best documented 2D/3D graphics API making it inexpensive and easy to obtain information on implementing OpenGL in hardware and software. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. Our software utilizes instancing of geometric data to reduce their memory footprint. Instancing allows an application to create a single representation of the geometric data for each type of object used in the scene. In the virtual model, the links of the body may be represented as instances of a single geometric description of a cylinder (or ellipsoids) with different dimensions, and joints may be represented as instances of a single geometric description of a sphere, combined with a modeling transformation to place each object in the correct location in the scene.

Instancing introduces extra complexity into the picking operation. If a matrix is associated with the link (or joint) geometry, the software cannot determine which of the instances of the links (or joints) has been picked. OpenGL solves this problem by maintaining a stack of matrices. This allows an application to associate matrix each stage of its hierarchy. Two functions which allows to push and pop matrix stacks: `glPushMatrix()` and `glPopMatrix()`. The `glPushMatrix()` function copies the current matrix and pushes it onto the stacks. The function `glPopMatrix()` pops off the top matrix on the stack and discards its contents. All other matrices in the stack are moved up one position.

Figure 4-18 Algorithm of rendering virtual human model using OpenGL

For illustration, the rendering hierarchy in Figure 4-18 explains the entire procedure for drawing one frame of the scene which displays the virtual human. Note that the hierarchy is the combination of the two hierarchical models shown in Figure 4-16 and Figure 4-17. The procedure begins with the function that render the pelvis, then `glPushMatrix()` to

remember the world coordinate system. The pelvis lower and upper trunk joints, thorax are then drawn (with the necessary translations and rotations). The function `glPushMatrix()` are called again to remember the current position. The left shoulder, left upper and lower arm, and hand are plotted. In order to render the head and right part of the upper body, the function `glPopMatrix()` is call to remove the top matrix on the stack. The same sequence is carried out to draw the entire body. Figure 4-19 depicts the human graphical models using the proposed algorithm with different link geometries: stick figures, diamond shapes, and ellipsoids.

Figure 4-19 Resulting virtual avatar rendered using the proposed algorithm

4.5.2 Interacting with modeling softwares

An alternative way to render and manipulate the virtual human is to use a modeling software (3ds Max, Maya, Blender, etc). For our project, 3ds Max is used to import virtual avatars into the viewer. The same way is used to render virtual environments in different applications. The environment is designed in 3ds Max, then imported into our software's viewer.

The file formats of 3ds Max is *.3ds, used as the bridge between our software and 3ds Max. The 3ds file format is made up of chunks, which describe what information is to follow and what it is made of, its ID and the location of the next block. The next chunk pointer is relative to the start of the current chunk and in bytes. The binary information in the 3ds file is written with the least significant byte coming first in an integer. For example: 4A 5C (2 bytes in hex) would be 5C high byte and 4A low byte. In a long form it would be: 4A 5C 3B 8F where 5C4A is the low word and 8F 3B is the high word. A chunk is defined by Table 4.3.

TABLE 4.3
BYTE STRUCTURE OF A CHUNK

start	end	size	name
0	1	2	Chunk ID
2	5	4	Pointer to next chunk relative to the place where chunk ID is, which means the length of the chunk

Chunks have an imposed hierarchy that is identified by its ID. A *.3ds file has the primary chunk ID 4D4Dh. This is always the first chunk of the file. Within the primary chunk are the main chunks. The data structures, used to store the information obtained from the *.3ds file, are presented in the Appendix B. The data structures are then used to reproduce the object loaded from the *.3ds file. Those variables can also be controlled by other source. This give us the ability to manipulate the *.3ds model using the data captured by the SmartSuit system.

4.6 Motion Capture Data Formats

Photos taken by cameras can be saved into several standard file formats, such as bmp, jpeg, tiff, png, etc. Motion pictures taken by video recorder can also be saved into standard file formats, e.g. wmv, avi, mp4, mov. Similarly, human motion captured by sensor systems can be saved into standard file format, such as BVA/BVH, ASF, and C3d.

Since the application domain of motion capture data is growing to various fields such as game, animation, motion picture, medicine, rehabilitation, etc, the needs for intermediate means for data interchange among the application are indispensable. Several standard file formats have been invented for storing motion capture data, for instance BioVision Hierarchy (BVA/BVH) by BioVision, a motion capture services company, Acclaim Skeleton ASF/AMC, or C3D by Motion Lab. Table 4.4 lists various motion capture data files which have been in use currently.

Since each format was created to support an associated motion tracking system, it contains enough data to regenerate the captured human motion, or at least display an animation. An interesting question may be raised: “is any one format better than another?” The response to this question depends on the targeted application of the motion data, which can vary from analysis, editing or simply playback. For instance, when concerned only with displaying an animation, the desirable aspects of the data are compactness and absolute positions for limbs, which eliminates the need for a hierarchical rendering algorithm. This means that it is both quick to load and display frames to the screen.

TABLE 4.4
AVAILABLE COMMERCIALIZED MOTION CAPTURE DATA FORMATS

File Extension	Associated Company/Description
ASC	Ascension
ASF/AMC	Acclaim
ASK/SDL	BioVision/Alias
BVA/BVH	BioVision
BRD	LambSoft Magnetic format
C3D	Biomechanics, Animation and Gait Analysis
CSM	3ds Max, Character Studio
DAT	Polhemous
GTR/HTR/TRC	Motion Analysis
MOT/SKL	Acclaim - Motion Analysis

However, since in most motion capture systems the raw data are not accurate, the data need to be edited. Hence, in many cases, the accessibility for motion editing is more important for the properties of displaying animations. The most critical property for motion editing is a hierarchical structure so that the bone motion can be processed locally and regardless of the changes made to parent limbs. The child objects (bones and joints) always maintain a specified transformational interdependence. Most of the modern motion capture formats are based on hierarchical structures, in which ASF/AMC, BVH and C3D formats are the most popular formats nowadays. Of these three file formats, the hierarchy of BVH files, the oldest format, is the most compact and succinct, due to their simplicity. Nevertheless, this also means that the format does not contain as much information as the ASF/AMC or C3D data. The problem with the BVH format lies at its assumption that bone vector comprises the

length and direction only. However, both ASF/AMC and C3D formats define rotation, length and direction for bone vectors. Motion editing process can be improved by details about the ambience in which data is captured. While BVH format does not consist of any of these details, ASC/AMC and C3D formats provide a better level of success, by including axis of gravity, the metric units used, and channel limits, etc. Because of the hierarchical characteristic, the rendering algorithm needs a recursive procedure draw the motion to the screen. Thus, the rendering of these hierarchical formats is more expensive than simply dumping vertex coordinates into the graphics pipeline. In many cases, after motion editing, files are converted to a specialized format, which is used only for displaying task. This section discusses two motion capture file formats, BioVision and Acclaim, which are used in SmartSuit software to enable our system to interact with other platforms.

4.6.1 BioVision Hierarchy (BVA/BVH)

The BVH format is the updated version of the BVA format with the add-in of a hierarchical data structure representing the skeleton's bones. The BVH format includes two parts where the first section details the hierarchy and initial pose of the skeleton and the second section describes the channel data for each frame. The example segment of BVH file in Figure 4-20 is used for explanation of the BVH format in the rest of this section. The hierarchical portion of the structure starts with the keyword `HIERARCHY`, which is followed by the keyword `ROOT` and the name of the bone that is the root of the skeletal hierarchy, e.g. Hips. This is normally where the coordinate system is located. The `ROOT` keyword indicates the start of a new skeletal hierarchical structure. Although the BVH file usually has only a single skeleton defined per file, it is capable of containing many skeletons. The remaining structure of the skeleton is defined in a recursive nature where each bone's definition, including children, is embraced in braces, which is bounded on the previous line with `JOINT` (or `ROOT` in the case of the root bone) followed by the name of the bone. Furthermore, although the hierarchical indentation is not necessary, it does assist in making the file more readable. Within the definition of each bone, the first line, started by `OFFSET`, shows the translation of the origin of the bone with respect to its parent's origin (or globally in the case of the root bone). For determining the bone length, the first child offset definition is used to infer the parental bone information and treat the offset data for other child nodes simply as offset values. The next line of a bone's definition is prefixed with `CHANNELS` which defines the DOFs for the

current bone. The concatenation order of the Euler angles when creating the bone's rotation matrix needs to follow the order depicted in the CHANNEL section. As the Euler order is specified for each bone, it is possible to have different orders for different bones. After the OFFSET and CHANNEL lines, the next non-nested lines in the bone definition are used to define child items, starting with JOINT. For end-effectors, "End Site" tag is used.

```

/*****
HIERARCHY
ROOT Hips
{
    OFFSET 0.00 0.00 0.00
    CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
    JOINT LeftHip
    {
        OFFSET 3.430000 0.000000 0.000000
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT LeftKnee
        {
            OFFSET 0.000000 -18.469999 0.000000
            CHANNELS 3 Zrotation Xrotation Yrotation
            JOINT LeftAnkle
            {
                OFFSET 0.000000 -17.950001 0.000000
                CHANNELS 3 Zrotation Xrotation Yrotation
                End Site
                {
                    OFFSET 0.000000 -3.119999 0.000000
                }
            }
        }
    }
    ...
}
MOTION
Frames: 20
Frame Time: 0.033333
0.00 39.68 0.00 0.65 ...
...
*****/

```

Figure 4-20 Sample segment of the BVH file

The second section of a BVH file, which is denoted with the MOTION, contains the number of frames in the animation, frame rate and the channel data. The line containing the number of frames starts with "Frames:", followed by a positive decimal integer, the number of frames. "Frame Time:" represents the duration of a single frame.

4.6.2 Acclaim ASF/AMC

The Acclaim format consists of two files, a skeleton file (Acclaim Skeleton File - ASF), and a motion file (Acclaim Motion Capture data - AMC). The sample of the ASF and AMC files are shown in Figure 4-21. This was done knowing that most of the time a single skeleton works for many different motions and rather than storing the same skeleton in each of the motion files it should be stored just once in another file.

```
:version 1.10
:name Bioskeleton
:units
    mass 1.0
    length 1.0
    angle deg
:documentation
    Data translated and provided by
    Biovision Motion Capture Studios
:root
    axis XYZ
    order TX TY TZ RZ RY RX
    position 0.0 0.0 0.0
    orientation 0.0 0.0 0.0
:bonedata
    begin
        id 1
        name hips
        direction 0.000000 1.000000 0.000000
        length 0.000000
        axis 0.00000 0.00000 0.00000 XYZ
        dof rx ry rz
        limits (-180.0 180.0)
                (-180.0 180.0)
                (-180.0 180.0)
    end
    begin
        id 2
        name hips1
    ...
    end
:hierarchy
begin
    root body_root1
    body_root1 hips
    hips hips1 hips2 hips3
...
end
```

Sample *.ASF file

```
:FULLY-SPECIFIED
:DEGREES
1
root -1.244205 36.710186 7.591899 0.958161 4.190043 -18.282991
hips 0.000000 0.000000 0.000000
chest 15.511776 -2.804996 -0.725314
neck 48.559605 0.000000 0.014236
head -38.332661 1.462782 -1.753684
leftcollar 0.000000 15.958783 0.921166
leftuparm -10.319685 -15.040003 63.091194
leftlowarm -27.769176 -15.856658 8.187016
lefthand 2.601753 -0.217064 -5.543770
rightcollar 0.000000 -8.470076 2.895008
rightuparm 6.496142 9.551583 -57.854118
rightlowarm -26.983490 11.338276 -5.716377
righthand -6.387745 -1.258509 5.876069
leftupleg 23.412262 -5.325913 12.099395
leftlowleg -6.933442 -6.276054 -1.363996
leftfoot -1.877641 4.455667 -6.275022
rightupleg 20.698696 3.189690 -8.377244
rightlowleg 3.445840 -6.717122 2.046032
rightfoot -8.162314 0.687809 9.000264
2
root -4.232432 36.723934 9.596100 -7.051147 1.678117 -7.711937
hips 0.000000 0.000000 0.000000
chest 31.863499 -19.017111 6.490547
...
```

Sample *.AMC file

Figure 4-21 Sample segment of *.ASF and *.AMC

a) Parsing the ASF file

In the ASF file a base pose is defined for the skeleton that is the starting point for the motion data. Each segment has information regarding the way the segment is to be drawn as well as information that can be used for physical dynamics programs, inverse kinematic programs or skinning programs. One of the peculiar features of the ASF file is the requirement that there be no gaps in the skeleton.. This has the effect of creating more skeletal segments than are usually found in other file formats. A limitation of the ASF definition is that only one root exists in the scene, this doesn't prevent a file from cleverly containing two skeletons attached to the root but it does make such a construction clumsy.

***`:version`** indicates the version of the skeleton definition.

***`:name`** allows the skeleton to be named something other than the file name.

***`:units`** denotes a section that defines the units to be used for various types of data.

***`:documentation`** section allows for the storage of documentation information that will persist from one file creation to the next.

***`:root`** section defines a special segment of the scene. This is the root segment of the skeleton hierarchy. The **`axis`** keyword in the root section defines the rotation order of the root object. The **`order`** keyword specifies the channels of motion that are applied to the root and in what order they will appear in the AMC file. The **`position`** and **`orientation`** keywords are each followed by a triplet of numbers indicating the starting position and orientation of the root.

***`:bonedata`** section contains a description of each of the segments in the hierarchy.

The segment definition is bracketed by a **`begin`** and **`end`** pair of keywords. Within the segment definition:

- **`id`** This is a number which provides a unique id for the segment
- **`name`** This gives the name of the segment. The segments that have numbers at the end will usually be the children of the segment with the same name but no number.
- **`direction`** This is the direction of the segment. This defines how the segment should be drawn, and defines the direction from the parent to the child segment(s).
- **`length`** The length of the segment. The direction and length of a segment determine the offset of the child from the parent, if there is a child of the segment.
- **`axis`** This gives an axis of rotation for the segment..
- **`dof`** This, which stands for 'Degrees of Freedom', specifies the number of motion channels and in what order they appear in the AMC file.
- **`limits`** This specification provides for putting limits on each of the channels in the dof specification.

b) Parsing the AMC file

The AMC file contains the motion data for a skeleton defined by an ASF file. The motion data is given a sample at a time. Each sample consists of a number of lines, a segment per line, containing the data. The start of a sample is denoted by the sample number alone on a

line. For each segment the segment name appears followed by the numbers in the order specified by the dof keyword in the ASF file. Figure 4-22 shows the GUI of the motion capture viewer unit of the SmartSuit software.

Figure 4-22 GUI of the Motion Capture Viewer

4.7 Operating flow

This section explains the main stream of the algorithm controlling the PC software. The main tasks for that program are

- 1) to interface with SmartSuit hardware to acquire the motion data via the PC serial communication methods, including RS232, USB and wireless transmission via Zigbee protocol, and then process the recorded data;

- 2) To obtain signals from input devices, such as mouse and keyboard, for users to interact with the software;
- 3) Regenerate the captured motion through the graphical display unit using OpenGL or models designed by commercial graphics software.

Figure 4-23 Flowchart for the software's main operating flow

As illustrated by the flowchart in Figure 4-23, the flow begins with the initialization routine, which clears all the variables, resets flags and rendering modes, create and configures necessary windows in the software GUI by the functions `glutCreateWindow` and `glutInitDisplayMode`, starts OpenGL Utility Toolkit (GLUT) by `glutInit(&argc, argv)`.

The program then create multiple functional threads, including mouse thread, visible thread, key thread, special thread, control light thread, and most importantly redraw thread and data collection thread. Multithreading is a technique in programming which allow multi-tasks to be processed in a certain interval of time. The threads exist within the same process and share resources such as memory and computational power. Mouse thread handles all the operation of the computer mouse in the software, for example rotating and moving the virtual 3D space for better view, pop up the control menu, and so on. Mouse thread's call-back function is looped by the standard OpenGL reiteration function `glutMouseFunc(mouse)`. Visible thread disables the graphical operations of an object when it is not visible to avoid unnecessary tasks. Visible thread's call-back function is reiterated by `glutVisibilityFunc(visible)`. Key thread is in charge for all the keyboard operations, such as controlling the virtual environment by keys. Key thread's call-back function is reiterated by `glutKeyboardFunc(key)`. Special thread monitors the uncertainties of the process. If there is any problem, it restarts all tasks by the function `glutPostRedisplay`. Special thread's call-back function is loop by `glutSpecialFunc(special)`. Control light thread handle all lighting and coloring issues. It make the virtual 3d environment more realistic. Control light thread's call-back function is repeated by `glutCreateMenu(controlLights)`.

Redraw thread is the main process for rendering the 3D environment and all the virtual objects appearing in the scene, such as human models, ground, gauges. During each loop of running, OpenGL makes the graphics controller in the computer hardware draw all the objects programmed in the 3D viewer. Each scene drawn after one loop is a frame. Since the interval between the loops is very small (several milliseconds, depending on how heavy is the work load), so what user see is just smooth motion (similar to animations). The redraw thread begins by setting up graphical factor, such as lighting, reflection, coloring, diffusion. The

mouse data and key board data are then imported from the mouse thread and key thread (for instance whether the user are trying to move the scene by mouse or to control some hot keys). The motion data are also imported from the data collection thread. Then the objects will be rendered with the all the imported data. If the flag for stick figure is on, a stick figure will be rendered with motion data. If the flag for diamond figure is on, a diamond figure will be rendered with motion data. If the flag for gauge is on, a 3x3 matrix of gauges will be rendered with motion data. The algorithm for drawing virtual human model has been explained in Section 4.5.1. This thread's call-back function is looped by the standard reiteration function `glutDisplayFunc(redraw)`.

Data collection thread is the only thread not controlled by OpenGL thread controller, which handle all threads discussed previously. The thread is begun in the main function `main (int argc, char **argv)` by the function `CreateThread()`, which creates a thread to execute within the address space of the calling process. The thread's call-back function first create a COM-port object by `CComPort* pComPort = new CComPort(3)`, in which the number embraced by the brackets is the COM port number which is assigned by Microsoft's Device Manager. Then the process collects motion data from the respective COM port using the algorithm presented in Section 4.1.3. After the raw data are recorded, the process decode them by segmenting the string in to meaningful numbers with respect to sensor information. The data are converted into angles as explained in Section 4.1.4. Calibration procedure, proposed in Section 4.2, is performed as a separate process if needed (normally when the user uses the system at the first time). After being processed, the data are published by putting them into public variables. This publishing allows other threads, especially Redraw thread, to read the motion data acquired from the SmartSuit hardware. The data can also me stored in standard data formats, which are discussed in Section 4.5, in a different process. The data-collecting process is reiterated by a `while(1)` loop within the call-back function.

Figure 4-24 shows the graphical user interface of the SmartSuit software, which plots the stick figure ready for displaying captured motion. The view can be adjust by computer mouse and keyboard for better observation. Figure 4-25 depicts the tools for instant numerical inspection, which are the three-by-three matrix of data gauges and graphs showing capture motion data.

Figure 4-24 SmartSuit graphical user interface

Figure 4-25 (a) Data gauges; (b) Data graphs

4.8 Summary

This chapter discusses PC software of SmartSuit. A software architecture is proposed for the programming tasks, from interfacing with SmartSuit hardware, to processing recorded motion data, and communicating with other software and data storage. The communication gateway between the software and the hardware is the serial communication ports (COM ports), which support RS232, USB and wireless transmission via Zigbee protocol. The protocol and algorithm for handling the COM ports are implemented so that the software can interface with the SmartSuit hardware no matter what COM port is being used. The transmission baud rate is set to 115200 bits/s, providing a sampling rate of 720 Hz. This sampling rate allows the system to operate in real-time. Calibration procedure, which forms a curve to estimate the scaling factor for adjusting the OLE data, is explained. The calibration process found that the scale factor bends the linear relationship between the encoder reading and the resulted joint angle to a non-linear curve with larger gradient when the joint gets closer to the limit of the range of motion. The human kinematic model is also proposed to map the sensor data into arm joint angles, which is retained by a hierarchical data structure of human motions. The algorithms that render the virtual environment for displaying and monitoring the regenerated motion captured data are discussed. There are two methods for rendering human models used in this software, including using standard OpenGL graphic functions and import model files created by other software such as 3dsMax and Maya. The study of standard motion data formats for SmartSuit cross-platform applications is also discussed. A comparison among the three most popular formats, which are BVH, ASF/AMC, and C3D, was studied. Structures of the two most popular motion files, BVA/BVH and ASF/AMC, are explained in details. This knowledge enables SmartSuit software to add the function of reading and writing the motion capture data formats. Finally, the main operating flow, including multiple threads that handle different programming aspects, of the software is discussed. The resulting graphical user interface of the SmartSuit software is shown Figure 4-24, which renders the stick figure ready for displaying captured motion. The view can be adjust by computer mouse and keyboard for better observation. The tools for instant numerical inspection, which are the three-by-three matrix of data gauges and graphs showing capture motion data, are depicted in Figure 4-25.

Chapter 5 Validation of SmartSuit Sensor as a Sensing Module

To assess the performance of the OLE as well as the sensor firmware, we conducted a variety of experiments. The first set of tests was conducted to evaluate the accuracy of the OLE in ideal condition, with rigid joint and link, actuated by PowerCube of Amtec robotics. The second set compared OLE's performance for arm bending motion with another popular commercial goniometric instrument, BIOPAC TSD130A twin-axis Goniometer. The purpose of these experiments is not only to examine the performance of the OLE, but also to validate that the OLE together with the embedded firmware and the processing software can be use as a goniometry device.

5.1 Repeatability and Accuracy Test of OLE

The objective of this experiment is to verify the accuracy and repeatability of the concept that relates the joint angle and the encoder displacement based on equation (3) derived in Chapter 2. The idea of this experiment is to test the performance of the OLE in the ideal working scenario with rigid links and joints with known joint diameter and center. The recorded data are converted into joint angles, which are compared with the readings from a precise rotary encoder installed inside experimental set-up.

5.1.1 Set-Up of PowerCube Comparison Experiment

The linear encoder assembly is mounted onto a jig that is attached to a PowerCube, a high-precision servomotor, shown in Figure 5-1. The jig composes of two rigid links and one joint with a diameter of 60 mm. The sensing module is on the jig similarly to that on the human body. Sensor assembly is fixed on one link, while the pulling cable is fix on the other link. Rotation of the jig causes the motion of the encoder head onto the code-strip. The PowerCube rotary module has a 2000 pulse per revolution encoder, which translates into 0.18° per pulse. The sensor displacement measurement was recorded from the linear encoder via the DSC while the PowerCube rotary module rotates from 0° to 90° at an interval of 10° .

Figure 5-1 Set-Up of PowerCube Comparison Experiment

Figure 5-2 Graph showing PowerCube Reading against Sensor Reading

5.1.2 Results from rigid link experiment

The readout from the linear encoder, in unit of number of lines is plotted against the PowerCube rotation which is in unit of degree. Figure 5-2 illustrates a linear relationship with a correlation coefficient of 0.9996 between the measurements from the two systems which attests the accuracy and repeatability of the OLE sensor. This experiment also confirms our initial assumption on the working principle of our sensors as denoted by

equations (1), (2) and (3). Furthermore, the constant radius (i.e. 31.4 mm), which is computed from the PowerCube rotation angle and the OLE travel distance, is very close to the actual radius of the joint of the jig (i.e. 31.5 mm). The equation to calculate the radius, derived from equation (3), is as follows

$$R_J = \frac{D_E}{\alpha_{PC}} \cdot \frac{360^\circ}{2\pi} \quad (25)$$

Here, R_J is the radius of the jig, D_E is the OLE recorded distance, and α_{PC} is the PowerCube rotation angle.

5.2 In-Situ test of OLE sensors

Different from the previous experiment carried out in an ideal condition, in-situ tests of wearable sensors are conducted to examine the performance of OLE in recording real human motion. A popular motion capture system, BOPAC's TSD130A twin-axis Goniometer is used in conjunction with OLE for comparison. Although optical motion capture systems, such as Vicon, are still considered the gold standard, such systems are relevant to entire body motion capture. Besides, electro-goniometers are better instrument for measuring individual human joint motion. In the in-situ experiment of OLE sensors, the two body motion sensing systems are fixed to the right arm of the subjects simultaneously as shown in Figure 5-3. When the subject flexes the arm, both two sensing systems will be able to record the arm movement at the same time. The performance of OLE wearable sensor is evaluated on the single-DOF elbow joint in order to establish the basis for other joint with more DOF.

5.2.1 Set-up of Comparison Experiment

The goniometer system measures the change in the position of a joint via resistance change in a set of strain gauges. The angular readout of the goniometer serves as a reference for the elbow bend angle. The goniometer is attached to the lateral side of the upper arm while we strap on our sensor to the posterior of the elbow joint where the ulna bone is located. The goniometer is attached to the arm using double-sided tapes while our sensor is incorporated onto a bare-bone substrate design for the comfort of the user to put on and remove. The OLE wearable sensor is strapped to the posterior of the elbow joint, where the ulna bone is located.

Figure 5-3 Set-Up of Comparison Experiment for BIOPAC Goniometer and OLE

Three healthy male subjects were recruited for the experiments. Table 5.1 shows the subjects' profile, with their age, height, upper arm length, and forearm length. Each of the subjects was required to perform two sets of experiments to test the accuracy of the sensor. For the first set of experiments, each subject is seated and asked to do a simple elbow flexion motion from 0° (when the arm is vertical, and pointing downward along the body) to 90° (when the upper arm is vertical, and the forearm is pointing forward from the body) and back to 0° for a few cycles at a slow pace. For the second set of experiments, the subjects were asked to flex the forearm as fast as possible while attempting to keep the upper arm still throughout the experiment. Figure 5-3 illustrates the experimental set-up for these comparison tests of the two motion capture systems.

TABLE 5.1
SUBJECTS' PROFILE

	subject 1	subject 2	subject 3
Age	26	26	30
Height	1.71	1.78	1.67
upper arm length	29.3	31.4	26.2
forearm length	26.5	28.7	25.1

Prior to the experiment, the goniometer and OLE require calibration in order to determine the radius of the elbow joint. For the calibration test, a set of readings from 5° to 80° , with a 5°

interval were taken with three repetitions. Using the results from the goniometer as a reference of the elbow bend angle, the output from the linear sensor, in units of number of lines were recorded and are plotted against that of the goniometer reading as shown in Figure 5-4. Using least square fitting on the collected data points, a correlation coefficient of 0.9834 between the data and the fitting line is computed. The slope of the straight line, G_r , can be used to estimate the elbow radius which will be used for subsequent calculation, as follows

$$R = G_r \frac{360^\circ}{2\pi} \quad (26)$$

The formula for calculating the correlation coefficient of two sets of data is:

$$\rho_{xy} = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{\sqrt{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2} \sqrt{n \sum_{i=1}^n y_i^2 - \left(\sum_{i=1}^n y_i \right)^2}} \quad (27)$$

Here, $\{x_i\}$ and $\{y_i\}$ are the two data sets, n is the number of data point in each set, ρ_{xy} is the correlation coefficient.

Figure 5-4 Graph showing Goniometer Reading against Sensor Reading

Figure 5-5 Flowchart of Experimental Procedure

5.2.2 Results of Comparison Experiment

After the calibration with the calibrated radius of the joint under consideration, the subjects are ready for the experiment (procedure shown in Figure 5.5). For the first set of experiment, the subjects were required to slowly bend their forearm from a straighten initial position at 0° (when the arm is vertical, and pointing downward along the body) to 90° (when the upper arm is vertical, and the forearm is pointing forward from the body) and then back to 0° for a few cycles (at the rate of about 0.6 Hz).

Figure 5-6 illustrates that the readings of our sensor as compared to that of the goniometer system for slow flexion of the elbow. From the figure, we can deduce a close relation of the

performance between the goniometer and the sensor. A statistical analysis of the performance and error between the two systems were carried out. Figure 5-6 shows that the RMS error of the elbow joint angle is 3.8044° and the correlation coefficient is 0.9955.

Figure 5-6 Slow Flexion Motion

For the second set of experiment, the subjects were required to flex their forearm to 45° and oscillate around that range and continue up to 90° and holding it there before straightening back to 0° for a few cycles. Figure 5-7 illustrates that the outcomes of our sensor as compare to that of the goniometer system. Table 5.2 shows that the RMS error of the elbow is 3.4786° , and the correlation coefficient is 0.9901. In addition, based on the graph, we can observe that the sensor performance is able to handle high frequency excitation (at the rate of about 2 Hz). Figure 5-7 also shows that there are the gaps between the OLE data and the goniometer data at the beginning and the end of the experiment. This can be explained by the buckling of the stainless steel cable, which pulls and pushes the encoder assemble to move

upon the code-strip. At the beginning of the experiment, the joint suddenly starts moving with high rate, causing a large acceleration, which exerts force on the stainless steel cable. The force results in the buckling of the cable. Similar phenomenon also happens at the end of the data curves.

TABLE 5.2
SUMMARY OF THE COMPARISON EXPERIMENT

Experiment	Root Mean Square (RMS)	Correlation Coefficient
Flexion Motion	3.8044	0.9955
Oscillation Motion	3.4786	0.9901

Figure 5-7 Fast flexion motion.

5.3 Summary

This chapter presents two sets of experiments to assess the performance of the OLE. The experiments consist of the tests on a rigid link-joint jig to assess the performance of the OLE in the ideal condition, and the in-situ tests on human subjects compare the performance of the OLE with BIOPAC goniometer. The results of the tests on a rigid link-joint jig show a correlation coefficient of 0.9996 between the measurements from the OLE and the PowerCube rotary encoder. This confirms the accuracy and repeatability of the OLE in the ideal working condition (rigid link and joint). In the in-situ experiments, the performance of OLE is assessed in comparison with the BIOPAC goniometer in two conditions: normal and fast joint movement. In the tests when the elbow joint is flexed at the normal rate of 0.6Hz, the RMS error of the joint angle is 3.8044° between measurements from OLE and those from the goniometer, and the correlation coefficient between them is 0.9955. In the tests when the elbow joint is flexed at the fast rate of 0.6Hz, the RMS error of the joint angle is 3.4786° between measurements from OLE and those from the goniometer, larger than the former tests in ideal condition, and the correlation coefficient between the two is 0.9955, smaller than the former tests in ideal condition. The in-situ experiment results with fast joint movements also show the gaps between the OLE data and the goniometer data at the beginning and the end of the data curves. This is caused by the buckling of the stainless steel cable, which pulls and pushes the encoder assemble to move upon the code-strip.

Chapter 6 Validation of SmartSuit as a Motion Capture System

In addition to validation of the OLE sensor as a new way of measuring body joint movements, further tests are carried out to use multiple SmartSuit sensing modules as a body motion capture system. In this chapter, the experiments on multiple OLE sensors as the SmartSuit are described. The repeatability and reliability of SmartSuit wearable sensor network were tested for in-situ experiments. The configuration of SmartSuit for this experiment is a single arm version with the wrist, elbow and shoulder modules as seen in Figure 6.1. The experimental procedure is adapted from a therapy session of stroke rehabilitation, which is one of the targeted applications of the SmartSuit system. The objectives of the repeatability and reliability experiments are to assess the SmartSuite as a motion capture system, and to validate the implementation of the CAN bus network, the SmartSuite embedded firmware, and the PC software.

Figure 6-1 Experiment set-up for repeatability and reliability tests for arm motion capture

6.1 Experimental Set-up

Five healthy individuals, male, aged 24-30 years, with no movement disorders in the arm and hand participated in the repeatability and reliability studies. They came from different disciplines of Nanyang Technological University. Table 6.1 shows the subjects' profile.

TABLE 6.1
SUBJECTS' PROFILE

	subject 1	subject 2	subject 3	subject 4	subject 5
Age	26	26	30	24	25
Height	1.71	1.78	1.67	1.82	1.74
upper arm length	29.3	31.4	26.2	32.2	30.3
forearm length	26.5	28.7	25.1	29.4	28.1

Each user is instructed to wear the SmartSuit on his arm, with the desired sensor placement, and using the sensor package. Wireless transmission, via a RF station using Zigbee protocol, from the SmartSuit hardware to the host computer is utilized. A Lithium battery pack with capacity of 2000mAh – 3.7V is used for power supply for the SmartSuit sensor network. The graphic display mode and data storing mode are both activated for the experiments on the SmartSuit PC software. The user will be able to observe the virtual human performing captured motion from SmartSuit in real-time, while the data are recorded into the host PC.

6.2 Validation Procedures

The repeatability of SmartSuit was tested by an arm-reaching task. To perform this task, subjects wearing the arm suit drove a virtual arm to reach a virtual ball above the virtual arm in the simulation environment. The virtual ball was controlled to move along a predefined route to guide the user. The subjects were required to move the arm to drive the virtual arm to follow the virtual ball. The details about changing the ball position and moving both the real and virtual arms are illustrated in Figure 6-2 and Figure 6-3. The virtual environment simulates an upper arm rehabilitation session. The procedure of the arm movement is:

- a. The entire arm first lifts up and when the shoulder pitches to the maximal angle,
- b. The elbow starts bending to the maximal angle and then returns; when the forearm and upper arm lie in one line,

- c. The wrist starts bending to the maximal angle and then returns;
- d. The wrist starts rolling to touch ball and then back; after that, the whole arm returns to the initial position.

In this routine, the shoulder, elbow, and wrist joints need to rotate from the initial rest state to the maximum tension state and then return. Each subject performed the arm-reaching task for 11 trials. The first trial was for practicing the task, and not included in the final result.

Figure 6-2 Arm movement testing procedure:

(1) Initial gesture); (2) Lift arm as vertical as possible; (3) Bend elbow backwards; (4) return to the gesture in (2);

Figure 6-3 Arm movement testing procedure (continued):

- (5) Bend wrist backwards, then return to the gesture in (2); (6) move the arm to horizontal gesture;
(7) Roll the wrist then move back; (8) Return to the gesture in (1)

6.3 Analysis

This section presents the statistical analysis method for the repeatability and reliability test data obtained from SmartSuit.

6.3.1 Repeatability testing

Repeatability of the sensors is assessed by the range and standard deviation of the data sets. Each raw data file (one subject) contained one data block of 10 trials of reaching the ball. For each of the ten trials, four SmartSuit sensors detected the corresponding joint movement and each sensor collected 1000 sample data during the arm-reaching task. Therefore, the arm-reaching task performed by each subject can be represented by a 3D array of data $\{X_{ijk}\}$, $i = 1, \dots, 10$, $j = 1, \dots, 10$, $k = 1, \dots, 4$ to specify the i^{th} position in the j^{th} trial (subject) for the k^{th} sensor. In order to measure repeatability, for each subject four terms are defined:

- Range, which is smallest interval which contains all the data of the interested set;
- Average of the range;
- Standard deviation, which is the measure of the variability of a data set; and
- The average of the standard deviation.

For each sensor, Range and Standard Deviation are highly correlated. Together these values give the approximate measure of the repeatability of each body motion capture sensor [24]. The maximum and minimum trial data averages were used to establish the range of each sensor. Range is computed by the functions in [25]:

$$R_k = \max_j \bar{X}_{jk} - \min_j \bar{X}_{jk}, \quad (28)$$

Where

$$\bar{X}_{jk} = \frac{1}{10} \sum_{i=1}^{10} X_{ijk}. \quad (29)$$

From individual data block ranges (R_k) and corresponding standard deviations of X_{jk} values, an overall value of average range and average standard deviation across all five subjects was computed. Data processing was performed in SPSS (Statistical Package for the Social Sciences). The formula for standard deviation is as follows:

$$SD_k = \sqrt{\frac{1}{N} \sum_{j=1}^N (\bar{X}_{jk} - \mu_k)^2}. \quad (30)$$

In which, μ_k is the mean calculated by the following expression

$$\mu_k = \frac{1}{N} \sum_{j=1}^N \bar{X}_{jk} . \quad (31)$$

6.3.2 Reliability analysis

Reliability analysis was conducted by computing an Intra-class coefficient (ICC), which describes how strongly units in the same group resemble each other, to identify the source of variability between measures. ICC operates on data structured as groups, rather than data structured as paired observations. ICC values close to one indicate high internal consistency in the measurement method, giving confidence that the test results reflect true scores. Reliability analyses were performed by randomly selecting two of the ten data blocks per subject, and further randomly selecting one of the ten trials within each data block, and then ICC was computed for these two trials in SPSS. This procedure was repeated 20 times.

For the reliability analysis, intra-class correlation coefficient is computed. Among the series of measurements performed on different subjects, two components of variability exist, the variability among their average values computed over each repeated measure σ_t^2 and the variability of the random errors σ_e^2 . ICC is a single quantity that describes the relative magnitude of the two components of the variability. The following formula is the estimation of ICC in short form, proposed by [26]

$$ICC = \frac{\sigma_e^2}{\sigma_e^2 + \sigma_t^2} \quad (32)$$

As σ_t^2 / σ_e^2 decreases, the measurement error explains a decreasing percentage of the variance in the data, reliability increases, and ICC approaches its maximum value of one. As σ_t^2 / σ_e^2 increases, the measurement error explains an increasing percentage of the variance in the data, reliability therefore decreases, and ICC approaches its minimum value of zero.

Figure 6-4 Sample data from each joint for a single data block (one subject) collected during the performance of the arm-reaching task. Angles (the vertical dimension) are in degrees. Time (the horizontal dimension) is in seconds. When the arm lies horizontally, the degree of the shoulder joint is 0 degree. When the upper arm, forearm and hand lie in one line, the angles of both elbow and wrist are 0 degrees. Wrist roll is the angle of the hand with respect to the forearm.

6.4 Results

6.4.1 Repeatability

Following experiments with the SmartSuit system, raw data block files were processed. A single data block file contains 10 trials of performing the arm-reaching task is shown in Figure 6-4 (Note that the angle of the shoulder joint was coded to range from -90° to 90°) The fluctuation of data presented on top of the curve in Figure 6-4 is attributable to the shaking of joint at the extremity of the arm gesture. The shaking was caused by the tension in the muscle. Five such data block files were processed to produce five average values for each sensor, as shown in Figure 6-5. From these data, the Range and Standard Deviation for each subject were calculated, as depicted in Figure 6-6. Repeatability of sensors is generally reported as the standard deviation of the Ranges. The average range was 2.819° (average of the range shown in Figure 6-6). In addition, the average standard deviation was 0.697° (average of the SD depicted in Figure 6-6).

Figure 6-5. Individual data block averages for repeatability. Each value is the average of 10 cycles of the arm-reaching task for each sensor per subject.

Figure 6-6 Average range and standard deviation (SD) for each subject.

6.4.2 Reliability

ICC analysis was performed for each sensor. The average ICC for each sensor ranged from 0.959 to 0.975 with an overall average of 0.967 ± 0.08 , as seen in Table 6.2. All the average ICC of the experiments are very close to 1.00, indicating that the proposed motion capture system has high reliability.

TABLE 6.2
INTRA-CLASS CORRELATION COEFFICIENT OF RELIABILITY

Shoulder	Elbow	Wrist (bend)	Wrist (roll)	Average
0.975	0.974	0.959	0.962	0.967

6.5 Discussion

The SmartSuit system was attached on the arm and evaluated for repeatability and reliability. We designed one protocol to test the sensor property by asking subjects to perform an arm-reaching task. Different types of joints' activities can be discriminated from the data. The experimental scheme emulates a stroke rehabilitation therapy section, when doctors want to assess the patient's joint movement for monitoring improvement progress after the stroke. Figure 6-4 shows the sample data set in the same time scale for one subject performing a complete experimental procedure depicted in Figure 6-2 and Figure 6-3. It is observed that the data tend to fluctuate on top and bottom of the curves. On top of the curves, the joints are at the extreme position, and the muscles are in high tension. High muscle tension will cause the instability of the joint. In addition on top and bottom of each joint's data curve, other joints are moving. These motions of other joint also affect the stability of the joint when dwelling, as shown in Figure 6-4.

After data analysis, the average range was calculated to be 2.819° . In addition, the average standard deviation was 0.697° . Although there is no previous work which tests the range and standard deviation of other motion capture systems to compare with SmartSuit, the numerical results show good repeatability of SmartSuit system. In addition, reliability analysis showed high ICC values for all channels within 0.959-0.975 with an overall average of 0.967, showing the ability of our system to perform and maintain its functions in routine circumstances, with different biometric subjects. The results also indicate that the implemented network topology using CAN bus and the embedded firmware are effective. The SmartSuit software is easy for the user for interacting with the SmartSuit hardware.

Chapter 7 Conclusion

This thesis concerns with the embedded programming and PC software design and validation of a new wearable sensing system, SmartSuit. A new sensing technology is proposed: OLE, which measures the linear distance using a transmitter-receiver pair sliding upon an engraved code strip, is utilized with a mathematical model to convert the linear distance travelled by the encoder into joint angle. This simple concept enables us to capture joint motion after a calibration procedure.

Our system is the first tracking system which uses OLEs and the combination of OLEs and accelerometers to perform the task of motion capture. The simplicity in technology allows us to minimize the production cost (cost of OLE + accelerometer + microcontroller + circuit and mechanical design is approximately (USD) $\$10 + \$15 + \$5 + 10 = \40 , respectively). Besides the low cost, our system has certain advantages over other motion tracking technologies used in rehabilitation. Compared with IMU, OLE does not suffer from data drift due to the integration of angular velocity (gyroscope) and ADC resolution errors, platform tilt errors, geomagnetic variation, and nearby ferrous distortions (magnetic sensor). Quadrature digital signals from the OLE and acceleration data from the accelerometer do not require excessive sensor fusion algorithms, such as Extended Kalman Filter normally used in IMU. In addition, the OLE directly captures the joint angle. Therefore, only one OLE is needed for individual joint assessment, while two IMUs are requisite for the same purpose. OLEs do not suffer from the difficulties in center-of-rotation alignment of rotary encoder (widely used in exoskeleton). OLEs are basically aligned along the plane where skin is stretched the most when the joint is moving. The detailed guide for OLE placement is discussed in section IV. Moreover, compared with exoskeleton, OLE is simple in design, easy to wear and has compact package. Our sensor network works without a restrictive work space. The current wireless protocol allows users to move freely in a region of ten-meter radius, which can be much extended with Bluetooth.

Admittedly, the best reconstructed human motion data are still not perfect. The OLE-based module is not able to capture the horizontal movement of the upper arm. The reason is that each OLE can capture only one DOF while the accelerometer cannot track its yaw. The problem can be completely resolved by replacing the shoulder

module with an IMU. However, this change will dramatically raise the cost of the system. Except for the singularity, the OLE-based module can accurately capture individual joint angle, and all the arm movements.

The data quality along with the sensing module's low cost, compact size, improved versatility and wearability suggest that our system may be used for numerous applications in home-based rehabilitation. The ergonomic design of the OLE-based system allows patients to don it themselves and use the system for their personal rehabilitation session with comfort. The ease of system setting, calibration and use would minimize technical issues that users may face in unsupervised home environment. In addition, the OLE-based system can be used with virtual-reality software to provide visual feedback, avoid boredom, and help users personalize the rehabilitation therapy and self-assess the progress of recovery.

The following works are presented in the thesis.

1) Body sensor network for upper extremity motion capture

A design for a sensing module is implemented, combining an OLE and an accelerometer into a single unit of compact size so that it can be comfortably worn on the human body. The sensing module is capable of measuring the joint angle by the OLE, and also tracking the roll and pitch orientations of the limb segment. The architecture of a body sensor network connecting the SmartSuit sensing modules is then designed to capture the multi-DOF motion of the human arm. The final design uses a bus topology for SmartSuit. The sensor network, including three sensing modules and one Data Concentrator, is connected through CAN bus. The Data Concentrator acquires captured motion data from the SmartSuit sensor nodes, and sends to PC wirelessly via ZigBee protocol.

Experiments were conducted to assess the performance of the OLE for the task of goniometry. The results show that OLEs have excellent accuracy and repeatability when tested on a rigid link-joint jig. This experiment also set the benchmark for in-situ experiment. In-situ experiments on different human subjects of OLE along with BIOPAC goniometer were conducted. For the experiments when the subject flex their elbow at a normal rate of about 0.6Hz, RMS error (between the OLE and the

goniometer) of the elbow joint angle is 3.8044° , and the correlation coefficient (between the OLE and the goniometer) is 0.9955. For the experiments when the subject flex their elbow as fast as possible, at a rate of about 2Hz, RMS error (between the OLE and the goniometer) of the elbow joint angle is 3.4786° , and the correlation coefficient (between the OLE and the goniometer) is 0.9901.

An experimental procedure, emulating a rehabilitation therapy section, is implemented. The experimental results show that the SmartSuit system has good repeatability with average range of 2.819° and average standard deviation of 0.697° . In addition the SmartSuit network performed well in reliability test with high Intra-Class Correlation coefficient of 0.967 (ranging from 0.959 to 0.975). These results illustrate the ability of our system to perform and maintain its functions in routine circumstances, with different biometric subjects.

2) Kinematics model

Accelerometers can measure only roll (about X-axis) and pitch (about X-axis); they are unable capture the entire motion of articulate bodies. The combination of an OLE and an accelerometer in to a single unit opens the possibility of compensating for the accelerometer singularity. A kinematic model of human arm with seven DOF is constructed following Denavit-Hartenberg representation to realize this compensation. The model maps the sensor data into human arm motions for kinematic analysis. The properness of model confirmed by experimental results from the repeatability and reliability tests.

3) SmartSuit motion processing software

The SmartSuit software architecture is proposed as a programming framework of the project. The software is an essential part of the SmartSuit motion capture system. It helps user to interface with the sensor hardware, obtain the desired motion data, process the data, and regenerate the motions for display or storage. The main features of the SmartSuit software are summarized as follows.

a) Motion data processing

This part investigates the way that the software interfaces with the hardware system. The communication gateway between the software and the hardware uses three main methods, including RS232, USB and wireless transmission via Zigbee protocol. The protocol and algorithm for handling the COM ports are implemented so that the software can interface with the SmartSuit hardware no matter what COM port is being used. The transmission baud rate is set to 115200 bits/s, which provide a sampling rate of 240 Hz. This sampling rate allows the system to operate in real-time.

The raw data acquired from the sensor units are then converted into more meaningful information, e.g. joint angles. The information is stored in a hierarchical structure, which models the human body. The hierarchical model consists of different primitive objects, which are special data structures bearing motion data.

b) Rendering the graphics

In the software, the graphics display unit demonstrates the captured motion data from the sensors. It is also where the virtual environment is placed to enable human interact with the cyber world. The virtual world can be rendered and manipulated using OpenGL. Human model is draw in a hierarchical procedure, and controlled by the data structure given by the Motion Data Processing Unit. OpenGL can only be used to render simple geometries. For complex geometries, an alternative is to design them in modeling softwares with 3ds file format, then import the resulting object into the software. A special set of functions interprets the 3ds file format and optimally reproduce the complex geometries. The graphics display unit also provides the visual feedback for user to observe their motion. In the validation experiment, the subjects used the graphics display as a guide for the movement procedure, and use the visual feedback to better control the virtual human in the software to perform the required tasks (reaching the virtual ball).

7.1 Contributions of the thesis

The thesis has the following contributions towards SmartSuit and a new body motion capture technology.

(1) *Design of the network architecture for SmartSuit:*

The sensor network architecture was designed to connect multiple SmartSuit modules and one Data Concentrator for capturing human arm motion. CAN bus was selected from the review of various network topology. The SmartSuit network was created so that the data are transformed from the sensor nodes to the Data Concentrator, and then to the host computer in the fastest and most direct way. The thesis presents an inexpensive implementation for tracking human motion that is more wearable than existing systems which require more infrastructures. The experimental results indicate that the SmartSuit system has comparable accuracy with commercial motion capture systems, and has good repeatability and reliability, showing the ability of our system to perform and maintain its functions in routine circumstances, with different biometric subjects.

(2) *Firmware architecture and programming:*

The firmware architecture for the SmartSuit sensor network is proposed, implemented and embedded into the microcontroller of the three SmartSuit modules and the Data Concentrator. The firmware allows the communication among the sensor nodes and Data Concentrator, transmission of the data from the sensors to the Data Concentrator through CAN bus and then to the host computer via RS232, USB or wireless transmission using Zigbee protocol. The firmware also avoid collision of data flows and synchronize the queue of motion data from the nodes to the CAN bus. The firmware maintain the transmission rate of 1Mbit/s through CAN bus, and the transmission of 115200 bits/s from the Data Concentrator to the host PC. The experiments validate that the firmware allows the data transmission within the sensor

network and from the Data Concentrator to the PC to be performed in real-time.

- (3) ***Kinematic model:*** Accelerometers are unable to track their heading orientation. Therefore the use of only accelerometers is insufficient to capture the motion of a human arm. With three OLE sensors to measure one DOF at the shoulder, one DOF at the elbow, and one DOF at the wrist, sensor network combining OLEs and the accelerometers are able to capture the motion of the human arm. The kinematic model of a 7-DOF human arm is proposed to convert sensor data from OLEs and accelerometers into human joint angles. Indeed the use of Denavit-Hartenberg representation to model an arm is not new. However, the use of the model to compensate the singularity of accelerometers using information provided by the OLEs is novel. The repeatability and reliability tests validate the joint angle calculation from the sensor data using the kinematic model.

7.2 Future works

Although much work has been done in this project, there are still rooms for improvements and unexplored aspects for this motion capture system.

7.2.1 Extension of the body sensor network to entire human body

The current body sensor network is capable of capturing the motion of a human arm. There is a demand for extending the sensor network to entire body for those applications that require full body motion tracking. However, there are several problems need to be solved for the extension.

- 1) The kinematic models for lower parts of human body need to be constructed. Kinematic model for human spine, which is a high-DOF mechanism, may be a challenging issue. In addition, modeling hip joints is as difficult as modeling shoulder joints.
- 2) When the number of sensors in the network increases, the load of data to be transmitted increases. Proper embedded communication protocol for large number of sensors is required to be carefully designed.
- 3) Making the sensor network wireless is also a challenging direction for further work. Because the current network is connected via CAN bus, this wireless configuration requires redesign of the board, and addition of wireless transmitter and controller. Wireless protocol also needs to be design to transmit the heavy load of data and to avoid traffic jam due to the large number of data channels.

7.2.2 Applications development

The SmartSuit system could be used to develop a physical training system used in Tai-chi or yoga lessons. Motions of masters are recorded using the motion capture system. Students also wearing a motion tracking system perform the routines. Motion data of students are compared with those of master. A feedback system would instruct students to follow the masters' motions. This would allow the training to be carried out remotely, in offline manner and through internet. In entertainment, the OLE-based motion capture system can be converted to a gaming device. Low cost feature of our system will make it promising in this low-end range of applications.

On the other hand, we are planning to build a motion data base with various motion data sets to further understand human motion. Restrictive recording constraints of currently commercial system limit the scope of current motion data sets, which prevents the broader application of motion processing. An inexpensive and wearable motion tracking system would enable the collection of large data sets. This motion data infrastructure could then support large-scale analysis of human motion, including its style, efficiency, and adaptability. The analysis of daily human motion would even progress beyond computer animation, help avert stress injuries, assist rehabilitative therapy, and enable design of improved computer interfaces.

References

- [1] C. D. Murray, E. Patchick, S. Pettifer, F. Caillette, T. Howard, “Immersive Virtual Reality as a Rehabilitative Technology for Phantom Limb Experience: A Protocol”, *CyberPsychology & Behavior*, Vol. 9-2, pp. 167-170, 2006.
- [2] M. Brand and A. Hertzmann, “Style machines”, *Proceedings of ACM Special Interest Group on Computer Graphics and Interactive Techniques*, pp. 183–192, New Orleans, Louisiana, United State, 2000.
- [3] H. Dee and D. Hogg, “Detecting inexplicable behavior”, *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 477-486, Kingston, UK, 2004.
- [4] M. Cohen, “Design principles for intelligent environments”, *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI’98)*, pp. 547 - 554 Madison, Wisconsin, United State, 1998.
- [5] M. Cohen, “The future of human-computer interaction or how I learned to stop worrying and love my intelligent room”, *IEEE Intelligent Systems*, Vol. 14-2, pp. 8-10, March 1999.
- [6] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer, “Easyliving: Technologies for intelligent environments”, *Second International Symposium on Handheld and Ubiquitous Computing (HUC 2000)*, pp. 12–27, Bristol, United Kingdom, September 2000.
- [7] Animazoo UK Ltd, 2009. [Online]. Available: <http://www.animazoo.com>.
- [8] SensAble Technologies, Inc, 2009. [Online]. Available: <http://www.sensable.com/>.

-
- [9] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, D. Colucci, "The HiBall tracker: High-performance wide-area tracking for virtual and augmented environments", in *Proc. ACM Symposium Virtual Reality Software Technoogy.*, London, United Kingdom, pp. 1–11, Dec. 1999.
- [10] 3rdTech, Inc., 2009. [Online]. Available: <http://www.3rdtech.com>.
- [11] T. G. Bishop, "Self-Tracker: A Smart Optical Sensor on Silicon", PhD thesis, University of North Carolina at Chapel Hill, United State, 1984.
- [12] Vicon, 2009. [Online]. Available: www.vicon.com.
- [13] N. Miller, O. C. Jenkins, M. Kallmann, M. J. Andmatric'c, "Motion capture from inertial sensing for untethered humanoid teleoperation", in *Proc. International Conference of Humanoid Robotics*, pp. 547–565, Los Angeles, California, United State, 2004.
- [14] N. M. Vallidis, "WHISPER: a spread spectrum approach to occlusion in acoustic tracking", PhD thesis, University North Carolina at Chapel Hill, United State, 2002.
- [15] L. Danisch, K. Englehart, and A. Trivett, "Spatially continuous six-degrees-of-freedom position and orientation sensor," *Sensor Review*, Vol. 19-2, pp. 106-112, 1999.
- [16] Measurand Inc., 2009. [Online]. Available: <http://www.measurand.com>.
- [17] G. Welch, E. Foxlin, "Motion Tracking: No Silver Bullet, but a Respectable Arsenal," *IEEE Comp. Graphics and Applications*, Vol. 22-6, pp. 24-38, 2002.

-
- [18] K. Y. Lim, Y. K. Goh, W. Dong, K. D. Nguyen, I.-M. Chen, S. H. Yeo, H. B. L. Duh, and C. G. Kim, "A Wearable, Self-Calibrating, Wireless Sensor Network for Body Motion Processing", *IEEE Int'l Conf. on Robotics and Automation*, pp. 1017 - 1022 , Pasadena, California, U.S, May 2008.
- [19] K. D. Nguyen, I.-M. Chen, Z. Luo, S. H. Yeo, H. B. L. Duh, "A Body Sensor Network for Arm Motion Tracking and Monitoring", *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, Missouri, United State, October 2009
- [20] R.L. Sharma, *Network Topology Optimization: The Art and Science of Network Design*, New York: Van Nostrand Reinhold, 1990.
- [21] W. Dong, K. Y. Lim, Y. K. Goh, K. D. Nguyen, I.-M. Chen, S. H. Yeo, and H. B. L. Duh, "A Low-cost Motion Tracker and Its Error Analysis", *IEEE International Conference on Robotics and Automation*, pp. 311 – 316, Pasadena, California, United State, May 2008.
- [22] K. H Low, *Robotics: Principles and System Modeling*, Prentice Hall, Singapore, 2002.
- [23] K. Luttgens, N. Hamilton, *Kinesiology: Scientific Basis of Human Motion*, 9th Ed., Madison, WI: Brown & Benchmark, 1997.
- [24] S. Wise, et al, "Evaluation of a fiber optic glove for semi-automated goniometric measurements", *Journal of Rehabilitation Research and Development*, Vol. 27-4, pp. 411–24, 1990.
- [25] L. Dipietro, A. M. Sabatini, P. Dario, "Evaluation of an instrumented glove for hand-movement acquisition", *J. Rehab. Res.*, 40(2), pp.172-90, 2003.
- [26] A. Donner, J. J. Koval, "The Estimation of Intraclass Correlation in the Analysis of Family Data", *Biometrics*, Vol. 36-1, pp. 19–25, 1980.

List of Publications

- [1] K. D. Nguyen, I.-M. Chen, Z. Luo, S. H. Yeo, H. B. L. Duh, “A Wearable Sensing System for Tracking and Monitoring of Functional Arm Movement”, *IEEE/ASME Transaction on Mechatronics*, accepted for publication.
- [2] K. D. Nguyen, I.-M. Chen, Z. Luo, S. H. Yeo, H. B. L. Duh, “A Body Sensor Network for Arm Motion Tracking and Monitoring”, *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, St. Louis, Missouri, United State, October 2009.
- [3] Z. Luo, I. M. Chen, K. D. Nguyen, C. K. Lim, et al, "A Wearable Sensor Network for the Control of Virtual Characters", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2009)*, Singapore, July 2009.
- [4] K. Y. Lim, Y. K. Goh, W. Dong, K. D. Nguyen, I.-M. Chen, S. H. Yeo, H. B. L. Duh, “A Preliminary Study on the Accuracy of Wireless Sensor Fusion for Biomotion Capture”, *The 5th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2008)*, pp. 99-102, Hong Kong, June 2008
- [5] K. D. Nguyen, I.-M. Chen, et al, “Toward a Dynamic Model of Robotic Marionettes”, *IEEE International Conference on Robotics, Automation & Mechatronics (RAM 2008)*, pp. 488-493, Chengdu, China, June 2008.
- [6] K. D. Nguyen, T. C. Ng, I.-M. Chen, “On Algorithms for Planning S-curve Motion Profiles”, *International Journal of Advanced Robotic Systems*, Vol. 5, No. 1, pp. 99-106, March 2008.

-
- [7] W. Dong, K. Y. Lim, Y. K. Goh, K. D. Nguyen, I.-M. Chen, S. H. Yeo, and H. B. L. Duh, "A Low-cost Motion Tracker and Its Error Analysis", *IEEE International Conference on Robotics and Automation (ICRA 2008)*, pp. 311 – 316, Pasadena, California, United State, May 2008.
- [8] K. Y. Lim, Y. K. Goh, W. Dong, K. D. Nguyen, I.-M. Chen, S. H. Yeo, H. B. L. Duh, and C. G. Kim, "A Wearable, Self-Calibrating, Wireless Sensor Network for Body Motion Processing", *IEEE International Conference on Robotics and Automation (ICRA 2008)*, pp. 1017 - 1022 , Pasadena, California, United State, May 2008.
- [9] K. D. Nguyen, I.-M. Chen, S H Yeo, B L Duh, "Motion Control of a Robotic Puppet through a Hybrid Motion Capture Device", *IEEE Conference on Automation Science and Engineering (CASE 2007)*, pp. 753 – 758, Arizona, United State, September 2007.
- [10] K. D. Nguyen, I-. M. Chen, T. C. Ng, "Planning Algorithms for S-curve Trajectories", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2007)*, pp. 1-6, ETH Zürich, Switzerland, September 2007.

Appendix A Serial communication interfaces

In this project, serial communication interfaces mainly used for data transmission are RS232, USB and EHSII WBAN (wireless communication via Zigbee protocol). This section will discuss the interfaces in details.

A.1 RS232

RS-232 is designed to handle communications between two devices with a distance limit of around 15m to 40m, depending on the bit rate and cable type. RS-232 uses unbalanced, or single-ended, lines. Each signal in the interface has one dedicated line whose voltage is referenced to a common ground.

Figure A.1 Actual schematic of a serial port

These are the three essential lines for 2-way RS-232 communications:

- TX. Carries data from the DTE to the DCE. Sometimes called TD or TXD.
- RX. Carries data from the DCE to the DTE. Sometimes called RD or RXD.
- SG. Signal ground. Sometimes called GND or SGND.

The remaining lines are flow-control and other status and control signals. The RS-232 standard defines uses for all of the signals, but applications are free to use the signals in any way as long as both ends agree on what the signals mean. The computer uses an output bit to let the other computer know when it's OK to send data. The DCE asserts CTS when ready to receive data,

and the DTE asserts RTS when ready to receive data. Before transmitting, the Data Concentrator reads the computer's flow-control output. If the signal's state indicates that the computer isn't ready for data, the microprocessor waits. All RS-232 inputs and outputs must be able to withstand a short circuit to any other RS-232 signal, including ground, without damage. The nine lines and their descriptions are listed in Table A.1. The schematic of a serial port is shown in Figure A.1

TABLE A.1
THE NINE LINES

Pin Number	Signal	Source	Type	Description
1	CD	DCE	Control	Carrier detect
2	RX	DCE	Data	Receive data
3	TX	DTE	Data	Transmit data
4	DTR	DTE	Control	Data terminal ready
5	SG	-	-	Signal ground
6	DSR	DCE	Control	Data set ready
7	RTS	DTE	Control	Request to send
8	CTS	DCE	Control	Clear to send
9	RI	DCE	Control	Ring indicator

A. 2 Universal Serial Bus

Universal serial bus (USB) is a standard interface with defined protocols enabling a computer, which contains host-controller hardware and software, to communicate with the motion capture system, SmartSuit, which contains device-controller hardware and firmware. The host computer assigns a driver to the attached device. The driver manages communications between the SmartSuit software and the USB host controller, and defines how the software can access to get the data from SmartSuit. The application software can use the COM-port functions to open serial ports for acquiring data via virtual COM ports with the driver support.

A USB board was designed to convert SmartSuit into a peripheral device, enabling data transmission via USB. When attached to the host computer, the USB board exchanges

information with the computer through enumeration process. The computer request a series of data structures called descriptors from the USB board, which configures the peripheral device (SmartSuit) and its intended use. During enumeration, a bus address is assigned to the USB board, and the current SmartSuit can draw from the bus is also specified. After enumeration, SmartSuit start working as the device it was designed to be, responds to device specific requests, and transmits data.

Depending on the targeted devices, USB connectors are classified into two main categories with different shapes: type A connectors on host devices that supply power and type B connectors on target devices that receive power. Each type in turn has two sub-classes: standard and mini. Table A.2 and Table A.3 illustrate the color codes and functions of the types, which are standard USB and mini-USB respectively. Table A.4 depicts the actual plugs and receptacles of the USB types. In this thesis, the utilized USB cable has standard type A at the host computer end, and mini-USB type B at the SmartSuit end to minimize the size of the USB board.

TABLE A.2
STANDARD USB TYPE A+B PINS

Pin	Wire color code	Function
1	Red	Source (5V)
2	White	Data
3	Green	Data
4	Black	Ground

TABLE A.3
MINI-USB TYPE A+B PINS

Pin	Wire color code	Function
1	Red	Source (5V)
2	White	Data
3	Green	Data
4	Joined to pin 5	Nil
5	Black	Ground

TABLE A.4
THE NINE LINES

Categories	Plus	Receptacles
Type A
Type B
Mini-USB Series A
Mini-USB Series B

A.3 EHSII WBAN

The middleware defining the transmission protocol resides in the RF station plugged to the host PC via a USB hub shown in Figure A.2, and communicates with the the RF board plugged to the Data Concentrator. The Middleware therefore provides an abstraction between the application layers (the PC) and the SmartSuit sensor network. The middleware is provided in the form of C# library(.dll), and the application may perform library functions calls directly to invoke the service(s) which are required. Figure A.3 illustrates the conceptual design between the middleware and the SmartSuit sensor network. The middleware supports two-way communication/interaction from the PC to the SmartSuit Data Concentrator. The programs that constitute the WBAN Middleware reside in both the RF board at Data Concentrator (lower Middleware) and the RF station at the host PC (upper Middleware).

Figure A.2 EHS-II wireless node

Figure A.3 EHS-II wireless node architecture

Appendix B 3ds File Format Data Structure

The data structures, used to store the information obtained from the *.3ds file, are as follows.

*Node specifies the type, location in the linked-list of the interested node, and the node's data.

```
struct Node {
    UserData user;
    Node *next;
    Node *childs;
    Node *parent;
    NodeTypes type;
    Word node_id;
    char name[64];
    Word flags1;
    Word flags2;
    Word parent_id;
    Matrix matrix;
    NodeData data;
};
```

*NodeData is a part of Node data structure. It includes all data belong to the interested node, such as ambient data, object data, camera, light, and so on, as follows.

```
typedef union NodeData {
    AmbientData ambient;
    ObjectData object;
    CameraData camera;
    TargetData target;
    LightData light;
    SpotData spot;
} NodeData;
```

*ObjectData consists of all data which define the object in the interested node, such as size, position, rotation information, name of the object.


```
typedef struct ObjectData {  
    Vector pivot;  
    char instance[64];  
    Vector bbox_min;  
    Vector bbox_max;  
    Vector pos;  
    Lin3Track pos_track;  
    Quat rot;  
    QuatTrack rot_track;  
    Vector scl;  
    Lin3Track scl_track;  
    Float morph_smooth;  
    char morph[64];  
    MorphTrack morph_track;  
    Bool hide;  
    BoolTrack hide_track;  
} ObjectData;
```

*LightData stores all light information, including position of light source, direction of light, intensity, listed below.

```
typedef struct LightData {  
    Vector pos;  
    Lin3Track pos_track;  
    Rgb col;  
    Lin3Track col_track;  
    Float hotspot;  
    Lin1Track hotspot_track;  
    Float falloff;  
    Lin1Track falloff_track;  
    Float roll;  
    Lin1Track roll_track;  
} LightData;
```

*CameraData keeps all the position, angle, rolling information of the camera used in the scene:

```
typedef struct CameraData {  
    Vector pos;  
    Lin3Track pos_track;  
    Float fov;  
    Lin1Track fov_track;  
    Float roll;  
    Lin1Track roll_track;  
} CameraData;
```

Appendix C List of Functions in the SmartSuit Software Programming

This appendix section lists all the library and functions used in the SmartSuit software programming. The software is programmed in C++, and built with Microsoft Visual Studio 2008. Graphics are rendered by OpenGL Standard Library and OpenGL Utility Toolkit.

```
#define GLUT_DISABLE_ATEXIT_HACK
#include <stdio.h>          /* Standard input/output */
#include <stdlib.h>         /* Standard library */
#include <string.h>
#include <math.h>           /* for cos(), sin(), and sqrt() */
#include <GL/glut.h>        /* OpenGL Utility Toolkit header */
#include <windows.h>
/* Some <math.h> files do not define M_PI... */
#ifndef M_PI
#define M_PI 3.14159265
#endif

DWORD WINAPI CollectDataThread_SmartSuite (LPVOID lpParam)
void draw_stick_figure()
void DrawBox(double dWidth, double dHeight, double dLength, bool bCentre)
void draw_diamond_figure()
void drawDiamond(float length)
void draw_all_graph (void)
void draw_graph (unsigned char gnode)
void output_text(GLfloat x, GLfloat y, char *text)
void insertPosition(char arr[], int& size, char element, int position)
static void drawFloor(void)
static void redraw(void)
static void mouse(int button, int state, int x, int y)
static void motion(int x, int y)
static void idle(void)
static void controlLights(int value)
static void visible(int vis)
```

```
static void key(unsigned char c, int x, int y)
static void special(int k, int x, int y)
static int supportsOneDotOne(void)
void init(void)
int main(int argc, char **argv)
void calibrate(void)
void movement_end_check(void)
void position(void)
DWORD WINAPI PositioningThread_SmartSuite (LPVOID lpParam)

class CComPort
{
public:
    CComPort(int iComPort);
    virtual ~CComPort();
    BOOL Initialize();
    bool Read(char sResult[5][20]); //, unsigned char node
    void Terminate();
    wchar_t m_sComPort[5];
protected:

    BOOL m_bPortReady;
    HANDLE m_hCom;
    DCB m_dcb;
    COMMTIMEOUTS m_CommTimeouts;

};
```

Appendix D List of Functions in the SmartSuit Firmware Programming

This appendix section lists the main libraries and functions in the SmartSuit firmware embedded in microprocessors of the Data Concentrator and the SmartSuit sensing module. The firmware is written in embedded C, built and downloaded to the processors by MPLAB C Compiler.

D.1 Firmware in the Data Concentrator

```
#if defined(__dsPIC33F__)
#include "p33fxxxx.h"
#elif defined(__PIC24H__)
#include "p24hxxxx.h"
#endif
#include <ECAN1Config.h>
#include <ECAN2Config.h>
#include <libpic30.h>
#include <uart.h>

void __attribute__((interrupt, no_auto_psv)) _U1RXInterrupt(void)
int main(void)
void ecan1WriteMessage(void)
void ecan2WriteMessage(void)
void clearIntrFlags(void)
void InitUART1()
void oscConfig(void)
void ul6toa(unsigned int ii,unsigned char *string)
void InitVariables(void)
void __attribute__((interrupt, no_auto_psv)) _C1Interrupt(void)
void __attribute__((interrupt, no_auto_psv)) _C2Interrupt(void)
void __attribute__((interrupt, no_auto_psv)) _DMA0Interrupt(void)
void __attribute__((interrupt, no_auto_psv)) _DMA1Interrupt(void)
void __attribute__((interrupt, no_auto_psv)) _DMA2Interrupt(void)
void __attribute__((interrupt, no_auto_psv)) _DMA3Interrupt(void)
void dma0init(void)
```

```

void dma2init(void)
void ecan1ClkInit(void)
void ecan1Init(void)
void ecan1WriteRxAcptFilter(int n, long identifier, unsigned int exide,
unsigned int bufPnt,unsigned int maskSel)
void ecan1WriteRxAcptMask(int m, long identifier, unsigned int mide, unsigned
int exide)
void ecan1WriteTxMsgBufId(unsigned int buf, long txIdentifier, unsigned int
ide, unsigned int remoteTransmit)
void ecan1WriteTxMsgBufData(unsigned int buf, unsigned int dataLength,
unsigned int data1, unsigned int data2, unsigned int data3, unsigned int
data4)
void ecan1DisableRXFilter(int n)
void dma1init(void)
void dma3init(void)
void ecan2ClkInit(void)
void ecan2Init(void)
void ecan2WriteRxAcptFilter(int n, long identifier, unsigned int exide,
unsigned int bufPnt,unsigned int maskSel)
void ecan2WriteRxAcptMask(int m, long identifier, unsigned int mide, unsigned
int exide)
void ecan2WriteTxMsgBufId(unsigned int buf, long txIdentifier, unsigned int
ide, unsigned int remoteTransmit)
void ecan2WriteTxMsgBufData(unsigned int buf, unsigned int dataLength,
unsigned int data1, unsigned int data2, unsigned int data3, unsigned int
data4)
void ecan2DisableRXFilter(int n)
void __attribute__((__interrupt__)) _OscillatorFail(void);
void __attribute__((__interrupt__)) _AddressError(void);
void __attribute__((__interrupt__)) _StackError(void);
void __attribute__((__interrupt__)) _MathError(void);
void __attribute__((__interrupt__)) _DMACError(void);
void __attribute__((__interrupt__)) _AltOscillatorFail(void);
void __attribute__((__interrupt__)) _AltAddressError(void);
void __attribute__((__interrupt__)) _AltStackError(void);
void __attribute__((__interrupt__)) _AltMathError(void);
void __attribute__((__interrupt__)) _AltDMACError(void);

```

D.2 Firmware in the SmartSuit sensing modules

```

#include <p33fxxxx.h>
#include <libpic30.h>
#include "TYPES.H"
#include "CAN.H"
#include "SPI.H"
#include "REGS2515.H"

void OSCInit(void);           // Set OSC to Internal FRC with PLL at
Fcy = 40 MHz
void ioMap(void);             // Assign RPn pin(s) to input/output functions
for this lab
void unlockIO(void);          // Unlock sequence for RPn configuration
void lockIO(void);            // Lock sequence for RPn configuration
void IOInit(void);            // Initialize switch and LED I/O
void SPIInit(void);           // Initialize UART for 9600/N/8/1/HW
Flow Control
void QEIIInit(void);          // Initialize switch and LED I/O
void ICInit(void);            // Initialize switch and LED I/O
void Init2515(void);
void MCP2515Reset(void);
void Delay_ms(unsigned char num_ms);
void SPIByteWrite(unsigned char addr, unsigned char data);
unsigned char SPIByteRead(unsigned char addr);
void RTS(unsigned char buffer);
void LoadTXBuffers(unsigned char buffers, unsigned char *ID);
void INTx_IO_Init(void);
void __attribute__((__interrupt__)) _INT0Interrupt(void); /*Declare external
interrupt ISRs*/
void __attribute__((__interrupt__)) _INT1Interrupt(void);
void __attribute__((__interrupt__)) _INT2Interrupt(void);
void __attribute__((__interrupt__)) _INT3Interrupt(void);
void __attribute__((__interrupt__)) _INT4Interrupt(void);
void __attribute__((__interrupt__)) _QEIIInterrupt(void)
int main(void)
unsigned char read_register(unsigned char register_name);

```

```
void write_register(unsigned char register_name, unsigned char
register_value);
void WriteSPI(unsigned char DataOut)
unsigned char ReadSPI( void )
void WriteSoftwareSPI(unsigned char DataOut)
unsigned char ReadSoftwareSPI( void )
```