

UNIVERSITY OF STRATHCLYDE

**Enhancing Livestock and Human Health
Monitoring via Analysis of Electronic
Sensor Data**

by

Shikha Sarkar

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering
Department of Electronic and Electrical Engineering

June 2018

Declaration of Authorship

I, Shikha Sarkar, declare that this thesis titled '*Enhancing Livestock and Human Health Monitoring via Analysis of Electronic Sensor Data*' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSITY OF STRATHCLYDE

Abstract

Faculty of Engineering
Department of Electronic and Electrical Engineering

Doctor of Philosophy

by Shikha Sarkar

This thesis presents a body of work involving novel algorithms for enhancing the effectiveness of low-cost sensors in monitoring applications. A holistic approach has been taken in this work in that modelling, simulation, and monitoring tools have been developed from scratch with a number of novel ideas. As its first contribution, the thesis presents a new simulation tool, WSNSIM - a tool for performance analysis of wireless sensor networks (WSN) formed by sensor nodes deployed on farm animals for monitoring of health and oestrus. In this application of wireless sensor networks, the mobility and herding patterns are modelled using statistical tools such as the Gamma density function, mean index of adequacy (MIA), exponential distribution, K-means clustering etc. to give rise to network simulation that is based on accurate herd behaviour. The simulation results are used in evaluation of novel protocol ideas customized to the needs of farm monitoring. The paper [153] presents a new simulation tool for performance analysis of wireless sensor networks (WSN) deployed on farm animals.

The second and third key contributions of the thesis investigate monitoring of human body joints for the purpose of gait and upper limb motion assessment. Unlike the standard approach of marker-based joint monitoring for motion assessment, this work investigates the viability of the Kinect sensor for joint motion monitoring. To this end, two novel tools are developed that incorporate statistical, image processing and computer vision algorithms. The first tool GLSKEL is an intuitive 3D interface and kinematics model for continuous motion capture and analysis of human gait that can be useful for clinical practitioners. The second tool, JAFAKEC helps with tracking and calculation of joint angles based on point cloud data. This functionality can be very helpful for monitoring of the gait and arm motions of mobility-impaired patients using the Kinect sensor. This thesis also details the mathematical methods and algorithms applied on the point cloud to improve accuracy of the joint angle calculation.

Acknowledgements

I would like to take this opportunity to express my deepest gratitude to my supervisor Dr. Lina Stankovic without whose continuous support and guidance my work would not have been possible. Besides her direction and advice, her keen interest and motivation has helped me stay motivated and focused over the years of my research.

I am also deeply grateful to my co-supervisor Prof. Ivan Andonovic, for his support. Among other faculty advisers, Prof. Philip Rowe and Dr. Vladimir Stankovic were extremely crucial. Dr. Vladimir Stankovic provided vital technical advice and Prof. Philip Rowe made it possible to use Bioengineering Laboratory for comparative studies that form the backbone of my research.

Among Bioengineering laboratory members, my sincerest gratitude goes to Dr. Andrew Kerr, Dr. Bruce Carse, and Dr. Ukadike Ugbolue for helping me with the VICON requirements and settings in the bioengineering laboratory experiments.

I would also like to profusely thank my fellow doctoral students Cheng Yang and Minxiang Ye for their help and cooperation in the bioengineering laboratory experiments, being the subjects and of course for their friendship. I would like to thank my fellow researchers Dr. Jing Liao, Yihan and Jie for enthusiastically participating in the experiments.

I would also like to thank senior researcher Dr. Bruce Stephen and fellow doctoral student Di Cao for providing me the cattle monitoring datasets.

I am grateful for the funding source that allowed me to pursue my doctoral studies, *the Strathclyde University Studentship*.

Finally, I thank my family and friends for being supportive and patient.

Contents

Declaration of Authorship	iii
Abstract	iv
Acknowledgements	v
List of Figures	xi
List of Tables	xv
Abbreviations	xvii
1 Introduction	1
1.1 Background	1
1.2 Application of Wireless Sensor Network for herd monitoring	2
1.3 Application of Microsoft Kinect in Healthcare	2
1.4 Evaluation of Microsoft Kinect for Gait Analysis	4
1.5 Evaluation of Microsoft Kinect for Upper Limb Motion Analysis	4
1.6 Summary and Main Contributions	6
1.7 Publications	7
1.8 An overview of the thesis	8
2 Performance Modelling of Wireless Sensor Networks for Dynamic Assets	9
2.1 Introduction	9
2.2 Related work on WSN modelling	12
2.2.1 Related work on WSN Protocol Designs	12
2.2.2 Related work on Network Simulators	13
2.2.3 Related work on Mobility Models from Animal Behaviour	15
2.3 The WSNSIM Simulation Model	16
2.4 Internal Details of the WSNSIM Simulator	18
2.5 Energy Depletion Model	24
2.6 Statistical Model of Spatial Distribution	25
2.7 Statistical Model of Cattle Mobility	27
2.8 Statistical Model for Heading Direction	29
2.9 Modelling of Spatial Regions	32
2.10 Directional Antennae	33
2.11 Novel Protocols designed using WSNSIM	34
2.12 Protocol Design	35
2.12.1 Modified LEACH	35

2.13	Simulation of LEACH vs Modified LEACH	36
2.14	A Novel Protocol Design - LEMSYP	37
2.15	CSMA based version of LEMSYP (Low Energy Multi-hop Synchronized Protocol)	39
2.16	TDMA based variant of LEMSYP (T-LEMSYP)	41
2.17	Energy Depletion Comparison between CSMA based LEMSYP and LEACH . . .	44
2.18	Energy Depletion Comparison between CSMA vs TDMA based LEMSYP	46
2.19	Verification and Validation	48
2.19.1	Verification of mobility behaviour with more data	49
2.20	Conclusion	51
3	Data Acquisition for Human Motion Analysis	53
3.1	Introduction	53
3.2	Known Challenges of Microsoft Kinect	54
3.3	Literature Review	55
3.4	Microsoft Kinect	57
3.5	Kinect for Xbox 360 and Kinect for Windows V1.8 Joint Positions	59
3.6	Kinect V2 Joint Positions	60
3.7	Depth Data Processing in Microsoft Kinect	63
3.8	Some more mass-market depth sensors	66
3.9	VICON MX System	69
3.9.1	VICON Nexux software	69
3.9.2	Drawbacks of VICON Nexux Software	70
3.10	Precision of Microsoft Kinect - Depth measurement accuracy:	71
3.10.1	Precision Calculation of Microsoft Kinect V2 Depth Sensor	78
3.11	Kinect 1 Recorder	81
3.12	Kinect 2 Recorder	82
3.13	The Kinematics Model	83
3.14	Interactive Measurement in 3D	89
3.15	Validation of GLSKEL	90
3.15.1	Validation of Skeletal Forward Kinematics	90
3.15.2	Validation of Inverse Kinematics	90
3.15.3	Validation of Point Cloud Processing	90
3.15.4	Validation of Interactive Measurement	90
3.16	Conclusion	91
4	Algorithms for Processing 3D Depth Image Data	93
4.1	Introduction	93
4.2	Defining a Topology on the Point Cloud	94
4.3	Geometric Graphs	94
4.3.1	Closest Pairs	96
4.3.2	Nearest Neighbours Graph	97
4.3.3	Euclidean Minimum Spanning Tree	97
4.3.4	Infinite Strip Graph	98
4.3.5	Sphere of Influence Graph	98
4.3.6	Relative Neighbourhood Graph	98
4.3.7	Gabriel Graph	99
4.3.8	Convex Hull	99
4.3.9	Geometric Graph using a Distance Threshold	99
4.4	Estimating Normals at the Cloud Points	101
4.5	Estimating Curvatures at the Cloud Points	104
4.6	Geodesic Distance	104
4.7	Geodesic Centroid on Geometric Graph	105
4.8	Geodesic or Graph Based Clusters	107

4.9	Bayesian Framework for Combining Limb Labels	109
4.10	Conclusion	110
5	Gait Monitoring Using Kinect Sensor	113
5.1	Clinical Gait Monitoring	113
5.2	Literature Review	117
5.2.1	Previous work on applications of Kinect for human lower limb and gait in healthcare	117
5.2.2	Previous work on algorithms for gait capture	118
5.2.3	Previous work on gait feature representation and gait recognition	119
5.3	JAFKEC-G for Gait System Overview	120
5.4	Algorithms for Gait Capture from Depth Image	120
5.4.1	Algorithm : Medial Axis from Binary image	120
5.4.2	Algorithm : Sectioning the 3D point cloud	122
5.4.3	Algorithm : Least Squares Line Fitting	124
5.4.4	Algorithm : Convex Polyhedron Fitting	126
5.4.5	Algorithm : Calculating Average of Nearly Aligned Polyhedral Outlines.	128
5.4.6	Algorithm : Use of Geodesic distance for Limb Labelling	129
5.5	Kinect-based lower limb motion analysis - Methods	133
5.6	Results	134
5.7	Conclusion and Discussion	145
6	Upper Limb Motion Analysis	147
6.1	Introduction	147
6.2	Gesture Design and hand tracking	147
6.3	Literature Review	148
6.4	Arm Motion Monitoring	151
6.5	Experimental Methodology	151
6.6	JAFKEC for Upperlimb Methods	154
6.7	Algorithmic Methodology	154
6.7.1	Normal Estimation	154
6.7.2	Geodesic Distance Labelling	155
6.7.3	Segmenting the point cloud	156
6.7.4	Graph from Geodesic Clusters	159
6.8	Results	161
6.8.1	Arm angle results of Kinect and VICON	163
6.9	Conclusion and Discussion	175
7	Conclusion	177
	Appendices	179
A	Cattle Motion Measurements	181

List of Figures

1.1	Public expenditures on healthcare U.K	3
2.1	WSMSIM architecture. Satellite image © 2011 Google, DigitalGlobe	18
2.2	Algorithm for obtaining cattle spread data from satellite images. Satellite image © 2011 Google, DigitalGlobe	19
2.3	WSNSIM general_event and its derived event types	21
2.4	Probability density functions describing the composition of cattle herds	26
2.5	Number of speed modes or clusters plotted against MIA	28
2.6	Speed distribution in the resting state	29
2.7	Speed distribution in the grazing and shifting states	29
2.8	CDFs and state transition probabilities	30
2.9	Heading Direction	30
2.10	Transition probabilities of some heading directions	31
2.11	Modelling Spatial Regions.Satellite image © 2011 Google, DigitalGlobe	32
2.12	Directional Antennae	33
2.13	LEACH vs. Modified LEACH	34
2.14	Number of Clusterheads in LEACH and Modified LEACH. Satellite image © 2011 Google, DigitalGlobe	37
2.15	State diagram of the CSMA LEMSYP protocol	38
2.16	State diagram of the TDMA LEMSYP protocol	43
2.17	Total Energy Consumption in LEACH and LEMSYP after one hour of simulation. Satellite image © 2011 Google, DigitalGlobe	44
2.18	Comparison of energy consumption by LEACH and LEMSYP	44
2.19	Total Energy Consumption in LEACH and LEMSYP after one hour of simulation - using data from previous work [170]. Satellite image © 2011 Google, DigitalGlobe	45
2.20	Comparison of energy consumption by LEACH and LEMSYP	45
2.21	Comparison of power consumption of node 46	46
2.22	Comparison of power consumption of node 28	47
2.23	Comparison of total (i.e. network-wide) power depletion between T-LEMSYP and C-LEMSYP	47
2.24	Synthetic herd round trip test	48
2.26	GPS fixes	49
2.25	Filtered GPS fixes for a single cow recorded over 7 days	50
3.1	Kinect V1 for Xbox 360	57
3.2	Kinect for Windows V1.8	58
3.3	Kinect for windows V2	58
3.4	Kinect for Xbox and V1.8 Joint Positions	60
3.5	Kinect V2 Joint Positions	62
3.6	The mechanism used by Kinect V1 ([1])	63
3.7	Microsoft Kinect V2 Internals [113]	64
3.8	Operating Principle of a ToF System	65
3.9	Creative Senz3D	66

3.10 Intel RealSense Camera F200	66
3.11 Intel RealSense SR300 Camera	67
3.12 Apple primesense Carmine1.09	67
3.13 Structure Sensor	68
3.14 Leap Motion	68
3.15 Gait Model constructed using VICON Nexus 2.1.1 software	69
3.16 Upper limb model constructed using VICON Nexus 2.1.1 software	70
3.17 Frames captured for the gym-ball	72
3.18 Point cloud for the gym-ball and its best-fit sphere	73
3.19 Position error as a function of distance from the Kinect sensor	73
3.20 Angular error defined in terms of the position error E	76
3.21 Limb's angular error vs. distance from the Kinect. Any distance $\leq 3.5m$ would satisfy the Bioengineering requirement.	76
3.22 Frames captured for the square flat surface	78
3.23 3D representation of point cloud with normals estimated	79
3.24 Position error as a function of distance from Kinect V2 sensor	79
3.25 Single subject using Kinect V1	81
3.26 Single subject Kinect V1 and point cloud viewer	81
3.27 Depth images using Kinect2	82
3.28 Near and far modes using Kinect2	82
3.29 Reference planes used in describing joint motions	84
3.30 Abduction/adduction and flexion/extension angles	85
3.31 The Pose controller Widget in GLSKEL - Forward Kinematics of GLSKEL	85
3.32 Left hip and right hip extension angles - Inverse Kinematics of GLSKEL	86
3.33 Screenshots from the Gait capture tools	86
3.34 Kinematic model, point cloud, and SDK joints from Kinect 1.0 displayed on GLSKEL	87
3.35 Kinematic model, point cloud and SDK joints from Kinect 1.8 displayed on GLSKEL	89
5.1 Human Walk Cycle [82]	115
5.2 Knee flexion and extension angles of gait cycles [90]	116
5.3 Knee flexion angle curves of normal subjects (blue) and osteo-arthritis patients (red) (Source: [38])	116
5.4 Depth frame converted into grey scale	122
5.5 Binarized frames representing the region around a knee	122
5.6 Medial axis skeleton from the depth image	123
5.7 Medial lines detected on binarized images in Figure 5.5	123
5.8 Automatically placed markers obtaining sections of the point cloud	124
5.9 Point cloud segmented by planes perpendicular to the limb's medial axis	125
5.10 A screenshot showing the labelled point cloud and the fitted limb axes	125
5.11 A point distribution and its convex hull.	127
5.12 Volume Integral Calculation	128
5.13 Convex Polyhedron Fitting	129
5.14 Geodesic distance based labelling	131
5.15 Multiple geodesic distance based labelling to improve classification of cloud points	131
5.16 Flowchart for JAFKEC-G gait algorithm	132
5.17 Normal person slow walk	135
5.18 Normal slow walk, Left leg	136
5.19 Patient type walk	136
5.20 Error Distribution using Kinect 1.0, SDK Skeletal data and Least Square Method	137
5.21 Normal slow walk, (Geodesic)	138
5.22 Patient type walk - Left leg, (Geodesic)	139
5.23 Patient type walk - Right leg, (Geodesic)	139

5.24	Error Distribution using Kinect 1.0, Geodesic Distance, Point Cloud data only	140
5.25	Healthy person, Subject 1 - Normal walk	141
5.26	Healthy person, Subject 2 - Normal walk	142
5.27	Healthy person - Normal walk, Left leg	142
5.28	Error Distribution using Kinect 1.8, JAFakec-G, Point cloud data only	143
5.29	Error Distribution using Kinect 1.8, JAFakec-G, best result-set	144
6.1	Kinect Recording Algorithm	152
6.2	Depth map captured for upper limb	152
6.3	Flowchart for JAFakec upper limb algorithm	153
6.4	Point cloud with normals	155
6.5	The cup-holding hand identified using geodesic distance	156
6.6	Geodesic Distance Bands with head as Source	157
6.7	Geodesic Distance Bands with Waist as Source	157
6.8	A distance discriminant based on geodesic distance	158
6.9	Demonstrating second order geodesic distance labelling	158
6.10	Arm motion monitoring	159
6.11	Segmenting the Point Cloud	159
6.12	Super nodes obtained from graph based clustering	160
6.13	Connectivity graph for super nodes	160
6.14	Kalman filter applied to eliminate added Gaussian noise	161
6.15	Unfiltered and Butterworth filtered plots of elbow angle	162
6.16	Upper body joint trajectories	162
6.17	Elbow angle of healthy subject 1, healthy subject 2 (trial 1), Right upperlimb	165
6.18	Elbow angle (trial 2, trial 3) of healthy subject 2, Right upperlimb	165
6.19	Elbow angle of healthy subject2, (trial 4) Right upperlimb	166
6.20	Error Distribution using Kinect V2, JAFakec-U, Healthy Male	167
6.21	Error Distribution using Kinect V2, JAFakec-U, best result-set	168
6.22	Elbow angle of healthy subject3, (trial 1) Right upperlimb	170
6.23	Elbow angle of healthy subject3, (trial 2 and trial 3) Right upperlimb	170
6.24	Error Distribution using Kinect V2, JAFakec-U, Healthy Female	171
6.25	Error Distribution using Kinect V2, JAFakec-U, best result-set	172
6.26	Elbow angle of mock patient	173
6.27	Error Distribution using Kinect V2, JAFakec-U, Mock Patient	174
A.1	Speed Distribution of Cow 11 over 7 days	182
A.2	Movement Pattern for Cow 11 over 7 days	182
A.3	Speed Distribution of Cow 13 over 7 days	183
A.4	Movement Pattern for Cow 13 over 7 days	183
A.5	Speed Distribution of Cow 14 over 7 days	184
A.6	Movement Pattern for Cow 14 over 7 days	184
A.7	Speed Distribution of Cow 15 over 7 days	185
A.8	Movement Pattern for Cow 15 over 7 days	185
A.9	Speed Distribution of Cow 16 over 7 days	186
A.10	Movement Pattern for Cow 16 over 7 days	186
A.11	Speed Distribution of Cow 17 over 7 days	187
A.12	Movement Pattern for Cow 17 over 7 days	187
A.13	Speed Distribution of Cow 18 over 7 days	188
A.14	Movement Pattern for Cow 18 over 7 days	188
A.15	Speed Distribution of Cow 19 over 7 days	189
A.16	Movement Pattern for Cow 19 over 7 days	189
A.17	Speed Distribution of Cow 20 over 7 days	190
A.18	Movement Pattern for Cow 20 over 7 days	190

A.19 Speed Distribution of Cow 21 over 7 days	191
A.20 Movement Pattern for Cow 21 over 7 days	191
A.21 Speed Distribution of Cow 22 over 7 days	192
A.22 Movement Pattern for Cow 22 over 7 days	192
A.23 Speed Distribution of Cow 23 over 7 days	193
A.24 Movement Pattern for Cow 23 over 7 days	193
A.25 Speed Distribution of Cow 24 over 7 days	194
A.26 Movement Pattern for Cow 24 over 7 days	194

List of Tables

2.1	Results of simulation for the large herd (N= 166)	36
2.2	Results of simulation for the small herd (N= 58)	36
3.1	Kinect for Xbox and V1.8 Joint Labels	59
3.2	Kinect V2 Joint Labels	61
3.3	Feature summary of mass-market depth sensors	68
3.4	Joint Variables	83
5.1	Algorithms used in the JAFKEC-G for gait system	133
6.1	Algorithms used in the JAFKEC Upperlimb system	154
A.1	Cow 11 Speed Data	182
A.2	Cow 13 Speed Data	183
A.3	Cow 14 Speed Data	184
A.4	Cow 15 Speed Data	185
A.5	Cow 16 Speed Data	186
A.6	Cow 17 Speed Data	187
A.7	Cow 18 Speed Data	188
A.8	Cow 19 Speed Data	189
A.9	Cow 20 Speed Data	190
A.10	Cow 21 Speed Data	191
A.11	Cow 22 Speed Data	192
A.12	Cow 23 Speed Data	193
A.13	Cow 24 Speed Data	194

Abbreviations

WSNSIM	The W ireless S ensor N etwork S imulator developed for this study.
WSN	W ireless S ensor N etwork
GPS	G lobal P ositioning S ystem
BS	B ase S tation
LEMSYP	The L ow E nergy M ultihop S ynchronized P rotocol developed for this study.
CSMA	C arrier S ense M ultiple A ccess
TDMA	T ime D ivision M ultiple A ccess
C-LEMSYP	The C SMa based variant of L ow E nergy M ultihop S ynchronized P rotocol developed for this study.
T-LEMSYP	The T DMa based variant of L ow E nergy M ultihop S ynchronized P rotocol proposed for this study.
GUI	G raphical U ser I nterface
GLSKEL	The O pen G L based S keletal motions viewer developed for this study.
JAFakec-G	The J oint A ngle from F rames A cquired using K in E Ct for G ait monitoring developed for this study.
JAFakec-U	The J oint A ngle from F rames A cquired using K in E Ct for U pperlimb monitoring developed for this study.

To my loving family and friends...

Chapter 1

Introduction

1.1 Background

Sensor technology has been going through some big advances in the recent years. The main strands of its progress are - (a) tiny radio transceivers are enabling ubiquitous monitoring of spatially distributed parameters, (b) new microelectronic sensing devices enabling precise measurement/detection of new parameters, (c) cheap scanning and imaging sensors are democratising dynamic capture of shape and form. In view of these developments, this work aims at enhancing data processing methods towards greater exploitation of sensor technologies. The explosive growth in data-acquisition and storage capabilities has resulted in what is called the *big data* revolution. The term *big data* broadly refers to analytic processing of vast amounts of data collected from multiple sources in order to enable intelligent decisions. This has given rise to a field of study called *data science* that develops and uses ideas from such diverse fields as statistics, mathematical modelling, signal processing, functional analysis, algorithms, etc. in order to apply them in processing of vast amounts of data. This work is a contribution to the field of data science, as it explores algorithms for dealing with collection, recording and processing of sensor data. In particular two application contexts were explored in this work - (a) that of monitoring livestock using wearable sensor-transceiver nodes, and (b) that of monitoring gait, hand tremor and dexterity of mobility challenged patients without the inconvenience of body mounted sensors or markers. The specialized systems traditionally used in the above applications involve high capital expenditure, running into tens of thousands of dollars. This work has progressed towards achieving a similar level of effectiveness at a much lower cost. The goal of this work was to enable livestock companies, hospitals, and care homes to benefit from inexpensive technologies for sensing, networking, and monitoring.

1.2 Application of Wireless Sensor Network for herd monitoring

Using industrial grade wireless sensor nodes in farm monitoring has many challenges. There are physical design problems to deal with harsh outdoor conditions like sun, rain and snow, but such problems can be addressed by the design of a protective casing. This work addresses the challenge of protocol design to take advantage of herding behaviour in order to drive down the power consumption and thus enhance battery life. Performance analysis of WSNs for cattle monitoring are supported by the simulation model developed in this thesis WSNSIM that captures the behaviour of the farm-area network and animal mobility to the level of detail required for accurate analysis.

The paper *Protocol Design for Farm Animal Monitoring using Simulation*, has been published in ADHOC-NOW July 2012 international conference and in Lecture Notes in Computer Science, LNCS 7363. Monitoring health and estrus (the period of sexual receptivity) of cattle in large herds is quite labor intensive and its efficiency can be much enhanced by WSN deployed on the cattle. Cattle herds have their own characteristic peculiarity in the spatial spread of nodes and their mobility patterns. The electronic sensor in this application is cow collar, transceiver on collar-band. A novel discrete event based network simulator, WSNSIM, has been developed for performance analysis of this class of wireless networks, and used for evaluation of new protocol ideas for cattle herd monitoring. Chapter 2 describes this contribution in detail.

1.3 Application of Microsoft Kinect in Healthcare

The average age of the global population is reaching unprecedented high levels. The people are living longer than ever before, and the number of people living far past their retirement age is increasing rapidly. The population over the age of 65 years is expected to be more than double that of the 1990s levels by 2030 [53]. The retired/senior population will profoundly affect national economy and business productivity of the UK (As shown in figure 1.1). With this steady increase of life expectancy and a steady growth in the senior population, there is a need for innovations towards cost reduction in healthcare. One of the active areas of research in this direction involves the use of consumer grade electronic devices towards health monitoring. Smart phone and its peripherals are being used for monitoring heart rate [99], blood pressure [17], sleep quality, hand tremor [36] and so on. Nintendo Wii sensors have been used in physical therapy [24], stroke rehabilitation [152] and balance feedback [28], [11].

Substantial effort is being made to deploy information technologies into the clinical space - such as administrative technologies (information integration systems, patient record systems, etc.), high tech hospital equipment such as surgical robot, Computerised Tomography(CT), Magnetic Resonance Imaging (MRI), etc. Deployment of technology at home care has the potential to reduce the cost and the pressure of the hospital resources. Advances in technology will make it

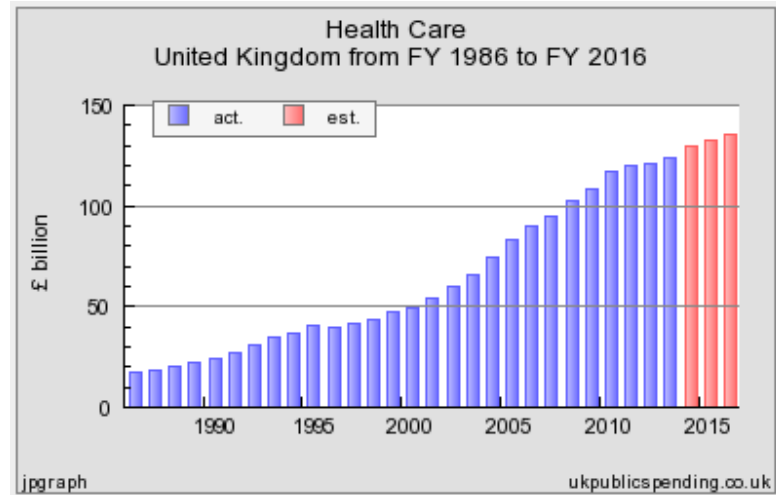


FIGURE 1.1: Public expenditures on healthcare U.K

possible for people to play a greater role in maintaining and monitoring their own health [45], [69].

With an aging population, there are a growing number of people suffering from mobility impairment due to such conditions as stroke, arthritis, rheumatism, osteoporosis, etc. This places enormous demands on healthcare systems, both acute hospital care and also routine monitoring and health maintenance at massive scale. Such patients need to be monitored during the course of treatment to assess their progression and response to their medication and therapy. There are hospital facilities to enable such monitoring but these are extremely expensive and require skilled staff to operate. With improvement in the body tracking technology in the rehabilitation scenarios such as the interactive capabilities, body movements can be captured and measured in greater number of ways. Singh et al. [165] have developed rehabilitative treatment interventions that personalise physical activity sessions and provide feedback to a patient based on relevant body motion to facilitate the awareness and understanding of their movement. With improvements in technology, the body movements can be measured and captured which can be applied in a broader scope of application in the healthcare domain. One of the goals of this study is to enable monitoring of patients using inexpensive 3D imaging sensors. The patients can be monitored in home or in care homes by using the highly affordable and portable Kinect sensor, which can capture gait and arm motions without inflicting any inconvenience.

Gait analysis is frequently used in clinical decision-making [77]. Improved measurement and analysis of gait information has significant clinical, diagnostic and therapeutic value. In the second and third year of research different versions of Kinect were studied, numerous algorithms and developed novel toolsets (called JAFKEC-G and JAFKEC-U) and various algorithms to help with the analysis of gait and tremor in hand captured by the Kinect sensor which can be used for monitoring the gait and arm tremor in home and care homes. Most optical kinematics solutions require large facilities with multiple cameras. Some portable systems with analytical tool-kits have recently been developed [201], [203] use markers on the subject's body. A lot of research has been done in marker-less motion capture systems [124] and various methods have been proposed in the surveys of Moselund et al. [123], Turaga et al. [186] and Aggarwal and Ryoo [3] for human motion capture and analysis. Most of these systems position multiple sensors in

a big workspace, typically a large laboratory setting, and requiring elaborate calibration. These are thus unsuitable for in-home use. The proposed systems provide a marker-less, non-intrusive option for gait and upper limb analysis and monitoring at home or at care homes for the elderly and the infirm.

1.4 Evaluation of Microsoft Kinect for Gait Analysis

The paper *Kinect-based lower limb motion analysis* published in the International Symposium on Biomechanics, July 2015 investigates the viability of using the Kinect sensor for the purposes of gait analysis in patients recovering from stroke. The Kinect sensor is a 3D scanning device that operates at 30 frames per second, capturing depth and RGB data. Use of Kinect has been studied, with little success, for recording of clinical data using analysis [49],[161]. Posture, gait, and mobility data can be produced from 3D point cloud sequence. Physiotherapy programs based on optical movement analysis systems are the norm, and they require large facilities and expensive installations comprising several cameras such as the VICON. The proposed Kinect based solution, JAFKEC-G system, provides a marker-less inexpensive alternative for gait analysis at home or at care homes. Capture of human movements is the primary application of the Kinect but the Kinect SDK does a quick and approximate computation of joint positions, which are not accurate enough for clinical gait monitoring. Chapter 5 presents the paper, which described the gait monitoring system JAFKEC-G. This paper details the mathematical algorithms applied on the point cloud to improve accuracy of the joint angle calculation.

1.5 Evaluation of Microsoft Kinect for Upper Limb Motion Analysis

Gesture recognition, particularly for the arms and hands has attracted substantial attention and as a result various approaches have been invented. Trackable gloves and various wearable devices have been used for gesture recognition, but vision-based approaches (i.e. approaches that can capture hand gestures without requiring wearable devices) tend to be more convenient and natural to use. The applications of gesture recognition include Human-Computer Interaction (HCI) in which gestures are used to represent the input to a computer. It allows for a more intuitive interface for manipulation of virtual reality 3D scenes and augmented reality games. In robotics gestures can be used to control and interact with robots in a more intuitive way. There are applications where medical robots are used with gesture tracking and tactile feedback to allow minimally invasive surgery using a probe inserted through a small incision. There has been attempts to replace mouse interactions with gesture based interactions for common usage like enlarging images, launching and closing an application, zoom-in/zoom-out, icon selection, drawing etc. Wearable virtual reality systems often use gesture for 3D navigation. Another major application of gesture recognition is computer gaming, where Microsoft Kinect has introduced gesture based interaction to the mass market. Human upper-limb gesture recognition can be

applied in the healthcare to allow for acquisition of diagnostic data as well as control of surgical devices. Human upper-limb is a complex and articulated organ consisting of many connected bones and joints. It is estimated that after a stroke, 50-75 % of patients suffer loss of neurological and motor control on their upper limb [128]. Video game controllers have been used in therapy and assessment of stroke survivors [106].

The research work carried out during the third year included monitoring of human hand and arm motions, and development of a new Kinect based solution - JAFKEC-U using Microsoft Kinect V2. Chapter 6 presents the said upper limb motion analysis work in detail.

1.6 Summary and Main Contributions

This thesis explores three core application areas. Each of these application areas have strong influence on how monitoring using electronic sensors can be utilized. The first contribution presented in Chapter 2 looks at monitoring livestock using wearable sensor nodes. In this work a novel discrete event simulator (WSNSIM) is presented, which is designed to evaluate protocol alternatives for performance modelling of wireless sensor networks aimed at livestock monitoring. It has many novel aspects like -(a) a synthetic herd generation module based on statistical modelling from satellite images and GPS data, (b) a new stochastic mobility model, (c) shape annotations (polygonal, lines, etc.) for representing prohibited zones, obstacles, and (d) range of directional antennas, (e) a sophisticated power consumption model. Novel image processing techniques has been implemented in WSNSIM for automatic detection of cattle positions from satellite images. New protocols - viz. modified-LEACH and LEMSYP were proposed and evaluated using WSNSIM with good improvement over the state of the art with regard to farm requirements. A novel protocol (called LEMSYP) has been designed and evaluated to address low power reliable data gathering from dynamic nodes.

For the second contribution presented in Chapter 5, Microsoft Kinect sensor (versions 1.0 and 1.8) have been used for monitoring human gait without using any wearable equipment. A novel monitoring tool JAFKEC for gait (JAFKEC-G) has been developed for visualising and analysing gait data using depth data as captured by the Kinect sensor. JAFKEC-G uses a suite of customised mathematical algorithms like geodesic distance labelling, principal axis, least squares fitting, convex hull etc. that enable marker-less gait capture for healthy and post-stroke patients. It is inexpensive, has a graphical interface, and can be used in home application scenarios. JAFKEC-G may be used at patient's home, care homes, and at doctors' clinics. A novel kinematic model of human body joints (presented in section 3.13), called GLSKEL, has been developed in order to map joint co-ordinates captured from Kinect into a realistic 3D visualisation of the motions. The key scientific contribution of GLSKEL is that it enables continuous motion capture (as opposed to measurement of aggregate characteristics) in an interactive 3D environment. The intuitive interface of GLSKEL can help bridge the gap between a research prototype and a novel product that can be used by clinical practitioners.

The third thesis contribution (presented in Chapter 6) shifts emphasis to consider these technologies in the context of upper limb motion analysis and monitoring. A novel system (called JAFKEC-U) for upper limb motion monitoring has been developed which is also marker-less and is suitable for home-based monitoring. This system uses Microsoft Kinect 2 sensor for monitoring human arm and hand motion. JAFKEC-U also uses a suite of mathematical algorithms like graph based clustering, geometric feature extraction, geodesic labelling etc. to enable arm angle capture.

1.7 Publications

1. [153] Shikha Sarkar, Lina Stankovic, and Ivan Andonovic. Protocol design for farm animal monitoring using simulation. In *Ad-hoc, Mobile, and Wireless Networks*, Belgrade, Serbia, pages 126-138. Springer, 2012.
2. [154] Shikha Sarkar, Lina Stankovic, Andy Kerr, and Philip Rowe. Kinect-based lower limb motion analysis. In *XXV Congress International Society of Biomechanics*, Glasgow, UK, 2015.

1.8 An overview of the thesis

In presenting the systems and algorithms mentioned above, this thesis is organized into the following chapters.

Chapter 2 discusses the Performance Modelling of Wireless Sensor Networks. It includes detailed description of a novel simulation model, WSNSIM, for herd monitoring, literature review, novel protocol design, spatial distribution models, energy depletion comparison and mobility models.

Chapter 3 discusses data acquisition for human motion analysis. Microsoft Kinect sensor along with other inexpensive depth sensor technologies, and VICON MX system have been discussed. This chapter also includes the literature review on limitations and applications of Kinect and other state-of-the-art sensors. It presents an experimental and mathematical analysis of precision limits of Kinect V1 and Kinect V2 sensors. Development of the Kinect data recorders (for both V1 and V2) is discussed in this chapter, along with a description of the novel Kinematic Model (GLSKEL) for interactive 3D animation of joint angles.

Chapter 4 discusses different Point Cloud Algorithms that are used in the literature and in the subsequent two contribution chapters.

Chapter 5 introduces the novel gait monitoring system JAFKEC-G. It also includes a survey of relevant literature pertaining specifically to the use of Kinect sensors for gait analysis. It presents methodologies, detailed description of the JAFKEC-G algorithms, and results obtained from Kinect sensors.

Chapter 6 discusses the novel JAFKEC-U for upper-limb monitoring system that uses Kinect V2. The chapter also includes a survey of the previous works pertaining specifically to the use of Kinect sensors for upper-limb analysis, and goes on to present the novel JAFKEC-U algorithms and methodology used for calculating arm joint angles from depth frames captured by Kinect sensor.

Chapter 7 presents the conclusion and the future work.

Chapter 2

Performance Modelling of Wireless Sensor Networks for Dynamic Assets

2.1 Introduction

The primary advantage of Wireless Sensor Networks (WSNs) is that its ad-hoc and wire-free deployment offers low-cost monitoring of large facilities and sites. Freedom from wiring also makes it robust with respect to failure modes involving wire-line disconnection. However it comes with a few challenges that wired networks do not have. These emanate from additional considerations such as interference, lower bandwidth, antenna range limitations, power consumption etc. Among them the challenge of power consumption is of paramount importance because being not tethered to an unlimited power supply, the wireless nodes have to carry their own limited battery power, which imposes a strong constraint on their operation [46]. A successful sensor network should be able to run for years without requiring maintenance. This is a challenge that requires taking advantage of every last opportunity of the application. The primary source of battery optimisation is minimization of the number and range of transmissions by WSN nodes and the duration of active listening by the nodes. This entails that the protocol design (which determines the transmission and active listening patterns) has to be customized according to the application at hand. Planning based on simulated performance models can help with the said optimisation process. The objective of this contribution is to develop a performance model for wireless sensor networks deployed on mobile assets. Designing WSNs for mobile assets is much more challenging than that for static assets because the network topology is changing as the hosts move. There are four aspects that must be modelled for the simulator to accurately compute performance parameters relevant to design.

1. Spatial distribution and mobility of hosts.

2. Physical phenomena - radio propagation (noise, error rates, collision), and power consumption.
3. The protocol (the algorithm and heuristics by which the hosts carry out the communication).
4. The traffic patterns (the pattern of need for communication).

The objective of the research is to develop a performance model that simulates the aforementioned aspects of a dynamic WSN and computes the following performance parameters as a by-product of the simulation process:

1. Power consumption efficiency of the network.
2. Latency of the communication.
3. Reachability of hosts.
4. Reliability of communication.

Animal rearing is a profitable but challenging business. On the one hand, owing to the overlap with public health and epidemiological concerns, its practice is highly regulated, and on the other hand owing to its fungible and perishable products, these businesses always remain under tight economic competition. Any technological advance that can enhance safety and competitiveness would be readily embraced by this industry. Monitoring health and estrus of cattle in large herds is quite labor intensive and its efficiency can be much enhanced by WSN deployed on the cattle. Sarkar [153] presents a simulation model for performance analysis of large scale deployments of this application of WSN. Cattle herds have their own characteristic peculiarity in the spatial spread of nodes and their mobility patterns. This fact ensures that the commonly used spread and mobility models [192] are not directly applicable in performance modeling specific to this application. A simulation model tailored for this application must also allow for: (i) directional signal propagation from WSN nodes mounted on cow-collars due to RF absorption by the cow's body, (ii) movement limitations in a typical farm such as fences, water bodies like ponds or pools, etc. (iii) model augmentation using herd behavior captured from satellite images and GPS tracked data. We noted that these custom features were not readily implementable on the existing network simulators (surveyed in [176]). Based on these requirements, a novel discrete event based network simulator called WSNSIM was developed for performance analysis of this class of wireless networks, and used it to evaluate new protocol ideas for cattle herd monitoring. WSN applications have to address additional design constraints that must be taken into account in the optimisation process, some of which are - (a) limited bandwidth of unlicensed bands used for WSNs (b) possible high node density due to large number of hosts in a confined area (c) rugged environments characterised by high RF noise and host failures (d) possible time-correlated traffic leading to bursty overcrowding of the channel. The presence of these multiple constraints and performance metrics that vary greatly across application scenarios entail that WSNs must be tuned to application scenarios at multiple layers. In TCP/IP the application layer can assume a high-level end-to-end communication model, whereas the routing

details may be taken for granted. In WSN, however, the routing needs to be tuned by application to achieve the necessary optimality. Thus, unlike other protocol stacks, WSN protocol stacks require customisation interface or application programming interface encompassing lower layers. This additional exposure of complexity can make it quite challenging to develop applications. One of the objectives of this research is to simplify the interface by which one specifies the low-level behaviour in the protocol. This objective is planned to be achieved by the simulation model through a clearly isolated interface for specifying protocol behaviour which would represent the application development interface to a WSN (could be identical to it, except for syntactic differences). Discrete event simulator like TOSSIM [108] is quite limiting as it is bound to the TinyOS and does not allow simulation beyond that of TinyOS nodes. Simulator ATMEU [139] is specific to the MICA2 platform and hence it is not suitable for general freeform protocol research. The main limitation of the Java based discrete event simulator NetTopo [163] is that it is not an event based simulator. MIXIM [97] and Castalia [18] are two WSN simulators based on OMNET++ framework [187] framework that involves programming in the low level message-handler interface. WSNSIM introduces a simpler programming interface which is much more compact and easier to understand. The simplicity and efficacy of this simulator interface would directly entail simplicity and efficacy of the WSN application programming interface (API) of the platform that the simulated hosts represent. The main novelty of WSNSIM is in the low complexity of the specification interface and its value-added mobility models derived for farm scenarios.

2.2 Related work on WSN modelling

This section will outline the state of the art that this work builds on. There are three branches of predecessors of this work (i) that of WSN protocols (ii) that of WSN simulators and (iii) that of farm animal behaviour relevant to WSN performance. So the following subsections survey related past work on these three aspects.

2.2.1 Related work on WSN Protocol Designs

The LEACH (Low Energy Adaptive Clustering Hierarchy) protocol [72] is one of the most cited protocols as this work led to a whole family of protocols that were based on LEACH in some way. The idea of LEACH is that it is a fully decentralized algorithm that works on simple nodes. Some of the nodes randomly self-elected as *cluster-heads* (and there is a rule not to self-elect too often), and the other nodes subscribe to their nearest cluster-head (the nearest node is estimated by the signal strength of the cluster-head advertisement packet). Then the cluster-head coordinates a TDMA (Time Division Multiple Access) schedule for a data transmission phase when the nodes transmit their sensor data to the nearest cluster-head, which then transmits collected data to the base station. LEACH meant a big improvement over direct transmission because the TDMA phase entails sleep scheduling, and because the two-hop transmission entails reduced transmission length (though not always) and possibility of data compression by the cluster-head. Another reason why this paper is often cited is that it gives mathematical formulae for power consumption in transmission and reception. These formulae have been used by many subsequent works, including this work. An improvement to LEACH [72] is proposed by the DAC algorithm [71], which seeks to improve the spatial spread of cluster-heads by asynchronous pre-emptive cluster election. The key idea is that there is a designated time interval for nodes to self-elect, during which nodes keep their receivers on so that if it hears of a nearby cluster-head, it gives up on its own possibility to self-elect. When a node decides to self-elect (given that it has not heard from any nearby self-elected cluster-head) it immediately advertises itself so that it pre-empts to discourage all nearby wannabe cluster-heads. This paper was useful in one other way, in that it introduces a metric for efficacy of a routing, expressed in terms the ratio of clustered transmission length to the transmission length of direct communication to the base station. This ratio represents the power efficiency of the network topology used for transmission. This metric was useful during a phase when the calculation of power estimation in the simulator, WSNSIM, was incomplete. The paper [136] introduces a protocol called KMMDA (K-Means like Minimum Mean Distance Algorithm) which seeks to improve LEACH by improving the spatial clustering. Since LEACH's cluster-head election does not take into account node positions, it might lead to much skewed spatial spread of cluster-heads. KMMDA makes some big demands on the node sophistication, in that each node is assumed to have a GPS receiver, and all the nodes get to know the positions of other nodes within their range using a TDMA. Firstly, having a GPS receiver significantly increases the cost and energy demand of a node, and the cost of spreading the position information to other nodes is glossed over. So while this protocol does demonstrate lower transmission lengths due to better spatial distribution of cluster-heads, it requires more expensive and energy demanding hardware. The protocol presented in [178] is quite similar to

KMMDA. The PEGASIS (Power-efficient Gathering in Sensor Information Systems) protocol [110] presents another improvement over the LEACH protocol. In PEGASIS, the nodes form chains from sensor nodes so that each node transmits and receives from a neighbouring node, and each member of the chain backbone has a designated next-forwarder. The gathered data move along the chain backbone and aggregated at each step, eventually reaching the base station. The STEM protocol presented in [157] proposes the key idea of a low duty-cycle periodic wakeup interspersed by long sleep (a low energy non-radio-listening state) intervals. A STEM node is woken up either by a continuous tone or a series of frames in two different variants of STEM. In STEM, the data channel is different from the wakeup channel. The MR-MAC protocol is similar protocol to STEM, but it uses two different channels of widely different frequency bands - a low data-rate, low frequency channel for exchange of wakeup and control packets (when the traffic is calm) and a high frequency (high energy) channel when the traffic gets busy.

2.2.2 Related work on Network Simulators

The first simulator evaluated was TOSSIM [108]. TOSSIM is a discrete event simulator for TinyOS networks. It is very efficient and has support for direct connection to hardware. It is tightly bound to the TinyOS capabilities and does not allow simulation of platform capabilities beyond that of TinyOS motes. This fact was quite limiting because the transmit function of TinyOS was not parameterised by the transmission range, nor did the receive functionality seem to record the received signal strength (at the time of evaluation, which may have since been addressed). Manipulating the transmission range and reading reception range programmatically are two key features needed for testing low-level protocol variations. The absence of these features at the time of evaluation (as of 2011) made TinyOS appear unsuitable for the intended investigation. Newer version of TinyOS may have addressed these shortcomings.

TOSSIM was a practical tool for specifically deploying TinyOS motes in applications, but looked like a wrong choice for general protocol research because its deployment is limited to a particular board (motes), and does not comply with open standards. Among the network simulators surveyed, OMNET++ [187] seemed to have the best look and feel among open source tools. It defines a general simulation framework with no inherent ontological commitment to the domain of network simulation. It can be used in a wide variety of simulation applications like manufacturing simulation, shop floor modelling etc. In OMNET++ the basic unit of structure is a module. Modules are connected together and have parameters to control their behaviour. Modules can be hierarchically defined and instantiated. The basic unit of behaviour in OMNET++ are messages. Messages are generated and consumed by modules. Primitive modules are implemented as C++ classes derived from the class `CSimpleModule`. Messages are also C++ classes derived from the class `CMessage`. The simulation kernel delivers messages to the virtual method `CSimpleModule::handleMessage(CMessage*)` for handling of messages. The module authors override this virtual method to define node behaviour. The entire behaviour of a module is encoded as a huge set of `if` branches in the handle message function to check various types of messages and act accordingly (usually sending more messages). Likewise, messages are specialized by sub-classing from an existing message type or from the root type `CMessage`. The entire protocol has to be modelled in terms of the aforementioned primitives. Besides, OMNET++

supports a domain specific language (DSL) to describe connections and structural composition. There are two frameworks built on the top of OMNET++ that target WSN applications - MIXIM [97] and Castalia [18]. These frameworks implement channel characteristics and radio models. MIXIM has many good qualities, in that it has a library of protocols built in, to make it easier for researchers to compare their ideas with existing protocols. The extension interface of Castalia and MIXIM remains the same as that of OMNET++, which is more complex than it needs to be. The proposed lambda based behaviour specification is much more compact and easier to understand than the *message handler call-back* method of Omnet++. The simulator NS3 [73] is a pure C++ library with Python bindings for which the simulation scenario is implemented as a 'main' function. The tool GTNetS [146] also supports the same method of usage, i.e., the simulation user writes the scenario as a main function using the facilities of C++ class library. These libraries have classes representing nodes, links, traffic, probability distributions, etc. When the simulation file is written, it is compiled into an executable which runs the simulation. As the simulation runs it saves one or more text files to log the data generated by the simulation which can then be analysed to estimate the network performance. So it might involve quite a substantial effort to write a WSN simulator using these libraries. This method of user interfacing is not very user friendly for a WSN. In WSN simulator it is best if some part of the input (especially the spatial node distribution) is defined graphically. It is also very useful if it is possible to visualise the mobility of nodes and network transmissions through an animation. It appears that NS3 and GTNetS are quite specialised on internet protocols (TCP/IP, UDP, WIFI, P2P, etc.). The simulation for NS3 and GTNetS runs primarily in non-interactive batch mode, i.e., define the input and let it run. In order to evaluate strategies like data gathering using a mobile collector node, it is useful to support interactive simulation - one in which the user can control simulation entities (e.g., the collector node) as the simulator runs. GTNetS, NS3, OMNET++ all have restrictions for commercial use, while academic use is free. Commercial licensing can be expensive. Another widely used simulator NS2 [117] is the precursor of NS3, and has the same usage model as NS3. It is still the most widely used simulator due to its long history of adoption. It too was primarily designed for the TCP/IP stack, and it supports only two wireless MAC protocols - 802.11, and a single-hop TDMA protocol. It has a stiff learning curve and requires advanced skills to perform meaningful simulations. It is difficult to customize it for WSN modelling. The paper [163] presents a Java-based WSN simulator, NetTopo, that provides both simulation and visualization functions to assist the investigation of algorithms in WSNs. NetTopo provides a common virtual WSN for the purpose of interaction between sensor devices and simulated virtual nodes. It supports the simulation of extremely large scale network and is useful for rapid prototyping of an algorithm. The main features of NetTopo are that - it is platform independent, extensible, flexible and practical. Users can define their own virtual sensor nodes with expected attributes, own algorithms and functions, customise sensor network topology layout, can create different device-based wrappers for visualization. NetTopo consists of several software modules such as node module, topology module, algorithms module, GUI module, painter, main control etc. The node module adopts factory method pattern that deals with the problem of creating objects without specifying the exact class of object that will be created. The algorithm module allows users to create their own algorithm that can be routing, clustering, controlling, scheduling, etc. which can be applied in the virtual WSN. The topology interface is used to define several basic methods representing actions of network topology. The

painter module interacts with Virtual WSN, node and GUI modules and is responsible for painting all kinds of symbols on the canvas. The GUI module displays a canvas, provides a tool bar and menu bar for action performances. The main limitation of the Net-Topo simulator is that it is not an event based simulator. So the timing characteristics and collision behaviour are not modelled by it. Such simulators are good for evaluating initial ideas, but for accurate estimation of protocol performance, one has to model the protocol at event level. The paper [139] presents the design and implementation of a WSN simulator called ATEMU. ATEMU provides simulation along with hardware emulation at a very low level. This simulator is specific to the MICA2 platform. It uses an XML based configuration interface. The ATEMU emulator can simulate arbitrary number of nodes, each of which can be configured to run a different sensor networking application. This simulator too is quite tightly bound to a particular platform (MICA2), and did not appear suitable for protocol research.

The WSN simulators surveyed above do not comprise an exhaustive list. The paper [176] presents an excellent survey of simulators available for WSN modelling. It presents 14 different simulation tools and points out their key features and limitations. This paper describes the key functional components of WSN simulators and presents comparisons of the available simulators with regard to these functional components.

2.2.3 Related work on Mobility Models from Animal Behaviour

The paper [66] reports a body of work in which sizable cattle herds were recorded using navigation equipment and a probabilistic mobility model was made from the recorded data. The model was used to predict spatial distribution of cattle. The proposed approach in this thesis also follows a probabilistic mobility model based on the cattle motions data recorded by [170], and based on satellite images obtained from Google-maps web service. A statistical model for herds was obtained by analysing cattle positions tracked in a number of satellite images. The image processing techniques used in extracting cattle positions from the image data are available in [19], [182]. Herd mobility and distribution seems to lend itself to statistical models, but it is possible to have more sophisticated rule-based motion models as in [114]. Majumder [114] presents a rule based motion model involving various kinds of vehicles within a port facility. If there is a need to repurpose the simulator into tracking of vehicular assets in an industrial compound (airport, large factory etc.), it would be necessary to utilise similar kinematics and rule based scriptable mobility models. The paper [170] analyses the movement patterns in an experimental herd to determine the connectivity of the network for various radio ranges. For each tracked cow, the minimum distance from its nearest base station, and the minimum distance to its nearest cow were analysed for viability of single hop and multi hop communication. The paper presents a situation where there is a base station outside the field, and one at a water trough, since it is a well-known congregation points. It was observed that the cattle gather close to the water trough over a specific two hours in the afternoon. A Depth First Search (DFS) was used to determine the connectivity for a given herd distribution and given radio range. The nodes visited by the DFS were identified as reachable by multi-hop communication. Regularity was noted about the time of the day and the network connectivity, which indicates that behavioural routine could be used to predict viable connection times allowing node energy to be conserved at all other

times. The paper [100] suggest implementation of protocols for herd management. It proposes duty cycles for transmission that is necessary to let the base station pick up data messages when the cattle briefly comes in its proximity. It proposes putting the base station at a well-known congregation point (the water trough) radically new ideas like using a mobile data collector (e.g. mounted on a dog, or on a farmer's tractor, to make occasional rounds of the field to collect the data. The key idea of this protocol is to take advantage of a special point-of-interest and the time an individual cow spends in its proximity, in order to design a transmission duty cycle accordingly. Two measures of network availability are defined (1) Connection availability (CA) - the fraction of time 't' during which the wireless sensor node on cow 'i' is able to communicate with a base station. (2) Connection duration (CD) the time interval during which the connection between a sensor node 'i' and the base station remains uninterrupted. This paper also suggests a Gaussian kernel based distribution of cattle. In summary, the paper introduces the communication challenge posed by short radio range, and cattle mobility. The paper does not implement a concrete protocol per se but proposes ideas and issues for protocol design. The paper [44] reports experimental results from evaluation of two data gathering methods : (a) a routing based data gathering and (b) a moving collector based data gathering. The key feature is that the transmission range is fixed and short, so that while the power consumption per transmit is low, the network might be quite disconnected and fragmented much of the time. A novel data collector scheme is proposed in which the data collector is not battery constrained, i.e., it has a high gain antenna and a large battery pack and can undergo frequent battery change. The collector moves around the field, ideally scans the field to pick up short range signals of the WSN nodes. The paper [170] presents a mathematical model of cattle herd behaviour using a formalism called *Markov Random Field*. This model takes as input the GPS tracked positions of cattle and tries to identify social and local preferences in cattle behaviour inferable from the dataset. The paper [89] describes the modelling of cow behaviour using stochastic automata with the aim of detecting lameness. Three different stochastic automaton models are proposed for describing cow activity. These models describe the cows' activity in the two behavioural scenarios, non-lame and lame. The transition probabilities of the fault model reveal that the probability of the measured data sequence belonging to either of the fault models has to be very low before a transition becomes probable. Diagnosis algorithms for the three approaches are implemented and tested using the real data measurements. The paper [88] has made a statistical model of Oestrus probability by correlating behaviour data with successful bovine pregnancies by artificial insemination. If it is feasible to implement such statistical models on embedded processors on the WSN nodes, it won't be necessary to transmit every reading - some analysis can be done locally on the node and transmission would be done only when the change warrants intervention from the staff. Similarly with lameness, if the intelligence of detecting an abnormality can be embedded on the nodes, it would be possible to reduce the communication overhead significantly.

2.3 The WSNSIM Simulation Model

The architecture of WSNSIM is shown in Figure 2.1 in terms of its functional components. The core of WSNSIM is written in C++ and the graphic user interface (GUI) is written in a popular

scripting language called Tcl/Tk [129]. The simulation scenario is captured interactively from user inputs, in a data format called “.sim”. The modeled scenario captured in the sim file includes such items as - the geometric shape of the rearing space (fences, prohibited regions etc.), initial herd configuration, directional antenna range, protocol parameters etc.

WSNSIM uses image processing techniques to extract cattle positions from satellite photographs and uses statistical methods of distribution fitting to create a probabilistic model for generation of herd scenarios.

WSNSIM supports definition of obstacles and fences (forbidden regions) in a two stage manner. Firstly one can define region annotations in the form of polygons. The orientation of the polygon is used to represent whether it bounds its inside or outside. The screenshot embedded in Figure. 2.1 shows the boundaries of a farm represented in WSNSIM. Previous investigation [170] has shown directional radio propagation properties for cow-collar transmitters. The antenna lies on one side of the cow’s neck, which casts an electromagnetic shadow on the other side of the cow. Moreover, electromagnetic waves are primarily dipole radiations, so the wave propagation consists of lobes. In the presence of such directionality of transmitters, the simulator must take into account the shape of the directional range, in order to be accurate about receptions and interference. WSNSIM supports polygonal definition of antenna ranges, in addition to the default circular ranges.

A Markov chain (i.e. probabilistic state transition) based model of node mobility is superposed in WSNSIM with a discrete event simulation of network events to allow modeling of protocols. Radio models, packet loss models, and power depletion models are invoked alongside the discrete event simulation of the protocol activities.

WSNSIM implements discrete event simulation kernel in C++. There is a queue of future events, which is unsurprisingly called the *event queue* of the simulator. Every event has a stipulated time of occurrence. Events are executed one after another by popping them from the event queue, so that their execution order is the same as the order of their stipulated time of occurrence. The *time* in the simulated world jumps between the times of consecutive events. If there is a big gap between the times of two consecutive events, the simulation time makes a big jump, whereas when event times occur densely the simulation progresses in small steps. The simulator core is a small piece of code that pops one event after another from event queue and executes them. The events are modelled by a C++ class hierarchy whose root class is **general_event**. The event queue is a queue of **general_event** pointers, and the framework calls a particular function (called *occur*) on the **general_event** pointer. This function is a *virtual* function, so that the resulting executed code is that of the *occur* method of the true class of the object pointed by the **general_event** pointer. As the events execute new events get scheduled (for example, when a **transmit_start** event executes, a corresponding **transmit_end** event is scheduled depending on the bit-rate and the packet size. The simulation progresses as one event leads to another. These events are at a particular layer of the simulator, which is largely hidden to the protocol specification interface.

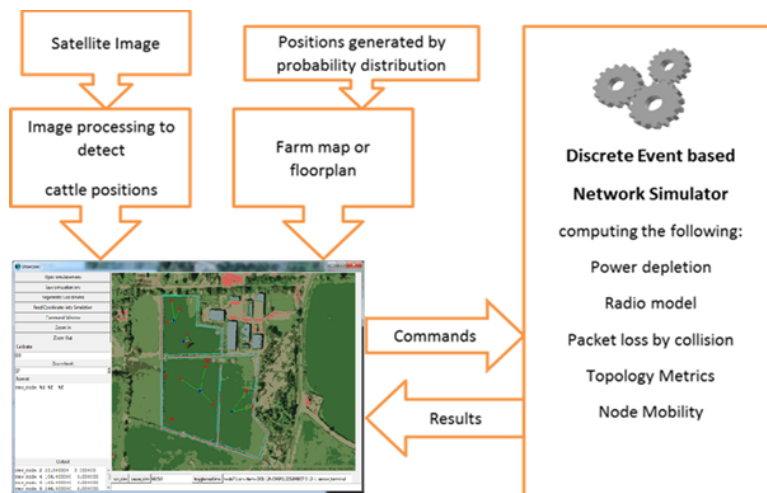


FIGURE 2.1: WSMSIM architecture. Satellite image © 2011 Google, DigitalGlobe

2.4 Internal Details of the WSNSIM Simulator

Following is a list of protocol aspects that needs to be assessed and customise in order to optimize performance in relation to a specific application of WSN. Essentially it needs to be able to simulate and compute protocol performance, reliability, and energy expenditure.

- CSMA MAC parameters and algorithms.
- Radio channel characteristics. These may be characterised by error rates and attenuation functions.
- Energy expenditure rates for each individual type of action as a function of relevant parameters. These may be characterised experimentally for particular node types.
- Node mobility and spatial distribution patterns.
- TDMA and sleep scheduling.
- Clock skew and its bearing on length of TDMA and sleep schedules.

There are a few fundamentally different approaches to simulation in terms of level of detail. On one end of the range there are 3D field models based on partial differential equations (Maxwell's) governing electronic flow and electromagnetic fields and waves. The next level of simulation is that of circuit level models based on ordinary differential equations (e.g. SPICE). The next level is that of discrete event simulation, followed by discrete time and cycle accurate simulation. Discrete event simulation can model the delays of all significant internal operations within the protocol and can help identify timing conflicts and collisions. Any coarser model (e.g. *cycle-accurate* or *round-accurate* simulation) glosses over the timing and power consumption characteristics of low level events, and hence misses out crucial details. Detailed timing computation is very important in WSN energy performance evaluation because the timing of each energisation level has a strong bearing on power consumption, and on the efficacy of transmission and reception. For example, a LEACH advertisement packet lost due to collision might imply that a node selects

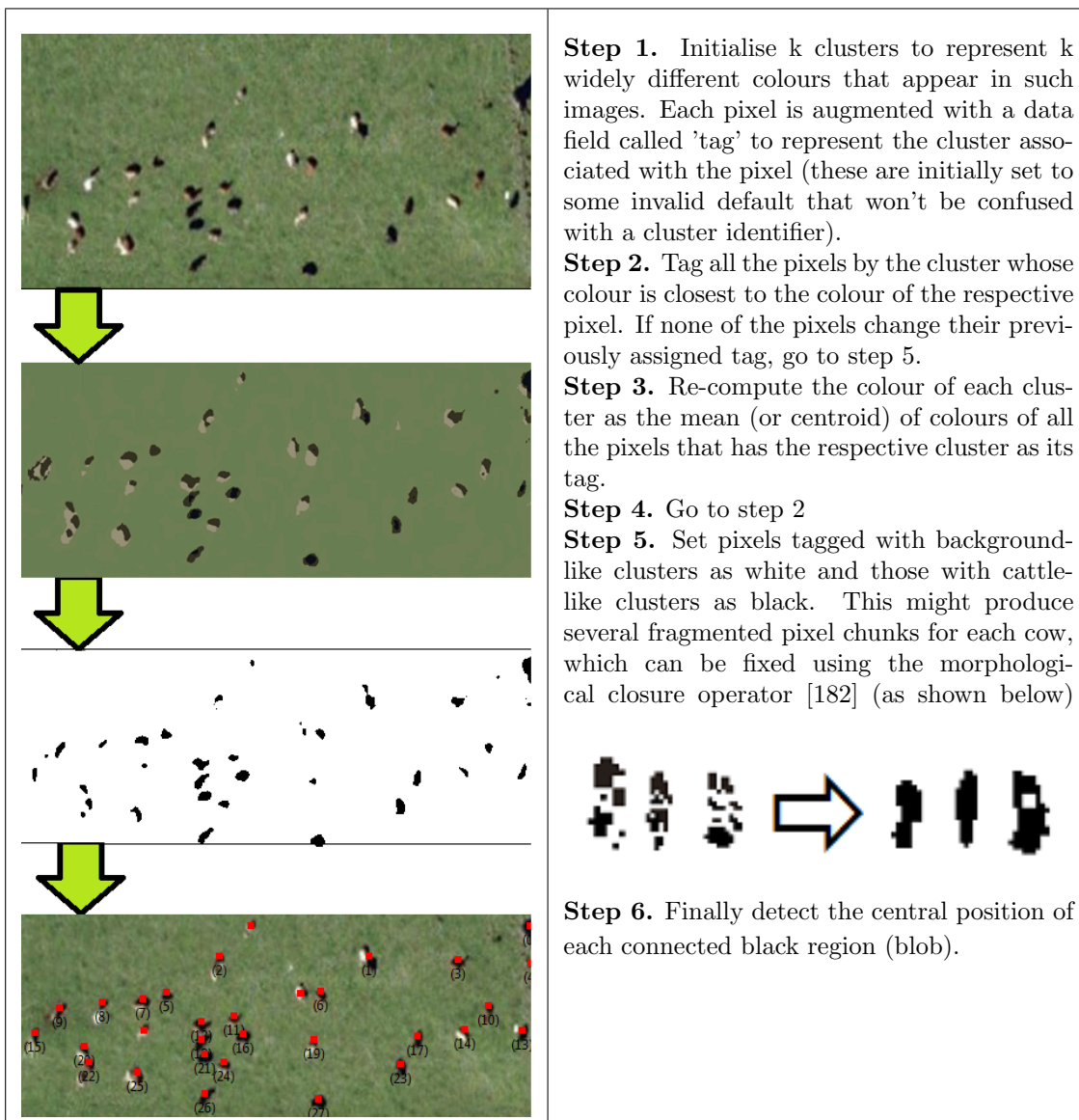


FIGURE 2.2: Algorithm for obtaining cattle spread data from satellite images. Satellite image © 2011 Google, DigitalGlobe

a far-away cluster-head as its cluster-head, and thereby leading to higher energy of subsequent transmission.

Discrete event simulation is the dominant method for network simulation, and in fact almost all of the current network simulation environments use discrete-event simulation methods to simulate the behaviour of a network. The use of discrete event model entails that the simulation progresses through occurrence of instantaneous events at discrete points in time, and the network state can be safely ignored between those discrete instants. The effect of continuous-time phenomena such as power dissipation can be lumped or accumulated at the discrete event instants. A key technical effort involved in designing a discrete event model is to identify events of interest. For example, while the act of a node transmitting a packet in a wireless network can be a complicated sequence of tasks (e.g. encoding individual bits into a waveform, calculating error control codes, symbol detection at the receiving end, decoding, and populating shift registers),

the simulation may model it as a few events such as (1) start of transmission (2) end of transmission. If the propagation delay is significant (which is not the case for the localised nature of sensor networks) then the start and end of reception might be significant events too, with timing characteristics dependent not only on the transmission time but also on the distance. Such events occur in increasing order of their time. The simulation model maintains an ordered list (or so it appears) of events, from which the very immediate event is picked up and executed at each step of simulation progress. As an event is executed, new events might get registered and existing scheduled events might get withdrawn depending on the semantics of the currently executed event. When a new event is registered, it is scheduled with a time of occurrence and the ordered list of future events is readjusted so that events occurs in the right order. The simulation has a current time, which is the timestamp of the current or the most recent event that occurred. The *single step of progress* for the simulation is as follows.

1. Remove the earliest pending event from the list of future events.
2. Set the current simulation time to the timestamp of the just removed event.
3. Carry out the state changes that occur due to the event action.

These state changes might include the creation of one or more additional events. For example, if the current event is that of a node sensing the carrier (as in CSMA) for transmitting a packet and if it finds the carrier busy, it would lead to creation of another carrier sense event for the same packet at a later time. This is the essence of the event processing step, which could be executed in an infinite loop. However, since the simulation needs to coexist with a GUI toolkit, the event processing function is invoked repeatedly from the event-loop of the GUI toolkit (the GUI has a separate event loop for processing of user interaction events such as mouse-clicks, keystrokes etc.). The aforementioned simulation event processing step **process_single_event** and invoke it from the *idle* handler of the GUI library. The key architectural aspects of the WSNSIM model are described as follows. The simulation is implemented in C++ and there are a few classes to model the various aspects of the protocol. The global aspects of a simulation instance are modelled by the class called simulation. The simulation has a **process_single_event** method, which gets called, to progress the simulation by a single event. Besides the simulation class, there are classes to represent various event types, nodes and packets. The essential code of the **process_single_event** is listed below. There are extra code in the actual program to implement pausing (preventing simulation progress beyond the pause time), and the optional real-time behaviour (preventing progression of simulation beyond progress of wall-clock time), however the following describes the essence.

```
void simulation::process_single_event()
{

general_event * next_event = _future_events.top(); // Get the next event

double evt_time = next_event->get_time(); // Get its timestamp
```

```

_time = evt_time;      // Set the simulation time as that timestamp

_future_events.pop(); // Remove the event from the future events queue

next_event->occur();  // Execute the event
}

```

The event queue is implemented using `std::priority_queue` of the standard C++ library, which is essentially a max-heap data structure, which can be made to behave as a min-heap using an appropriate comparison operator. In our case, the declaration of `_future_events` (a data member of simulation) is as follows:

```

std::priority_queue<general_event*,std::vector<general_event*>,
event_ordering_operator> _future_events;

```

It contains pointers to `general_event` and maintains them in such a way that the `top()` method gets the element (i.e. event) with the smallest value of time, and the `pop()` method removes that element. The class `general_event` serves as a base class for all types of events that can occur in this simulation. There is a virtual method called `occur` that is overridden by all classes derived from `general_event`. The following figure 2.3 from Visual Studio *Class View* shows the classes derived so far from `general_event`. The aforementioned `process_single_event` calls



FIGURE 2.3: WSNSIM `general_event` and its derived event types

the virtual function `occur` on the general event pointers, but these pointers point to instances (i.e. objects) of more specific event types, and as such the actual method invoked by the above virtual call depends on what type the object actually is. If the current event pointed to by the `general_event` pointer is actually a `carrier_sense_and_transmit` object, then the C++ virtual

function mechanism ensures that the function `carrier_sense_and_transmit::occur` is called. The manner in which the same method name gets specialized definitions based on object's *type* called polymorphism. Following is a description of some of the aforementioned event types.

The `carrier_sense_and_transmit` event represents the act of transmission attempt with carrier sensing. This event captured on the node that is doing the carrier sensing and the packet that is being attempted to be sent. When this event takes place, its *occur* method checks if the node in question is within the range of a transmission that is currently taking place (i.e. whether there is a nearby transmission whose `transmit_start` has occurred but `transmit_end` has not yet occurred). If so, it schedules another `carrier_sense_and_transmit` event to occur at a later time. If the carrier is not found busy, it schedules a `transmit_start` event with the packet in question at a time determined by the latency of the carrier sensing operation turnaround. The `transmit_start` event represents the start of transmission of a packet. It changes the status of the node to *transmitting* and registers the node as one of the *currently transmitting* nodes within the simulation object. This registration was not strictly necessary, as a global computation could check the *transmitting* status of the node, but is rather an optimisation to avoid having to search through all nodes to determine which nodes transfer it. With this orientation only the currently transmitting nodes must be searched. Its *occur* method schedules a `transmit_end` event after a time interval determined by the packet length and the bit-rate of transmission. In order to detect collisions, when a new transmit event is registered, the simulator goes through all the currently active transmissions, and if there are any nodes within the intersection region of space (i.e. the region of overlap between the new and existing transmissions), all those (transmission, receiver) pairs are marked as collided. For these collided receivers, the pair (the new transmission, receiver) is also marked as collided (i.e. not-only the ongoing transmissions are garbled, the new one is also garbled due to collision with pre-existing transmissions). It may be possible to further refine the collision behaviour by considering relative signal strengths (since a sufficiently weak reception when colliding against strong reception may not garble it), however WSNSIM currently does not simulate such behaviour. If two receptions overlap in time, they are simply taken as collided, irrespective of signal strength. Following is the algorithm of collision detection in pseudo-code form

ALGORITHM COLLISION DETECTION

Data: A new transmission of duration δt starting at time t and position \mathbf{x} and with range r

Data: Existing on-going transmissions starting at positions \mathbf{x}_i and range r_i

Data: Receiving node positions \mathbf{z}_k

foreach *receiving node n with position \mathbf{z}_k* **do**

if \mathbf{z}_k *is within the overlap region of ranges r and any of r_i for $i = 0, 1, \dots$,* **then**

 Mark the new transmission reception for the node at \mathbf{z}_k as collided;

 Mark the pre-existing on-going transmissions for node at \mathbf{z}_k as collided.;

end

end

Algorithm 1: Collision detection algorithm used in WSNSIM

The `transmit_end` represents the end of a transmission and its occur method has many accounting tasks. Firstly, it deregisters the node from *currently transmitting* list, and changes its status from transmitting to non-transmitting (i.e. idle). It also ensures that all the nodes within

the range of the transmission, for which there were no collisions, gets the packet in their packet buffer. For a receiving node, a collision is defined by two or more transmissions received by it simultaneously.

The *occur* method should also take care of accumulating the transmission energy expenditure of the transmitting node, and the reception energy expenditure of the nodes that were within the range of the transmission.

The **listen_start** event marks the beginning of passive listening for a node. The **listen_end** method marks the end of passive listening mode. When a sleep is scheduled, a **listen_end** event is scheduled followed by a **listen_start** (i.e. a timed wake-up). These events serve to track the passive listening energy expenditure, and can also reveal, for example, if the sleep schedule made a node miss packets unexpectedly.

The aforementioned class hierarchy shows events named after a protocol (e.g. **leach_self_election**, **cluster_member_registration** etc.) however, as the program's architecture evolved, it was no longer necessary to define protocol-specific event types. This is facilitated by a category of events broadly called *lambda events*. These are events that take a so called *lambda expression* as input. In C++ (and in many other programming languages), a lambda is an anonymous function object that can be scheduled for later execution. A relatively new C++ language specification (known as C++0x or C++11) published in June 2011 introduced this language feature that offers a convenient notation for network simulation. With the lambda notation one can embed blocks of code within a body of code, so as to treat the embedded block as a function object, which in turn can be scheduled for later execution. To illustrate the use of lambda, the following line within a function can now ensure that the block of code passed as the final argument is treated like an argument, which can be scheduled to be executed periodically along with other simulation events.

```
node_do_periodic(node, start_time, period, {block_of_code});
```

Before the c++0x specification, the 4th argument (or parameter) passed to **node_do_periodic** could be a function pointer or an object (e.g. - an instance of a subclass of **general_event** on which a virtual method would be called by the framework. With C++0x, the code object can be embedded, thereby significantly increasing the clarity.

One might argue that when the same code needs to be repeated due to embedded lambdas, then it reduces clarity, but the counter-argument to that position is that repeating blocks of code would indicate a misuse. One can always replace a lambda with a named function when a behaviour is reused in two places. That idiom is supported naturally by the framework of callable function objects.

Similarly to the above periodic action, a packet reception handler may be specified using embedded action syntax as follows:

```
node_on_message_rcv(node, msg_filter, {block_of_code});
```

Similarly, **msg_filter** would be yet another lambda expression whose job would be to specify the criterion under which the message would be handled by this particular handler.

One could argue that the filter could have been a conditional (if-then-else) statement within the handler block, but having a separate place for the filter criterion brings more clarity to the protocol specification.

Thus, using the aforementioned generic primitives, it would be possible to replace the otherwise hard-coded event behaviour `leach_self_election` with the following periodic action *lambda*:

```
node_do_periodic(node, 0.0, leach_cycle_length, {

// Code to perform probabilistic self-election

// Transmit advertisement frame

});
```

The principle of layer separation will be as follows:

The MAC and PHY services (e.g. `carrier_sense_and_transmit`) are encapsulated as functions callable at routing layer. The MAC and PHY phenomena (e.g. radio, energy) is hidden underneath the implementation of the MAC and PHY services. The routing layer is the target layer where protocols are being designed, so this layer is implemented in terms of nodal *programs*. These nodal programs are bodies of code meant to represent the behaviour of nodes in terms of transmitting, receiving, and forwarding packets.

2.5 Energy Depletion Model

WSNSIM considers four modes of energy depletion :

1. Node transmitting data.
2. Node actively receiving data.
3. Node listening but not actively receiving data.
4. Node in non-listening sleep mode.

The first two are the dominant modes of energy expenditure. The transmission expenditure is a function of the amount of transmission (expressed in terms of the number of bits transferred), and the range of transmission (expressed as a distance beyond which attenuation makes the transmission signal undetectable). The reception expenditure on the other hand is a function of the number of bits received. The exact equations describing the expenditure would depend on the electrical design of the physical node and thus would vary between node architectures, but it seems reasonable to adopt the model presented in a heavily cited work [72]. WSNSIM adopted the expenditure model presented in [72], the key equations of which are as follows:

$$\text{Transmission Energy Expenditure} = E_{tx} * k + \epsilon_{amp} * k * d^2$$

$$\text{Reception Energy Expenditure} = E_{rx} * k$$

Where k is the number of bits transferred, d is the transmission range, and $E_{tx}, E_{rx}, \epsilon_{amp}$ are coefficients capturing power consumption characteristics of the node. As in [72], the values of these coefficients used by WSNSIM are as follows:

$$E_{tx} = E_{rx} = 50nJ/bit$$

$$\epsilon_{amp} = 100pJ/bit/m^2$$

The expenditure model for idle listening (in which the node is listening for data but there is no data in the channel) is taken as the following.

$$\text{Idle Listening Energy Expenditure} = E_{idle} * \Delta t$$

Where Δt is the time spent in the said mode, and E_{idle} is a device-specific coefficient, set as 5 $\mu J/s$ by default. If the Mica mote with a reception power of 1mW is taken as the model for radio power, it has to be converted into the E_{rx} specification described above in conjunction with the data rate. E_{rx} , data rate, and receive power are related by the following equation.

$$\text{Receive power in W} = (\text{Receive data rate in bit/s}) * (E_{rx} \text{ in Joules/bit})$$

By this account, a specified receive power of 1 mW in conjunction with a data rate of 20 kbits/s corresponds to $E_{rx} = 50$ nJ/bit. The aforementioned coefficients can be modified for a simulation scenario in WSNSIM if the intention is to compare two node designs with different radio power characteristics, or if the intention is to evaluate a single node architecture under different radio power settings.

2.6 Statistical Model of Spatial Distribution

The spatial distribution of nodes has a strong influence on the protocol performance and thus an accurate model of the spatial distribution would improve the accuracy of protocol assessment. Researchers usually assume a uniformly random distribution of nodes in a rectangular region connected like a torus [192].

Here a more rigorous approach has been taken for modelling the spatial spread of nodes. In this approach statistical models have been produced from cattle herds recorded in satellite images (from Google maps) and from an experiment conducted by Strathclyde researchers in which cows were tracked using geospatial positioning. A statistical probability distribution was fitted to the recorded distance of each cow from its four nearest neighbors in the herd. The best fit distribution was found to be the Gamma distribution. Normal, Gamma, and Weibull distributions were fitted using a maximum-likelihood estimator available in the statistical package R, and the distribution with lowest fit-error was chosen. Following is the density function for the gamma distribution:

$$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

(Here α, β are parameters defining the distribution, and in the normalizing denominator is the Gamma function). For the estimated best fit, the Gamma function parameters are: $\alpha=2.208$ (the shape parameter) and $\beta=0.231$ (the spread parameter). Figure 2.4 shows a plot of the observed PDF alongside the density function of the best-fit gamma distribution. A structural

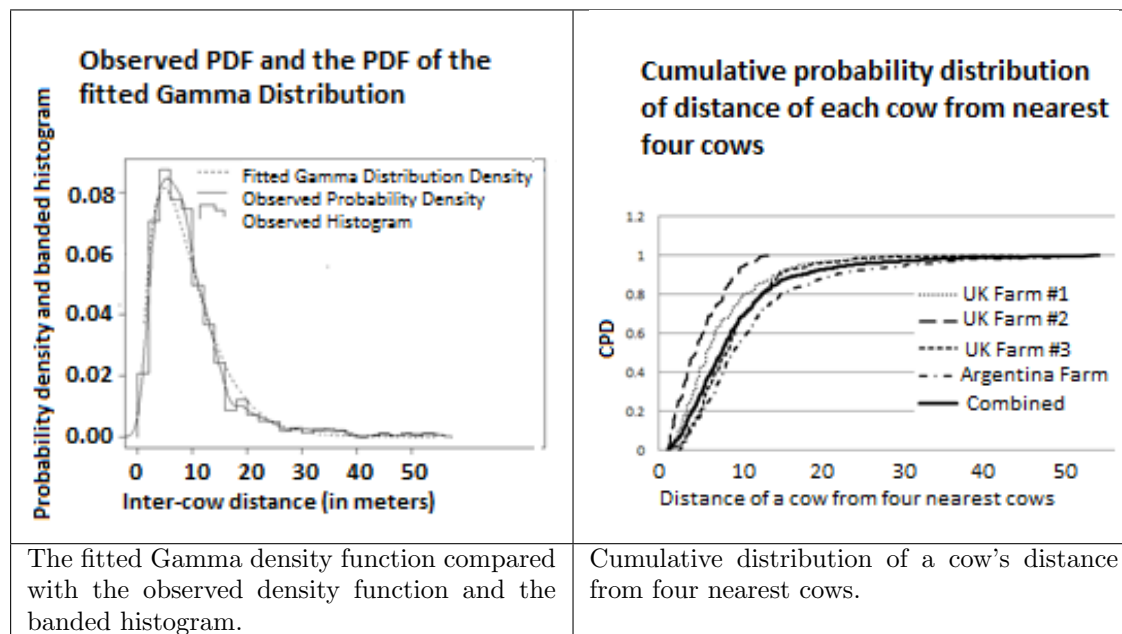


FIGURE 2.4: Probability density functions describing the composition of cattle herds

recursion based growth algorithm was used for generating synthetic herds by sampling deviates from the fitted distribution. An intuitive view of that algorithm is to view the herd growth as a crystal growth scenario with atoms that can form up to four bonds with other atoms, but unlike atomic bonds, the cattle are not uniformly separated in angles. The 360° angle around a cow is divided into four 90° sectors; the herd is grown as new cows join the neighbourhood of an existing cow in one of the unfilled 90° sectors at a distance sampled from the aforementioned fitted distribution. Following is an outline of the said algorithm:

- Step 1. Start with one or more initial cows (seed cows).
- Step 2. Choose a cow at random from the current herd which still has an empty neighbourhood sector.
- Step 3. Sample a distance from the said gamma distribution.
- Step 4. Place a new cow at that distance in the unoccupied sector at a random angle between zero and 90 degrees within the sector and mark that sector of the chosen existing cow as occupied. Reject the new placement if it goes beyond the farm boundary.
- Step 5. If the total required herd size is not reached, go to step 2, else the work is done.

2.7 Statistical Model of Cattle Mobility

The satellite images are static snapshots of herds, and they serve to suggest and corroborate models of spatial spread and formations occurring in herds, but they do not reveal anything about the patterns of mobility. In order to discover and study the patterns of mobility, timed position data have been recorded using GPS devices mounted on cattle collars. The movement behaviour of cattle was modelled as a finite state machine with states defined in terms of speed and direction. Within each state the speed or direction follows a continuous probability distribution fitted from experimental data [170]. The speed was partitioned into three states. The speed states were named as *resting*, *grazing*, and *shifting* - and experimental speed values were snapped to each of the three cluster centres formed by k-means clustering in speed values. The three states were chosen based on subjective observation of the herd and on expert opinion. Resting is the phase in which the cow is completely immobile; grazing is the state in which the cow is very slowly mobile and in the process of feeding from the grass patches. Shifting is the state in which the cow moves from one place to another in a relative haste.

The mobility model is implemented as a Markov process (i.e. a stochastic process in which the probabilistic transitions are dependent on the current state). The states and the transition probabilities are deduced by statistical analysis of GPS traces. The traces comes in the form of time series in node positions, the time series in the raw dataset is not uniformly spaced (i.e. time intervals between consecutive GPS fixes is not of a fixed duration). Thus, in order to derive simulation model from such raw datasets, the following workflow is used:

Motions Time Series → [An interpolator]→[Evenly spaced motions time series] → Speeds and directions → Clustering → Maximum likelihood fitting of statistical distributions.

As stated earlier, the speed distribution was modelled to cluster around three *modes* or *states* in the space of speed. There are statistical measures of the quality of a cluster partition. One such measure is called the *Mean Index of Adequacy* (MIA). The lower the MIA value the more compact (and hence of better quality) is the clustering.

MIA was calculated based on the speeds computed from GPS datasets for a range of values for number of modes or clusters (denoted by the symbol k). Figure 2.5 presents MIA values for various values of k . Although $k = 2$ produced a slightly lower MIA than $k = 3$, that was probably because grazing speeds formed a smooth continuum of speeds all the way to the shifting speed. The value $k = 3$ is chosen based on subjective perception of the three modes mentioned earlier - i.e. shifting, grazing, and resting.

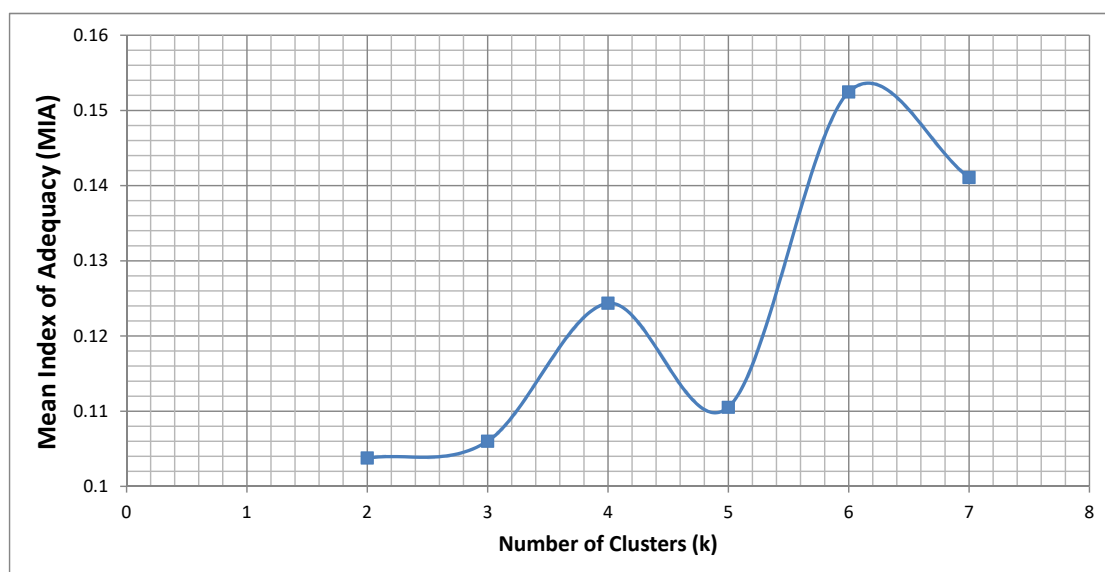


FIGURE 2.5: Number of speed modes or clusters plotted against MIA

Statistical distributions were fitted to speeds within each cluster. Fig. 2.6 shows the speed distribution in the resting state (the lowest speed cluster). A piecewise linear equation was used to approximate the inverse function of its CDF (as shown in light gray).

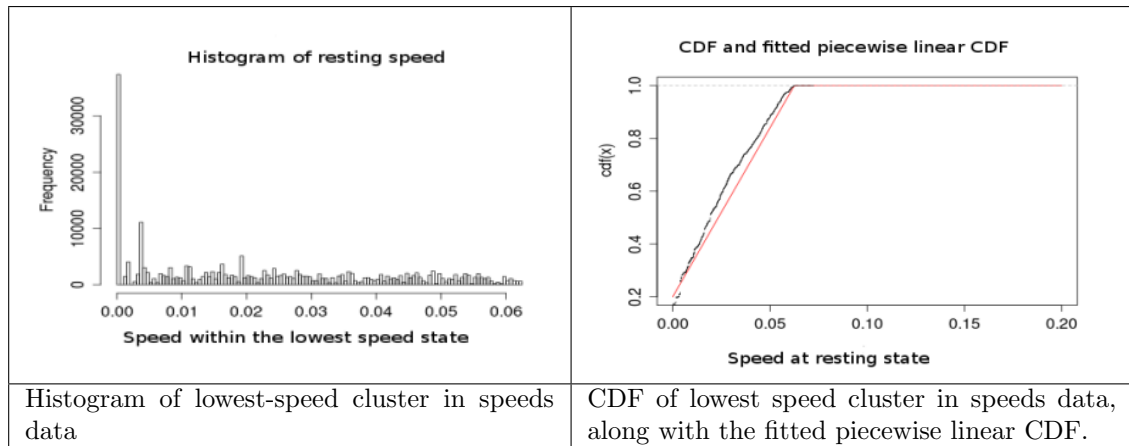


FIGURE 2.6: Speed distribution in the resting state

The speeds in the two other states are modeled as shifted Gamma distributions. The density plots with statistical estimates and the best-fit curve are given in Fig. 2.7. Another important aspect of the model is the amount of time spent in each speed state. The speed at each trace was computed using finite difference and speed state categorization was assigned to each position. This would show long runs of each speed state. For a given cow the time until transition to a different speed state is taken to be the time spent in the current speed-state. Exponential distributions were found to be good fits for these variables. Fig. 2.8 shows plots of the dataset and of the best-fit exponential distributions. State transition probabilities are computed as statistical conditional probabilities derived from the transitions data (the values are shown in the bottom right panel of Fig.2.8).

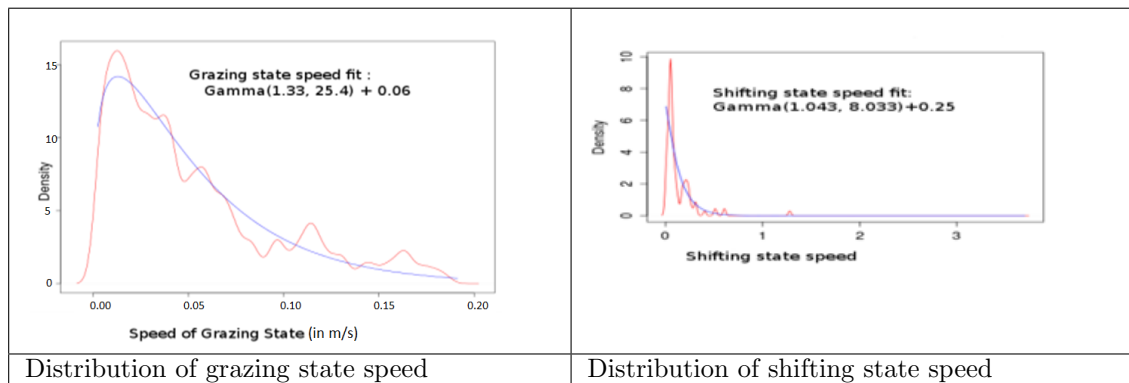


FIGURE 2.7: Speed distribution in the grazing and shifting states

2.8 Statistical Model for Heading Direction

Speed and the direction transitions are modelled from the GPS tracked herd data using the state transition probabilities. For this, the task was to derive a statistical model of directional

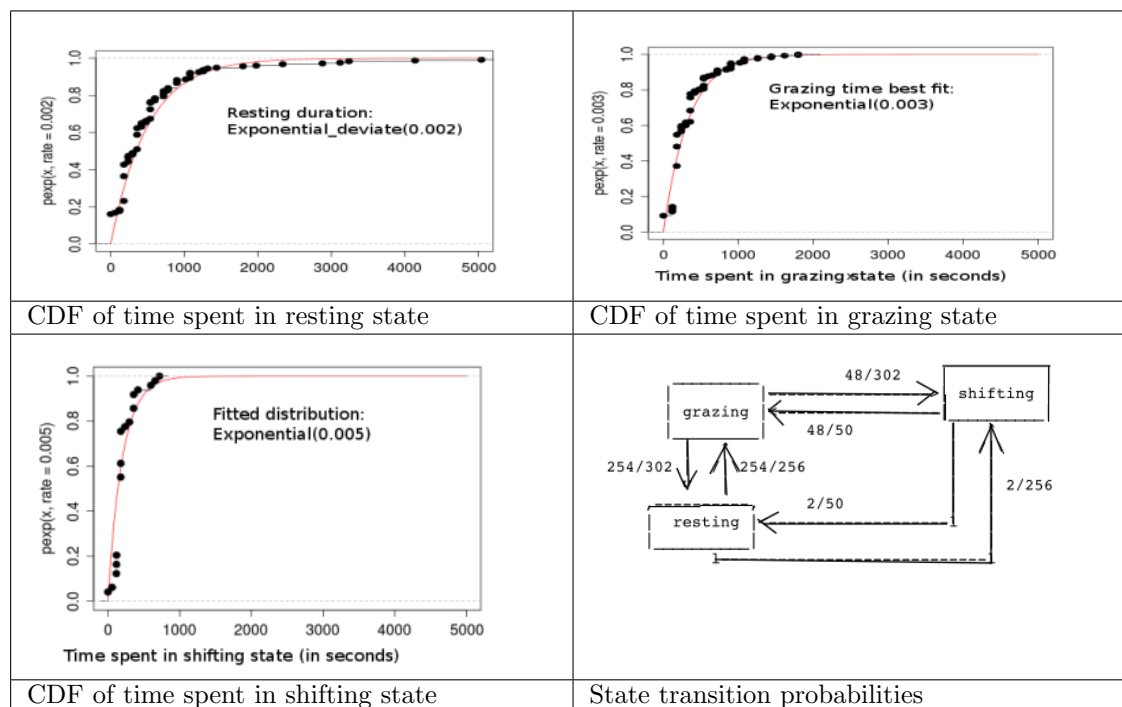


FIGURE 2.8: CDFs and state transition probabilities

changes in cattle motion. There are two aspects to this. Firstly there has to be a model for initial distribution of heading directions, and then there has to be a model for when and how the headings change. The initial heading directions were classified into 8 absolute directions as shown in the following diagram:

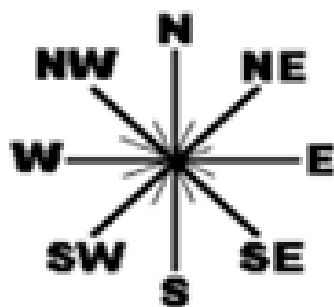


FIGURE 2.9: Heading Direction

Heading directions were lumped into the eight directional states (viz. East, North-East (i.e. 45 degree north of east), North, North-West, West, South-West, South, and South-East) and the transition probabilities were obtained from the GPS tracked dataset. Fig. 2.10 shows the probabilities of direction transition, in which the arrow direction represents the direction of the state transition, with the probability of the transition noted alongside the arrow. The relative frequencies of these 8 directions, as noted from the GPS tracked experiment were as follows. The overwhelming majority of west heading are probably explained by the fact that the readings were taken during an afternoon and that the cows probably have an affinity towards the sun (i.e. they seem to like to face towards the sunny direction). WEST: 546, EAST: 119, NORTH:

66, SOUTH: 92, SOUTH EAST: 64, NORTH EAST: 36, SOUTH WEST: 91, NORTH WEST: 66 The aforementioned directions samples were directly taken from the experimental dataset. Next a denser set of directions was sampled from the interpolated motion dataset, and noted the changes in direction to find the conditional probabilities of directional transition. The transition probabilities were calculated based on the GPS dataset used in [100] with a simple conditional probability calculation. The probability of transition between to state S_2 given that current state is S_1 is calculated as:

$$P(S_2|S_1) = \frac{N(S_1, S_2)}{\sum_k N(S_1, S_k)}$$

where $N(S_i, S_j)$ stands for the number of times the GPS dataset had a transition from S_i to S_j .

Based on the interpolated data, following are some of the conditional probabilities for change of directions. In the following pictures the direction of the arrow represents the direction of the state transition, with the probability of the transition noted alongside the arrow.

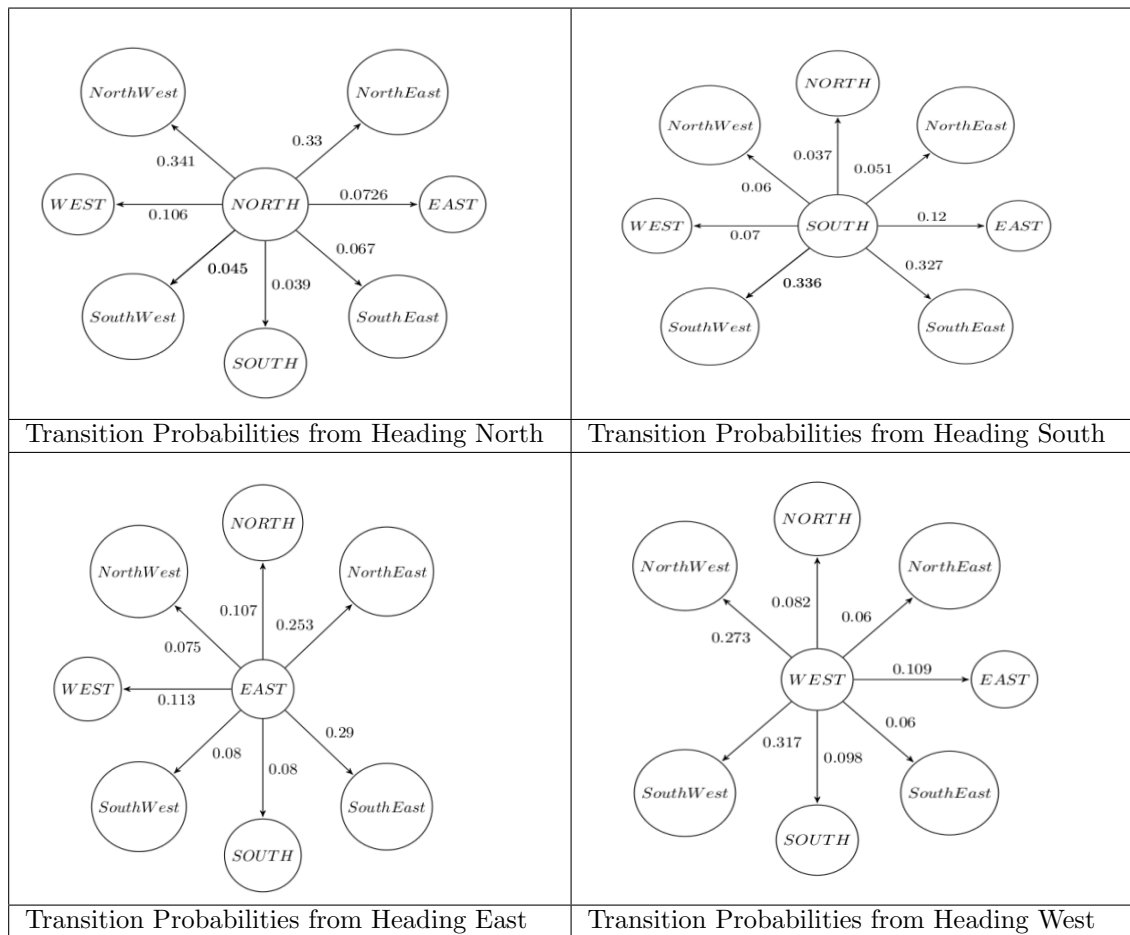


FIGURE 2.10: Transition probabilities of some heading directions

2.9 Modelling of Spatial Regions

Motion models in most simulators use a rectangular arena in which the nodes are free to roam about using various algorithms/models for movement. In some of these the rectangle is assumed to have topological seam connecting two opposite sides (so that the surface is connected to itself like a torus). However real enclosures come in many more shapes and also there are enclosed internal regions that are not accessible to the cattle, e.g. a fenced area, a pond, buildings, etc. WSNSIM has an improved simulation model in which the enclosure can be of any polygonal shape and can have any number of internal inaccessible regions. The geometric shapes defined for the simulation can represent more than just obstacles and fences. For example, one can make the shape annotations represent regions of high affinity for nodes and paths for data collector, or range of directional antenna. The orientation of the polygon is used to represent whether it bounds its inside or its outside. The orientation is defined by the order of the points. If the points are given in anti-clockwise order then it bounds the inside and if given in clockwise order it bounds outside.

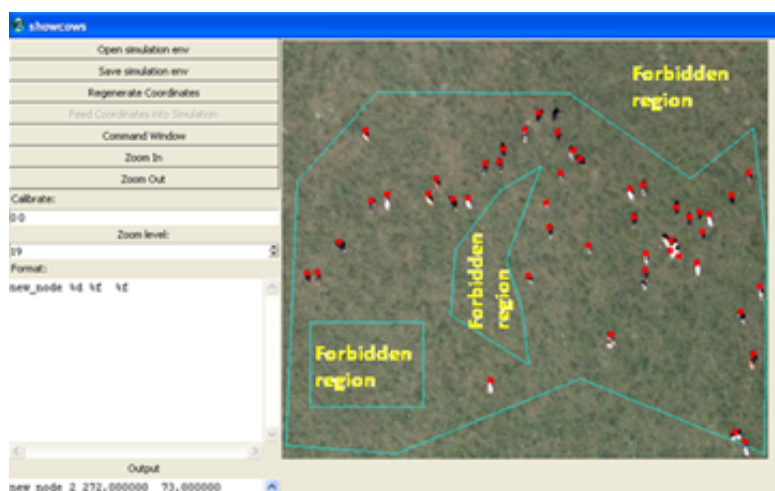


FIGURE 2.11: Modelling Spatial Regions. Satellite image © 2011 Google, DigitalGlobe

2.10 Directional Antennae

Previous investigations [170] has shown directional radio propagation properties for cow-collar transmitters. The antenna lies on one side of the cow's neck, which casts an electromagnetic shadow on the other side of the cow. Moreover, electromagnetic waves are primarily dipole radiations, so the wave propagation consists of lobes. In the presence of such directionality of transmitters, the simulator must take into account the shape of the directional range, in order to be accurate about receptions and interference. The facility for polygonal definition of antenna ranges has been implemented in addition to the default omnidirectional ranges. For example, the following directional propagation lobes may be approximated by the polygonal approximation shown in the following diagram: The polygonal range is defined with reference to the position

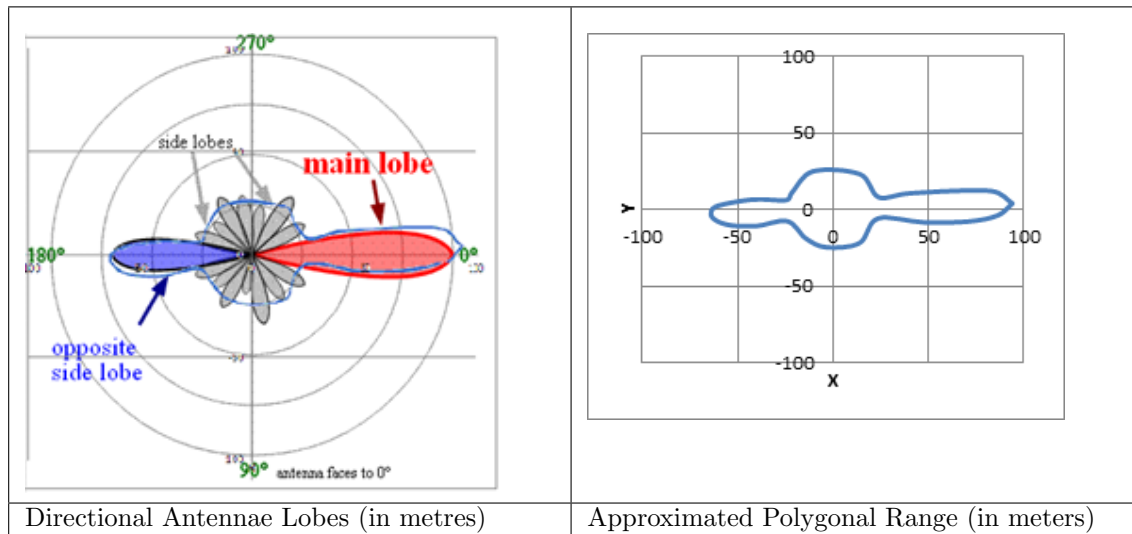


FIGURE 2.12: Directional Antennae

and heading of the node. A node falling within the range and with its receiver turned on will actually receive the transmitted packets. It is possible to write antenna ranges defined in terms of mathematical equations. To do that, one needs to derive a subclass of the region class and override the *contains* method according to the mathematical equation. For a polar equation $r = f(\theta)$ of the range, the condition for containment of a Cartesian point (x, y) in that range would be :

$$x^2 + y^2 < (f(\arctan(y/x)))^2 \quad (2.1)$$

The equation form of range representation is an alternative to the polygon form of representation. The computational aspect of range that matters to WSN simulation is the query that - whether a given receiver position is *inside* or *outside* the range. For polygon representation, the *inside-outside* test is carried out using a crossing-number algorithm, whereas the inequality in Eqn 2.1 is used for the function representation. Attenuation due to the bodies of neighbouring cows is not currently supported and it is expected that such influences are relatively minor for radio waves.

2.11 Novel Protocols designed using WSNSIM

Most WSN protocols are data centric, in which the network transmits sensor readings to data-sink nodes or base stations. Ideas have been borrowed here from such data centric protocols as LEACH [72] and STEM [157] and evaluated a protocol that addresses the shortcomings of either taken in isolation. LEACH has the shortcoming that there is nothing in the protocol to ensure spatial spread of cluster centers. As a result, often the distributions of cluster-heads get skewed leaving a large number of nodes un-reachable (and un-listened-to). The proposed modification of LEACH involves inference of spatial clusters through the first few communication rounds so that each node re-calibrates its self-election probability according to the estimated size of its local neighbourhood. The self-election probability is corrected to $1/(\text{size of local neighbourhood})$, so that small isolated sub-herds in space do not get frequently excommunicated due to spatial isolation. Figure 2.13 shows the results on evaluating this LEACH variant using WSNSIM. The number of excommunicated nodes reduces dramatically without compromising on the power consumption. In the result plotted below, the number of excommunicated nodes in LEACH reaches upto 45 whereas in Modified LEACH the maximum number is 4 (the other factors being constant - i.e. the mobility and the herd scenarios being the same between the two cases).

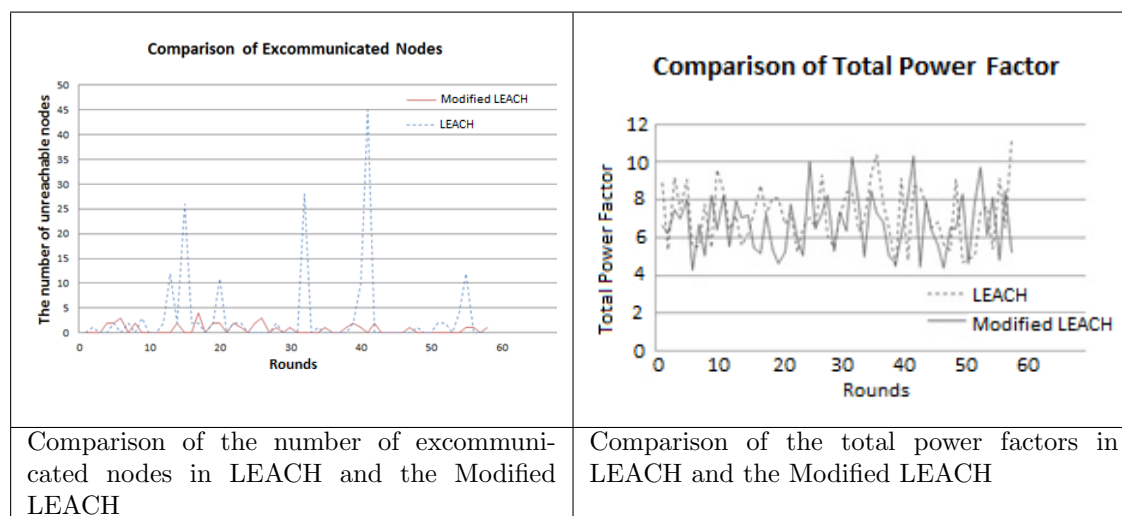


FIGURE 2.13: LEACH vs. Modified LEACH

2.12 Protocol Design

Papers on protocols like [71], [110], [136], [178], etc. were studied in order to gain some insight on the key ideas behind WSN protocol design. In LEACH [72] during the start of each round, cluster heads were selected in a centralized or distributed way which were rotated to ensure the energy dissipation across the sensors is balanced. Cluster head selection algorithm could be improved to reduce energy consumptions which can increase the lifetime of the sensor network.

2.12.1 Modified LEACH

Modified LEACH is a change to the LEACH protocol that takes advantage of the knowledge of spatial clustering of the nodes to modify the threshold function used for Cluster Head (CH) self-election. The key is to modify the threshold function such that the following is satisfied:

$$Threshold(n) = \max\left(\frac{1}{sc(n)}, Threshold_{LEACH}(n)\right)$$

Where, $Threshold_{LEACH}(n)$ is the self election threshold for the node as exactly as calculated in the LEACH protocol. Here $sc(n)$ is the number of nodes in the spatial cluster containing the node n . The knowledge of $sc(n)$ is not initially available but it becomes available as the network topology and the information on spatial neighbourhood emerges with the rounds of execution of the protocol. This change was inspired by the k-means routing protocol [196] and takes into account spatial distribution of nodes. Two measures of performance were used to assess WSNs: (a) power expenditure, (b) the average number of excommunicated nodes. When a non-cluster-head node lies outside the range of all cluster-head's advertisement transmissions, it will not be able to communicate its readings during that round - a non-desirable occurrence. Such nodes are termed as *excommunicated nodes* for that round. The measure of power is implemented based on [71]. This measure is an approximate metric representation of power efficiency obtained due to indirect routing of messages, compared with direct transmission by every node to the base station and is defined as the ratio between two transmission powers as follows:

$$PowerFactor = \frac{\text{Transmit power consumption with indirect routing}}{\text{Transmit power consumption with direct power to BS}}$$

This is a dimensionless quantity and both the numerator and the denominator may be approximated by adding the squares of the relevant transmission distances (based on a simple open space radio model). Different farms with varied sizes of herds and from different locations were chosen for experimenting and simulation. Results shown here are for a representative herd in Argentina with 166 cows, chosen to represent a fairly large farm.

2.13 Simulation of LEACH vs Modified LEACH

This section reports the results of a comparison made between LEACH and the proposed modified LEACH. The study was conducted with herd configurations observed in real satellite images of herds. The key protocol parameter for LEACH-like algorithms is the number of cluster-heads, for which the symbol k is used. Two different values of k were used viz. $k = 4$ and $k = 6$. The position of the base station (BS) was also varied across simulation scenarios. The results from these scenario combinations is summarized in table 2.1. The results show the expected trend that the clustering protocols offer greater relative advantage when the herd is further away from the base station. This is the expected result because when the base station is far away, the distance from the base station becomes the dominant distance (i.e. much greater than the distances within the herd). So the relative efficiency derived from clustering protocol is greater. The power factor represents exactly this measure of performance (i.e. relative efficiency of the multi-hop protocol over a direct all-nodes-to-base-station protocol).

Figure 2.13 shows the relative power factors of modified LEACH versus LEACH varying over communication rounds assuming a far-away location of the base station to the east of the herd. The modified LEACH protocol is *modified* in that the cluster-heads are selected from pre-computed spatial clusters (i.e. the clusters being computed using a k -means clustering algorithm), and not randomly self-elected from all nodes. The simulation result shows that spatial clustering does help in reducing the power expenditure for periodic data gathering. The proposed modified LEACH outperformed classical LEACH in terms of power efficiency.

Tables 2.1 and 2.2 show the variation in number of excommunicated nodes per communication round for both protocols. The improved spatial spreading of cluster-heads in the modified protocol ensures that there are fewer nodes that fail to reach the base station. The plot in figure 2.14 shows a comparison between the number of cluster-heads between LEACH and the modified version. It demonstrates that the reduction in excommunicated nodes is achieved without any significant change in the number of cluster-heads.

	Base Station Coordinate	LEACH	Modified LEACH (K=6)
Average power reduction	At north-west corner	7.6197	6.94177
	Outside, far away (west)	7.1974	6.71775
	Inside, at the herd centre	20.249	18.8245
Excommunicated nodes		3.103	0.689

TABLE 2.1: Results of simulation for the large herd (N= 166)

	Base Station Coordinate	LEACH	Modified LEACH (K=4)
Average power reduction	Outside, far away (north)	14.707	14.502
	At north-west corner	13.102	13.104
	Inside, at the herd centre	30.425	29.963
Excommunicated nodes		2.724	2.05

TABLE 2.2: Results of simulation for the small herd (N= 58)

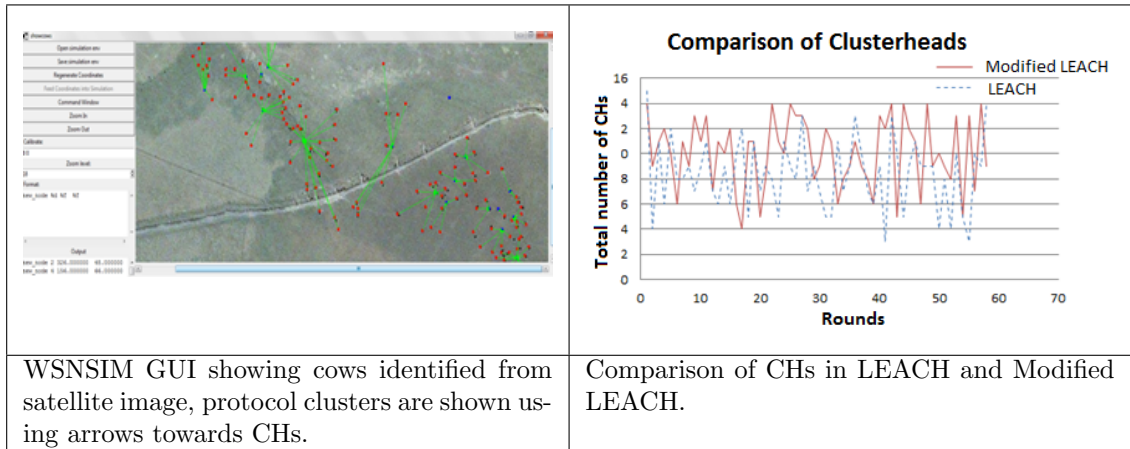


FIGURE 2.14: Number of Clusterheads in LEACH and Modified LEACH. Satellite image © 2011 Google, DigitalGlobe

2.14 A Novel Protocol Design - LEMSYP

In LEACH, the nodes wake-up from low-power state to high-power state based on clocked timers. The assumption here is that although there is clock drift, the differences between clock rates is not so high that the nodes would miss an entire round. This assumption is reasonable because the nodes get an opportunity to synchronize their clocks in each round of communication with the base station.

By the nature of the application (i.e. that of animal monitoring), it is adequate to get a snapshot of the herd condition every several minutes. So a low-energy STEM-like protocol was developed (and named *LEMSYP*) that uses an idle duty-cycle to monitor the channel for a wakeup signal. The wakeup signal is a train of small packets broadcast with high power from a non-power-constrained central transmitter. In this protocol the nodes will normally stay in a low duty-cycle (LDC) periodic-listening idle phase, from which nodes may be awakened by a train of beacons from the base station. The train is long enough to accommodate at least one listening phase of a duty cycle, and each node's subsequent wake-up time would be synchronized according to the serial number of the received wakeup packet. Thus LEMSYP manages to synchronize just fine despite having a much longer time between data collection rounds, thereby reducing the power consumption dramatically. Figure 2.15 shows the state diagram of the LEMSYP protocol.

For mobile assets monitoring, it is necessary to assume that there is no prior information about the topology of the network other than general statistical property of their spatial distribution. So the topology has to be inferred on the fly. Simultaneity and synchronicity of the node clocks can not be assumed because clocks on inexpensive microcontrollers are not accurately synchronized and so are skewed with respect to one another. So, one of the foremost functions of the protocol is to synchronize node clocks (otherwise subsequent timed operations will fail due to accumulated asynchrony). It is assumed in the protocol description that the base station (BS) is not battery constrained and may make long range transmissions to cover the entire area containing the WSN nodes. The unconstrained transmission from the base station (BS) covers

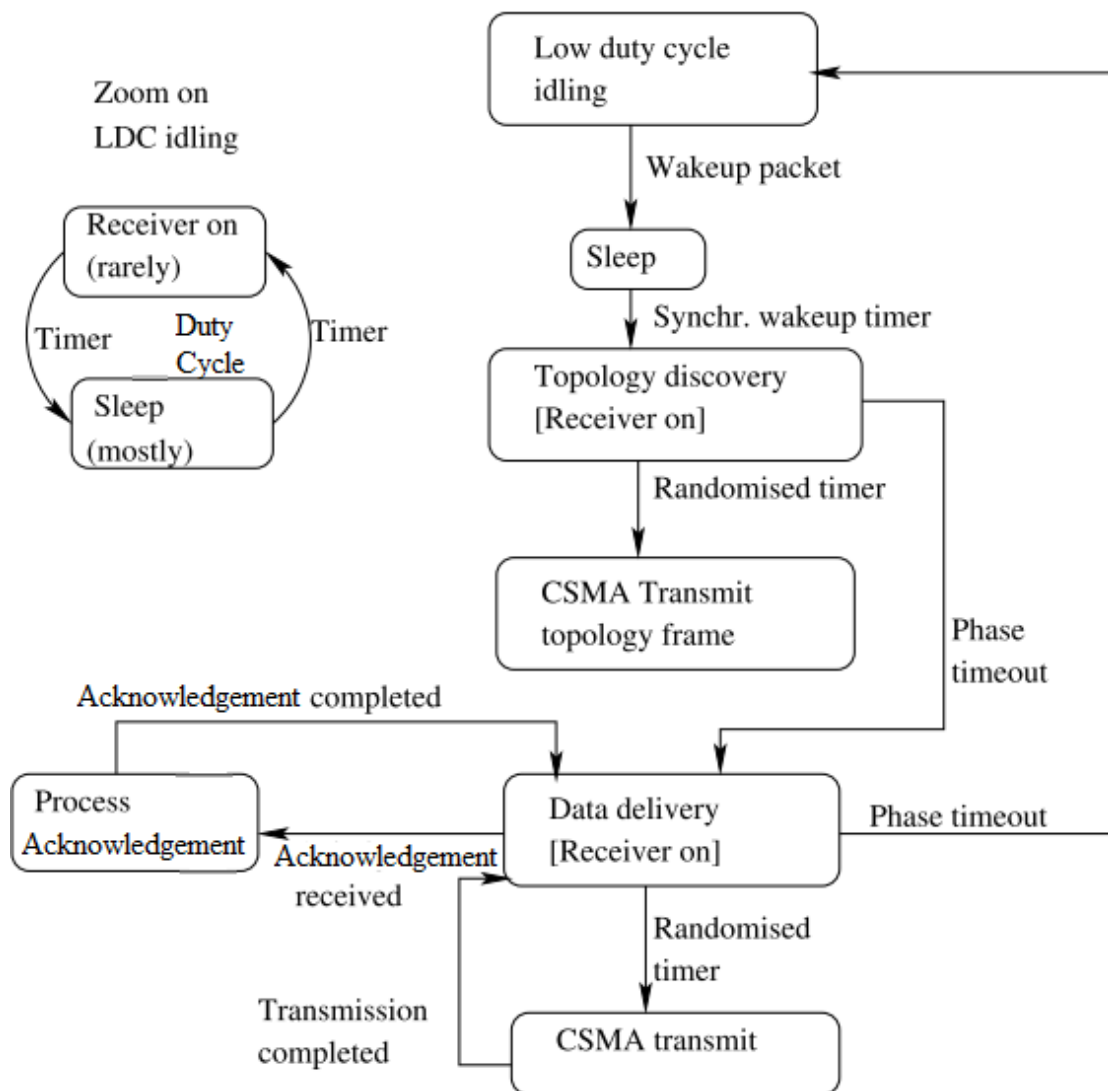


FIGURE 2.15: State diagram of the CSMA LEMSY protocol

only one direction of communication. The sensor data originates at the constrained nodes. So the direction of communication for the sensor data needs to use an energy efficient routing.

2.15 CSMA based version of LEMSY (Low Energy Multi-hop Synchronized Protocol)

In this protocol the nodes will normally have a small duty-cycle periodic listening phase, from which nodes may be awakened by a train of beacons from the base station. After such a synchronised wakeup, there would be a brief multi-hop transmission phase by nodes to pass the data to the base station. This protocol is quite low on energy expenditure. The protocol has five phases:

1. Idle duty cycle
2. Synchronized wakeup
3. Topology discovery
4. Sensor data delivery
5. Acknowledgement

The Idle-duty cycle phase:

The default state of the WSN nodes is a very low energy execution state that includes a very small duty cycle of periodically turning on the receiver circuit. In this duty cycle the time over which the receiver is on is much smaller than the off times. This periodic reception is meant for picking up possible wakeup packets sent by the BS. It is the BS's responsibility to wake up the network occasionally using a dense sequence of wakeup packets. The length of the sequence of the wakeup packets has to be sufficient to ensure that all of the nodes have received a wakeup packet during their short periodic reception span.

The synchronized wakeup phase:

The second phase starts for a node when it happens to receive a wakeup packet from the base station. The wakeup packets contain one very important piece of information - a time span till the scheduled wakeup. If the synchronised wakeup time is scheduled to be t_s and the time at which the n th wakeup packet is sent is t_n , then the wakeup packet specifies $t_n = t_s - t_n$ as the time till synchronised wakeup. The recipient node is then expected to make transition to a new state where it sleeps and schedules a one off timer to wake up after time t_n . So, by design every node wakes up almost simultaneously since the clock skew over the short span t_n is not very significant. Although the clocks of the nodes may have skewed substantially during the time before the wakeup, the drift is corrected every time a synchronised wakeup is coordinated by the base station.

The topology discovery phase:

The third phase of the protocol is that of topology discovery. Recall that in the second phase, the nodes have received a wakeup packet from the BS. So, the nodes have an estimate of their respective distances from the BS based on received signal strength index (RSSI) calculation. In the third phase, the nodes try to inform the other nodes within their range about their mutual

distance and their distance from the BS. So the nodes transmit short packets with a fixed transmit power and using CSMA at randomised times. If there are no collisions, then one transmit per node is enough to inform all neighbours but in the presence of collisions more transmit might be necessary. Discrete event simulation can inform about the number of transmissions necessary to make it extremely improbable that a node is not discovered by its neighbours. The topology discovery packets also contain residual battery energy of the transmitting nodes. At the end of this phase, each node gets to know its neighbourhood and can make routing decisions based on this information for subsequent communications.

The sensor data delivery phase:

In the fourth phase, the nodes actually transmit and forward sensor readings towards BS in multi-hop manner. The energy expended during this phase is strongly dependent on the amount of data that gets sent to the BS and a primary source of optimization in this phase is to not transmit what is not important to the application. This is not something that the protocol itself can handle and it may be achieved by having much of the intelligence in the nodes' processor. For example, sophisticated diagnostic algorithms can be implemented to run as embedded code in the node so that the decision to transmit is made only when a condition of interest is detected. However the protocol can achieve much efficiency through energy optimal routing. Hence the third phase of our protocol tries to route packets in a manner that tries to minimise energy expenditure.

After the fourth phase starts, a node that needs to send a detected reading or condition, broadcasts a data packet but mentions within the packet a preferred recipient. This choice of preferred recipient is made based on a heuristic algorithm working on the local topology information gathered by each node in phase three. The data delivery phase is multi-hop, so the nodes try to forward their message through other nodes that are well placed to make the forwarding fruitful and efficient. This is primarily done through a metric that serves as the probability of selecting a particular node as a forwarding neighbour. For each neighbour known during the topology discovery phase, a metric is calculated on the basis of their distances from the BS, their residual battery charge and their proximity from the node in question. This metric is almost zero if the neighbour is worse placed than the current node, and there are other neighbours that are better placed than the current node. However if there are no other nodes that are better placed than the current node, then the packet may be sent to one of the worst placed nodes as a *spread-back* packet with the hope that a better route may eventually be found or the node will eventually move closer to the base station or to other better placed nodes.

In this protocol variant, a node transmits using CSMA and at a randomised time in two circumstances:

1. It is the originator of the data, i.e., its processor has detected a condition that needs to be reported.
2. It is forwarding a message originated by another node.

When a node is an originator, depending on the severity of the condition it might generate two packets (or more) to different neighbours to make it less likely to get lost. When a node is closer to the BS than all its neighbours, then it would normally be the responsibility of this node to

transmit directly to the BS. In case that BS is outside its normal transmission range, it will hold on to the packet and spread it back to its neighbours with the additional information that it is a spread back packet so that there will be multiple nodes that would try to transmit these packets to the BS in a later cycle. The objective of the spread back process would be to ensure that there are several nodes relatively close to the BS that would try to pass the message to the BS in a later cycle.

The acknowledgement phase:

The final phase of the protocol is that of acknowledgement by the BS. Since the BS is not energy constrained, there is no need for a multi-hop propagation and the acknowledgement message is sent directly by BS using high strength transmission. Every message has a unique identifier that gets mentioned in the acknowledgement packet after the message is received by the BS. When the originator and spread back message bearers receive an acknowledgement for the packet, they internally make it as sent and make so further attempts to send the same message in a later cycle. Similarly if a message is not acknowledged, it is attempted to send in a subsequent cycle.

2.16 TDMA based variant of LEMSYP (T-LEMSYP)

Version 2: TDMA based variant of LEMSYP (T-LEMSYP) The carrier sensing capability is very useful for sharing a communication medium between multiple transmitters in that it allows a transmitter to know when the medium is already in use. The usual behavior/algorithm used by transmitters is to back off when the medium is found busy and an attempt the transmission later on. This approach however is only a heuristic and neither guarantees prevention of collision nor ensures efficient use of bandwidth. These deficiencies are illustrated by two well known problems :

1. Hidden node problem
2. Exposed node problem

The *hidden node* problem is that of an unavoidable collision. This happens when a node receives garbled message due to two transmitters within its range transmitting simultaneously but the two transmitters are sufficiently far apart to not detect each other's transmission. The *exposed node* problem refers to the situation in which a node A intends to transmit to another node B, but backs off because of carrier-sensing (i.e. detecting) a transmission from node C, whilst B's location is outside C's range. Unlike the hidden node problem, the exposed node problem does not garble data but wastes bandwidth. These problems can be remedied by a protocol that schedules transmissions in such a way that collisions don't happen. A schedule is made by dividing up the available time into slots for each potential transmitter. This approach is called time division multiple access (TDMA). After the synchronised wakeup phase of C-LEMSYP, the nodes are fairly well synchronised with respect to each other. This is an ideal situation for carrying out TDMA communication for the subsequent phases. One possibility is to have the base station transmit a schedule for the subsequent TDMA, however, that is not really necessary

because the nodes can work out a TDMA schedule based on their unique IDs. The nodes may be initialised with serial numbers $1, 2, \dots, n$. This initialisation can be a one off assignment that need not change until a major overhaul of nodes. These serial numbers can be used by the nodes to work out a TDMA slotting autonomously (e.g., the node with serial number j will schedule its topology discovery packet at time $j * \delta t$ from the synchronised wakeup time. The node receivers have to be kept on during the topology discovery phase because the relative distances between the nodes are still not known. However, during the data delivery phase, the nodes know which other nodes are within their range. So the nodes can keep their receivers on only during the TDMA slots of the nodes within their range. Thus based on the above idea, the T-LEMSYP is specified primarily as a TDMA based protocol during the topology discovery and the data delivery phases. The main challenge of TDMA based protocols is to keep the nodes synchronised as WSN node clocks will naturally drift away from each other over time due to manufacturing uncertainties in the crystals and the dependence of clock rate on voltage and temperature. This phenomenon is called clock skew. Therefore the T-LEMSYP uses centralised synchronization mechanism by which the nodes synchronise themselves. Further mutual synchronisation is possible during the data delivery phase if each node sends out its local time in the data packets.

The protocol phases of the TDMA based LEMSYP are described as follows (only the parts that are different from C-LEMSYP are mentioned here):

The topology discovery phase:

The third phase of the protocol is that of topology discovery. The nodes have their serial numbers $1, 2, \dots, n$. So they can autonomously compute their TDMA slots according to their serial numbers. All nodes keep their receivers on during this phase. The next question that arises is about the duration over which this phase is carried out. The necessary time is obviously determined by the total number of nodes and the nodes have to be informed of this number somehow. This information may be embedded in the firmware; however a more flexible approach would involve an initialisation packet from the base station to inform this number before the topology discovery phase.

The sensor data delivery phase:

In the fourth phase, the nodes transmit and forward sensor readings towards BS in multi-hop manner. Each node carries out its transmission within its TDMA slot. The TDMA slots are allocated in periodic cycles adding each cycle a node gets two delivery slots. The slots are allocated as follows: In the first half of the cycle, the node serial numbers are incremented by one for each slot, (i.e., it goes like - slot for node 1 followed by slot for node 2 followed by slot for node 3 and so on). In the second half of a periodic cycle, the serial numbers are counted down ($n, n - 1, n - 2 \dots$ so on). This is to ensure that a node does not need to skip a full cycle on forwarding a packet just because its TDMA slot happened to be before the sender's slot. By virtue of the topology discovery phase, each node knows as to which nodes are within its range, so the nodes can choose to tune in during the TDMA slots of only those nodes.

The acknowledgement phase:

The acknowledgement by BS is identical to that of C-LEMSYP, however since the data delivery phase is carried out by TDMA; it is possible to significantly increase the reliability of the multi-hop links by scheduling an acknowledgement packet from multi-hop recipients immediately after receipt of the packet.

Figure 2.16 shows the state diagram of the TDMA based variant of the LEMSYP protocol.

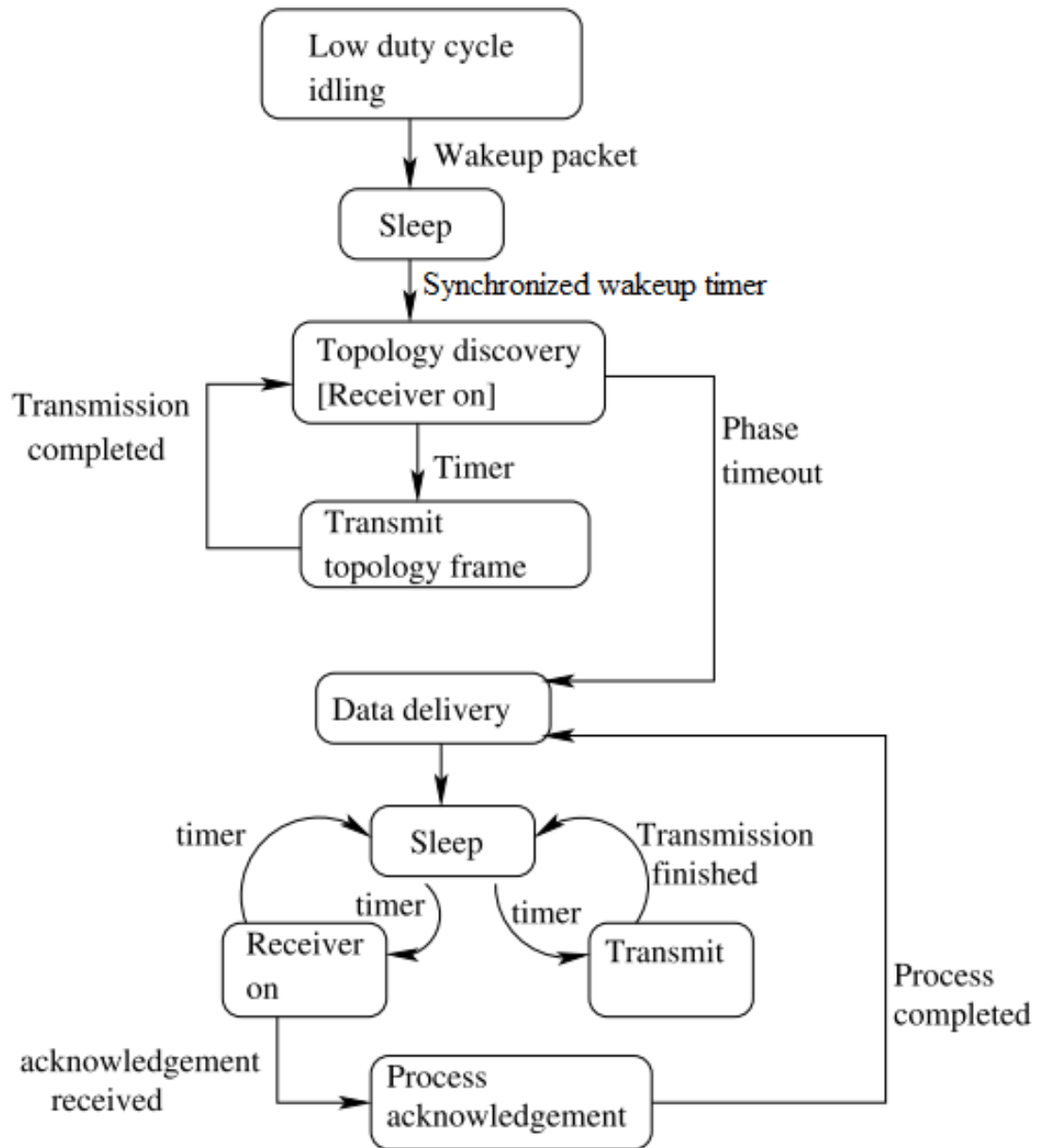


FIGURE 2.16: State diagram of the TDMA LEMSYP protocol

2.17 Energy Depletion Comparison between CSMA based LEMSYP and LEACH

Two different herd scenarios were modelled in WSNMIM for both LEACH and LEMSYP based data gathering. The results of these scenarios are presented in this section. The first herd was obtained by algorithmic cattle detection from satellite image. The second herd was obtained from a measurement exercise where cattle positions were recorded using GPS receivers mounted on collar bands.

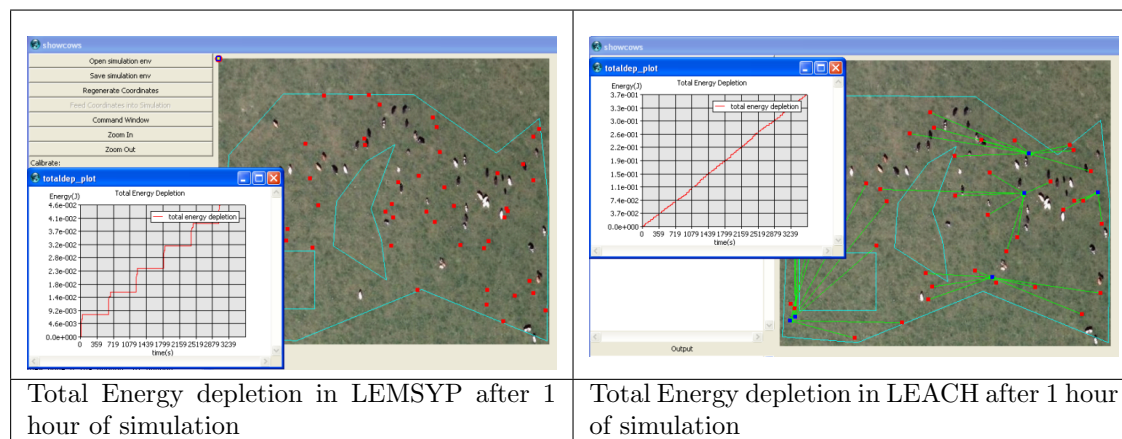


FIGURE 2.17: Total Energy Consumption in LEACH and LEMSYP after one hour of simulation. Satellite image © 2011 Google, DigitalGlobe

The results of simulation comparing the power performance of LEACH and LEMSYP are shown in figure 2.17. This scenario was based on a herd captured from a satellite image. Figure 2.18 shows plots of energy depletion for LEACH and LEMSYP for the first herd (i.e. the one obtained from satellite image). These results were recorded after one hour of simulation.

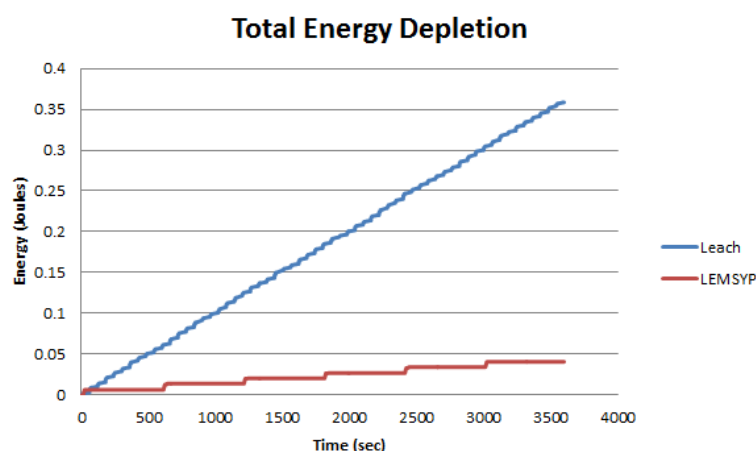


FIGURE 2.18: Comparison of energy consumption by LEACH and LEMSYP

Modelled and tested using WSNSIM the farm where the cow-collar experiment was carried out, using samples of initial distribution obtained from the dataset. This was using the mobility model [153]. Energy depletion comparison between LEACH and LEMSY for the second herd

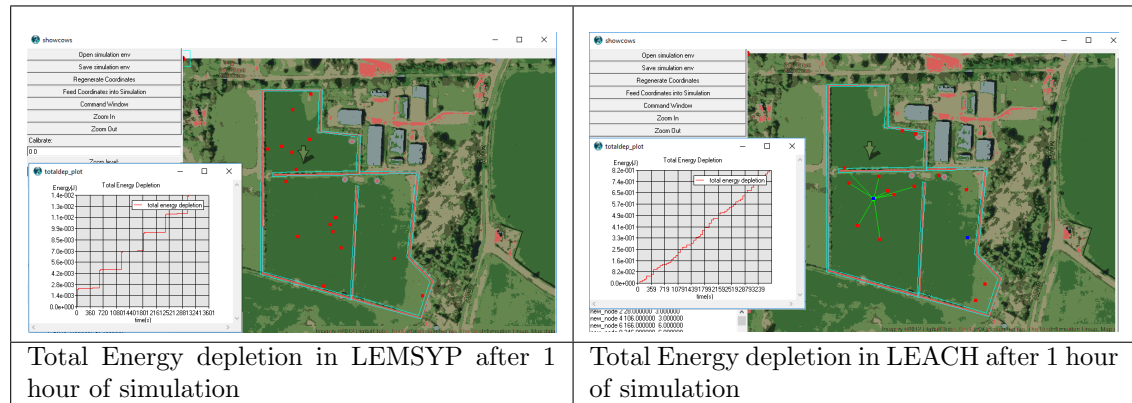


FIGURE 2.19: Total Energy Consumption in LEACH and LEMSY after one hour of simulation - using data from previous work [170]. Satellite image © 2011 Google, DigitalGlobe

(i.e. the one recorded from the GPS based measurement exercise) is shown in figure 2.20. These results were also recorded after one hour of simulation.

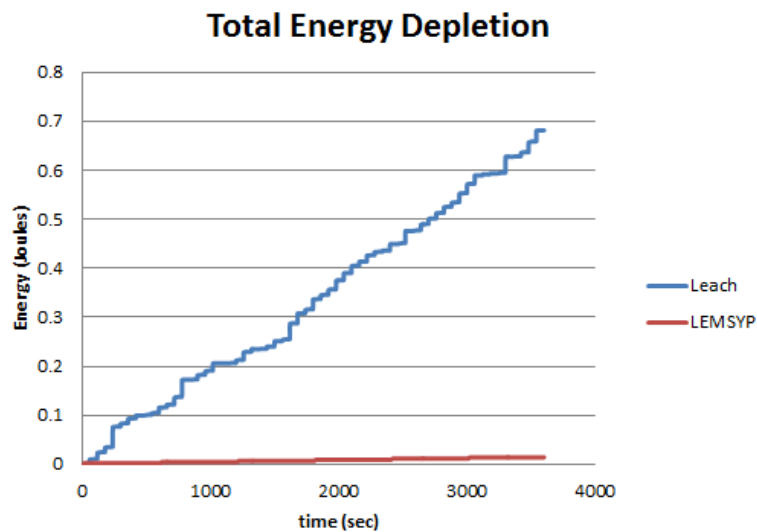


FIGURE 2.20: Comparison of energy consumption by LEACH and LEMSY

2.18 Energy Depletion Comparison between CSMA vs TDMA based LEMSYP

The TDMA based variant of LEMSYP primarily differs from the CSMA based variant in that it designates certain time slots for message transmission. The time slots may be pre-programmed (e.g. computed based on the node ID) or a schedule may be communicated by the base station. WSNSIM provides both these options. A comparison study was carried out between the two variants. The CSMA variant has the advantage that it doesn't involve any prior arrangement necessary for creating time division schedules, and that it can scale fairly unboundedly without imposing constraint on the number of nodes. However despite carrier sensing, CSMA can run into collisions due to the hidden node problem described earlier. On running the CSMA LEMSYP simulation on a typical herd scenario for several hours, an average collision rate of 1.6 per round was noted. TDMA does not suffer from this problem because collisions are excluded by scheduling, but then TDMA requires more careful configuration according to the network size. TDMA has the additional advantage in terms of low power consumption because the nodes can switch off receiver circuits except during designated TDMA slots. Figures 2.21 and 2.22 present the power consumption comparison between CSMA and TDMA variants for two randomly selected nodes. Figure 2.23 presents a similar comparison for the full network.

This shows that T-LEMSYP is superior to C-LEMSYP both in terms of collision avoidance and power consumption.

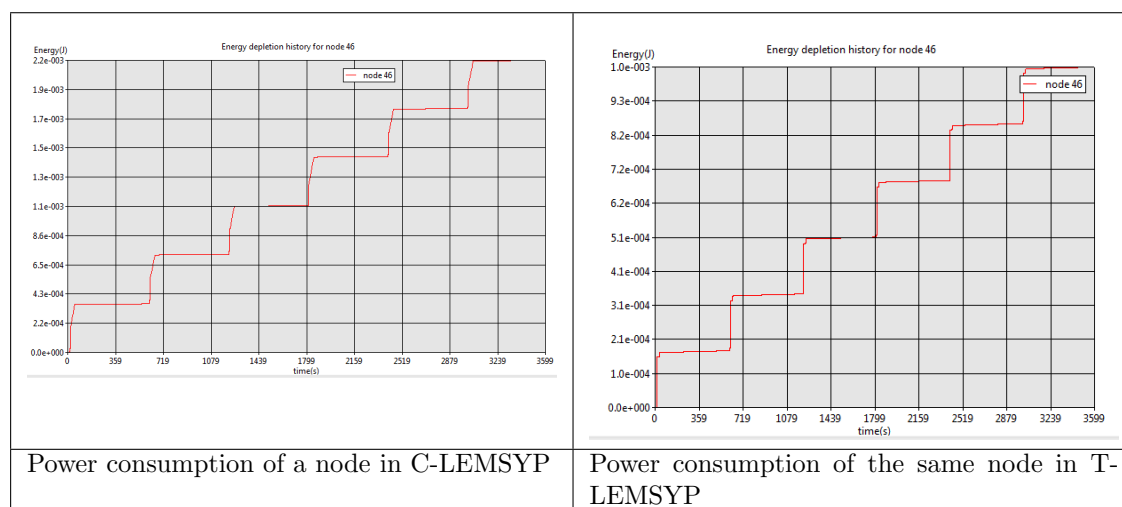


FIGURE 2.21: Comparison of power consumption of node 46

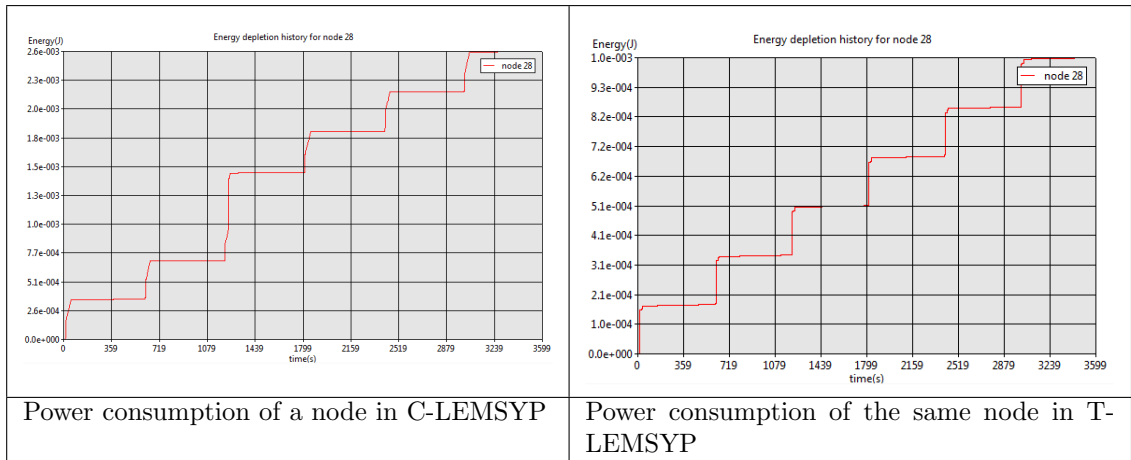


FIGURE 2.22: Comparison of power consumption of node 28

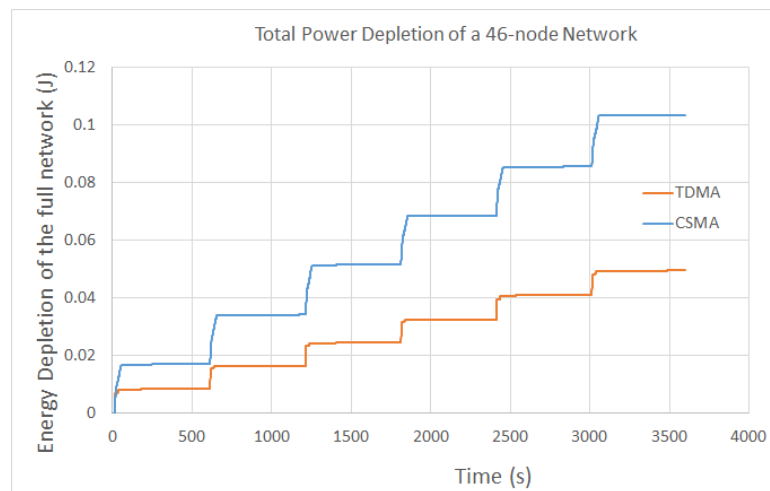


FIGURE 2.23: Comparison of total (i.e. network-wide) power depletion between T-LEMSYP and C-LEMSYP

2.19 Verification and Validation

The WSNSIM simulator was verified using a set of tests representing well understood and simple scenarios for which certain aspects of the results are known by analytical calculation. To facilitate this, R [142] bindings were written for the elementary units of WSNSIM functionality. The unit-tests are Tcl and R scripts that can be executed and report a failure if the understood outcome is not met. Some tests make randomized and periodic transmissions from static nodes for which the packet collision rate can be estimated without simulation. There are some tests for which the output is a stream of random variates that are fed into R as a data-vector and its distribution fitted and compared against the expected distribution parameter. For example, there is a test script that generates herd positions using the structural recursion algorithm described earlier in this chapter. The generated herd positions are taken and the distances between each generated node and its four nearest neighbors recorded. This data is then fed into R to verify that the distribution agrees with the parameters obtained from satellite images.

Round trip verification of the synthetic herd model was done as follows:

- WSNSIM generates synthetic herd.
- Record the inter-cow distances and fit distribution on R's `fitdistr` function.
- Check if the fitted distribution parameter matches with the distribution found from observed herd data (as described in figure 2.24).

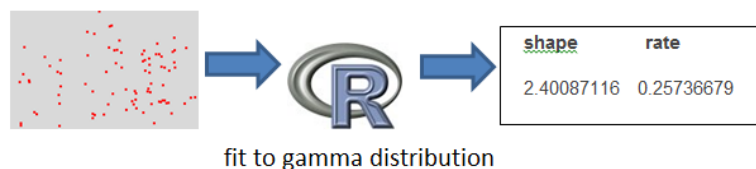


FIGURE 2.24: Synthetic herd round trip test

Round trip verification of statistical models was done as follows:

- WSNSIM generates random deviates from its C++ code.
- Feed these deviates into R's distribution fit function and check if the fitted parameter matches with the deviate generator's parameters.
- To do this, the WSNSIM was built as a DLL and integrated into the R process.
- Following is an example of how calls are made into WSNSIM (shown in RED) from R:

```
ex <-Expondev(0.2, 1)
expdev <- NULL
x <- 1:1000;
for (i in seq(along=x)) {
```

```
expdev <- c(expdev, Expondev_dev(ex))
}
fitdistr(expdev, "exponential")
```

2.19.1 Verification of mobility behaviour with more data

The initial mobility model was based on a subset of a larger set of mobility data acquired continuously from a week long measurement exercise involving more than a dozen cows. This larger dataset is presented in appendix A. The said sub-set was the dataset selected for a previous study [170]. This subset data-set had 1400 positions recorded on a single day. This dataset was already pre-processed and cleaned up for the purposes of a previous publication. At that time the full dataset seemed to be missing. It was later on that the full dataset could be retrieved. This was a much larger set containing 1.1 million GPS tracked positions. At a first glance the raw form of the full-dataset seemed corrupted. Often the position changed by several kilo-meters within a second. A particular snapshot of this data is shown in figure 2.26. It turned out to be the systemic noise of the GPS reception process. The said noise could be removed using a Butterworth filter. Figure 2.25 shows a plot of GPS fixes recorded for one of the cows over 7 days.

The full dataset comprising all GPS fixes is presented in appendix A. The spatial distribution from the larger dataset was found to be in agreement with the initial model.

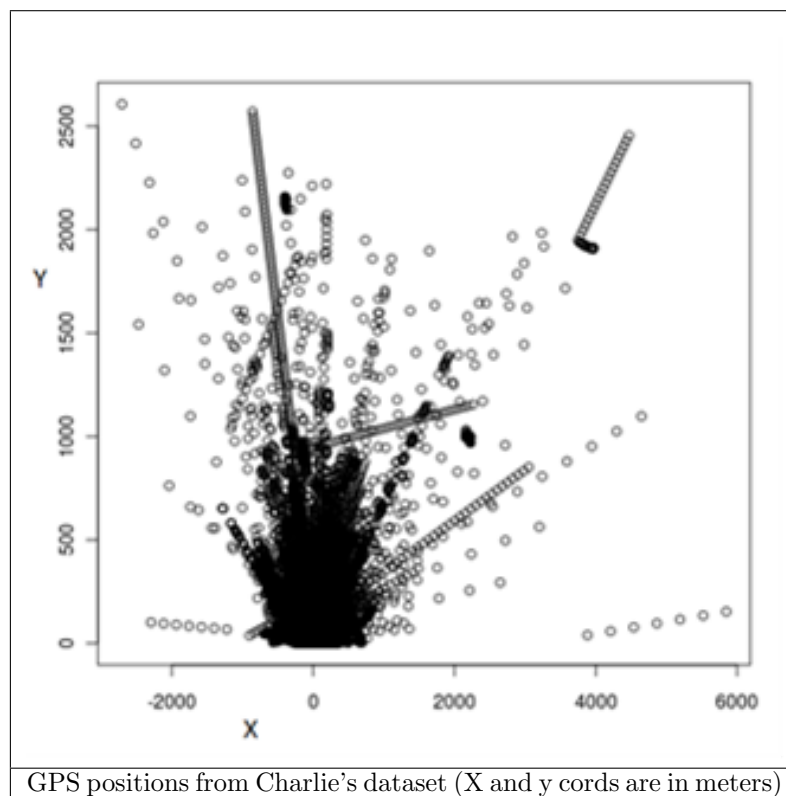


FIGURE 2.26: GPS fixes

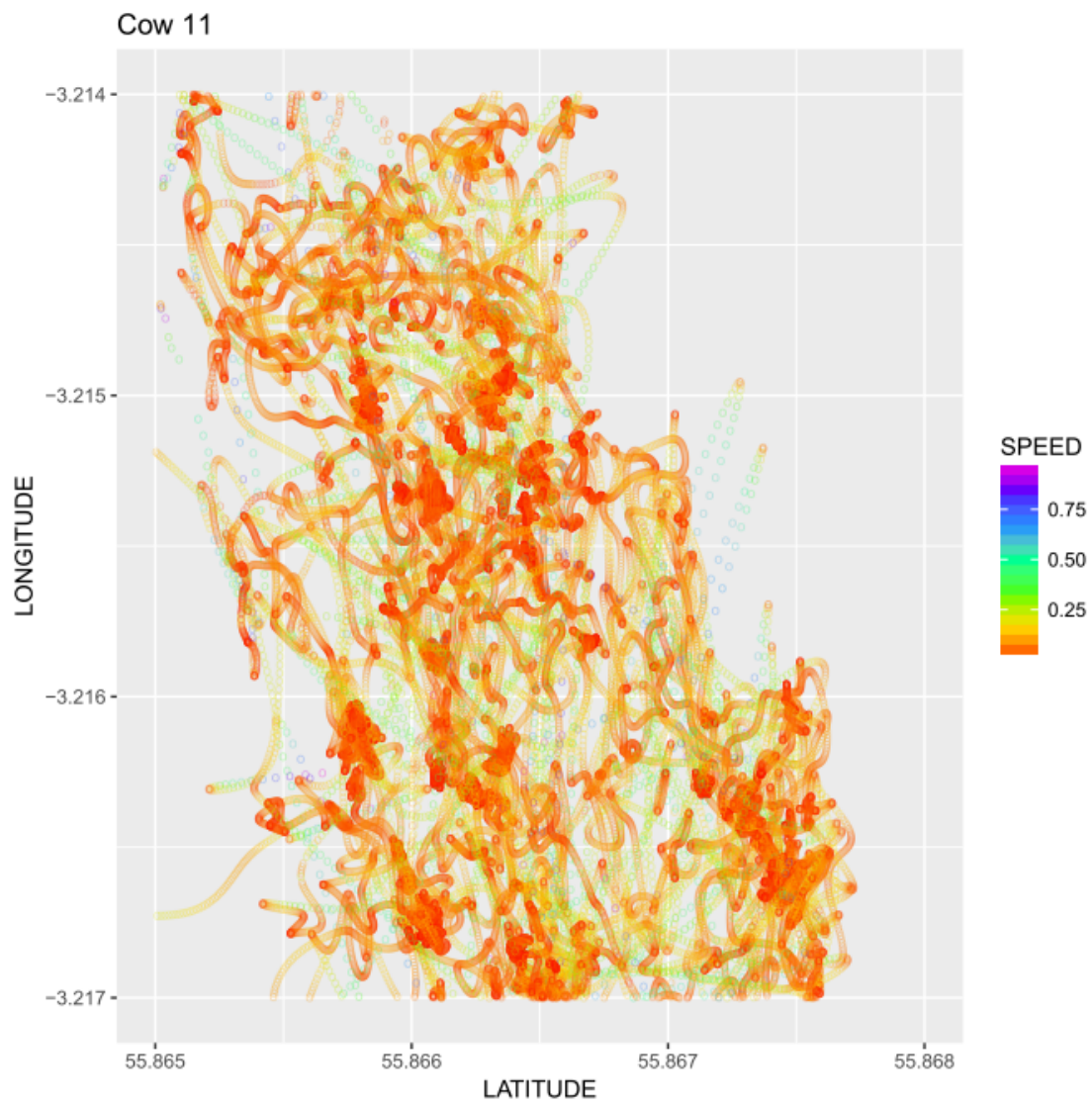


FIGURE 2.25: Filtered GPS fixes for a single cow recorded over 7 days

2.20 Conclusion

WSNSIM is a novel discrete events based simulator for WSN simulation, developed with the goal of producing a value-added specialized interface for specifying protocol and mobility behaviours for analysis and evaluation in the domain of farm monitoring. A key novel aspect of this simulator is the compactness and intuitiveness of its protocol specification interface. In this tool it is possible to specify an entire protocol in about 400 lines of modelling code that sits on the top of the simulator framework. The tool incorporates statistical models of spatial distribution and movements of livestock, so that the node mobility is accurately modelled separately from protocol specification. In a general purpose simulator, the mobility and protocol specifications often become entangled thereby making the model overly complex. Application experiments performed with WSNSIM indicate that this tool can be easily applied in comparing the key performance aspects of data gathering WSN protocols. New protocols were proposed and evaluated using WSNSIM that show low power-consumption characteristics regard to farm requirements.

A variant of the LEACH [72] protocol (modified LEACH) has been proposed, implemented in WSNSIM and evaluated with the goal of reducing unreachable nodes in real herd distributions, without compromising on the power consumption aspect.

A new protocol, called LEMSYP, with two variants have been designed and evaluated using WSNSIM with the goal of having a low-power reliable protocol suitable for farm monitoring. It is a multi-hop protocol that uses short duty cycle to reduce active listening by nodes, and uses a centralised wakeup mechanism similar to the STEM protocol [157]. The two said variants are based on using (1) **TDMA** and (2) **CSMA** for the bulk of the data transfer phase. A comparison study demonstrates the low-power nature of both variants, and that the TDMA variant is a bit superior to the CSMA variant. These novel protocol variants address low-power and reliable data gathering requirements in a farm scenario.

The said protocols have been modelled and evaluated successfully using synthetic herd data, satellite image data, and real-time GPS data.

Besides, modelling features described above, image processing techniques have been used within the WSNSIM tool-chain for automatic detection of cattle positions from satellite images. A novel synthetic herd generation module based on statistical models was implemented for quick generation of simulated herd scenarios.

Polygonal and other shape annotation tools have been added to the simulator for representing obstacles, prohibited zones, and ranges of directional antennas.

A new probabilistic mobility model has been developed from GPS tracked herd data collected previously towards [170]. The new mobility model is more amenable to discrete event simulation than the model created in [170].

A work-flow has been developed by which similar mobility models can be populated from mobility data on other dynamic assets, e.g. other animals. This work-flow involves inferring speed states, heading directions, time between transitions, and transition probabilities from raw experimental time series of tracked positions.

The novelty aspects of WSNSIM are enumerated as follows:

1. Development of a purpose built WSN simulator for farm applications

2. Succinct protocol modelling specification using a new feature of C++
3. A work-flow for designing mobility models from raw GPS time series
4. A tool for extracting herd positions from satellite image of cattle herds
5. A statistical model based generator of synthetic herds for quick generation of simulation scenarios

Chapter 3

Data Acquisition for Human Motion Analysis

3.1 Introduction

Kinect sensor is a 3D scanning device that operates at 30 frames per second and has democratised 3D scanning and many exciting applications are coming up alongside its primary market of gaming interactivity. This work explores the use of Kinect for recording of clinical data using analysis of the point cloud captured by it. Posture, gait, and mobility data can be produced from 3D point cloud sequence that the Kinect is able to produce. Kinect's 3D capture capability could be useful to a variety of applications. Primarily Kinect is a device for video games applications. It is a device with a low resolution depth scanner with a limited range developed by the PrimeSense Company in collaboration with Microsoft. It provides features such as full body 3D motion capture, facial recognition, skeletal tracking, depth sensor to capture depth data, colour camera, etc. It uses a structured lighting based sensor to capture of the depth field of objects within its range, and this depth field is used by algorithms to infer gestures of the player. The Kinect device comes with a software component called the Kinect SDK that provides a high level programming interface to the Kinect's capabilities.

The Kinect sensor returns the depth stream data as a collection of frames. The data is represented in a XYZ frame. A key functionality of the Kinect SDK is to detect the pose of players very quickly but the accuracy of this skeletal pose is quite rough and ready. It uses a statistical learning algorithm to detect the pose, which produces an approximate detection of skeletal parameters, which manifests as unrealistic oscillations in joint angles in the animation replay. This inaccuracy is a trade-off made in favour of performance because accuracy of the gesture capture is less important than computational performance in gaming applications, as games can always snap a perceived gesture to a lattice of gesture intents. This lack of accuracy comes in the way of potential applications that require measurement of pose parameters more precisely. The main assumption of this work is that the depth frame is much more accurate than the skeletal

capture provided by the Microsoft SDK, and it is possible to use better estimation methods to get a more accurate set of skeletal coordinates.

The Kinect recorder that was used in this work is based on Microsoft SDK that has an automatically separating the background from the point cloud feature. The encoding of each pixel in the depth frame in Kinect 1 is represented by a 16 bit short integer. The most significant 13 bits in that frame represents the depth in mm. That allows a range of 0 to 8k mm with 1mm depth resolution. However the actual resolution and the range is below that maximum allowed by the representation. According to the technical specification, the allowed depth range is between 0.8 m and 4m from the Kinect, and the resolution is greater than 10 mm. The three Least Significant Bits (LSB) are used to represent any person detected by the SDK's skeleton engine. This allows for an easy way of classifying the pixels that belong to the person. Kinect is well supported for exploring different applications through software availability. Device drivers and programming libraries are available in all platforms. C++ (various open source libraries) and Tcl/Tk has been used as technology primitives for the 3D interface.

There is a huge market for efficient, portable and affordable monitoring devices meant for health-care applications, and it seems that the Kinect could serve that market with much utility. In this work a Kinect based system and algorithms has been explored towards reducing the cost of gait monitoring while meeting the accuracy requirements of the application. Kinect V2 was released in July 2014. It has higher resolution of the point cloud (twice that of V1.8). Accuracy of Kinect V1 limits its use for collecting data for the hand and fingers, whereas Kinect V2 captures the finger data in greater detail allowing to do the upper limb monitoring presented in this work.

3.2 Known Challenges of Microsoft Kinect

Automatic detection of human limbs is a difficult pattern recognition problem, which is compounded by the relatively low resolution of Kinect. As a result the systems reported in the literature so far has only claimed to identify broad-brush features of gait, rather than fine grained motion capture data. In this context *fine-grained motion-capture* refers to assessing joint angles as a continuous function of time. Whereas the systems reported in the literature and presented in Section 3.3 usually deal with assessment of average stride length, gait cycle time etc. As the Kinect is a novel piece of hardware, application have not been able to fully reach its exploitation potential. This is not a problem with the device but with the inadequacy of the algorithms applied on the captured frames. For example, classifying a scattered set of points accurately into individual limbs or sub-limbs is hard to achieve, especially in a robust way that it works for all configurations. The Kinect sensor requires to be fixed to a specific location in home environment and its range of capture is approximately ten meters for Kinect V1. This limitation states that events like fall must occur directly in front of the physical location of Kinect. Occlusion (by household objects, non-target individuals, pets etc.) is also a problem for home-based continuous monitoring. Kinect cannot capture the fine movements of the foot and thus currently the clinical gait diagnostic potential is limited to gross movements. Xu et al. [198] stated that the accuracy levels of Kinect SDK skeletal data for both V1 and V2 are joint dependent and posture dependent, varies over a very large range. Their results show that the

upper body joints were more accurate than the lower body joints. By their measurements the lower joints (below hip) were off by more than 10cm. Clark et al. [29] found that the Kinect system has the inability to assess internal/external joint rotations in the peripheral limbs which cannot be performed accurately as Kinect is unable to accurately determine a non-joint center. This is a constraint in joint data measurements which limits the angular data for the peripheral joints to abduction/adduction and flexion/extension.

3.3 Literature Review

The Microsoft Kinect incorporates infra-red light and a video camera to create a 3D map of the area in front of it [118] and uses a randomized decision forest algorithm to automatically determine anatomical landmarks on the body, such as joint centers, in close to real time [162]. The results of previous studies are promising, and have shown that the depth sensor itself is accurate for assessing 3D position in a workplace environment [49] and that joint centers derived from the Microsoft Kinect can be used to classify dance gestures [145]. While inexpensive devices such as the Nintendo Wii Balance Board, a clinically feasible alternative to a force platform [28], can provide postural control information related to function [141], it cannot accurately differentiate joint movements. This is most commonly achieved using systems that require multiple cameras and tracking markers placed on the skin, making them cumbersome to house and transport, expensive and requiring extensive technical expertise to operate and interpret. However, the recent development of Microsoft Kinect in computer gaming technology is inexpensive, portable and does not require markers to determine anatomical landmarks. Consequently it may overcome the limitations associated with laboratory-based movement analysis systems. Another study demonstrated the Kinect sensor validity for postural assessment and control [30]. Their experimental results validated Kinect to accurately assess postural control tests which are a lateral reach, a forward reach and a one leg standing balance test. The subject tries to stand in some landmark poses (e.g. one leg up) and the Kinect detects whether and how long that pose is sustained. They provided detailed quantitative joint-by-joint results and found that Kinect was successful to assess the kinematics strategies of postural control. Dutta [49] presents an evaluation of Kinect for clinical applications. This is a somewhat pessimistic result. The analysis finds that the accuracy is much lower than that of biomedical grade systems (e.g. VICON). Stone et al. [171] uses a machine learning method called randomized decision tree, trained using about 900k depth images. It does not need markers to identify body-parts and infers the labelling from pure depth data and prior knowledge of configurations. In their work, Kinect is used to estimate overall Gait characteristics (e.g. stride length, stride time) but not instantaneous time series of joint angles. The work [161] uses a Bayesian machine learning approach to estimate limb orientations from Kinect's depth image in real time. Orientations are approximate and is highly inaccurate for walking frames. Cole et al. [33] presents a joint coordinate system for the representation of 3D positions and orientation of human limbs. Biomechanical Validation is done by a detailed comparison of the Kinect with a VICON system of upper-body and lower-body joint movements of Kinect motion capture data for rehabilitation treatments [55]. Bonnechere et al. [16] concluded that the evidence so far suggests that Kinect is appropriate for functional ability assessment of motion in healthy people.

Kinect has been applied to many areas of physical evaluation such as gait assessment [173], [62], [58], [169], balance tests for stroke patients [31], Timed Up and Go [189], [96], tests for fall prevention in clinical and at home [50], [147], exercise game for elderly people [93], [151], [150], etc. but the validity for general movements or postures using Kinect has yet to be established [198], [127]. Stone and Skubic [173] applied Kinect in continuous in-home gait analysis. In this paper the use of the inexpensive Microsoft Kinect for obtaining measurements of temporal and spatial gait parameters has been compared to an existing web-camera based system, along with a VICON motion capture system for ground truth. It presents an investigation of the Kinect as a technique for acquiring spatial and temporal gait parameters from the depth data of the Kinect. Two different Kinects placed in two different locations have been used for a single person simultaneously. In a coaching context of elderly people, the joint centre positions obtained from Kinect were found to have a variability of 10 cm [127], but the joint angles were not evaluated. Llorens et al. [194] developed a game for estimating chronic stroke patients foot locations and found that virtual training had a significant time effect in the recovery of balance condition in stroke patients. Sadihov et al. [150] developed three rehabilitation mini-games application of motor tasks and game play which are 1) a rope pulling game, 2) a table wiping game and 3) a meteor deflection game using Kinect. The implemented games were presented to stroke patients and the therapists received positive feedback. Mastrokaris et al. [115] used Kinect to detect backward, forward and sideways falls accurately. Zhang et al. [206] used Kinect to detect fall from chair and fall from standing with 94% accuracy. Dutta et al. [49] found that the Kinect V1 depth sensor had an accuracy level with an average error of 14.1 mm to 34.8 mm in different directions and locations. Xu et al. [197] demonstrated that the error in the skeletal tracking of Kinect V1 and V2 sensor struggles in sitting postures and is very large as the joint tracking algorithm was designed for subjects in standing positions in front of the sensor and was not designed for awkward postures. They also found that although Kinect V2 has improved resolution and better depth sensor than Kinect V1, the accuracy of joint centre location identification is not substantially improved.

3.4 Microsoft Kinect

Microsoft Kinect (Figure 3.1) is a sensor device introduced to the mass market primarily for gaming interactivity. It uses a structured lighting based sensor to capture of the depth field of objects within its range, and this depth field is used by algorithms to infer gestures of the player or the subject. It emits a structured pattern of infra-red laser and has a camera to capture reflections of the laser beams from the objects in front. The captured image of the reflected infra-red pattern is then processed by a processor inside the Kinect to work out a depth map of the scene. The distance and the spatial orientation of the surfaces in the surrounding scene decides how the laser dots are reflected, which in turn gets captured by the infra-red camera. Subsequent processing produces a set of 3D points representing the scene. The 3D depth field can then be processed in a computer to identify or register objects of interest. The pose computed by the Kinect SDK in the view field is quite crude but effective and does not provide accurate results [169]. It can be used to measure general trends of human motion and movements in rehabilitation [127].



FIGURE 3.1: Kinect V1 for Xbox 360

The first version is Kinect for Xbox, Figure 3.1, consists of one colour camera, one infrared camera that captures the depth data and one laser light source. Depth data have an 11 bit resolution with values ranging from 0 to 2047 and the produced data streams have a resolution of 640x480 at 30 Hz. It was built to track players that are up to 12 feet (4.0 meters) away from the sensor. But it fails to track objects that are very close (80 cm), and there might be a need to track objects at a very close range for different applications. Kinect for windows has a new firmware which enables near mode tracking. Near Mode allows the Kinect to be more sensitive to closer objects ranging from 40cm to 200cm and the default mode range is from 80cm to 400cm.

Kinect for Windows, Figure 3.2 is a developing device and is not for gaming applications. It is a specially designed PC-centric sensor that helps developers to write their own code and develop real-life applications with human gestures and body motions. The main components of a Kinect device are its color sensor, IR depth sensors, IR emitter, microphone arrays, and a stepper motor that can be tilted to change the Kinect camera angles.

The latest version is Kinect for windows V2, Figure 3.3. It is based on Time-of-flight technology and supports skeletal tracking up to six complete skeletons compared to two with the Kinect for Xbox and Kinect 1.8 and tracking 25 joint positions per individual compared to 20 joints detected by earlier versions. Also infrared data stream can be accessed using Kinect 2. The raw color data stream is at 1920x1020 resolution. All versions of Kinect support capturing data at the rate of 30 frames per second.



FIGURE 3.2: Kinect for Windows V1.8



FIGURE 3.3: Kinect for windows V2

3.5 Kinect for Xbox 360 and Kinect for Windows V1.8 Joint Positions

Kinect V1 tracks upto 20 joints per person as shown in Figure 3.4, Table 3.1 and tracks upto 2 bodies together in the same frame. Kinect provides two methods for representing the skeletal joints. In the first method, (x,y,z) co-ordinates of all the joints are returned. In the second method, the bone orientations are returned. The bone orientations are given as quaternions defined by the angular displacement that would bring the bones from a standardized orientation to its current orientation. The standard bone orientation is that aligned with the vertical axis. The first method is used for investigating the skeleton data as for Kinect V1 API did not provide any bone angle data.

Joint Name	Value	Description
HIP_CENTER	1	Hip centre
SPINE	2	Spine
SHOULDER_CENTER	3	Shoulder centre
HEAD	4	Head
SHOULDER_LEFT	5	Left shoulder
ELBOW_LEFT	6	Left elbow
WRIST_LEFT	7	Left wrist
HAND_LEFT	8	Hand left
SHOULDER_RIGHT	9	Right shoulder
ELBOW_RIGHT	10	Right elbow
WRIST_RIGHT	11	Right wrist
HAND_RIGHT	12	Right hand
HIP_LEFT	13	Left hip
KNEE_LEFT	14	Left knee
ANKLE_LEFT	15	Left ankle
FOOT_LEFT	16	Left foot
HIP_RIGHT	17	Right hip
KNEE_RIGHT	18	Right knee
ANKLE_RIGHT	19	Right ankle
FOOT_RIGHT	20	Right foot

TABLE 3.1: Kinect for Xbox and V1.8 Joint Labels

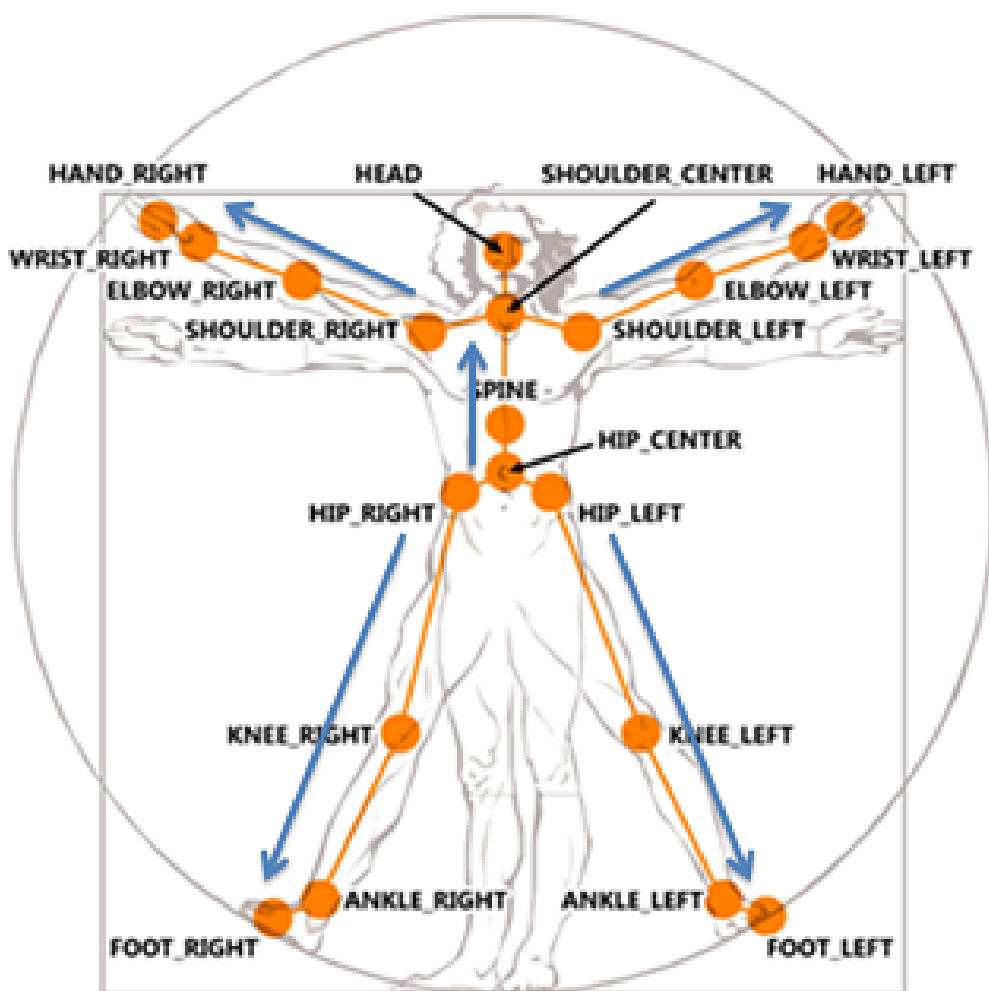


FIGURE 3.4: Kinect for Xbox and V1.8 Joint Positions

3.6 Kinect V2 Joint Positions

Kinect V2 tracks up to 25 joints per individual, shown in Figure 3.5 and Table 3.2 with hand state tracking for two persons. It can track up to 6 bodies simultaneously.

Joint Name	Value	Description
AnkleLeft	14	Left ankle
AnkleRight	18	Right ankle
ElbowLeft	5	Left elbow
ElbowRight	9	Right elbow
FootLeft	15	Left foot
FootRight	19	Right foot
HandLeft	7	Left hand
HandRight	11	Right hand
HandTipLeft	21	Tip of the left hand
HandTipRight	23	Tip of the right hand
Head	3	Head
HipLeft	12	Left hip
HipRight	16	Right hip
KneeLeft	13	Left knee
KneeRight	17	Right knee
Neck	2	Neck
ShoulderLeft	4	Left shoulder
ShoulderRight	8	Right shoulder
SpineBase	0	Base of the spine
SpineMid	1	Middle of the spine
SpineShoulder	20	Spine at the shoulder
ThumbLeft	22	Left thumb
ThumbRight	24	Right thumb
WristLeft	6	Left wrist
WristRight	10	Right wrist

TABLE 3.2: Kinect V2 Joint Labels

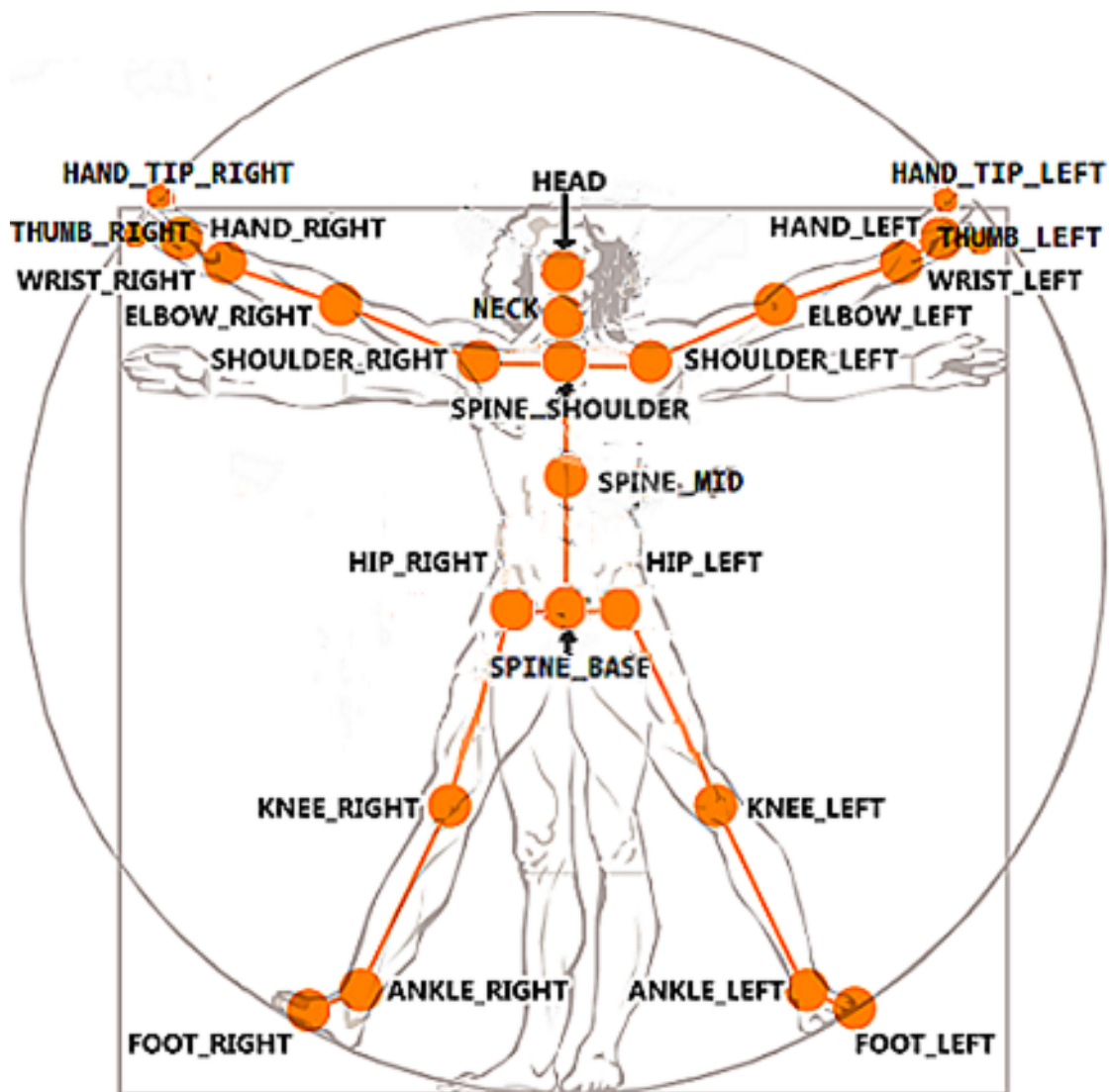


FIGURE 3.5: Kinect V2 Joint Positions

3.7 Depth Data Processing in Microsoft Kinect

The Kinect sensor beams a structured illumination pattern using infra-red laser, and captures the corresponding reflections from the scene as a monochrome image of the patterns. A CMOS (Complementary Metal-Oxide Semiconductor) sensor array samples the image, which then is processed by a system-on-chip microprocessor to create a map of distances of reflecting surfaces from the Kinect. This distance map is processed further in software according to the needs of the application.

Figure 3.6 shows the physical components of the Kinect V1.

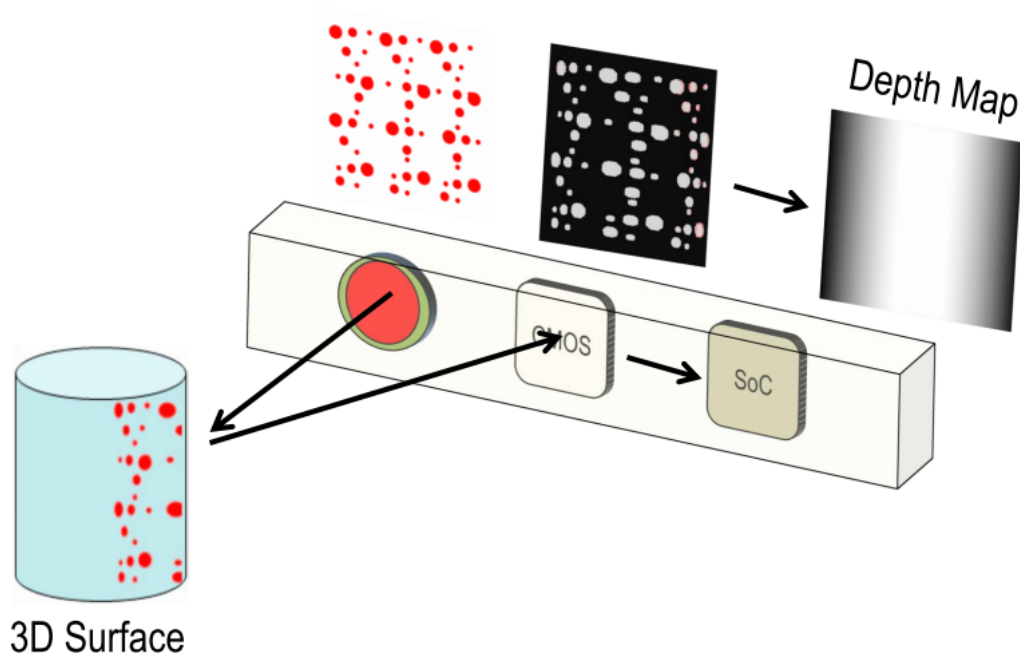


FIGURE 3.6: The mechanism used by Kinect V1 ([1])

The sequence of the operations is as follows. Whenever a capture event is triggered, the PrimeSense System-on-Chip (SoC) processor sends a signal to the infra-red emitter to turn on the infra-red light, and sends another signal to the monochrome camera to initiate capture of the reflected image. The IR emitter meanwhile starts sending an infrared light invisible to human eyes to the objects in front of the device, and the infra-red camera takes a snapshot of it in a frame-buffer. The Primesense SoC processes this frame-buffer and translates the image into a depth-map.

The Kinect 360 sensor was built to track people that are up to 12 feet (4.0 meters) away from the sensor and it fails to track objects that are very close (closer than 80 cm). The Kinect for Windows sensor has newer firmware which enables *Near Mode* tracking. Using *near mode*, Kinect for Windows supports tracking of objects as close as 40 cm in front of the device. In terms of full range and resolution, both the sensors behave the same way. The next generation Kinect 2, has shifted from a structured light approach to a time-of-flight camera. This shift has enabled a quicker response time and a *single-camera* implementation. The new Kinect systems can obtain the same resolution at distance approaching 6 meters and accommodate more people

in the frame, and supports recognition of several more anatomical features. The Kinect 2 system has a field of view of 70 degrees horizontally and 60 degrees vertically, a 920x1080 camera changing from 24-bit RGB color to 16-bit YUV and the depth camera can capture data in the range between 0.50 and 4.50 m. The video, depth frame and recognized skeletal data, streams at 30 fps. The depth resolution has improved dramatically for Kinect 2 from a 320x240 to 512x424. In addition to the RGB stream and the depth frame, the new Kinect can also stream the IR frames. The IR frame can be used as a night vision device, which can take images in darkness or low light. The internal details of Kinect V2 is shown in Figure 3.7. Since this version supports USB 3.0, the data transfer speed is much improved. The 3D resolution of the depth frame depends upon the position of an object with respect to the sensor. The best compromise between resolution and captured region size depends on the application. For the gait capture application, the best distance seems to be about 2.5 m from the sensor, discussed in Section 3.10.

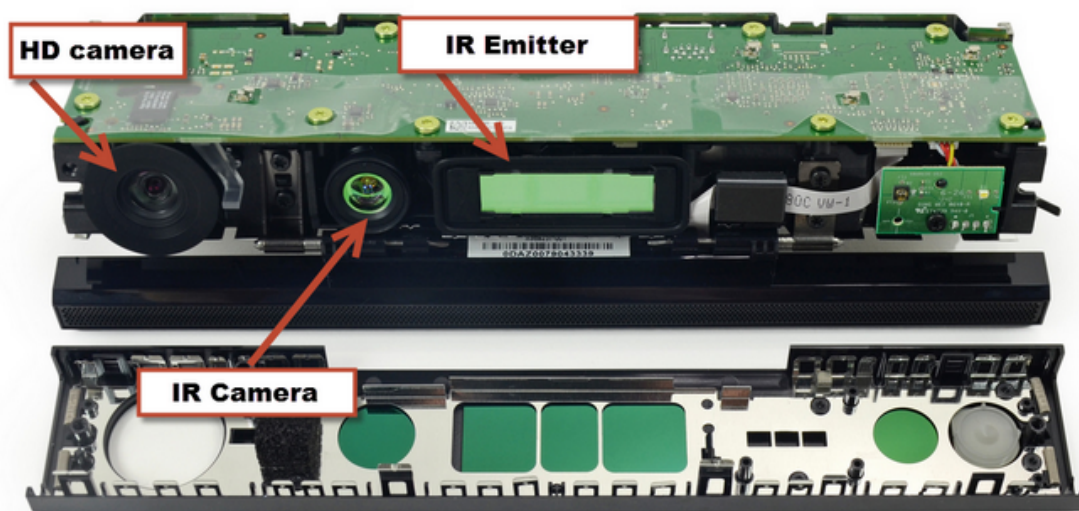


FIGURE 3.7: Microsoft Kinect V2 Internals [113]

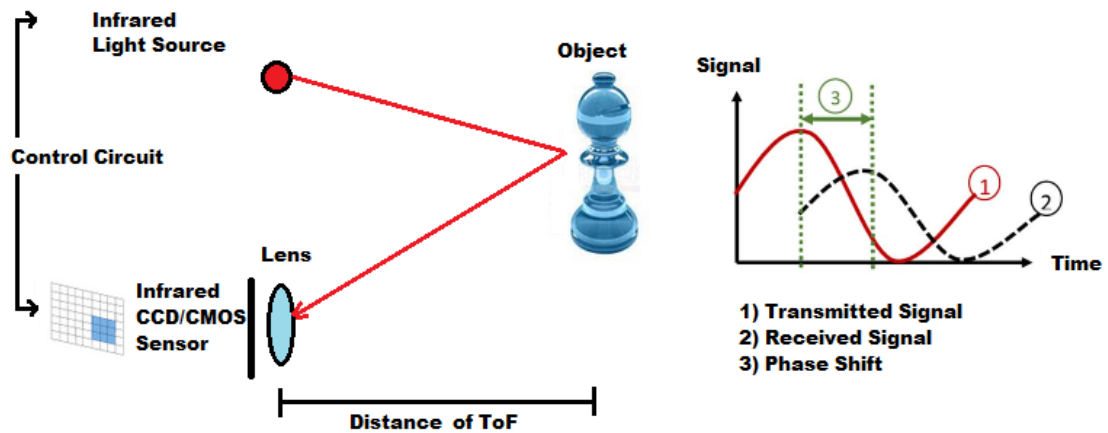


FIGURE 3.8: Operating Principle of a ToF System

Time-of-Flight depth sensor in Kinect 2 uses laser-inducing light to illuminate the scene and uses the actual reflected light to compute the depth and uses multiple modulation frequencies (10-130 MHz). Bamji et al. [9] found that the depth measurement of Kinect 2 is in the range 0.8-4.2 m with the accuracy of the measured range at 0.5%. The operating principle of a ToF system is described in the Figure 3.8. ToF sensors can measure depth by estimating the time delay from the light emission to the light detection. Each light pulse is varied in frequency, and this variation makes each pulse identifiable on detection. So, on detecting a pulse that was emitted recently and reflected back from a surface, it can determine as to which pulse it was and thereby the time spent by the pulse in flight. Considering the travel time of the light pulse, i.e., the time of flight, the distance between a single spot or pixel in the scene and the depth sensor can be calculated. Breuer et al. [22] presents the mechanism and mathematical theory behind time-of-flight imaging. The basic idea is that the illumination is done using a pulsed beams of light and each pixel sensor produces a correlation signal (which in turn is a convolution-type integral computed over a time window) made from the reflected incident pulse and the known source pulse. The distance from the reflection point determines the correlation value, and thereby gives a measure of the distance. Increasing the pulse frequency improves accuracy but limits the range over which the correlation unambiguously represents distance. Time-of-flight imaging systems of this type need to strike a trade-off between accuracy and range. Kinect V2 being a packaged product, presents a particular choice of the trade-off point. An interesting investigation tool might be one that did not fix a trade-off option and give programmatic control over these parameters.

3.8 Some more mass-market depth sensors

Creative Senz3D

Creative Senz3D Figure 3.9 is a Depth and Gesture Recognition Camera for Personal Computers. It costs \$140. This device captures point cloud information. This device is an earlier version of Intel RealSense Figure 3.11 and is not as good as Kinect. The Senz3D is rated for 0.5ft to 3.25ft which is a much shorter range than the Kinect.



FIGURE 3.9: Creative Senz3D

Intel RealSense Camera F200 The RealSense camera F200 shown in Figure 3.10 is a stand-alone camera that can be attached to a desktop or laptop. It is designed for the range of 0.2m to 1.2m while the Kinect V2 is optimized to the range between 0.5m and 4.5m. Kinect V2 could capture the full body actions with all joints while the RealSense is for a desktop usage to capture faces and gestures.



FIGURE 3.10: Intel RealSense Camera F200

Intel RealSense SR300

Intel RealSense SR300 shown in Figure 3.11 is the latest version of Intel RealSense F200 which is used as gesture recognition camera. Intel RealSense technology provides a platform for implementing gesture-based human interaction technique with computer. The current price is £190 for one unit. RealSense camera can be used to capture finger gestures and faces in desktop usage but Kinect can capture the full body actions with all body joints. The effective distance range of RealSense is optimized to 0.2m to 1.2m whereas the range of Kinect V2 is 0.5m to 4.5m. Alyuz et al. [5] used RealSense camera F200, Figure 3.10, for feature extraction like appearance features



FIGURE 3.11: Intel RealSense SR300 Camera

to locate head position and facial expressions. Vasquez et al. [188] have used Intel RealSense SDK for voice and face recognition in their multi-modal recognition application. The device they used was an earlier version of RealSense, Creative Sense 3D Figure 3.9 and Microsoft Kinect.

Apple Primesense Carmine 1.09

Apple Primesense Carmine 1.09 shown in Figure 3.12 is a short-range RGB 3D webcam sensor for point cloud acquisition and its main functions are facial scanning, eye and movement tracking, gesture control, 3D scanning and 3D modeling, 3D mapping, 3D rendering, industrial automation and user extraction. It costs \$325. Galbally et al. [60] have used PrimeSense Carmine 1.09 sensor and Microsoft Kinect for 3D face spoofing and interoperability experiments of 3D and 2.5D systems.



FIGURE 3.12: Apple primesense Carmine1.09

Structure sensor

The Structure Sensor is the first 3D sensor for mobile devices that is compatible with the 4th generation iPad and the iPad mini with Retina Display. It is a cable-free range imaging device that provides depth data for 3D imaging which has great potential for 3D mapping and gaming and costs £379. Kilgus et al. [95] used structure sensor for visualizing complex 3D anatomy and proposed that it can be used for medical and forensic studies.

Leap motion controller



FIGURE 3.13: Structure Sensor



FIGURE 3.14: Leap Motion

Leap Motion is a device for hand gesture control. It returns only some hand pose features and a set of relevant hand points but does not return a complete depth map. Leap motion sensor is not always able to recognize all the fingers [193]. Vikram et al. [190] uses Leap motion for handwriting and gesture recognition applications. It costs £50.







Feature ↓ Device →	Kinect for Xbox 1.0	Kinect for Windows 1.8	Microsoft Kinect V2	Leap Motion Controller	Intel RealSense SR300	Structure Sensor
						
Technology	Structured Light	Structured Light	Time-of-Flight	Stereo Cameras	Time-of-Flight	Structured light
RGB camera	640x480 @ 30fps	640x480 @ 30fps	1920x1080 @ 30fps	undisclosed	1280x720 @ 60fps	640x480 @ 30/60fps
Depth Sensors	320x240 @ 30fps	640x480 @ 30fps	512x424 @ 30fps	undisclosed	640x480 @ 60fps	640x480 @ 30/60fps
Microphone	Voice-array	Quad-array	Quad-array	-	Dual-array	-
Horizontal Field of View (FOV)	57 degrees	57 degrees	70 degrees	150 degrees	70 degrees	58 degrees
Vertical FOV	43 degrees	43 degrees	60 degrees	120 degrees	43 degrees	45 degrees
Range	1.2 m to 3.5 m	0.4 m to 4.5 m	0.5 m to 4.5 m	0.03 m to 0.6 m	0.2 m to 1.2 m	0.4 m to 3.5 m
USB	1.0	2.0	3.0	3.0	3.0	2.0
Skeletons Tracked	2	2	6	-	-	-
Body Joints	20	20	25	-	-	-
Portability	No	No	No	Yes	Yes	Yes
Gestures Tracking	Yes	Yes	Yes	Yes	Yes	Yes
SDK	Yes	Yes	Yes	Yes	Yes	Yes
Price	£30	£199	£199	£50	£190	£379

TABLE 3.3: Feature summary of mass-market depth sensors

3.9 VICON MX System

The VICON motion capture system (VICON MX, Oxford Metrics Ltd, Oxford, UK) is a state-of-the-art infra-red marker-tracking system that offers millimeter resolution of 3D spatial displacements. It is the golden standard and the benchmark for the proposed system to compare against. The VICON system consists of twelve cameras outfitted with an array of IR LEDs, IR optical filters and a set of reflective dots. The VICON raw data is quite noisy and has gaps. These are addressed by gap filling and a combination of various filters which are chosen using a graphic user interface for VICON, which applies tools like 6-th order Butterworth filter and Woltring filter, in order to produce the smoothed data.

3.9.1 VICON Nexux software

In the VICON software, a three-dimensional model is constructed which is the representation of the original markers placed on the subject. Figure 3.15 is a screenshot of a gait model and 3.16 is a screenshot of an upper limb model using VICON Nexux 2.1.1 software in the Bioengineering VICON laboratory. The VICON knee angle data and the VICON elbow angle data are simulated and computed for the gait and upper limb experiments for the comparison with Kinect results.

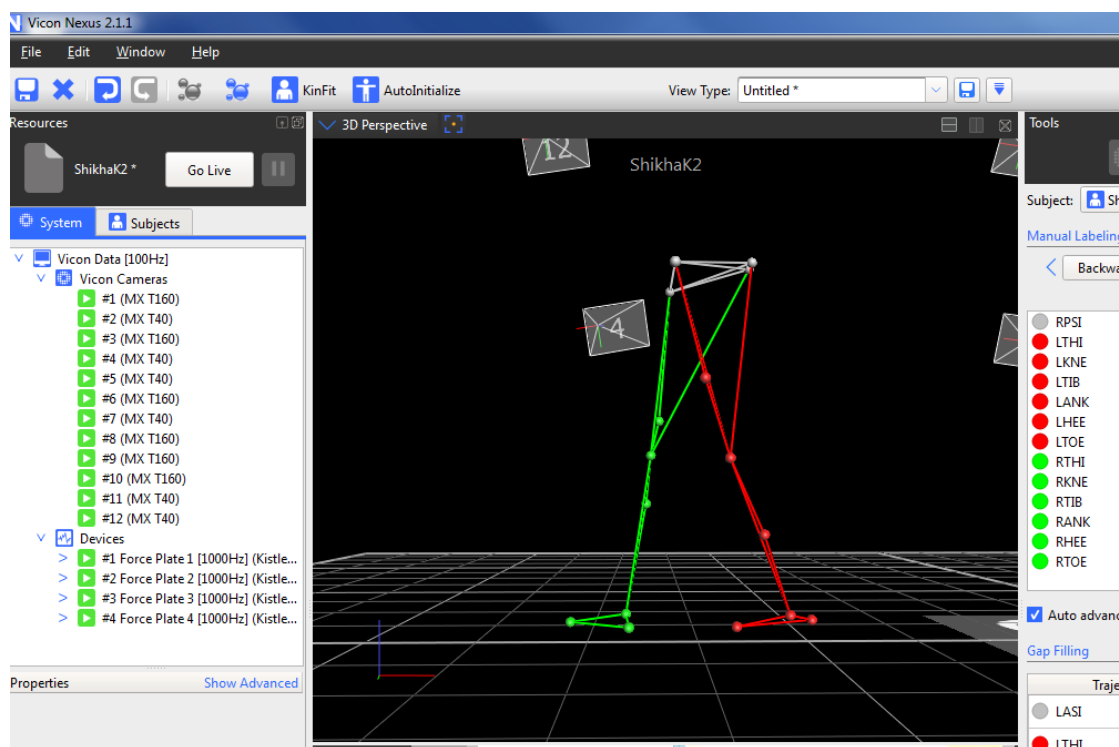


FIGURE 3.15: Gait Model constructed using VICON Nexux 2.1.1 software

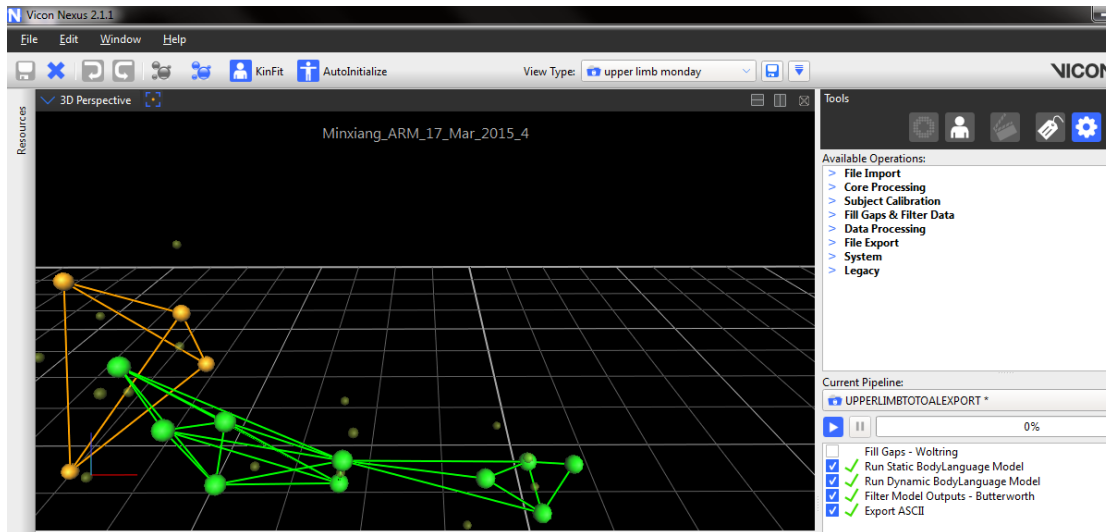


FIGURE 3.16: Upper limb model constructed using VICON Nexus 2.1.1 software

3.9.2 Drawbacks of VICON Nexus Software

The VICON Plug-in-Gait model in VICON/Nexus software was developed two decades ago and has several drawbacks such as overlapping trajectories and ghost markers [2]. It is extremely sensitive to errors in location and orientation of the knee-joint axis, and considerable care and practice was needed. It does not allow consistent musculoskeletal analysis, it uses limited number of markers, the regression equation used to estimate the hip joint centre is not the best. Also it does not allow to customize models.

3.10 Precision of Microsoft Kinect - Depth measurement accuracy:

The following calibration procedure was carried out to estimate the error of Kinect V1. In this experiment a sufficiently inflated gym ball was used. The Kinect V1 sensor was located statically and was used as the reference plane. The gym-ball was placed at various distances from the Kinect i.e., it was progressively moved towards Kinect V1 from 4.0 m to 0.8 m with average regular steps of approximately 40 cm during data collection and its depth frames were recorded as shown in Figure 3.17. Pagliari et al. [130] used a plane wall as the reference point and progressively moved the sensor Kinect V1 at various distances from 0.8 m to 4.0 m for error estimation of Kinect V1. Khoshelham [94] also used a planar wall as a reference object and used plane fitting residues as a measure of accuracy. Both of these studies found that the Kinect V1 error is about 0.01 m for 3.0 m distance. This matches almost exactly with our estimates. They (i.e. [94], [130]) have used precise measurement of distance to specify the reference values, whereas the current work used a spherical object as reference so that precise measurement is not necessary. The deviation from spherical shape is taken as the difference from reference (i.e. the error). In this experiment an inflatable exercise ball (often called a *gym-ball*) was used as the said spherical reference. When fully inflated, the gym-ball assumes a fairly perfect spherical shape. The gym-ball was placed at various distances from the Kinect and its depth frames were recorded. The point cloud corresponding to the gym-ball was isolated from the rest of the scene for a number of depth-frames and a least-squares sphere was fitted to the point-cloud. The mean deviation of the points from the least-squares fitted sphere was taken as the error measure.

Figure 3.18 shows the points in a particular frame's point-cloud (shown in red) displayed along with the best-fit sphere (shown in green).

In Figure 3.18, some points can be seen to be outside the green sphere whereas there are some inside. The overall deviation from the best-fit sphere is plotted against the distance to show the trend of position errors. The point cloud corresponding to the gym-ball was isolated from the rest of the scene for a number of depth-frames and a least-squares sphere was fitted to the point-cloud. The mean deviation of the points from the least-squares fitted sphere was taken as the error measure. Figure 3.19 shows a plot of the least squares error function. The following error formula represents the mean of the actual Euclidean distance of the sample points from the surface of the least squares fitted sphere:

$$err = \frac{1}{N} \sum_{i=1}^N |\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2} - R|$$

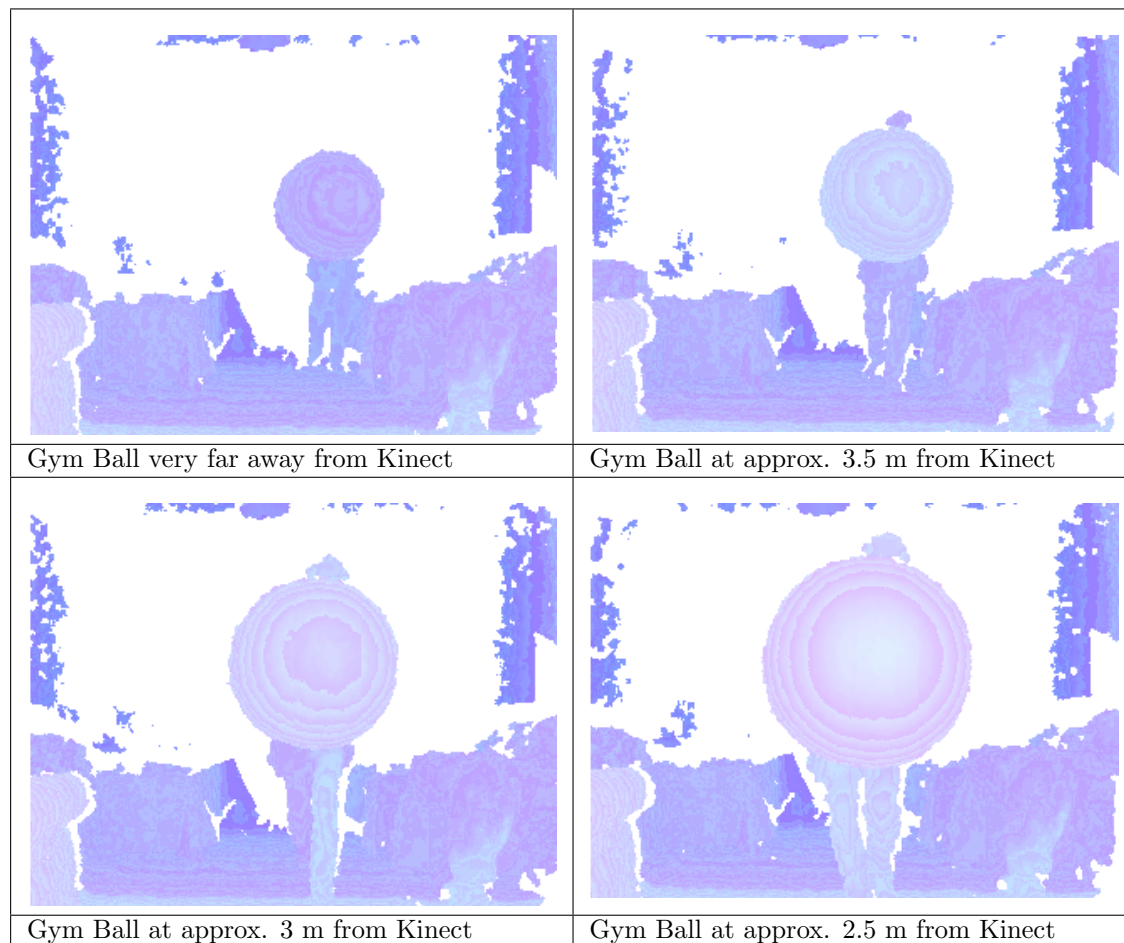


FIGURE 3.17: Frames captured for the gym-ball

Let the point cloud be represented by the set $(x_i, y_i, z_i) : i = 1, \dots, N$. For ease of analysis, the transformed coordinates $(u_i, v_i, w_i) : i = 1, \dots, N$ are defined as follows so that their sum vanishes.

Let $u_i = x_i - \bar{x}$, $v_i = y_i - \bar{y}$, $w_i = z_i - \bar{z}$

where

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^N y_i, \quad \bar{z} = \frac{1}{N} \sum_{i=1}^N z_i$$

This change of variables only makes a translation transform w.r.t. the original variables. So when a Least Squares sphere is computed in (u, v, w) coordinates, the coordinates in (x, y, z) spaces can be computed by shifting back by $(\bar{x}, \bar{y}, \bar{z})$. The error function can be formulated as follows,

$$S = \sum_{i=1}^N (g(u_i, v_i, w_i))^2$$

where,

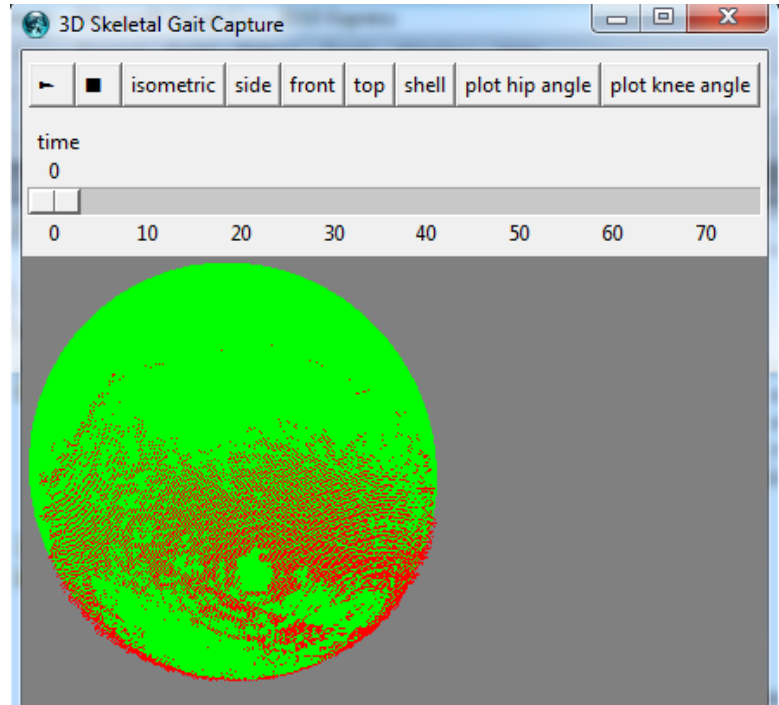


FIGURE 3.18: Point cloud for the gym-ball and its best-fit sphere

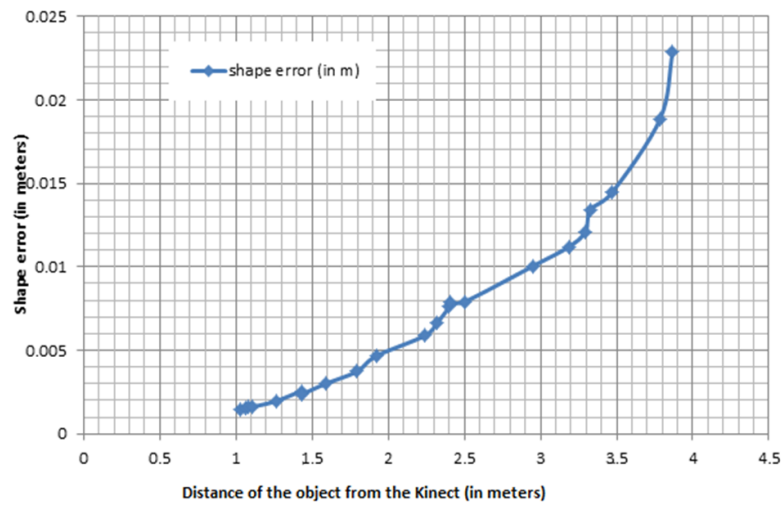


FIGURE 3.19: Position error as a function of distance from the Kinect sensor

$$g(u, v, w) = (u - u_c)^2 + (v - v_c)^2 + (w - w_c)^2 - R^2$$

R is the radius of the sphere and (u_c, v_c, w_c) is the centre of the sphere in $u - v - w$ space.

Let α denote R^2 , so $\frac{\partial g}{\partial \alpha} = -1$, which is used as follows.

$$\frac{\partial S}{\partial \alpha} = 2 \sum_{i=1}^N g(u_i, v_i, w_i) \frac{\partial g(u_i, v_i, w_i)}{\partial \alpha}$$

$$= -2 \sum_{i=1}^N g(u_i, v_i, w_i)$$

for S to be minimum,

$$\frac{\partial S}{\partial \alpha} = 0$$

Thus,

$$\begin{aligned} \sum_{i=1}^N g(u_i, v_i, w_i) &= 0 & (3.1) \\ \frac{\partial S}{\partial u_c} &= 2 \sum_{i=1}^N g(u_i, v_i, w_i) \frac{\partial g(u_i, v_i, w_i)}{\partial u_c} \\ &= 2 \sum_{i=1}^N (-2(u_i - u_c)g(u_i, v_i, w_i)) \\ &= -4 \sum_{i=1}^N u_i g(u_i, v_i, w_i) + 4u_c \sum_{i=1}^N g(u_i, v_i, w_i) \end{aligned}$$

By 3.1,

$$\frac{\partial S}{\partial u_c} = -4 \sum_{i=1}^N u_i g(u_i, v_i, w_i)$$

For minimum, $\frac{\partial S}{\partial u_c} = 0$ gives, $\sum_{i=1}^N u_i g(u_i, v_i, w_i) = 0$

Similarly,

$$\frac{\partial S}{\partial v_c} = 0 \text{ gives, } \sum_{i=1}^N v_i g(u_i, v_i, w_i) = 0$$

$$\text{and } \frac{\partial S}{\partial w_c} = 0 \text{ gives, } \sum_{i=1}^N w_i g(u_i, v_i, w_i) = 0$$

Next, expanding the equation $\sum_{i=1}^N u_i g(u_i, v_i, w_i) = 0$ leads to the following :

$$\sum_{i=1}^N u_i (u_i^2 - 2u_i u_c + u_c^2 + v_i^2 - 2v_i v_c + v_c^2 + w_i^2 - 2w_i w_c + w_c^2 - \alpha) = 0$$

$$\begin{aligned}
& u_i^3 - 2u_c \sum_{i=1}^N u_i^2 + u_c^2 \sum_{i=1}^N u_i + \sum_{i=1}^N v_i^2 u_i - 2v_c \sum_{i=1}^N u_i v_i + v_c^2 \sum_{i=1}^N u_i \\
& + \sum_{i=1}^N w_i^2 u_i - 2w_c \sum_{i=1}^N u_i w_i + w_c^2 \sum_{i=1}^N u_i - \alpha \sum_{i=1}^N u_i = 0
\end{aligned} \tag{3.2}$$

Let,

$$\sum_{i=1}^N u_i = S_u, \quad \sum_{i=1}^N u_i^2 = S_{uu}, \quad \sum_{i=1}^N u_i^3 = S_{uuu}, \quad \sum_{i=1}^N v_i^2 u_i = S_{vuu}, \quad \sum_{i=1}^N u_i v_i = S_{uv}, \quad \text{etc.}$$

Thus,

$$S_{uuu} - 2u_c S_{uu} + u_c^2 S_u + S_{vuu} - 2v_c S_{uv} + v_c^2 S_u + S_{wuu} - 2w_c S_{uw} + w_c^2 S_u - \alpha S_u = 0$$

But $S_u = 0$

So,

$$2[u_c(S_{uu}) + v_c(S_{uv}) + w_c(S_{uw})] = S_{uuu} + S_{vuu} + S_{wuu}$$

i.e.,

$$u_c S_{uu} + v_c S_{uv} + w_c S_{uw} = \frac{1}{2}(S_{uuu} + S_{vuu} + S_{wuu})$$

Similarly, from $\sum_{i=1}^N v_i g(u_i, v_i, w_i) = 0$ and $\sum_{i=1}^N w_i g(u_i, v_i, w_i) = 0$ the following equations are obtained:

$$u_c S_{uv} + v_c S_{vv} + w_c S_{vw} = \frac{1}{2}(S_{uvv} + S_{vvv} + S_{wvv})$$

$$u_c S_{wu} + v_c S_{wv} + w_c S_{ww} = \frac{1}{2}(S_{uwv} + S_{vww} + S_{www})$$

This gives three linear equations in u_c, v_c and w_c which can be solved to determine the values of u_c, v_c and w_c .

Expanding 3.1 to find α ,

$$\sum_{i=1}^N g(u_i, v_i, w_i) = 0$$

$$\sum_{i=1}^N (u_i^2 - 2u_i u_c + u_c^2 + v_i^2 - 2v_i v_c + v_c^2 + w_i^2 - 2w_i w_c + w_c^2 - \alpha) = 0$$

$$S_{uu} + S_{vv} + S_{ww} + N(u_c^2 + v_c^2 + w_c^2) = N\alpha$$

$$\alpha = u_c^2 + v_c^2 + w_c^2 + \frac{S_{uu} + S_{vv} + S_{ww}}{N}$$

Once the values of α, u_c, v_c, w_c are found, the precision estimate would be:

$$Error = \frac{1}{R^2} \sqrt{\frac{1}{N} \sum_{i=1}^N (g(u_i, v_i, w_i))^2}$$

Algorithm 2 presents the aforementioned sphere-fitting formulation in pseudo-code form. In this pseudo-code notation, the arrays are shown in boldface. Arithmetic operations involving arrays are defined to be element-wise (e.g. the product $\mathbf{v} * \mathbf{w}$ is an array whose i^{th} element is the product of the i^{th} element of \mathbf{u} and the i^{th} element of \mathbf{v} . Similarly the arithmetic operation involving an array and a non-array as in $\mathbf{x} - x_{avg}$ is an array whose i^{th} element is the corresponding element of \mathbf{x} minus the non-array (i.e. scalar) x_{avg} . Position error leads to angular error in the determination of limb orientation. Consider that the limb's orientation is being defined in terms of its two end points in 3D. The perceived end points might have error according to the aforementioned position error trend. The worst angular deviation configuration is given in Figure 3.20.

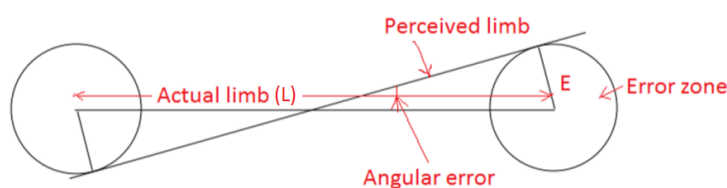


FIGURE 3.20: Angular error defined in terms of the position error E

The Figure 3.20 shows that angular error is $\arcsin(\frac{2E}{L})$ where L is the length of the limb and E is the position error. As the position error varies with distance from the Kinect sensor, so does the limb angle error.

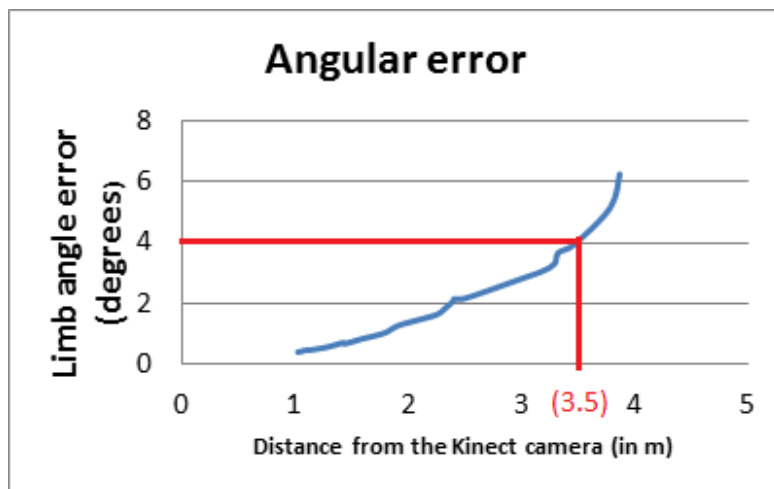


FIGURE 3.21: Limb's angular error vs. distance from the Kinect. Any distance $\leq 3.5m$ would satisfy the Bioengineering requirement.

The result in Figure 3.21 shows that with a properly designed algorithm it is possible to achieve an angular accuracy of 3 degrees when the subject is about 3m away from the Kinect camera. In bioengineering application the maximum acceptable error is 5 degrees. So the subject was placed at about 2.5m (minimum) from the Kinect camera for the best compromise between the accuracy and frame span.

ALGORITHM SPHERE FIT

Data: A set of N 3D points onto which a sphere is fitted. The array of $x, y,$ and z coordinates are denoted by $\mathbf{x}, \mathbf{y},$ and \mathbf{z}

Result: A sphere fitted on the given points

$$x_{avg} \leftarrow \frac{1}{N} \sum_{i=1}^N \mathbf{x};$$

$$y_{avg} \leftarrow \frac{1}{N} \sum_{i=1}^N \mathbf{y};$$

$$z_{avg} \leftarrow \frac{1}{N} \sum_{i=1}^N \mathbf{z};$$

$$\mathbf{u} \leftarrow \mathbf{x} - x_{avg}; \mathbf{v} \leftarrow \mathbf{y} - y_{avg}; \mathbf{w} \leftarrow \mathbf{z} - z_{avg};$$

$$S_{uu} \leftarrow \sum_{i=1}^N \mathbf{u} * \mathbf{u};$$

$$S_{vu} \leftarrow \sum_{i=1}^N \mathbf{u} * \mathbf{v};$$

$$S_{uw} \leftarrow \sum_{i=1}^N \mathbf{u} * \mathbf{w};$$

$$S_{ww} \leftarrow \sum_{i=1}^N \mathbf{w} * \mathbf{w};$$

$$S_{vv} \leftarrow \sum_{i=1}^N \mathbf{v} * \mathbf{v};$$

$$S_{vw} \leftarrow \sum_{i=1}^N \mathbf{v} * \mathbf{w};$$

$$S_{uuu} \leftarrow \sum_{i=1}^N \mathbf{u} * \mathbf{u} * \mathbf{u};$$

$$S_{uuv} \leftarrow \sum_{i=1}^N \mathbf{u} * \mathbf{u} * \mathbf{v};$$

$$S_{uuw} \leftarrow \sum_{i=1}^N \mathbf{u} * \mathbf{u} * \mathbf{w};$$

$$S_{vvu} \leftarrow \sum_{i=1}^N \mathbf{v} * \mathbf{v} * \mathbf{u};$$

$$S_{vvv} \leftarrow \sum_{i=1}^N \mathbf{v} * \mathbf{v} * \mathbf{v};$$

$$S_{v vw} \leftarrow \sum_{i=1}^N \mathbf{v} * \mathbf{v} * \mathbf{w};$$

$$S_{wvu} \leftarrow \sum_{i=1}^N \mathbf{w} * \mathbf{w} * \mathbf{u};$$

$$S_{wvv} \leftarrow \sum_{i=1}^N \mathbf{w} * \mathbf{w} * \mathbf{v};$$

$$S_{www} \leftarrow \sum_{i=1}^N \mathbf{w} * \mathbf{w} * \mathbf{w};$$

$$A \leftarrow \begin{pmatrix} S_{uu} & S_{vu} & S_{uw} \\ S_{vu} & S_{vv} & S_{vw} \\ S_{uw} & S_{vw} & S_{ww} \end{pmatrix};$$

$$B \leftarrow \frac{1}{2} \begin{pmatrix} S_{uuu} + S_{vvu} + S_{wvu} \\ S_{uuv} + S_{vvv} + S_{wvv} \\ S_{uuw} + S_{vvw} + S_{www} \end{pmatrix};$$

Solve the linear system $Ax = B$. Let the solution be $\begin{pmatrix} u_c \\ v_c \\ w_c \end{pmatrix};$

$$\alpha \leftarrow u_c^2 + v_c^2 + w_c^2 + (S_{uu} + S_{vv} + S_{ww})/N;$$

$$R \leftarrow \sqrt{\alpha};$$

$$x_c \leftarrow u_c + x_{avg};$$

$$y_c \leftarrow v_c + y_{avg};$$

$$z_c \leftarrow w_c + z_{avg};$$

$$\text{Fitting error} \leftarrow \frac{1}{N} \sum_{i=1}^N \left| \sqrt{(\mathbf{x} - x_c)^2 + (\mathbf{y} - y_c)^2 + (\mathbf{z} - z_c)^2} - R \right|;$$

Return the sphere centred at x_c, y_c, z_c with radius R ;

Algorithm 2: Finding the best-fit sphere for a set of points

3.10.1 Precision Calculation of Microsoft Kinect V2 Depth Sensor

A planar reference object was chosen for estimating the error in the depth frame of Kinect V2.

A plane was fitted using least squares method and this plane was taken to be the reference geometry. Any deviation of the sampled points from the reference plane was taken as error. The average error was noted for several positions of the reference object starting from about 4.0 metres and slowly bringing it closer to the Kinect 2 sensor upto 0.8 metres as shown in Figure 3.22.

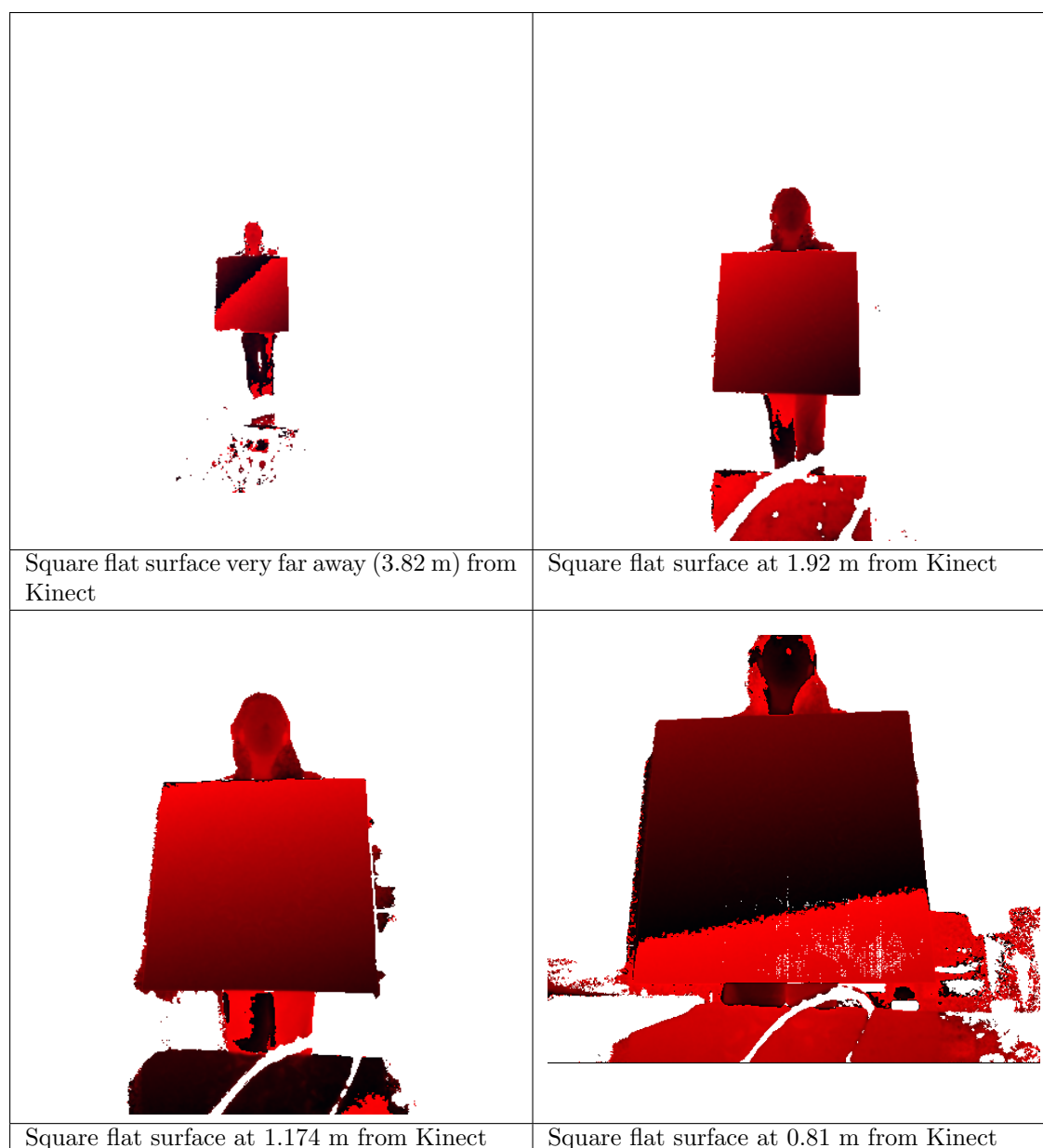


FIGURE 3.22: Frames captured for the square flat surface

Figure 3.23 shows the 3D representation of point cloud with normals estimated. In this figure, the normals are not explicitly shown as arrows but in the form of shading that normals facilitate.

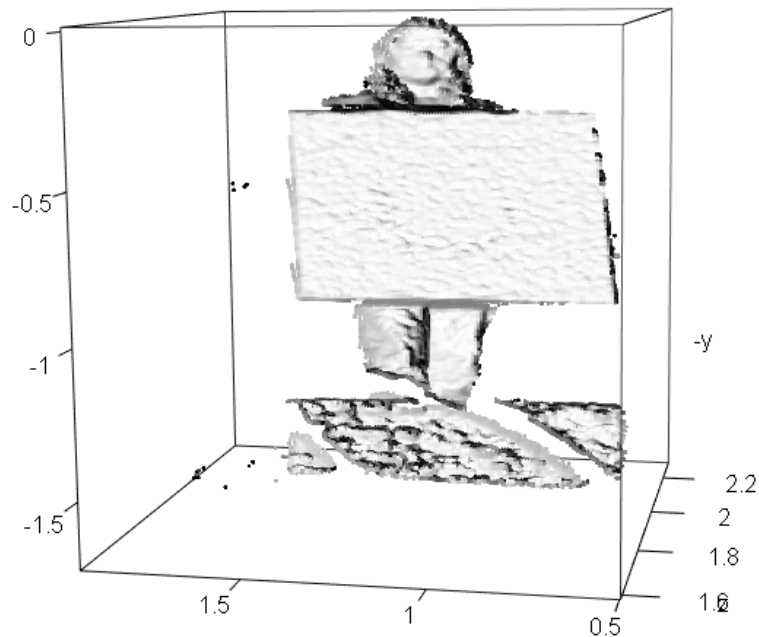


FIGURE 3.23: 3D representation of point cloud with normals estimated

The average deviation from the fitted plane was plotted against distance between the reference object and the Kinect is shown in Figure 3.24.

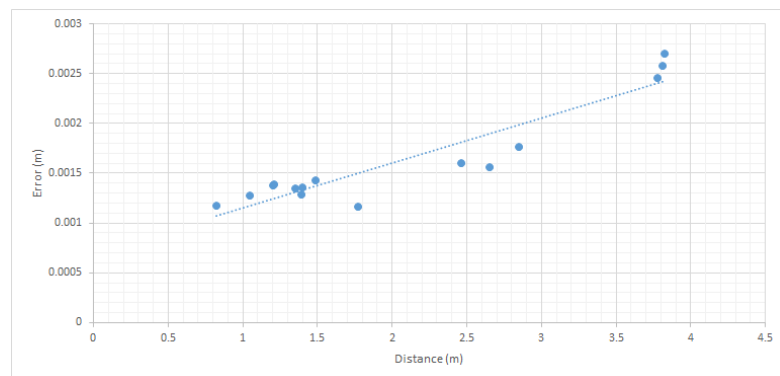


FIGURE 3.24: Position error as a function of distance from Kinect V2 sensor

The mathematical derivation of the plane fitting formulation is the same as that of normal estimation as given in Section 4.4. After all, normals are estimated by locally fitting a plane with points sampled from the topological neighbourhood of each point. In the normal estimation method, each call to plane fitting receives a small number of points whereas in this error estimation exercise, a large number of points sampled on the reference object's depth image were fitted using the least squares method. Algorithm 3 presents the plane fitting calculation in pseudo-code form. As in Algorithm 2, here bold-faced symbols represent arrays, and arithmetic on arrays is

defined to be element-wise.

ALGORITHM PLANE FIT

Data: A set of N 3D points onto which a plane is fitted. The arrays of x, y , and z coordinates of the points are denoted by \mathbf{x} , \mathbf{y} , and \mathbf{z}

Result: A plane fitted on the given points

$$S_{xx} \leftarrow \sum \mathbf{x} * \mathbf{x};$$

$$S_{yy} \leftarrow \sum \mathbf{y} * \mathbf{y};$$

$$S_{zz} \leftarrow \sum \mathbf{z} * \mathbf{z};$$

$$S_{xy} \leftarrow \sum \mathbf{x} * \mathbf{y};$$

$$S_{yz} \leftarrow \sum \mathbf{y} * \mathbf{z};$$

$$S_{zx} \leftarrow \sum \mathbf{z} * \mathbf{x};$$

$$S_x \leftarrow \sum \mathbf{x};$$

$$S_y \leftarrow \sum \mathbf{y};$$

$$S_z \leftarrow \sum \mathbf{z};$$

$$x_{avg} \leftarrow \frac{S_x}{N};$$

$$y_{avg} \leftarrow \frac{S_y}{N};$$

$$z_{avg} \leftarrow \frac{S_z}{N};$$

$$A_{11} \leftarrow S_{xx} - 2 * x_{avg} * S_x + N * x_{avg} * x_{avg};$$

$$A_{22} \leftarrow S_{yy} - 2 * y_{avg} * S_y + N * y_{avg} * y_{avg};$$

$$A_{33} \leftarrow S_{zz} - 2 * z_{avg} * S_z + N * z_{avg} * z_{avg};$$

$$A_{21} \leftarrow A_{12} \leftarrow S_{xy} - x_{avg} * S_y - y_{avg} * S_x + N * x_{avg} * y_{avg};$$

$$A_{31} \leftarrow A_{13} \leftarrow S_{zx} - x_{avg} * S_z - z_{avg} * S_x + N * x_{avg} * z_{avg};$$

$$A_{23} \leftarrow A_{32} \leftarrow S_{yz} - z_{avg} * S_y - y_{avg} * S_z + N * z_{avg} * y_{avg};$$

$$A \leftarrow \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix};$$

Perform eigenvalue decomposition of A ;

Let $\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$ be the eigenvector of A that corresponds to the smallest eigenvalue of A ;

Return the plane passing through the point $(x_{avg}, y_{avg}, z_{avg})$ with normal (n_x, n_y, n_z) ;

Algorithm 3: Algorithm for computation of best-fit plane for a set of points

Yang et al. in [202] presents a study in which the accuracy of Kinect 2 is presented as a spatial function over vertical and horizontal planes (assuming that the kinect's base is placed horizontally). Instead of showing a continuous map of accuracy on these planes, some broad contours have been shown. The contour representing an accuracy of upto 2mm on both vertical and horizontal planes ends at a distance of 3 m from the Kinect. This measurement is in perfect agreement with the estimated accuracy. This is shown by the trend-line in Figure 3.24, which crosses 2 mm error level for a distance of 3m.

3.11 Kinect 1 Recorder

The Kinect SDK returns a frame buffer of depth values which is a rectangular array of integers representing the distance at which a ray from the Kinect sensor goes through the corresponding pixel and hits a surface. There is a calculation procedure by which the array element position and the depth values can be combined to give the x, y, z co-ordinates of a point. Many such points put together to make a cloud of points in 3D space. A Kinect recorder was developed specifically for capturing and storing the recorded depth data from Kinect V1 at the rate of 30fps because the skeletal data generated directly by the Kinect SDK was unsuitable. To view the point cloud of the subject in isolation, the background has been separated using a special bitmap provided by the Kinect SDK. Some recorded images are shown in Figure 3.25 and 3.26.

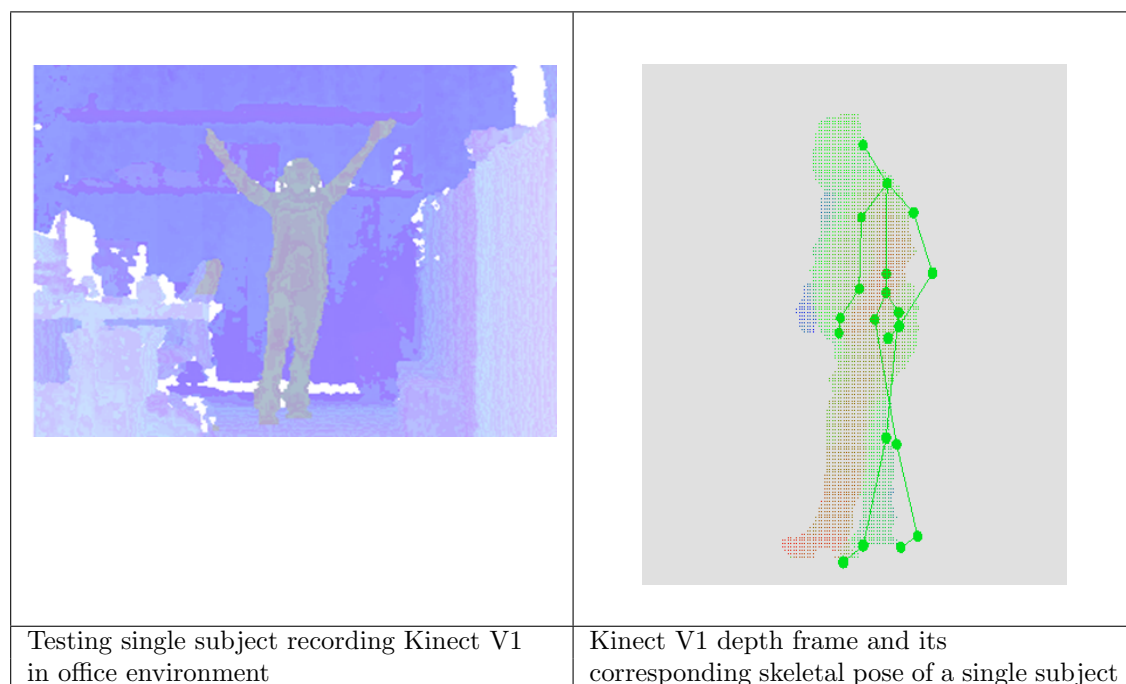


FIGURE 3.25: Single subject using Kinect V1

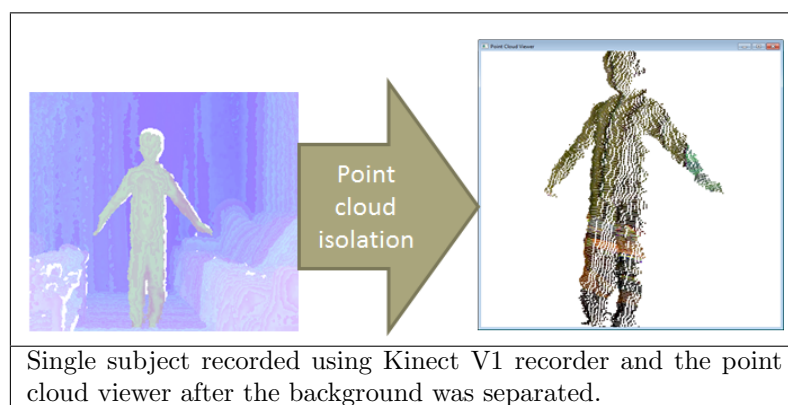


FIGURE 3.26: Single subject Kinect V1 and point cloud viewer

3.12 Kinect 2 Recorder

A Kinect 2 recorder has been developed which captures and stores the recorded data at the rate of 30fps with background removal being done using a special bit-map returned by the Kinect SDK while recording. Unlike the Kinect 1 recorded, the background was removed during the recording phase to reduce the space requirements as the image resolution is much higher for V2. The Kinect 2 recorder has been tested and it detects single and multiple people. Following are some screenshots of the frames captured using the Kinect2 recorder described in Figure 3.27.

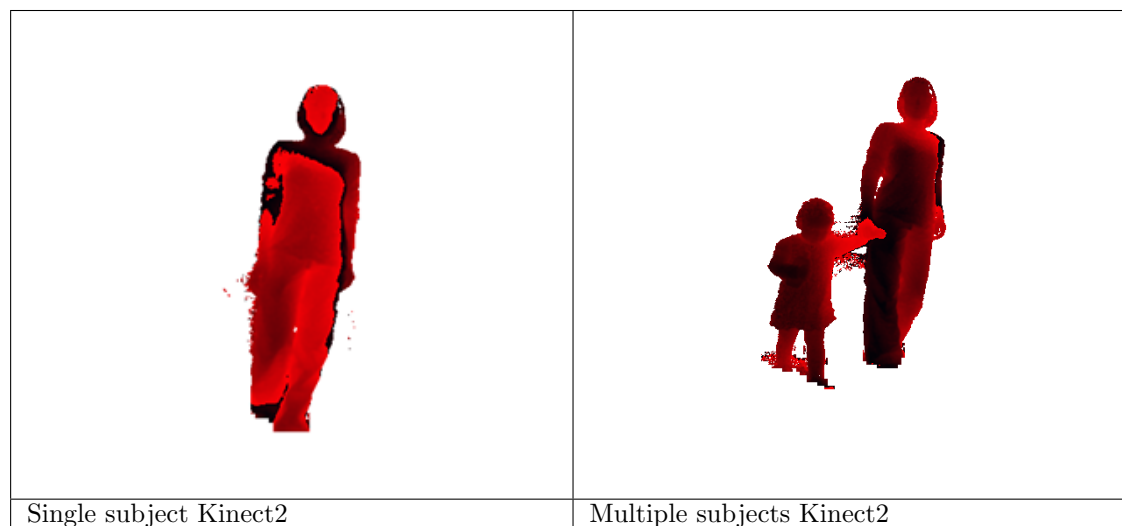


FIGURE 3.27: Depth images using Kinect2

It was observed that the hands detection in near mode is not good sometimes but in far mode is quite reliable which is described in Figure 3.28.

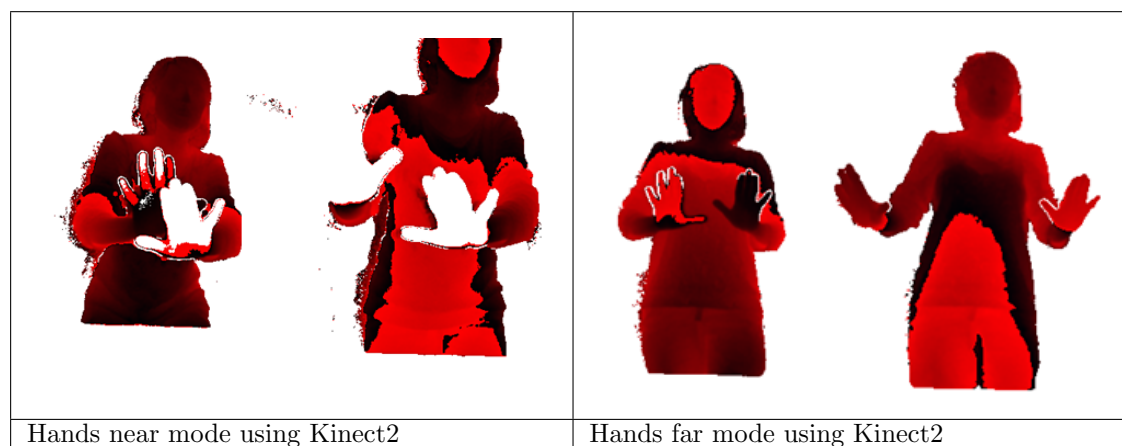


FIGURE 3.28: Near and far modes using Kinect2

3.13 The Kinematics Model

In human skeletal tracking, Kinect SDK provides 3D position of 20 joints and limb ends but the positions are observed to oscillate for no reason (i.e. even when the subject is perfectly still). Also when the subject moves rapidly, quite garbled positions are returned by the skeletal SDK. As a first step towards addressing these garbled results, a new graphical gait visualization and analysis tool called GLSKEL has been developed within this work. The tool has an interactive measurement interface that is useful for getting joint parameters for individual frames of the captured 3D frames. Although interactive measurement does not scale to allow measurement of thousands of frames, it was useful in analysing selected poses and configurations. The skeletal model in GLSKEL is a tree-structured hierarchical representation of the bones and each bone in the hierarchy has a parent joint node and a child joint node. Every joint can be a parent and a child joint unless it is a leaf joint, such as hand_right, head, etc. In the skeletal hierarchy, parent joint nodes are always above the child joint node. The 20 joints estimated by Kinect 1 or Kinect 1.8 SDK are described in 3.1 and 25 joints obtained from Kinect 2 SDK in 3.2.

The data from Kinect is to fit a kinematics model.

Forward kinematics: Given the joint variables, render the skeletal model for visualising the posture defined by a given assignment of values of those variables.

Inverse kinematics: Deriving the values of the joint variables from 3D position data acquired from Kinect SDK.

The forward kinematics was implemented as a set of hierarchical transforms applied on the skeletal joints. This made it possible to control the skeleton using GUI (slider widgets) representing each joint variable to create any pose of interest. The transforms were implemented as successive rotations applied for each joint variable. The joint variables implemented are listed in Table 3.4.

Joint Variable Names used in Forward Kinematics	
left_knee_flexion	right_knee_flexion
left_hip_abduction	right_hip_abduction
left_hip_extension	right_hip_extension
left_shoulder_extension	right_shoulder_extension
left_shoulder_abduction	right_shoulder_abduction
left_elbow_extension	right_elbow_extension
neck_extension	neck_abduction
neck_transverse_rotation	waist_extension
waist_transverse_rotation	waist_abduction
left_ankle_extension	right_ankle_extension

TABLE 3.4: Joint Variables

These names of joint variables can be understood with respect to the reference planes used in biomechanics literature. Figure 3.29 shows the names of these planes. The limb rotations made in a transverse plane are named with the suffix 'transverse_rotation'. The limb rotations made in

a sagittal plane are named with suffix 'flexion' or 'extension' according as whether the rotation brings the limb backward or forward respectively. For a limb that can go both forward and backward, the naming is a matter of sign. When such a variable is named with suffix 'flexion', the backward direction is taken as positive, and when named with suffix 'extension' the forward direction is taken as positive. Likewise, the angles in the frontal plane are named with the suffix 'abduction'. Figure 3.30 shows a few extension and abduction angles associated with human gait.

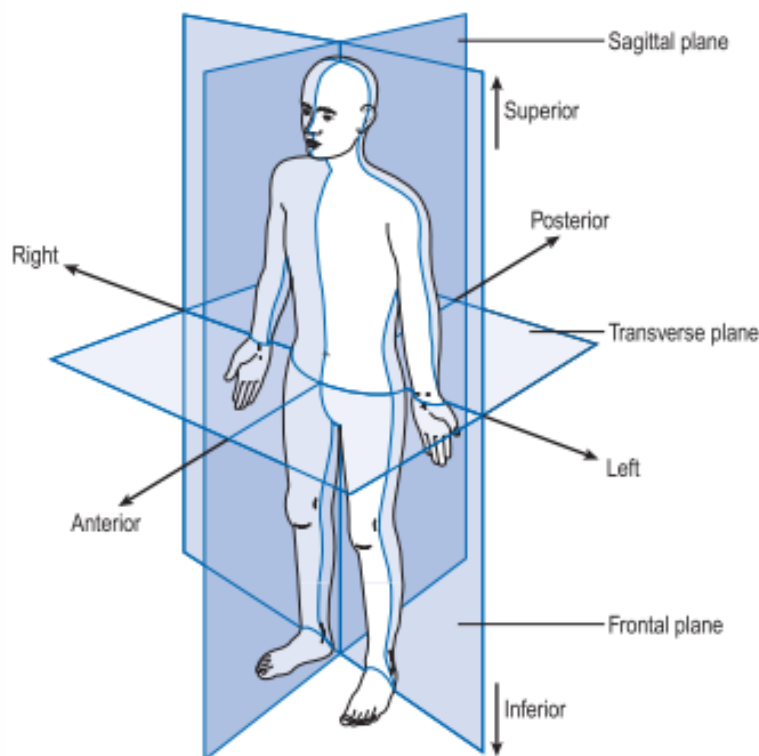


FIGURE 3.29: Reference planes used in describing joint motions

In GLSKEL, there is a panel where a realistic skeleton is rendered with specified joint angles. The captured joint angles can be displayed as 3D animation on the skeleton. Figure 3.33 shows a few screenshots of the software interface. The skeletal animation screen also supports a pose creation interface using set of sliders to control the joint angles. The screenshot (Figure 3.31) shows that tool in action. Right now the forward kinematics is not directly used along with joint angles obtained from the new algorithms. It would be easy to feed the joint variables of the new algorithm to the skeleton.

The kinematics model of an articulated human body was implemented as a hierarchy of transforms - mostly rotational transforms. The hierarchical structure arises from the fact that with reference to a chosen body frame fixed with the hip, *superior* joint motions bodily transform their *subordinate* joint positions but the sub-ordinate joint motions do not cause bodily displacement to their superior joint motions. For example, the shoulder joint motion would displace the elbow, wrist, and finger joints, but a finger or wrist joint motion does not cause displacement of the

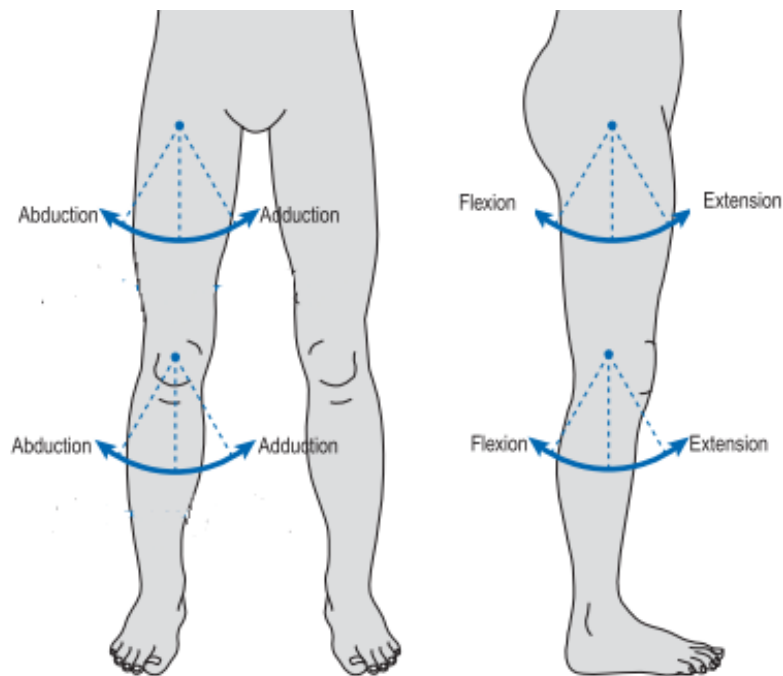


FIGURE 3.30: Abduction/adduction and flexion/extension angles

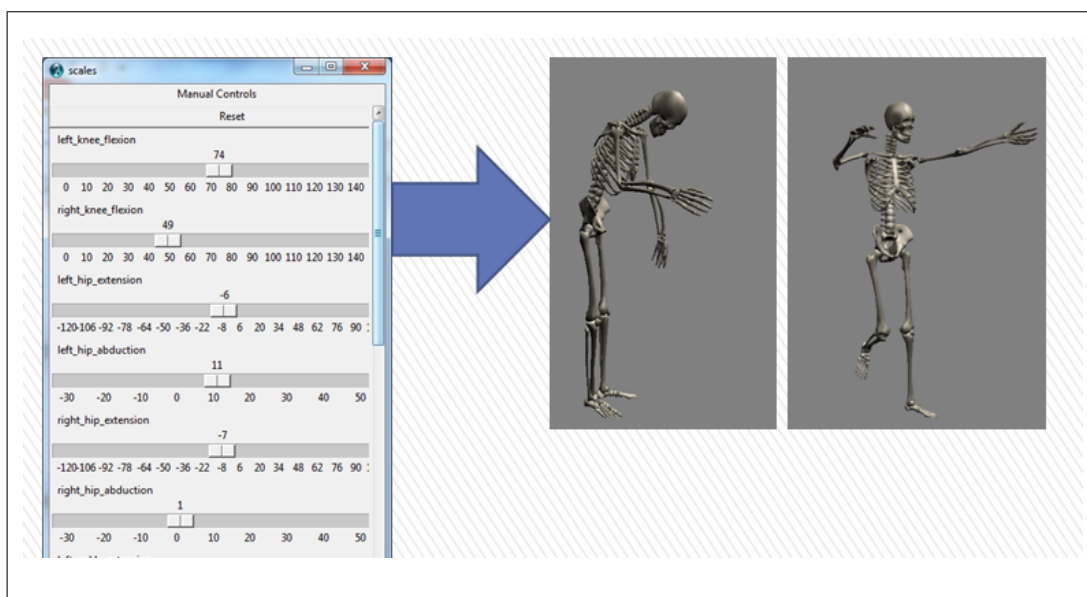


FIGURE 3.31: The Pose controller Widget in GLSKEL - Forward Kinematics of GLSKEL

shoulder position. Hence is the *hierarchy* - it is as if the subordinate joints are embedded within the coordinate system of the their relatively superior joints, but not the other way round.

In computer graphics, rigid motions are usually implemented as transformation matrices. Translation, rotation, scaling, mirroring, shearing etc. are various transformations that can be represented by such matrices. The visual model of a skeleton has two main components - meshes (usually made of triangles) to represent the shape, and matrices to implement motions. Rotation and translation transforms describe how the meshes are oriented and positioned in space at any given time. The hierarchy of motions is implemented using a stack of transformation

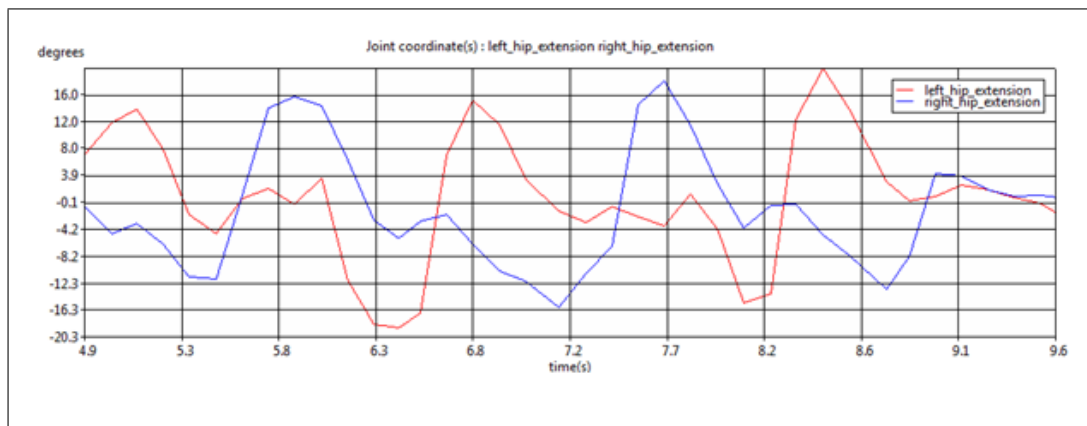


FIGURE 3.32: Left hip and right hip extension angles - Inverse Kinematics of GLSKEL

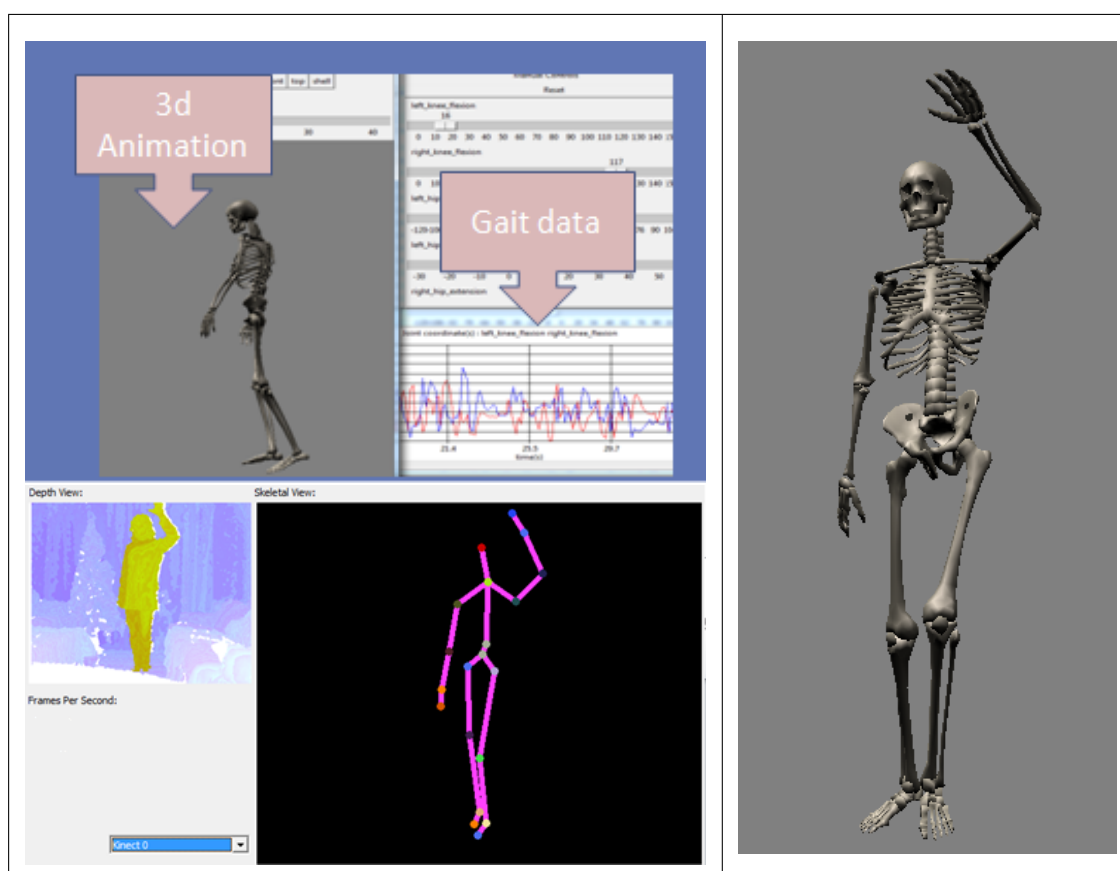


FIGURE 3.33: Screenshots from the Gait capture tools

matrices. As stated in the previous paragraph, a motion can have its subordinate and superior motions. In a computer implementation, the transformation matrix for a subordinate motion is higher up in the matrix stack, than the superior motion. This stack is maintained in such a way that the superior motion matrices apply to the subordinate limbs, but the sub-ordinate motion does not apply body parts that are outside its motional realm (e.g. ankle motion must not have any effect on the waist). The computer graphics libraries provide a way of editing the matrix stack using the *push matrix* and *pop matrix* calls. Motions are generally parenthesized (or sandwiched) between calls to *push* and *pop* matrix. Pushing a matrix creates an additional

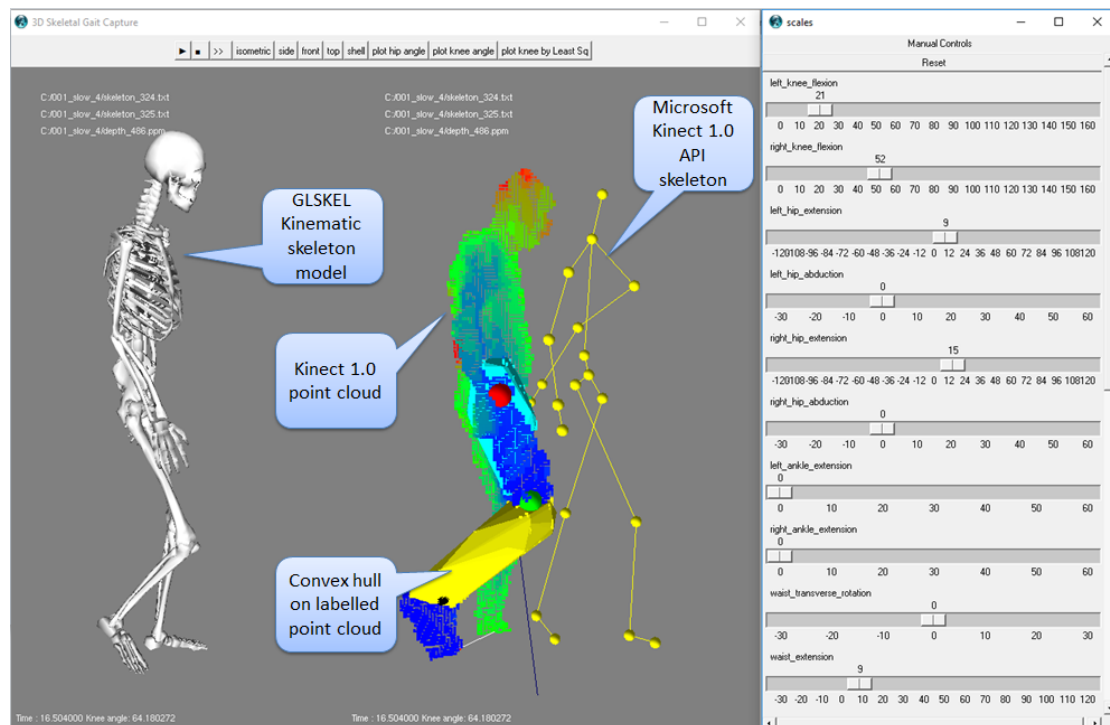


FIGURE 3.34: Kinematic model, point cloud, and SDK joints from Kinect 1.0 displayed on GLSKEL

layer of matrix on the matrix stack. This matrix is identity matrix when just pushed. At any time during the graphics scene preparation, the net transformation that applies to any mesh rendered at that point is the product of all the matrices on the stack. So as the transformation matrices corresponding to the superior motions are lower down the stack, they will apply to all the subsequent stack states that push on the top of it. The subordinate motions are withdrawn from the matrix stack once the part that were strictly under their motion are drawn. Algorithm 4 shows how the program applies the transforms while drawing an animated 3D skeleton.

Inverse Kinematics is to calculate the joint angles from the obtained 3D joint data. Figure 3.32 shows the calculated angles of left hip extension and the right hip extension using the GLSKEL inverse kinematics. It shows the plot of two complementary joint variables, one from the left hip and the other from the right hip, as obtained by the inverse kinematics calculation during a few walk cycles. Ideally these two cycles should have looked perfectly periodic and complementary (i.e. the two plots should have been identical with a phase separation of half the period). It appears that the machine learning algorithm used by Kinect V1 is optimised for speed (can process over 200 frames per second) but makes substantial compromise on accuracy. A continual flicker was noted in lower joint positions (knees, feet etc.) in the captured frames even when the subject only moved the arms and kept the lower limbs still.

Data: A set of 3D meshes representing the body parts (from Kinect)

Data: Joint motion data (from Kinect)

Result: 3D graphical rendering of the whole body

Insert a new identity matrix to the top of stack;

Draw Mesh (“Hip”);

Insert a new identity matrix to the top of stack;

waist_motion();

Draw Mesh (“Abdomen”);

Draw Mesh (“Chest”);

Draw Mesh (“Left Collar”);

Draw Mesh (“Right Collar”);

Insert a new identity matrix to the top of stack;

neck_motion();

Draw Mesh (“Neck”);

Draw Mesh (“Jaw”);

Draw Mesh (“Head”);

Remove the topmost matrix from the stack;

Insert a new identity matrix to the top of stack;

left_shoulder_motion();

Draw Mesh (“Left Shoulder”);

Insert a new identity matrix to the top of stack;

left_elbow_motion();

Draw Mesh (“Left Forearm”);

Insert a new identity matrix to the top of stack;

Draw Mesh (“Left Hand”);

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Insert a new identity matrix to the top of stack;

right_shoulder_motion();

Draw Mesh (“Right Shoulder”);

Insert a new identity matrix to the top of stack;

right_elbow_motion();

Draw Mesh (“Right Forearm”);

Insert a new identity matrix to the top of stack;

Draw Mesh (“Right Hand”);

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Insert a new identity matrix to the top of stack;

left_hip_motion();

Draw Mesh (“Left Thigh”);

Insert a new identity matrix to the top of stack;

left_knee_motion();

Draw Mesh (“Left Shin”);

Insert a new identity matrix to the top of stack;

left_ankle_motion();

Draw Mesh (“Left Foot”);

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Insert a new identity matrix to the top of stack;

right_hip_motion();

Draw Mesh (“Right Thigh”);

Insert a new identity matrix to the top of stack;

right_knee_motion();

Draw Mesh (“Right Shin”);

Insert a new identity matrix to the top of stack;

right_ankle_motion();

Draw Mesh (“Right Foot”);

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Remove the topmost matrix from the stack;

Algorithm 4: Skeletal Animation using 3D Graphics

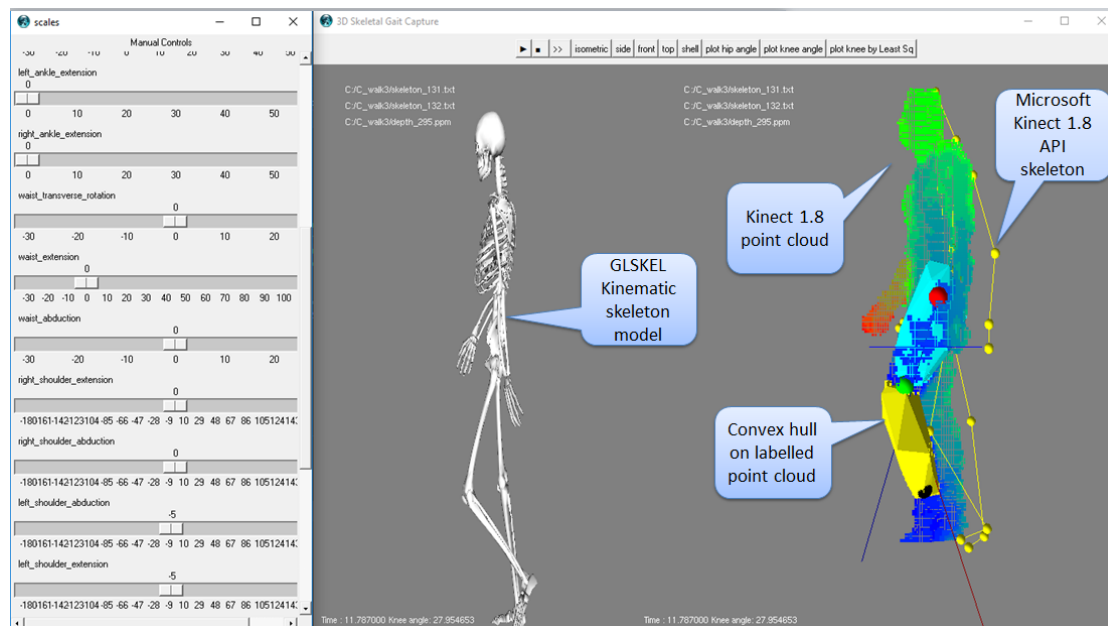


FIGURE 3.35: Kinematic model, point cloud and SDK joints from Kinect 1.8 displayed on GLSKEL

3.14 Interactive Measurement in 3D

The point cloud derived from Kinect depth frame is plotted using 3D computer graphics which allows turning around, spinning, moving, zooming etc. The 3D interaction window is shown in Figures 3.34 and 3.35. Such an intuitive interaction interface may be useful to a clinician assessing the motion or posture. A doctor or expert might pause the animation (i.e. freeze a frame) and take measurements on it. GLSKEL allows the user to measure angles and lengths on the 3D point cloud using mouse clicks. For length measurement on the depth frame, the user can choose two points on the point cloud. The points are chosen by double-clicking on the 3D view. The mouse-click corresponds to a 3D ray in model space. The point nearest to the projection centre whose neighbourhood intersects the ray is taken to be the *selected* point. Visual feedback is given to the user about the selection by enclosing the selected point in a small but clearly visible cubical block. When two points are chosen, GLSKEL not only shows the distance but also various geometric measurements like coordinates, angle with horizontal, angle with vertical, projected angle on the horizontal plane etc. Three points may be selected to explicitly measure the angle subtended at the second point by the first and the third selected point. These features are helpful for manual measurements; the automated angle estimation algorithms will be presented in chapter 5.

3.15 Validation of GLSKEL

The following validation steps were carried out in order to verify correctness of GLSKEL.

3.15.1 Validation of Skeletal Forward Kinematics

Some interactive widgets were used for testing if the joint angle kinematics (in particular the *forward* kinematics, i.e. calculation of pose from the joint angles) was implemented correctly. These interactive widgets consisted of slider buttons that could be moved continuously by dragging the mouse in pressed position. There is one such *slider* per hinge-type joint (like the elbow or knee) and two such sliders for each ball and socket joint (like shoulder and hip) to capture the additional degree of freedom. The verification task involve producing some target poses using the required joint motions invoked using the slider. The key was to observe the skeleton motion along with the slider motion to see that there were no unexpected motions (e.g. wrong joints moving, non-smooth/jerky movements, hierarchy violation etc.).

3.15.2 Validation of Inverse Kinematics

This step involved verification of calculation of joint angles from joint-positions. This was achieved through a round-trip calculation through forward kinematics. First joint angles were used in forward-kinematics to compute joint positions. These positions were passed to inverse kinematics to compute joint angles. These joint angles were compared against the original joint angles used for producing the pose in the first place. Although this merely verifies consistency between forward and inverse kinematics, this is a useful measure of correctness because forward kinematics part of the calculation was validated separately as described in the previous section.

3.15.3 Validation of Point Cloud Processing

The processing of Kinect's depth frame goes through several steps. Starting with recording frames on disk, through limb labelling, to joint angle calculation. Intermediate steps involved such steps as least squares fitting, convex hull, and principal axis calculation. Each step was tested in isolation. For example, the first part of the processing pipeline could be verified by inspecting and measuring the animated 3D view of the point cloud in an OpenGL based viewer. The convex hull could be verified by rendering the hull polyhedron along with the points. The principal axis could also be verified by visually rendering the axis as a coloured line.

3.15.4 Validation of Interactive Measurement

Point-sets with pre-defined mutual distances and mutual angles were loaded into GLSKEL. The point coordinates, and mutual distances and angles were measured within the tool and compared against the original pre-defined values. Additional glyphs were introduced at the coordinates as measured. It was visually inspected that the glyph positions coincided with the points.

3.16 Conclusion

The contribution of this chapter was to evaluate the state of the art in inexpensive 3D depth capture and evaluation of Microsoft Kinect from the perspective of accuracy. The skeletal joints captured by Kinect SDK are found to be inadequate for clinical use. The joint angles calculated from 3D joint positions are only approximate and observed to move a lot even when the subject is actually standstill. However, the point cloud accuracy as shown in Figure 3.19 is adequate for precise determination of joint angles. The forward and inverse kinematics method presented in this chapter is an exact calculation (i.e., not an approximation). Thus the real problem lies in the inaccuracy of the algorithm that computes the joint positions from the point cloud. The Microsoft SDK is known to use local features and Markov decision forest algorithm for labelling limbs. While this method is fast, it is evidently not accurate enough and it is possible to improve the accuracy by sacrificing the speed of processing. The joint coordinates from the Kinect v1.8 driver was observed to have relatively low flickering oscillations compared to v1.0. Kinect v2 joint coordinates were similar in accuracy and oscillation to 1.8. This shows that the developers at Microsoft are making some improvements to the joint recognition algorithms but they are not quite accurate enough for clinical monitoring. The following chapters 5 and 6 present new algorithms that enhance the accuracy of the limb detection using geometric and graph-theoretic methods. Kinect's internal algorithms are proprietary and unavailable for improvisation. There have been several attempts to overcome this limitation through calibration (e.g. [74]). The algorithmic development takes a similar approach towards estimating joint angles - in improving the accuracy based on raw data produced by Kinect. The main novelty of GLSKEL is that it enables continuous motion gait capture in an interactive 3D environment. Its intuitive interface can help bridge the gap between a research prototype and a product that can be used by clinical practitioners.

Chapter 4

Algorithms for Processing 3D Depth Image Data

4.1 Introduction

Chapter 3 shows how point-clouds and textured meshes are becoming a common media type with the democratisation of 3D scanning. In the coming years, as 3D scanners make their way into smart phones and tablets, such media are expected to become nearly as ubiquitous as image and video. Until the novelty wears off and algorithms become standardized for such media, it is going to be a golden age for innovations in software aimed at manipulating and analysing such media [134],[109]. A point cloud is a useful representation in its own right, but post-processing algorithms can extract substantial additional information hidden in the point-cloud. This chapter discusses in detail the algorithm of JAFKEC-G framework for gait analysis, as discussed in Chapter 5 and in the JAFKEC-U framework, as discussed in Chapter 6.

Point cloud is a very raw representation of the scanned surface. A bunch of point coordinates (i.e. x, y, z values) are all there is to it. Think of this as a raw material that has to go through a number of processing stages in order to become more meaningful and useful. The individual stages may be seen as addition of annotations on the point-cloud but a stage might as well introduce an entirely new representation of the underlying surface. It is known that the point-cloud comes from points sampled on one or more distinct surfaces of the scanned scene, so it would seem useful to group the points as belonging to individual surfaces. Once such a segmentation is done, it would be possible to label the individual segments more meaningfully than one could label individual points. The problem of segmenting the point clouds arises in the field of Computer Aided Design (CAD) where low level geometric features can be used to determine whether the surface is developable, planar, cylindrical, spherical and so on. The problem of segmentation of human limbs is more involved because the local morphology is dominated by non-mathematical features of biological and textile origin. The best way to see the following sections is to see them as description of a processing pipeline that deals with assigning meanings and representations to raw point clouds.

4.2 Defining a Topology on the Point Cloud

This section presents the key definitions of Defining a Topology on the Point Cloud as used in a geometric contexts. A brief outline would be presented here. The following texts present a more detailed discourse: [135], [43], [26], [103], [70].

Topology deals with properties of connectivity and neighbourhood between points and point-sets. The scanned points are discrete and isolated but it is possible to define a topology involving them using proximity relationships. We can say that points that are closer than a certain given distance are connected by a neighbourhood relation. Such pairwise relationships have the structure of binary relations, which is covered by a field called Graph theory. The main definitions of graph theory are presented in section 4.3. When the neighbourhood is constructed relationship as a graph, a vast array of tools and algorithms can be used available in graph theory.

There are multiple ways of constructing the graph. A simple neighbourhood relationship may be defined using a fixed sized ball. If two points are within a fixed sized ball of diameter d , an edge is connected between the points. This definition of neighbourhood requires the specification of an appropriately chosen distance threshold. An alternative approach that does not require the choice of a distance threshold is that of a *relative neighbourhood graph* in which two points are connected by an edge if the *lune* (to be elaborated next) formed by them does not contain any other point. If the distance between the two points is d , then the *lune* between the two points is the region of intersection of two spheres centred at each point with radius d . There is another criterion similar to the *relative neighbourhood graph* in which two points are connected with an edge if the smallest sphere containing the two points has no other point in it. The key difference here is that the *lune* is replaced by the sphere centred at the mid-point of the two points, with diameter equal to the distance between the points. It should be noted that one can trim the neighbourhood graph by deleting edges and nodes based on additional criteria when more information becomes available. The graph structure will prove useful in several ways and let us start with a first application.

4.3 Geometric Graphs

This section presents the key definitions of graphs as used in a geometric contexts. A brief outline would be presented here. The following texts present a more detailed discourse : [104], [54], [42], [21], [15].

A graph is a structure commonly used for representing the existence of a relation between objects. A graph is defined as follows:

DEFINITION: A graph G is a pair (V, E) where V is a set of N_v distinct objects v_0, \dots, v_{N_v-1} , and E is a set of pairs $(v_i, v_j), 0 \leq i, j \leq N_v - 1, i \neq j$. The elements of V are called vertices and the elements of E are called edges. The vertices might correspond to entities with a varied set of properties and identities but insofar as the graph structure is concerned, they are uniformly treated as distinct entities without any further ontological commitment. An edge (v_i, v_j) is denoted as $v_i v_j$. An edge $v_i v_j$ represents a binary relation between v_i and v_j . Depending on

the application, $v_i v_j$ and $v_j v_i$ may or may not mean the same edge. If the order of the pair of vertices defining an edge is significant, the edge and the graph containing it are called *directed*. If the edges are unordered pairs (i.e. if for any given edge $v_i v_j$, the re-ordering v_i and v_j is immaterial to the identity of the edge) then the edges and the graph as a whole are called *undirected*. An undirected graph represents a symmetric binary relation between vertices. The current application deals with a mutually symmetric adjacency relationships between points - therefore it is sufficient to deal with only undirected graphs for the purposes of this work.

If $v_i v_j \in E$ the vertices v_i and v_j are said to be incident on the edge. Under the same condition v_i and v_j are said to be adjacent or neighbours to each other. Two edges v_i, v_j and v_m, v_n are adjacent if either $v_i \in \{v_m, v_n\}$ or $v_j \in \{v_m, v_n\}$. In a spanning graph the edges cover all the vertices. A graph $G = (V, E)$ is called empty if $E = \phi$, and complete if E is the set of all possible edges. A graph $G' = (V', E')$ is called a sub-graph of G if $V' \subseteq V$ and $E' \subseteq E$. A sub-graph $G' = (V', E')$ of a graph $G = (V, E)$ which contains all such vertices and edges of G that are reachable through edge traversals from any vertex in V' , is called a *connected component*.

A *path* is a sequence of vertices $v_{i_0} \dots v_{i_j}$, such that each pair of consecutive vertices is an edge in the graph. We say that such a path is between v_{i_0} and v_{i_j} . A closed path is a sequence of vertices $v_{i_0} \dots v_{i_j}$ such that $v_{i_0} \dots v_{i_j}$, is a path and $v_{i_0} v_{i_j}$ is an edge in the graph. A closed path is a cycle if all its vertices are distinct. A Hamiltonian cycle is a cycle containing all vertices of the graph.

The union of graphs (V, E) and (W, F) is $(V, E) \cup (W, F) = (V \cup W, E \cup F)$.

A graph is connected if there is a path between every pair of distinct vertices. A graph is n -connected if there are n different paths between any two distinct vertices, or equivalently, if the removal of any $(n - 1)$ vertices leaves the graph connected. A graph without cycles is a forest. A tree is a connected graph without cycles. One vertex of the tree can be denoted as the root.

A graph is called a geometric graph if the vertices represent points in a Euclidean space and the edges represent some geometric relation between the vertices. The length of an edge of a geometric graph is the Euclidean distance between its two vertices. The length of a geometric graph is the sum of the lengths of all edges and the length of a path is the sum of the lengths of the edges in that path. A *hypergraph* is a generalization of graph which extends the definition of edges beyond binary relations. The edges of a hypergraph are called *hyperedges* that could be tuples of vertices (i.e. not just pairs). Following is a formal definition.

DEFINITION (based on [42]) A *hypergraph* G is a pair (V, E) where V is a non-empty finite set of distinct elements, and E is a subset of $P(V) - V, \phi$, where $P(V)$ is the power-set of the vertices. The power-set $P(V)$ (also often denoted by the notation 2^V) is the set of all possible subsets of V .

Let v_0, \dots, v_{N_v} be vertices in an Euclidean space. A *polyline* is a finite ordered sequence of line segments $v_{i_0} v_{i_1}, v_{i_1} v_{i_2}, \dots, v_{i_{n-1}} v_{i_n}$ such that $v_{i_j} = v_{i_k}$ if and only if $j = k$. A polyline of consecutive line segments $v_{i_0} v_{i_1}, \dots, v_{i_{n-1}} v_{i_n}$ is denoted by $v_{i_0} \dots v_{i_n}$. A *polygon* $v_{i_0} \dots v_{i_n}$ is a polyline $v_{i_0} \dots v_{i_n}$ that is closed by segment $v_{i_n} v_{i_0}$.

Polylines and polygons can be uniquely represented by a graph (V, E) where V is the set of vertices of that polyline or polygon and E the set of edges $v_i v_j$ that are the line segments. A polyline of N_v vertices has $N_v - 1$ edges; a polygon has N_v edges. Since every vertex is shared

by one or two edges, there is a unique ordering of edges in a polyline or polygon $v_{i_0} \dots v_{i_n}$. A triangle is the unique polygon of three non-collinear vertices, that is, there is only one set of three edges joining the three vertices.

A *closed polyhedron* is a 3D shape formed by plane polygons (called faces) such that every line segment of a polygon is shared by exactly one other polygon and no subset of polygons has the same property. An open polyhedron is a connected subset of polygons of a closed polyhedron. A polyhedron is simple if there is no pair of non-adjacent polygons sharing a point.

In the restricted case that the polyhedron has no through-passages, i.e. it is topologically equivalent to a sphere, Euler's formula applies: $N_v - N_e + N_t = 2$. Since for a closed triangulation $3N_t = 2N_e$, it follows that for a polyhedron of triangles without through-passages $N_t = 2N_v - 4$ holds. A polyhedron of triangles can be uniquely represented by a hyper-graph (V, T) where V is the set of vertices and T the set of triangles $v_i v_j v_k$ of the polyhedron. A tetrahedron is the unique polyhedron of four non-coplanar vertices, that is, there is only one set of four triangles joining the four vertices.

A simplex or k -simplex is the unique k D structure of $k + 1$ vertices not lying in a $(k - 1)$ D hyper-plane that joins its vertices by $k + 1$ $(k - 1)$ -simplices; a 1-simplex of two vertices is a line segment between these vertices. So, a 2-simplex is a triangle and a 3-simplex is a tetrahedron, and so on. A simplex represents the simplest non-degenerate closed shape formed by flat segments of one lower dimension.

The geometric graph concepts are crucial in JAFKEC because the raw data in a depth/range image comes in the form of scattered discrete points and it becomes necessary to introduce some topology on it artificially using geometric criteria. Topology describes how discrete objects are connected to each other and thereby allows one to define algorithms that can analyse global structure using local traversals. The following subsections provide definitions of operations useful for analysis of geometric graphs.

4.3.1 Closest Pairs

This sub-section presents the definition of *closest pairs* as used in a geometric contexts. A brief formal definition is given here. [92] and [12] present a more detailed discussion. DEFINITION (CLOSEST PAIRS (CP)) Let V be a set of vertices in k dimensional space. The *closest pair* in a set vertices V (with N_v elements) is the pair of vertices whose distance is shortest among all possible pairs of vertices. There can be more than one pair whose distance is equal to the closest distance, in which case one of the tied pairs can be arbitrarily be chosen as the closest pair.

In a brute force implementation one might think of choosing the shortest from among the $N_v \times N_v$ possible pairs. However with a divide-and-conquer algorithm, the closest pair can be found in $O(N_v \log N_v)$ time. Following is a rough sketch of the divide-and-conquer algorithm:

1. Sort the points along some spatial direction.
2. Choose a middle (median) point along the sorted direction. Divide the array into two parts as split by the hyperplane perpendicular to the sorting direction and passing through the median point.

3. Recursively find the closest pair in both split parts. Compare the two closest pairs from the two split half-spaces and return the closer one.

The closest pair algorithm may be used as a sub-routine in order to finding a connectivity graph, by consecutively adding edges corresponding to closest pairs from the remaining vertices.

4.3.2 Nearest Neighbours Graph

This section presents the key definitions of Nearest Neighbours Graph as used in a geometric contexts. A brief outline would be presented here. The following texts present a more detailed discourse : [92],[167],[41]

DEFINITION (NEAREST NEIGHBOURS GRAPH (NNG)) Let V be N_v vertices in k dimensions. The Nearest Neighbours Graph of V is the graph (V, E) with E as the edges that join each vertex v_i with one v_j satisfying $d(v_i, v_j) \leq d(v_i, v_k)$ for all $v_k \neq v_i$.

Note that the Nearest Neighbours Graph is not unique if there is more than one v_j such that $d(v_i, v_j) \leq d(v_i, v_k)$ for all $v_k \neq v_j$. The Nearest Neighbours Graph is generally disconnected. Since all the pairs of vertices that are each other's nearest neighbour contain the pairs with the smallest distance of all, $CP \subseteq NNG$. The Nearest Neighbours Graph in k dimensions can be constructed in $O(N_v(\log N_v)^{k-1})$ time [12], provided that the maximum number of vertices joined to each vertex is independent of N_v .

4.3.3 Euclidean Minimum Spanning Tree

This section presents a brief definition of Euclidean Minimum Spanning Tree as used in geometric contexts. The following texts present the idea in greater detail: [159],[52]

DEFINITION (EUCLIDEAN MINIMUM SPANNING TREE (EMST)) Let V be a set of vertices in k dimensions. A Euclidean Minimum Spanning Tree of V is a spanning tree (i.e. a tree covering all vertices) of minimum length.

The EMST of a set of vertices is generally not unique. In an EMST, each vertex must be joined to its nearest vertex and thus $NNG \subseteq EMST$. In the event that NNG and EMST are not unique, this can be interpreted as - there exists an NNG that is a subset of the EMST. Each NNG actually is a minimum spanning forest (i.e. there is no requirement of being a connected structure). In the special cases where the NNG is connected it is equivalent to an EMST. EMST can be computed in $O(N_v \log N_v)$ time [159], which is optimal. EMST is a bit different from the minimum spanning tree algorithms for graphs. On a graph-based MST. In a graph-based MST algorithm, the input is a graph and its edge weights. By contrast, in EMST the input is a set of points in space and a distance function to compute the distance between any pair. If a graph is made by putting edges between nearby points, graph based MST algorithms like Prim's and Kruskal's algorithm ([34], [167]) can be used for approximating an EMST.

4.3.4 Infinite Strip Graph

This section presents a brief definition of the Infinite Strip Graph as used in a geometric problems. [41] provides a more detailed discussion on the topic. The Infinite Strip Graph (∞ -SG) joins two vertices if and only if the associated *infinite strip* is empty [41].

DEFINITION (INFINITE STRIP) Let v_1 and v_2 be two vertices in k dimensional space. The infinite strip is the open space bounded by two parallel $(k - 1)$ dimensional planes through v_1 and v_2 perpendicular to the direction joining v_1 and v_2 .

This is yet another definition of neighbourhood graph, although in this case the neighbourhood size is infinite. This and the other neighbourhood graph definitions specify different criteria for which pairs of vertices are to have edges between them (or in other words which pairs are not to have edges between them). The objective of such criteria is to not end up with too many edges and yet capturing the essential structure of the connectivity between the points. For infinite strip neighbourhood, the neighbour may well be infinite but the criterion still implies locality in the sense that the neighbourhood depends only on two vertices.

If no two infinite strips for different pairs of vertices coincide, the Euclidean Minimum Spanning Tree must join a pair of vertices whose infinite strip is empty. So in non-degenerate cases ∞ -SG \subseteq EMST. Therefore, one can examine each of the $N_v - 1$ edges in the Euclidean Minimum Spanning Tree and check if any vertex lies in the corresponding infinite strip. So the Infinite Strip Graph can be constructed in $O(N_v^2)$ time.

4.3.5 Sphere of Influence Graph

This section presents a brief definition of the *Sphere of Influence* graph in the context of geometric problems. [41] and [184] provide a more detailed discussion of the idea.

DEFINITION (SPHERE OF INFLUENCE GRAPH (SIG)) Let V be a set of vertices in k D. For each vertex v_i , let r_{v_i} be the distance to its closest vertex. The Sphere of Influence Graph joins two vertices v_1 and v_2 , if and only if the sphere centered at v_1 with radius r_{v_1} , and the sphere centered at v_2 with radius r_{v_2} , intersect in more than one point.

The SIG may be a disconnected graph. Each vertex in a SIG is joined with its nearest neighbour, so that $\text{NNG} \subseteq \text{SIG}$. The Sphere of Influence Graph can be constructed optimally in $O(N_v \log N_v)$ time [184].

So, SIG is yet another approach of inducing a topology to a set of scattered points.

4.3.6 Relative Neighbourhood Graph

This section presents a brief definition of the *Relative Neighbourhood Graph*. More detailed description may be found in [105], [41], [183], [86]. The Relative Neighbourhood Graph (RNG) joins two vertices if and only if their *relative neighbourhood* (as defined below) is empty.

DEFINITION (RELATIVE NEIGHBOURHOOD (RN)) Let v_1, v_2 be two vertices in k dimensional space. The associated relative neighbourhood is the interior of the intersection of the two k -D balls centered at v_1 and v_2 with radius $d(v_1, v_2)$.

Two vertices v_1 and v_2 with an empty RN are considered *relatively close*, i.e. if $d(v_1, v_2) \leq \max(d(v_1, v_i), d(v_2, v_i))$, for all $v_i \neq v_1, v_2$. The RNG can be constructed in $O(N_v \log N_v)$ time for 2D points [86]. In the general number of dimensions, it can be computed in $O(N_v^3)$ time [183].

4.3.7 Gabriel Graph

This section presents a brief definition of the Gabriel Graph. More detailed description may be found in [59], [105], and [116]. The Gabriel Graph (GG), named after its originator [59], joins two vertices if and only if their *Gabriel neighbourhood* (as defined below) is empty.

DEFINITION (GABRIEL NEIGHBOURHOOD) Let v_1, v_2 be two vertices in k dimensions. The Gabriel neighbourhood associated with v_1 and v_2 is the interior of the smallest k dimensional ball touching v_1 and v_2 .

The Gabriel neighbourhood sphere has radius $d(v_1, v_2)/2$. Since the Gabriel neighbourhood is a spatial region that is fully enclosed within the RNG neighbourhood, it is empty when the latter is empty. So $\text{RNG} \subseteq \text{GG}$. The Gabriel was originally used in analysis of geographic data but is a general tool in the context of inducing topology to a set of scattered points. The Gabriel Graph in 2D can be computed optimally in $O(N_v \log N_v)$ (as in [116]).

4.3.8 Convex Hull

This section presents a brief definition of the convex hull. [167] provides a detailed discussion of the subject. Barber et al. [10] presents an efficient convex hull algorithm. The convex hull of a set of points is the minimal convex region that encloses a given set of scattered points.

DEFINITION (CONVEX HULL (CH)) Suppose that V is a set of vertices in k dimensional space. The Convex Hull of V is the hypergraph (V, F) where F is the set of $(k - 1)$ -simplices $v_{i_0} \dots v_{i_k}$ such that no other vertices lie in the half-space on one side of the hyper-plane through $v_{i_0} \dots v_{i_k}$, nor inside nor on simplex $v_{i_0} \dots v_{i_k}$. That is, all of the other vertices lie entirely on one side of the hyper-plane containing each simplex.

The Convex Hull is the smallest convex polyhedron containing all the vertices of V . [140] presents the optimal 2-D and 3-D Convex Hull algorithms that can construct the structure in $O(N_v \log N_v)$ time.

This concludes the list of sub-sections that present methods from the literature for inducing a topology on a set of scattered points. Once a neighbourhood structure is inferred from points in a depth/range image, further calculations may be carried out on it for local and global analysis towards feature detection.

4.3.9 Geometric Graph using a Distance Threshold

In the preceding subsections, the geometric graph induction algorithms were scale-free - i.e. the same algorithm could work irrespective of the scale of distances involved. Such algorithms have

great intellectual appeal because it does not involve any problem-specific choice of parameters. It is however possible to have a simpler graph induction algorithm if one allows the algorithm to be parametrized by a problem-specific factor. Following is one such algorithm in which edges are connected based on a specified distance threshold. If the distance between two 3D points in the point-cloud is less than a distance D_{max} then the two points are said to be neighbours (i.e. a graph edge is formed between them).

Algorithm for Geometric Graph creation using distance threshold;

Data: Set of 3D points P

Result: A graph G derived from distance relationship

Create an index for fast neighbourhood query;

Create an empty graph G with $V = \Phi$ and $E = \Phi$;

foreach point p in P **do**

 Create a node $v(p)$ with geometric position p ;

$V \leftarrow V \cup \{v(p)\}$;

end

foreach vertex $v(p)$ in V **do**

foreach nearest N_{max} neighbouring points q of p **do**

if $|p - q| < D_{max}$ **then**

 Create edge $e_{pq} = (v(p), v(q))$;

$E \leftarrow E \cup \{e_{pq}\}$;

$Adj(v(p)) \leftarrow Adj(v(p)) \cup \{v(q)\}$;

$Adj(v(q)) \leftarrow Adj(v(q)) \cup \{v(p)\}$;

end

end

end

Algorithm 5: Distance Threshold based Geometric Graph Creation

4.4 Estimating Normals at the Cloud Points

This section describes the method of estimating normals from a local neighbourhood of scattered points. The following texts present a more detailed discourse: [65],[32], [122], [143].

It is useful to compute local surface properties (like normal, curvature, etc.) from the point cloud because these properties can serve as features to feed into classification algorithms.

A fundamental local property of a surface is its normal. Normal is the spatial direction that is perpendicular to the surface. Smooth surfaces can be flat or curvy and even when they are curvy, they look flat when one looks closely enough into any local neighbourhood on the surface. That is, when one zooms enough on smooth differentiable surfaces, the curved nature fades (or flattens) away and all that remains is (locally) geometrically equivalent to a plane that would be tangential to the surface at that point. The direction perpendicular to this locally tangential plane is the normal to the surface at that point. Of course the normal keeps changing as one moves along the surface (unless the surface is actually planar). At a given point the rate at which the normal direction changes is called its *curvature* - which is another local property of interest. The higher the curvature at a point the more curvy/contorted is the surface is at that point. The lower the curvature magnitude, the flatter and more plane-like is the surface at that point.

When the surface is represented by a point cloud (as is the case here), the normal at a given point can not be computed just from the point itself but it can be estimated from the point and its neighbouring points. Recall that a smooth surface is locally planar (i.e. if one zooms close enough to it, the surface looks like a flat plane). Due to this phenomenon the earth appears flat from the common everyday experience despite its spherical shape. This means that if a small neighbourhood of points is sampled from a point-cloud, these points should approximately belong to a plane that could serve as a localised approximation of the surface and hence the normal to this plane would represent the normal to the surface.

The neighbourhood graph helps in this computation by giving the points in the neighbourhood of the point in question. Next comes the sub-problem of finding a plane that approximately contains a given set of points. It is an elementary result from coordinate geometry that three points exactly determine a plane in 3D space. So, one possible approach would be to take three-point-planes obtained from each three-at-a-time choice of points in the neighbourhood and then take their average plane as the estimate. This would involve nC_3 three-point-plane calculations followed by the averaging calculation. This is feasible solution but has the following disadvantages : (i) it is computationally much more expensive than it needs to be (ii) it doesn't make any guarantee on optimality of the solution.

This solution is usually sub-optimal, especially in the presence of noise. There is a less expensive and more robust way of fitting an approximate plane through n given points, using what is called the least squares fitting method.

The least squares method is a general method of solving overdetermined systems of equations, i.e. where there are more equations than variables. When faced with an overdetermined system of equations (i.e. equations of the form $f_1 = 0, f_2 = 0, \dots, f_n = 0$, where much fewer than n variables are involved in the equations, instead of the likely futile task of trying to find the variable assignments that make all of f_1, f_2, f_3 etc. vanish, it is feasible to find a variable assignment that brings an appropriate aggregate of $f_1, f_2, f_3, \dots, f_n$ as close to zero as possible.

The popularly chosen aggregate is the sum of squares, because squaring cancels (i.e. makes positive) the sign of individual terms, so that by bringing the aggregate close to zero couldn't possibly mean that terms with opposite signs conspire to bring the aggregate close to zero but each term is individually substantially different from zero. Taking the absolute value function instead of squaring would also have dropped the sign, but the absolute-value function introduces non-differentiability in the aggregate function, thereby making it unsuitable to most mathematical function minimisation procedures. A higher even-power (e.g. the 4th power) could potentially be used instead of the square but the square is simpler and square's derivative is linear (which in turn allows for efficient computational solution procedure) and the squared error term has solid theoretical underpinnings (e.g. Gauss proved that least squares solution is equivalent to maximum likelihood estimation if the fit-error is assumed to be normally distributed).

So, back to the problem of fitting a plane through a set of points. There is an easy but slightly fragile way and a complicated but more robust way. The easy way is to assume that the plane is described in the form where z is expressed as a function of x and y , say $f(x, y) = Ax + By + C$. Thus the equations are: $Ax_1 + By_1 + C - z_1 = 0$, $Ax_2 + By_2 + C - z_2 = 0$, \dots , $Ax_n + By_n + C - z_n = 0$ (and please note that A , B , and C are the variables to solve for). Thus it is possible to express the aggregate error function as the sum of squares of the left-hand-side expressions of these equations. This function is quadratic in A , B and C , which means that it is a parabolic hypersurface in a 4d space. When the function is quadratic in its variables it has exactly one optimum and this can be found by solving a system of linear equations. In this case the system of equations can be obtained by a simple calculus result - that the derivative vanishes at the optimum point. So if partial derivatives are taken of the sum of squared error function and equated to zero, then one gets exactly three linear equations which one can solve to obtain the best fit A , B , and C . This works but has the disadvantage that when the points are approximately on a vertical plane (i.e. approximately on a plane perpendicular to the x-y plane), the assumption that z can be expressed as $Ax + By + C$ breaks down. This leads to numerical instability and potentially singularity of the system of equations. This difficulty is merely due to singularity of the parametrization and not an intrinsic geometric singularity. This can be avoided using what is called geometric least squares or orthogonal regression. In this method, instead of the taking the difference between the modelled and datum z values in the error function, one could take the orthogonal distance of the points from the plane as the error terms. In essence, the geometric least squares process involves finding the plane for which the sum of squared distances of the points from the plane is as low as possible. Let's say the plane is defined by the equation $a * x + b * y + c * z + d = 0$, where the normal (a, b, c) is normalized (i.e. $a^2 + b^2 + c^2 = 1$), the distance of a point (x_i, y_i, z_i) from the plane is given by :

$$D(x_i, y_i, z_i) = (a * x_i + b * y_i + c * z_i + d)$$

The sum of squares of all such distances, the sum being taken over each neighbourhood point is an error function that is needed to be minimized in order to find the best a, b, c , and d that fits the points. Given a set of n points, the squared error function is the sum

$$E = \sum_{i=1}^n (D(x_i, y_i, z_i))^2$$

Derivatives vanish at the stationary point, therefore:

$$\frac{\partial E}{\partial a} = \sum_{i=1}^n 2x_i(ax_i + by_i + cz_i - d) = 0$$

$$\frac{\partial E}{\partial b} = \sum_{i=1}^n 2y_i(ax_i + by_i + cz_i - d) = 0$$

$$\frac{\partial E}{\partial c} = \sum_{i=1}^n 2z_i(ax_i + by_i + cz_i - d) = 0$$

$$\frac{\partial E}{\partial d} = \sum_{i=1}^n -(ax_i + by_i + cz_i - d) = 0$$

The last of these equations gives:

$$d = a \frac{\sum_{i=1}^n x_i}{n} + b \frac{\sum_{i=1}^n y_i}{n} + c \frac{\sum_{i=1}^n z_i}{n}$$

This geometrically means that the best fit plane passes through the centroid of the points. If one solves the first three equations in a coordinate system centred at the centroid, it can be seen that the constant term d vanishes in that system. Thus if the new coordinates are called x', y', z' (i.e. with the definitions $x'_i = x_i - \frac{\sum_{k=1}^n x_k}{n}$, $y'_i = y_i - \frac{\sum_{k=1}^n y_k}{n}$, $z'_i = z_i - \frac{\sum_{k=1}^n z_k}{n}$), the following set of equations are obtained.

$$\sum_{i=1}^n 2x'_i(ax'_i + by'_i + cz'_i) = 0$$

$$\sum_{i=1}^n 2y'_i(ax'_i + by'_i + cz'_i) = 0$$

$$\sum_{i=1}^n 2z'_i(ax'_i + by'_i + cz'_i) = 0$$

or

$$a \sum_{i=1}^n (x'_i)^2 + b \sum_{i=1}^n x'_i y'_i + c \sum_{i=1}^n x'_i z'_i = 0$$

$$a \sum_{i=1}^n x'_i y'_i + b \sum_{i=1}^n (y'_i)^2 + c \sum_{i=1}^n y'_i z'_i = 0$$

$$a \sum_{i=1}^n x'_i z'_i + b \sum_{i=1}^n y'_i z'_i + c \sum_{i=1}^n (z'_i)^2 = 0$$

This has a trivial solution $a = 0, b = 0, c = 0$ but that solution is useless and it violates the requirement that (a, b, c) is a normalized vector. On taking into account the normalisation condition $a^2 + b^2 + c^2 = 1$, this reduces to an Eigenvalue problem for the following matrix.

$$\begin{pmatrix} \sum_{i=1}^n (x'_i)^2 & \sum_{i=1}^n x'_i y'_i & \sum_{i=1}^n x'_i z'_i \\ \sum_{i=1}^n x'_i y'_i & \sum_{i=1}^n (y'_i)^2 & \sum_{i=1}^n y'_i z'_i \\ \sum_{i=1}^n x'_i z'_i & \sum_{i=1}^n y'_i z'_i & \sum_{i=1}^n (z'_i)^2 \end{pmatrix}$$

The Eigenvalues correspond to the sum of squares of distance and the corresponding Eigenvectors are the associated values of a , b , and c . The least squares solution then corresponds to the smallest Eigenvalue. The above method is a 3-dimensional special case of the general k -dimensional method given in [156]. A pseudo-code form of this calculation is given in Algorithm 3 of chapter 3. In that chapter the best-fit plane fitted to a point-cloud scanned on a planar board was used as reference for estimation of Kinect's accuracy.

4.5 Estimating Curvatures at the Cloud Points

This section presents the key definitions of Estimating Curvatures at the Cloud Points as used in a geometric contexts. A brief outline would be presented here. The following texts present a more detailed discourse: [122],[143]

Curvature is a measure of how curved or contorted the surface is at a given point. It is useful as a local feature for classifying the point clouds. Curvature at a point on a surface is the rate at which the normal direction changes as one moves away from the point. This definition implies that curvature is directional, in that it would depend on the direction chosen in moving away from the point. Among all the possible directions there is a specific direction along which the curvature is the highest. This is called the principal curvature direction - it arises from the intrinsic geometry of the surface and does not depend on the coordinates or parametrization of the surface. The principal curvature direction, along with the curvature value can serve as a useful feature in classifying the surface points. There are actually two directions called principal curvature directions, one direction along which the surface curves the fastest (i.e. the normal vector changes at highest rate) and another direction perpendicular to it. Patterns of surface curvature is classified into (A) *ellipsoid-like* - for which the surface curves away on the same side of the tangent plane along both principal directions, (B) *cylinder-like* for which the normal does not change locally along one of the principal directions and (C) *hyperboloid-like* - for which the surface curves away on opposite sides of the tangent plane along the two principal directions. Such classification is characterised by the product of the two signed principal curvature values and this product is called Gaussian curvature. Curvature values are local properties and could depend on small features like creases and kinks, however, if one applies a smoothing filter on the normals prior to curvature calculation, that helps with reducing the overly localised curvature variations. There are two ways of estimating curvature - (i) estimation by discrete points and (ii) using multivariate calculus on an approximated algebraic surface.

4.6 Geodesic Distance

This section presents the key definitions of Geodesic Distance as used in geometric contexts. A brief outline would be presented here. The following texts present a more detailed discourse: [168],[181]

Geodesic lines represent shortest paths between points on a surface such that the paths are confined to the surface. Calculus of variations provides a toolkit for calculation of geodesics on

analytic surfaces, but such a treatment would require the entire surface to be approximated by a single differentiable analytic form. This would be hard to achieve for a discrete definition derived from point-clouds, so the approach that has been taken is that of approximating the geodesics as shortest paths on the surface-like graphs discussed in the section 4.2.

The nodes in such graphs are points sampled on the imaged surface, and as such the distances between sampled points may be annotated on the graph edges as weights to allow shortest-path algorithms to work. The shortest paths on such graphs approximately correspond to the geodesic paths on the surface. Since at each node the directional resolution is limited, it can be argued that the approximation is too crude to be considered equivalent to the true surface geodesics. Irrespective of whether the graph-based geodesics correspond to true surface geodesics, it does provide a very useful feature for limb classification. Algorithm 6 presents the pseudocode form of the geodesic distance labelling process.

Geodesic distance labelling Algorithm;

Data: Geometric graph G

Data: A set of distinguished vertices $V_s \subset V$ of G , called the source vertices

Result: A distance labelling $D(v(p))$ of each vertex $v(p)$

```

foreach vertex  $v(p)$  in  $V$  do
  if  $v(p) \in V_s$  then
    | Initialise  $D(v(p))$  to 0
  end
  else if  $Adj(v(p)) \cap V_s \neq \Phi$  then
    | Initialise  $D(v(p))$  to  $\inf_q \{|p - q| : v(q) \in Adj(v(p)) \cap V_s\}$ ;
  end
  else
    | Initialise  $D(v)$  to  $\infty$ ;
  end
end
while  $V/V_s \neq \Phi$  do
  foreach  $v(p) \in \{v(p) : D(v(p)) = \inf_q \{D(v(q)) : v(q) \in V/V_s\}$  do
    | foreach  $v(q) \in Adj(v(p))$  do
      |  $D(v(q)) \leftarrow D(v(p)) + |p - q|$ ;
    | end
    |  $V_s \leftarrow V_s \cup v(p)$ ;
  | end
end

```

Algorithm 6: Geodesic distance labelling

4.7 Geodesic Centroid on Geometric Graph

As previously discussed, graph-based geodesics are used for defining features on the scanned points. Such features are defined with reference to certain special points. For example, the lowest and highest points in the vertical direction can serve as start point for geodesic distance

labelling. Similar special points can serve as start and end points for Graph-Cut and Max-Flow algorithms. It is often useful to find the most central point in a sub-region of the point-cloud to serve as a similar special point for an algorithm. The geometric centroid of the scanned points is not useful in this context because the geometric centroid could lie outside the region for a non-convex point-set. The centroid we need is a central point on the surface segment in question. The notion of centrality in a graph needs to be clarified. Following are two ways of defining a central node.

- Two extremities of the minimum spanning tree are defined using two traversals - start with an arbitrary node and go to the farthest reachable node. That would be the first extremity. Next start from the first extremity and go to the farthest reachable node and that would be the second extremity. The point that lies halfway on the shortest path between the two extremities is a central node.
- A central node could be defined by finding the root node that best balances the minimum spanning tree of the graph.

A pseudocode implementation based on the first definition is given in Algorithm 7.

Algorithm for Geodesic Centroid;

Data: Geometric graph G

Result: A graph based geodesic centroid of G

Initialise E_{MST} to Φ ;

foreach vertex $v(p) \in V(G)$ **do**

 | $root(v(p)) \leftarrow v(p)$;

end

foreach edge $e_{pq} \in E(G)$ in increasing order of $|p - q|$ **do**

 | **if** $root^*(v(p)) \neq root^*(v(q))$ **then**

 | $E_{MST} \leftarrow E_{MST} \cup e_{pq}$;

 | $root(v(p)) \leftarrow root(v(q))$;

 | **end**

end

Now there is a spanning tree represented as a set of edges E_{MST} ;

Choose an arbitrary vertex $v(p) \in V(G)$; Find the vertex $v(q)$ farthest from $v(p)$ through edges of E_{MST} ;

Find the vertex $v(r)$ farthest from $v(q)$ through edges of E_{MST} ;

Find the vertex $v(r)$ that is halfway between $v(r)$ and $v(q)$ through edges of E_{MST} ;

Return $v(r)$ as the graph-based geodesic centroid

Algorithm 7: Algorithm for Geodesic Centroid of a Geometric Graph

In algorithm 7, $root^*(v(p))$ refers to the transitive terminus through the $root$ function. Intuitively, it is the vertex you converge on if you keep applying the $root$ function iteratively from $v(p)$. It may be written as a recursive function as follows :

The $root^*$ value for a vertex identifies its connected component in the partially constructed MST graph. So the **if** statement in the above algorithm checks if the edge e_{pq} connects two distinct components of the partially constructed MST graphs.

The geodesic distance between two points in the point cloud is conflated with the graph distance between their corresponding vertices in the geometric graph. Thus, equipped with the centroid

Function $root^*$;
Data: A vertex $v(p)$ of the geometric graph G
Result: The transitive terminus of $v(p)$
if $root(v(p)) = v(p)$ **then**
 | return $v(p)$
end
else
 | return $root^*(root(v(p)))$
end

Algorithm 8: Function to compute terminal $root$ of a vertex

and distance function, one can execute a clustering algorithm on the lines of k -means clustering in Euclidean space. The k-means clustering algorithm is described in algorithm 9.

Graph based K-means clustering;
Data: A geometric graph G , and the number k
Result: A partition of the vertices of G
Randomly choose k distinct vertices from G and call it the set CC ;
Create an empty set and call it $CC_{previous}$; **while** $CC \neq CC_{previous}$ **do**
 | $CC_{previous} \leftarrow CC$;
 | Assign to each vertex of G , $cluster(v(p)) \leftarrow$ Geodesically nearest vertex in CC ;
 | Divide the vertices of G into k partitions so that each partition is identified by equality of $cluster$ assignment;
 | Recompute the graph based centroid of each subgraph formed by vertices in each partition;
 | Assign $CC \leftarrow$ the set of centroids of the partitioned subgraphs;
end

Algorithm 9: Algorithm to segment the point cloud into geodesic clusters

Algorithm 7 is a crucial tool for finding a central location that is guaranteed to stay on the surface. Contrast this with the euclidean centroid that would generally lie away from the surface. This can be used in a surface-constrained clustering algorithm. If the k-means clustering were to be applied on the points based on euclidean distance and Euclidean centroids, points that are distant in geodesic sense, but closer in space would tend to stick together in the same cluster. Since it is desirable to segment the surface into segments, it is not desirable to have two topologically distant regions to form the same cluster due to proximity in euclidean space. For example, when the cup is close to the mouth, it would become a challenging task for the labelling stage if the cup and the mouth formed a cluster. It is essentially a k-means clustering algorithm based on distances measured along the surface.

4.8 Geodesic or Graph Based Clusters

This section presents the key definitions of Graph Theoretic Clusters as used in a geometric contexts. A brief outline would be presented here. The following texts present a more detailed discourse: [85],[195], [8]

Clustering is the process of grouping a set of points into several groups based on some similarity or proximity criterion. A common clustering algorithm is the k-means algorithm. In k-means algorithm, the data-points are grouped into clusters by iteratively re-defining the cluster or group

assignments. In each step of the iteration the cluster means are recomputed and the data-points are re-assigned to the nearest re-computed mean. The initial values of the cluster means are set arbitrarily and after a number of iterations the assignments converge a certain best clustered state. It can be proved that the convergence is always reached.

The two steps that get iterated alternately are called the *assignment step* and the *update step*. In the assignment step each data-point is assigned as group member of its nearest mean and the update step calculates the new mean for the data points assigned to the same group.

In the k-means algorithm, the 'mean' is a point in the space of the data points and the mean is calculated as the centroid of the data-points. Accordingly, the assignments are made based on the lowest Euclidean distance criterion. We can generalize the notion of 'mean' in k-means algorithm so that it represents something different from the centroid of its cluster members. The 'mean' need not necessarily be a point. For example, one could say that a cluster centre represents a line in space of the data points and a point is assigned to the line to which its orthogonal distance is lowest. Likewise the line itself is the best-fit line of its cluster members. In that case the algorithm would be as described below.

Initially k lines are initialised randomly or according to a fairly arbitrary procedure. Let the set of these lines be called L . Carry out the following two steps repeatedly until the assignment step stops changing the point-to-line assignments.

Assignment step: Assign each point to the line that is closest to the point. The distance is just the length of the perpendicular dropped from the point to the line. Halt the algorithm if none of the assignments change.

Update step: Update the geometry (i.e equation) of each line as the least squares fit line that approximates the assigned points. the new means to be the centroids of the observations in the new clusters. \square

Theorem: The k-lines algorithm converges.

Proof: It would be sufficient to show that the algorithm is a monotonic coordinate descent over a positive definite function. Consider the following positive definite function:

$$E = \sum_{\lambda \in L} \sum_{p \in M(\lambda)} D^2(p, \lambda)$$

Here L is the set of lines. For a given line λ , $M(\lambda)$ is the set of points assigned to the line. For a given point p and a given line λ , $D^2(p, \lambda)$ is the square of the distance between the line and the point. In both the assignment step and the update step the above function either reduces or remains the same. The e-step reduces the above function by minimising over the point-to-line assignments. In the update step the function reduces by minimising the over the possible geometries of the lines. \square

In the above it became clear how a new coordinate-descent algorithm is formed by generalizing the notion of the cluster. It is desirable to similarly generalize the clustering to a graph based definition of clusters. The key difference here is in the following two aspects:

- The centre of the cluster is taken to be the geodesic or graph-constrained centroid of the cluster.

- Geodesic proximity is taken to be the criterion of assignment of points to its cluster, i.e. every point gets assigned to the cluster centre of minimum geodesic distance.

4.9 Bayesian Framework for Combining Limb Labels

This section presents the key definitions of Bayesian Framework for Combining Limb Labels as used in a geometric contexts. A brief outline would be presented here. The following texts present a more detailed discourse: [208], [149]

In a Bayesian framework [149] there are rules to attribute belief on a given outcome in response to given sets of possible observations. The main objective is to combine such beliefs to compute the updated belief in response to actual observations. Beliefs can be treated like probabilities and combined using Bayes' theorem (see [149]). If the first observation implies a belief p_1 and a second observation implies a belief p_2 on a given outcome and the two observations are independent of each other, the combined belief on the same outcome is given by the following :

$$P_{combined} = \frac{p_1 p_2}{p_1 p_2 + (1 - p_1)(1 - p_2)} \quad (4.1)$$

Similarly if there are n independent observations implying a belief of p_1, p_2, \dots, p_n on the specific outcome, then the combined belief is given as follows :

$$P_{combined} = \frac{\prod_{i=1}^n p_i}{\prod_{i=1}^n p_i + \prod_{i=1}^n (1 - p_i)} \quad (4.2)$$

To understand how this works consider the two possibilities in question. One possibility is that the outcome is true and the other is that it is false. The suggestion that the i^{th} observation indicates the outcome with probability p_i can be seen as an indicator of the outcome with confidence level p_i . This means that the indicator is randomly correct about the outcome p_i -fraction (or $100p_i$ %) of the time. In the first possibility, all the indications are correct and in the second possibility all of the indications are wrong. So, what is the probability that the first possibility holds in light of all those partially reliable indicators indicating the outcome? In the absence of any further information, the best estimate would be to split the probability between the two possibilities in the ratio of probabilities of the two combinations of indications that are associated with each possibility. In other words, the probability of a possibility is proportional to the *probability of the combination of indicators that indicate that possibility*. This quantity, i.e. *the probability of the combination of indicators that amount to a given possibility* is called its (i.e. the possibility's) *likelihood*. The likelihood is what the possibility's probability is proportional to, but is not equal to. Instead, the probability of the possibility is determined by normalising its likelihood using the likelihoods of all the possibilities spanned by the indicators.

In the above situation there are two possibilities - (1) *the outcome is true* and (2) *the outcome is false*. If the likelihood function is denoted as L then the probability of the outcome is given as follows :

$$P(\text{outcome is true}) = \frac{L(\text{outcome is true})}{L(\text{outcome is true}) + L(\text{outcome is false})} \quad (4.3)$$

If there are N possibilities spanned by the indicators, then the Bayesian probability of a given possibility is as follows:

$$P(\text{possibility}_i) = \frac{L(\text{possibility}_i)}{\sum_{j=1}^N L(\text{possibility}_j)} \quad (4.4)$$

In the original setting, the indicators are mutually independent, so the likelihood of the first possibility (i.e. all the indicators are correct) is the product of the individual p_i values. Likewise the likelihood of the second possibility (i.e. all the indicators are incorrect) is the product of all the $(1 - p_i)$ values. Thus one gets equations 4.1 and 4.2.

The general framework presented in equation 4.4 is more widely applicable and does not require the indicators to be independent. When two or more indicators are not independent of each other, it becomes necessary to know the indication-level implied by each joint combination of the said indicators. For example, if indicators I_1 and I_2 are not independent, then the probability of both being correct is not the product of their individual probabilities p_1 and p_2 . For such cases the combination-specific probabilities would have to be specified in tables known as *joint-distribution tables* and/or *conditional probability tables*. In a joint distribution table, the probability of each conjunctive combination of indicators is given explicitly. In a conditional distribution table, one needs to specify the conditional probability of correctness of one indicator given the other. The objective of either table is to specify combined probabilities of mutually dependent indicators.

Bayesian belief network (BBN) is a tool that allows specification of such dependences using a graphical notation and tabular input of conditional/joint distributions. It may be useful to simplify matters by flattening or expanding the dependent combinations and treat each combination as a separate indicator. This simplification allows belief-propagation work according to equation 4.2. This approach can be used in corroborating evidence from multiple indicators and would form the basis of a probabilistic rule-base for updating reliability functions from measurements. A special status is given to a particular indicator called the *prior*, which represents the a-priori knowledge about the probability of the outcomes in the absence of any transient observations. This distinction is useful in eliciting a base-line for the uncertainty.

So far a very special case was being considered in which a Bayesian framework can be used. It is special (i.e. not general enough) in more than one way. Firstly it is being applied to a single proposition, i.e. there is just *one* hypothesis whose probability is sought to be computed from some indicators. It is more common to try and estimate a probability distribution over a range of hypotheses.

4.10 Conclusion

A depth image captured by the Kinect sensor is a very crude representation of a 3D scene. The depth frame consists of tens of thousands of points sampled on the surfaces that were in front of the Kinect when the frame was captured. To get from this raw representation to detection of limb angles involves multiple stages of processing. A substantial body of research and exploration has been done towards each processing stage. This chapter presented the mathematical and algorithmic tools that were explored and found useful in processing and annotation of 3D

depth image data.

The chapter started with methods of defining connectivity/neighbourhood/adjacency relationships between individual sampled points. It was shown how neighbourhood graphs and polyhedrons could be inferred from scattered points. Subsequently it was shown how local geometric properties like normal and curvature could be estimated from the points and their neighbourhood graphs using geometric least squares method. This was followed by some algorithms that were based on a geodesic distance function defined on the approximated surface model represented by the said neighbourhood graphs. The geodesic distance function is extremely useful in defining recognizable features in the point cloud. It is using the geodesic-distance based features that the limbs are detected and thereby calculate angles between limbs. Novel concepts of graph-theoretic centroid and graph-theoretic clustering were introduced next. These algorithms play a crucial role in limb detection. The chapter ended with a section on how to use a Bayesian method for combining beliefs on any specific labelling.

Chapter 5

Gait Monitoring Using Kinect Sensor

5.1 Clinical Gait Monitoring

Afflictions of the brain and the nervous system, such as stroke, cerebral palsy, multiple sclerosis, Parkinson's syndrome etc. cause degradation of patient's motor functions. Such conditions can impair balance, walking gait and get in the way of doing everyday tasks, such as eating, drinking, picking things up etc. The severity and patterns of difficulty varies from patient to patient and often change with progress from healing and regress from deterioration. It is important to assess and track such variations to help with medical diagnosis. A variety of interventions and therapies have been developed to relieve symptoms of such mobility conditions. The ability to measure and quantify a patient's motor performance parameters is extremely valuable to clinical intervention processes. It serves as a feedback in the investigative aspects of therapy as well as helps with the choice of therapeutic alternatives. It helps with classification of types, stage, and severity of a patient's condition and makes it possible to unambiguously communicate such information. Gait monitoring is particularly useful for clinical study of patients with mobility problems caused by orthopaedic, rheumatic, and neurological disorders. There are high precision measurement facilities (Gait laboratories) but such facilities are expensive and rare. Also, results from such laboratories do not translate well to clinicians in practical terms [164]. As a result, in many areas of clinical practice, motor ability continues to be measured through the use of standardized rating scales based on an expert opinion. These classification scales include the Expanded Disability Status Scale (EDSS) [98], the Gross Motor Function Classification System [131], the Unified Parkinson's Disease Rating Scale (UPDRS) and the Movement Disorder Society sponsored UPDRS (MDS-UPDRS) [63]. These scales involve clinical tests in the presence of experts. The patients are asked to perform certain functional movements - such as walking, turning keys, picking up a small object, feeding oneself with a spoon, touching the finger to nose etc. The actions are observed by the expert who then grades them on an ordinal scale for each action. These scores are then combined and summarized into aggregates for certain

categories and an overall score. The general score and individual scores are then combined and summarized into system scores and/or a global score which may be used to characterize patients and judge specific treatments. Despite careful design, standardised rating scales have inherent issues of sensitivity and reliability [76]. The apparatus for classifying and articulating these assessments in a meaningfully quantifiable way are arguably somewhat blunt. Such examinations and assessments uses standardized rating scale such as Gait Assessment Rating Scale (GARS) for gait analysis. It takes about 1 hour of neurologist's time and hence are very expensive. Figure 5.1 shows the phases of a gait cycle that can be characterised primarily by how the knee angle varies along the periodic cycle. The gait cycle of a person with mobility or coordination problem is usually quite different from that of a healthy person. Figure 5.3 shows plots of knee angles along the gait cycle, with the blue traces for a normal healthy person, and the red traces corresponding to the gait of people with osteoarthritis. It is very useful monitor the progression of the disorder and the effect of medication through gait monitoring. Such monitoring currently requires expensive facilities (e.g. VICON) that cost in excess of tens of thousands of pounds.

This high cost limits the availability and frequency of such monitoring sessions. On the one hand, the gait analysis laboratories offer opportunities for more accurate, reliable and precise motion tracking but their practical requirements and cost place huge restrictions across different potential sites. On the other hand, the various movement classification scales may be limited in their precision and reliability but do have the benefit of being more widely deployable across a range of different healthcare contexts. These factors have motivated keen interest in recent development in sensor technology and robust wireless data transfer. The automated sensor assessment could reduce the cost of gait analysis. A gaming console device, the Microsoft Kinect sensor, (that costs under £100) can be used for such gait monitoring in the comfort of the patients' own home and care-homes. Its 3D imaging capabilities have been described as inadequate [49] for healthcare applications with the primary objection been its lack of accuracy. The previous rehabilitation monitoring application of Kinect has worked around the low accuracy and not addressed it (e.g. by way of monitoring aggregated characteristics rather than full motion capture). This work develops a set of algorithms which enhance the accuracy and demonstrate the efficacy of Kinect as an inexpensive alternative to elaborate setups like VICON. The developed algorithms have been encapsulated in a gait monitoring system software, *Joint Angle from Frames Acquired using KinECt (JAFKEC)* [154].

Understanding the walk cycle is an important aspect for gait modelling. The walk cycle can be broken down into various phases [23] that are based on the sequence of feet to the point of contact to the ground. The left stance phase of a stride starts with right foot on the ground and the left heel starting to strike the ground. In this phase the body is supported by both feet until the right foot starts to move up and the right toe leaves the ground. Then the right foot leaves the ground and starts swinging forward. The right heel strikes the ground and both the feet are on the ground again. Next the left toe leaves the ground and the left stance phase is completed. The period in which right toe leaves the ground, the right leg swings forward and the right heel strikes the ground is the right swing phase which is a subinterval of the left stance phase. The right stance phase is initiated by the right swing phase. The analogous phases proceeds with the roles of the right leg and the left leg switched. The walking cycle [82] is defined by alternating periods of single and double support as shown in Figure 5.1. Kadaba et al. [90] introduced a

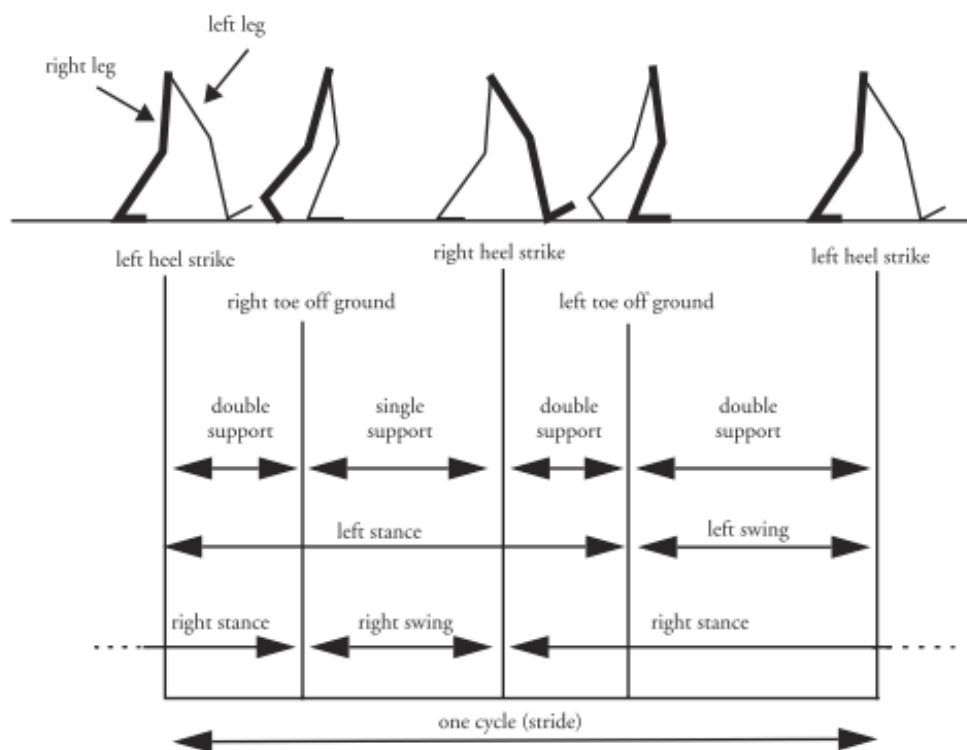


FIGURE 5.1: Human Walk Cycle [82]

statistical method for lower limb gait analysis and calculated the joint angle data. They [91] have used the 6-camera 50Hz VICON system to collect lower limb joint kinematics data and used VICON Clinical Manager software to calculate the joint angles. Figure 5.2 shows a plot of the knee flexion angle for a typical gait cycle. Such a plot can be of diagnostic value, as the pattern of the plot could reveal the nature and intensity of the disorder. For example, Figure 5.3 shows the knee flexion angle over gait cycles for a normal person (the blue trace) and a person with osteoarthritis (the red trace).

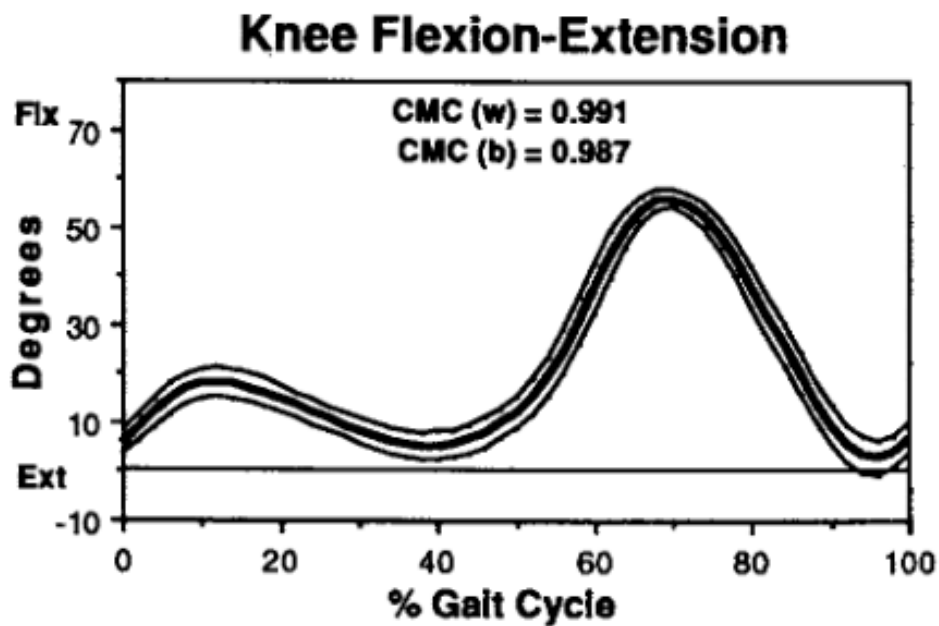


FIGURE 5.2: Knee flexion and extension angles of gait cycles [90]

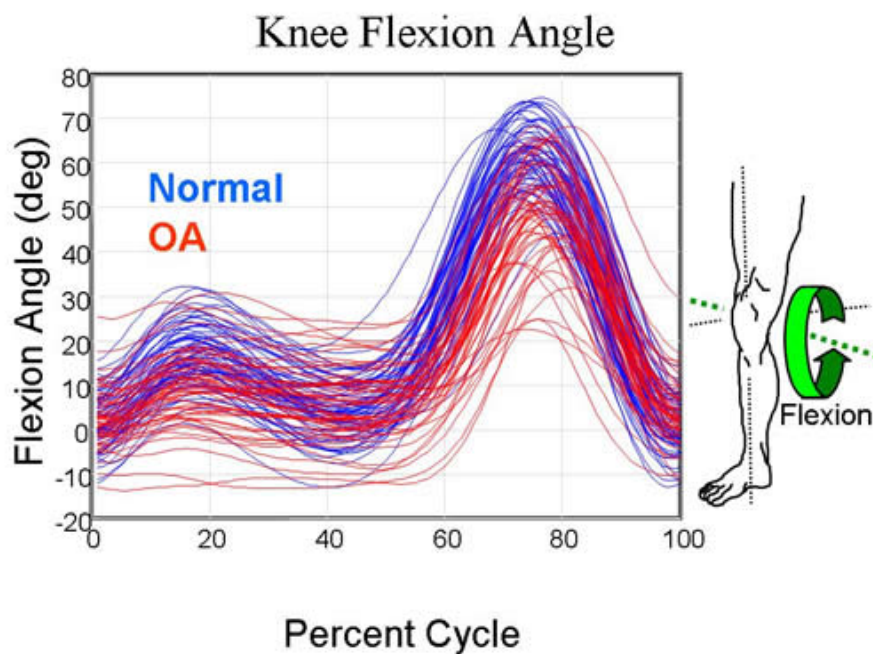


FIGURE 5.3: Knee flexion angle curves of normal subjects (blue) and osteo-arthritis patients (red) (Source: [38])

5.2 Literature Review

The surveyed papers may be classified into three categories: (a) papers on applications of Kinect for human lower limb and gait in healthcare (b) papers on gait feature representation and gait recognition (c) papers on algorithms for gait capture. Following is a survey of papers reviewed.

5.2.1 Previous work on applications of Kinect for human lower limb and gait in healthcare

The gait analysis laboratories offer highly precise, reliable and accurate motion capture but their high cost and large facility requirements make their deployment hard to afford for frequent monitoring. Infrequent use reduces the scope of continuous tracking and fine-tuning of clinical interventions. On the other hand, low cost consumer-grade sensors and depth sensing cameras like Microsoft Kinect have the potential to produce more cost effective and easy-to-use systems. Motion frames captured by Kinect has been used in the context of clinical applications by many researchers in the last few years. The evidence so far suggests that Kinect is appropriate for functional ability assessment of motion in healthy people [16]. Stone and Skubic [174], [171] have worked using Kinect motion data in order to detect feet touching the ground. Their system automatically generated alerts to clinicians in response to specific gait patterns of patients residing monitored continuously at their homes. Mentiplay [119] have focussed on specific characteristics such as static foot pressure. Stone and Skubic [173] applied Kinect in continuous in-home gait analysis. Ning and Guo [126] have used Kinect depth data for assessing spinal loading. Stone and Skubic [172] efficiently used Kinect in the context of Parkinson Disease (PD). Parajuli et al. [133] developed a senior health monitoring system using the Kinect to analyze gait, performed posture recognition and transitions between sitting and standing and also compared normal versus abnormal walking to help prevent falls of elderly people at home. Wearable, mobile device sensors and biomarkers have been used for patients with PD to quantify gait and tremor [51], [132], [80] but many older people are reluctant and sometimes strongly opposed to the use wearable devices because of inconvenience [39]. Remote health monitoring and activity tracking using wireless devices [6] can be used to assess recovery progress and streamline the communication between the patient and doctor. Geerse et al. [62] extracted human gait features using the skeletal joint data using multiple Kinect with 10-meter walking test. Kinect has been used by Dolatabadi et al. [47] in patients' home to monitor changes in spatio-temporal gait parameters like step length, stride length, stance time and cadence during the recovery period. Pfister et al. [138] measured stride timings using Kinect, compared with VICON and stated that Kinect requires adjustments to accepted as a clinical tool for gait monitoring purposes.

5.2.2 Previous work on algorithms for gait capture

Human identification by gait has been done by Goffredo et al. [64] using Kinect using markerless feature extraction method by implementing the K-nearest neighbour (KNN) algorithm. A recognition rate of 73.6% is achieved using the KNN classifier with $k=5$. Cunado et al. [35] extracted gait signature from a sequence of walking using the KNN classification to Fourier series components to represent the motion of the upper leg. For more accurate lower body segmented parts associated with the most important body motion, the human leg motion is modelled as a pendulum [35]. Cippitelli et al. [27] proposes a low complexity algorithm which is applied to extract data from Kinect in real time and computes joint trajectories in scanned human motion from a side view. The joint extraction is used to analyse gait for the “Get Up and Go Test”. Rimminen et al. [147] used near-field imaging floor sensors for fall detection. Two-state Markov chain was used for fall classification (falling, getting up) and pose estimation was implemented using Bayesian filtering. Gabel et al. [58] developed a method of full body gait analysis through the use of Kinect-based data and a multiple additive regression tree algorithm [57], [56]. The system monitored the time of the stride, stance and swing phases of a gait cycle, as well as angular velocities of arm movements; however, measurements of lower limb angular velocities and core posture were also noted to be possible. Xue et al. [199] applied the wavelet decomposition of Gait Energy Image (GEI) and obtained skeleton parameters to distinguish between different types of gaits like normal walk, walking with volleyball etc. Support Vector Machine (SVM) has been used as classifier for different gaits. Lopez et al. [111] also used SVM for classification of gait data. Tsanas [185] presents a novel speech signal processing algorithms for high-accuracy classification of Parkinsons Disease. Schnabel [155] measured geodesic distances of human body parts using graph-based representation of the Kinect depth data and tracked various full-body movements for 3D pose estimation using efficient RANSAC algorithm for point cloud shape detection. Shotton et al. [161] uses a Bayesian machine learning approach to estimate limb orientations from Kinect’s depth image in real time. Schwarz et al. [158] measured geodesic distances of human body parts using graph-based representation of the Kinect depth data and tracked various full-body movements for 3D pose estimation.

5.2.3 Previous work on gait feature representation and gait recognition

Gait recognition techniques can be broadly classified into model based methods [64], [177], [199] and appearance based [200], [166] methods. Model based methods usually extract the motion features, fits various kinds of stick figures through tracking the body parts and measures the parameters using fitted models. Appearance based methods characterize the whole motion pattern of the human body and are vulnerable to illumination, variation in clothing, etc.

Various gait recognition methods have been proposed to address multiple covariates such as variation in clothing, load carrying, walking speed, occlusion and unconstrained paths. Lopez-Fernandez et al. [111] presented a rotation invariant gait descriptor for multi-view recognition on in-constrained paths using support vector machine (SVM) classification. Hofmann et al. [78] implemented Principal Component Analysis and Linear Discriminant Analysis (PCA+LDA). Also for classification, nearest-neighbor classification has been used in depth based gait recognition technique from Kinect data. Sivapalan et al. [166] developed a frontal gait recognition system, Gait Energy Volume (GEV), using appearance-based techniques and Kinect depth data. Tang et al. [179] proposed a Multi-View Synthesizing Method (MVSM) for gait view angle estimation using depth data from Kinect. In their system, Gaussian curvature and mean curvature are used to extract 3D gait features. The method could be used for practical surveillance applications. Goffredo et al. [64] implemented a marker-less model-based approach for gait biometrics for self-calibrating view-independent gait feature representation and estimated the poses of the lower limbs based on marker-less motion estimation. They have presented a marker-less view-independent gait analysis algorithm in which the extraction of gait features are invariant to change in view.

The present contribution (JAFKEC-G) is in principle on the lines of [64] in that it also estimates joint coordinates based on image features. However, in terms of algorithmic details, the proposed approach is quite different. Goffredo [64] uses coordinates and Euclidean distances in 2D pixel space for defining features, whereas the current work uses coordinates in 3D space and defines features in terms of geodesic distance.

5.3 JAFKEC-G for Gait System Overview

As introduced in the introduction section, JAFKEC-G is a tool developed for visualising and analysing gait data as captured by the Kinect sensor. It uses the depth field image captured using an infrared laser array projection. Making it based on the depth field image alone offers the advantage that it can operate without requiring the patient to wear additional objects like reflecting markers or sensors. The setup required is minimal, so that the system could be used in GP's or nurses' chambers, rather than specialized body tracking laboratories. The time typically allocated per patient in a GP or nurse appointment is only a few minutes, and an elaborate setup process (e.g. changing clothes, wearing markers/sensors) would not be practical in such scenarios. A table mounted kinect sensor and a 3 meters of walking space would be adequate to capture the required depth image. JAFKEC-G system consists of the following software components: 1. A 3D point cloud viewer to render the depth frame as directly captured by the Kinect. For each frame captured by Kinect the depth values for each captured pixel is stored. The *display* method of the viewer goes through the pixels and renders 3D glyphs for each pixel that belongs to the person (patient) detected in the frame. The point cloud animation is rendered in real time and allows the user to interact with the 3D view for obtaining the coordinates of any point selected using mouse clicks. 2. A kinematics skeleton model viewer that renders a realistic skeleton according to a given set of joint coordinates. This skeleton can be controlled by a stream of joint coordinated computed from the Kinect data. 3. JAFKEC-G has set of functions dedicated for automatic measurement of knee angle based on the point cloud data.

5.4 Algorithms for Gait Capture from Depth Image

The following algorithms were implemented and tested within JAFKEC-G to capture an accurate time-series of joint angles from the depth image captured by the Kinect sensor.

5.4.1 Algorithm : Medial Axis from Binary image

The depth frames are converted into grey-levels as shown in Figure 5.4, which is then converted into a two-level bit-map (binary image), shown in Figure 5.5 to keep only the leg that is closer to the Kinect. Next a thinning algorithm or a medial-axis algorithm is used for computing a set of skeletal line segments that represent the central line going through the middle of the silhouette of the binary image.

Following is the outline of a thinning algorithm based on [67].

Binary image thinning algorithm;

Data: A binary image given as a doubly indexed array $P(i, j)$ of pixel values

Result: A skeletal image formed by thinning of the original image

```

iteration  $\leftarrow$  0 continue  $\leftarrow$  true; while continue do
  continue  $\leftarrow$  false; foreach pixel position  $P(i, j)$  do
     $P_1 \leftarrow P(i, j)$ ;
     $P_2 \leftarrow P(i, j - 1)$ ;
     $P_3 \leftarrow P(i + 1, j - 1)$ ;
     $P_4 \leftarrow P(i + 1, j)$ ;
     $P_5 \leftarrow P(i + 1, j + 1)$ ;
     $P_6 \leftarrow P(i, j + 1)$ ;
     $P_7 \leftarrow P(i - 1, j + 1)$ ;
     $P_8 \leftarrow P(i - 1, j)$ ;
     $P_9 \leftarrow P(i - 1, j - 1)$ ;
     $C_1 \leftarrow \neg P_2 \& (P_3 | P_4) + \neg P_4 \& (P_5 | P_6) + \neg P_6 \& (P_7 | P_8) + \neg P_8 \& (P_1 | P_2)$ ;
     $N_1 \leftarrow (P_9 | P_2) + (P_3 | P_4) + (P_5 | P_6) + (P_7 | P_8)$ ;
     $N_3 \leftarrow (P_2 | P_3) + (P_4 | P_5) + (P_6 | P_7) + (P_8 | P_9)$ ;
     $N \leftarrow \min(N_1, N_2)$ ;
    iteration  $\leftarrow$  iteration + 1;
    if ( $C == 1 \& 2 \leq N \leq 3$ ) then
      if iteration  $\equiv 0 \pmod{2}$  then
         $Q \leftarrow \neg((P_2 | P_3 | \neg P_5) \& P_4)$ ;
      end
      else
         $Q \leftarrow \neg((P_6 | P_7 | \neg P_9) \& P_8)$ ;
      end
    end
    if  $Q$  then
      Delete pixel  $P(i, j)$ ;
      continue  $\leftarrow$  true;;
    end
  end
end

```

Algorithm 10: Binary image thinning algorithm

Hough transform is used to get canonical parameters of straight lines containing the medial axis segments. These Hough transformed points are then grouped into clusters using the k-means clustering method.

The medial axis algorithm frequently produced nice straight line segments as shown in Figure 5.6 but almost equally often produced branching lines and loops that led to difficulty of telling apart the actual skeletal segments from branches and loops. It turned out that such branches and loops are unavoidable in medial axis algorithms. Figure 5.7 shows the skeletal segments as



FIGURE 5.4: Depth frame converted into grey scale



FIGURE 5.5: Binarized frames representing the region around a knee

computed by Algorithm 10 corresponding to the binary leg segments shown in Figure 5.5. It can be seen that many of the images ended up with undesirable branches. Small bumps and dents on the boundary of the binary image can give rise to new branches in the medial axis. This approach is thus unsuitable for the proposed application.

5.4.2 Algorithm : Sectioning the 3D point cloud

A new algorithm was developed for estimation of angle of the knee closer to the Kinect (Figure 5.8). The leg closer to the Kinect is isolated by dissecting the point cloud using a sagittal plane. A few thin slices are taken from the point-cloud corresponding to the leg nearer to the camera. These are somewhat semi-circular point-sets, representing the leg’s sections at various heights.

Two pairs of such slices are chosen to represent the femur and tibia each. The longitudinal axial points of the limb section is estimated as the centre of the best fit semi-circle running through the sample points on the section, and the line through these axial points is taken as the geometric idealisation of the limbs. It was found that the best fit semi-circle is not always the

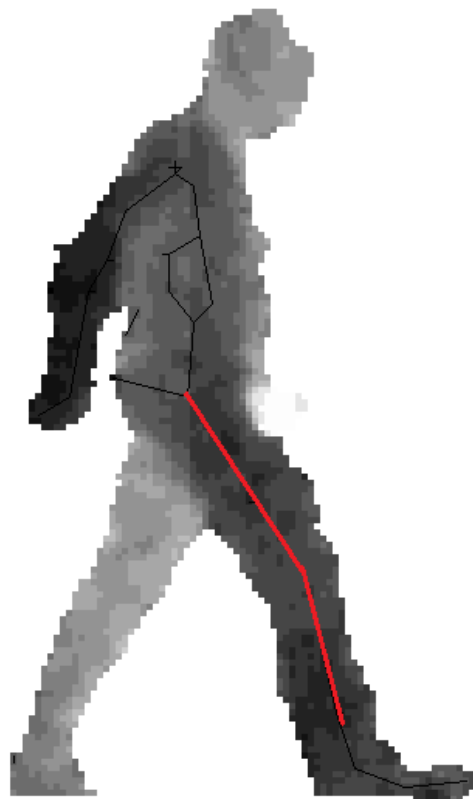


FIGURE 5.6: Medial axis skeleton from the depth image

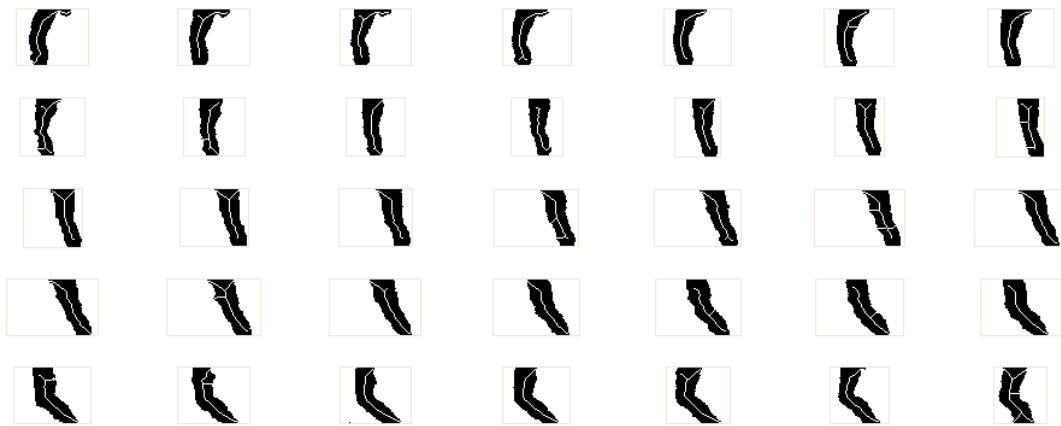


FIGURE 5.7: Medial lines detected on binarized images in Figure 5.5

best measure of centre of the limb cross-section. The sectional rim centroids work better (i.e. in better agreement with the VICON data) than the centre of the best-fit semi-circles. Despite the improvement, the accuracy of the angle results were lower than the desired level and it was not possible to improve on it by changing the parameters. So it was decided to adopt a new approach.

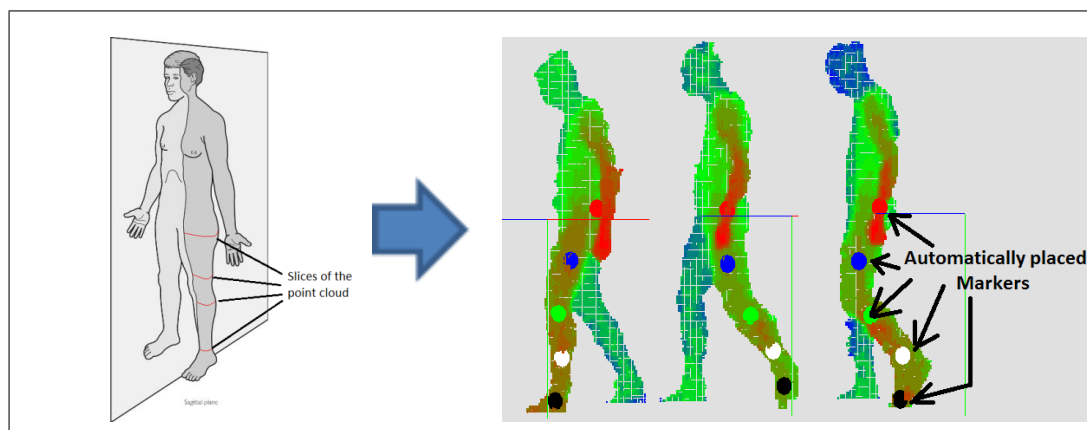


FIGURE 5.8: Automatically placed markers obtaining sections of the point cloud

5.4.3 Algorithm : Least Squares Line Fitting

The entire problem of gait capture could be broken down into a few stages. The first stage is that of foreground vs. background separation. This is done using subtraction of an empty background frame. The next step is that of limb labelling, which refers to assigning limb labels on parts of the point cloud. The next step involves determination of medial or skeletal lines corresponding to the limbs. These skeletal lines are geometric idealizations of the limbs much like the lines going through the markers. The algorithm for calculating these medial lines is the processing stage that this work investigates. The limb labelling process gives rise to designated segments of point clouds that could each represent a limb. The method previously used in GLSKEL involved taking two sections of the limb's point cloud at the upper and the lower ends and join a line through the centres of those sections. This works with partial success and is particularly inaccurate in some configurations. In the subsequently improved method, several sections (i.e. not just two, unlike the previous method) were taken and a line was fitted through the centres. This modification improved it partially. In a further improvement all the points (i.e. not just sections) from the labelled limb's point cloud were used for fitting a line such that the sum of square of distances of the line from the points is minimised.

The method boils down to determination of eigenvectors of the covariance matrix of x , y , and z components of the points. The points on fema and tibia are approximately on a semi cylindrical surface. A line that minimizes the distance from all of those points would be the axis of the cylinder. So an error function was constructed that represents the sum of squares of the distances from the line expressed as a function of line parameters. Then the partial derivatives of the said error function w.r.t. the line parameters can be equated to zero to give rise to a system of linear equations whose solution represents the best fit line.

The labelling of fema and tibia points was done by sectioning it using horizontal planes but when the limbs make a non-zero angle with the vertical axis, the horizontal planes intersect the limbs obliquely. This effect creates an axial asymmetry in the parts near the top and bottom of the limbs. Ideally the segmentation should be done using planes that are perpendicular to the limbs. However in order to compute such planes it is necessary to find the angle that the limbs make with a vertical reference plane. This is a case of cyclic dependency. This problem was solved by successive refinement of the sectional planes. This is illustrated in Figure 5.9.

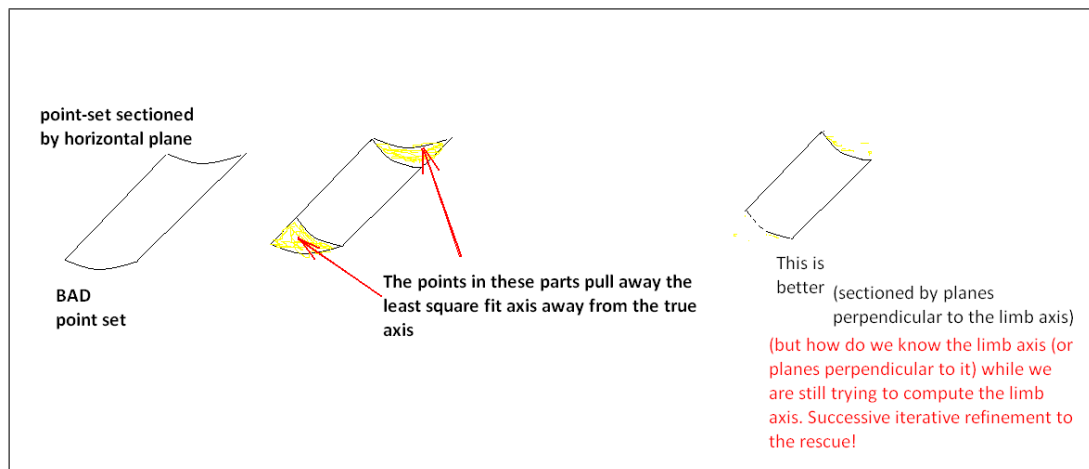


FIGURE 5.9: Point cloud segmented by planes perpendicular to the limb's medial axis

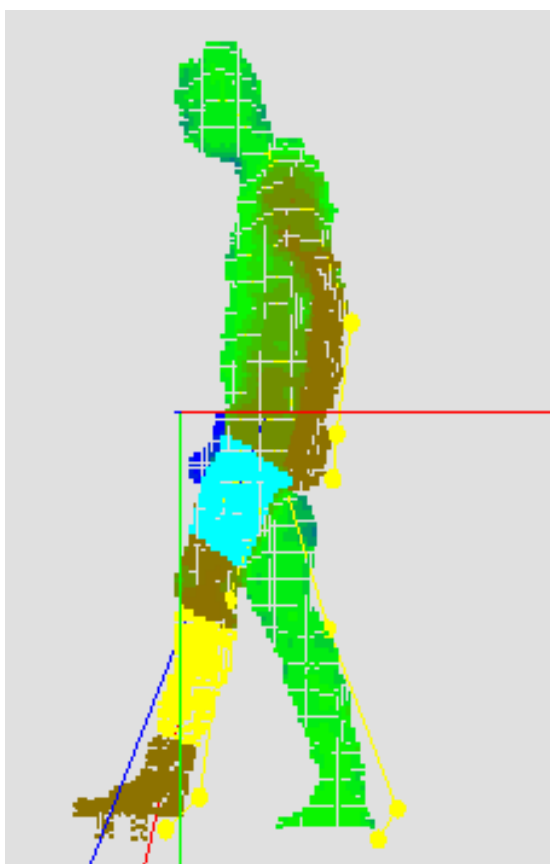


FIGURE 5.10: A screenshot showing the labelled point cloud and the fitted limb axes

This works quite well in most cases (Figure 5.10) but some special problems have been identified. This method substantially improves the agreement with VICON measurements. Earlier single sections are used which were more prone to inaccuracy than this new method which takes into account much larger number of points at a time. Despite the improvement, this method is sensitive to small asymmetries formed by non-uniform distribution of points and partial obscuration of the fema by the swinging hand. This problem was overcome by (1) fitting a surface and resampling points from the surface, and by (2) changing the aforementioned least square

function so as to fit an elliptical frustum through the points. An elliptical frustum would have a few more degrees of freedom and therefore expected to produce a tighter fit of the points.

In summary, this stage of development have the following workflow in JAFKEC-G for capturing gait in terms of knee angles.

1. Recording the frames coming from the Kinect device (a) using a compressive CODEC, and (b) taking advantage of the foreground vs. background separation supported by the Kinect SDK library.
2. The depth frame is loaded and displayed in 3D as a point-cloud.
3. The leg closer to the camera is isolated by dissecting the point cloud using a sagittal plane.
4. A few thin slices are taken from the point-cloud corresponding to the leg nearer to the camera. These are somewhat semi-circular point sets representing the leg at various heights.
5. Two pairs of such slices are chosen to represent the tibia and fema each. The longitudinal axial points of the limb section is estimated as the centre of the best fit semi-circle running through the sample points on the section and the line through these axial points is taken as the geometric idealisation of the limbs.
6. Segments of the point cloud are labelled for each limb, which is then used to fit a 3D medial axis by least squares.

5.4.4 Algorithm : Convex Polyhedron Fitting

Algorithm 3 has the shortcoming that when the swinging hand obscures a part of the fema, the point-distribution gets quite distorted even after discarding the points corresponding to the hand, as it casts a shadow on the femur, thereby producing a hole (i.e., marked by absence of points) in the femur's point cloud. The hole is healed by a new algorithm that fits a convex polyhedron on the femur using the convex hull algorithm [10]. There are multiple algorithms for computing the convex hull. An intricate but efficient algorithm is present in the CGAL library [75]. For completeness a simplified version of convex hull computation is presented in algorithm 11.

Algorithm 11 has quadratic ($O(N^2)$) cost but the CGAL algorithm [75] uses a hierarchical decomposition scheme to reduce the cost to $O(N \log(N))$. The convex hull thus computed is the minimal convex polyhedron containing all the points on the fema, and when the swinging hand casts a shadow on the fema, the convex hull produces triangular facets to fill in the concave

ALGORITHM CONVEX HULL

Data: A set of N 3D points whose convex hull is being requested

Result: A set of triangular facets representing the convex hull boundary

Project the points on a plane (say the xy plane);

Find a pair of points that would be an edge on the hull;

(This can be done using the projected points as if it was the first step of 2d convex hull algorithm - i.e. choose the topmost point and then choose a second point so that all other points lie on one side of the line joining the two points).

The corresponding pair of 3D points serve as the initial seed for growing the 3D convex hull.

Find a third point in 3D such that all other points lie entirely on one side of the plane passing through the first three points.;

These three points constitute the first triangular facet of the convex hull.;

For each edge on triangle, find another point to build a new triangle that satisfies the same property that all other points lie entirely on one side of the plane of the triangle. ;

Each triangle gives rise to new edges on which one builds triangles in the same manner as described above;

Repeat this process until there are no open edges left (where an open edge refers to an edge that is adjacent to just one face).;

Ultimately a closed volume will be formed by the incrementally added triangles.

Return this closed volume as the convex hull;

Algorithm 11: A 3D convex hull algorithm

cavity formed by the shadow cast by the swinging hand. The convex hull polyhedron has the added advantage in that the inertia matrix can be calculated irrespective of the point-density.

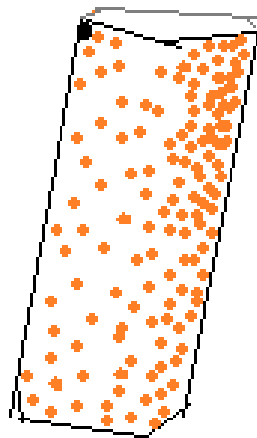


FIGURE 5.11: A point distribution and its convex hull.

Consider Figure 5.11 in which the point cloud is shown using the orange coloured dots and the convex hull using the outlines. If the inertia tensor is calculated purely from the (scattered) points, it would pull the principal axis towards the region where the density of points is higher (as the points get treated as point masses), whereas the convex-hull polyhedron's inertia tensor would implicitly assume homogeneous mass distribution over the whole volume. Point sample density would have no impact on the inertia tensor.

A mathematical algorithm (as in [121]) can be used to compute the moment of inertia from the facets of the convex hull polyhedron. Figure 5.12 broadly describes the mathematical idea behind such a computation. The moment of inertia is a volume integral, which gets reduced to a surface integrals by Gauss' divergence theorem. Then the surface integrals are projected onto

planes of convenience, and these are reduced into line integrals by Green’s theorem, from which the integral values can be readily computed per facet, which then can be assembled back into the target volume integral.

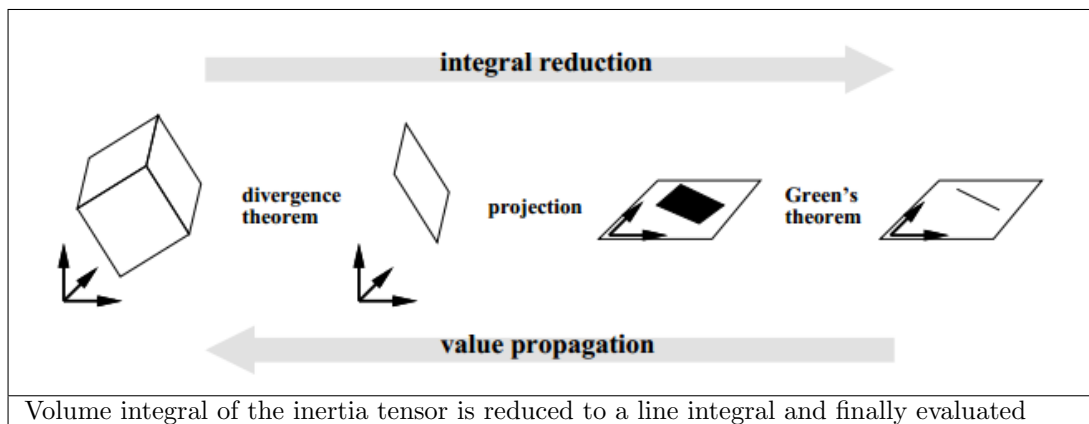


FIGURE 5.12: Volume Integral Calculation

ALGORITHM: COMPUTE PRINCIPAL AXIS;

Data: A polyhedron P defined by its edges E and faces F

Result: Its principal axis

Compute inertia matrix of the polyhedron using [121];

Compute the eigenvalues and corresponding eigenvectors of the inertia matrix;

Let A be the eigenvector corresponding to the largest eigenvalue;

Normalise A into a unit vector;

Return A ;

Algorithm 12: Algorithm for computing principal axis from a limb

5.4.5 Algorithm : Calculating Average of Nearly Aligned Polyhedral Outlines.

There is a phase in which the swinging hand partially eclipses the fema in such a way that it does not make a cavity that can be filled by the convex hull, but it chips away a corner of the fema so that the point-set envelope is nearly convex. Such a configuration is shown in Figure 5.13.

In such case it becomes necessary to use a different approach because the convex hull’s principal axis would incorporate the asymmetry introduced by the corner of the fema getting chipped away. So a method was developed based on an algorithm to average out the selected outlines of the convex polyhedron as seen from the Kinect’s view. The outline edges are the edges that bound the silhouette of the polyhedron. Outline edges are selected that are approximately aligned with the principal axis. This method improves over the volume principal axis because the limbs are slender objects and their silhouette edges that run along the principal direction further corroborates the principal direction. Algorithm 13 presents this refinement process in pseudo-code notation.

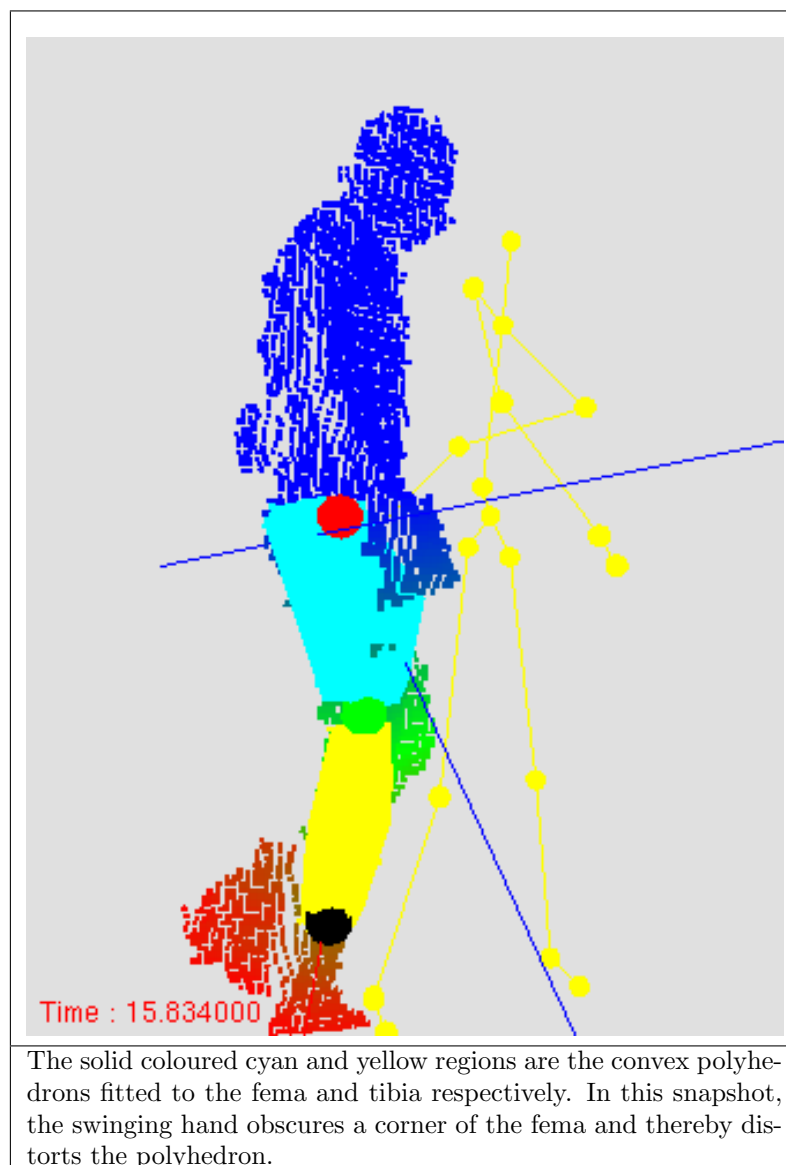


FIGURE 5.13: Convex Polyhedron Fitting

5.4.6 Algorithm : Use of Geodesic distance for Limb Labelling

This is a key algorithm in the current JAFKEC-G framework. Geodesic distance is the shortest distance between two points on a surface, as measured on the surface. Geodesic distance labelling plays a convenient role in labelling the point-cloud. Initially the Microsoft Kinect SDK is used as a source of hint for labelling the point-cloud. This soon turned out to be a bad choice due to its inaccuracies leading to incorrect labelling. Geodesic distance was used as an alternative labelling mechanism. To facilitate calculation of geodesic distance, a local neighbourhood graph is formed from the point cloud, and single-source shortest path distances are calculated based on local distance metrics on the neighbourhood graph. An example of this is the method of labelling leg points. In this case the points above the waist are labelled as $distance = 0$, and let Dijkstra algorithm run in order to populate the rest of the points with the shortest distance from the points above the waist. If the distances are normalised by dividing the distance labels by the

ALGORITHM: REFINE PRINCIPAL AXIS;**Data:** A polyhedron P defined by its edges E and faces F **Data:** The kinect's view direction as a 3D cartesian unit vector d **Data:** The principal's axis direction as a 3D cartesian unit vector a **Result:** A refined principal axis $S \leftarrow []$ /* initially empty list of silhouette edges */**foreach** edge e in E **do** $F_1, F_2 \leftarrow$ The faces adjacent to edge e . $n_1, n_2 \leftarrow$ unit normals to F_1, F_2 respectively **if** $n_1 \cdot d * n_2 \cdot d \leq 0.0$ **then** Add e to S ; **end****end** $L_{max} \leftarrow 0$ **foreach** edge e in S **do** Let $\eta(e)$ be the unit vector along edge e ; alignment(e) $\leftarrow |\eta \cdot a|$; $L_{max} \leftarrow L_{max} +$ length of e ;**end**Sort S in the decreasing order of alignment(e); $L \leftarrow 0$; $A \leftarrow$ 3D zero vector $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$;**foreach** edge e in sorted S **do** **if** $L < \frac{L_{max}}{2}$ **then** $\lambda(e) \leftarrow$ length of e ; $L \leftarrow L + \lambda(e)$; $A \leftarrow A + \lambda(e)\eta(e)$; **end****end**Normalise A into a unit vector;Return A ;**Algorithm 13:** Algorithm for refining principal axis direction using nearly aligned silhouette edges

maximum distance, the stratification by such normalised distance can give classification of the points into such features as fema, knee, tibia, ankle etc., irrespective of the scale and absolute position of the point-cloud.

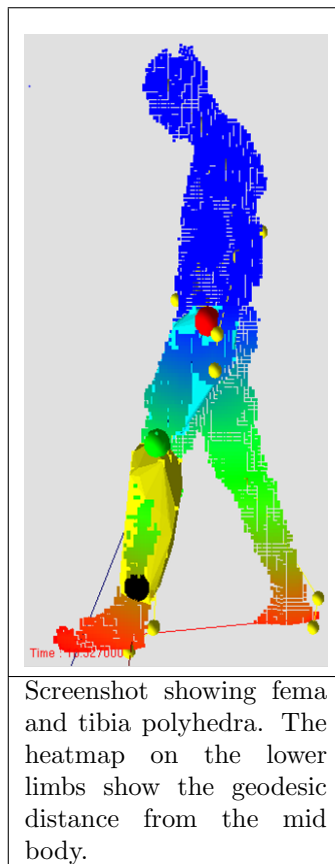


FIGURE 5.14: Geodesic distance based labelling

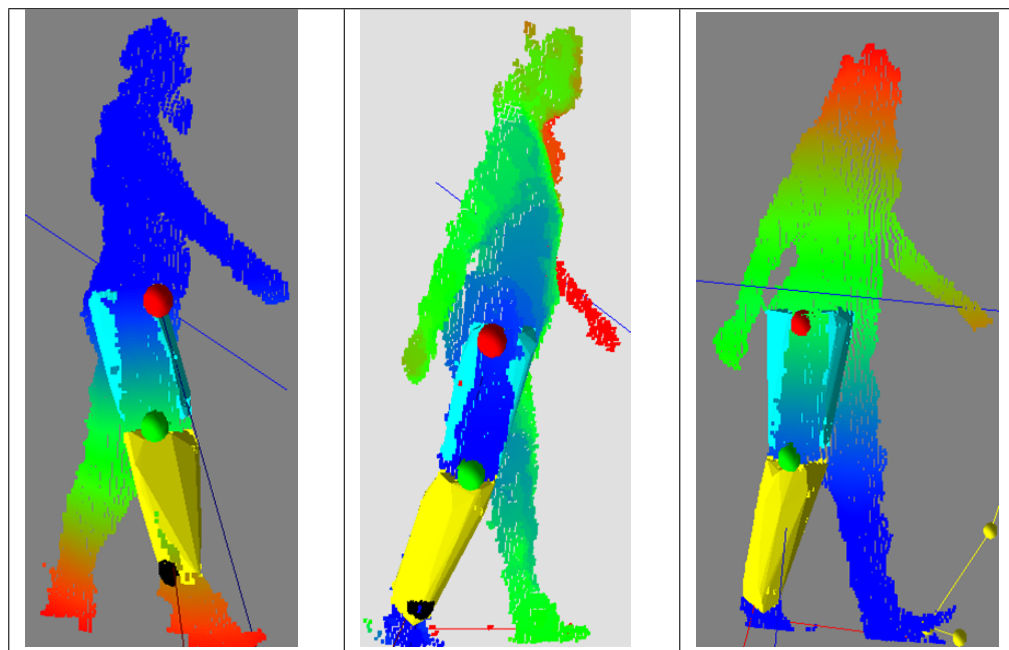


FIGURE 5.15: Multiple geodesic distance based labelling to improve classification of cloud points

The following Figure 5.16 describes the flowchart of the novel JAFKEC-G system for gait.

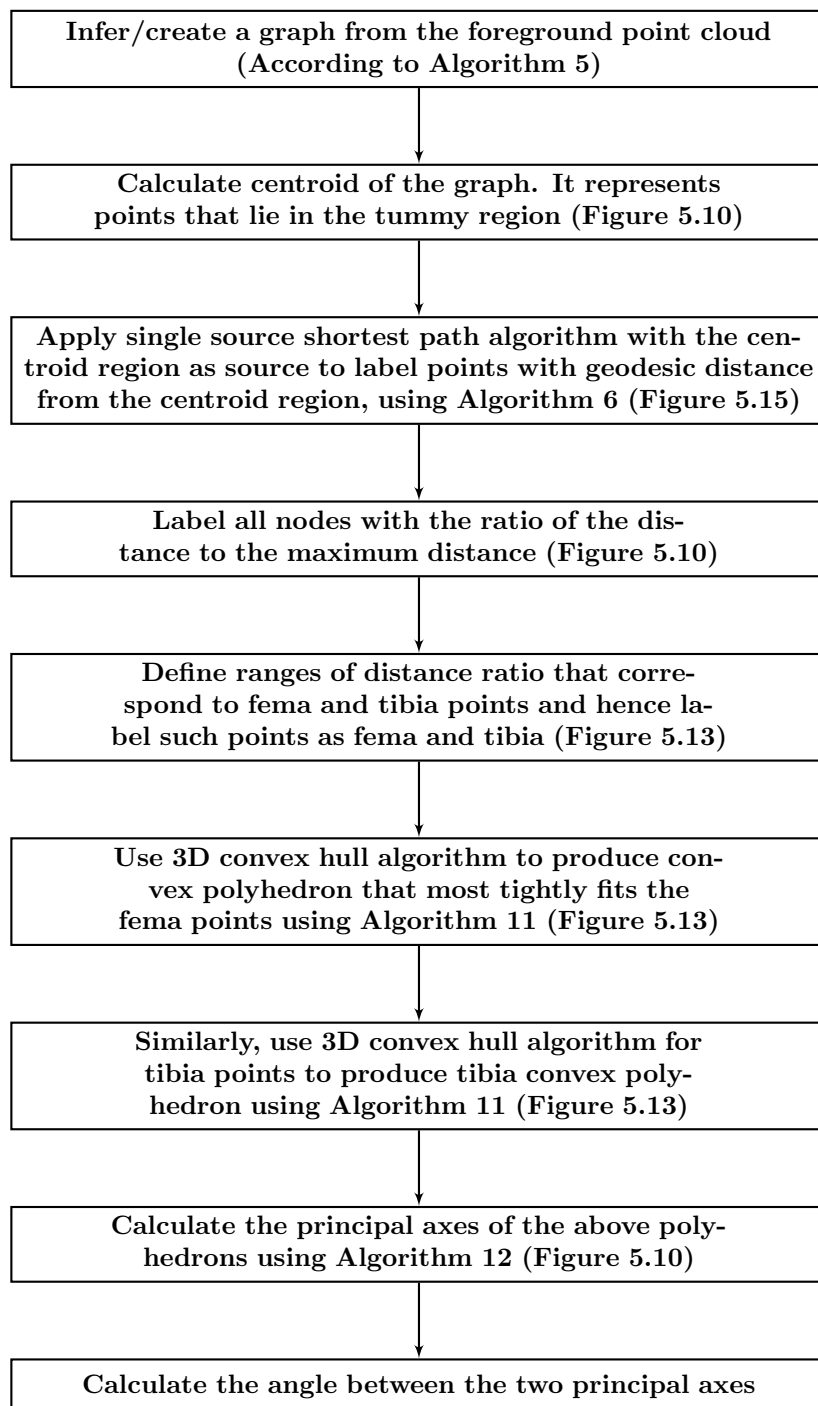


FIGURE 5.16: Flowchart for JAFKEC-G gait algorithm

5.5 Kinect-based lower limb motion analysis - Methods

The proposed approach starts with an investigation of the achievable accuracy and suggests using some low-complexity algorithms, such as shortest path algorithm, least squares line fitting, 3D convex hull algorithm, multiple geodesic distance labelling, etc. (table 5.1) to improve the accuracy of joint positions. The captured depth frame from the Kinect recorder is compressed via run length encoding to allow storage of more frames in a limited memory. The stored point cloud is then used to perform body part labelling and pattern recognition in order to detect the limbs of interest. This process is followed by feature extraction via multiple geodesic distance labelling, shortest path algorithm and model fitting on specific subsets of points once the geodesic distances are attached to the points.

Algorithm	Application in JAFKEC-G
Geodesic distance labelling	Forming a mesh representing the surface
Shortest path algorithm	Defining features of points for limb classification
3D Convex hull	Assembling the classified points for a single limb into a polyhedron so that point-density does not have an effect on axis calculation
Least-square line fitting	Determining the principal direction of an oblong polyhedron or point-set corresponding to the limbs of interest.
Sectioning of point cloud	Segmenting into sub regions such as for specifying source regions for shortest path and for feature definition.

TABLE 5.1: Algorithms used in the JAFKEC-G for gait system

With these features attached to each point, points are classified in the space of the features as belonging to specific limbs. Finally, the principal axes of the limbs are computed and the joint angles are calculated as angles between the principal axes. Experiments: Two different healthy males and one female were invited for recording using Kinect for a few gait trials. Participants were instructed to walk across a 10 meter walkway in the laboratory for the recording using the Kinect at different speeds (fast, moderate or casual walk, slow) for the gait data capture of both the left and right legs for approximately 1 to 2 minutes for each trial. Spatiotemporal data were collected simultaneously using a 12 camera VICON system with reflective markers placed according to the VICON requirement.

5.6 Results

Experiments were performed to evaluate the proposed system, and results are benchmarked with the state-of-the-art VICON optical movement analysis system.

Several experiments were performed and recorded in the University Biomechanics laboratory using different versions of Kinect sensor viz, Kinect 1.0 and Kinect 1.8 and were compared with VICON data simultaneously collected. Subjects were instructed to walk on the 10 metre walkway for approximately 1 to 2 minutes for each trial.

These results, using a range of algorithms discussed in Chapter 4, Chapter 5 and under different walking conditions are shown in Figures 5.26, 5.25, 5.27 shows that the results obtained using the Kinect version 1.8 are a good match with the VICON results.

Some more Kinect and VICON results are shown in Figures 5.17, 5.18, 5.19, 5.21, 5.22, 5.23.

The results are grouped in three categories. Category 1 data are collected using Kinect V1.0 sensor, uses the Least Squares Line Fitting Method and Microsoft SDK skeleton for labelling. Category 2 data are collected using Kinect V1.0 sensor, the Geodesic Distance based labelling method and using only depth data. Category 3 data are collected using Kinect V1.8 sensor, uses the JAFKEC-G method and using only depth data.

Experiment Category 1

Sensor used: Kinect V1.0

Point cloud data of a healthy person and mock patient was collected using Microsoft Kinect V1.0 sensor. Data were simultaneously collected using the VICON system. During the experiment, the person walked 10 metres distance (right leg or left leg facing the Kinect) in moderate to slow speed.

Methods: Knee joint angles were calculated using the Least Squares Line Fitting Method and Microsoft SDK skeleton for labelling. The knee joint angle was compared with the VICON results. The reference convention is such that when the knee is not flexed at all, the angle is zero or close to zero (as opposed to 180 degrees), and it increases as the knee flexes.

Observation: The following plots show comparison of the knee angle as reported by Kinect against VICON results. The knee angle is the flexion angle made between the tibia and the fema. There are some jitters in the experiments 5.1, 5.4, 5.5 plots using Kinect. The Kinect sensor did not capture the first few frames that captured the initial few steps in the experiment 5.2 plot because the subject was out of range.

Experiment 5.1. Normal person slow walk (Right leg)

Experiment 5.2. Normal person slow walk (Left leg)

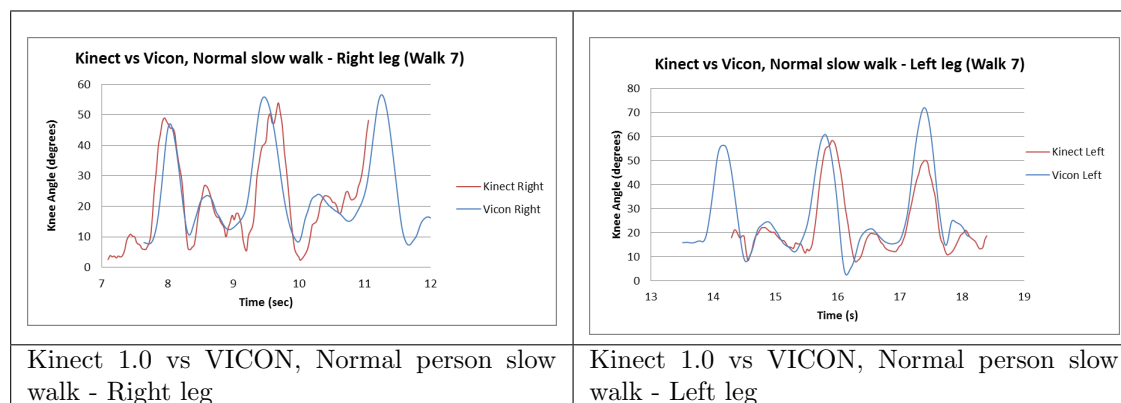


FIGURE 5.17: Normal person slow walk

Experiment 5.3. Normal person slow walk(Left leg) - Subject 2

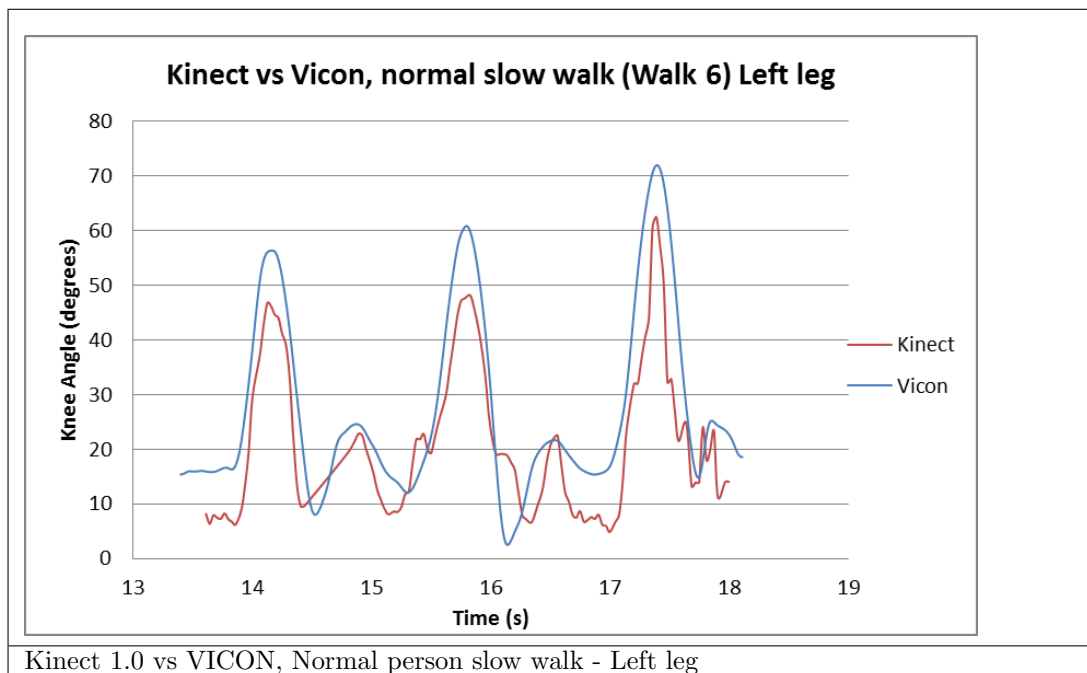


FIGURE 5.18: Normal slow walk, Left leg

Experiment 5.4. Patient type walk (Right leg)

Experiment 5.5. Patient type walk (Left leg)

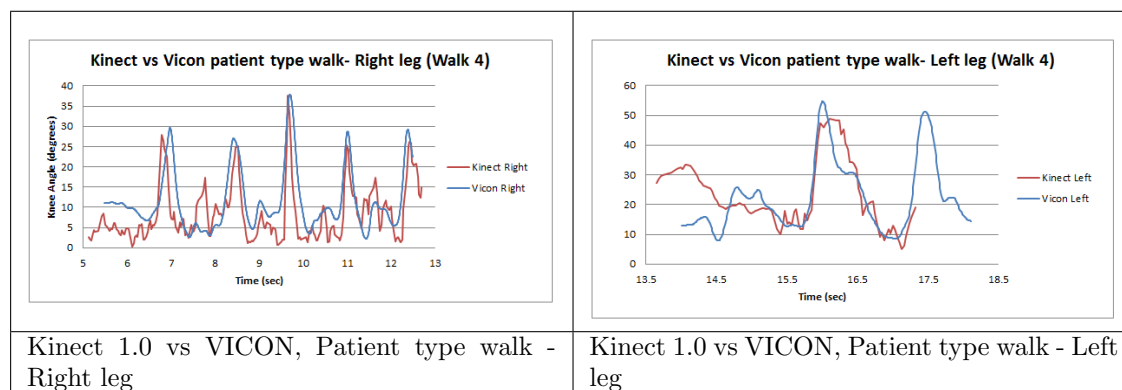


FIGURE 5.19: Patient type walk

Statistical Results:

Average angle error for this set-up (i.e. using Kinect 1.0 with Least Square line fitting from skeletal data) is **6.765 degrees**, using data from experiments 5.1, 5.2, 5.3, 5.4, 5.5. The error distribution (as probability density function) is given in Figure 5.20.

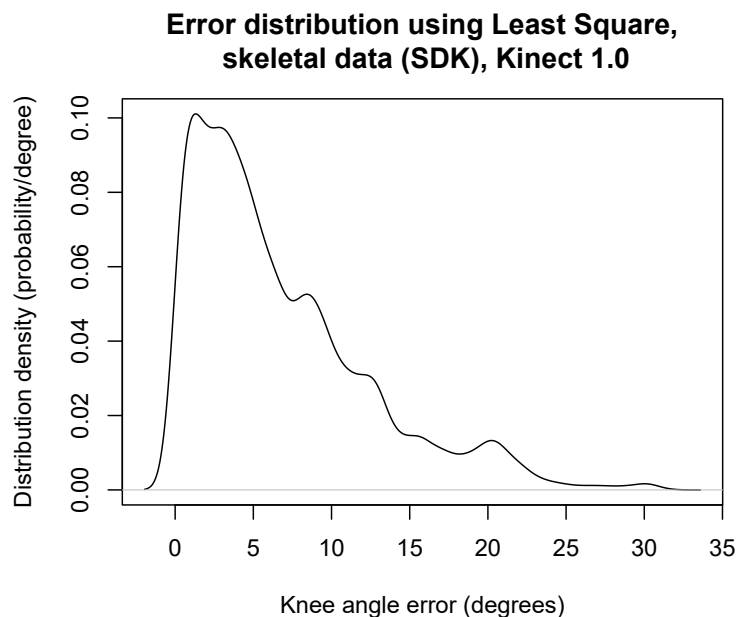


FIGURE 5.20: Error Distribution using Kinect 1.0, SDK Skeletal data and Least Square Method

Conclusion:

These are not good results. The calculated average knee joint angle error is **6.765 degrees**. All the above experiments exhibited over 5 degree difference with respect to the gold standard VICON. This shows that the Least Squares Line Fitting Method together with Microsoft SDK skeleton for labelling are not sufficient to accurately infer joint angles. In particular the labelling inaccuracy is the biggest source of error. The fitting/principal-axis algorithm works pretty well, but the labelling (i.e. limb classification) errors is what causes the long tail of the error distribution.

Experiment Category 2: Using Kinect point cloud data only, not using the Kinect SDK skeletal data.

Sensor used: Kinect V1.0

Using the point cloud data collected in experiments 5.1, 5.2, 5.5, 5.4 (Point cloud data of healthy person and mock patient) and using different method for Kinect data processing.

During the experiments, the person walked 10 metres distance (right leg or left leg facing the Kinect) in moderate to slow speed. Kinect data were compared with VICON data that were simultaneously collected.

Methods: Knee joint angles were calculated using the Geodesic Distance based labelling, using only depth data. The knee joint angle was compared with the VICON results. The reference convention is such that when the knee is not flexed at all, the angle is zero or close to zero (as opposed to 180 degrees), and it increases as the knee flexes.

Observation: The following plots show comparison of the knee angle as reported by Kinect against VICON results. The knee angle is the flexion angle made between the tibia and the fema. There are some jitters in the experiments 5.6, 5.8, 5.9 plots using Kinect. The Kinect sensor did not capture the first few frames that captured the initial few steps in the experiment 5.8 plot because the subject was out of range.

Experiment 5.6. Normal person slow walk (Right leg)

Experiment 5.7. Normal person slow walk (Left leg)

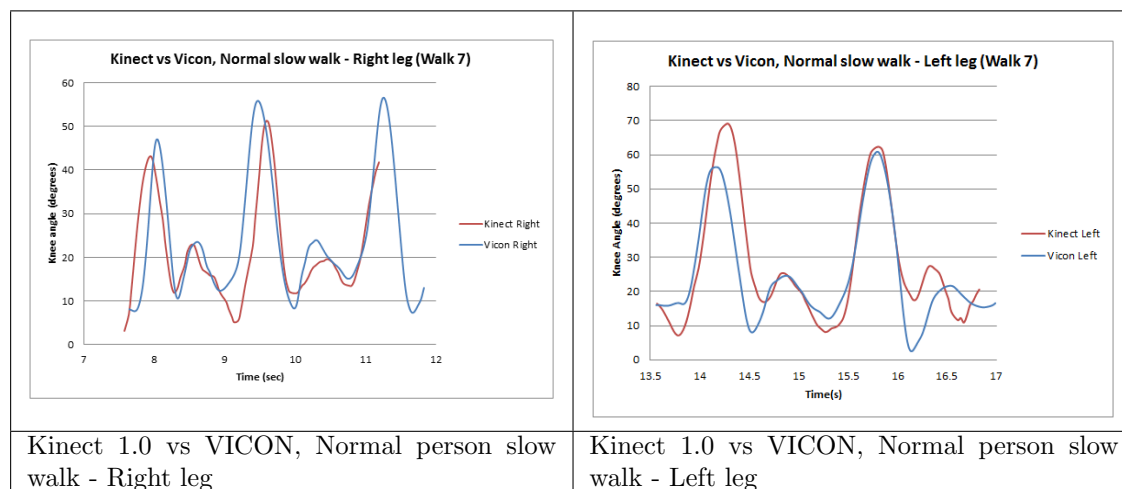
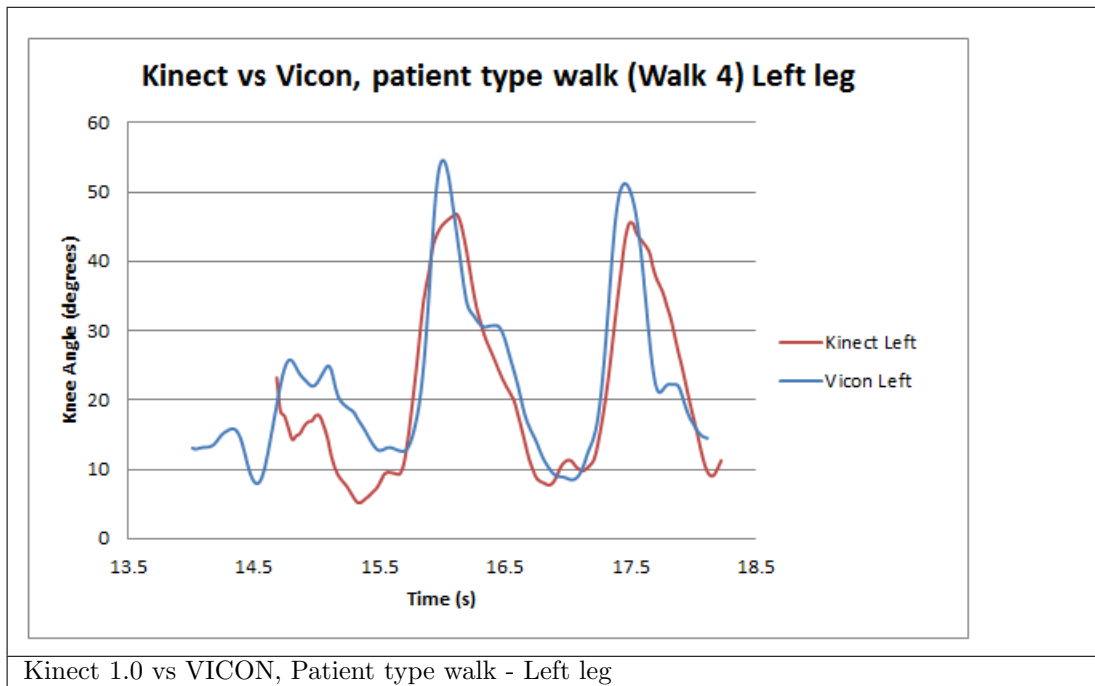


FIGURE 5.21: Normal slow walk, (Geodesic)

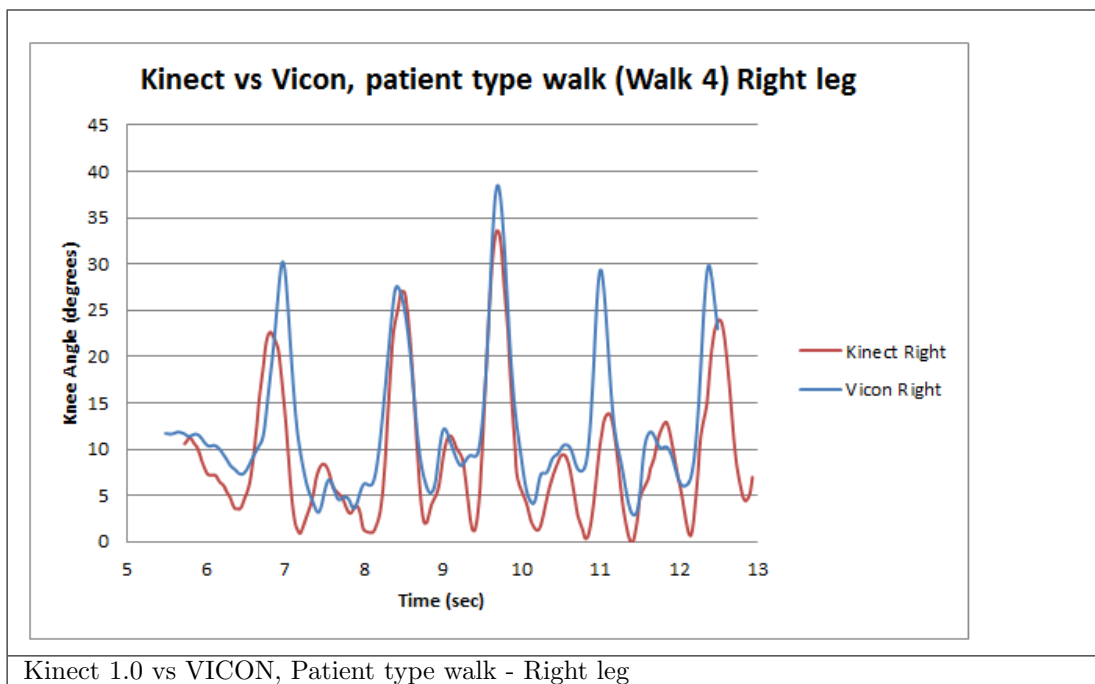
Experiment 5.8. Patient type walk (Left leg)



Kinect 1.0 vs VICON, Patient type walk - Left leg

FIGURE 5.22: Patient type walk - Left leg, (Geodesic)

Experiment 5.9. Patient type walk (Right leg)



Kinect 1.0 vs VICON, Patient type walk - Right leg

FIGURE 5.23: Patient type walk - Right leg, (Geodesic)

Statistical Results:

Average angle error using Kinect 1.0, Geodesic Distance and the point cloud data only (not the skeletal data) is **5.46 degrees**, using data from experiments 5.7, 5.6, 5.8, 5.9. The error distribution (as probability density function) is given in Figure 5.24.

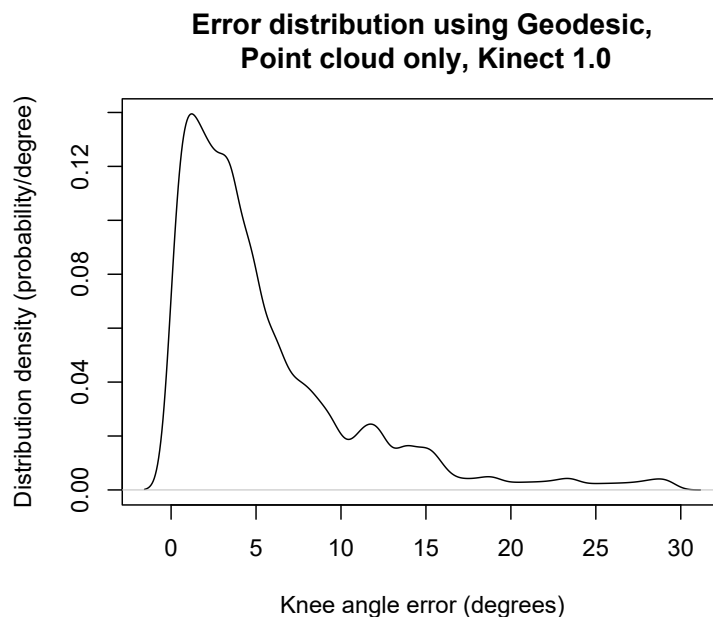


FIGURE 5.24: Error Distribution using Kinect 1.0, Geodesic Distance, Point Cloud data only

Conclusion:

The Kinect based angles have improved over the previous experiment due to the introduction of Geodesic distance based labelling of the depth data (i.e. ignoring the Kinect SDK skeleton). The calculated average knee joint angle error, **5.46 degrees**, is much better than the Kinect skeleton based method used in experiments 5.17 and 5.19. The average has improved from **6.765 degrees** to **5.46 degrees**. This new method shows a significant relative improvement over the method used in Category 1, but are not really good results, especially if one examines the error distribution shown in Figure 5.24. All the above experiments exhibited over 5 degree difference with respect to the gold standard VICON. This shows that the Geodesic distance based labelling is not sufficient to accurately infer joint angles.

Experiment Category 3

Sensor used: Kinect V1.8

Point cloud data of a few healthy people (male) were collected using Microsoft Kinect V1.8 sensor. Data were simultaneously collected using the VICON system. During the experiment, the person walked 10 metres distance (right leg or left leg facing the Kinect) in moderate to slow speed.

Methods: Knee joint angles were calculated using the JAFKEC-G for Gait system, described in flowchart 5.16. The algorithms used were a combination of Geodesic distance based labelling, shortest path algorithm, 3D convex hull, Least-square line fitting and Sectioning of point cloud 5.1. The knee joint angle was compared with the VICON results. The reference convention is such that when the knee is not flexed at all, the angle is zero or close to zero (as opposed to 180 degrees), and it increases as the knee flexes.

Observation: The following plots show comparison of the knee angle as reported by Kinect V1.8 against VICON results. The knee angle is the flexion angle made between the tibia and the fema. Kinect did not capture the first few frames that captured the initial few steps in the right leg plot because the subject was out of range.

Experiment 5.10. Healthy person (Subject 1) normal walk (Right leg)

Experiment 5.11. Healthy person (Subject 1) normal walk (Left leg)

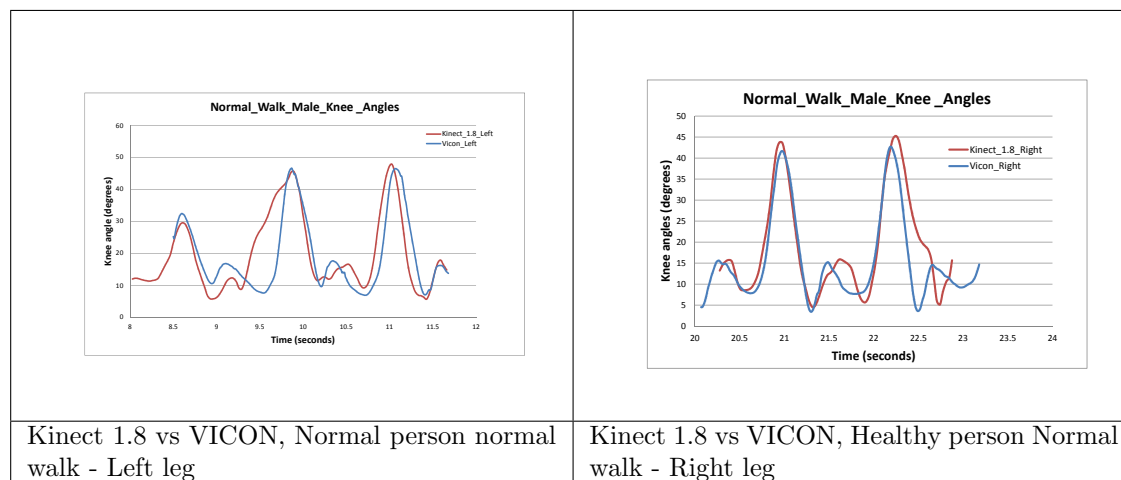


FIGURE 5.25: Healthy person, Subject 1 - Normal walk

Experiment 5.12. Healthy person (Subject 2) normal walk (Right leg)

Experiment 5.13. Healthy person (Subject 2) normal walk (Left leg)

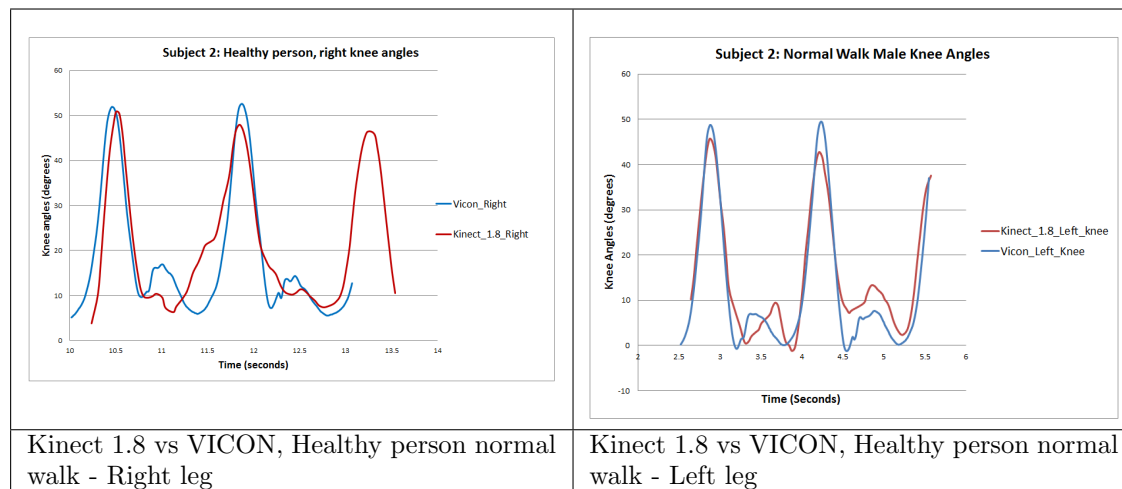


FIGURE 5.26: Healthy person, Subject 2 - Normal walk

Experiment 5.14. Healthy person (Subject 3) normal walk (Left leg)

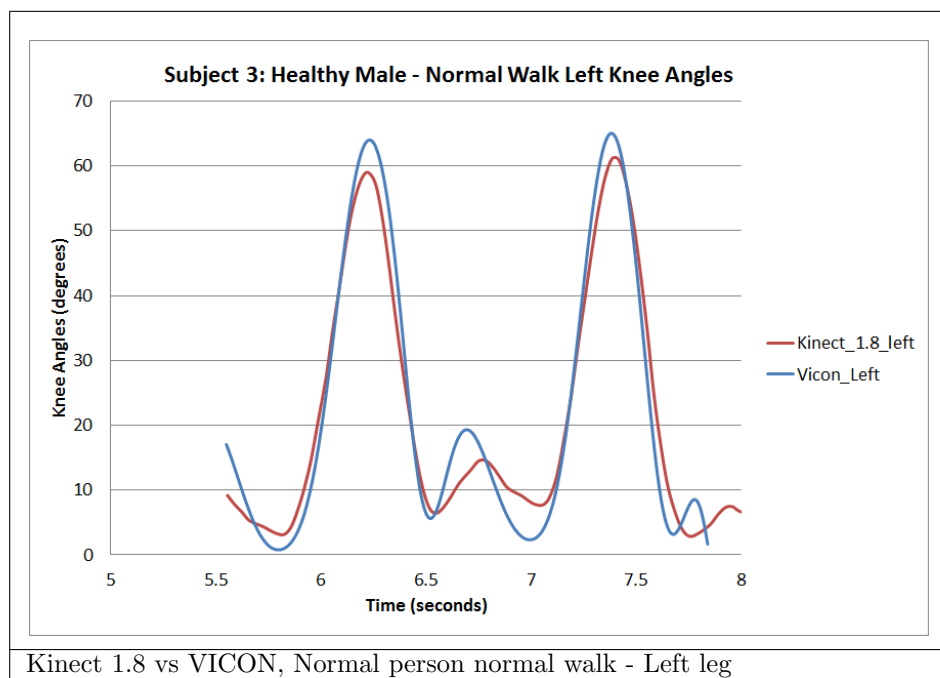


FIGURE 5.27: Healthy person - Normal walk, Left leg

Statistical Results:

Average angle error using Kinect V1.8, JAFKEC-G and the point cloud data only (not the skeletal data) is **4.974 degrees**, using data from experiments 5.10, 5.11, 5.12, 5.13, 5.14. The error distribution (as probability density function) is given in Figure 5.28.

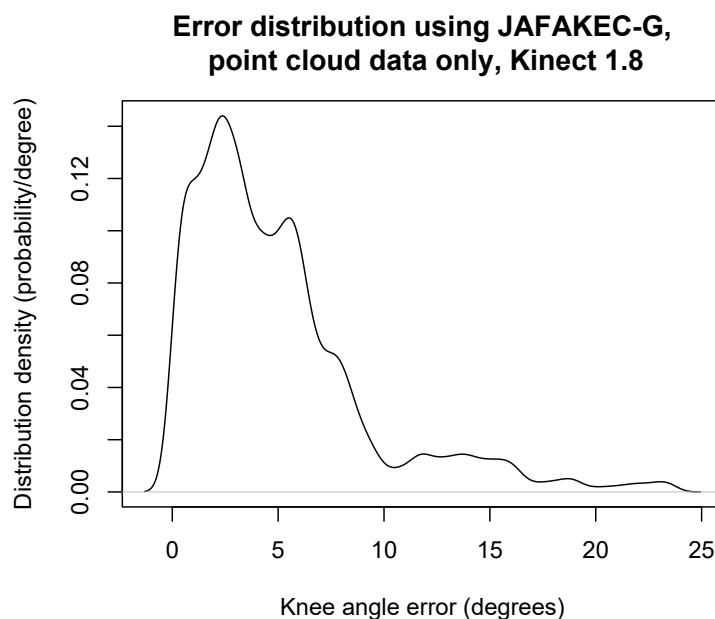


FIGURE 5.28: Error Distribution using Kinect 1.8, JAFKEC-G, Point cloud data only

Conclusion:

JAFKEC-G made some algorithmic improvements and as a result it was possible to get a smoother gait angle curve, and the average error was reduced to just under 5 degrees. Firstly JAFKEC-G uses a point-cloud of higher resolution as compared to Kinect V1.0 which was used for previous experiments (described in Section 3.4 and table 3.3). Although there is a large tail in the error distribution due to labelling errors, the main hump in the distribution is centered around 3 degrees. This is promising because if the the labelling errors can be eradicated, the remaining error distribution would be well below 5 degrees.

Experiment Category: Best case, Experiment 5.14

Sensor used: Kinect V1.8

Statistical Results:

Average angle error using Kinect V1.8, JAFKEC-G and the point cloud data only (not the skeletal data) of the best case, experiment 5.14, is **3.811 degrees**. The error distribution (as probability density function) is given in Figure 5.29.

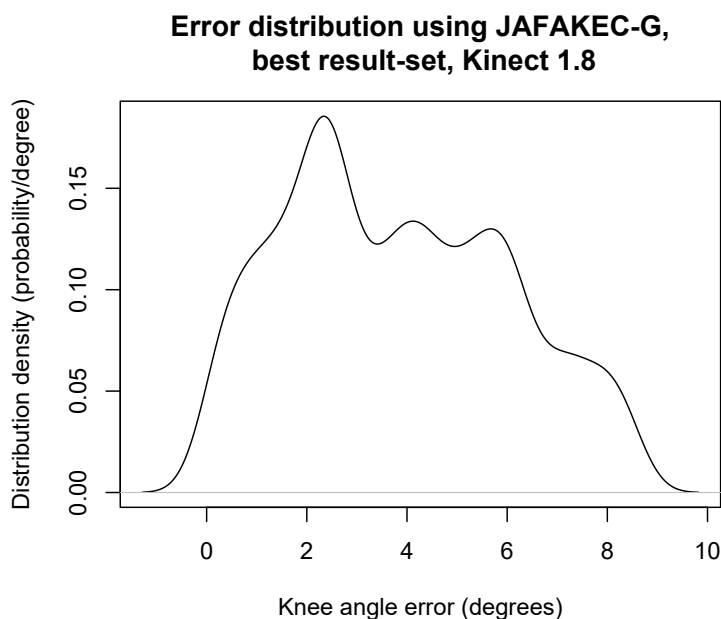


FIGURE 5.29: Error Distribution using Kinect 1.8, JAFKEC-G, best result-set

Conclusion:

The previous results were arbitrarily chosen from a recording session, however this particular result was obtained by choosing the best case from the JAFKEC-G recordings. The algorithms remain the same as the previous one, but due to low incidence of labelling errors, the average angle error was about **3.811 degrees**. This shows that if the limb labelling errors can be reduced, the JAFKEC-G using Kinect V1.8 sensor can measure knee angles within the required accuracy level.

5.7 Conclusion and Discussion

The joint angles obtained from the skeleton computed by Microsoft Kinect SDK were too inaccurate for clinical use, but the depth frame is shown to achieve better accuracy in capturing the gait cycle of a subject walking at a distance of about 3m from the Kinect sensor. An assortment of geometric algorithms have been used to improve the accuracy of gait capture. The VICON system, on the other hand, uses markers and captures the reflection from markers to work out the joint angles. Figure 5.25 shows that the accuracy of Kinect 1.8 results using the proposed JAFKEC-G for gait system is within a range of 5 degrees from the angle computed by VICON. The error in measurement is mainly because of Kinect sensors inability to properly capture the extension and flexion peak amplitudes due to slower and constantly fluctuating sampling frequency between 30 to 37 Hz and could not be stabilized. Kinect missed 8 to 18% of the steps on average recording and appeared to be difficult when the knees are crossed. Fernandez et al.[55] found that the knee angular errors in Kinect ranges between 6.78 degrees and 8.98 degrees when compared with VICON. Kinect based algorithms in the current state of the art can be used to measure general trends and parameters of gait or posture but not for clinical-grade continuous motion capture [127, 169]. Xu et al. [197] also presents a similar pessimistic result that Kinect-based tracking struggles to obtain accuracy of joint angles and that although Kinect V2 improves substantially over V1 in terms of resolution, the joint angle tracking is not improved significantly.

The aforementioned pessimistic results were based on Kinect SDK's joint detection process, whereas JAFKEC-G takes a fresh approach - using a new suite of mathematical algorithms for marker-less gait capture. Using the proposed algorithms, the joint angles so calculated in JAFKEC-G are significantly better than that of Microsoft SDK. During the swing phase knee angles in the gait cycle are getting computed accurately but during the stance phase there still some room for improvement. The Kinect V2 device has the potential to be implemented in the clinical setting as a tool to measure human gait accurately after future improvements [169] and establishment of standardised methods are required.

This chapter demonstrated the potential of low-cost ubiquitous sensing technologies in assessment of human gait problems arising from neurological issues. This work explores the feasibility of Kinect for clinical assessments, but there is need for further improvements before full clinical adoption can be attempted. To make a transition from a research project to a full-blown clinical application, it would be necessary to organize collaborative work involving imaging technologists and clinicians. That would help understand the salient features sought by clinicians and establish a new vocabulary of features afforded by the new technologies. There is a continuum between completely objective measurements and purely subjective opinions. Objective measurements are hard to develop, but once fully developed, has the advantage of being robust, accurate and fast. Whereas subjective assessment by a clinical expert aligns better with the traditional clinical practice. A collaborative approach should be directed towards establishing a mix or middle-ground between these two extremes, so that some aspects are measured as numerical values on which the subsequent assessment is based. Imaging is one area in which clinician community has fast embraced latest technology, because of the immense efficiency boost (i.e. speed-up) it had to offer. In the past X-ray and tomography plates used to take weeks to pass through the radiology departments due to the time consuming manual assessment involved. With the advent of digital

image processing systems, the assessment is heavily assisted by computer algorithms. On the one hand while this has dramatically reduced the assessment time, on the other hand it has transformed the profession of clinical radiology. Previously a radiologist would be expected to have a full mental model of the case, which they would discuss with surgeons and diagnosticians but nowadays this very human exchange is replaced by quickly produced robotic specification. As a result, the overall human attention given to each individual case has reduced to much lower levels, which has its downsides. Being a more economical arrangement its advent is unstoppable, but it has made the clinical practice quite a bit less soulful. If a system like JAFKEC-G achieves full-blown adoption in clinical practice, it would be important to observe the long term effects it has on the practice. Large data volume, and speed of processing is not necessarily the best direction when it comes to matters of healthcare. It always has to be weighed against the downside arising from shallower human involvement. In the present healthcare infrastructure, gait assessment is primarily a clinic based or lab based procedure. This entails that the patient has to be physically present in the clinic or hospital for the assessment to take place. If the novel technologies could surmount this barrier and enable continuous, real-time monitoring at the patient's home or care-home facilities, that would allow for much better situational awareness. In such an arrangement, the patients or their carers can check for any deterioration as an everyday routine rather than requiring expensive hospital appointments just for the assessment. Clinical intervention may be scheduled only when some instability is noted in the continuous monitoring outcomes. The low cost of Kinect devices is a key enabler of this vision to become the reality one day, and hopefully the work presented in this chapter would be a useful stepping stone on the way to that goal.

Chapter 6

Upper Limb Motion Analysis

6.1 Introduction

Depth perception has greatly advanced the state of the art for several image recognition problems - including pose [160], object recognition [101] and hand tracking [81]. Hand tracking is used for recognizing and tracking objects, detecting hand-object interactions such as grasp and release, and recognizing actions using the intervals given by the hand events, detecting tremor, etc.

The Kinect sensor provides a rich point cloud out of which a discrete set of gestures are recognized in real time for gaming interactions. This work aims at a more continuous capture of hand motions towards enabling monitoring rehabilitation of patients at care homes or GP surgeries.

While in principle monitoring of the hand is very similar to gait monitoring, in practice the nature of the motion is very different. The gait has a rapid cyclic/periodic pattern of motion whereas hand motions are relatively one-off and can be much more varied than gait. A wide variety of motions are relevant to everyday tasks - like opening a jar, pouring out liquid, sipping a drink, locking a door, waving etc. This additional complexity made the upper-limb motion monitoring more challenging than gait. This called for a fresh approach to upper-limb motions - i.e. gait algorithms could not be re-used/re-purposed for upper limb.

This chapter evaluates the suitability of a suite of algorithms within the novel JAFKEC-U framework that could operate on point cloud data captured by the Kinect sensor for the purpose of upper limb motion analysis. This will help to support clinicians in assessing if a particular treatment plan is progressing well, monitoring if the patient is able to perform everyday tasks like drinking a cup of tea, which might be difficult for stroke impaired patients.

6.2 Gesture Design and hand tracking

Recent version of Kinect (Kinect 2) sensor is now enabling greater fidelity in terms of hand and finger motions. Also in Kinect 2, the higher resolution allows one to place the subject further away, and hence allows a greater field of view. Kinect 2 has high sample density on the surfaces which means that surface normal vectors can be estimated at each sampled point with higher

accuracy and normal vectors help with additional feature computation.

The registered point cloud has some scattered isolated points and small clusters in several frames. This is an undesirable artefact, and was absent in previous versions. Algorithms take longer to run due to the larger point-sets and use more RAM. Markers (used for VICON) are creating large holes and local distortions in the point cloud. This is causing incorrect normal vectors getting calculated near the markers.

6.3 Literature Review

An important aspect of motion assessment is to study how the body motions needs to coordinate in order to achieve adequate level of control required for performing everyday tasks. R. Munoz-Salinas et al. [125] evaluated the influence of depth information in the motion assessment process. They proposed the concept of depth silhouette as a mechanism to incorporate depth information for motion assessment using Hidden Markov Models (HMM), silhouette compression and concluded that use of depth silhouettes increases the recognition accuracy significantly. Dominio et al. [48] proposed an algorithm to combine multiple depth-based descriptors for hand (fingers and palm) motion assessment. They have implemented different features extracted from the depth data for capturing relevant properties of the hand gestures. Classical tracking approaches can be adopted if gestures in the lexicon only carry trajectory information, i.e., if the shape of hand does not convey extra information. The systems CONDENSATION [83] and CAMSHIFT [20] have been shown to successfully track gestures but these systems are susceptible to lose the tracked objects when the scene illumination changes or when occluded by new objects. Hand gesture is a sequence of hand postures connected by continuous motions. One of the most widely used techniques for motion assessment is HMM [13],[207], [120]. Ramamoorthy et al. [144] developed a HMM based real time dynamic motion assessment system with a static shape recognition system. They used a Kalman filter based hand contour tracker which uses both hand shape and motion pattern of the gesture for recognition. The algorithm has the ability to detect the start and end points of gesture sequences. Issues and drawbacks with HMM approach consist of trajectory spotting for gesture temporal segmentation and finding the optimal parameters set like initial probabilities. Derpanis [40] reviews Human Computer Interaction (HCI) vision-based hand gestures and concludes that feature extraction, classification method, and gesture representation should be improved to facilitate HCI. Hand location, angle and velocity features are combined in [205] using HMM. Hand is localized by skin-color analysis and tracked by connecting the centroid of moving hand regions. They compared the utility of the three features; location, angle, and velocity features for motion assessment, which are obtained from the coordinates in a gesture trajectory. It concluded that angular features are most effective, location and velocity features are ranked second and third respectively. Another widely used technique for object tracking is particle filters [7] where they have used algorithm with re-sampling and recursive filtering process. Particle filters [84] are very popular due to their ability to closely approximate complex real-world multimodal posterior densities using sets of weighted random samples. Black and Jepson [14] proposed a particle filter approach and CONDENSATION-based trajectory motion assessment algorithm. Perez et al. [137] have integrated a particle filter for color histogram tracking and enhanced tracking under complex background and occlusion, and then applied the

particle filter framework to multiple objects tracking. Oikonomidis et al. [81] have proposed a markerless model for tracking two interacting hands without objects. They have used Kinect sensor and achieved accurate and robust 3D hand tracking at 15Hz. Jiang et al. [87] suggested a number of heuristics for selecting Kinect-based gesture patterns specific to patients with upper extremity impairments. Optical flow, edges and shading information have been implemented together in [112] for articulated hand tracking. Romero et al. [148] presented a real-time non-parametric method that compares observed hand poses to a large database containing objects in the hand such as a cup or a ball. The method estimates hand pose against large occlusion of the hand from object and segmentation errors.

One of the common methods for hand segmentation without the use of depth information is the skin color maps where skin color-based segmentation suffers significantly when there are lighting changes. Oikonomidis [81] and Tang [180] combined skin color detector on the RGB image and clustering on the depth threshold to achieve better hand segmentation. They used Microsoft Kinect to overcome the ambiguities of color images. Tang [180] was able to recognize 'grasp' and 'drop' gestures with very high accuracy using Microsoft Kinect to capture images, depth data and a Support Vector Machine (SVM) that performed training and testing on the images and their corresponding features. Wachs et al. [191] discusses methods using generic algorithms, artificial neural network etc. in designing and modelling the hand motion assessment. They found that their motion assessment algorithm depends on the application domain, and environment of the user and gets effective results where there is uncertainty in the exact positions of fingers and hand. The article also discusses different applications controllable by hand gestures. Suk et al. [175] proposed a new method for recognizing isolated hand gestures and continuous hand gestures using a continuous video stream implementing a Dynamic Bayesian Network (DBN) model. The features utilized are direction codes for hand motion, the positional relation between face and hands and the positional relation between the two hands. Spatio-temporal position of continuous stream of hand motion is extracted and hand gesture data are temporally segmented into subsequences and modelled using a Finite State Machine (FSM) in [204]. The hand motion profile for each gesture was constructed to estimate the dominant motion from an image sequence that can be used as input to a robot to repeat the intended operations of the hand gestures. Davis and Shah [37] developed a Finite State Machine (FSM) model-based approach to recognize human-hand gestures. The FSM was used to decompose gestures into four qualitatively distinct phases of a hand gesture that occurs in a fixed order which were represented as a list of gesture vectors. The vector list was matched with the stored gesture vector models.

Lange et al. [102] proposed a Kinect-based interactive game-based rehabilitation tool for people with balance impairment. Lei et al. [107] used Microsoft Kinect depth data to detect and track hand and kitchen objects to recognize a set of seven kitchen activities like place, move, chop, mix, pour, spoon and scoop. Their system works well both in tracking objects and determining the start and end of actions and does the action recognition almost perfectly. Chang et al. [24] applied Kinect for physical rehabilitation. They developed a 3D virtual environment system called Kinerehab that assist therapists in their work with two young adult students with motor impairments. Instructions were given to the participants to perform a number of motions such as lifting arms upwards, to the front and sides. These motions were repeated several times using the

Kinerehab and without using it. The results shows that Kinerehab improve participants' motor functionalities significantly. Chang et al. [25] used Kinect for Cerebral Palsy (CP) rehabilitation. They showed that two adolescent participants with CP demonstrated high motivations for exercising with Kinect and significantly improved their performance for upper limb rehabilitation during the intervention phases. Hondori et al. [79] developed a system to spot and monitor patients at home settings of specific *activities of daily living* (ADL) with eating and drinking problems using sensor fusion (Microsoft Kinect and Sensor Fusion). Altanis et al. [4] proposed an approach called Kinems based on their study on a game called 'Uni.Paca.Girl' developed using Kinect. Experiment was conducted with two students suffering from gross motor skills problems and motor impairments. The students with motor disabilities were asked to perform some basic body motions in a therapeutic session and to repeat the same motions playing the game. They found that both the students enjoyed doing the exercises while playing the game and their performance of doing exercise improved remarkably. Gallo et al. [61] explored the benefits of contact-less technologies in a medical context such as in hospitals where sterility is extremely important. They have developed a user interface that allows users to navigate and visualize 3D medical images without touching a mouse or a keyboard. Microsoft Kinect was used to capture the gesture data and enabled for remote medical image exploration. They represented the hand as a point cloud. Hand and arm gestures were used to translate gesture into action.

While the aforementioned applications of the Kinect sensor attempt to use the device variously in the context of clinical applications, none of them take on the harder problem of actually calculating the joint angles from point cloud data. The presented contribution (JAFKEC-U) aimed to do just that - i.e. calculate upper limb motion in terms of a time-history of joint-angles, from the depth frame sequence captured by Kinect.

6.4 Arm Motion Monitoring

The objective of this part of the work is to develop algorithms for monitoring hand dexterity of patients with partial motion impairment, using Kinect V2.

During the course of the doctoral project, the Kinect technology went through improvement in resolution and a change of its underlying physics. By the time the upper-limb monitoring system was being developed, the Kinect 2 sensors had superseded the Kinect version 1.x series. It was clear from the big improvement in the resolution that the previous versions (Kinect 1) would soon be discontinued. Therefore despite already having developed a toolset (i.e., recorder, codec, etc.), it seemed wise to move on to the newer version i.e. Kinect V2.

For the upper-limb experiments, the typical frequency of hand tremor is 5-6 Hz, and can go up to a maximum of 9Hz [68]. Kinect 2 frame rate is 30 FPS, so in theory it is possible to pick up hand-tremors using Kinect 2. However, the approach taken in this work is to track the motion of the arm. In order to bring this tool to clinical practice, it would be necessary to define a baseline band against which deviations would be interpreted as degree of abnormality.

In sub-acute stroke patients, often the upper limb coordination gets disrupted, and it becomes important to monitor its progression of recovery through a regime of physiotherapy and medication. The proposed Kinect based tool could be used to monitor a typical *everyday task* of drinking from a cup. The Kinect SDK has a built-in game-quality (i.e. low quality) skeletal coordinate interface, and the objective, as with the gait monitoring tool, is to try clinical quality motion capture. The starting point for this computation is therefore the depth frame or point cloud, rather than the skeleton frame.

6.5 Experimental Methodology

A cylindrical cup was used in the experiment. The subject was required to lift the cup and carry it to his/her mouth, as if for sipping some tea. The cup and the arm would be labelled (i.e. registered) in the recorded frames. Depth images of healthy person were recorded using Kinect and VICON simultaneously in the bioengineering laboratory. The JAFKEC for upperlimb (JAFKEC-U) system is then used for visualising and analysing the gait from the depth frames captured by Kinect.

The first stage of JAFKEC-U is a recorder that stores a series of depth frames on computer's secondary storage (i.e. hard disk or SSD drive). This is shown in Figure 6.1.

The raw depth frames thus recorded encodes a set of 3D points sampled on the objects in front of the Kinect device. If the points were visualized on their own, they would predominantly look like silhouettes with a hint of depth, as shown in Figure 6.2. This form of point-clouds is processed further and annotated with the results of the processing.

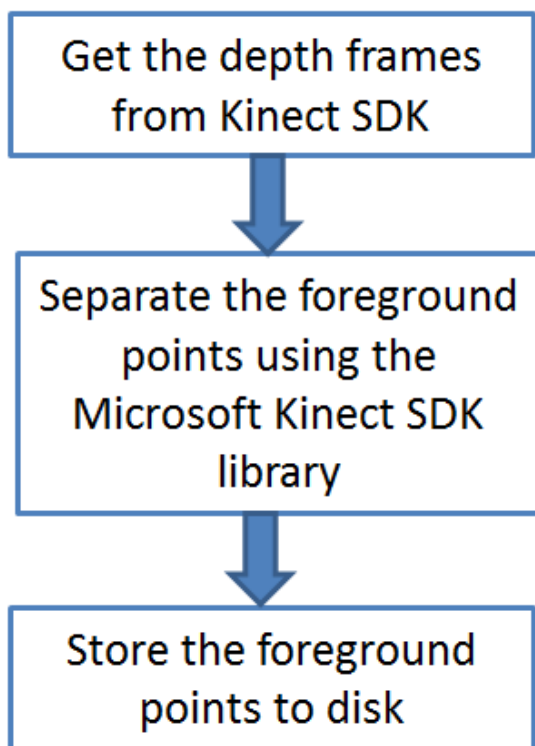


FIGURE 6.1: Kinect Recording Algorithm

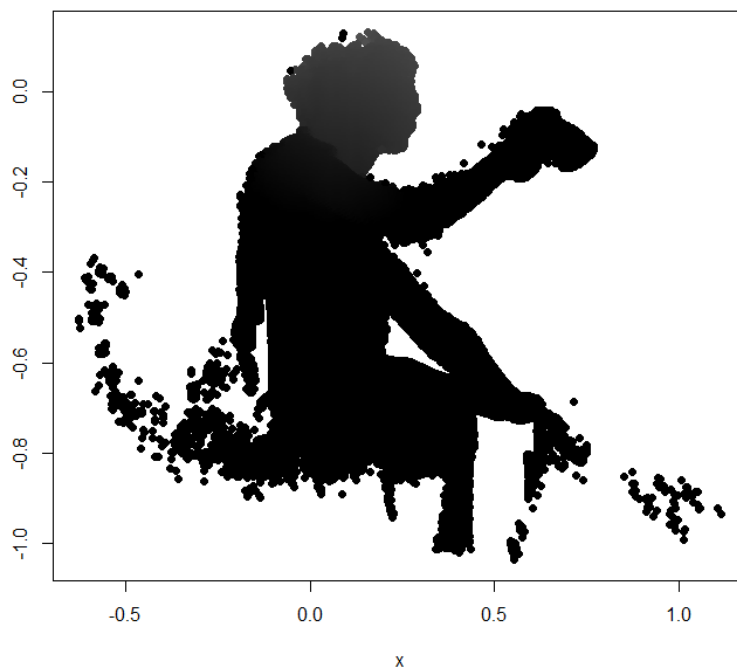


FIGURE 6.2: Depth map captured for upper limb

The following Figure 6.3 describes the flowchart of the novel JAFKEC system for upper limb.

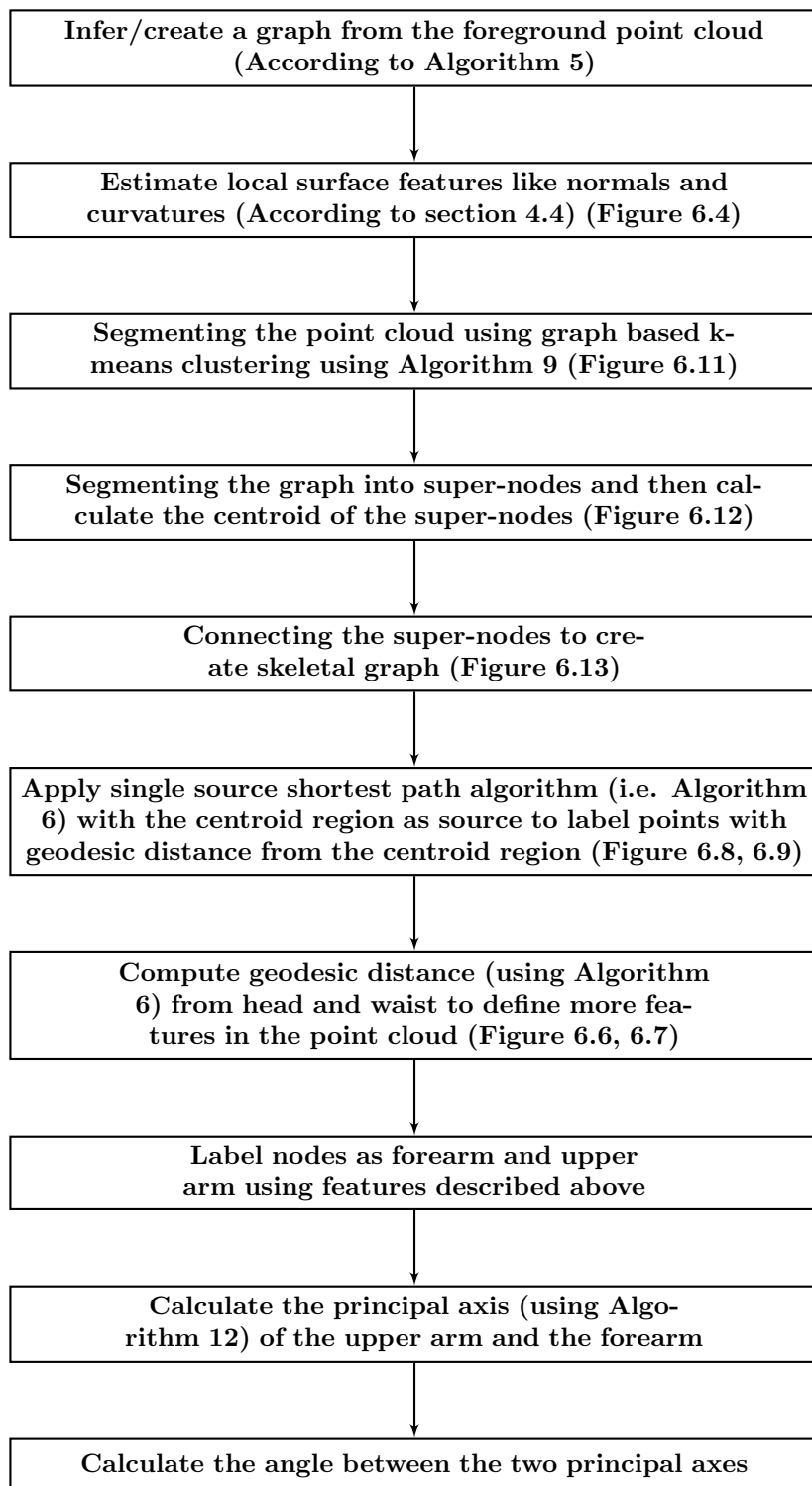


FIGURE 6.3: Flowchart for JAFKEC upper limb algorithm

6.6 JAFKEC for Upperlimb Methods

The proposed approach starts with an investigation of the achievable accuracy and suggests using some low-complexity algorithms, such as shortest path algorithm, K-lines clustering algorithm (described in section 6.7), least squares line fitting, segmenting the point cloud (described in section 6.7.3), and multiple geodesic distance labelling (described in Algorithm 6), (as listed in table 6.1) to improve the accuracy of joint positions. The captured depth frame from the Kinect recorder is compressed via run length encoding to allow storage of more frames in a limited memory. The stored point cloud is then used to perform hand-arm part labelling and pattern recognition in order to detect the limbs of interest. This process is followed by feature extraction via multiple geodesic distance labelling, shortest path algorithm and model fitting on specific subsets of points once the geodesic distances are attached to the points. A graph based clustering algorithm was used to segment and label the point cloud.

Algorithm	Application in JAFKEC-U
Geometric Graph Creation	Deciding the neighbourhood relationship treating each point as a graph node
Geodesic distance labelling	Forming a mesh representing the surface
Shortest path algorithm	Defining features of points for limb classification
Graph based geodesic distance	Labelling the point cloud with several Geodesic Distances
Graph based K-means clustering	Clustering the geometric graphs into sub-graphs
Segmenting the point cloud	Determining the key positions or point-set corresponding to the arm of interest.
Least-square line fitting	Determining the principal direction of a point-set corresponding to the limbs of interest.
Sectioning of point cloud	Segmenting into sub regions such as for specifying source regions for shortest path and for feature definition.

TABLE 6.1: Algorithms used in the JAFKEC Upperlimb system

With these features attached to each point, points were classified in the space of the features as belonging to the specific limb. Finally, the principal axes of the upper-limb was computed and the joint angles were calculated as angle between the principal axes.

6.7 Algorithmic Methodology

The sub-sections of this section describe the algorithmic methods used in upper-limb motion capture.

6.7.1 Normal Estimation

Normals computed using methods described in section 4.4 were useful in multiple ways. Firstly the normals were used in approximation of curvature on an edge, and the curvature in turn was used in modifying the edge-length function with the aim of snipping the graph at highly curved points. Normal values were also used shading the 3D rendering of the depth-frames.

The least-squares fitting process leaves a choice between two opposing directions for the estimated

normal (as mathematically expected). This ambiguity is resolved by the fact that the normal should not make an obtuse angle with the vector from the original point to the camera (or else the point wouldn't have been visible).

Figure 6.4 demonstrates the obvious effect of normal estimation - in that the point cloud could be shaded using a diffuse shading model.

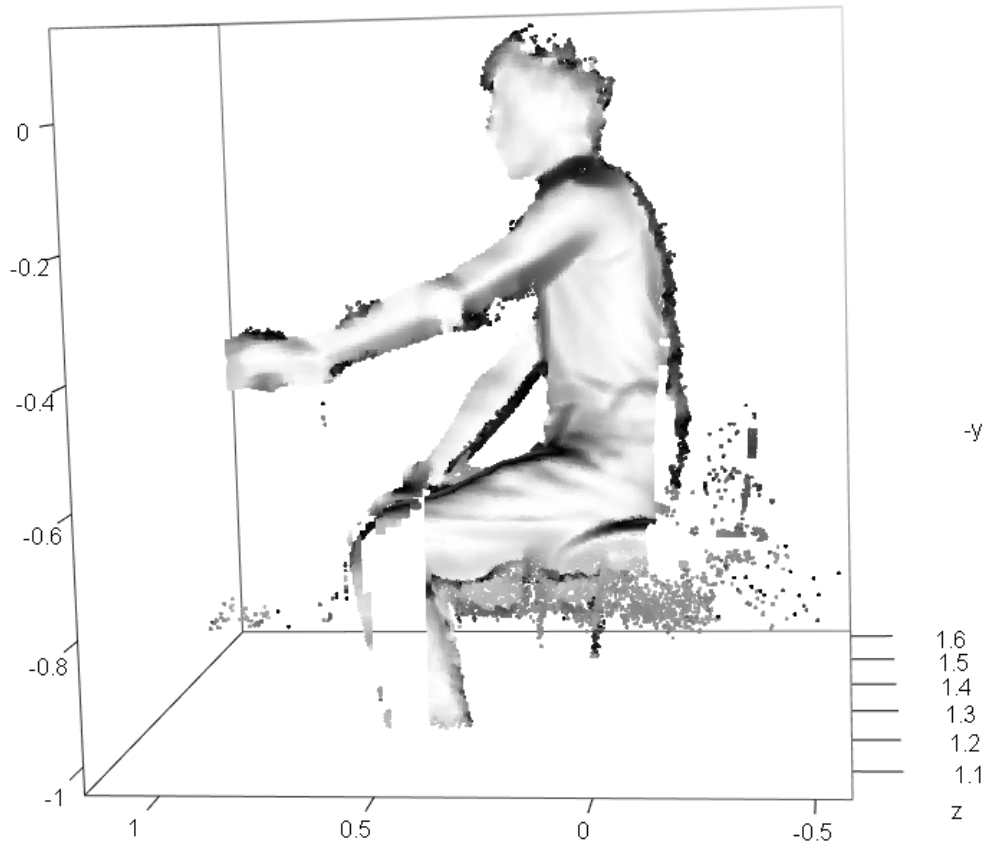


FIGURE 6.4: Point cloud with normals

6.7.2 Geodesic Distance Labelling

Labelling points with geodesic distances from designated sub-regions is useful in identifying key areas of the point-cloud. Once the following three key positions are identified, it becomes trivial to compute the joint angle in question.

- (a) wrist position
- (b) elbow position
- (c) shoulder joint position.

Easiest of the three is to classify the wrist and the cup based on geodesic distance. It is enough to use a simple linear classifier in the space of geodesic distance. For example, the heatmaps in Figure 6.5 shows geodesic distance measured from a few algorithmically specified regions. It can

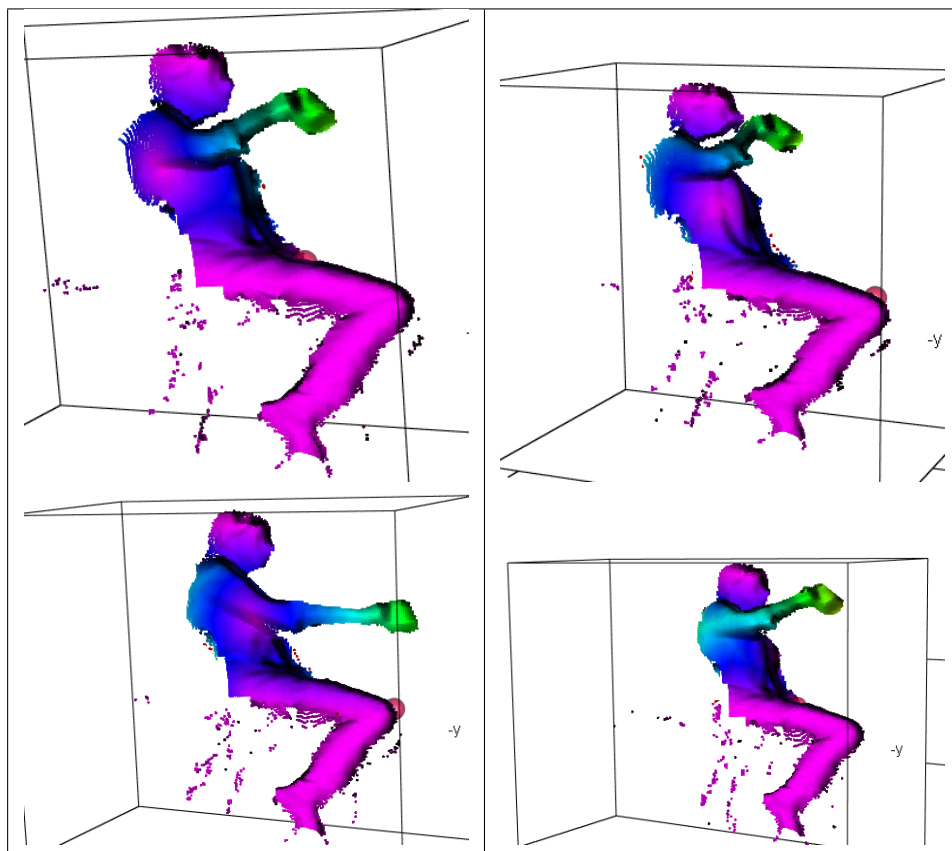


FIGURE 6.5: The cup-holding hand identified using geodesic distance

be seen that the hand sticks out unambiguously as the most distant entity from the specified sources. In the case of Figure 6.5 the chosen source region-set was the union of (a) the geodesic centroid, (b) lower half and (c) a small top slice of the head. Alternately the entire back side of the subject may be specified as the source to yield a similar linear classification of hand and cup.

Details of the geodesic distance labelling algorithm is given in section 4.6

Figures 6.6 and 6.7 show contour bands based on geodesic distance iso-lines. The boundaries between coloured bands in these figures are lines of constant geodesic distance from a source region. It should be noted how the distance flows along the surface form. The euclidean distance contours would not follow the surface shape and would be a family of concentric spheres centred at the source region.

Figures 6.8, 6.9 and 6.10 show some more cases of labelling parts of the upper-body with geodesic distance values.

6.7.3 Segmenting the point cloud

The upper-body point-cloud was segmented into geodesic clusters using the algorithm described in section 4.8. The segmentation follows the general schema of a k -means clustering algorithm except that the centroid computation and the proximity computation required in the clustering steps are based on measuring the distance along the edges, i.e. along the surface represented by

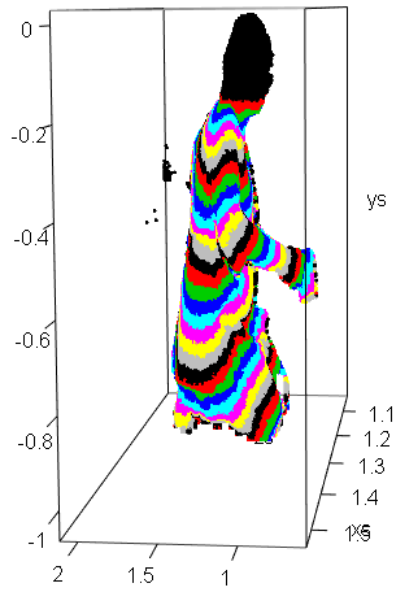


FIGURE 6.6: Geodesic Distance Bands with head as Source

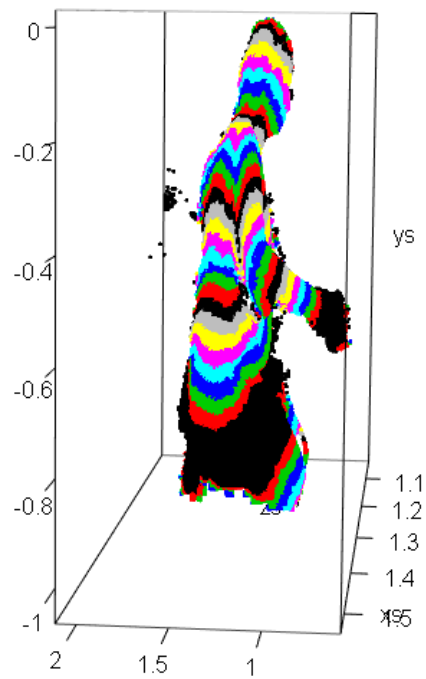


FIGURE 6.7: Geodesic Distance Bands with Waist as Source

the graph.

The clusters are then used for identification of certain designated clusters - i.e. shoulder, elbow, and hand. Figure 6.11 shows such a segmented depth-image, in which each cluster is given a different colour to help with visual inspection.

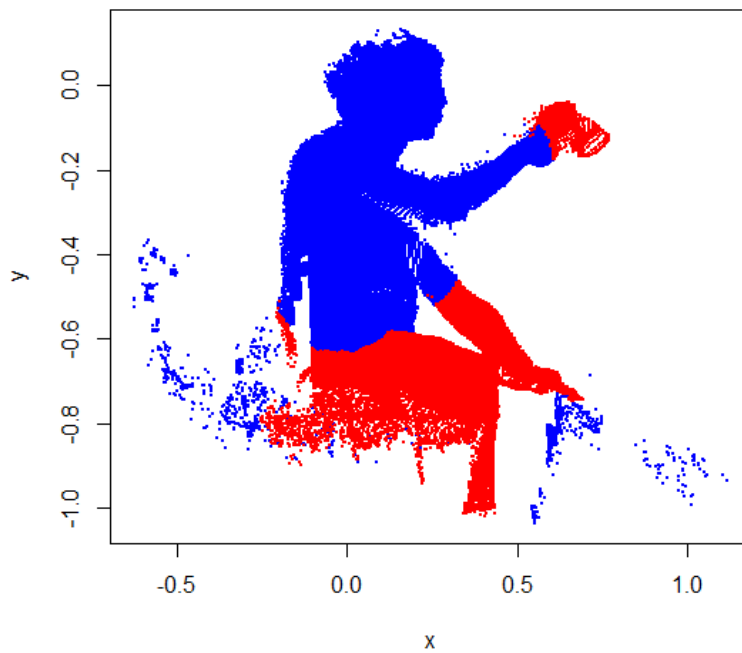


FIGURE 6.8: A distance discriminant based on geodesic distance

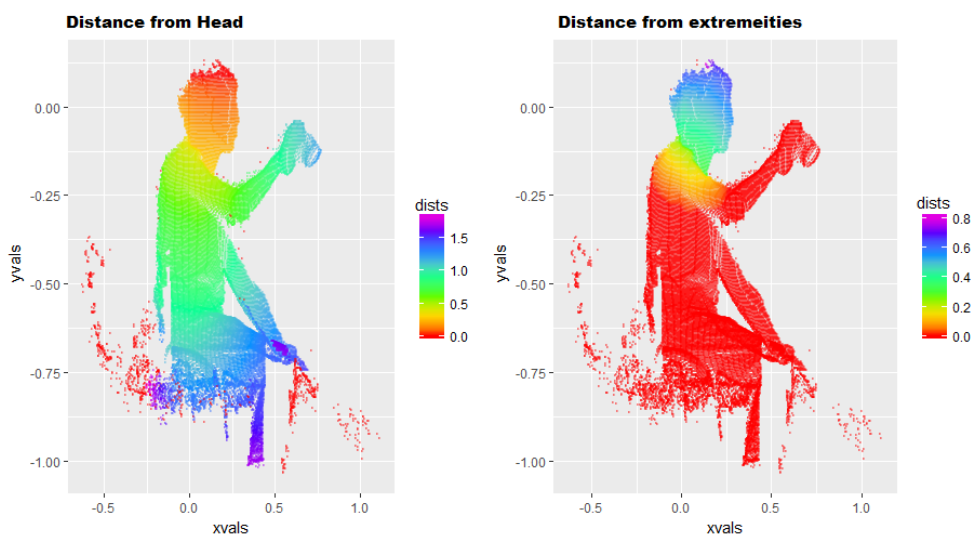


FIGURE 6.9: Demonstrating second order geodesic distance labelling

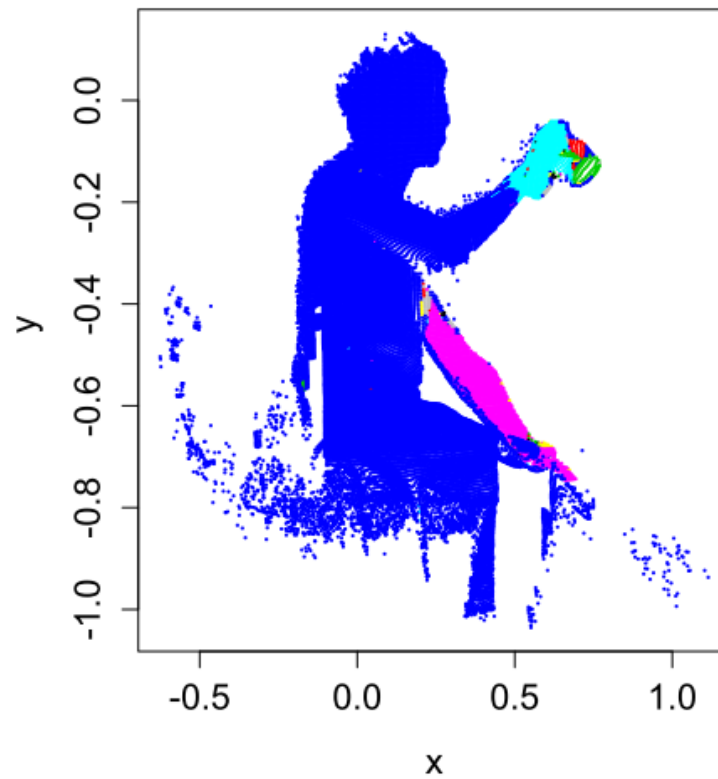


FIGURE 6.10: Arm motion monitoring



FIGURE 6.11: Segmenting the Point Cloud

6.7.4 Graph from Geodesic Clusters

The geodesic clusters have mutual neighbourhood relationships by bordering points being adjacent. Each cluster may be treated as a super-node in a high-level graph structure (as shown in Figure 6.12). Based on adjacency between clusters, one could then form a graph by joining supernodes corresponding to adjacent clusters (as shown in Figure 6.13).



FIGURE 6.12: Super nodes obtained from graph based clustering

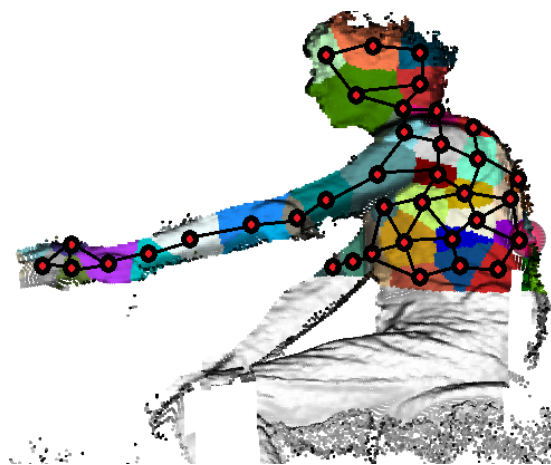


FIGURE 6.13: Connectivity graph for super nodes

Such a graph would be much smaller than the original geometric graph representing the surface and yet represent the overall topology of the object. This structure is amenable to further analysis.

6.8 Results

Angle measurements were found to have some noise in it. This noise originates from the noises in point cloud capture and artefacts of the algorithms used for measurement and feature extraction. The prevalent approach adopted for eliminating such noise is that of filtering. Kalman filters are commonly used for elimination of random uncorrelated noise. However, Kalman filtering did not make any difference to the quality of the results. Initially the suspicion was on incorrect implementation of the Kalman filter, but then the same Kalman filter code did work well to eliminate artificially added Gaussian noise (as shown in Figure 6.14). This suggests that the nature of the noise present in these results must not have been uncorrelated -like the additive Gaussian noise. These noise probably arises from configuration dependent artefacts of the capture and processing algorithms, and therefore bears some correlation with the measured signal. The noise was in the form of high frequency flickers, so low pass filtering seemed to be an option. A high (6^{th}) order Butterworth filter was thus used for filtering as is used for VICON systems. Figure 6.15 shows a plot of the said Butterworth filtered elbow angle alongside the original noisy signal. An artefact of this filter is that the filtered signal has a lagging phase-shift, but that is not a problem as the absolute phase is not of any interest.

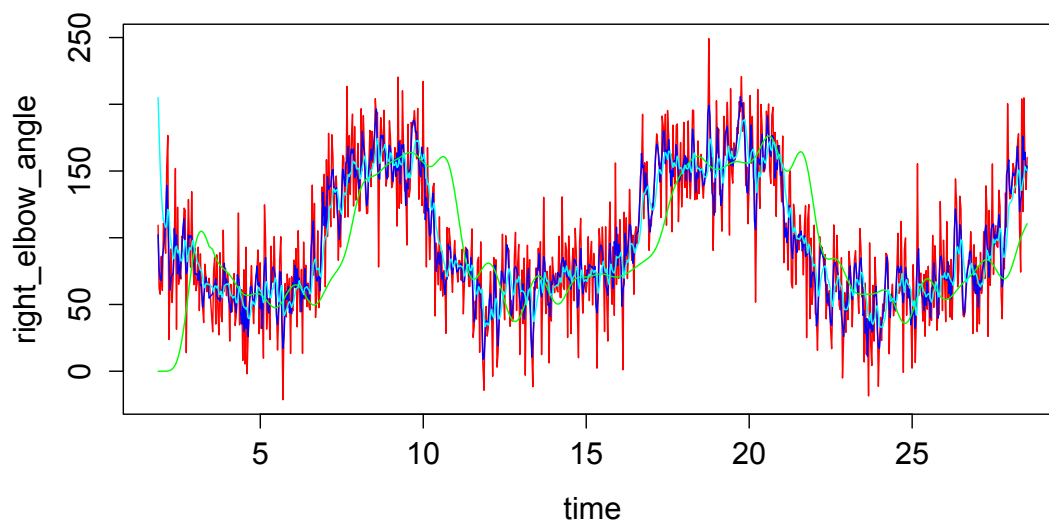


FIGURE 6.14: Kalman filter applied to eliminate added Gaussian noise

For clinical assessments, plots of joint angles against time may or may not be the best format for presenting the motion. Time-vs-variable plots are popular in engineering and physical sciences for motions but it is entirely possible that clinicians assess more effectively based on an animated representations, as is common in cardiac and foetal imaging. There is a wide range of formats in which the animation can be presented.

One way to represent/visualize the results would be in the form of a stick figure or a 3D skeleton model. Figure 6.16 shows such a stick-figure representation in which the joints are represented as spheres, that leave a trail of positions as the motion progresses, showing the trajectory taken

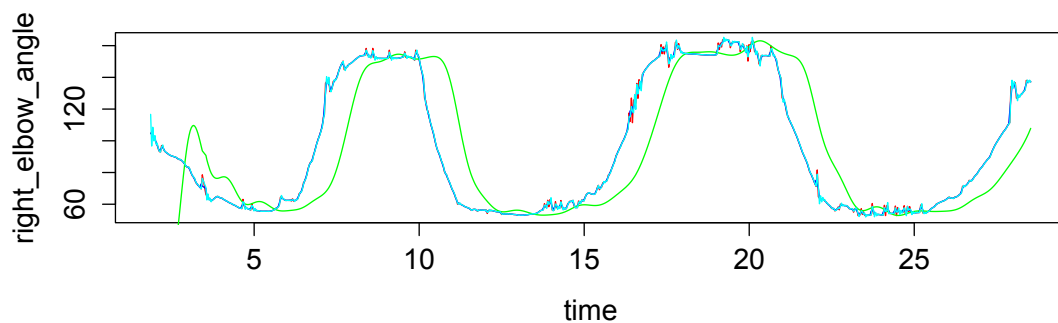


FIGURE 6.15: Unfiltered and Butterworth filtered plots of elbow angle

by the joint position in 3D space.

On the other hand, the animation could be that of a shaded point-cloud faithfully representing the shape, form, and motion of the patient without making it personally identifiable. The point cloud’s image of the face can be made completely unidentifiable by applying a Laplace smoother on the face region.

It is important that such an animated view is not a passive video but an interactive 3D view that can be rotated, zoomed, paused, and measured.

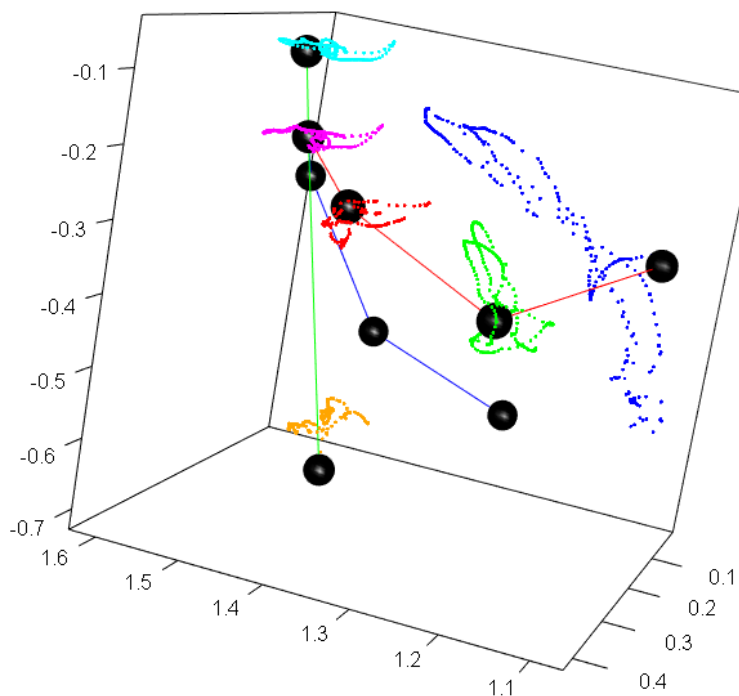


FIGURE 6.16: Upper body joint trajectories

6.8.1 Arm angle results of Kinect and VICON

As described earlier, VICON is an established clinical-grade motion capture system. It is based on tracking marker positions recorded using multiple video cameras. Reflecting markers are on the limbs of interest, and the motion is captured by VICON as video streams recorded by multiple digital video cameras. The post-processing software accurately estimates and tracks marker positions in 3D space. The post-processing software is agnostic as to what each marker is attached to. It is up to the user to specify a topology (i.e. connectivity structure) for the markers. Once the marker positions are captured, one can define joint labels thereby connecting the dots between the marker positions. The VICON cameras capture hundreds of frames per second, and the redundant array of multiple cameras help estimate the positions and hence angles extremely accurately. VICON is a very expensive piece of kit, but it can provide a standard to measure against. Some typical upper limb operation cycles using both VICON and Kinect V2 were recorded and plotted the corresponding joint angles estimated by the two devices on the same plots as separate data series. Some of such plots are given in Figure 6.17, 6.18, 6.19, 6.22, 6.23, 6.26. These combined plots show that the results of processing with JAFKEC-U on Kinect sensor V2 point cloud data are in close agreement with that from VICON.

Experiment Category - Healthy Male

Sensor used: Kinect V2

Five different experiments were carried out of two healthy males and Point cloud data of the healthy person's right arm motion were collected using Microsoft Kinect V2 sensor. Data were simultaneously collected using the VICON system. During the experiments, the subject lifted a cylindrical cup and carried it to his mouth, as if drinking some juice or sipping some tea. The right upperlimb was facing the Kinect, in moderate speed.

Methods: Elbow joint angles were calculated using the JAFKEC for Upperlimb system, described in flowchart 6.3. The algorithms used were a combination of geometric graph creation, geodesic distance based labelling, shortest path algorithm, graph based K-means clustering, segmenting the point cloud, least-square line fitting and sectioning of point cloud 6.1. The upperlimb joint angle was compared with the VICON results. The reference convention is such that when the elbow is not flexed at all, the angle is zero or close to zero (as opposed to 180 degrees), and it increases as the elbow flexes.

Observation: Figure 6.1, 6.2, 6.3, 6.4, 6.5 shows comparison of the right elbow angle as reported by Kinect against VICON results. The elbow angle is the flexion angle made between the upper arm and the forearm. In experiment 6.4, Kinect V2 failed to detect the subject in the range from 3.62 second and 5.00 second. This has caused the JAFKEC-U to have a smooth but deviated pattern that does not match with the corresponding VICON data. So, the Kinect V2 point cloud data in the range 1.02 seconds to 3.62 seconds and the range from 5.00 seconds to 8.39 seconds should be considered for matching with VICON data.

Experiment 6.1. Healthy male (Subject 1) drinking from a cup - trial 1

Experiment 6.2. Healthy male (Subject 2) drinking from a cup - trial 1

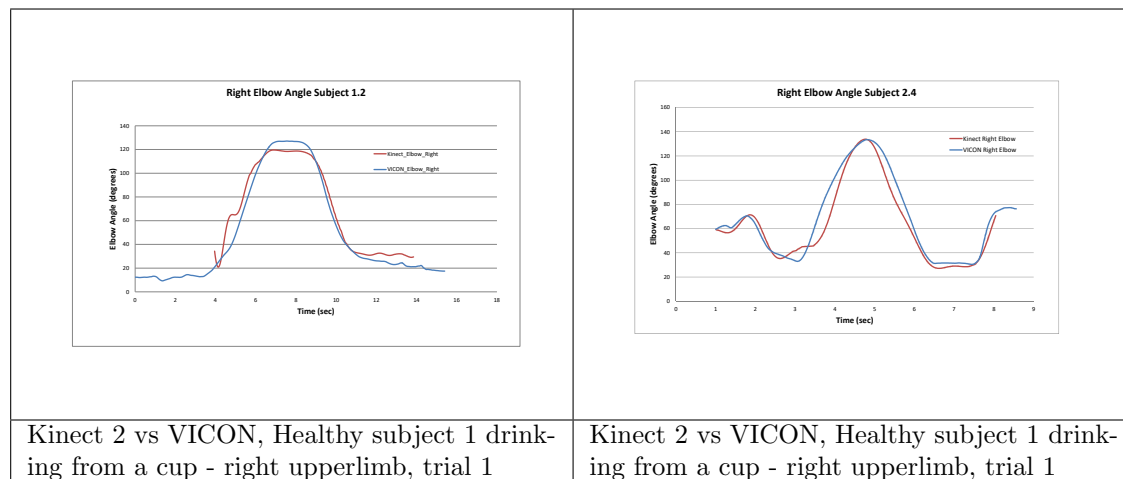


FIGURE 6.17: Elbow angle of healthy subject 1, healthy subject 2 (trial 1), Right upperlimb

Experiment 6.3. Healthy male (Subject 2) drinking from a cup - trial 2

Experiment 6.4. Healthy male (Subject 2) drinking from a cup - trial 3

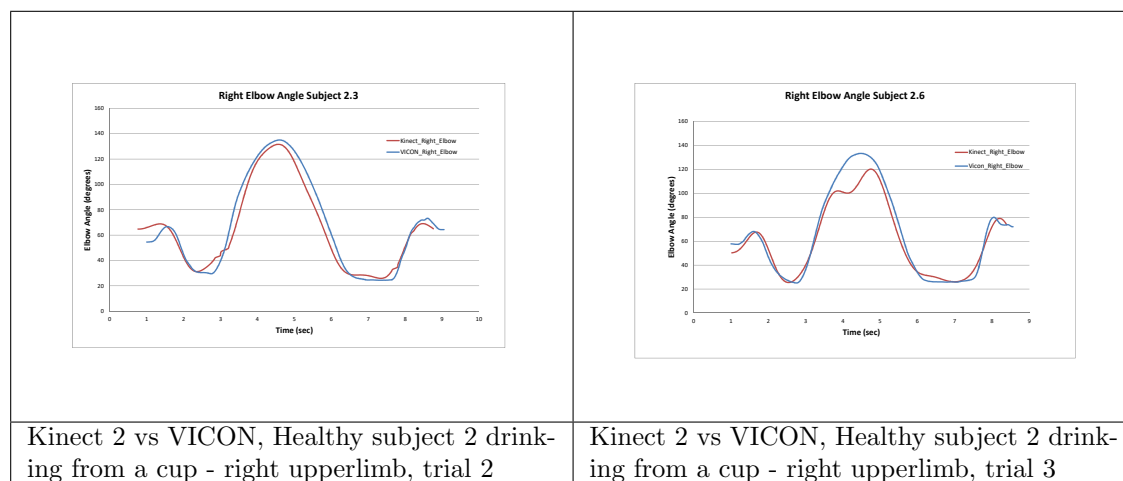
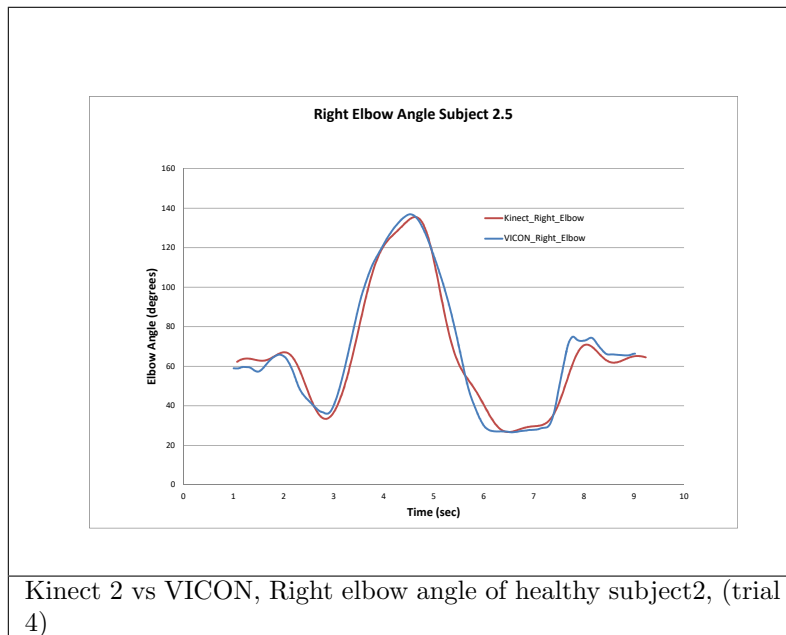


FIGURE 6.18: Elbow angle (trial 2, trial 3) of healthy subject 2, Right upperlimb

Experiment 6.5. Healthy male (Subject 2) drinking from a cup - trial 4



Kinect 2 vs VICON, Right elbow angle of healthy subject2, (trial 4)

FIGURE 6.19: Elbow angle of healthy subject2, (trial 4) Right upperlimb

Statistical Results: Healthy Male

Average angle error using Kinect V2, JAFKEC-U and the point cloud data is **6.14 degrees**, using data from experiments 6.1, 6.2, 6.3, 6.5. The error distribution (as probability density function) is given in Figure 6.20.

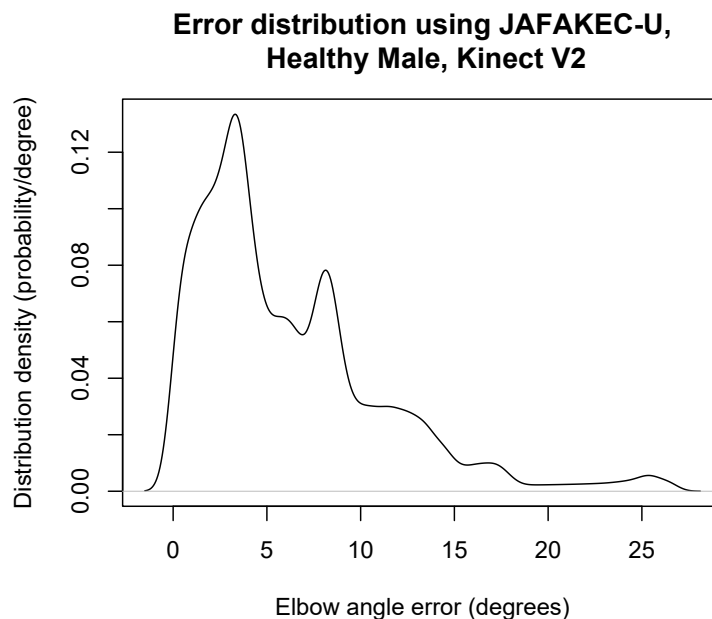


FIGURE 6.20: Error Distribution using Kinect V2, JAFKEC-U, Healthy Male

Conclusion:

In this case the average error for the elbow angle was 6.14 degrees - about 1 degree more than the desirable bound of 5 degrees. The error is defined to be the difference between JAFKEC-U and VICON results. The average error is only slightly above the desired margin, but the error distribution plot shows a substantial tail region running well above 10 degrees. This shows that the labelling algorithms need further work. The precision analysis in sections 3.10 and 3.10.1 promised much lower errors but that analysis focussed on the errors entailed by the point cloud's intrinsic accuracy and that of least-squares fitting algorithms. What it does not take into account is errors arising from incorrect labelling of points. Despite high resolution of the point cloud and a mathematically optimal fitting algorithm, it is possible to get bad results due to errors in classifying points. It was not possible to analyse precision bounds for labelling errors.

Experiment Category: Best case - Male, Experiment 6.5

Sensor used: Kinect V2

Statistical Results:

Average angle error using Kinect V2, JAFKEC-U and the point cloud data only of the best case, experiment 6.5, is **4.651 degrees**. The error distribution (as probability density function) is given in Figure 6.21.

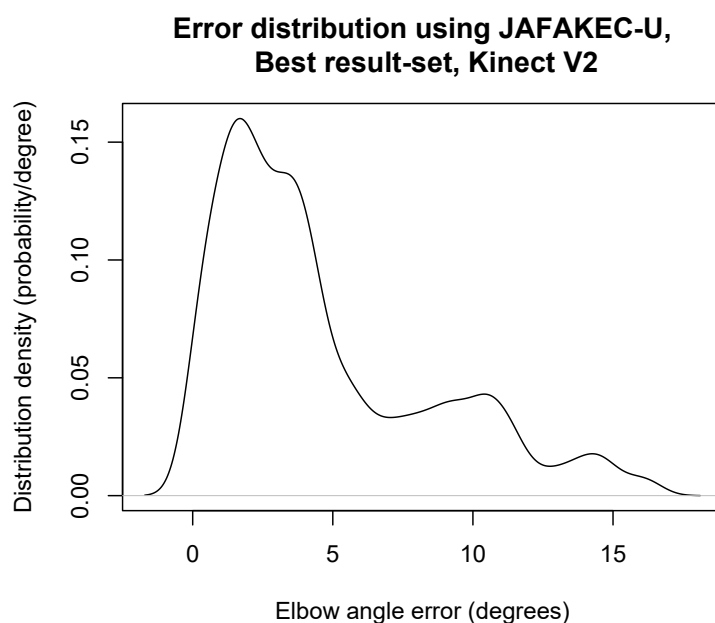


FIGURE 6.21: Error Distribution using Kinect V2, JAFKEC-U, best result-set

Conclusion:

In this case the average error (4.651 degrees) was well within the desirable margin. The corresponding time-series plots show close agreement between JAFKEC-U and VICON results. There is a tail region in the error distribution plot which is mainly attributable to labelling errors, indicating that the labelling algorithm needs further improvement.

Experiment Category - Healthy Female

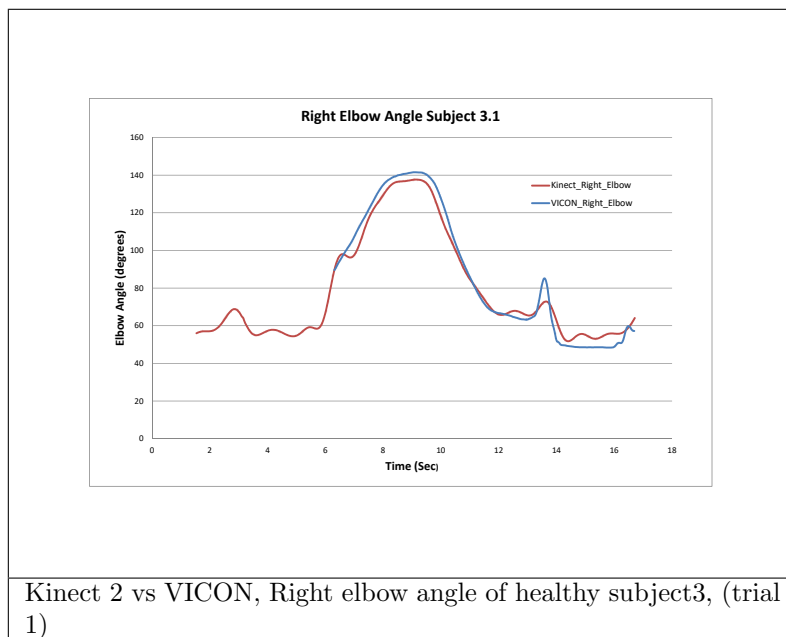
Sensor used: Kinect V2

Three different experiments were carried out and Point cloud data of the healthy person's right arm motion were collected using Microsoft Kinect V2 sensor. Data were simultaneously collected using the VICON system. During the experiments, the subject lifted a cylindrical cup and carried it to his mouth, as if drinking some juice or sipping some tea. The right upperlimb was facing the Kinect, in moderate speed.

Methods: Elbow joint angles were calculated using the JAFKEC for Upperlimb system, described in flowchart 6.3. The algorithms used were a combination of geometric graph creation, geodesic distance based labelling, shortest path algorithm, graph based K-means clustering, segmenting the point cloud, least-square line fitting and sectioning of point cloud 6.1. The upperlimb joint angle was compared with the VICON results. The reference convention is such that when the elbow is not flexed at all, the angle is zero or close to zero (as opposed to 180 degrees), and it increases as the elbow flexes.

Observation: Figure 6.6, 6.7, 6.8 shows comparison of the right elbow angle as reported by Kinect against VICON results. The elbow angle is the flexion angle made between the upper arm and the forearm. In experiment 6.6, VICON fails to detect few data points in the initial phase of recording.

Experiment 6.6. Healthy female (Subject 3) drinking from a cup - trial 1

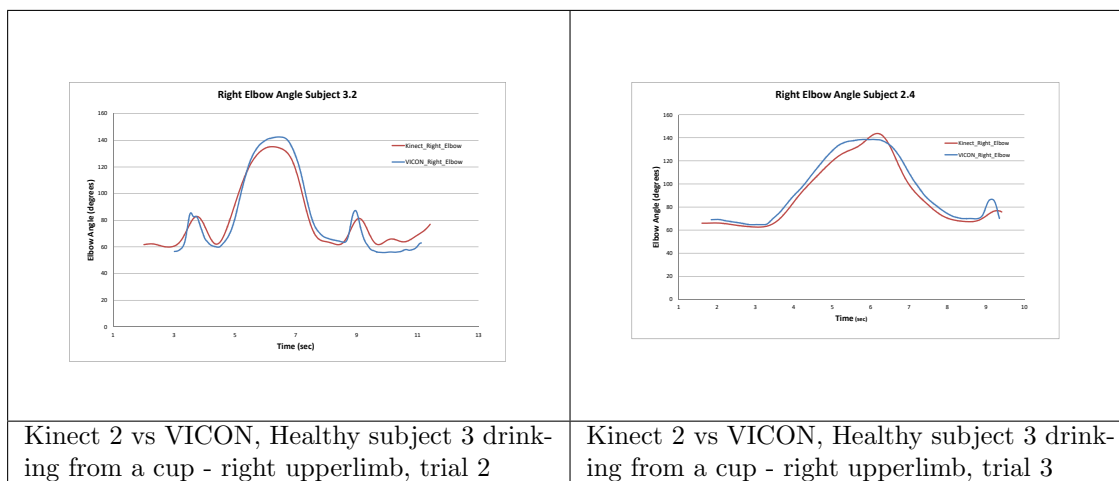


Kinect 2 vs VICON, Right elbow angle of healthy subject3, (trial 1)

FIGURE 6.22: Elbow angle of healthy subject3, (trial 1) Right upperlimb

Experiment 6.7. Healthy female (Subject 3) drinking from a cup - trial 2

Experiment 6.8. Healthy female (Subject 3) drinking from a cup - trial 3



Kinect 2 vs VICON, Healthy subject 3 drinking from a cup - right upperlimb, trial 2

Kinect 2 vs VICON, Healthy subject 3 drinking from a cup - right upperlimb, trial 3

FIGURE 6.23: Elbow angle of healthy subject3, (trial 2 and trial 3) Right upperlimb

Statistical Results: Healthy Female

Average angle error using Kinect V2, JAFKEC-U and the point cloud data is **5.18 degrees**, using data from experiments 6.6, 6.7, 6.8. The error distribution (as probability density function) is given in Figure 6.24.

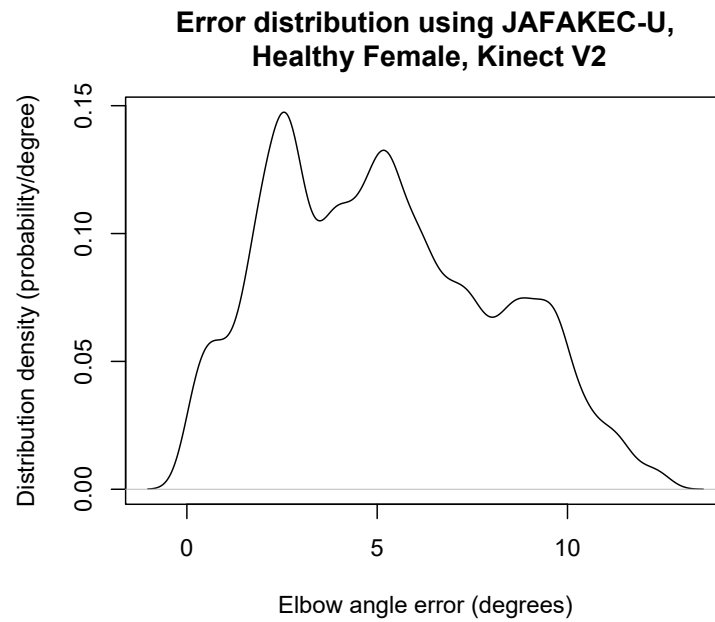


FIGURE 6.24: Error Distribution using Kinect V2, JAFKEC-U, Healthy Female

Conclusion:

In this case the average error is 5.18 degrees, which is only just above the desirable margin. The error distribution plot shows a long tail, which suggests that the labelling algorithms of JAFKEC-U needs further improvement.

Experiment Category: Best case - Female, Experiment 6.6

Sensor used: Kinect V2

Statistical Results:

Average angle error using Kinect V2, JAFKEC-U and the point cloud data only of the best case, experiment 6.6, is **4.43 degrees**. The error distribution (as probability density function) is given in Figure 6.25.

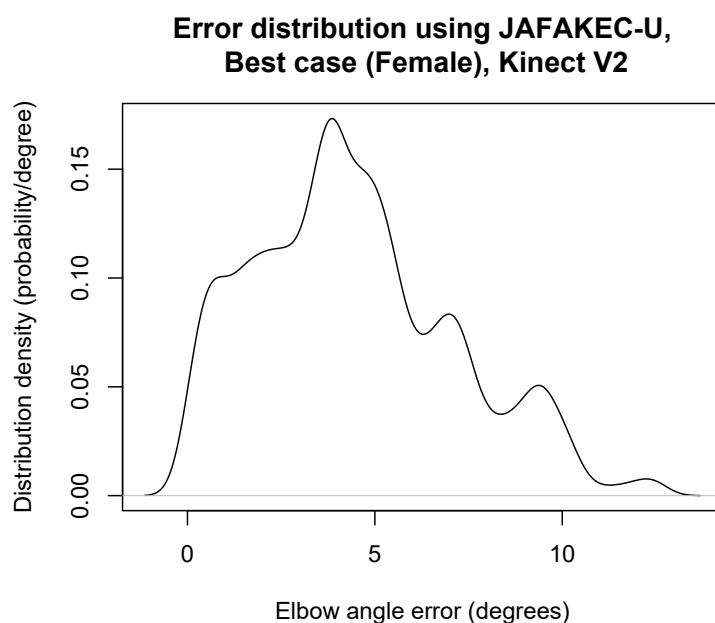


FIGURE 6.25: Error Distribution using Kinect V2, JAFKEC-U, best result-set

Conclusion:

In this case the average error (i.e. difference between VICON and JAFKEC-U) for the elbow angle is 4.43 degrees, which is well within the desirable margin. The corresponding time-domain plot also shows very good agreement between JAFKEC-U and VICON. This case is very promising and shows that JAFKEC-U is capable of meeting the accuracy requirement. It would still be advisable to further improve the labelling algorithms.

Experiment Category - Patient type motion

Sensor used: Kinect V2

Point cloud data of a mock patient was collected using Microsoft Kinect V2 sensor. The subject's upperlimb motion was similar to a stroke patient and the data were simultaneously collected using the VICON system. During the experiment, the subject lifted a cylindrical cup and carried it to his mouth, as if drinking some juice or sipping some tea. The right arm and hand were facing the Kinect, in slow speed.

Methods: Elbow joint angles were calculated using the JAFKEC for Upperlimb system, described in flowchart 6.3. The algorithms used were a combination of geometric graph creation, geodesic distance based labelling, shortest path algorithm, graph based K-means clustering, segmenting the point cloud, least-square line fitting and sectioning of point cloud 6.1. The upperlimb joint angle was compared with the VICON results. The reference convention is such that when the elbow is not flexed at all, the angle is zero or close to zero (as opposed to 180 degrees), and it increases as the elbow flexes.

Observation: Figure 6.26 show comparison of the right elbow angle as reported by Kinect against VICON results. The elbow angle is the flexion angle made between the upper arm and the forearm.

Experiment 6.9. Mock Patient drinking from a cup - trial 1

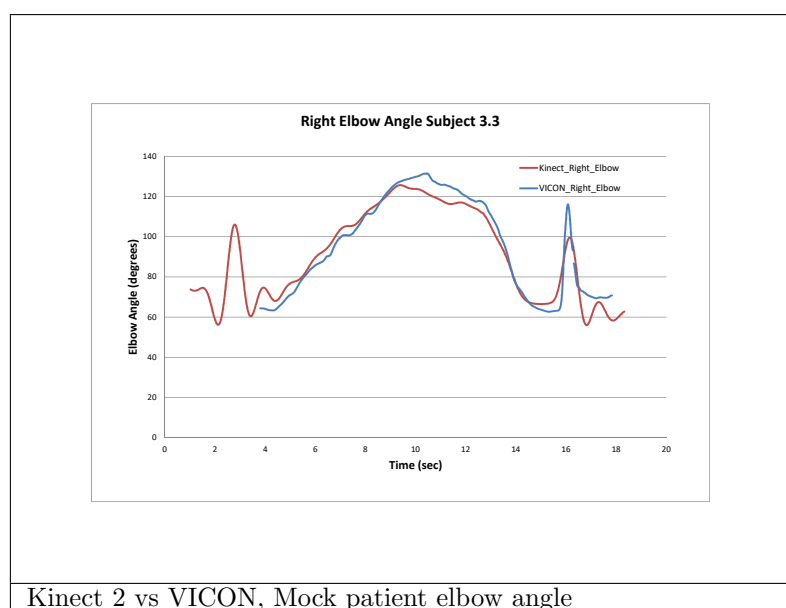


FIGURE 6.26: Elbow angle of mock patient

Statistical Results: Patient type motion

Average angle error using Kinect V2, JAFKEC-U and the point cloud data is **4.8 degrees**, using data from the experiment 6.9. The error distribution (as probability density function) is given in Figure 6.27.

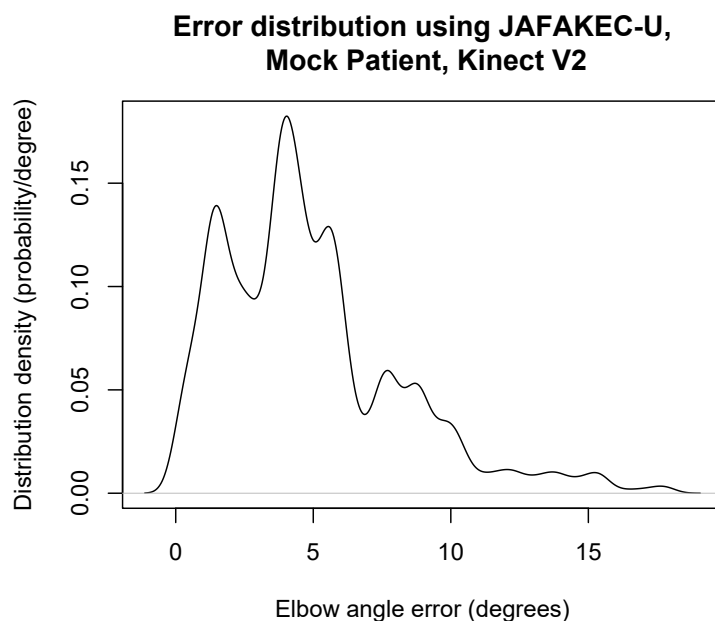


FIGURE 6.27: Error Distribution using Kinect V2, JAFKEC-U, Mock Patient

Conclusion:

This patient-type motion produced a good average error (4.8 degrees) which is within the required bounds. This and the other experiments show that JAFKEC-U is able to meet the average error criterion in a majority of the cases. However, there is a concern about the long tail in the error distribution which manifests as intermittent flickers of inaccurate angles. A closer look suggests that these are caused by partly incorrect labelling of points. Further research into the labelling algorithms could serve to tighten up the accuracy.

6.9 Conclusion and Discussion

The depth frame data from the Kinect sensor have been used to calculate the human arm angle which might be used to monitor human tremors for patients with neurological conditions like stroke, parkinson's disease, etc. An assortment of geometric algorithms have been used to improve the accuracy of arm angle capture. VICON, on the other hand, uses markers and captures the reflection from markers and works out the joint angles. The VICON raw data is quite noisy and has gaps. These are addressed by gap filling and a combination of various filters which are chosen using a graphic user interface for VICON, (6-th order Butterworth filter and Woltring with predefined parameters) to produce the smoothed data. The results presented in this chapter using the proposed JAFKEC-U system shows the comparison with the VICON data. JAFKEC-U, using a suite of customised mathematical algorithms, enables marker-less arm angle capture for healthy and post-stroke patients, inexpensive in home application, easy to use GUI and fast. Using the proposed algorithms, arm angles are getting computed relatively accurately.

Xu et al. [197] reports that both Kinect V1 and V2 sensors struggles to achieve accuracy in sitting postures as the algorithm was designed for subjects in standing positions. [55] found that the shoulder angle errors measured using Kinect ranges between 7.19 degrees to 13.19 degrees when compared to VICON. The elbow angle difference between JAFKEC-U (using Kinect V2) and VICON system is mostly less than 5 degrees but occasionally the difference spikes upto 7 degrees. The higher difference usually happens for maximally flexed elbow configuration. Shoulder angles using the JAFKEC-U ranges JAFKEC for upperlimb (arm and hand) application is fully automatic and works by the real-time tracking of the hand positions, the cup and the recognition of the gestures. It works well even in the presence of background clutter. The system works reasonably well with healthy males and females.

This chapter explored the algorithmic methods of utilising inexpensive Kinect sensor devices in the study of upper limb impairments. The advantage of using Kinect in such applications is that it can enable affordable continuous monitoring. It can complement, or perhaps eventually substitute expensive monitoring facilities. So the purported novelty of the JAFKEC tool for upper-limb monitoring is that it , carers or clinicians and can be deployed at every site where upper-limb monitoring can be useful. The marker-less motion assessment technologies discussed here provide inexpensive and faster ways of doing the same information access in real time.

Chapter 7

Conclusion

This work presents novel methods, techniques and algorithms for dealing with collection, recording, and processing of sensor data. Three different areas of work were explored and discussed in this thesis.

The first contribution presented in Chapter 2 is that of monitoring livestock using wearable sensor nodes. In this work, a novel discrete event simulator, WSNSIM, was presented in order to design protocol for performance modelling of wireless sensor networks. New protocols - viz. modified-LEACH and LEMSYP were proposed and evaluated using WSNSIM with good improvement over the state of the art with regard to farm requirements. The new protocol, modified-LEACH, was implemented in WSNSIM and evaluated with the goal of reducing unreachable nodes in real herd distributions, without compromising on power consumption. Another new protocol called LEMSYP with two variants were designed and evaluated with the goal of reducing power consumption. LEMSYP achieves low power reliable data gathering from dynamic nodes using a low duty cycle operation. Two variants of LEMSYP were modelled and their performance compared. The first variant is called C-LEMSYP and is based on using CSMA during the data transfer phase. The second variant is called T-LEMSYP and is based predominantly on TDMA. The comparative study revealed that both the variants had low power consumption, with the T-LEMSYP being significantly more efficient than C-LEMSYP.

The simulations were done in WSNSIM against herd scenarios derived from satellite image data, synthetic herd data and real time GPS data. A novel synthetic herd generation module based on statistical modelling from satellite images was developed and implemented. Automatic detection of cattle positions from satellite images was implemented in WSNSIM using image processing techniques. Also a probabilistic and statistical mobility model has been developed from GPS tracked herd data.

In addition to the above research on sensor networking protocols, a mass-market 3D depth sensor called Kinect was used in development of a novel human motion monitoring toolkit. A 3D motion visualization and analysis tool called GLSKEL was developed in Chapter 3 which was developed further into a marker-less approach to monitoring of human gait and human upper limb. A snapshot of the current state-of-art of depth sensor technology is presented to understand the new development opportunities and possibilities, as well as the constraints on the kind of systems that can be explored. Data capture accuracy has the central impact on how

the monitoring system responds, as it can serve as a direct measure of how far movements like gait or hand and arm movements can be recorded and analysed for clinical use. A kinematic model of human body joints was developed within GLSKEL in order to map joint co-ordinates captured from Kinect into a realistic 3D visualisation of the motion. The output of this model is a rendered 3D skeleton that can replicate the movements captured by Kinect. The same virtual skeleton can be interactively posed and manipulated using mouse based and graphical widgets like sliders. The key scientific contribution of GLSKEL is that it enables continuous motion capture (as opposed to measurement of aggregate characteristics) in an interactive 3D environment. GLSKEL's intuitive interface can help bridge the gap between a research prototype and a product that can be used by clinical practitioners.

The second contribution (presented in Chapter 5) and the third contribution (presented in Chapter 6) are monitoring human gait and upper limb using Microsoft Kinect sensor without using any wearable equipment or markers. A key message from these two bodies of work is that the development of these systems and applications should depend on multidisciplinary aspects and would consider design, technical, clinical and social concerns. Novel monitoring tools viz. JAFKEC-G, for human gait and JAFKEC-U for human upper limb monitoring have been developed using a wide range of computational algorithms. JAFKEC-G, using a suite of customised mathematical algorithms, enables marker-less gait capture for healthy and post-stroke patients which is inexpensive in home application scenarios. The joint angles were compared against corresponding angles measured using the VICON system. The accuracy of Kinect 1.8 results using the JAFKEC-G system is on an average within 5 degrees from the angle computed by VICON.

JAFKEC-U for upper-limb (arm and hand) enables elbow angle capture for healthy and post-stroke patients. This novel system is also marker-less, enables easy to use GUI and is amenable to inexpensive in-home application which might be used by patients, caregivers, or clinicians to facilitate continuous and remote monitoring. JAFKEC-U targets the Kinect V2 sensor to capture data for healthy people and for the post-stroke patients for monitoring. The accuracy of Kinect V2 results using this novel system was compared with VICON data. The geometric methods for limb axis determination is accurate enough but it seems that the limb labelling algorithm needs further research.

Besides the aforementioned algorithms for automated joint motion capture, GLSKEL, JAFKEC-G and JAFKEC-U systems allow for interactive measurement of motion and limb parameters. Recorded depth frames allow for interactive 3D measurements and play-back of the motions from various angles and view-points. It is hoped that these systems can be helpful to doctors, care-givers and mobility impaired patients in reducing the cost of motion monitoring.

Appendices

Appendix A

Cattle Motion Measurements

Spatio-temporal distributions of node-mobility and inter-node distance decides several key parameters with respect to short-range wireless transmissions. For multi-hop protocols, the spatial spread determines the shortest range of transmission that would still make it highly probable for the messages to be delivered to the base station. Range of transmission crucially determines battery life. So the objective of protocol design is to minimise transmission range while making it sufficient for message delivery. Thus in order to facilitate efficient protocol design, the simulation tool must model spatial spread and motion of nodes. And in order to model the mobility and spatial spread behaviour, it was important to collect statistical information from multiple sources. Satellite images of ranches and farms provided static glimpses of herds, and dynamic behaviour was based on recording cattle movements continuously over an extended period (7 days) for cows in a Scottish farm. In this exercise, 15 cows were given GPS receiver collar-bands but 2 of them malfunctioned and did not produce datasets. The other 13 produced dynamic datasets that enabled determination of distributions of cattle mobility in terms of speed and change in directions. The following figures present plots of this dynamic dataset that WSNSIM's mobility model is based on. Figures A.2, A.4, A.6, A.8, A.10, A.12, A.14, A.16, A.18, A.20, A.22, A.24, A.26 show plots of positions of cows with speeds displayed as a heat-map. It can be seen from the heat-map that the cattle spend most of the time stagnating and grazing at very low speeds, and occasionally shifting from one place to another at relatively high pace. This trend is explicitly seen in distribution plots shown in figures A.1, A.3, A.5, A.7, A.9, A.11, A.13, A.15, A.17, A.19, A.21, A.23, A.25. Descriptive statistics (mean, median, range and quartiles) on speed are presented in the tables A.1, A.2, A.3, A.4, A.5, A.6, A.7, A.8, A.9, A.10, A.11, A.12, A.13.

TABLE A.1: Cow 11 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0002659	0.0223900	0.0583300	0.1206000	0.1202000	0.9999000

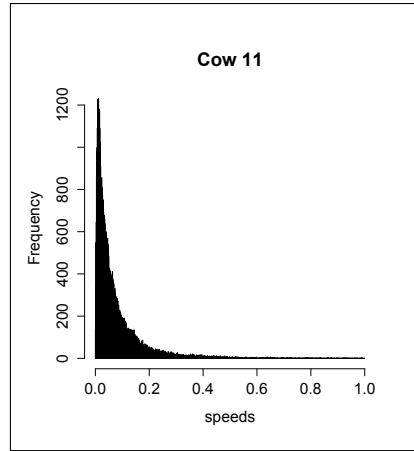


FIGURE A.1: Speed Distribution of Cow 11 over 7 days

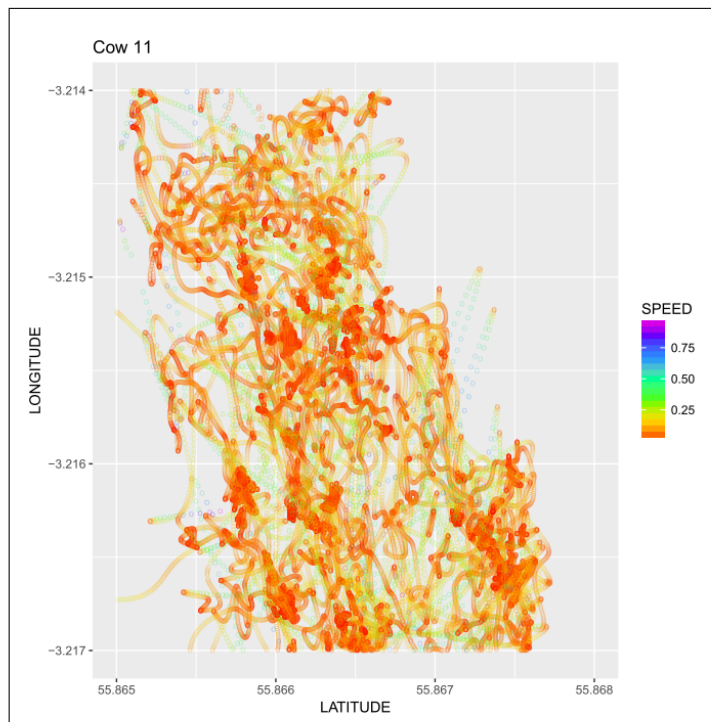


FIGURE A.2: Movement Pattern for Cow 11 over 7 days

TABLE A.2: Cow 13 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000045	0.014630	0.035750	0.065040	0.076890	0.998000

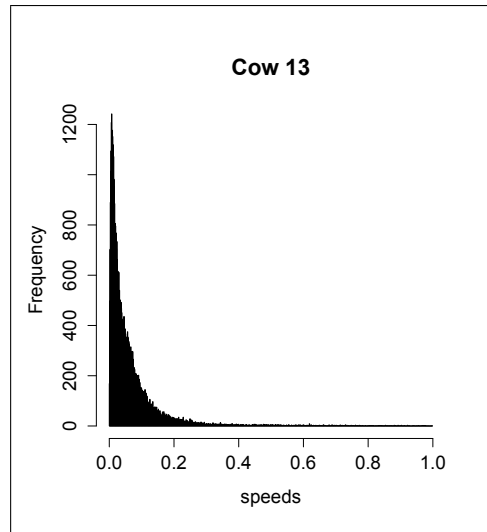


FIGURE A.3: Speed Distribution of Cow 13 over 7 days

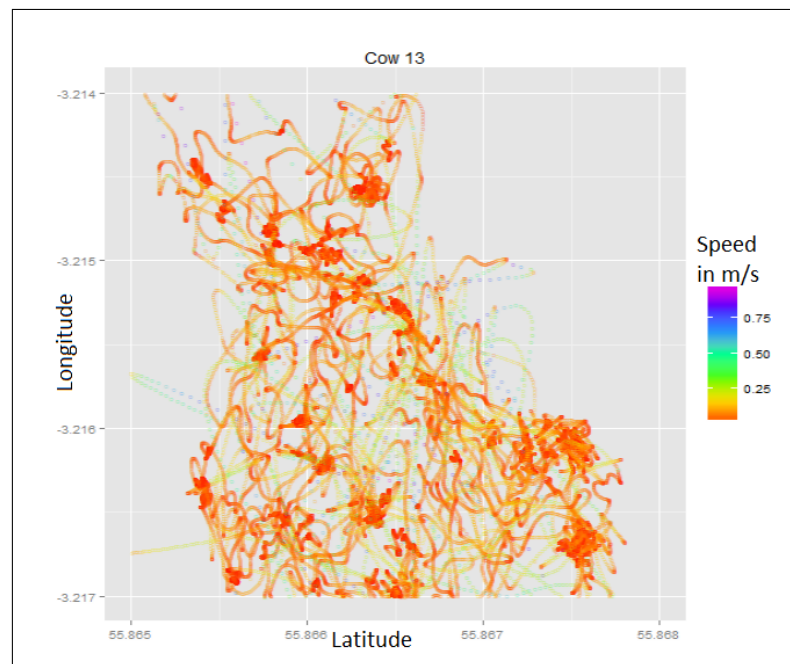


FIGURE A.4: Movement Pattern for Cow 13 over 7 days

TABLE A.3: Cow 14 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000869	0.0165100	0.0363100	0.0700700	0.0776200	0.9995000

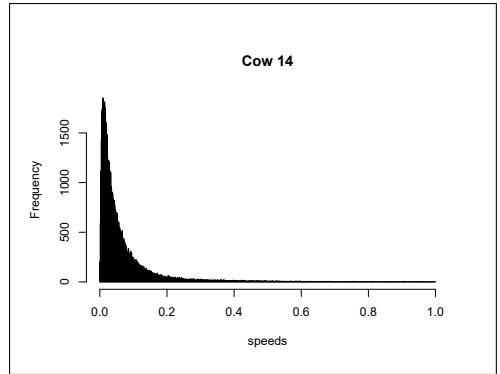


FIGURE A.5: Speed Distribution of Cow 14 over 7 days

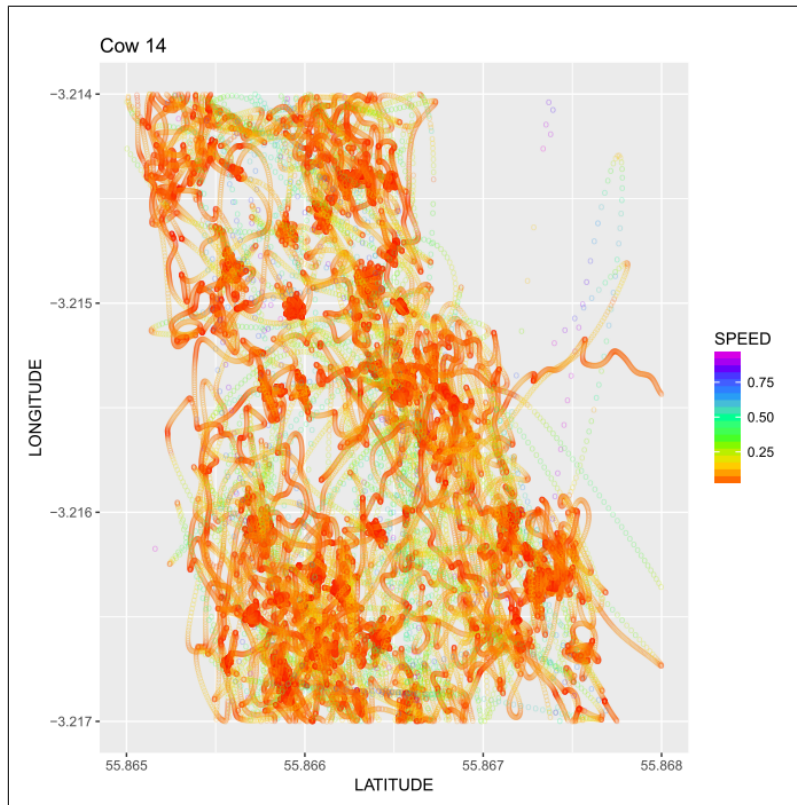


FIGURE A.6: Movement Pattern for Cow 14 over 7 days

TABLE A.4: Cow 15 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000511	0.0188900	0.0417400	0.0781400	0.0934800	0.9986000

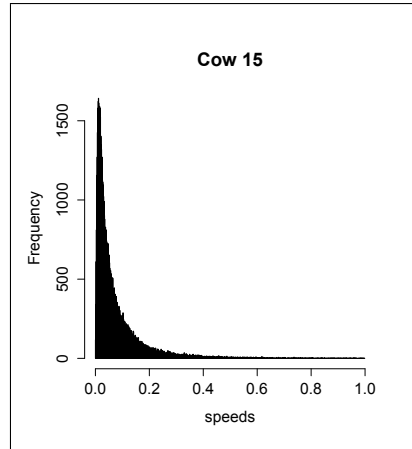


FIGURE A.7: Speed Distribution of Cow 15 over 7 days

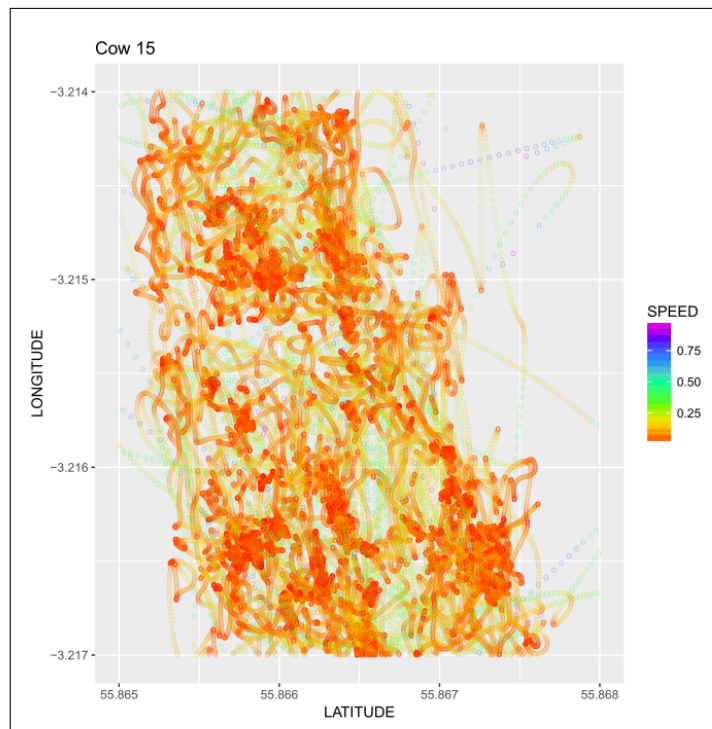


FIGURE A.8: Movement Pattern for Cow 15 over 7 days

TABLE A.5: Cow 16 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000259	0.0182300	0.0427200	0.0766300	0.0926800	0.9998000

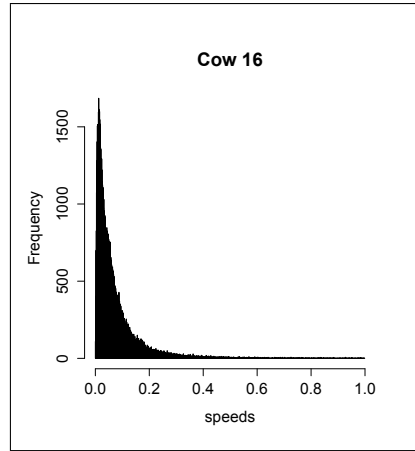


FIGURE A.9: Speed Distribution of Cow 16 over 7 days

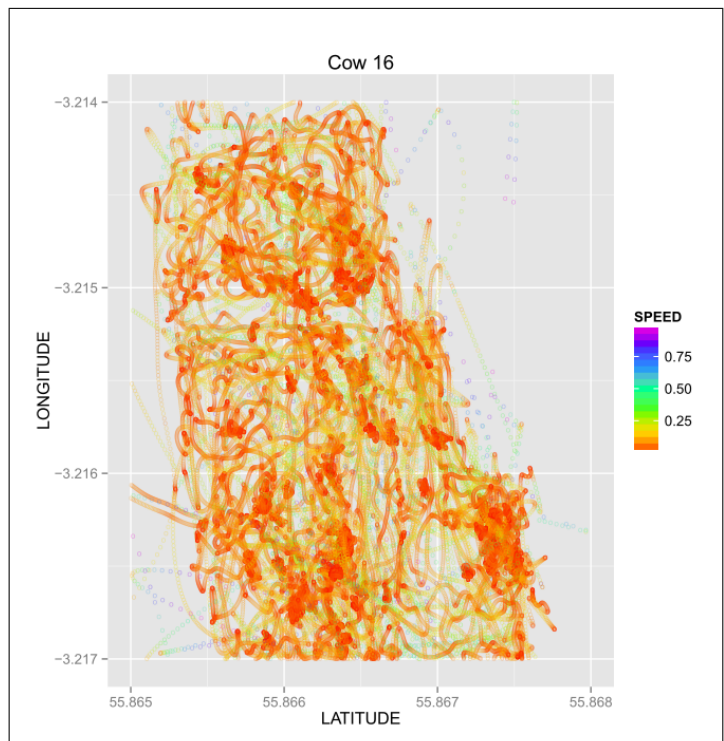


FIGURE A.10: Movement Pattern for Cow 16 over 7 days

TABLE A.6: Cow 17 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000391	0.0198700	0.0506500	0.0901200	0.1082000	0.9980000

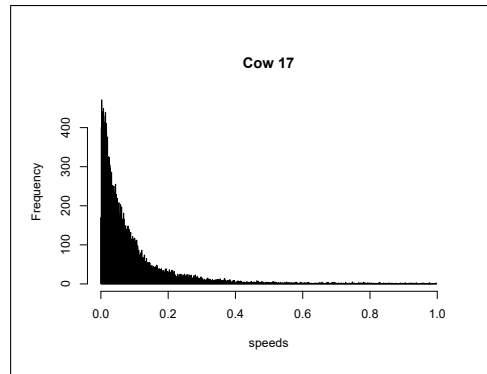


FIGURE A.11: Speed Distribution of Cow 17 over 7 days

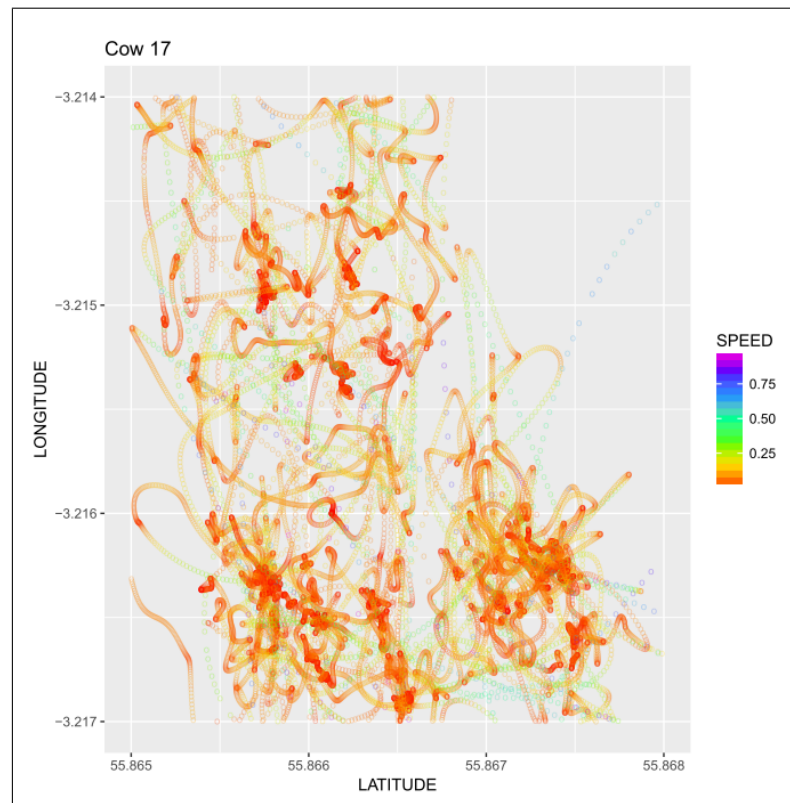


FIGURE A.12: Movement Pattern for Cow 17 over 7 days

TABLE A.7: Cow 18 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000563	0.0215000	0.0473700	0.0881400	0.1043000	1.0000000

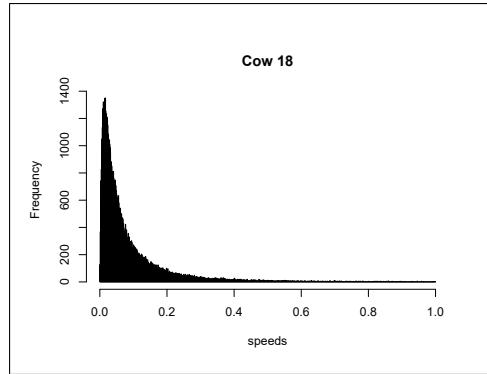


FIGURE A.13: Speed Distribution of Cow 18 over 7 days

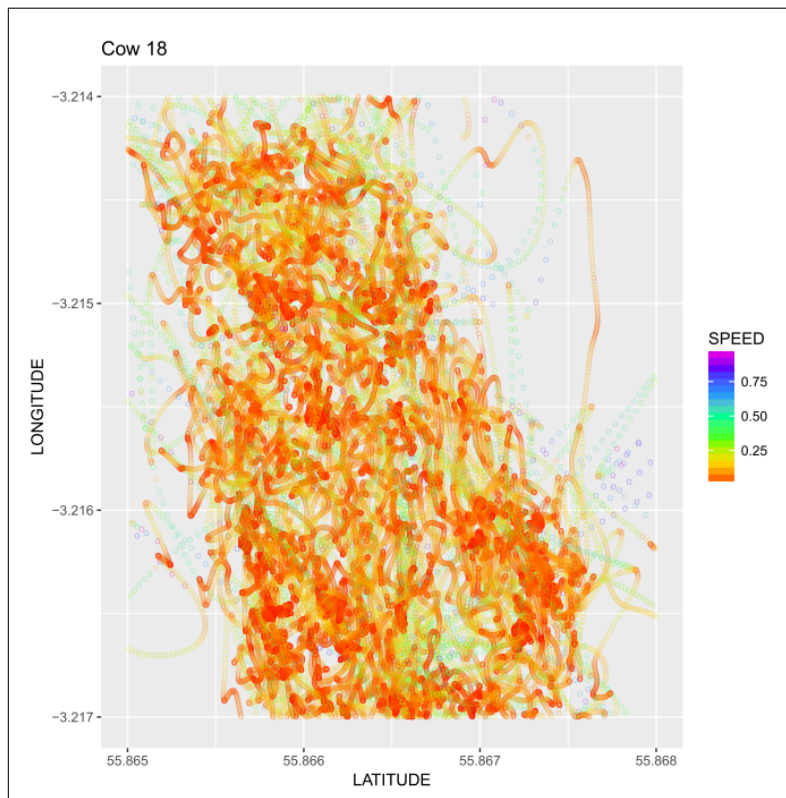


FIGURE A.14: Movement Pattern for Cow 18 over 7 days

TABLE A.8: Cow 19 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0001986	0.0232300	0.0537100	0.0907200	0.1084000	1.0000000

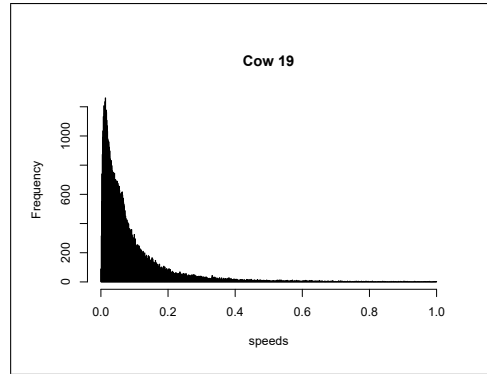


FIGURE A.15: Speed Distribution of Cow 19 over 7 days

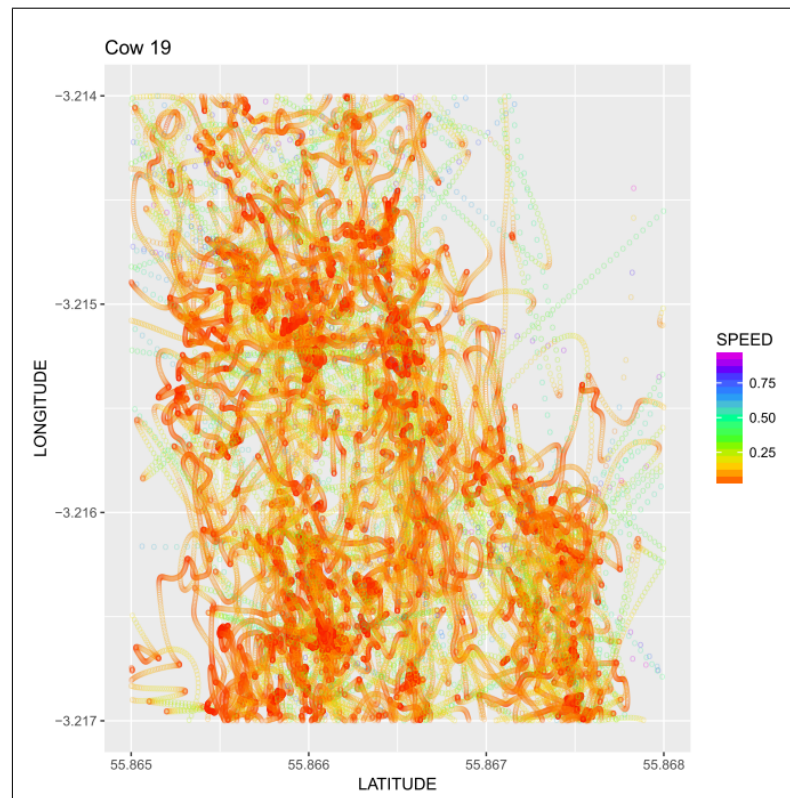


FIGURE A.16: Movement Pattern for Cow 19 over 7 days

TABLE A.9: Cow 20 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000684	0.0173900	0.0417300	0.0874400	0.0952200	0.9995000

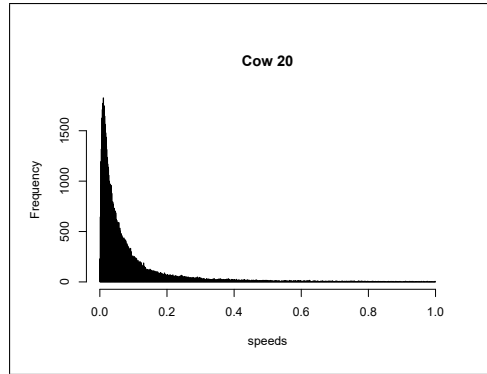


FIGURE A.17: Speed Distribution of Cow 20 over 7 days

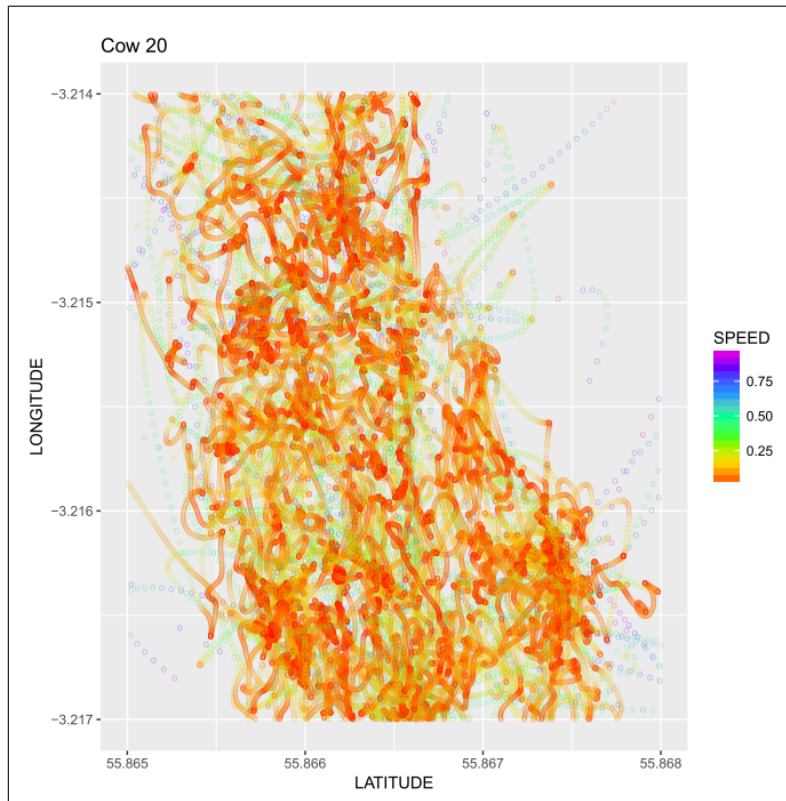


FIGURE A.18: Movement Pattern for Cow 20 over 7 days

TABLE A.10: Cow 21 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000942	0.0198000	0.0408900	0.0766700	0.0869100	0.9997000

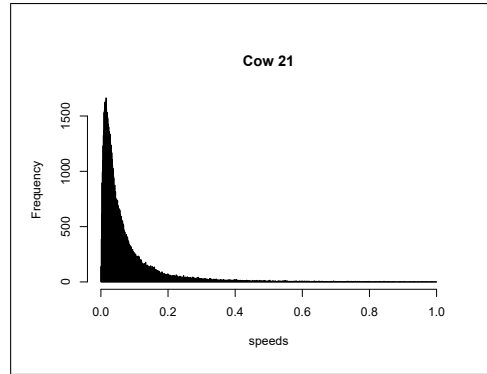


FIGURE A.19: Speed Distribution of Cow 21 over 7 days

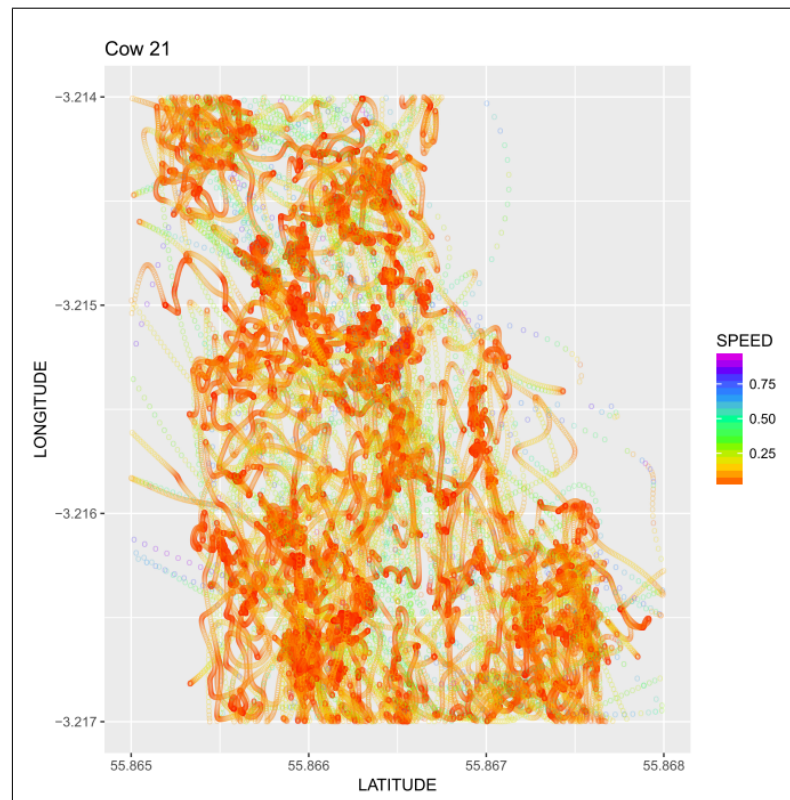


FIGURE A.20: Movement Pattern for Cow 21 over 7 days

TABLE A.11: Cow 22 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000334	0.0197700	0.0472500	0.0888700	0.1063000	0.9992000

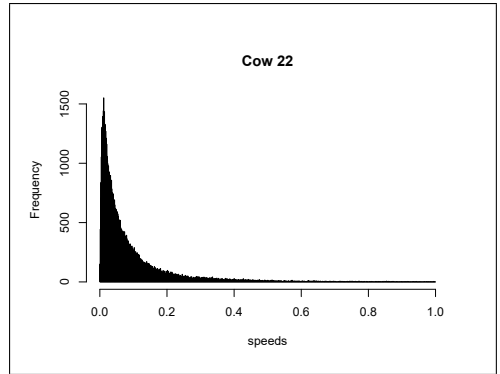


FIGURE A.21: Speed Distribution of Cow 22 over 7 days

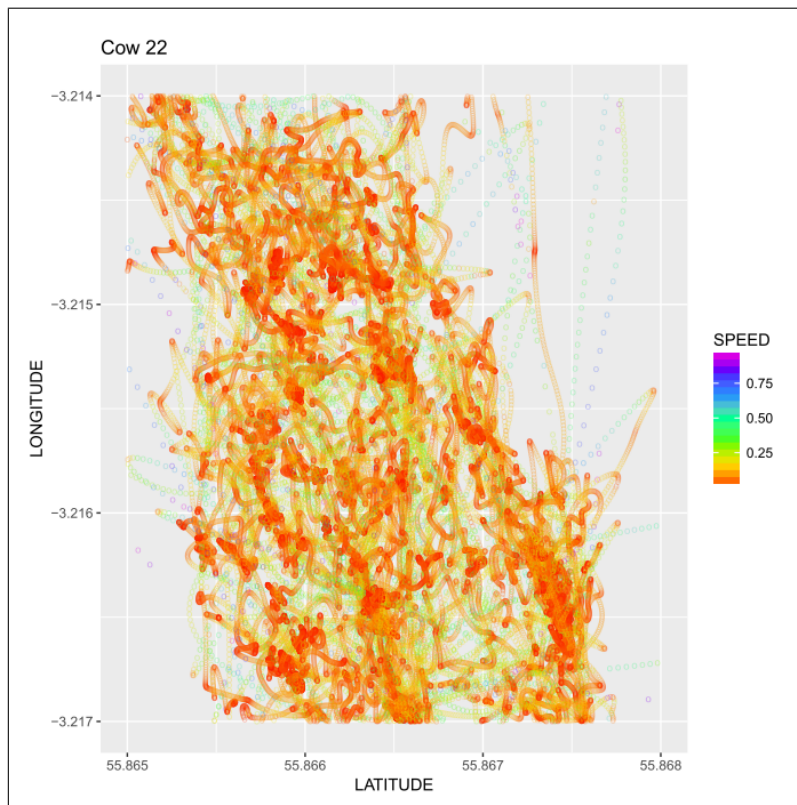


FIGURE A.22: Movement Pattern for Cow 22 over 7 days

TABLE A.12: Cow 23 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000454	0.0162600	0.0368800	0.0739200	0.0827900	0.9999000

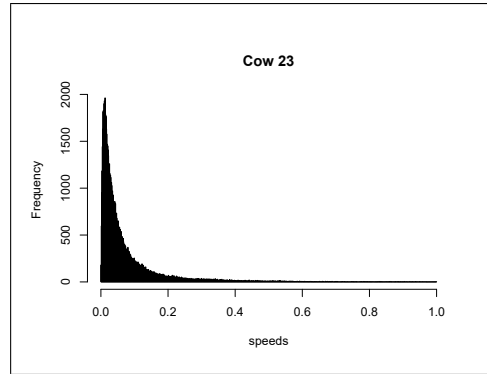


FIGURE A.23: Speed Distribution of Cow 23 over 7 days

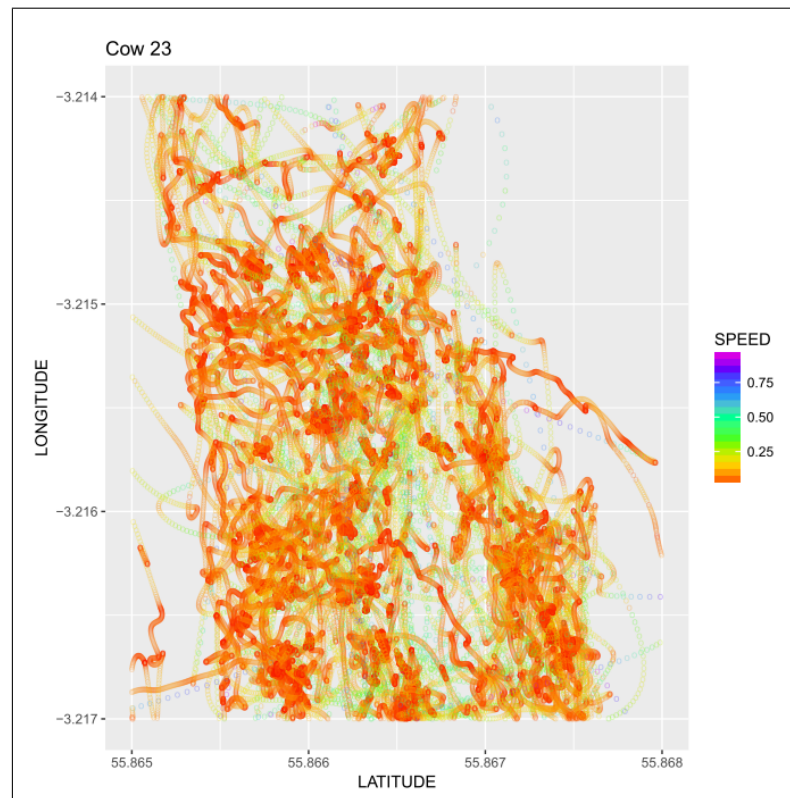


FIGURE A.24: Movement Pattern for Cow 23 over 7 days

TABLE A.13: Cow 24 Speed Data

Cattle Movement Speed in m/s					
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0000708	0.0215200	0.0482400	0.0900400	0.1069000	0.9977000

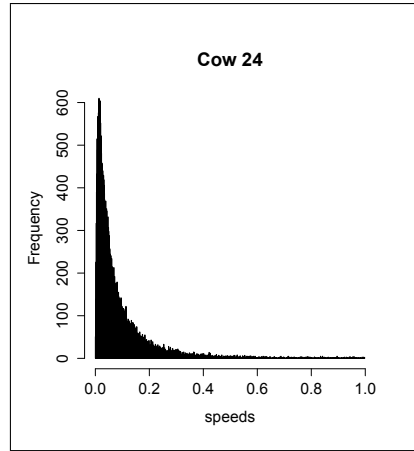


FIGURE A.25: Speed Distribution of Cow 24 over 7 days

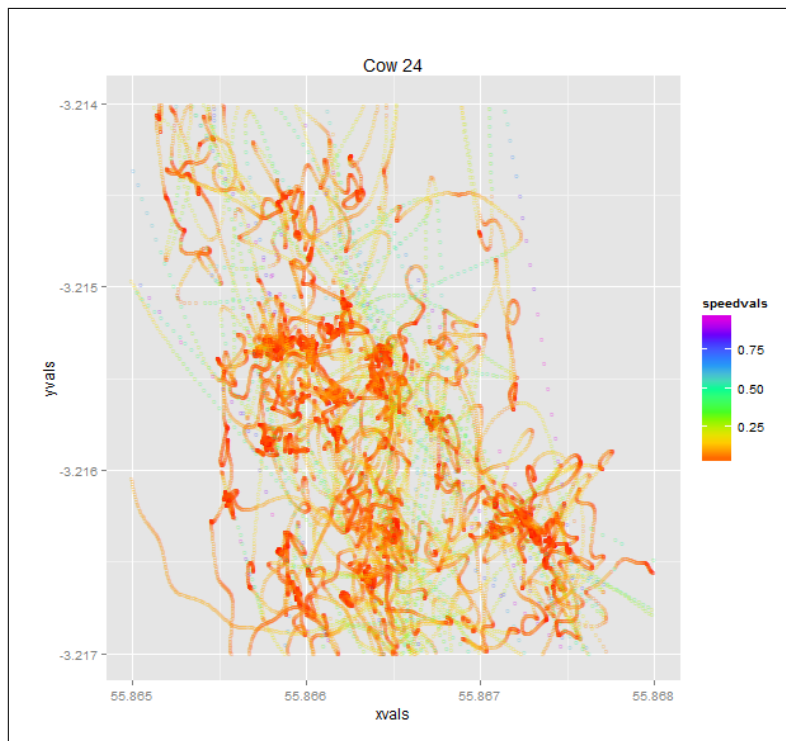


FIGURE A.26: Movement Pattern for Cow 24 over 7 days

A subset of this mobility dataset was selected for modelling as a Markov Random Field (MRF) in [170]. The same subset of data was used in this work for fitting decoupled univariate distributions for speed and heading. Subsequently the full dataset was used in validating the model. This modelling approach is described in the following section.

References

- [1] How the kinect works. <http://www.depthbiomechanics.co.uk/?p=100>. Accessed: 2018-03-10.
- [2] Vicon nexus reference guide. <https://docs.vicon.com/download/attachments/42696722/Vicon%20Nexus%20Reference%20Guide.pdf>. Accessed: 2018-03-09.
- [3] Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.
- [4] Giannis Altanis, Michalis Boloudakis, Symeon Retalis, and Nikos Nikou. Children with motor impairments play a kinect learning game: first findings from a pilot case in an authentic classroom environment. *J Interact Design Architect*, 19:91–104, 2013.
- [5] Nese Alyuz, Eda Okur, Ece Oktay, Utku Genc, Sinem Aslan, Sinem Emine Mete, David Stanhill, Bert Arnrich, and Asli Arslan Esme. Towards an emotional engagement model: Can affective states of a learner be automatically detected in a 1: 1 learning scenario. In *Proceedings of the 6th Workshop on Personalization Approaches in Learning Environments (PALE 2016)*. 24th conference on User Modeling, Adaptation, and Personalization (UMAP 2016), CEUR workshop proceedings, this volume, 2016.
- [6] Geoff Appelboom, Annie H Yang, Brandon R Christophe, Eliza M Bruce, Justine Slomian, Olivier Bruyère, Samuel S Bruce, Brad E Zacharia, Jean-Yves Reginster, and E Sander Connolly. The promise of wearable activity sensors to define patient recovery. *Journal of Clinical Neuroscience*, 21(7):1089–1093, 2014.
- [7] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.
- [8] J Gary Augustson and Jack Minker. An analysis of some graph theoretical cluster techniques. *Journal of the ACM (JACM)*, 17(4):571–588, 1970.
- [9] Cyrus S Bamji, Patrick O’Connor, Tamer Elkhatib, Swati Mehta, Barry Thompson, Lawrence A Prather, Dane Snow, Onur Can Akkaya, Andy Daniel, Andrew D Payne, et al. A 0.13 μm cmos system-on-chip for a 512×424 time-of-flight image sensor with multi-frequency photo-demodulation up to 130 mhz and 2 gs/s adc. *IEEE Journal of Solid-State Circuits*, 50(1):303–319, 2015.

- [10] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, December 1996.
- [11] Hamid Bateni. Changes in balance in older adults based on use of physical therapy vs the wii fit gaming system: a preliminary study. *Physiotherapy*, 98(3):211–216, 2012.
- [12] Jon Louis Bentley and Michael Ian Shamos. Divide-and-conquer in multidimensional space. In *Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 220–230. ACM, 1976.
- [13] Sara Bilal, Rini Akmeliawati, Amir A. Shafie, and Momoh Jimoh E. Salami. Hidden markov model for human to computer interaction: a study on human hand gesture recognition. *Artificial Intelligence Review*, 40(4):495–516, 2013.
- [14] Michael J Black and Allan D Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *European conference on computer vision*, pages 909–924. Springer, 1998.
- [15] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013.
- [16] Bruno Bonnechere, Bart Jansen, P Salvia, H Bouzahouene, L Omelina, Fedor Moiseev, Victor Sholukha, Jan Cornelis, Marcel Rooze, and S Van Sint Jan. Validity and reliability of the kinect within functional assessment activities: comparison with standard stereophotogrammetry. *Gait & posture*, 39(1):593–598, 2014.
- [17] Judit Bort-Roig, Nicholas D Gilson, Anna Puig-Ribera, Ruth S Contreras, and Stewart G Trost. Measuring and influencing physical activity with smartphone technology: a systematic review. *Sports Medicine*, 44(5):671–686, 2014.
- [18] Athanassios Boulis et al. Castalia: A simulator for wireless sensor networks and body area networks. *National ICT Australia Ltd, Australia*, 2009.
- [19] Alan C Bovik. *Handbook of image and video processing*. Academic Press, 2010.
- [20] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface, 1998.
- [21] A. Brandsttdt, V. Le, and J. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999.
- [22] Timo Breuer, Christoph Bodensteiner, and Michael Arens. Low-cost commodity depth sensor comparison and accuracy analysis. In *SPIE Security+ Defence*, pages 92500G–92500G. International Society for Optics and Photonics, 2014.
- [23] Armin Bruderlin and Thomas W Calvert. Goal-directed, dynamic animation of human walking. In *ACM SIGGRAPH Computer Graphics*, volume 23, pages 233–242. ACM, 1989.
- [24] Yao-Jen Chang, Shu-Fang Chen, and Jun-Da Huang. A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in developmental disabilities*, 32(6):2566–2570, 2011.

- [25] Yao-Jen Chang, Wen-Ying Han, and Yu-Chi Tsai. A kinect-based upper limb rehabilitation system to assist people with cerebral palsy. *Research in Developmental Disabilities*, 34(11):3654 – 3659, 2013.
- [26] S.W. Cheng, T.K. Dey, and J. Shewchuk. *Delaunay Mesh Generation*. Chapman & Hall/CRC Computer and Information Science Series. Taylor & Francis, 2012.
- [27] Enea Cippitelli, Samuele Gasparrini, Susanna Spinsante, and Ennio Gambi. Kinect as a tool for gait analysis: validation of a real-time joint extraction algorithm working in side view. *Sensors*, 15(1):1417–1434, 2015.
- [28] Ross A Clark, Adam L Bryant, Yonghao Pua, Paul McCrory, Kim Bennell, and Michael Hunt. Validity and reliability of the nintendo wii balance board for assessment of standing balance. *Gait & posture*, 31(3):307–310, 2010.
- [29] Ross A Clark, Yong-Hao Pua, Karine Fortin, Callan Ritchie, Kate E Webster, Linda Denehy, and Adam L Bryant. Validity of the microsoft kinect for assessment of postural control. *Gait & posture*, 36(3):372–377, 2012.
- [30] Ross A. Clark, Yong-Hao Pua, Karine Fortin, Callan Ritchie, Kate E. Webster, Linda Denehy, and Adam L. Bryant. Validity of the microsoft kinect for assessment of postural control. *Gait & Posture*, 36(3):372 – 377, 2012.
- [31] Ross A Clark, Stephanie Vernon, Benjamin F Mentiplay, Kimberly J Miller, Jennifer L McGinley, Yong Hao Pua, Kade Paterson, and Kelly J Bower. Instrumenting gait assessment using the kinect in people living with stroke: reliability and association with balance tests. *Journal of neuroengineering and rehabilitation*, 12(1):15, 2015.
- [32] Daniel Cohen-Or, Chen Greif, Tao Ju, Niloy J Mitra, Ariel Shamir, Olga Sorkine-Hornung, and Hao Richard Zhang. *A sampler of useful computational tools for applied geometry, computer graphics, and image processing*. AK Peters/CRC Press, 2015.
- [33] G. K. Cole, B. M. Nigg, J. L. Ronsky, and M. R. Yeadon. Application of the joint coordinate system to three-dimensional joint attitude and movement representation: A standardization proposal. 115(4):344–349.
- [34] T.H. Cormen. *Introduction to Algorithms*. Computer science. MIT Press, 2009.
- [35] David Cunado, Mark S Nixon, and John N Carter. Automatic extraction and description of human gait models for recognition purposes. *Computer Vision and Image Understanding*, 90(1):1–41, 2003.
- [36] Jean-François Daneault, Benoit Carignan, Carl Éric Codère, Abbas F Sadikot, and Christian Duval. Using a smart phone as a standalone platform for detection and monitoring of pathological tremors. *Frontiers in human neuroscience*, 6:357, 2013.
- [37] James Davis and Mubarak Shah. Recognizing hand gestures. In *European Conference on Computer Vision*, pages 331–340. Springer, 1994.
- [38] Kevin Deluzio. Waveform data analysis techniques. <http://my.me.queensu.ca/People/Deluzio/DataAnalysis.html>. Accessed: 2017-02-12.

- [39] George Demiris, Marilyn J Rantz, Myra A Aud, Karen D Marek, Harry W Tyrer, Marjorie Skubic, and Ali A Hussam. Older adults' attitudes towards and perceptions of smart hometechnologies: a pilot study. *Medical informatics and the Internet in medicine*, 29(2):87–94, 2004.
- [40] Konstantinos G Derpanis. A review of vision-based hand gestures. *Unpublished*. Feb, 2004.
- [41] Luc Devroye. The expected size of some graphs in computational geometry. *Computers & mathematics with applications*, 15(1):53–64, 1988.
- [42] T. K. Dey and J. Pach. Extremal problems for geometric hypergraphs. *Discrete & Computational Geometry*, 19(4):473–484, 1998.
- [43] T.K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2011.
- [44] Cao Di, Tsung Wu, Hock Guan Goh, Bruce Stephen, Kaehsiang Kwong, Craig Michie, and Ivan Andonovic. Exploitation of wireless telemetry for livestock condition monitoring. In *Canadian Society for Bioengineering, CIGR, Quebec City, Canada*, June 2010.
- [45] Eric Dishman. Inventing wellness systems for aging in place. *Computer*, 37(5):34–41, 2004.
- [46] L Doherty, BA Warneke, BE Boser, and KSJ Pister. Energy and performance considerations for smart dust. *International Journal of Parallel and Distributed Systems and Networks*, 4(3):121–133, 2001.
- [47] Elham Dolatabadi, Babak Taati, Gemma S Parra-Dominguez, and Alex Mihailidis. A markerless motion tracking approach to understand changes in gait and balance: A case study. In *Proceedings of the Rehabilitation Engineering and Assistive Technology Society of North America Annual Conference*, pages 20–24, 2013.
- [48] Fabio Dominio, Mauro Donadeo, and Pietro Zanuttigh. Combining multiple depth-based descriptors for hand gesture recognition. *Pattern Recognition Letters*, 50:101 – 111, 2014. Depth Image Analysis.
- [49] Tilak Dutta. Evaluation of the kinect sensor for 3-d kinematic measurement in the workplace. *Applied Ergonomics*, 43(4):645 – 649, 2012.
- [50] Andreas Ejupi, Matthew Brodie, Yves J Gschwind, Stephen R Lord, Wolfgang L Zagler, and Kim Delbaere. Kinect-based five-times-sit-to-stand test for clinical and in-home assessment of fall risk in older people. *Gerontology*, 62(1):118–124, 2015.
- [51] Mahmoud El-Gohary, Sean Pearson, James McNames, Martina Mancini, Fay Horak, Sabato Mellone, and Lorenzo Chiari. Continuous monitoring of turning in patients with movement disability. *Sensors*, 14(1):356–369, 2013.
- [52] David Eppstein. Dynamic euclidean minimum spanning trees and extrema of binary functions. *Discrete & Computational Geometry*, 13(1):111–122, 1995.

- [53] EUROSTAT. Population structure and ageing. http://ec.europa.eu/eurostat/statistics-explained/index.php/Population_structure_and_ageing. Accessed: 2017-02-23.
- [54] Stefan Felsner. *Geometric graphs and arrangements: some chapters from combinatorial geometry*. Springer Science & Business Media, 2012.
- [55] A. Fern'andez-Baena, A. Susin, and X. Lligadas. Biomechanical validation of upper-body and lower-body joint movements of kinect motion capture data for rehabilitation treatments. In *Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on*, pages 656–661, Sept 2012.
- [56] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
- [57] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.
- [58] Moshe Gabel, Ran Gilad-Bachrach, Erin Renshaw, and Assaf Schuster. Full body gait analysis with kinect. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1964–1967. IEEE, 2012.
- [59] K. Ruben Gabriel and Robert R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969.
- [60] Javier Galbally and Riccardo Satta. Three-dimensional and two-and-a-half-dimensional face recognition spoofing using three-dimensional printed models. *IET Biometrics*, 5(2):83–91, 2016.
- [61] Luigi Gallo, Alessio Pierluigi Placitelli, and Mario Ciampi. Controller-free exploration of medical image data: Experiencing the kinect. In *Computer-based medical systems (CBMS), 2011 24th international symposium on*, pages 1–6. IEEE, 2011.
- [62] Daphne J Geerse, Bert H Coolen, and Melvyn Roerdink. Kinematic validation of a multi-kinect v2 instrumented 10-meter walkway for quantitative gait assessments. *PLoS one*, 10(10):e0139913, 2015.
- [63] Christopher G Goetz, Barbara C Tilley, Stephanie R Shaftman, Glenn T Stebbins, Stanley Fahn, Pablo Martinez-Martin, Werner Poewe, Cristina Sampaio, Matthew B Stern, Richard Dodel, et al. Movement disorder society-sponsored revision of the unified parkinson's disease rating scale (mds-updrs): Scale presentation and clinimetric testing results. *Movement disorders*, 23(15):2129–2170, 2008.
- [64] Michela Goffredo, Imed Bouchrika, John N Carter, and Mark S Nixon. Self-calibrating view-invariant gait biometrics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(4):997–1008, 2010.
- [65] Markus Gross and Hanspeter Pfister. *Point-based graphics*. Morgan Kaufmann, 2011.

- [66] Ying Guo, Geoff Poulton, Peter Corke, GJ Bishop-Hurley, Tim Wark, and David L Swain. Using accelerometer, high sample rate gps and magnetometer data to develop a cattle movement and behaviour model. *Ecological Modelling*, 220(17):2068–2075, 2009.
- [67] Zicheng Guo and Richard W Hall. Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, 32(3):359–373, 1989.
- [68] DM Halliday, BA Conway, SF Farmer, U Shahani, AJC Russell, and JR Rosenberg. Coherence between low-frequency activation of the motor cortex and tremor in patients with essential tremor. *The Lancet*, 355(9210):1149–1153, 2000.
- [69] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics*, 43(5):1318–1334, 2013.
- [70] Charles D. Hansen and Chris R. Johnson, editors. *Visualization Handbook*. Butterworth-Heinemann, Burlington, 2005.
- [71] Peter Hebden and Adrian R Pearce. Distributed asynchronous clustering for self-organisation of wireless sensor networks. In *Intelligent Sensing and Information Processing, 2006. ICISIP 2006. Fourth International Conference on*, pages 37–42. IEEE, 2006.
- [72] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10 pp. vol.2–, Jan 2000.
- [73] Thomas R Henderson, Mathieu Lacage, George F Riley, C Dowell, and JB Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 2008.
- [74] Daniel Herrera, Juho Kannala, and Janne Heikkilä. Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):2058–2064, 2012.
- [75] Susan Hert and Stefan Schirra. 3D convex hulls. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.11.1 edition, 2018.
- [76] Jeremy C Hobart, Stefan J Cano, John P Zajicek, and Alan J Thompson. Rating scales as outcome measures for clinical trials in neurology: problems, solutions, and recommendations. *The Lancet Neurology*, 6(12):1094–1105, 2007.
- [77] Diana HODGINS. The importance of measuring human gait. *Medical Device Technology*, 19(5), 2008.
- [78] Martin Hofmann, Sebastian Bachmann, and Gerhard Rigoll. 2.5 d gait biometrics using the depth gradient histogram energy image. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, pages 399–403. IEEE, 2012.
- [79] Hossein Mousavi Hondori, Maryam Khademi, and Cristina V Lopes. Monitoring intake gestures using sensor fusion (microsoft kinect and inertial sensors) for smart home tele-rehab setting. In *2012 1st Annual IEEE Healthcare Innovation Conference*, 2012.

- [80] Fay B Horak and Martina Mancini. Objective biomarkers of balance and gait for parkinson's disease using body-worn sensors. *Movement Disorders*, 28(11):1544–1551, 2013.
- [81] Nikolaos Kyriazis Iason Oikonomidis and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*, pages 101.1–101.11. BMVA Press, 2011. <http://dx.doi.org/10.5244/C.25.101>.
- [82] Verne Thompson Inman, Henry James Ralston, and Frank Todd. *Human walking*. Williams & Wilkins, 1981.
- [83] Michael Isard and Andrew Blake. Condensation—conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.
- [84] Michael Isard and Andrew Blake. Condensationconditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.
- [85] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.
- [86] Jerzy W. Jaromczyk and Mirosaw Kowaluk. Constructing the relative neighborhood graph in 3-dimensional euclidean space. *Discrete Applied Mathematics*, 31(2):181 – 191, 1991.
- [87] Hairong Jiang, Juan P Wachs, and Bradley S Duerstock. Facilitated gesture recognition based interfaces for people with upper extremity physical impairments. In *Iberoamerican Congress on Pattern Recognition*, pages 228–235. Springer, 2012.
- [88] Ragnar Jónsson, Mogens Blanke, Niels Kjølstad Poulsen, Fabio Caponetti, and Søren Højsgaard. Oestrus detection in dairy cows from activity and lying data using on-line individual models. *Computers and Electronics in Agriculture*, 76(1):6–15, 2011.
- [89] Ragnar Ingi Jónsson. Modelling cow behaviour using stochastic automata. Technical report, Ruhr-Universität Bochum, 2010.
- [90] MP Kadaba, HK Ramakrishnan, ME Wootten, J Gainey, G Gorton, and GVB Cochran. Repeatability of kinematic, kinetic, and electromyographic data in normal adult gait. *Journal of Orthopaedic Research*, 7(6):849–860, 1989.
- [91] Mrn P Kadaba, HK Ramakrishnan, and ME Wootten. Measurement of lower extremity kinematics during level walking. *Journal of orthopaedic research*, 8(3):383–392, 1990.
- [92] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.
- [93] Hiroki Kayama, Kazuya Okamoto, Shu Nishiguchi, Minoru Yamada, Tomohiro Kuroda, and Tomoki Aoyama. Effect of a kinect-based exercise game on improving executive cognitive performance in community-dwelling elderly: case control study. *Journal of medical Internet research*, 16(2):e61, 2014.
- [94] Kouros Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

- [95] T Kilgus, R Bux, AM Franz, W Johnen, E Heim, M Fangerau, M Müller, K Yen, and L Maier-Hein. Structure sensor for mobile markerless augmented reality. In *SPIE Medical Imaging*, pages 97861L–97861L. International Society for Optics and Photonics, 2016.
- [96] Naofumi Kitsunezaki, Eijiro Adachi, Takashi Masuda, and Jun-ichi Mizusawa. Kinect applications for the physical rehabilitation. In *Medical Measurements and Applications Proceedings (MeMeA), 2013 IEEE International Symposium on*, pages 294–299. IEEE, 2013.
- [97] Andreas Köpke, Michael Swigulski, Karl Wessel, Daniel Willkomm, PT Haneveld, Tom EV Parker, Otto W Visser, Hermann S Lichte, and Stefan Valentin. Simulating wireless and mobile networks in omnet++ the mixim vision. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 71. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
- [98] John F Kurtzke. Rating neurologic impairment in multiple sclerosis an expanded disability status scale (edss). *Neurology*, 33(11):1444–1444, 1983.
- [99] Sungjun Kwon, Hyunseok Kim, and Kwang Suk Park. Validation of heart rate extraction using video imaging on a built-in camera system of a smartphone. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 2174–2177. IEEE, 2012.
- [100] Kae Hsiang Kwong, Tsung Wu, Hock Guan Goh, Konstantinos Sasloglou, Bruce Stephen, Ian Glover, Chong Shen, Wencai Du, Craig Michie, and Ivan Andonovic. Practical considerations for wireless sensor networks in cattle monitoring applications. *Computers and Electronics in Agriculture*, 81:33–44, 2012.
- [101] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A scalable tree-based approach for joint object and pose recognition. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI’11*, pages 1474–1480. AAAI Press, 2011.
- [102] Belinda Lange, Chien-Yen Chang, Evan Suma, Bradley Newman, Albert Skip Rizzo, and Mark Bolas. Development and evaluation of low cost game-based balance rehabilitation tool using the microsoft kinect sensor. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1831–1834. IEEE, 2011.
- [103] E. Langetepe and G. Zachmann. *Geometric Data Structures for Computer Graphics*. Ak Peters Series. Taylor & Francis, 2006.
- [104] Elmar Langetepe and Gabriel Zachmann. *Geometric Data Structures for Computer Graphics*. AK Peters, 1 edition, 2006.
- [105] Philip M. Lankford. Regionalization: Theory and alternative algorithms. *Geographical Analysis*, 1(2):196–212, 1969.
- [106] R.S. Leder, G. Azcarate, R. Savage, S. Savage, L.E. Sucar, D. Reinkensmeyer, C. Toxtli, E. Roth, and A. Molina. Nintendo wii remote for computer simulated arm and wrist

- therapy in stroke survivors with upper extremity hemiparesis. In *Virtual Rehabilitation, 2008*, pages 74–74, Aug 2008.
- [107] Jinna Lei, Xiaofeng Ren, and Dieter Fox. Fine-grained kitchen activity recognition using rgb-d. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 208–211. ACM, 2012.
- [108] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.
- [109] Olivier Lézoray and Leo Grady. *Image processing and analysis with graphs: theory and practice*. CRC Press, 2012.
- [110] Stephanie Lindsey and Cauligi S Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *Aerospace conference proceedings, 2002. IEEE*, volume 3, pages 3–1125. IEEE, 2002.
- [111] D López-Fernández, FJ Madrid-Cuevas, A Carmona-Poyato, R Muñoz-Salinas, and R Medina-Carnicer. A new approach for multi-view gait recognition on unconstrained paths. *Journal of Visual Communication and Image Representation*, 38:396–406, 2016.
- [112] Shan Lu, D. Metaxas, D. Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–443–50 vol.2, June 2003.
- [113] Thomas Maier. Distance sensors: Sound, light and vision. https://tams.informatik.uni-hamburg.de/lehre/2016ws/seminar/ir/doc/slides/ThomasMaier-Distance_Sensors_Sound_Light_and_Vision.pdf. Accessed: 2018-03-10.
- [114] Jayanta Majumder, Dracos Vassalos, Shikha Sarkar, Hyunseok Kim, Luis Guarin, Anthony York, and Terje Dahlberg. Simulation based planning of ferry terminal operations. In *Proceedings 6th International Conference on Computer and IT Applications in the Maritime Industries (COMPIT 2007), Cortona, 2007*.
- [115] Georgios Mastorakis and Dimitrios Makris. Fall detection system using kinects infrared sensor. *Journal of Real-Time Image Processing*, 9(4):635–646, 2014.
- [116] D. W. Matula and R. R. Sokal. Properties Of Gabriel Graphs Relevant To Geographic Variation Research And The Clustering Of Points In The Plane. *Geographical Analysis*, 12:205–222, 1980.
- [117] S. McCanne, S. Floyd, and K. Fall. ns2 (network simulator 2). <http://www-nrg.ee.lbl.gov/ns/>.
- [118] Fabio Menna, Fabio Remondino, Roberto Battisti, and Erica Nocerino. Geometric investigation of a gaming active device, 2011.
- [119] Benjamin F Mentiplay, Ross A Clark, Alexandra Mullins, Adam L Bryant, Simon Bartold, and Kade Paterson. Reliability and validity of the microsoft kinect for evaluating static foot posture. *Journal of foot and ankle research*, 6(1):1, 2013.

- [120] B. W. Miners, O. A. Basir, and M. S. Kamel. Understanding hand gestures using approximate graph matching. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(2):239–248, March 2005.
- [121] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2):31–50, 1996.
- [122] Niloy J Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328. ACM, 2003.
- [123] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231 – 268, 2001.
- [124] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.
- [125] Rafael Munoz-Salinas, R. Medina-Carnicer, F.J. Madrid-Cuevas, and A. Carmona-Poyato. Depth silhouettes for gesture recognition. *Pattern Recognition Letters*, 29(3):319 – 329, 2008.
- [126] Xiaopeng Ning and Guodong Guo. Assessing spinal loading using the kinect depth sensor: a feasibility study. *Sensors J*, 13(4):1139–1140, 2013.
- [127] Štěpán Obdržálek, Gregorij Kurillo, Ferda Offi, Ruzena Bajcsy, Edmund Seto, Holly Jimison, and Michael Pavel. Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population. In *Engineering in medicine and biology society (EMBC), 2012 annual international conference of the IEEE*, pages 1188–1193. IEEE, 2012.
- [128] Tom Skyhøj Olsen. Arm and leg paresis as outcome predictors in stroke rehabilitation. *Stroke*, 21(2):247–251, 1990.
- [129] John K. Ousterhout, Ken Jones, Eric Foster-Johnson, Donal Fellows, Brian Griffin, and David Welton. *Tcl and the Tk Toolkit*. Addison-Wesley Professional Computing Series. Addison-Wesley, Upper Saddle River, New Jersey, 2 edition, 2010.
- [130] Diana Pagliari and Livio Pinto. Calibration of kinect for xbox one and comparison between the two generations of microsoft sensors. *Sensors*, 15(11):27569–27589, 2015.
- [131] Robert J Palisano, Peter Rosenbaum, Doreen Bartlett, and Michael H Livingston. Content validity of the expanded and revised gross motor function classification system. *Developmental Medicine & Child Neurology*, 50(10):744–750, 2008.
- [132] Alexandros Pantelopoulos and Nikolaos G Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):1–12, 2010.
- [133] Monish Parajuli, Dat Tran, Wanli Ma, and Dharmendra Sharma. Senior health monitoring using kinect. In *Communications and Electronics (ICCE), 2012 Fourth International Conference on*, pages 309–312. IEEE, 2012.

- [134] Rick Parent. *Computer animation: algorithms and techniques*. Newnes, 2012.
- [135] Valerio Pascucci, Xavier Tricoche, Hans Hagen, and Julien Tierny. *Topological Methods in Data Analysis and Visualization: theory, algorithms, and applications*. Springer Science & Business Media, 2010.
- [136] Wei Peng and David J Edwards. K-means like minimum mean distance algorithm for wireless sensor networks. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 1, pages V1–120. IEEE, 2010.
- [137] Patrick Pérez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision*, pages 661–675. Springer, 2002.
- [138] Alexandra Pfister, Alexandre M West, Shaw Bronner, and Jack Adam Noah. Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis. *Journal of medical engineering & technology*, 38(5):274–280, 2014.
- [139] Jonathan Polley, Dionysus Blazakis, Jonathan McGee, Daniel Rusk, and John S Baras. Atemu: a fine-grained sensor network simulator. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 145–152. IEEE, 2004.
- [140] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20(2):87–93, February 1977.
- [141] Yong-Hao Pua, Zhiqi Liang, Peck-Hoon Ong, Adam L Bryant, Ngai-Nung Lo, and Ross A Clark. Associations of knee extensor strength and standing balance with physical function in knee osteoarthritis. *Arthritis care & research*, 63(12):1706–1714, 2011.
- [142] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.
- [143] Tahir Rabbani. Automatic reconstruction of industrial installations using point clouds and images. *Publications on Geodesy*, 62, 2017.
- [144] Aditya Ramamoorthy, Namrata Vaswani, Santanu Chaudhury, and Subhashis Banerjee. Recognition of dynamic hand gestures. *Pattern Recognition*, 36(9):2069–2081, 2003.
- [145] Michalis Raptis, Darko Kirovski, and Hugues Hoppe. Real-time classification of dance gestures from skeleton animation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 147–156. ACM, 2011.
- [146] George F Riley. The georgia tech network simulator. In *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pages 5–12. ACM, 2003.
- [147] Henry Rimminen, Juha Lindström, Matti Linnavuo, and Raimo Sepponen. Detection of falls among the elderly by a floor sensor using the electric near field. *IEEE Transactions on Information Technology in Biomedicine*, 14(6):1475–1476, 2010.

- [148] Javier Romero, Hedvig Kjellström, and Danica Kragic. Hands in action: real-time 3d reconstruction of hands in interaction with objects. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 458–463. IEEE, 2010.
- [149] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010.
- [150] Davud Sadihov, Bastian Migge, Roger Gassert, and Yeongmi Kim. Prototype of a vr upper-limb rehabilitation system enhanced with motion-based tactile feedback. In *World Haptics Conference (WHC), 2013*, pages 449–454. IEEE, 2013.
- [151] Sanjay Saini, Dayang Rohaya Awang Rambli, Suziah Sulaiman, Mohamed Nordin Zakaria, and Siti Rohkmah Mohd Shukri. A low-cost game framework for a home-based stroke rehabilitation system. In *Computer & Information Science (ICCIS), 2012 International Conference on*, volume 1, pages 55–60. IEEE, 2012.
- [152] Gustavo Saposnik, Robert Teasell, Muhammad Mamdani, Judith Hall, William McIlroy, Donna Cheung, Kevin E Thorpe, Leonardo G Cohen, Mark Bayley, et al. Effectiveness of virtual reality using wii gaming technology in stroke rehabilitation. *Stroke*, 41(7):1477–1484, 2010.
- [153] Shikha Sarkar, Lina Stankovic, and Ivan Andonovic. Protocol design for farm animal monitoring using simulation. In *Ad-hoc, Mobile, and Wireless Networks*, pages 126–138. Springer, 2012.
- [154] Shikha Sarkar, Lina Stankovic, Andy Kerr, and Philip Rowe. Kinect-based lower limb motion analysis. In *XXV Congress International Society of Biomechanics*, 2015.
- [155] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.
- [156] Philip J. Schneider and David Eberly. *Geometric Tools for Computer Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.
- [157] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani Srivastava. Optimizing sensor networks in the energy-latency-density design space. *Mobile Computing, IEEE Transactions on*, 1(1):70–80, 2002.
- [158] Loren Arthur Schwarz, Artashes Mkhitarian, Diana Mateus, and Nassir Navab. Human skeleton tracking from depth data using geodesic distances and optical flow. *Image and Vision Computing*, 30(3):217–226, 2012.
- [159] Michael Ian Shamos and Dan Hoey. Closest-point problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 151–162. IEEE, 1975.
- [160] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '11*, pages 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society.

- [161] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, January 2013.
- [162] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.
- [163] Lei Shu, Manfred Hauswirth, Han-Chieh Chao, Min Chen, and Yan Zhang. Nettopo: A framework of simulation and visualization for wireless sensor networks. *Ad Hoc Networks*, 9(5):799–820, 2011.
- [164] Sheldon R Simon. Quantification of human motion: gait analysis benefits and limitations to its application to clinical problems. *Journal of biomechanics*, 37(12):1869–1880, 2004.
- [165] Aneesha Singh, Annina Klapper, Jinni Jia, Antonio Fidalgo, Ana Tajadura-Jiménez, Natalie Kanakam, Nadia Bianchi-Berthouze, and Amanda Williams. Motivating people with chronic pain to do physical activity: opportunities for technology design. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 2803–2812. ACM, 2014.
- [166] Sabesan Sivapalan, Daniel Chen, Simon Denman, Sridha Sridharan, and Clinton Fookes. Gait energy volumes and frontal gait recognition using depth images. In *Biometrics (IJCB), 2011 International Joint Conference on*, pages 1–6. IEEE, 2011.
- [167] Steven S Skiena. *The algorithm design manual: Text*, volume 1. Springer Science & Business Media, 1998.
- [168] Pierre Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.
- [169] Shmuel Springer and Galit Yogev Seligmann. Validity of the kinect for gait assessment: A focused review. *Sensors*, 16(2):194, 2016.
- [170] Bruce Stephen, Cathy Dwyer, Jimmy Hyslop, Matthew Bell, David Ross, Kae Hsiang Kwong, Craig Michie, and Ivan Andonovic. Statistical interaction modeling of bovine herd behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(6):820–829, 2011.
- [171] E.E. Stone and M. Skubic. Evaluation of an inexpensive depth camera for passive in-home fall risk assessment. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference on*, pages 71–77, May 2011.
- [172] Erik E Stone and Marjorie Skubic. Passive in-home measurement of stride-to-stride gait variability comparing vision and kinect sensing. In *2011 Annual international conference of the IEEE engineering in medicine and biology society*, pages 6491–6494. IEEE, 2011.
- [173] Erik E Stone and Marjorie Skubic. Unobtrusive, continuous, in-home gait measurement using the microsoft kinect. *IEEE Transactions on Biomedical Engineering*, 60(10):2925–2932, 2013.

- [174] Erik E Stone, Marjorie Skubic, and Jessica Back. Automated health alerts from kinect-based in-home gait measurements. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2961–2964. IEEE, 2014.
- [175] Heung-II Suk, Bong-Kee Sin, and Seong-Whan Lee. Hand gesture recognition based on dynamic bayesian network framework. *Pattern Recognition*, 43(9):3059–3072, 2010.
- [176] Harsh Sundani, Haoyue Li, Vijay Devabhaktuni, Mansoor Alam, and Prabir Bhattacharya. Wireless sensor network simulators a survey and comparisons. *International Journal of Computer Networks*, 2(5):249–265, 2011.
- [177] Faezeh Tafazzoli and Reza Safabakhsh. Model-based human gait recognition using leg and arm movements. *Engineering applications of artificial intelligence*, 23(8):1237–1246, 2010.
- [178] Liansheng Tan, Yanlin Gong, and Gong Chen. A balanced parallel clustering protocol for wireless sensor networks using k-means techniques. In *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*, pages 300–305. IEEE, 2008.
- [179] Jin Tang, Jian Luo, Tardi Tjahjadi, and Yan Gao. 2.5 d multi-view gait recognition based on point cloud registration. *Sensors*, 14(4):6124–6143, 2014.
- [180] Matthew Tang. Recognizing hand gestures with microsofts kinect. *Palo Alto: Department of Electrical Engineering of Stanford University: [sn]*, 2011.
- [181] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
- [182] Karl Tombre and Bart Lamiroy. Graphics recognition-from re-engineering to retrieval. In *2013 12th International Conference on Document Analysis and Recognition*, volume 1, pages 148–148. IEEE Computer Society, 2003.
- [183] Godfried T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.
- [184] Godfried T Toussaint. *A graph-theoretical primal sketch*. MacGill University. School of Computer Science, 1986.
- [185] A. Tsanas, M.A. Little, P.E. McSharry, J. Spielman, and L.O. Ramig. Novel speech signal processing algorithms for high-accuracy classification of parkinsons disease. *Biomedical Engineering, IEEE Transactions on*, 59(5):1264–1271, May 2012.
- [186] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, Nov 2008.
- [187] András Varga et al. The omnet++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM-2001)*, volume 9, page 185. sn, 2001.
- [188] Harold Vasquez, Hector Simon Vargas, and Luis Enrique Sucar. Using gestures to interact with a service robot using kinect 2. *Research in Computing Science*, 96:85–93, 2015.

- [189] Stephanie Vernon, Kade Paterson, Kelly Bower, Jennifer McGinley, Kimberly Miller, Yong-Hao Pua, and Ross A Clark. Quantifying individual components of the timed up and go using the kinect in people living with stroke. *Neurorehabilitation and neural repair*, 29(1):48–53, 2015.
- [190] Sharad Vikram, Lei Li, and Stuart Russell. Handwriting and gestures in the air, recognizing on the fly. In *Proceedings of the CHI*, volume 13, pages 1179–1184, 2013.
- [191] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Commun. ACM*, 54(2):60–71, February 2011.
- [192] Klaus Wehrle, Mesut Günes, and James Gross. *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.
- [193] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393, 2013.
- [194] B Wiederhold and G Riva. Balance recovery through virtual stepping exercises using kinect skeleton tracking: a follow-up study with chronic stroke patients. *Annual Review of Cybertherapy and Telemedicine 2012: Advanced Technologies in the Behavioral, Social and Neurosciences*, 181:108–112, 2012.
- [195] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.
- [196] Kaixin Xu and Mario Gerla. A heterogeneous routing protocol based on a new stable clustering scheme. In *MILCOM 2002. Proceedings*, volume 2, pages 838–843. IEEE, 2002.
- [197] Xu Xu and Raymond W McGorry. The validity of the first and second generation microsoft kinect for identifying joint center locations during static postures. *Applied ergonomics*, 49:47–54, 2015.
- [198] Xu Xu, Raymond W McGorry, Li-Shan Chou, Jia-hua Lin, and Chien-chi Chang. Accuracy of the microsoft kinect for measuring gait parameters during treadmill walking. *Gait & posture*, 42(2):145–151, 2015.
- [199] Zhaojun Xue, Dong Ming, Wei Song, Baikun Wan, and Shijiu Jin. Infrared gait recognition based on wavelet transform and support vector machine. *Pattern recognition*, 43(8):2904–2910, 2010.
- [200] Yan Yan, Hanzi Wang, Si Chen, Xiaochun Cao, and David Zhang. Quadratic projection based feature extraction with its application to biometric recognition. *Pattern Recognition*, 56:40–49, 2016.
- [201] Cheng Yang, Ukadike C Ugbolue, Andrew Kerr, Vladimir Stankovic, Lina Stankovic, Bruce Carse, Konstantinos T Kalirntas, and Philip J Rowe. Autonomous gait event detection with portable single-camera gait kinematics analysis system. *Journal of Sensors*, 2016, 2016.

- [202] Lin Yang, Longyu Zhang, Haiwei Dong, Abdulhameed Alelaiwi, and Abdulmotaleb El Sadik. Evaluating and improving the depth accuracy of kinect for windows v2. *IEEE Sensors Journal*, 15(8):4275–4285, 2015.
- [203] Minxiang Ye, Cheng Yang, Vladimir Stankovic, Lina Stankovic, and Andrew Kerr. Kinematics analysis multimedia system for rehabilitation. In *International Conference on Image Analysis and Processing*, pages 571–579. Springer, 2015.
- [204] Mohammed Yeasin and Subhasis Chaudhuri. Visual understanding of dynamic hand gestures. *Pattern Recognition*, 33(11):1805–1817, 2000.
- [205] Ho-Sub Yoon, Jung Soh, Younglae J Bae, and Hyun Seung Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern recognition*, 34(7):1491–1501, 2001.
- [206] Chenyang Zhang, Yingli Tian, and Elizabeth Capezuti. Privacy preserving automatic fall detection for elderly using rgbd cameras. In *International Conference on Computers for Handicapped Persons*, pages 625–633. Springer, 2012.
- [207] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang. A framework for hand gesture recognition based on accelerometer and emg sensors. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(6):1064–1076, Nov 2011.
- [208] Ziheng Zhou, Adam Prugel-Bennett, and Robert I Damper. A bayesian framework for extracting human gait using strong prior knowledge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1738–1752, 2006.