Data-driven phase extraction for anomaly detection of repetitive human movements

Jatesiktat, Prayook

2019

Jatesiktat, P. (2019). Data-driven phase extraction for anomaly detection of repetitive human movements. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/83253>

<https://doi.org/10.32657/10220/47998>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**DATA-DRIVEN PHASE EXTRACTION
FOR ANOMALY DETECTION
OF REPETITIVE HUMAN MOVEMENTS**

**PRAYOOK JATESIKTAT
SCHOOL OF MECHANICAL AND AEROSPACE
ENGINEERING**

2019

**DATA-DRIVEN PHASE EXTRACTION
FOR ANOMALY DETECTION
OF REPETITIVE HUMAN MOVEMENTS**

PRAYOOK JATESIKTAT

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2019

PRAYOOK JATESIKTAT

Data-driven Phase Extraction for Anomaly Detection of Repetitive Human Movements

Nanyang Technological University

School of Mechanical and Aerospace Engineering

Robotics Research Centre

50 Nanyang Avenue

N3-01a-01

Singapore 639798

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

04 - Jan - 2019

Date

Prayook Jatesiktat

Prayook Jatesiktat

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

04-Jan-2019

.....
Date

.....
Wei Tech Ang

Authorship Attribution Statement

This thesis contains material from four papers published in the following peer-reviewed conferences where I am the first author.

Parts of Chapter 3 are published in P. Jatesiktat and W. T. Ang, “Recovery of forearm occluded trajectory in kinect using a wrist-mounted inertial measurement unit,” in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Seogwipo, South Korea, 2017, pp. 807–812. DOI: 10.1109/EMBC.2017.8036947. The contributions of the co-authors are as follows

- Assoc. Prof. Wei Tech Ang provided the initial project direction and edited the manuscript draft.
- I designed, implemented the system, analyzed the evaluation result, and wrote the manuscript.

Chapter 4 is published as P. Jatesiktat, D. Anopas, and W. T. Ang, “Personalized markerless upper-body tracking with a depth camera and wrist-worn inertial measurement units,” in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Honolulu, HI, USA, 2018, pp. 1–6. DOI: 10.1109/EMBC.2018.8513068. The contributions of the co-authors are as follows

- Assoc. Prof. Wei Tech Ang provided the initial project direction and edited the manuscript draft.
- Dollaporn Anopas facilitated the subject recruitment and the data collection.
- I designed, implemented the system, and analyzed the evaluation result, and wrote the manuscript.

Chapter 5 is an improved version of P. Jatesiktat, D. Anopas, and W. T. Ang, “Unsupervised phase learning and extraction from repetitive movements,” in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Honolulu, HI, USA, 2018, pp. 227–230. DOI: 10.1109/EMBC.2018.8512196. The contributions of the co-authors are as follows

- Assoc. Prof. Wei Tech Ang provided the initial project direction and edited the manuscript draft.

- Dollaporn Anopas designed the treadmill and implemented the feature extraction from rat's gait.
- I designed and implemented the phase learning and extraction algorithm, and wrote the manuscript.

Chapter 7 is published as P. Jatesiktat, M. J. Foo, G. M. Lim, *et al.*, "Sdf-net: Real-time rigid object tracking using a deep signed distance network," in *2018 International Conference on Computational Science (ICCS)*, Y. Shi, H. Fu, Y. Tian, *et al.*, Eds., Wuxi, China: Springer International Publishing, 2018, pp. 28–42, ISBN: 978-3-319-93698-7. DOI: 10.1007/978-3-319-93698-7_3. The contributions of the co-authors are as follows

- Assoc. Prof. Wei Tech Ang provided the initial project direction and edited the manuscript draft.
- Guan Ming Lim did the literature survey and facilitated the data collection for the evaluation. He also wrote the manuscript.
- Ming Jeat Foo co-developed with me to produce the optimization-based fitting approach (Section 7.1.6) and implemented the method for evaluation. He also wrote the manuscript.
- I designed and implemented neural networks to learn signed distance function of objects. I designed and implemented the ICP-based fitting approach (Section 7.1.5) for evaluation. I also wrote the manuscript.

04 - Jan - 2019

Date

Prayook Jatesiktat

Prayook Jatesiktat

Acknowledgments

First of all, I would like to thank my supervisor, Professor Wei Tech Ang, for accepting me to this Ph.D. journey in the first place. His understanding of my different background and his openness to my proposed solutions really give me enjoyable research experiences.

During my first two years, my research work would not make much progress without a helping hand from my senior, Yan Niang Aye. His insights of electrical circuit design from his project have been a great boost for my work. I also appreciate much help from Shee Cheng Yap who is always there to answer my technical questions. Towards the end, I would not be able to collect valuable data from stroke patients without the help from Christopher Kuah. His expertise as an occupational therapist really helps to push my project forward.

One impressive thing of being in this research group is the experience in research collaborations. Thanks to highly capable colleagues, Ming Jeat Foo, Guan Ming Lim, and Dollaporn Anopas who give commitments towards the hard work and share the excitement of meeting paper submission deadlines with me. In addition, thanks to friends in my research group, Jie Kai Er and Carlo Tiseo, who have never denied my request for help.

Besides the academic stuff, the network of Thai graduate students in NTU can always provide me with tricks and trips of how not to miss important things in my school. Without them, the school life would be difficult and full of mistakes.

I would like to especially thank my on-demand English advisor, Manasawee Wongpradoo, who kindly proofread all my papers and helped me with my English even before I entered this Ph.D. program.

Finally, I would like to express my gratitude to every event in the past that built me up to the current version of me. Thanks for all the love and care I got from my family. Thanks to all my teachers who give me seeds of knowledge and wisdom in the past. Many of those seeds have unexpectedly grown up in me and start to bear some fruits.

Contents

Acknowledgement	7
Table of Contents	9
List of Figures	15
List of Tables	19
Abstract	21
1 Introduction	23
1.1 Background	23
1.2 Objective	24
1.3 Scope	25
1.4 Organization	26
2 Literature Review	27
2.1 The Need for Supervised Therapy at Home	27
2.2 Existing Sensors/Systems for 3D Motion Capture	28
2.2.1 Non-optical Systems	28
2.2.2 Optical Systems	30
2.3 Quantitative Assessment Based on Kinematic Measures	34
2.3.1 Functional Range of Motion	35
2.3.2 Movement Speed	35
2.3.3 Movement Efficiency & Accuracy	36
2.3.4 Movement Smoothness	36
2.3.5 Inter-Joint Coordination	37
2.3.6 Control Strategy	37
2.3.7 Torque Production	37

2.3.8	Compensatory Movement	37
2.4	Existing Commercial Home Rehab Platform	38
2.4.1	Kinetisense	38
2.4.2	Jintronix Rehabilitation System	38
2.4.3	VirtualRehab	39
2.4.4	Vera System	40
2.5	Research Gaps	40
2.5.1	Hardware/Sensing System	40
2.5.2	High-Level Movement Analysis	41
3	Hardware Design & Integration	43
3.1	System Overview	43
3.1.1	Hardware	44
3.1.2	System Configurations	44
3.1.3	Software Implementation	45
3.2	Sensor Node Prototype Design	45
3.2.1	Power Supply System	47
3.2.2	Clock Fusion	48
3.2.3	Form Factor & Case Design	48
3.3	Wireless Communication	49
3.3.1	SimpliTI Protocol	49
3.4	Time Synchronization	51
3.4.1	Sound-based Synchronization	51
3.4.2	Broadcast-based Synchronization	51
3.5	Sensor Calibration	52
3.5.1	Magnetometer Calibration	52
3.5.2	Accelerometer Calibration	53
3.5.3	Gyroscope Bias Compensation	53
3.5.4	Kinect Depth Camera Projection Model	54
3.6	Reference Frames	54
3.7	Quaternion Notation	55
3.8	Quaternion-based Kalman Filter	55
3.9	Transformation Between Reference Frames	57
3.9.1	Vicon-to-Kinect Transformation	57
3.9.2	World-to-Kinect Transformation	58
3.10	IMU Orientation in Kinect Reference Frame	59

4	Upper-body Pose Estimation	61
4.1	Pose Estimation Method	63
4.1.1	Frame Inputs & Expected Outputs	63
4.1.2	Human Mesh Model	63
4.1.3	Parameterization of a Point on the Mesh	66
4.1.4	Energy Function	67
4.1.5	Vector of Residuals	70
4.1.6	Levenberg-Marquardt (LM) Iteration	70
4.1.7	Jacobian Calculation	72
4.1.8	Pose Initialization	74
4.1.9	Sparse Sampling from Depth Pixels	74
4.1.10	Correspondence Continuous Update (Mesh walk)	75
4.1.11	Correspondence Discrete Update	75
4.2	Model Personalization	76
4.2.1	Rough Shape Estimation using Gradient Descent	76
4.2.2	Fine Shape Tuning with Shape & Pose Fitting	77
4.3	Evaluation	78
4.3.1	Gold Standard Measurement	78
4.3.2	Subjects & Movement Sequences	78
4.3.3	Parameter Configurations	79
4.4	Results & Discussions	80
4.4.1	Model Personalization	80
4.4.2	Quantitative Pose Estimation Results	81
4.4.3	Qualitative Improvement Over Kinect SDK	83
4.4.4	Limitations	85
4.5	Upper-body Features Extraction	86
4.6	Conclusion	88
5	Unsupervised Phase Learning & Extraction	91
5.1	Phase Learning & Extraction Method	93
5.1.1	Method Overview	93
5.1.2	Neural Network Architecture	93
5.1.3	Training Data Preparation	95
5.1.4	Cost Function	96
5.1.5	Initialization & Optimization	100
5.1.6	Phase Extraction	100

5.2	Validation Experiment on Synthetic Data	101
5.2.1	Synthetic Signal Generator	101
5.2.2	Signal Disturbance	102
5.2.3	Training Configuration	102
5.2.4	Results & Discussions	103
5.3	Experiment on Repetitive Human Movements	104
5.3.1	Data Source	104
5.3.2	Data Preprocessing	105
5.3.3	Training Configuration	108
5.3.4	Results & Discussions	108
5.4	Additional Discussions	115
5.4.1	Flip Mapping Problem	115
5.4.2	Constant Phase Offset	116
5.4.3	Average Movement Extraction	119
5.5	Conclusion	119
6	Anomaly Detection in a Repetitive Movement	121
6.1	Anomaly Detection Method	123
6.1.1	Feature Extraction	124
6.1.2	Phase Learning & Extraction	125
6.1.3	Phase Sequence Smoothing	125
6.1.4	Monotonically Progressing Phase Assignment using Circular Dynamic Time Warping (CDTW)	126
6.1.5	Anomaly Score from Phase Progression	128
6.1.6	Period Segmentation	130
6.1.7	Detect Anomaly with Feature Reconstruction Error	130
6.2	Quantitative Experiment on Healthy Subjects	134
6.2.1	Evaluation Method	135
6.2.2	Results: Receiver Operating Characteristic (ROC) Curve	138
6.2.3	Results: Dimensionality Bottleneck	141
6.2.4	Examples of Detected Anomalies	143
6.2.5	Undetected Anomaly	148
6.2.6	False Anomaly	149
6.3	Qualitative Experiment on Unhealthy Subjects	150
6.3.1	Evaluation Method	150
6.3.2	Results & Discussions	150

6.4	Conclusion	160
7	Rigid Object Tracking Using a Deep Signed Distance Network	161
7.1	Rigid Object Tracking Method	163
7.1.1	Method Overview	163
7.1.2	Building a Signed Distance Network	164
7.1.3	Current Frame Pose Prediction	166
7.1.4	Object Pixel Sampling	167
7.1.5	ICP-based Fitting Approach	168
7.1.6	Optimization-based Fitting Approach	169
7.2	Evaluation	171
7.2.1	Quantitative Evaluation	171
7.2.2	Qualitative Evaluation	171
7.3	Results & Discussion	172
7.3.1	The Trained Networks	172
7.3.2	Object Tracking	173
7.4	Limitation	177
7.5	Towards Real-time Articulated Object Tracking	177
7.5.1	Articulated Signed Distance Field Network (ASDF-Net)	178
7.5.2	Data Generation and Training	179
7.5.3	Notations	179
7.5.4	Network Validation	180
7.5.5	Tracking Method	180
7.5.6	Preliminary Evaluation	181
7.5.7	Preliminary Results	181
7.5.8	Future Research Direction	182
7.6	Conclusion	183
8	Conclusion	185
8.1	Contributions	185
8.2	Future Research Direction	186
A	Circuit Design	189
B	Subject-Related Information	193
B.1	Subject Group 1	193
B.2	Subject Group 2	193

B.2.1	Healthy Subjects	193
B.2.2	Unhealthy Subjects	194
Bibliography		197

List of Figures

2.1	Full-body configuration of Xsens MVN BIOMECH Link	29
2.2	Perception Neuron System	29
2.3	Opal system	29
2.4	Nexonar system	30
2.5	Track STAR system	30
2.6	OptiTrack minimum configuration with six cameras	31
2.7	Microsoft Kinect for Xbox 360	31
2.8	Main components of Intel RealSense 3D R200 Camera	32
2.9	Microsoft Kinect for Xbox One with Time-of-Flight depth camera. .	33
2.10	A screen capture from Kinetisense during a range of motion assessment.	38
2.11	Rehabilitation exercise using JRS Wave	39
2.12	VirtualRehab screen shots	40
2.13	Vera system	40
3.1	Interaction diagram of hardware components in the system	44
3.2	Hardware prototype	45
3.3	Kinect V2 with one IMU mounting on top using a 3D-printed case .	46
3.4	Key components of the designed IMU and their logical connections.	46
3.5	Key modules in the power supply system	48
3.6	IMU case	49
3.7	The sensor node in the 3D-printed case.	49
3.8	Comparison between non-pipelined and pipelined implementations.	51
3.9	Data flow diagram of quaternion-based Kalman filter.	56
3.10	Calibration panel	57
3.11	Kinect reference frame and base-IMU reference frame	59

4.1	Original high-resolution SMPL template meshes and the modified low-resolution SMPL template meshes	65
4.2	Joint positions and their degree-of-freedom	66
4.3	Jacobian matrix structure	72
4.4	Flowchart of one Levenberg-Marquardt (LM) iteration.	73
4.5	Calculation of pose estimation in one frame	75
4.6	Recording environment for evaluation	78
4.7	Upper-body marker positions defined by Plug-in Gait model	79
4.8	A subject in the personalization process	80
4.9	Shape fitting results from six subjects in a controlled pose	81
4.10	Accuracy profiles at all threshold distances	82
4.11	Comparisons of joint position errors	83
4.12	Qualitative comparisons between Kinect SDK, the proposed method, and the gold standard	85
4.13	Joint locations and their ID for reference	86
5.1	Neural network architecture for phase extraction	94
5.2	Calculation of phase progression penalty from one pair of two consecutive input windows.	97
5.3	The phase progression penalty profile function $\mathcal{P}$	98
5.4	Phase collapsing issue	99
5.5	Three synthetic signals used in the experiment: heart, bow, and acrobat shape.	101
5.6	Learning results from synthetic signal at the 5000th iteration	103
5.7	Comparisons between the ground truth phase sequences and the extracted phase sequences	104
5.8	Spherical Rotation and Azimuthal Equidistant Projection	107
5.9	Progression of phase learning from four repetitive exercises	109
5.10	Pre-phase distribution of COMB exercise from unhealthy subjects	111
5.11	Pre-phase distribution of RECT exercise from unhealthy subjects	112
5.12	Pre-phase distribution of TAP exercise from unhealthy subjects	113
5.13	Pre-phase distribution of POUR exercise from unhealthy subjects	114
5.14	Extracted phase sequences from unseen RECT movements	116
5.15	Extracted phase sequences from unseen TAP movements	117
5.16	Extracted phase sequences from unseen POUR movements	117
5.17	Extracted phase sequences from unseen COMB movements	118

5.18	Extracted phase sequences from a messy movement	118
5.19	Plots of average movement features against extracted phases	119
5.20	Average trajectory of forearm pointing direction in AZEQ representation	120
6.1	A data flow diagram of all the key steps in the anomaly detection .	124
6.2	Comparison of the data structure used in the original DTW and Circular DTW	127
6.3	The template sequence used in cyclic dynamic time warping	128
6.4	Comparison between an extracted phase sequence, its smoothed phase sequence, and its approximately closest monotonically progressing (ACMP) phase sequence	130
6.5	Multiple stages of data processing to calculate reconstruction error of a period.	131
6.6	Reconstruction of one data channel during a segmented period. . .	133
6.7	Receiver Operating Characteristic (ROC) curve from four exercises.	140
6.8	Anomaly detection result from sequences of COMB and RECT exercises with simulated anomalies	144
6.9	Anomaly plots from a COMB exercise sequence with excessive torso compensation	145
6.10	Anomaly plots from a POUR exercise sequence without the forearm pronation.	146
6.11	Anomaly plots from a TAP exercise sequence with some leaks of period reconstruction error	147
6.12	Anomaly plots from a POUR exercise sequence with high phase fluctuation	147
6.13	Anomaly plots from a COMB exercise sequence with undetected anomalies	148
6.14	Anomaly plots from a POUR exercise sequence with false anomaly detection	149
6.15	Anomaly plots from a RECT exercise sequence by an unhealthy subject STV	151
6.16	Anomaly plots from a RECT exercise sequence by an unhealthy subject KKT	152
6.17	Anomaly plots from a RECT exercise sequence by an unhealthy subject LSL	152

6.18	Anomaly plots from a COMB exercise sequence by an unhealthy subject LKL	153
6.19	Anomaly plots from a COMB exercise sequence by an unhealthy subject TKB	154
6.20	Anomaly plots from a COMB exercise sequence by an unhealthy subject CMT	154
6.21	Anomaly plots from a COMB exercise sequence by an unhealthy subject NAH	155
6.22	Anomaly plots from a TAP exercise sequence by an unhealthy subject CST	156
6.23	Anomaly plots from a TAP exercise sequence by an unhealthy subject LSL	156
6.24	Anomaly plots from a TAP exercise sequence by an unhealthy subject NRH	157
6.25	Anomaly plots from a TAP exercise sequence by an unhealthy subject NHC	157
6.26	Anomaly plots from a POUR exercise sequence by an unhealthy subject SGH	158
6.27	Anomaly plots from a POUR exercise sequence by an unhealthy subject QSL	159
6.28	Anomaly plots from a POUR exercise sequence by an unhealthy subject NAV	159
7.1	Intuition of using SDF-Net in a rigid object tracking	164
7.2	Object surface reconstruction from the converged point cloud . . .	173
7.3	Examples from the object tracking in different scenarios	176
7.4	Comparisons between SDF-Net and ASDF-Net	178
7.5	Human surface reconstruction from the converged point cloud . . .	180
7.6	Full-body tracking results using ASDF-Net	182
A.1	Schematic diagram 1. Microcontroller, wireless module, and FRAM	189
A.2	Schematic diagram 2. Accelerometer, gyroscope, magnetometer, and barometer	190
A.3	Schematic diagram 3. Battery charging module and voltage regulator	190
A.4	3D-rendering of the designed PCB (front)	191
A.5	3D-rendering of the designed PCB (back)	191

List of Tables

3.1	Requirement specifications of the IMU design.	43
6.1	Area under the curve (AUC) from four exercises and the accuracies at the optimal point	139
6.2	Average numbers of principal components needed to explain 95% of variance in one period of each feature channel	142
7.1	Comparisons of the RMS errors in translation, orientation, and the runtime of different object tracking algorithms	175
7.2	Comparison of our object tracking methods using two different net- works	177

Abstract

Human movements during a specific task usually consist of inconsistency and variations. They are caused by different strategies, the pace of movement, or even anthropometric structure of each subject. This dissertation aims to develop a norm modelling methodology that can model a generic repetitive movement in a data-driven way from a collection of the movement from healthy subjects without any manual data labelling or segmentation. Then, this model can be used to assess the abnormality of the movement during rehabilitation exercises of stroke patients. To make the method invariant to the variation in phase progression, a neural network is trained to extract phases from unlabeled time-series of kinematic features. This unsupervised learning method enhanced the generalization of the modelling process to be applied to a wide range of repetitive exercises without the need for handcrafting new features. The trained model is used to segment and time-warp the exercise data into standardized periods. This standardization allows the principal component analysis reconstruction to be applied for anomaly detection. The whole modelling workflow not only works on a gold-standard marker-based motion capture system, but it is also compatible with our motion capture system which integrates a Kinect sensor with wrist-worn inertial measurement units (IMU). The compactness and affordability of the system allow this method to be used in a home-based rehabilitation scenario where space and cost are the constraints. An evaluation on four different exercises shows good potential in separating anomalous movements from normal ones with an average area under the curve (AUC) of 0.9872 and the test on actual stroke survivors is also performed. In addition, a new real-time depth-based object tracking method is invented by training a neural network to learn a continuous signed distance field of a target object. Its flexibility allows the method to be applied to various rigid and articulated objects such as a human body.

Chapter 1

Introduction

1.1 Background

The worldwide number of stroke survivors was about 62 million in 2005 and expected to increase to 77 million in 2030 [1]. With a nearly constant rate, the need for rehabilitation and long-term care will grow approximately 24% between 2007 and 2030 [1]. With a similar trend, the world's population is growing toward an aging society. United Nation predicts that the proportion of people who age 60 years or more will increase to 21.5% globally in 2050 [2]. This trend is even faster in highly developed areas such as Europe or North America. From a physical ability perspective, these two groups of people, *i.e.*, post-stroke patients and the frail elderly, commonly need extra care in a form of rehabilitation exercises and assessments.

This scenario is putting a big strain on hospitals and public healthcare systems. For a stroke patient to relearn his missing abilities, he needs a considerable amount of rehabilitation before the improvement plateaus in six months [3]. However, the limited healthcare resource and the high cost of being inpatient force them to discharge after a month of intensive rehabilitation in the hospital. Unlike hospitals, the continuity of care at patients' home is very limited as a therapist can visit just a few homes in a day. A patient cannot meet their therapist every day [4]. As a result, their exercises at home are always self-directed and their therapists do not know whether the exercises are done properly and frequently as prescribed. Furthermore, it is difficult for a patient to assess their own performance quantitatively and to honestly report to their therapist without bias.

Tele-rehabilitation technologies are used to mitigate the problem by allowing therapists to monitor their patient remotely and reduce their time spent on traveling. Still, the direct monitoring by replaying the whole exercise session is not scalable.

If a patient performs a daily prescribed exercise for 30 minutes, the only way for a therapist to generate insightful feedback for them is to watch through the whole 30 minutes of the exercise session to find out what is wrong in the movements. With increasing stroke survivors doing rehabilitation workout every single day, we will never have enough therapist to work on this inefficient process of feedback generation.

One way to make this feedback generation more efficient is to present the whole exercise record to the therapist in a more compact and understandable form. For example, on a single timeline, it could show the boundary of each repetition with an estimation of how well the patient performs in each repetition. At a glance, the therapist can choose a few short sections that he should pay attention to and watch the video record of those few sections to generate feedback. This cooperation between a clinician and the machine could improve the quality of the tele-rehabilitation and make it more scalable.

However, summarizing a series of human movement into such compact information on a timeline required a full stack of an automated system to sense, understand, and interpret the data. This study focuses on engineering all the missing components starting from the hardware layer to the human motion capture and the high-level analysis of the movement. To practically achieve the goal, this sensing system must meet the balance between its required space, accuracy, ease of use, and affordability.

1.2 Objective

The objective of this dissertation is to develop a home-based sensing system that captures the kinematics of upper-body movement and assess the abnormality of prescribed exercises during rehabilitation sessions of stroke patients. The direction of research and development of the sensing system are pushed towards the objective under following aims.

- To integrate of off-the-shelf depth camera (Kinect V2) and a few newly designed inertial measurement units (IMU). The integration includes synchronization, sensor fusion, and a short-range wireless communication infrastructure.
- To enhance the measuring capability of kinematic information of gross upper-body movements (*i.e.*, torso, shoulder, elbow, and wrist) over the Kinect alone

through the additional sensors and subject personalization technique.

- To design a data processing methodology that automatically learns to flag anomalous sections and predict what could be wrong in a repetitive rehabilitation exercise. This capability can help therapists save time in monitoring and give insightful feedback to their patients. The method should be as automatic as possible starting from the healthy norm model generation to the anomaly detection. This will allow the number of exercises in the system to expand without engineering new features but by collecting more exercise data from healthy subjects.
- To explore the use of artificial neural networks in real-time tracking of an arbitrary object including rigid and articulated objects. This exploration could benefit the future stage of the project which will cover the entire upper limb (*i.e.*, proximal and distal) and will require the sensing of hand-object interactions.

1.3 Scope

- This study targets the post-stroke patient group with enough capability to follow their prescribed upper-limb movements without any assistance. Although this technology has the potential of serving a wider group of patients with different pathologies, stroke is the first one to receive our attention due to its high rate of incident.
- The proposed sensing system is limited to kinematic information of gross upper-body movements (*i.e.*, torso, shoulder, elbow, and forearm) and does not include fine movements of hand and fingers. This sensing system is not designed to have a better accuracy over the marker-based motion capture system which is a current gold standard. Instead, it is designed to find a suitable point for home-based upper-limb rehabilitation which the trade-off between compactness, accuracy, and affordability is considered.
- Information generated from this system is not directly targeting the patients but at assisting therapists in monitoring their patients' daily exercise sessions. As the feedback from a therapist can be delayed, the processing of the kinematic features and its digest do not need to be produced in real-time during the exercise session.

- The type of exercise allowed to be added to the system is limited to repetitive exercises of unilateral upper-limb movement only. Those exercises are assumed to contain only poses that the subject faces toward the camera.
- This thesis mainly focuses on multiple data processing methodologies that allow the objective to happen. The scope does not include the full application design and the deployment which requires graphical user interfaces and a client-server infrastructure.
- This thesis is not focusing on the implementation of another tele-rehabilitation platform to have more feature sets or to be superior to all other existing platforms in all cases. Instead, this thesis is focusing on filling some gaps that the current platforms cannot provide.

1.4 Organization

This thesis consists of eight chapters including this introduction. The rest of the thesis is organized as follows.

Chapter 2 reviews existing motion capture systems, how they are used in movement assessment, and capabilities of commercial tele-rehabilitation systems. The chapter also identifies some research gaps.

Chapter 3 focuses on the design of the inertial measurement unit (IMU) and how the integration with the Kinect is done. The aim of this chapter is to retrieve the synchronized IMU orientation in the Kinect reference frame.

Chapter 4 explains the human pose tracking method. The extracted kinematic data will be used in Chapter 5 and 6 to model and extract useful information.

Chapter 5 proposes an unsupervised way for an artificial neural network to learn how to extract phase information from a repetitive movement without the need for manual segmentation.

Chapter 6 utilizes information from Chapter 4 and 5 to detect anomalies in the repetitive exercise. The method is also tested on healthy subjects and post-stroke subjects.

Chapter 7 proposes a way to track an arbitrary rigid object using a single depth camera in real-time. The chapter also extends the method toward a more general case of articulated objects for human tracking.

Finally, the thesis is concluded in Chapter 8 with contributions and future research directions.

Chapter 2

Literature Review

2.1 The Need for Supervised Therapy at Home

In Singapore, the limited healthcare resource and the high cost of being inpatient usually force a stroke survivor to discharge after a month of supervised therapy in a hospital. After the discharge, a patient rarely receives home-based rehabilitation by a therapist because it is expensive and not widely available in Singapore [5]. A patient will receive those exercise prescriptions from a therapist before the discharge to perform at home after discharge. This practice forces the patient to follow the prescribed exercises at home in an unsupervised or self-directed manner without any feedback or monitoring from his therapist. In a better scenario, monitoring service from a therapist in a form of home visit can only happen about nine times on average during the first three months after stroke and just two times during the three following months [4].

However, it is known that the supervised stroke rehabilitation during the first six month post-stroke was associated with greater and faster recovery in comparison to unsupervised therapy [5]. The current care model that could help them to increase the participation rate in the supervised therapy is the community-based rehabilitation. Nonetheless, accessibility to a community rehab center in some locations can be limited and most of the daily exercises are still at home. To make this home-based rehabilitation become more *supervised*, some kind of feedback from a therapist is needed. One unanswered question in the field of stroke rehabilitation is “How do physiotherapists deliver feedback in current practice?” [6]. For the context of this study, the question can be paraphrased to “with the limitation in the therapists’ time resource, how can they efficiently deliver the feedback to their stroke patients at home on their daily exercise routine?”

2.2 Existing Sensors/Systems for 3D Motion Capture

To search for the most suitable sensing combination, various types of systems have been reviewed. Under the scope of kinematic sensing, they can be separated into two groups which are non-optical and optical.

2.2.1 Non-optical Systems

Most of non-optical systems are based on Inertial Measurement Unit (IMU) which means accelerometer, gyroscope, and/or magnetometer. However, other types of sensors such as electromagnetic wave and ultrasonic wave are also used. One key advantage of such systems is that there is no occlusion.

Full-body motion capture systems such as Xsens MVN BIOMECH Link (Figure 2.1) [7], and Perception Neuron (Figure 2.2) [8] are designed by using a combination of multiples nine-axis IMUs (accelerometer, gyroscope, and magnetometer) attached on human body segments to be measured. Configuration can vary from 17 IMUs for full-body capture to 11 IMUs for upper body capture, and to only six IMUs for a single-sided upper body capture. Perception Neuron has a special configuration with 21 IMUs for fingers motion capture. These systems can give an animated human skeleton in real-time based on their own sensor fusion algorithms. Setup time for this kind of system is around 5-10 minutes for a healthy subject and could be much longer for a stroke patient. These IMU-based systems have a wide range of prices starting from about 2,000 SGD for Perception Neuron to 58,700 SGD for Xsens MVN BIOMECH Link. However, even with the more affordable system, the main problem with this kind of system in home rehab scenario is the difficulty for a patient to suit up all the sensors properly.

Without human skeleton tracking capability, there are a few types of generic motion capture sensors. APDM Opal (Figure 2.3) [9] is a fully wireless nine-axis IMU sensor system with ISM band wireless mesh synchronization. Each node has built-in sensor fusion for absolute quaternion orientation (3 DOF). The system with six nodes costs about 20,000 SGD.

Nexonar (Figure 2.4) [10] is a wireless motion tracking system using ultrasonic pulses generated from each Nexonar Beacon. Nexonar 3DListener and multiple beacons are used for six DOF tracking with 0.1 mm tracking accuracy. However, this system can track only four beacons per 3DListener and the price starts from 6,750

Figure 2.1: Full-body configuration of Xsens MVN BIOMECH Link

Figure 2.2: Perception Neuron System (left). Upper body configurations (right)

Figure 2.3: Opal system. access point (left) and trackers (right)

Figure 2.4: Nexonar 3DListener (left) and Mini Triplet Beacon (right)

Figure 2.5: TrackSTAR. Mid-Range Transmitter (left) and Model 800 6-DOF Sensor (right)

SGD with only one 3DListener.

TrackSTAR (Figure 2.5) [11] is a motion tracking system using Electromagnetic transmitter to generate a tracking volume and sensors to track position and orientation in six DOF. It has 1.4 mm RMS and 0.5 degrees RMS accuracy in small tracking range of 660 mm from the transmitter. The sensor node is wired and mainly designed for tracking inside the human body for surgical purposes.

Ultra-WideBand radio positioning system (UWB) is also used in motion capture [12]. This kind of systems usually composes of moving nodes that can generate UWB pulse and a few antennas that will receive the signal at different points in the time domain. The differences are used to calculate the 3D position of the moving node. The most accurate commercial system is from Time Domain[®]. The accuracy of this system ranges from 10 cm in non-line-of-sight (NLOS) condition to 2 cm in line-of-sight (LOS) condition [13]. However, the moving node is large and expensive. Each moving node (P410 module) costs about 2,700 SGD.

2.2.2 Optical Systems

For 3D imaging, there are three techniques used in commercial products nowadays which are stereoscopic vision, structured light, and time-of-flight technique [14].

The stereoscopic vision technique uses two or more cameras at different angles to retrieve images. Common features in both scenes are extracted and used to triangulate the distance and recover the depth of each feature point. However,

Figure 2.6: OptiTrack configuration with six cameras (minimum setting)

Figure 2.7: Microsoft Kinect for Xbox 360 with IR projector, RGB camera, and IR Sensor (left to right)

markers can be used to replace those feature points such as OptiTrack (Figure 2.6) [15] and Vicon [16], [17]. While these systems can provide sub-millimeter accuracy in a large capture volume—1.5 m diameter x 2.1 m for a small volume setting, they also require large setup area—6 m diameter for a minimum setting which costs high up to 10,150 SGD—and considerable expertise in setting up and use. Most importantly, they need accurate marker attachments on the patient's body which is the main problem in real clinical practice [18]. However, they are mainly used as a gold standard to evaluate the accuracy of various state-of-the-art motion capture techniques.

The structured light technique uses an infrared light source to project a pattern onto a surface. The deformation of the pattern is recorded by an infrared camera to calculate the depth of each area using parallax. Microsoft Kinect (version 1) (Figure 2.7) is a depth camera with structured light technique released in 2010 mainly for a gaming purpose. The sensor comes with a robust skeleton extraction library which is trained on a randomized decision forest on synthetic training data [19]. With an affordable price of 349 SGD, the device becomes popular in research and development communities. However, the production line has been stopped due to the released of Kinect Version 2.

Kinect V1 has been extensively evaluated in the literature. For general indoor depth measurement, the random error ranges from 2-3 mm at 0.5 m distance up to around 4 cm at 5 m distance [20]. The result also shows the artifact from depth

Figure 2.8: Main components of Intel RealSense 3D R200 Camera. It needs two IR cameras for the stereo depth imaging.

quantization. The point spacing between two consecutive quantized levels in depth direction is around 2 mm at 1 m distance, 2.5 cm at 3 m distance, and raises to 7 cm at 5 m distance. This artifact can be mitigated by a real-time depth filtering with GPU [21]. Another evaluation [22] compared the Kinect with the Vicon system as the gold standard. Over the range of 1-3 m, Kinect has an RMS error of 10.9 mm in the depth direction and around 6 mm for the vertical and horizontal direction. According to those results, measurement beyond 3 m distance from the sensor should be avoided.

Pfister *et al.* compare Microsoft Kinect to Vicon system in gait analysis [23] and conclude that the Kinect's measurement is not consistent enough for a clinical assessment. Its sensitivity needs improvement before clinical implementation. Mokhtarzadeh *et al.* also experiment similar comparison for elbow and knee flexion and extension [24]. The large difference up to 20 degrees of elbow flexion angle shows an inconsistency of the sensor. Another type of inconsistency is also reported by Obdrzalek *et al.* in [25] that leg length variability can be as high as 10 cm since it is difficult to determine knee location while the legs are straightened. This variation also applies to the arm length.

However, Webster and Celik used Kinect to measure wrist, elbow, and shoulder positions in comparison to OptiTrack system [26]. They support the Kinect to be sufficiently accurate and responsive enough for gross movement-based impairment assessment and rehabilitation progress tracking. The conclusion agreed with Galna *et al.* in [27] that Kinect is excellent for gross movements but they also added that it is very poor for fine movement such as hand clapping. Moreover, they stated that the measurement of forearm pronation/supination is not included in the Kinect skeletal model.

Released in 2015, Intel RealSense 3D R200 Camera (Figure 2.8) [28] is a depth camera which is miniaturized for mobile device (100 mm x 9 mm x 3.8 mm). Similar

Figure 2.9: Microsoft Kinect for Xbox One with Time-of-Flight depth camera.

to the first version of Microsoft Kinect, R200 comes with an infrared projector to create a structured light pattern on the environment. However, R200 includes two infrared sensors that allow R200 to use active and passive stereoscopic vision technique together to get better range and depth accuracy. In late 2017, two improved versions, Intel RealSense D415 and D435, have been released with the price of 200 SGD and 235 SGD respectively. They work on the same principal as R200 but with a wider range (0.16 - 10 m) and higher frame rate (up to 90 fps). Unlike Microsoft Kinect, Intel RealSense cameras do not provide human skeleton tracking in their application programming interface (API). Therefore, the cameras are not seen to be used in physical rehabilitation.

The time-of-flight technique uses an infrared light source to project a modulated wave to a surface. The phase difference of the reflected waves is translated into the depth of that area. Microsoft Kinect V2 (Figure 2.9) is a time-of-flight camera released in 2014. Unlike the first version, it provides a wider field-of-view ($70^\circ \times 60^\circ$) and a better signal to noise ratio for the depth map compared to its predecessor [29] but it cost less at 265 SGD.

Yang *et al.* evaluated five attributes about depth measurement of Kinect V2 [30]. Their accuracy distribution map shows that the accuracy error is below 4 mm within 3 m from the sensor which is significantly improved from its predecessor. Moreover, depth quantization is also improved since the point spacing between two consecutive quantized levels is just about 4 mm on average at 3 m distance.

In contrast, focusing on the skeleton extraction reveals inconsistent results. Clark *et al.* tested the Kinect V2 in an assessment of standing balance and postural control [31]. Concurrent validity in comparison with a marker-based three-dimensional motion analysis (Vicon) was observed to be excellent for anterior-posterior displacement range but poor for medial-lateral displacement range. In another similar experiment by Mentiplay *et al.*, concurrent validity of Kinect V2 was retrieved in gait

assessment task and concluded that it may not give accurate lower body kinematic data due to its consistently being poor for modest validity score [32]. Moreover, Xu and McGorry also state that Kinect V2 does not substantially improve the accuracy of joint center location identification compared to the first version [33].

Even though Kinect V2 comes with a large improvement in depth accuracy, it still has drawbacks in its skeleton extraction which needs to be improved. Moreover, since it uses Continuous-Wave Intensity Modulation approach, four types of errors could occur [34].

- **A Systematic Distance Error** is a wiggling error pattern in order of ± 5 cm. This error is usually handled in the device driver.
- **A Depth Inhomogeneity Error** is an error at some pixels at object boundaries. Mixing of light reflected from different depths could lead to wrong depth result or “flying pixels”.
- **A Multi-path Effect Error** occurs because the active light can also take an indirect path and mixed with the correct signal that travels the direct path. This effect could contaminate the distance result in the victim pixels.
- **A Dynamic Error** or *motion artifacts* are caused by the change of observed depth and/or reflectivity during the acquisition. This can lead to erroneous depth result at object boundaries.

2.3 Quantitative Assessment Based on Kinematic Measures

Regardless of the sensors/systems used in the motion capture, the kinematic measures can be interpreted in many different ways to explain the quality of the movement. This section extends a review done by Reyes-Guzmán *et al.* [35] with some additional updates. Some metrics which require an active robotic device such as active movement index are excluded from this review as they are not affordable for a home-based setup.

2.3.1 Functional Range of Motion

Joint Range of Motion

A joint range of motion (ROM) is an angle between the minimum and the maximum range of a joint during a whole movement. It is used for various reaching tasks [36] and a drinking task [37]. In the latter case, the whole movement is segmented into five predefined stages for more detailed ROM analysis.

As ROM of the main joint such as shoulder, elbow, hip, and knee can be calculated easily from the Kinect skeleton, the ROM-related features are widely available in existing commercial home rehab platforms. Kinetisense [38] measures one joint at a time while the user tries to maximize the range of that joint. Jintronix [39] uses the ROM measurement from the user to control games. All the ROM exercises, ROM assessments, and ROM-related games for lower limbs are also seen in VirtualRehab [40].

Reachable Workspace

Reachable workspace is defined as Cartesian coordinate workspace of hand position in reaching movements [41]. An indirect measure of this metric appears in some games in the SeeMe platform [42]. In this game, the user has to virtually wipe the dirty area of a window to clean it.

2.3.2 Movement Speed

Time Required to Perform a Functional Task

Time measurement commonly appears in many classic clinical assessments such as Box and Block Test, Nine Hole Peg Test, etc. This measure is used mostly in reaching movements [43].

Joint Velocity

The maximum and the mean hand velocity is normally used when both the hand current and the target position are known [41], [44]. These measures are already built into some products such as Jintronix [39] which reports the average speed of each joint in reaching games.

2.3.3 Movement Efficiency & Accuracy

Path Variability

Theoretically, a trajectory of a healthy movement from the starting point to the ending location should be a straight line [44]. Therefore, the deviation from the direct path is frequently used to assess reaching movements [36], [44]. Jintronix [39] measures how far the movement is from the ideal path in a virtual reaching game distributed in four groups (0-5 cm, 5-10 cm, 10-15 cm, and 15-20 cm).

End-point Error

At the end of a movement, this metric measures the distance between the index finger and the target location [45]. Without a proper stopping or good proprioception, the hand might not reach the target or overshoot beyond the target.

2.3.4 Movement Smoothness

Mean to Maximum Velocity Ratio

Mean to Maximum Velocity Ratio in a movement of a healthy subject should be close to one. This metric could detect the excessive amount of acceleration and deceleration which might be a movement disorder [46].

Number of Velocity Peaks / Movement Units

The velocity profile of a healthy hand movement in a reaching task normally has only one peak [47]. More peaks mean the existence of multiple acceleration and deceleration. Motor recovery leads to less number of peaks due to the better smoothness of the movement [48].

Mean Arrest Period Ratio

Movement disorders usually cause several stops during a movement toward an objective [48]. Therefore, the time ratio of the stopping period against the total movement time is also considered [49].

Jerk

This metric is defined as the rate of change in the acceleration profile [48]. However, multiple ways to calculate jerk-related measures are proposed [50] including zero-

crossing in acceleration profile in the context of reaching task [51] and the activity of daily living (ADL) [52].

Spectral Arc-Length

By measuring the amplitude of the high-frequency components from the velocity profile using Fourier transform [53], smoothness of the movement can be described.

2.3.5 Inter-Joint Coordination

For reaching movements, it was found that the shoulder and elbow flexion-extension movements are aligned almost linearly [54]. Similarly, in the transport phase of a drinking task (hand-to-mouth), the correlation between elbow flexion-extension and shoulder abduction-adduction is used as metric [52]. Among this work, higher correlation is interpreted as better joint coordination.

2.3.6 Control Strategy

In a reaching movement, the timing of the peak in the velocity profile can tell some characteristics of the reach [55]. Shifting of the peak to the left means longer deceleration periods. On the opposite, shifting to the right could mean a more abrupt deceleration period.

2.3.7 Torque Production

The measure of elbow maximum angular velocity (MEAV) is used to infer the level of torque generated in a motor task [47], [54].

2.3.8 Compensatory Movement

Some form of compensation measurements such as trunk displacement is found to be correlated with Fugl-Meyer assessment score [56]. In a game controlled by shoulder abduction/adduction, the player's torso movement can be detected and provide feedback to discourage the player from doing the compensation [57], [58]. The trunk compensation could be measured indirectly from a face tilt during a rehab gameplay [59].

Figure 2.10: A screen capture from Kinetisense during a range of motion assessment.

2.4 Existing Commercial Home Rehab Platform

Because of the affordability and the integrated full-body skeleton tracking, Microsoft Kinect V2 stands at the most suitable point in home rehab application compared to other motion capture systems. Although Kinect alone seems to be not enough for clinical assessment, a few home rehabilitation platforms have already been commercialized.

2.4.1 Kinetisense

Kinetisense (Figure 2.10) [38] is a Windows application used for body pose and range of motion measurement using Kinect V2. This tool allows a patient to record their medical report and submit to a practitioner to evaluate treatment progress.

Unlike our system which focuses on continuous and repetitive rehabilitation exercise, Kinetisense focuses on the recording of one repetition of any movement with only a few seconds in length for assessment purpose. The recorded data include the range of motion of all key joints and some balancing info. These data have to be analyzed manually as if Kinetisense is just a replacement for goniometers and inclinometers.

2.4.2 Jintronix Rehabilitation System

Jintronix Rehabilitation System [39] composes of two components, JRS Portal and JRS Wave. JRS Wave (Figure 2.11) will be set up at the patient's home as a client application in the form of virtual games designed by physical and occupational therapists. Their motion will be recorded by Microsoft Kinect V2 and translated into

Figure 2.11: Rehabilitation exercise using JRS Wave

performance information such as speed, precision, and range of motion. JRS Wave will transmit them to an online server which is accessible via JRS Portal. JRS Portal is a web-based application that allows both patient and clinicians to access the information from anywhere.

2.4.3 VirtualRehab

Similar to Jintronix, VirtualRehab [40] composes of three modules which are 1) ROM measurements, 2) ROM exercises, and 3) a suite of therapeutic game-like exercises. For the ROM exercise module, the system has a capability to measure the repetitions and the quality of movement.

The game-based exercises on both Jintronix and VirtualRehab for upper limb are limited to two main categories. The first one is the group of reaching tasks which focuses at shifting hand position to a virtual target for the score but other details of the movement are ignored. The second category is the games that are controlled by one specific range-of-motion from one joint. For both groups, the performance assessments during the exercise are commonly known and can be pre-programmed easily. For example, in the reaching task, the hand should reach the target in a straight line to get a good score. In ROM task, shoulder extension of 180 degrees can be assessed as normal. A missing thing here is the support for more complex movements that require multi-joint coordination to accomplish other than those reaching tasks in the games. A complex movement could be designed from actual functional movements or activity of daily living (ADL) to let them re-learn the capability that is more useful to them.

Figure 2.12: VirtualRehab screen shots. **Left:** ROM measurement module. **Middle:** ROM exercise module. **Right:** Game module.

Figure 2.13: **Left:** Vera system at patient's home. **Right:** A screen shot during an exercise.

2.4.4 Vera System

Similar to Jintronix but without gaming, Vera system (Figure 2.13) [60] is converting traditional rehabilitation into the digital form. This tool allows a physical therapist to customize a specific exercise routine to a patient's home and review progress remotely. Most importantly, the avatar on the patient's screen can coach the patient by a demonstration, give real-time feedback on whether or not the patient is properly doing the exercise, or even help the patient count. However, the concept of being *proper* must be pre-defined manually for each exercise [61]. In addition, with the 3D record from Kinect V2, the therapist is able to monitor the exercise in details and gives suggestions from anywhere. Unlike other systems, it only focuses on post-surgical rehabilitation of lower extremity which is not in the scope of this study.

2.5 Research Gaps

2.5.1 Hardware/Sensing System

Inconclusive Practicality of Kinect V2

In term of joint tracking accuracy of Kinect V2, even during static poses with no occlusion, the upper-limb joint center (*i.e.*, shoulder, elbow, and wrist) tracking error is 64.4 mm on average with a wide range of error from 26 mm to 218 mm [33].

According to the literature review in Section 2.2.2, it is not conclusive whether the joint tracking on Kinect V2 is accurate enough for physical rehabilitation applications. However, all current commercial home rehab platforms have already used Kinect V2 as their main sensor because of its affordability and the ease of setting up at home. With this gap of inconclusive capability of Kinect V2, we also want to evaluate its capability for our anomaly detection application with and without the additional improvement to the sensing hardware and method.

A Need of a More Stable Motion Capture System for Home Setup

From our own observation of the upper limb skeleton result from the Kinect Software Development Kit (SDK) 2.0, the joint tracking is not stable enough for us to use in our application. Specifically, Kinect cannot reliably track the forearm pronation/supination which is one crucial part of upper-limb movement assessment [62]. These facts force us to develop our own tracking method with two additional wrist-worn inertial measurement units which can give the information of forearm orientation regardless of the occlusion and enhances the overall accuracy of the upper-limb skeleton tracking.

2.5.2 High-Level Movement Analysis

On top of the motion capture system, the extracted kinematic features can be used to model a movement to assist a therapist in rehabilitation exercise monitoring. The following characteristics are not yet seen in existing home rehab systems and will be explored by this study.

Model any Repetitive Exercise by Learning from Demonstration

From the literature, there are some measures that are robust enough to be commercialized. Some of those are the range of motion (ROM), speed, and path variability. Some systems additionally implement the movement correction suggestion in repetitive movement. Such capability is enabled by pre-programming all the features to detect things that could go wrong in an exercise [61]. This requires expert knowledge and a software developer to encode that expertise into every single exercise prescription. For some complex movements, it might be very hard to create a specific set of rules to deal with variability in the movement. With this difficulty, a new exercise prescription could not be added to the system easily by the therapist alone.

This could be fixed if the model of an exercise can be learned from examples of the healthy movements without any extra labeling work from the expert. This learning method allows us to overcome the difficulty of modeling complex movement and go beyond common range-of-motion tasks or reaching task and model movement with more complex joint coordination. The corpus of exercise prescriptions will not be bounded to only simple pre-programmed movements anymore because a therapist can design and adds the new exercise to the system through movements demonstrated by healthy subjects. This learn-able characteristic can benefit the system to grow faster than encoding expert knowledge into each individual exercise.

Anomaly Detection Assistance

Unlike Kinetisense that collect only a few seconds of ROM and transfer them to a practitioner for manual assessment, the length of the exercise record in our case is much longer. Looking through the 30-minute plots of all kinematic features will overwhelm a therapist as he might not know what kind of waveform to expect for each particular kinematic feature. Some systems allow the therapist to replay the video record on demand to inspect and generate more insightful feedback. However, this might not be so useful as a therapist will never have enough time to look through the whole record to find out what is wrong with the exercise. As a result, the desired insightful feedback might not be generated despite the large pile of recorded data from tele-rehabilitation systems.

The research gap here is how to transform the existing motion capture data into a more compacted form so that the therapist can look at it and know right away that which sections he should pay attention to and watch the video record of those few sections to generate insightful feedback.

There are two main quantities that are very compact and can be useful in this scenario. The first quantity is the instantaneous phase of the repetitive movement. The waveform of phase in time series can tell what is going on in the session such as the pace of the exercise, or how much the patient needs to struggle to complete each repetition. The second quantity is how far the movement is from the healthy norm or *anomaly score*. This can directly help a therapist to spot an anomaly at glance.

Chapter 3

Hardware Design & Integration

3.1 System Overview

The sensing system can be described as follows. The user is wearing an IMU on each of his wrists. He will be in front of a Microsoft Kinect V2 sensor while he is exercising. All the visual information from Kinect and all raw data from IMUs (*i.e.*, accelerometer, gyroscope, and magnetometer) will be transmitted to a computer for data processing. Therefore, the wearable devices need to have a fast-enough data transmission rate and a good data collision avoidance policy. Moreover, all the sensors should be able to synchronize wirelessly since the latency between data generating time at a sensor and data acquisition time at the main computer is not constant. This configuration is shown in Figure 3.1.

The system requirements can be summarized in Table 3.1. The main goal of this chapter is to retrieve the synchronized orientation of the two wrist-worn IMUs in the Kinect reference frame. This will be an important piece of information for human pose estimation in Chapter 4.

Table 3.1: Requirement specifications of the IMU design.

Requirements	Specifications
Continuous operational duration	Maximum of one hour
Operational area	Within 3 m from Kinect
Wireless data transmission rate	20 + 20 + 1 Kbps (Simultaneously from three devices to the access point)
Time synchronization	Kinect clock and three wireless devices

Figure 3.1: Every component operates in an area around the access point. The access point needs to receive raw data from three wireless sensor nodes simultaneously. All the signal processing and sensor fusion will be done on the computer. Dashed lines represent wireless communications.

3.1.1 Hardware

With limitations of off-the-shelf IMU hardware such as the inability to program the firmware to allow synchronization with Kinect, a new piece of hardware is designed from the ground up as shown in Figure 3.2.

3.1.2 System Configurations

Microsoft Kinect V2 will be placed around 1.5-2.5 m in front of the user. An IMU will be tied to each wrist. One IMU must be mounted rigidly on the Kinect to reduce the setup time (Figure 3.3).

Since the orientation calibration between the Kinect reference frame and the magnetic reference frame required a lot of steps and special calibration tools, we try to simplify the setup process. With an additional IMU mounting above the Kinect using a 3D-printed case designed specifically to fit Kinect V2 and the IMU. With pre-known orientation offset of the 3D-printed case, we can calculate orientations of other two IMUs in the Kinect reference frame regardless of how user positions the Kinect (as described in Section 3.9). Importantly, the mounting point must avoid the strong magnetic disturbance generated by Kinect.

During the exercise, the base IMU and the wrist-mounted IMUs will stream raw

Figure 3.2: Our hardware prototype

data to an access point connecting to the main computer. This access point is a CC2530 wireless module (Texas Instruments) with a customized firmware developed by us.

3.1.3 Software Implementation

We tested Kinect SDK 2.0 on C# (Windows Presentation Foundation (WPF) Application) and visualized 3D information (point cloud, body segments) in real-time using OpenTK library (C# wrapper of OpenGL and OpenCL). With this implemented visualization platform, we could observe unexpected behaviors of Kinect V2 and IMUs. Moreover, debugging is more convenient with the visualization.

Our platform needs four pieces of software to collaborate to achieve the objective. The first part is the firmware for LPC54102 microcontroller on the wearable device. This firmware is developed by using *LPCXpresso* IDE. The second and the third part are two pieces of firmware for each CC2530 wireless module on the wearable device and the access point respectively. Those pieces of firmware are developed by using *IAR Embedded Workbench*. The fourth part is the application for data collection on the main computer which is developed in C# by using *Visual Studio 2017*.

3.2 Sensor Node Prototype Design

Our latest IMU hardware specifications are summarized here.

Figure 3.3: Kinect V2 with one IMU mounting on top using a 3D-printed case (base IMU). The IMU must be placed with some distance from the Kinect to avoid magnetic disturbance to the magnetometer. A buzzer is extended from the circuit of the IMU and attached directly in front of the Kinect microphone for the purpose of synchronization.

Figure 3.4: Key components of the designed IMU and their logical connections.

- **Main microcontroller:** Arm Cortex M4 with floating-point calculation unit (LPC54102, NXP). Up to 100 MHz clock frequency. 104 KBytes SRAM (Static Random Access Memory). 512 KBytes flash for program code.
- **Non-volatile storage:** 32 KBytes FRAM (Ferroelectric Random Access Memory).
- **Sensors:** One three-axis accelerometer (LIS3DSH, STMicroelectronics), one three-axis gyroscope (L3G4200D, STMicroelectronics), one three-axis magnetometer (LSM303D, STMicroelectronics), one barometer (LPS25HB, STMicroelectronics), one capacitive plate to tell the state of being worn.
- **Wireless communication:** 2.4 GHz wireless module running on a dedicated 8051-based microprocessor (CC2530, Texas Instruments) with 8 KBytes SRAM. It supports ZigBee and SimpliciTI Protocol.
- **User interface:** Piezo buzzer. General-purpose red and green LED. White LED for charging status report. One push button.
- **Battery:** 450mAh Lithium Polymer battery with micro-USB charging port.
- **Size:** $31 \times 37 \times 10$ mm (without case)

Figure 3.4 shows the connection between key components which are elaborated in the following sections. Its schematic design and PCB layout are included in Appendix A.

3.2.1 Power Supply System

The hardware is designed to run on a rechargeable Lithium Polymer battery. For ease of use, the automatic power source transition between the battery and the USB charging cable is implemented as shown in Figure 3.5. Therefore, the device will automatically switch to the USB power source while the battery is being charged and switch back smoothly to the battery when the charging cable is unplugged.

Battery over-discharging is an issue which causes the battery to generate excessive gas internally, swell, and deform. To prevent the issue, the Undervoltage-Lockout (UVLO) is applied at the regulator to stop drawing current when the battery voltage goes below 3.0 V.

Figure 3.5: Key modules in the power supply system. The power input to the regulator is chosen automatically between the USB port and the battery. The switching depends on the existence of the USB power using P-MOSFET. The input voltage to the regulator is used by the main MCU (gray line) to monitor the charging status and the battery level. An accurate schematic can be seen in Figure A.3.

3.2.2 Clock Fusion

The clock drifting among wireless devices is a critical problem that prevents the exercise session from lasting long. Using timestamps from the internal RC oscillator of the main MCU causes a drift up to 10 ms within a minute. To mitigate this problem, the MAC timer from the CC2530 wireless module is chosen as a main timestamp clock source because its specification guarantees the drift range of within 40 ppm for its accurate wireless transmission. A new timestamp from the CC2530 wireless module is sent to the main MCU regularly (every 50 ms) to fix the drift from the internal RC oscillator.

3.2.3 Form Factor & Case Design

All the thick components are located only on the front side to minimize the thickness of the PCB to 5 mm. With the battery, the size becomes $31 \times 37 \times 10$ mm which is comparable to a general smartwatch in the market (e.g., The dimensions of an Apple watch are $42 \times 36 \times 10$ mm). Even though a sleek and strong case design is not a focus of this project, a 3D-printed case is designed to assemble all the parts together to prove the concept. The drawing and the printed prototype are shown in Figure 3.6 and 3.7. This case's dimensions are $41 \times 41 \times 15$ mm.

To miniaturize them furthermore, it is possible to change the LPC54102 MCU package from LQFP-64 to WLSCP-49 to gain about 135 mm^2 of free space on the PCB. More PCB layers can be added to remove copper wires on the outer surface and free more rooms for components to get a tighter arrangement. Finally, molded plastic could reduce the thickness of the container.

Figure 3.6: The case composes of four components which are 1) a container 2) a sliding lid 3) an external button and 4) an aluminum plate holder. The holder has a curved surface at the bottom to push the aluminum plate 2 mm beyond the container external surface to touch the user's skin. A twist-to-lock mechanism is used to hold it into place and keep enough gap from the battery surface. The aluminum plate acts as a capacitive sensor to check if it is being worn.

3.3 Wireless Communication

3.3.1 SimpliciTI Protocol

The SimpliciTI protocol stack is based on a proprietary Texas Instruments specification. The protocol targets at small low-power radio-frequency networks. As the protocol does not provide much bandwidth, a careful design in the application layer is needed to maximize the data rate used in the raw data streaming.

Our goal here is to transmit raw sensor data (25 bytes per sample) at the rate of

Figure 3.7: The sensor node in the 3D-printed case.

100 Hz from two wearable IMUs and the same data from the base IMU at 5 Hz. Since the base IMU does not have any movements, it does not need to update frequently.

Dataflow Management

SimpliciTI allows the maximum payload of 111 bytes per transmission. In order to reduce overhead in the transmission, a maximum of four consecutive samples will be sent together in one packet. Each sample will be augmented by a one-byte frame counter for the access point to be aware if a frame is skipped.

Polling & Time Sharing

From our experiments, unorganized transmission from three end-devices to an access point can decrease its data rate up to 20% due to the collision on the same media channel. One way to avoid collision is that the access point will poll three end-devices in a loop. This method can ensure collision-free transmissions. The small polling packet from the access point contains the latest frame number received from the destination end-device as an acknowledgment. The destination end-device will always respond with an unreliable transmission to maintain the data rate since the transmission will be confirmed as receiving or missing in the next polling. After a polling event, if the access point does not receive a reply within 4 ms, it will ignore and poll the next end-device. This strategy provides a reliable transmission with no collision.

Pipelining

A pipelining technique is used to reduce waiting time after a poll. Because we know that the end device will always take some amount of time to prepare data and reply, the access point does not need to suddenly wait for the reply after the poll. Instead, the access point can spend this time to manage previously received data and transmit them to the main computer via UART. After that, the access point will wait until it receives a packet or 4 ms waiting time is out. This technique can improve around 5% of the total data rate in our experiment. This concept is visualized in Figure 3.8b.

With the proposed strategy, the access point can acquire the data from three IMUs simultaneously with the rate of 100, 100, and 5 Hz at 25 bytes per sample. This total rate (41 Kbps) utilizes around 90% of the maximum data rate for a single hop transmission (46 Kbps).

Figure 3.8: Comparison between non-pipelined and pipelined implementations. Each block represents an action of the access point. ED#1, ED#2, and ED#3 represent each destination of the polling.

3.4 Time Synchronization

There are two main steps in time synchronization. First, we need to know the time offset between the base IMU clock and the Kinect clock using a sound-based synchronization. Second, we need to know the time offset between the base IMU clock and the other IMU clocks in the system via a broadcast-based synchronization.

3.4.1 Sound-based Synchronization (Base IMU and Kinect)

Since the data from Kinect and IMU use different timing systems, synchronization is needed to find the time offset between the two clocks. We extend the buzzer of the base IMU and attached it right in front of the Kinect microphone array (Figure 3.3). Using this setting, we can trigger two events at the same time. The first event happens when the base IMU commands the buzzer to stop the sound. The second event is when Kinect stops hearing the sound. These two timestamps on the different systems are used to find the time offset between the two clocks.

Because the clocks on two different types of hardware might progress with slightly different speed, this offset can gradually change over time in an approximately linear manner. Therefore, the buzzer stopping event is repeated five times. Five pairs of timestamps from two clocks are fitted to a linear function with linear regression. This function is used to convert between the two sources of timestamps.

3.4.2 Broadcast-based Synchronization (Among three IMUs)

The target of this synchronization is to know the difference of timing among multiple devices, *i.e.*, three IMUs. For the SimpliciTI protocol, broadcasting capability is

built-in and can be used for synchronization.

Our method works on an assumption that the devices that receive the same broadcasted packet always receive it at the same time. The procedure can be described as follows.

1. The access point broadcasts a single packet. Then, each IMU tries to scan for the packet. If it can receive the packet, it will report the receiving timestamp to the PC.
2. Because the broadcasting packet is unreliable, PC has to broadcast many times until all three IMUs report their timestamp. With the information, we can calculate the time difference between each pair of IMUs.

3.5 Sensor Calibration

3.5.1 Magnetometer Calibration

We use a three-axis magnetometer to measure the direction of earth's magnetic field. However, the magnetic field intensity measured by a magnetometer always contains two kinds of interference, hard-iron, and soft-iron interference. The hard-iron interference occurs from nearby magnetic sources such as a permanent magnet and a speaker. It causes a constant offset from the earth's magnetic field reading. The soft-iron interference occurs from nearby ferromagnetic materials such as metal bars, screws, and battery contacts. It causes symmetrical distortion in the measurement.

Without such interference, thousands of magnetometer samples from a randomly rotating IMU should form a point cloud of a perfect sphere. In reality, interference from the device itself will distort that sphere into a shifted-tilted ellipse. Fortunately, this distortion is static as long as those interference sources have no change in relative position and rotation to the magnetometer. This could be fixed by the following calibration.

We randomly rotate the IMU and collect about 3,000 samples from an uncalibrated magnetometer. We visualize those samples in 3D to make sure that they distribute evenly on an ellipsoid. Then, a linear least square method [63] is used to fit an ellipsoid to the collected points to get nine coefficients in the general equation of an ellipsoid (Eq. 3.1).

$$C_1x^2 + C_2y^2 + C_3z^2 + 2C_4xy + 2C_5xz + 2C_6yz + 2C_7x + 2C_8y + 2C_9z = 1 \quad (3.1)$$

These nine coefficients (C_1 to C_9) are used to find the ellipsoid center, the three radii, and the directions of three ellipsoid axes [63]. Then, these properties of the ellipsoid are used to create a linear transformation that transforms this ellipsoid into a unit sphere. This transformation (Eq.3.2) will be applied to all the raw magnetometer samples to correct hard-iron and soft-iron interference.

$$\begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = E \times \begin{bmatrix} 1/a & 0 & 0 \\ 0 & 1/b & 0 \\ 0 & 0 & 1/c \end{bmatrix} \times E^T \times \left(\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \right) \quad (3.2)$$

x_e , y_e , and z_e are the raw data. First, they are subtracted by the ellipsoid center (c_x , c_y , and c_z) to move its center to the origin. E is a three-by-three rotation matrix that described the direction of three ellipsoid's radii. Parameter a , b , and c are radii of three ellipsoid's axes according to the rotation matrix. This transformation scales the ellipsoid along three axes to have the same unit radius and provides a unit sphere as a result. On the left, x_s , y_s , and z_s are the calibrated result.

Although this problem can be fixed, there are a few more external factors that should be concerned. First, the soft-iron interference from the building structure will limit the usage to a local area where the magnetic field is fairly constant. Second, the device should be at least one foot away from other electronic devices or other ferromagnetic materials since they cannot be fixed in the calibration process.

3.5.2 Accelerometer Calibration

To get a near-perfect accelerometer calibration, the sensor needs to measure the same gravitational acceleration from more than 30 different stable orientations. This could be done by a rotational stage or a robot arm. Due to the impracticality of the process for a normal user, the factory calibration setting that comes with the device is directly used.

3.5.3 Gyroscope Bias Compensation

Gyroscope bias is one of the most sensitive parameters in IMU as it changes largely according to temperature. Our system has to keep tracking the bias during the operation. We simply compare two consecutive samples from all three axes of the gyroscope. If all three pairs have value difference within 3 deg/s (*i.e.*, range of its static random noise), the value will be collected in the current bias average since the IMU is considered stable. Average bias will change slowly and subtraction of this

bias will work similarly to a low-pass filter.

3.5.4 Kinect Depth Camera Projection Model

Each individual Kinect for Xbox One (Kinect V2) has a specific set of six intrinsic parameters to model its image projection kept in the hardware. Those factory-calibrated parameters are

- FocalLengthX (f_x), FocalLengthY (f_y)
- PrincipalPointX (o_x), PrincipalPointY (o_y)
- RadialDistortionSecondOrder (d_2), RadialDistortionFourthOrder (d_4), RadialDistortionSixthOrder (d_6)

They can be accessed via Kinect SDK and used to project a 3D point in the Kinect reference frame (k_x, k_y, k_z) to a 2D pixel position (p_x, p_y) using the following equations.

$$t_x = \frac{k_x}{k_z} \quad t_y = \frac{-k_y}{k_z} \quad (3.3)$$

$$r = \sqrt{t_x^2 + t_y^2} \quad (3.4)$$

$$p_x = o_x + t_x f_x (1 + d_2 r^2 + d_4 r^4 + d_6 r^6) \quad (3.5)$$

$$p_y = o_y + t_y f_y (1 + d_2 r^2 + d_4 r^4 + d_6 r^6) \quad (3.6)$$

These equations can be used to synthesize depth images with radial distortion from any synthesized 3D model. On the other hand, the back-projection from a depth pixel to a 3D point needs a specific table lookup provided by the Kinect SDK. This lookup table is uniquely generated on each Kinect device.

3.6 Reference Frames

There are three main reference frames in the rehabilitation exercise configuration.

World Reference Frame $\{W\}$

This frame is defined by the direction of gravity and the local magnetic field. The X, Y, and Z-axis are mapped to the magnetic north, the magnetic west, and the upward direction. The quaternion-based Kalman filter in Section 3.8 gives each IMU's orientation in this reference frame.

Kinect Reference Frame $\{K\}$

This frame is defined by the depth camera and the infrared camera of the Kinect which shared the same extrinsic parameters, *i.e.*, position and orientation. Its X, Y, and Z-axis are mapped to the left, the top, and the front direction of the device respectively. All the visual sensing information from Kinect SDK is in the Kinect reference frame except RGB image which needs SDK's *CoordinateMapper* library to map their position to the Kinect reference frame.

Vicon Reference Frame $\{E\}$

Vicon is an 8-camera marker-based motion capture system using as a gold standard in our evaluation. The frame is defined by the Vicon calibration process.

Other Reference Frames

For other reference frame notations, $\{B\}$ is the base-IMU frame, $\{A\}$ is the arm-IMU frame, and $\{P\}$ is the calibration panel frame. To simplify the explanation, all arm-related notations in this thesis are for the right arm.

3.7 Quaternion Notation

Quaternion is chosen as the main representation of orientation between reference frames. The notation I_Jq means a unit quaternion that can convert a 3D vector (V) in a reference frame $\{I\}$ to a reference frame $\{J\}$ by

$${}_JV = {}^I_Jq \times {}_IV \times {}^I_Jq^{-1} \quad \text{or} \quad {}_JV = \text{rot}({}^I_Jq, {}_IV)$$

The meaning of operation $\times$ depends on its operands. It could be a quaternion multiplication, a matrix multiplication, or a cross product.

3.8 Quaternion-based Kalman Filter

All nine axes (*i.e.*, accelerometer, gyroscope, and magnetometer) are used to estimate the three-DOF orientation of an IMU. In this work, we use a quaternion-based Kalman orientation filter proposed by Comotti *et al.* [64] in order to find B_Wq and A_Wq . A data flow diagram of this process is shown in Figure 3.9.

Figure 3.9: Data flow diagram of quaternion-based Kalman filter.

The key idea of this filter is to combine two redundant (but noisy) orientations. The first orientation (observed orientation) can be calculated from the acceleration vector and the calibrated ambient magnetic field vector. The second orientation (predicted orientation) is a projection of the previous estimated orientation using angular velocity from the gyroscope.

However, their vector observation step is modified by replacing the Gauss-Newton optimization method with an analytical solution. Our vector observation step assumes that the observed acceleration points in the same direction as Z-axis of the world reference frame, *i.e.*, gravity direction. This assumption leads to a finding of all three world coordinate axes in the IMU reference frame by using only the measured acceleration direction ($\hat{g}$) and the measured magnetic direction ($\hat{m}$) in the unit vector form.

$$V'_Z = \hat{g} \quad (3.7)$$

$$V'_Y = \hat{g} \times \hat{m} \quad (3.8)$$

$$V'_X = (\hat{g} \times \hat{m}) \times \hat{g} \quad (3.9)$$

The three orthogonal axes (V'_X , V'_Y , and V'_Z) will be normalized and fed into van Waveren's method [65] to convert them to an observed quaternion (${}^{IMU}_W q$). However, its negative version $-({}^{IMU}_W q)$ also represents the same orientation. In order to get a smooth filter result, the solution that is closer to the current predicted orientation

Figure 3.10: This calibration panel is designed to be seen by both the Kinect and the Vicon system at the same time.

must be chosen as ${}^{IMU}_W q_{observed}$. That is

$${}^{IMU}_W q_{observed} = \begin{cases} {}^{IMU}_W q & \text{if } {}^{IMU}_W q_{predicted} \cdot {}^{IMU}_W q > 0 \\ - {}^{IMU}_W q & \text{otherwise.} \end{cases} \quad (3.10)$$

3.9 Transformation Between Reference Frames

A method to find a transformation between two different sensing systems is to let two systems observe the same object. Then, the two different results will lead to a transformation.

3.9.1 Vicon-to-Kinect Transformation

A calibration panel with four reflective markers in L-shape (Fig.3.10) is exposed to both the Vicon and the Kinect system. 3D marker positions from Vicon are used to create the direction of X, Y, and Z-axis of the panel and its origin (${}_E P_o$). The three axes are turned to an orientation quaternion (${}_E^P q$) using van Waveren's method [65].

Unlike Vicon, Kinect cannot sense the depth data on highly reflective surfaces and needs some workarounds to get the panel's orientation and origin. First, four bright points on Kinect's infrared image are detected and localized in 2D. Using those four 2D positions, three non-reflective areas on the panel are estimated as

shown in Figure 3.10. Then, three groups of 3D points are collected from three areas on the panel, (1) X-axis group, (2) Y-axis group (3) panel plane group. Principal component analysis (PCA) is used in each group. The first principal component in group one and two will become the panel's X-axis and the panel's Y-axis respectively. The third principal component of group three will become the panel's Z-axis. The three axes are converted to an orientation quaternion (P_Kq) using van Waveren's method [65]. The virtual intersection point between X and Y-axis will become the panel's origin in the Kinect reference frame (${}_K P_o$).

The orientation between both frames (E_Kq) and the Vicon origin position in the Kinect reference frame (${}_K E_o$) can be found by

$${}^E_Kq = {}^P_Kq \times {}^P_Eq^{-1} \quad (3.11)$$

$${}_K E_o = {}_K P_o - \text{rot}({}^E_Kq, {}_E P_o) \quad (3.12)$$

Now, any position in the Vicon reference frame (${}_E V$) can be transformed to a position in the Kinect reference frame (${}_K V$) by

$${}_K V = {}_K E_o + \text{rot}({}^E_Kq, {}_E V) \quad (3.13)$$

3.9.2 World-to-Kinect Transformation

In this case, only an orientation of the world reference frame in the Kinect reference frame (W_Kq) is needed. Our method follows Eq.3.14.

$${}^W_Kq = {}^B_Kq \times {}^B_Wq^{-1} \quad (3.14)$$

B_Wq is an output from the base-IMU orientation filter which is always available. B_Kq is a constant orientation quaternion of base-IMU in the Kinect reference frame which is needed to be found in a calibration process. This value is constant because the base-IMU is rigidly attached to the Kinect.

An IMU is attached on the calibration panel by aligning the IMU axes parallel to the panel axes. The method explained in Section 3.9.1 is used to get P_Kq from the Kinect while the orientation filter running on the IMU gives the panel orientation in the world reference frame (P_Wq). B_Kq can be calculated from

$${}^B_Kq = {}^P_Kq \times {}^P_Wq^{-1} \times {}^B_Wq \quad (3.15)$$

Figure 3.11: The Kinect reference frame $\{K\}$ and base-IMU reference frame $\{B\}$ have a fixed relative orientation.

This calibration process allows the users to move the Kinect sensor around without the need of re-calibration because B_Kq does not change as shown in Figure 3.11.

3.10 IMU Orientation in Kinect Reference Frame

After all the calibration is completed, B_Kq will be ready. The quaternion-based Kalman filter will give out B_Wq from the base-IMU, and A_Wq from the wrist-worn IMU. The wrist-worn IMU orientation in the Kinect reference frame can be calculated by

$${}^A_Kq = {}^B_Kq \times {}^B_Wq^{-1} \times {}^A_Wq \quad (3.16)$$

This information is found to be useful in solving the unreliable forearm orientation in Kinect SDK [66]. Therefore, it is used in Chapter 4 to guide the pose estimation in occlusion and add a reliable tracking of forearm pronation/supination.

Chapter 4

Upper-body Pose Estimation

Human pose estimation from a single depth camera has become a popular research topic due to the emergence of low-cost depth sensors such as Kinect (Microsoft, USA) in 2010. Current state-of-the-art techniques can be divided into discriminative and generative methods which can be used together. Generative methods search in a pose space to fit a template to the observation. The model could be in various forms such as a mesh [67], a sphere set [68], or articulated Gaussian kernels [69]. Those purely generative methods always rely on the result from the previous frames. Therefore, it is hard to recover when the tracking is lost from a fast movement. This leads to the discriminative initialization to improve the robustness of the tracking. A common way to initialize the pose parameter is to turn the observed frame to a low-dimensional feature vector to allow a quick search in a discrete pose database [70]–[72] and then refine the result using a generative method.

On the other hand, there are a few per-pixel inferring strategies that classify each depth pixel into body parts [73], regress to joint positions [73], or regress to correspondences on a model surface [74]. They can achieve a good joint prediction accuracy with a decision forest learning from a massive synthetic training data. One of them [73] was implemented into Kinect software development kit (SDK) and was evaluated multiple times in different scenarios. For gait analysis, Pfister *et al.* [23] conclude that the measurement from Kinect V1 is not consistent enough for their clinical assessment. Its sensitivity needs improvement before the clinical implementation. Even with the more accurate Kinect V2, the average upper limb joint position error is in the range of 50-101 mm and they will get off-track at 1-5% of the time [75]. This can be interpreted to 1-5 frames every 3.3 seconds (at 30 fps) that a joint position is totally wrong. Those errors should be addressed in some applications that required better accuracy such as rehabilitative assessment.

This chapter aims to fill the current gap in the skeleton tracking for upper-limb rehabilitative exercise assessment tasks. Our body-tracking solution is inspired by an emerging hand-tracking technique from Microsoft [76] together with the availability of an anatomically-accurate human mesh model which can morph its shape to fit individual users [77]. Our method fits an articulated mesh model to the depth image sequence using an optimization method that updates both pose parameters and correspondence matching simultaneously [76]. As the hand and the upper body have significant differences such as scale, skeleton topology, and joint configurations, various adaptations are needed. For the optimization, two new terms are introduced to the cost function. The first term is generated from a 3D unsigned distance field which gives a penalty when the model is floating between the point cloud surface and the depth camera. The second term involves two additional wrist-worn IMUs which allows the tracking method to partially recover an occluded forearm.

Using IMUs to improve a visual-based pose estimation was first introduced in the context of multi-view silhouettes from multiple RGB cameras [78]–[80]. However, with a single depth camera, IMU orientations have only been used for a table lookup in the discriminative step [72]. We apply them differently by using IMU information directly in the cost function of the generative step. Importantly, as we target rehabilitative patients as our main user group, requiring an IMU on their head or at the back of their trunk could be difficult in a home-based rehabilitation setup. Therefore, a less obtrusive configuration of two wrist-worn IMUs together with a depth camera is proposed.

As we target at home-based rehabilitation exercise assessment, some subtle details that are usually ignored in other pose tracking methods are addressed in this work. For example, the pose estimation method should be able to detect the scapular elevation because it is a common compensatory movement that a stroke patient should be encouraged to avoid. The forearm pronation/supination should also be tracked as they can infer the state of recovery. It has never been tracked reliably with a depth-based visual motion capture system.

In addition, model personalization has become a crucial part in various pose estimation techniques. The subject’s shape could be estimated from multi-view silhouettes [81], a laser scan [82], or depth images. The most general method is to have shape fitted every frame which is computationally expensive [67]. A more common way is to fit the shape of the template model using just one or two frames of a predefined pose and use that fixed shape for the pose tracking [69], [83]. A shape personalization method is also implemented in our work with a similar concept to

Helten *et al.*'s work [83] but based on a publicly-released statistical model [77].

4.1 Pose Estimation Method

4.1.1 Frame Inputs & Expected Outputs

A Kinect V2 sensor provides depth images at 30 Hz. For each depth frame, Kinect SDK is used to find all associated 3D points (point cloud), the body silhouette, and the 3D joint positions (Kinect skeleton). The two wrist-worn IMUs are sampled at 80 Hz and one base-IMU are sampled at 5 Hz. With this configuration, the orientations of both wrist-worn IMUs in the camera reference frame can be calculated using the method described in Section 3.10.

The final goal is to search for a pose θ that best fits those observations. To evaluate the quality of fit, a triangulated surface will be rendered from an articulated human model called Skinned Multi-Person Linear model (SMPL) [77]. This model is driven by a pose (θ) and a personalized shape (β). Therefore, the problem can be divided into two phases. The first step is the model personalization which searches for both pose θ and shape β parameters from only one depth frame per subject (Section 4.2). The personalized shape β from this process will then be used as a static value in the second step where we can only search for the pose θ to speed up the process (Section 4.1.6).

4.1.2 Human Mesh Model

Unlike other statistical human models such as SCAPE [84] or Delta [67], the SMPL model renders its skin mesh in a closed-form calculation without any iterative optimization. This characteristic allows any points on the surface mesh to be differentiable with respect to shape (β) and pose (θ) parameters.

Skinned Multi-Person Linear model (SMPL) [77] needs two input parameters to produce all joint center positions and all mesh vertex positions.

The first input parameter is the shape parameter (β) which is defined as a 10-dimensional vector. This shape vector is used to control the shape of the model. Each dimension is associated with different aspects of the shape such as the height, limb length, fatness, etc.

The second input parameter is the pose parameter (θ) which is defined as a 72-dimensional vector in the case of all-joint articulation. All 72 dimensions can be divided into 24 groups of three. Each group associates with an articulation of one

joint in a form of rotation vector (3DOF). The rotation vector direction points along the axis of rotation and its magnitude tells the angle of rotation.

The calculation of the final joint and vertex positions is driven by five static components that work together.

1. **Template mesh.** This is a triangle mesh of a human in T-pose created by an artist. This template defines 3D positions of all 6,890 vertices in this mesh.
2. **β -dependent shape modifier.** This is a linear function that converts the shape parameter β into 3D position shifting of each vertex in the template mesh. This component allows the shape of the template model to be adjustable.
3. **Joint regression.** This is also a linear function that calculates all joint locations in the template pose (T-pose) from all the vertices that have already been modified by the shape parameter β . Therefore, the template joint position is only affected by β . Each joint location is used to define a local origin and a relative position to its parent joint for each body segment for the articulation.
4. **θ -dependent shape modifier.** Because human has different soft tissue deformations depending on different poses and the artifact from linear blend skinning can be unnatural, SMPL model proposed an additional correction to the template mesh to compensate for those deformations and artifacts after the articulation. This modification is also a linear function to the rotation matrix representation of all rotation vectors in the pose parameter θ .
5. **Linear blend skinning (LBS) weights.** Each vertex on the mesh is associated with one or more segments with weights that sum to one. For example, a vertex on the mid-forearm is associated with the forearm segment alone with the weight of one. A vertex on the elbow might be associated with the forearm segment with a weight of 0.6 and the upper arm segment with a weight of 0.4. These weights help the surface to blend smoothly at the transition between two segments.

All the coefficients and weights in these linear function are optimized from over a thousand high-resolution 3D scans of human shapes [77].

With the known relative joint position of the model in T-pose, the articulation of the skeleton by θ is done through a chain of forward-kinematic tree from the root to every branch. All the vertices from the modified template will be translated and

Figure 4.1: Original high-resolution SMPL template meshes and the modified low-resolution SMPL template meshes. The male templates (left) and the female templates (right).

oriented together with its associated segments defined by LBS weights. The result of this rendering process is articulated joint positions together with an articulated mesh $\mathcal{P}(\theta) \in \mathbb{R}^{3 \times M}$ where M is the number of vertices. The detailed rendering formulation of SMPL model is available in the original article [77].

The original SMPL model contains 6,890 vertices with relatively high density on the face details. A 3D surface simplification software, *Qslim 2.1*, in *mode 2* is used to reduce the mesh resolution through a sequence of vertex pair contractions [85]. In each round, the software chooses a pair of adjacent vertices that can be merged with minimum effect to the surface and merges them together to remove one vertex, two faces, and three edges. The merged vertex will be placed on the optimal spot on the edge between the two chosen vertices. The contraction is repeated until the number of vertices reaches the set target. The software is set to prevent the contraction around the shoulders, the elbows, and the wrists to avoid artifacts from joint bending. Moreover, each hand of the model are replaced by a round orb as this study does not intend to track hands. The final model contains 1,000 vertices with one-to-one associations to vertices on the original model. The original models and the simplified models are shown in Figure 4.1.

Instead of using three degrees-of-freedom (DOF) in all the joints, we remove one DOF that represents the *roll* angle in every joint except the joints on the spine and the head. This constraint is applied to avoid the *candy wrapper artifact* when a joint twists due to the use of linear blend skinning. If needed, the *roll* information of the arm segments can be recovered later after the fitting process by using the IMU orientations. Our final model has 40 DOF for the upper-body model (from the

Figure 4.2: Joint positions (white points) and the DOF allowed in each of them. The root joint is at the pelvis.

hips and above) and 48 DOF for the full-body model including global translation and orientation. All DOF in a model are represented by $\theta \in \mathbb{R}^{N_\theta}$. The position of joints and their DOFs are shown in Figure 4.2.

4.1.3 Parameterization of a Point on the Mesh

Any point on the mesh can be represented in the piecewise 2D parameter space

$$\Omega = \{(f, u, v) | f \in F; u \in [0, 1]; v \in [0, 1]; u + v \in [0, 1]\} \quad (4.1)$$

where f is a triangular face on the mesh. (u, v) represents a barycentric coordinate (u, v, w) where $w = 1 - u - v$. The set F contains all triangular faces on the articulated mesh.

Given an articulated mesh $\mathcal{P}(\theta) \in \mathbb{R}^{3 \times M}$ rendered from the SMPL model with a specific shape vector β . The surface position function $S_p : \Omega \times \mathbb{R}^{N_\theta} \mapsto \mathbb{R}^3$ is defined

by

$$S_p((f, u, v), \theta) = u\mathbf{p}_u + v\mathbf{p}_v + w\mathbf{p}_w \quad (4.2)$$

where $\mathbf{p}_u$, $\mathbf{p}_v$, and $\mathbf{p}_w$ are 3D position of three associated corners of triangle f on the articulated mesh $\mathcal{P}(\theta)$ and $w = 1 - u - v$

For the surface normal function $S_\perp : \Omega \times \mathbb{R}^{N_\theta} \mapsto \mathbb{R}^3$, $S_\perp(\nu, \theta)$ is a 3D unit normal vector from the mesh position ν on the model articulated by θ . Having the same normal vectors on one triangular face will zero the gradient of normal penalty term relative to u and v . We avoid the issue by calculating normal vector of each vertex from area-weighted sum of surrounding face normals [86]. Then, we use them in similar way as position function. The function is defined by

$$S_\perp((f, u, v), \theta) = \Pi(u\mathbf{n}_u + v\mathbf{n}_v + w\mathbf{n}_w) \quad (4.3)$$

where $\Pi(\cdot)$ is a vector normalization. $\mathbf{n}_u$, $\mathbf{n}_v$, and $\mathbf{n}_w$ are associated normal vectors at the three corners of triangle f .

4.1.4 Energy Function

The energy function contains all kinds of designed penalty that give higher value when the simulated model does not match with the observation or when a constraint is violated. The total energy function is a weighted sum of multiple penalty terms which is defined as

$$E(\theta, \mathcal{U}) = E_{\text{data}}(\theta, \mathcal{U}) + \sum_{\tau \in \text{Terms}} \lambda_\tau E_\tau(\theta) \quad (4.4)$$

where $\text{Terms} = \{bg, bgJoint, float, limit, imu\}$

data term

For each depth image, N_s pixels will be sampled from the body silhouette area to get 3D positions $\mathbf{p}_i$ and surface normals $\mathbf{n}_i$ (Section 4.1.9). Each sampled point matches with a correspondent point ν_i on the piecewise 2D parameter space Ω . The point ν_i keeps moving to a better place in the process of the continuous update (Section 4.1.10) and the discrete update (Section 4.1.11).

Let $\mathcal{U} = \{\nu_i | \nu_i \in \Omega\}_{i=1}^{N_s}$ be a set of current correspondences. The *data* term penalizes the position and the normal differences between the sampled points from

the depth image $(\mathbf{p}_i, \mathbf{n}_i)$ and their correspondences (ν_i) on the model.

$$E_{\text{data}}(\theta, \mathcal{U}) = \frac{1}{N_s} \sum_{i=1}^{N_s} \left(\frac{\|S_p(\nu_i, \theta) - \mathbf{p}_i\|^2}{\sigma_p^2} + \frac{\|S_{\perp}(\nu_i, \theta) - \mathbf{n}_i\|^2}{\sigma_n^2} \right) \quad (4.5)$$

bg term

The *bg* term penalizes when some parts of the model are being projected outside the 2D silhouette.

$$E_{\text{bg}}(\theta) = \frac{1}{H} \sum_{h=1}^H \|\Pi_{\triangleleft}(S_p(\nu_h^{\text{bg}}, \theta)) - \mathcal{D}_{2\text{D}}(\Pi_{\triangleleft}(S_p(\nu_h^{\text{bg}}, \theta)))\|^2 \quad (4.6)$$

$\mathcal{U}^{\text{bg}} = \{\nu_h^{\text{bg}}\}_{h=1}^H \subseteq \Omega$ is a set of selected points on the model which may cause the penalty. H points in this set are re-selected every iteration from the area of the model where the normal vector points perpendicular to the Kinect's optical axis (Z-axis). Those points will roughly surround the model when projected to the 2D image and help to push the model closer to the silhouette without using excessive points to cover everywhere on the surface mesh.

$\Pi_{\triangleleft} : \mathbb{R}^3 \mapsto \mathbb{R}^2$ is a perspective projection from the 3D space to the 2D depth image space.

$\mathcal{D}_{2\text{D}} : \mathbb{R}^2 \mapsto \mathbb{R}^2$ maps a projected 2D position on the depth image to a 2D position of the closest pixel on the silhouette. If the input is in the silhouette, $\mathcal{D}_{2\text{D}}$ will act as an identical function. This function is pre-calculated on a distance transform using Quasi-Euclidean distance metric.

bgJoint term

This term works in the same way as the *bg* term which penalizes when positions from 16 selected joints ($\mathcal{J}_{\text{bg}}$) are being projected outside the silhouette area.

$$E_{\text{bgJoint}}(\theta) = \frac{1}{|\mathcal{J}_{\text{bg}}|} \sum_{j \in \mathcal{J}_{\text{bg}}} \|\Pi_{\triangleleft}(\mathcal{J}_j(\theta)) - \mathcal{D}_{2\text{D}}(\Pi_{\triangleleft}(\mathcal{J}_j(\theta)))\|^2 \quad (4.7)$$

where $\mathcal{J}_j(\theta)$ is the 3D position of joint j under pose θ .

limit term

This term aims to limit each individual joint to stay in a possible range of motion.

$$E_{\text{limit}}(\theta) = \frac{1}{N_\theta - 6} \sum_{i=1}^{N_\theta - 6} (\max(0, \psi_i^{\min} - \psi_i(\theta)) + \max(0, \psi_i(\theta) - \psi_i^{\max}))^2 \quad (4.8)$$

$\psi^{\min} \in \mathbb{R}^{N_\theta - 6}$ and $\psi^{\max} \in \mathbb{R}^{N_\theta - 6}$ are soft lower and upper bounds of $N_\theta - 6$ joint angle respectively. These boundaries are set according to our observation.

imu term

This term penalizes the difference between the current forearm pointing direction from the model and the observation from the wrist-worn IMUs.

$$E_{\text{imu}}(\theta) = \|\mathbb{C}_{\text{left}}(\theta) - \mathbb{P}_{\text{left}}\|^2 + \|\mathbb{C}_{\text{right}}(\theta) - \mathbb{P}_{\text{right}}\|^2 \quad (4.9)$$

$\mathbb{C}_{\text{left}}(\theta)$ and $\mathbb{C}_{\text{right}}(\theta)$ extract left and right forearm pointing direction as a 3D unit vector from the skeleton of pose θ

$$\mathbb{C}_{\text{left}}(\theta) = \Pi(\mathcal{J}_{\text{leftWrist}}(\theta) - \mathcal{J}_{\text{leftElbow}}(\theta)) \quad (4.10)$$

$$\mathbb{C}_{\text{right}}(\theta) = \Pi(\mathcal{J}_{\text{rightWrist}}(\theta) - \mathcal{J}_{\text{rightElbow}}(\theta)) \quad (4.11)$$

$\mathbb{P}_{\text{left}}$ and $\mathbb{P}_{\text{right}}$ are 3D unit vectors from left and right wrist-worn IMUs. As the IMU orientation in the camera reference frame is calculated, we can choose the direction of the IMU axis that points along the forearm pointing direction. For example, if the right forearm pointing direction is aligned with the Y-axis of the IMU on the wrist, we get

$$\mathbb{P}_{\text{right}} = \text{rot}({}^A_K q, \hat{j}) \quad (4.12)$$

where $\hat{j}$ is the unit vector along Y-axis.

float term

The *float* term is introduced to deal with some parts of the mesh model that is currently projecting to the 2D silhouette area (no penalty from *bg* term) but is floating between the point cloud surface and the camera. That part of the model should be

pushed to the closest spot in the volume behind the point cloud.

$$E_{\text{float}}(\theta) = \frac{1}{Q} \sum_{q=1}^Q \|S_p(\nu_q^{\text{float}}, \theta) - \mathcal{D}_{3D}(S_p(\nu_q^{\text{float}}, \theta))\|^2 \quad (4.13)$$

$\mathcal{U}^{\text{float}} = \{\nu_q^{\text{float}}\}_{q=1}^Q \subseteq \mathcal{U}^{\text{bg}}$ is a set of selected points on the model. These points are inside the 2D body silhouette when projected but are shallower than the point cloud surface. This set will get updated every Levenberg-Marquardt iteration.

$\mathcal{D}_{3D} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ maps a 3D position from the model to 3D position of the closest point in the volume behind the 3D point cloud surface. If the input is behind the point cloud surface, $\mathcal{D}_{3D}$ will act as an identical function. This function is pre-estimated by discretizing the capture volume with a resolution of 0.02 m.

4.1.5 Vector of Residuals

All energy terms can be converted to a sum-of-squares form in order to use Levenberg-Marquardt algorithm. By applying similar conversion to all energy terms and given that $\Theta = (\theta, \mathcal{U})$, the total energy term (Eq.4.4) becomes

$$E(\Theta) = \sum_{k=1}^K r_k^2(\Theta) \quad (4.14)$$

$\mathbf{r}(\Theta)$ is a *vector of residuals* which contains residual vectors from all energy terms.

$$\mathbf{r}(\Theta) = [\mathbf{r}_{\text{data}}(\Theta); \mathbf{r}_{\text{bg}}(\theta); \mathbf{r}_{\text{bgJoint}}(\theta); \mathbf{r}_{\text{float}}(\theta); \mathbf{r}_{\text{limit}}(\theta); \mathbf{r}_{\text{imu}}(\theta)]^\top \in \mathbb{R}^K \quad (4.15)$$

where $K = K_{\text{data}} + K_{\text{bg}} + K_{\text{bgJoint}} + K_{\text{float}} + K_{\text{limit}} + K_{\text{imu}}$ and $K_{\text{data}} = 6N_s$, $K_{\text{bg}} = 2H$, $K_{\text{bgJoint}} = 2|\mathcal{J}_{bg}|$, $K_{\text{float}} = 3Q$, $K_{\text{limit}} = N_\theta - 6$, $K_{\text{imu}} = 6$.

4.1.6 Levenberg-Marquardt (LM) Iteration

The goal is to find the pose parameter θ and the correspondence $\mathcal{U}$ that minimize the energy function.

$$\hat{\theta}, \hat{\mathcal{U}} = \underset{\theta, \mathcal{U}}{\operatorname{argmin}} E(\theta, \mathcal{U}) \quad (4.16)$$

In each iteration, the LM algorithm tries to find an update $\Delta\Theta = (\Delta\mathcal{U}, \Delta\theta)$ that brings the current search to the lower energy state by forming the following equation system.

$$(J(\Theta)^\top J(\Theta) + \gamma I) \Delta\Theta = J(\Theta)^\top \mathbf{r}(\Theta) \quad (4.17)$$

To solve for $\Delta\Theta$ efficiently without the assembly of a Jacobian matrix $J(\Theta)$ with a huge size of K -by- $(2N_s + N_\theta)$, the system can be re-arranged using Schur complement to

$$\begin{bmatrix} C & A \\ A^\top & B \end{bmatrix} \begin{bmatrix} \Delta\mathcal{U} \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} \quad (4.18)$$

where

$$\begin{aligned} C &= J_{\text{data},\mathcal{U}}(\Theta)^\top J_{\text{data},\mathcal{U}}(\Theta) + \gamma I \\ A &= J_{\text{data},\mathcal{U}}(\Theta)^\top J_{\text{data},\theta}(\Theta) \\ B &= J_{\text{data},\theta}(\Theta)^\top J_{\text{data},\theta}(\Theta) + \sum_{\tau \in \text{Terms}} (J_\tau(\theta)^\top J_\tau(\theta)) + \gamma I \\ c &= J_{\text{data},\mathcal{U}}(\Theta)^\top r_{\text{data}}(\Theta) \\ b &= J_{\text{data},\theta}(\Theta)^\top r_{\text{data}}(\Theta) + \sum_{\tau \in \text{Terms}} (J_\tau(\theta)^\top r_\tau(\theta)) \\ J_{\text{data}}(\Theta) &= [J_{\text{data},\mathcal{U}}(\Theta); J_{\text{data},\theta}(\Theta)] \end{aligned}$$

$J_{\text{data},\mathcal{U}}(\Theta)$ is a $6N_s$ -by- $2N_s$ Jacobian matrix of data term residual vector $r_{\text{data}}(\Theta)$ with relative to $\mathcal{U}$ (Figure 4.3).

$J_{\text{data},\theta}(\Theta)$ is a $6N_s$ -by- N_θ Jacobian matrix of data term residual vector $r_{\text{data}}(\Theta)$ with relative to θ .

$J_\tau(\theta)$ is a K_τ -by- N_θ Jacobian matrix of τ term residual vector $r_\tau(\theta)$ with relative to θ .

γI adds damping parameter to the diagonal elements of C and B .

Then, a remarkably smaller symmetric positive definite system is available to solve for $\Delta\theta$

$$(B - A^\top C^{-1} A) \Delta\theta = b - A^\top C^{-1} c \quad (4.19)$$

Then, we can calculate

$$\Delta\mathcal{U} = C^{-1}(c - A\Delta\theta) \quad (4.20)$$

As the matrix $J_{\text{data},\mathcal{U}}(\Theta)$ has a sparse structure as shown in Figure 4.3, C and C^{-1} can be calculated cheaply without a full-matrix multiplication and inversion. Each small 2-by-2 block in C can be inverted independently to get C^{-1} .

We solve for $\Delta\Theta(\gamma)$ as a function of the damping parameter γ and use it to evaluate the new point. The damping parameter γ will be increased exponentially until $E(\Theta \oplus \Delta\Theta(\gamma'))$ becomes lower than $E(\Theta)$. Then we update $\Theta \leftarrow \Theta \oplus \Delta\Theta(\gamma')$

Figure 4.3: **Left:** Jacobian matrix $J_{\text{data}, \mathcal{U}}(\Theta)$ with a sparse structure. **Middle:** A six-by-two Jacobian matrix from one correspondent pair. **Right:** Matrix C .

to finish one iteration. One iteration of LM can be shown as a flowchart in Figure 4.4. The update operator $\oplus$ for θ is a simple subtraction. However, the update operator for $\mathcal{U}$ needs a mesh walking process explained in Section 4.1.10.

4.1.7 Jacobian Calculation

One common way to calculate the exact Jacobian from a complex energy function is to use an automatic differentiation library such as DiffSharp [87]. However, when DiffSharp uses forward-mode automatic differentiation to calculate Jacobian relative to θ , it needs N_θ times of the whole model rendering process and residual evaluation which slow the whole process down. Therefore, we simplify some parts of the model rendering process to allow each individual gradient to be in a short closed-form analytical formula and reduce the number of rendering round to one.

Linear blend skinning (LBS) is a technique that allows a vertex on the mesh surface to belong to multiple bones by a static ratio for a smooth surface skinning. This blending makes the joint-to-vertex dependency table overly complex with real values between zero and one. We simplify this by making one vertex belong to only one bone with the maximum blending ratio and create a boolean dependency table $V_D(i_\theta, i_v) \in \{0, 1\}$ that tells whether a change of joint parameter i_θ will affect the vertex i_v according to the kinematic chain. On the other hand, joint-to-face dependency table $F_D(i_\theta, f) \in \{0, 1\}$ could also be done in a similar way. This simplification is to facilitate our analytical Jacobian estimation only but the residue calculation still uses LBS.

For a point $\nu = (f, u, v)$, if its 3D position or normal vector depends on a joint

Figure 4.4: Flowchart of one Levenberg-Marquardt (LM) iteration.

parameter $\theta_{i\angle}$, a change in $\theta_{i\angle}$ alone will act as rotating the 3D point around an axis with the joint position as the center of rotation. Therefore, the partial derivative of the position and normal (in x , y , and z components) with respect to $\theta_{i\angle}$ can be analyzed as

$$\frac{\partial \mathbf{p}}{\partial \theta_{i\angle}} = F_D(i_\theta, f) \cdot [\mathcal{A}_i(\theta)]_{\times} (\mathbf{p} - \mathbf{C}_i(\theta)) \quad (4.21)$$

$$\frac{\partial \mathbf{n}}{\partial \theta_{i\angle}} = F_D(i_\theta, f) \cdot [\mathcal{A}_i(\theta)]_{\times} \mathbf{C}_i(\theta) \quad (4.22)$$

$\mathbf{p} = S_p((f, u, v), \theta) = [p_x; p_y; p_z]^\top$ is a 3D position on face f which is rendered from pose θ .

$\mathbf{n} = S_\perp((f, u, v), \theta) = [n_x; n_y; n_z]^\top$ is a 3D normal on face f which is rendered from pose θ .

$\mathcal{A}_i(\theta)$ and $\mathbf{C}_i(\theta)$ are the axis of rotation and 3D joint position associated with parameter $\theta_{i\angle}$ respectively when the model is in θ pose. Operator $[\cdot]_{\times}$ generates three-by-three skew symmetric matrix from a unit vector input.

The other simplification is about the perspective projection from 3D space to

the depth image. By removing the complexity of the radial distortion correction, the partial derivative of the 2D position in the image space (in x and y component) with respect to $\theta_{i\angle}$ will be

$$\frac{\partial x_{\text{pixel}}}{\partial \theta_{i\angle}} = \frac{f_x}{p_z} \cdot \frac{\partial p_x}{\partial \theta_{i\angle}} - \frac{f_x p_x}{p_z^2} \cdot \frac{\partial p_z}{\partial \theta_{i\angle}} \quad (4.23)$$

$$\frac{\partial y_{\text{pixel}}}{\partial \theta_{i\angle}} = -\frac{f_y}{p_z} \cdot \frac{\partial p_y}{\partial \theta_{i\angle}} + \frac{f_y p_y}{p_z^2} \cdot \frac{\partial p_z}{\partial \theta_{i\angle}} \quad (4.24)$$

where $(x_{\text{pixel}}, y_{\text{pixel}})$ are a 2D projected position from a 3D point (p_x, p_y, p_z) . f_x and f_y are camera's focal length.

With these approximations of derivative, the calculation of Jacobian matrices $J_{\text{data},\theta}(\Theta)$, $J_{\text{bg}}(\Theta)$, $J_{\text{bgJoint}}(\Theta)$, and $J_{\text{float}}(\Theta)$ are about 10 times faster than the calculation by the automatic differentiation library. Jacobian matrices $J_{\text{data},\mathcal{U}}(\Theta)$, $J_{\text{imu}}(\Theta)$, and $J_{\text{limit}}(\Theta)$ are also calculated analytically instead of using an automatic differentiation library.

4.1.8 Pose Initialization

Pose parameters are initialized from two sources. The first pose is the result from the previous frame. The second pose is built analytically to replicate the Kinect skeleton. Each starting point will be optimized independently on its own thread. The final pose with the lowest energy will be selected. The overall picture of the calculation in one frame are summarized in Figure 4.5.

4.1.9 Sparse Sampling from Depth Pixels

Normally, the number of depth pixels in the silhouette can be as large as 50,000. Fitting a model to all available pixels cost excessive computation time. Therefore, they are subsampled to N_s pixels. A sampling strategy from Taylor *et al.*'s hand tracking [76] is adopted and modified to ensure two concerns within a sparser sampling. First, there should be enough samples from small body segments (*i.e.*, arm and head). Second, samples should contain pixels from the near-edge area of the silhouette in every body segment. Hence, the Kinect skeleton is utilized to categorize pixels into multiple zones. Each zone will be sampled according to an assigned quota to ensure a good distribution on the silhouette.

Each sampled pixel has its associated 3D position p_i , and its neighbor points usually form a 3D surface. We collect neighbor pixels that are within five units of

Figure 4.5: Calculation of pose estimation in one frame can be processed in parallel for different initializations.

Chebyshev distance in the pixel domain and within 0.05 m (Euclidean distance) in the camera space domain. Principal Component Analysis (PCA) is used to analyze those points in 3D space. The third principal component is selected as the normal vector on that sampled point. As a result, we get the 3D positions $\{\mathbf{p}_i | \mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N_s}$ and the 3D normals $\{\mathbf{n}_i | \mathbf{n}_i \in \mathbb{R}^3; \|\mathbf{n}_i\| = 1\}_{i=1}^{N_s}$ associated to those sampled pixels.

4.1.10 Correspondence Continuous Update (Mesh walk)

When a correspondence point $\mathcal{U} = (f, u, v)$ has to be updated by $(\Delta u, \Delta v)$, the point could move beyond the boundary of the triangle face f to an adjacent face and so on. In that case, we adopt a mesh walking concept proposed by Taylor *et al.* [88], which tries to maintain the tangential continuity when the point has to transfer between two faces. The update repeats until the updated distance is finished.

4.1.11 Correspondence Discrete Update

Similar to the Taylor *et al.*'s method [76], in order to initialize correspondence $\mathcal{U}$ for the next iteration, the latest result from the mesh walking step will be compared to

a set of candidates to choose the best correspondence for the i^{th} sampled point.

$$\nu_i \leftarrow \underset{\nu \in \{\nu_i\} \cup \mathcal{U}_{t\%10}^{\text{cand}}}{\text{argmin}} \left(\frac{\|S_p(\nu, \theta) - p_i\|^2}{\sigma_p^2} + \frac{\|S_\perp(\nu, \theta) - n_i\|^2}{\sigma_n^2} \right) \quad (4.25)$$

$\mathcal{U}_{t\%10}^{\text{cand}} \subseteq \Omega$ is a pre-randomized set of correspondence candidates for iteration t . The random is designed to have three times more density on the upper limbs, neck, and head than the rest of the body.

As the surface area of the human model is much larger than the hand model, instead of increasing the number of the candidates in $\mathcal{U}^{\text{cand}}$ to cover the whole area, a new strategy is proposed to mitigate the problem while using a low number of candidates.

After an update using Eq.4.25, ν_i will be compared to the centroid position from three adjacent faces. If one of those adjacent candidates gives a better match, ν_i will be updated to that point. This process will be repeated until ν_i stops updating. By this way, the ν_i will quickly move to a better point even with a sparse candidate distribution.

4.2 Model Personalization

Model personalization is found to be significant for human pose estimation accuracy [68], [83]. Knowing the size of all body segments before the actual movement tracking can help the articulated mesh model to fit well with the observed point cloud retrieved from Kinect and it adds stability to the tracking.

A method is designed to search for a 10-dimensional shape vector (β) which matches the shape of the subject from one depth image frame with a controlled pose shown in Figure 4.9. There are two steps in the process; a rough estimation, and a fine-tuning.

4.2.1 Rough Shape Estimation using Gradient Descent

This step searches for the shape vector (β) that provides the template model with bone lengths and sizes of body parts that are close to the lengths and sizes from Kinect skeleton tracking results. Pose parameters are ignored in this step. An error function is defined as

$$E_s(\beta) = \sum_{\ell \in \text{Segments}} (\mathcal{L}_\ell(\beta) - \mathcal{K}_\ell)^2 \quad (4.26)$$

$\mathcal{K}_\ell$ is the length of measurement ℓ from the Kinect skeleton.

$\mathcal{L}_\ell(\beta)$ is the length of measurement ℓ from the template model with shape β .

Segments = {*upperArm*, *forearm*, *wholeLeg*, *hipToShoulder*, *shoulderToShoulder*, *ankleToHead*, *wholeArm*}. Each measurement can be described in the following details.

- *upperArm*: Average length of both upper arms.
- *forearm*: Average length of both forearms.
- *wholeLeg*: Sum of the average upper leg length (hip to knee) and the average lower leg length (knee to ankle).
- *hipToShoulder*: Distance from the average position of the hips to the average position of the shoulders.
- *shoulderToShoulder*: Distance from the left shoulder to the right shoulder.
- *ankleToHead*: Distance from the average position of the ankles to the head position (at the jaw height).
- *wholeArm*: Sum of the average upper arm length, the average forearm length, and the average hand length.

The shape vector β is initialized at the mean shape (a zero vector). A gradient descent method is used to iteratively update β by

$$\beta \leftarrow \beta - \lambda \nabla E_s(\beta) \quad (4.27)$$

$\nabla E_s(\beta)$ is the gradient of the error function (Eq.4.26) at point β calculated from an automatic differentiation library. The learning rate λ is initialized at 1. Whenever the update increases the error, λ will be reduced by half. The optimization is considered converged when λ is below 0.01. After that, β will be used to initialize the fine shape tuning step.

4.2.2 Fine Shape Tuning with Shape & Pose Fitting

The explained pose estimation method is modified to search for correspondences ($\mathcal{U}$), shape (β), and pose (θ) together as Θ_β . The same method is used to fit a 48-DOF full-body model ($N_\theta = 48$) to the full-body point cloud. As the pose is controlled, the energy terms we used are *data*, *bg*, *bgJoint*, and *limit* term. As the efficiency is

Figure 4.6: Recording environment for evaluation. The chair in the middle was replaced by a standing subject in a record.

not an issue here, an automatic differentiation library is used to calculate Jacobian matrices without any model simplification. The optimization will keep running until it converges without the limited number of iteration.

4.3 Evaluation

4.3.1 Gold Standard Measurement

To measure the accuracy, a Vicon motion capture system with eight infrared cameras are used as a gold standard. The transformation between the Vicon and the Kinect reference frame is described in Section 3.9. A synchronization between Kinect and Vicon is done by sending a UDP packet via a LAN cable at the beginning of each record. The setup environment is shown in Figure 4.6.

Plug-in Gait model for the upper body with 27 marker locations (from the hips and above) is chosen (Figure 4.7). In order to calculate shoulder, elbow, and wrist joint centers from those surface marker positions, shoulder offset, elbow width, and wrist thickness need to be measured from each subject. The detailed calculations of joint center calculation are described in a Plug-in Gait's document [89].

4.3.2 Subjects & Movement Sequences

Three healthy males (MA, MB, and MC) and three healthy females (FA, FB, and FC) are recruited as our subjects in this experiment. For male subjects, their shirts will be taken off and all the markers are attached directly to their skin. For female subjects,

Figure 4.7: Upper-body marker positions defined by Plug-in Gait model [89].

they will put on a motion capture suit and all the markers are attached to the suit with Velcro tape.

Each subject will move in six sequences. Some example movements are arm raising, shoulder lifting, torso tilting, air punching, forearm rolling, hair combing, jumping, etc. All the movements are done with both hands close as this study does not model the fine structure of hands and fingers. In total, 39,311 depth frames (about 22 minutes) are collected. All the recorded data are made publicly available [90].

4.3.3 Parameter Configurations

The configurations used in this evaluation are

- $N_s = 250$, $N_\theta = 40$, $\sigma_p = 0.003$, $\sigma_n = 0.25$
- $\lambda_{bg} = 10$, $\lambda_{bgJoint} = 25$, $\lambda_{float} = 0.25$
- $\lambda_{limit} = 10000$, $\lambda_{imu} = 15$, $H_{max} = 150$
- The maximum LM iteration allowed per frame is 20
- The initial damping parameter $\gamma = 1000$
- The moving average window size at the post-processing step covers three consecutive depth frames.

Figure 4.8: Subject FB in the personalization process. **Left:** Full-body point cloud with Kinect’s joint positions. **Middle:** Result from the rough shape estimation (Section 4.2.1). **Right:** Result from the fine shape tuning (Section 4.2.2).

4.4 Results & Discussions

4.4.1 Model Personalization

Our personalization strategy can adjust the template model to the different sizes for both male and female subjects. The fitting results can be seen in Figure 4.8 and 4.9. In addition, this personalization technique is also applied successfully to 26 more subjects (11 males, 15 females) with different ages and shapes in Chapter 5 and 6. The subject details are available in Appendix B.

Figure 4.9: Shape fitting results from six subjects in a controlled pose. Note that male (top row) and female (bottom row) use two different templates.

4.4.2 Quantitative Pose Estimation Results

All the position errors of the upper limbs (shoulder, elbow, and wrist) can be seen in Figure 4.10, and Figure 4.11. Other joints such as head, neck, and spine are not evaluated quantitatively as there is no common definition of those joint positions between the gold standard model and the SMPL model. Those figures show two sets of results. The first set is from the whole dataset. The second set is from 1,452 selected frames with occlusions.

Figure 4.10: Accuracy profiles at all threshold distances. **Left column:** from full dataset. **Right column:** from challenging frames with occlusions. At any distance thresholds, the proposed method always gives a better overall accuracy over Kinect SDK.

Figure 4.11: Comparisons of joint position errors from full dataset (**left**) and from frames with occlusions (**right**). Our method reduces the average error from Kinect SDK by 25.9% in general cases and 43.7% in cases with occlusions.

Improvements Over Kinect SDK

On average, the proposed method decreases the error of the upper limb joint positions by 25.9% (from 48.4 mm to 35.9 mm) in general cases and 43.7% (from 99.3 mm to 55.9 mm) in cases with occlusions. The error profiles show that the accuracies of our methods (with or without IMU) are better than those of Kinect SDK at any error distance threshold. The differences become much larger in the occlusion case.

Among the three joints, the wrist gains the most error reduction at 31.4% while the elbow and the shoulder gain similar error reductions at 23.56% and 23.46% respectively.

The Proposed Method Without the *imu* Term

In order to look closely at the effectiveness of the *imu* term, the evaluation is repeated without *imu* term in the optimization. On average, dropping the *imu* term increases the error slightly by 4.0% (2.8% for the wrist, 6.2% for the elbow, 2.2% for the shoulder). In contrast, the effect of dropping the *imu* term becomes clear with occlusions. In the occlusive case, dropping the *imu* term causes the average error to increase by 24.9% (42.5% for the wrist, 24.5% for the elbow, 4.1% for the shoulder). Unlike the wrist and the elbow, the shoulder position error does not increase much as the IMUs are too far to directly affect the shoulder.

4.4.3 Qualitative Improvement Over Kinect SDK

Flexible Spline

When a subject tilts his body to one side *i.e.*, rotation about the frontal plane, Kinect SDK always has a problem in tracking joints along the body core. Neck, SpineShoulder, SpineMid, and SpineBase joint positions will form a strictly straight line which

is incorrect. The Kinect skeleton also wrongly tracks an arm on one side as a consequence. On the other hand, the SMPL skeleton model has multiple DOF at the spine which allow the upper torso section to bend naturally and fit the model more correctly to the observation (Figure 4.12a).

Occluded Shoulder

When a shoulder is occluded by an arm section or by the torso, Kinect skeleton usually picks a shoulder position lower than it should (Figure 4.12b). Our method can correct such a problem in most of the cases.

Occluded Arm

Occasionally, an arm can be partially or fully occluded by the other arm, by the head, or by the torso. In those cases, Kinect usually has notably less clue to infer the elbow and shoulder positions. However, the wrist-mounted IMUs give our method access to the forearm pointing direction which often helps when the visual information is not complete. Figure 4.12c, 4.12d, and 4.12e show some of those cases seen in the dataset.

Arm Jumping in Noisy Frame

When forearms are between the torso and the camera, the depth sensor is likely to generate a lot of *flying-pixel* noise at the edge of the forearms. It commonly confuses the Kinect skeleton for upper limb tracking. Our method usually mitigates such problem.

Shoulder Lift

When subjects lift up one or both of their shoulders, the movement range extracted from Kinect SDK will be diminished and become too subtle to be observed easily. Thanks to the design of SMPL model that does not stick shoulders rigidly to the torso, the shoulder can move around in a reasonable range and such movements can be captured as shown in Figure 4.12f. This feature opens a new opportunity to detect a common compensatory movement in upper-limb rehabilitation exercises without any markers.

Figure 4.12: The cyan linkages are the joint positions from the Vicon system (the gold standard). The pink linkages are the Kinect skeleton. The yellow linkages are from the proposed method. **(a)** When a subject tilts his body, notice that the Kinect spine is unnaturally straight. The spine from our method is more realistic. **(b)** When a shoulder is occluded, notice that shoulder positions from the Kinect skeleton are off to the lower positions. **(c)** The right forearm is behind the head. **(d)** The right arm swings to the space behind the torso. **(e)** The right arm is heavily occluded by the left arm. **(f)** One shoulder is lifted.

4.4.4 Limitations

A few issues about the correctness of the tracking have been observed. First, self-penetration still occurs in our tracking but not often. It could be avoided by adding a self-penetration penalty to the energy function. Second, as our method relies on the Kinect skeleton as an initialization pose in every frame, the tracking can simply go wrong if the subject turns more than 90 degrees away from the camera. However, such large turn can simply be excluded from the design of exercise prescription.

Currently, the algorithm takes about 1.5 seconds per frame on a CPU (Intel Core i7-3370). This speed limits our method to a group of applications which allows delayed feedbacks such as rehabilitation exercise assessment.

Figure 4.13: Joint locations and their ID for reference. The arrows are X and Y-axis of the upper torso reference frame (joint 9)

4.5 Upper-body Features Extraction

Since the pose representation in the form of each DOF is designed to facilitate the numerical optimization, this representation is not easy for a human to comprehend directly. Therefore, a new representation is proposed in this section to make a pose more understandable to a human. Importantly, this representation must allow the conversion back to the original representation for visualization purpose. This representation will be used in Chapter 5 and 6.

The new representation composes of three dimensions relating to the orientation of the upper torso, and ten dimensions relating to each upper-limb. To simplify the explanation, this section will describe the features for the upper torso and the right limb. This representation has 13 dimensions in total.

Upper Torso Orientation (three dimensions)

The orientation of joint 9 (Figure 4.13) will be used to calculate these three features. To do so, a temporary starting orientation of the upper torso (joint 9) is defined to have

- Y-axis pointing upward according to the gravity,
- X-axis pointing in the direction of a cross product between the upward direction and the negative Z-axis of the Kinect reference frame, and
- Z-axis is a cross product of X and Y-axis.

This orientation will be rotated two times to match the observation. The first rotation (one DOF of **left/right turning** angle) is done around Y-axis of the starting orientation. The second rotation (2 DOF of frontal & lateral flexion) will be done around an axis perpendicular to both the first rotation axis and the observed Y-axis of joint 9. The angle is used as the magnitude multiplied to the second rotation axis. Since the second rotation axis has only X and Z components, X component can be interpreted as the **frontal flexion**, and Z component can be interpreted as the **lateral flexion**.

The positive **left/right turning** means the user is twisting the upper torso to the left (relative to the Kinect optical axis). The positive **frontal flexion** means the user is tilting the upper torso forward (relative to the gravity). The positive **lateral flexion** means the user is tilting the upper torso to the right side (relative to the gravity).

These torso-related features are selected because a trunk displacement measure in a reach-to-grasp task can explain more than 50% of the variance in Fugl-Meyer Assessment (FMA) score [56]. Furthermore, the torso motion is the most focused feature among the methods of compensation detection [58], [59].

Right Upper-Arm Pointing Direction (three dimensions)

The pointing direction from joint 17 (right shoulder) to joint 19 (right elbow) in the reference frame of joint 9 (upper torso) is normalized and used as the right upper-arm pointing direction.

Right Forearm Pointing Direction (three dimensions)

The pointing direction from joint 19 (right elbow) to joint 21 (right wrist) in the reference frame of joint 9 (upper torso) is normalized and used as the right upper-arm pointing direction.

Right Forearm Pronation (one dimension)

Forearm pronation is defined according to the plain that the forearm will sweep through if the elbow flexion/extension is done at that moment. With the thumbs-up gesture, if the thumb points parallel to the plane, we consider the pronation angle as zero. The deviation of the parallel angle towards the pronation side is considered positive. On the other hand, rolling the forearm to the other side (supination) will give a negative angle.

The plane can be retrieved from the triangulation of joint 17 (right shoulder), joint 19 (right elbow), and joint 21 (right wrist) when the elbow is not fully extended. However, when the elbow is fully extended and those three joints are aligned is a nearly straight line, the forearm pronation will not be available. The gap will be filled with interpolation later.

The thumb pointing direction can be retrieved from the orientation of the IMU on the right wrist (A_K1). If the thumb pointing direction is aligned with the negative X-axis of the IMU, we get the thumb pointing direction as $rot({}^A_Kq, -\hat{i})$ where $\hat{i}$ is the unit vector along X-axis.

Right Wrist Position (three dimensions)

The wrist position is defined by the position of joint 21 in a new reference frame called *core reference frame*. The origin of this frame is calculated by projecting the chest center point (the midpoint between joint 13 and 14) to the table plane. The Y-axis of this frame is pointing upward (defined by gravity). The X-axis has the direction of the cross product between Y-axis and the vector that point from the *core* origin to the Kinect origin. The X-axis points to the left side of the subject. As a result, the Z-axis points forward parallel to the ground plane.

4.6 Conclusion

With an upper-body rehabilitation exercise assessment as the main target application, a method to improve the Kinect skeleton tracking has been designed, imple-

mented, and evaluated. Our key contributions are 1) the integration of the personalizable SMPL model with the simultaneous pose-correspondence optimization technique which allows the detection of scapular movements, 2) the demonstration of the benefit of adding *imu* term directly to the generative pose optimizer in the context of a single depth camera, and 3) the release of the first public mocap dataset that contains the record from Kinect V2, synchronized IMUs, and a marker-based motion capture system [90]. The evaluation on the dataset shows both qualitative and quantitative improvements over the Kinect SDK.

Chapter 5

Unsupervised Phase Learning & Extraction from Repetitive Movements

Movements of human are challenging to model due to their large variations in anatomy, subject conditions, or even the state of the environment. The most studied movements are in the group of repetitive tasks such as bipedal gait, quadrupedal gait, athletic or rehabilitative exercises. Modeling a particular piece of movement usually needs an expert to observe the movements and to engineer a heuristic way to extract key characteristics from motion capture data to make it useful. This chapter will demonstrate that some parts of such time-consuming process can be done automatically with machine learning.

One fundamental feature of repetitive movements is phase. By knowing phase at all sample points in a movement record, a lot more information such as repetition count, frequency, and average movement trajectory can also be extracted. However, modeling a movement for phase extraction is not a trivial task because the same movement can have variations in speed and trajectory.

There are a few common techniques to assign phase value to every sample in the record. The simplest way is to manually assign phase to the beginning and the ending point of every period and fill in the gaps with linear [91] or non-linear interpolation. However, this method is not scalable with large data size. To make it more automatic, specific feature detection algorithms can be designed to assign some pre-known phases associated with the features. For example, in a rat gait analysis, the direction of horizontal velocity is used to indicate the beginning of the stance phase [92]. In a human gait analysis, the signature of a heel strike can be used to identify the transition point from a swing phase to a stance phase [93]. Sometimes, digital

markers such as a mechanical switch or capacitive sensors on the object are used to mark the beginning and the end of each movement segment [37], [94]. Nonetheless, those features and digital markers are usually designed specifically for each movement, and they need to be redesigned for every new type of movement.

To automate the gap filling process, one common technique is to use linear interpolation. However, it is inaccurate in the situation where phase progressions in the movement are inconsistent. Therefore, dynamic time warping (DTW) is introduced to deal with the inconsistency in the phase progression. DTW has become a common technique in human movement research to align multiple trials of a movement [95]. Still, the need of manually-segmented template prevents the whole procedure from being fully automated.

If the generalization of the procedure is desired, there are two common ways of phase extraction.

The first way is Hilbert transform [96] which assumes that the input signal itself is just the real part of a complex signal. The method can recover the imaginary part of the signal from the real part. Then, the instantaneous phase can be calculated from the angle in the complex coordinate [97].

The second way is to convolute the input signal with a complex Morlet wavelet [98]. The phases of the complex time series result can be used to estimate the phases of the signal. This complex convolution can be thought as a parametric alternative of the Hilbert transform where the center frequency and the bandwidth of the wavelet can be chosen [99].

However, some common limitations of the Hilbert transform and the complex wavelet convolution are the assumption that the signal should be similar to a sinusoidal pattern for the method to have a good phase estimation. If the signal rises and falls multiple times in one period, it can disturb the extracted phase sequence. In addition, both methods are designed for single-channel time-series data. If there are multiple channels from the same source, there is no obvious way to combine all those channels to extract just one phase sequence.

So far, the use of learning methods to enhance the automation of phase extraction has not been explored. Unlike other classification problems with discrete output, one difficulty of learning the phase extraction is the continuity of the phase which makes a manual labeling process tedious and inaccurate. Because of this fact, we believe that unsupervised learning is a more practical way to solve this phase extraction problem.

In this chapter, an unsupervised learning method is designed to automatically

model a repetitive movement and use that model to extract phase at the level of each individual sample. As the learning method is unsupervised, the training data of the target repetitive movement do not need to be labeled with phase. The learning algorithm only needs one or more continuous records of the target repetitive movement without the need for time-consuming pre-segmentation.

5.1 Phase Learning & Extraction Method

5.1.1 Method Overview

The proposed method contains two steps, a learning step and an inferencing step. First, a neural network will be trained with high-dimensional time-series data recorded from one or more records of the target movement. As the data do not come with phase, a special cost function is designed to force the network to organize itself and learn the concept of phase.

Every pair of two consecutive sampling windows will be mapped to two consecutive phase values. This mapping is done by the neural network. A proper mapping should align these two consecutive phase values in a gradually progressive way. Collectively, with a large enough number of pairs in the training data, the network will eventually be able to find a usable phase mapping to all samples in the training data. After the learning, the network will be able to infer a phase value by feed-forwarding a small window of the movement data through the trained network.

The artificial neural network is chosen over other models because of its capability to adapt itself to provide a complicated non-linear continuous transformation that fits and satisfies most of the cost functions.

5.1.2 Neural Network Architecture

The network architecture is shown in Figure 5.1. The input layer consists of two sections which are the signal section and its heading direction section.

The first section is a window of $2w + 1$ consecutive samples of the multi-channel signal. A channel in this context means a one-dimensional time-series sequence of an actual recorded signal or any extra extracted feature that is from the same source. Each sample of the signal is a K -dimensional vector.

The second section is a window of $2w + 1$ consecutive samples of normalized vectors of K slopes. Each vector is called *signal heading direction*. Because the signal is a sequence of K -dimensional vector, the signal heading direction at each sample

Figure 5.1: Neural network architecture for phase extraction. The output from the feed-forwarding process will always land on the unit circle which represents the instantaneous phase of the input window.

is also a K -dimensional vector. The calculation of the signal heading direction is described in Section 5.1.3.

The network has Q fully connected hidden layers with hyperbolic tangent activation function. Each hidden layer contains H hidden nodes. All nodes in the last hidden layer are fully-connected to two nodes in the pre-phase layer with linear activation function. The result in the pre-phase layer will be normalized to a 2D unit vector at the output layer.

The feed-forwarding function $\mathbf{U}(\Theta, s) = [x \ y]^\top$ can be seen as a transformation from the $2K(2w+1)$ -dimensional input layer (s) to a 2D point on a unit circle (x, y) . All the trainable weights and biases in the network are represented by the symbol Θ . This 2D representation of phase is chosen to avoid the discontinuous wrapping point (at 0 and 2π) in the 1D phase representation that usually confuses the neural

network. However, to simplify plots in this article, the 2D phase is converted to the 1D phase value in a range of $[0, 2\pi)$. The conversion from the 2D form (x, y) to the 1D form (ϕ) is done by

$$\phi = \arctan 2(y, x) \%(2\pi). \quad (5.1)$$

Since the *arctan2* operator has its range of $(-\pi, \pi]$, the modulo operator (%) is used to map it to a positive range of $[0, 2\pi)$.

5.1.3 Training Data Preparation

Because this phase learning method is designed to be general for any multidimensional quasi-periodic time-series data with K dimensions per sample, this section describes pre-processing steps that convert an existing K -dimensional time-series data into the form that will be used in the training process. The retrieval of the K -dimensional time-series data may vary across applications. For example, the processing of the upper-limb movement into K -dimensional time series data is explained in Section 5.3.2.

The training data is first standardized so that all K dimensions (from the signal section) have the mean of zero and the standard deviation of one.

At each standardized sample $\mathbf{p}_t$, the $2r + 1$ neighboring samples $\mathbf{p}_{t-r}, \dots, \mathbf{p}_{t+r}$ are used to calculate the K -dimensional signal heading direction $\mathbf{h}_t$. For each dimension, a linear regression is performed on $2r + 1$ data samples to extract one slope. Then, K independent slopes from all K dimensions will be combined and normalized to a unit vector $\mathbf{h}_t$.

As the periodic signal can be seen as a repetitive trajectory that moves in a K -dimensional space with a mixture of arbitrary speeds, the normalization helps the phase learning to be invariant to the speed of the movement in the K -dimensional signal space.

All pairs of two-consecutive input windows will be packed together as $\langle \bar{s}_i, \hat{s}_i \rangle$ where $\bar{s}_i$ is from the earlier window and $\hat{s}_i$ is from the later window with a one-sample shift. If the first input window $\bar{s}_i$ contains $\mathbf{p}_{t-w}, \dots, \mathbf{p}_{t+w}$ and $\mathbf{h}_{t-w}, \dots, \mathbf{h}_{t+w}$, the second input window $\hat{s}_i$ will contain $\mathbf{p}_{t+1-w}, \dots, \mathbf{p}_{t+1+w}$ and $\mathbf{h}_{t+1-w}, \dots, \mathbf{h}_{t+1+w}$.

From one or more data sequences, collectively, the training data will contain N pairs of input data.

5.1.4 Cost Function

In the training, all the weights and biases parameters Θ are searched to minimize a cost function defined as

$$\begin{aligned}
F(\Theta) = & \frac{1}{N} \sum_{i=1}^N \mathcal{P}(\arctan 2(\bar{x}_i \hat{y}_i - \hat{x}_i \bar{y}_i, \bar{x}_i \hat{x}_i + \bar{y}_i \hat{y}_i)) \\
& + \alpha \left[\left(\frac{1}{N} \sum_{i=1}^N \bar{x}_i \right)^2 + \left(\frac{1}{N} \sum_{i=1}^N \bar{y}_i \right)^2 \right] \\
& + \frac{\gamma}{N} \sum_{i=1}^N \frac{1}{\sigma \sqrt{2\pi}} \exp \left(- \frac{|\mathbf{V}(\Theta, \bar{s}_i)|^2}{2\sigma^2} \right) \\
& + \frac{\lambda}{N} \sum_{i=1}^N \frac{1}{\sigma \sqrt{2\pi}} \exp \left(- \frac{|\mathbf{V}(\Theta, \xi(\bar{s}_i, \sigma_w))|^2}{2\sigma^2} \right)
\end{aligned} \tag{5.2}$$

where $\mathbf{U}(\Theta, \bar{s}_i) = [\bar{x}_i \ \bar{y}_i]^\top$ and $\mathbf{U}(\Theta, \hat{s}_i) = [\hat{x}_i \ \hat{y}_i]^\top$ are the feed-forwarding calculations from data pair i . The function $\mathbf{V}(\Theta, \bar{s}_i)$ is a 2D activation vector from the pre-phase layer. The parameters α , γ , and λ are the weights for those terms. This cost function contains four penalties which are *phase progression penalty*, *distribution penalty*, *singularity penalty*, and *marginal penalty* from the first to the forth line respectively.

Phase Progression Penalty

The first *phase progression penalty* term from Eq.5.2 is designed to force every pair of consecutive inputs to have a proper amount of phase progression. The calculation of this penalty from a pair of consecutive inputs can be illustrated in Figure 5.2.

Consider a data pair i which is transformed to the first output point $(\bar{x}_i \ \bar{y}_i)$ and the second output point $(\hat{x}_i \ \hat{y}_i)$ on the unit circle. The penalty for this pair of data depends on the amount of the phase progression from the first to the second output. To make the calculation of the progression automatically differentiable, their cross product and dot product are computed. Given that $g \in (-\pi, \pi]$ is the unknown phase progression in the unit of radian. The cross product $(\bar{x}_i \hat{y}_i - \hat{x}_i \bar{y}_i)$ will be equal to $\sin(g)$ and the dot product $(\bar{x}_i \hat{x}_i + \bar{y}_i \hat{y}_i)$ will be equal to $\cos(g)$. Then, g can be recovered using *arctan2* function as

$$g = \arctan 2(\sin(g), \cos(g)) \tag{5.3}$$

Figure 5.2: Calculation of phase progression penalty from one pair of two consecutive input windows.

$$g = \arctan 2(\bar{x}_i \hat{y}_i - \hat{x}_i \bar{y}_i, \bar{x}_i \hat{x}_i + \bar{y}_i \hat{y}_i). \quad (5.4)$$

The function $\mathcal{P} : (-\pi, \pi] \mapsto \mathbb{R}$ is a penalty profile that maps a phase progression to a penalty value. The function is defined as

$$\mathcal{P}(g) = \begin{cases} \cos(\frac{\pi}{2}(g - c)/(2\pi + c - b)) & \text{if } -\pi < g \leq c \\ \cos(\frac{\pi}{2}(g - c)/(a - c)) & \text{if } c < g \leq a \\ 0 & \text{if } a < g \leq b \\ \cos(\frac{\pi}{2}(g - c - 2\pi)/(2\pi + c - b)) & \text{if } b < g \leq \pi \end{cases} \quad (5.5)$$

while c , a and b are adjustable parameters under conditions $-\pi < c < a < b < \pi$. This penalty profile can be visualized in Figure 5.3. This function is designed to be continuous throughout the domain $(-\pi, \pi]$ including the warping point between π and $-\pi$.

This penalty profile contains three characteristics. First, it should penalize negative phase progression as it does not happen in nature. Second, the phase progression that is too large should be penalized as it does not happen on a general signal with a sufficient sampling rate to obtain the complete waveform. Third, the slope

Figure 5.3: The phase progression penalty profile function $\mathcal{P}$. The positions of the peak (c) and the valley (a, b) are adjustable. Slopes on this penalty profile are designed to encourage every pair of consecutive phase values to adjust its phase progression to the range between a and b .

must induce every pair of consecutive windows to gradually adjust their phase progression (g) towards the proper range between a and b . This process can happen through the back-propagation in the optimization process.

These parameters (a , b , and c) can be adjusted according to the sampling rate and the estimated period length of the target signal. If the prior knowledge about the signal is known, the parameter a can be set as the minimum phase progression of the target signal in one sample pair. On the other hand, the parameter b can be set as the maximum phase progression of the target signal in one sample pair.

The purpose of the parameter c is to destroy the stability of the unwanted *phase collapse* situation (Figure 5.4) especially when $a \leq 0$. Intuitively, the penalty should be the highest at the most negative phase progression of $-\pi$ or $c = -\pi$. However, that condition unintentionally creates a stable point for the *phase collapse* state by providing a near-zero gradient around the phase progression of $-\pi$. This stability can be removed by shifting the parameter c away from $-\pi$ (such as $c = -0.75\pi$). This shift creates an obvious slope around the phase progression of $-\pi$ which encourages any pair of consecutive samples that is mapped to the opposite side of the unit circle to move closer and finally close the loop.

Distribution Penalty

The second term of Eq.5.2 is called *distribution penalty*. It calculates the centroid from all samples in the training data and measures the squared distance from the origin (*i.e.*, the center of the unit circle). This term reflects how well the outputs are distributed on the unit circle. In the case that the outputs are not evenly distributed,

Figure 5.4: If the parameter a is set as zero or below and the parameter c is set as $-\pi$, it is likely that the trained network will map the training data along a diagonal line of the unit circle in the pre-phase layer and does not diverge from this mapping. This *phase collapse* situation can be fixed by shifting the parameter c away from $-\pi$ (such as $c = -0.75\pi$).

the centroid will shift away from the center, and the network will be penalized. However, the *distribution penalty* term does not work perfectly by itself. Sometimes, the neural network is able to transform all the training data into one straight line on the pre-phase layer and create two roughly equal clusters of phase on the opposite side of the unit circle (Figure 5.4). This mapping behavior perfectly satisfies *distribution penalty* term without reaching their objective of distributing all the mapping throughout the unit circle. This problem is solved by adjusting the phase progression penalty profile parameter c to be away from $-\pi$ as being discussed in the previous paragraph.

Singularity Penalty

The third term of Eq.5.2 is called *singularity penalty* term. This penalty is designed to prevent the training data distribution in the pre-phase layer from staying too close to the origin. Due to the normalization in the output layer, the area around the origin is the place where the phase output can be unstable especially with noisy inputs. This penalty can be visualized as a 2D Gaussian function with the highest penalty at the origin in the pre-phase space. The penalty reduces with more distance from the origin.

Marginal Penalty

As the target application needs to learn the phase with data from healthy subjects and use the learned network to extract phase from unhealthy subjects such as stroke patient, the movement patterns of unhealthy subjects may vary significantly from the healthy movement patterns. Due to this fact, the *marginal penalty* term is introduced to improve the generalization for such variation in the unseen movement.

Instead of using the normal input data ($\bar{s}_i$) as being used in *singularity penalty*, this term adds some random noise to the training data before feeding it to the network. The function $\xi(\bar{s}_i, \sigma_w)$ in Eq.5.2 adds Gaussian noise with zero mean and σ_w standard deviation to all the dimensions of $\bar{s}_i$ following by normalizations in all heading direction data. The added noises are randomly generated in every training iteration. This term expands the margin of the data in the input space and the network needs to avoid the mapping of those margin to the origin in the pre-phase layer.

5.1.5 Initialization & Optimization

All the weights are initialized randomly from a Gaussian distribution with zero mean and standard deviation of $1/\sqrt{n}$ where n is the number of nodes in the previous layer of that weight. All the biases are initialized as zeros.

The cost function (Eq.5.2) is implemented on TensorFlow. TensorFlow [100] is a tool that provides automatic differentiation for a large chain of data flow together with gradient-based optimizations. This tool allows the creation and experimentation of highly complex cost functions without the reimplementing of the gradient calculation in every single adjustment to the cost function. Adam optimization algorithm [101] with *learning_rate*=0.001, *beta1*=0.9, *beta2*=0.999, and *epsilon*= 10^{-8} is used to minimize the cost function.

5.1.6 Phase Extraction

After the training, a newly unseen record can go through the same pre-processing described in Section 5.1.3 but with all pre-calculated parameters from training data (K means and K standard deviations). Then, each sample of the standardized data with augmented heading direction will be fed into the input layer of the trained neural network. The feed-forwarding will infer the phase for that sample.

Figure 5.5: Three synthetic signals used in the experiment. **Row A:** Heart shape. **Row B:** Bow shape. **Row C:** Acrobat shape.

5.2 Validation Experiment on Synthetic Data

This section validates the capability of the method to learn from periodic signals that form 2D loops in different shapes. The use of synthetic data allows us to have the ground truth of phase to compare with the extracted phase. Such comparison cannot be done quantitatively with any actual signals.

5.2.1 Synthetic Signal Generator

Three shapes of signal are chosen. Each signal is governed by two signal generators $G_x(t)$ and $G_y(t)$. They are designed to have an equal period of 2π i.e., $G(t + 2\pi) = G(t)$. Figure 5.5 shows the shape and the waveform of each signal with and without disturbance.

Heart shape

This is an uncommon loop with both smooth and sharp turning points. The signal is generated by

$$G_x(t) = 1.8 \sin^3(t) \quad (5.6)$$

$$G_y(t) = 1.3 \cos(t) - 0.5 \cos(2t) - 0.2 \cos(3t) - 0.1 \cos(4t). \quad (5.7)$$

Bow shape

The loop is twisted and contains one crossing point in the phase-space plot. To properly extract phase from this shape, the trained network must create a transformation that untwists the trajectory into a non-crossing loop. The signal is generated

by

$$G_x(t) = 1.4 \cos(t) \quad (5.8)$$

$$G_y(t) = 2.9 \sin(t) \cos(t). \quad (5.9)$$

Acrobat shape

This shape is selected to explore the case of multiple crossing points in a loop. In addition, the waveforms of both channels go up and down twice in one period which makes them challenging for other techniques such as Hilbert transform. The signal is generated by

$$G_x(t) = 1.6 \cos(t)(\sin(0.5t + 0.5) + \cos(1.5t + 0.5)) \quad (5.10)$$

$$G_y(t) = 1.6 \sin(t)(\sin(0.5t + 0.5) - \cos(1.5t + 0.5)). \quad (5.11)$$

5.2.2 Signal Disturbance

To simulate uncertainties, the generated data will be disturbed in both temporal and spatial ways.

The time sampling starts at $t = 0$ and the progression in every step will be sampled from a uniform distribution in a range from 0 to $4\pi/100$. The signal is expected to be sampled around 100 times per period. This random progression creates inconsistency in every step of the phase progression.

The sampled data in both x and y channels will be added with Gaussian noise with zero mean and 0.05 standard deviation. These disturbed data are ready for the process described in Section 5.1.3.

5.2.3 Training Configuration

For each shape, a sequence of 15,000 samples from the disturbed signal is used as the training data. The neural network contains 5 hidden layers with 30 nodes in each layer ($Q = 5$, $H = 30$). Other parameters are $r = 7$, $w = 3$, $a = 0$, $b = 4\pi/180$, $c = -0.75\pi$, $\alpha = 1$, $\gamma = 1$, $\lambda = 0$ and $\sigma = 1$. The training is done for 5,000 iterations.

Figure 5.6: Learning results at the 5000th iteration. **Row A:** Training data distribution at the pre-phase layer and the normalized layer. All the synthetic shapes are unfolded into one clean loop. **Row B:** Standardized training data in phase-space diagrams painted with the extracted phase.

5.2.4 Results & Discussions

Training Data Distribution in Pre-phase Layer

For all three shapes, all the training data are mapped to one loop surrounding the origin in the pre-phase layer (Figure 5.6A). For the bow shape and the acrobat shape, all the crossing points are unfolded within 500 and 1,500 iterations respectively. Figure 5.6B shows that each part of the signal is mapped consistently to a narrow range of phase. The results from every training iteration are available in the supplementary videos [102].

Phase Extraction from Unseen Data

The trained networks are used to extract phase from a newly generated sequence (15,000 samples) with the same kind of disturbances. Some examples of the phase extraction results are shown in Figure 5.7.

Because the data is synthetic, the phase at any sample of the signal is known as $t\%(2\pi)$. A perfect phase extraction should provide a constant difference between the phase output and $t\%(2\pi)$. This phase difference at a sample can be calculated by

$$D(\phi_1, \phi_2) = \min((\phi_1 - \phi_2)\%(2\pi), |(\phi_1 - \phi_2)\%(2\pi) - 2\pi|) \quad (5.12)$$

Therefore, the variation of the phase difference can be used to evaluate the quality

Figure 5.7: **Top row:** Disturbed signals generated for testing purpose. **Bottom row:** Comparisons between the ground truth phase sequences and the extracted phase sequences. A perfect phase recovery is expected to provide a sequence of constant phase difference.

of the phase extraction. Our phase extraction provides fairly constant differences from the ground truth phase with standard deviations of 0.049, 0.047, 0.048 rad for the heart, the bow, and the acrobat shape respectively.

5.3 Experiment on Repetitive Human Movements

5.3.1 Data Source

For this experiment, 12 healthy subjects and 14 unhealthy subjects are recruited to collect training data and test data respectively. The details of the subjects and the inclusion/exclusion criteria can be seen in Appendix B.

Four Repetitive Exercises

Four different repetitive exercises in a sitting pose are designed for this experiment.

- Rectangular drawing (**RECT**) : a subject slides his right hand to 4 spots on a table to draw a rectangle.
- Hair combing (**COMB**) : a subject starts with his hand on a table and lifts the hand to comb his hair and puts the hand back on the table.
- Tap in a loop (**TAP**) : a subject starts with his right hand on a table and raises the hand to touch his nose and left shoulder and puts the hand back on the table.
- Water pouring (**POUR**) : a subject holds an empty cup and perform a pouring action to a bucket on the table and moves the hand back to the starting posi-

tion. If the subject cannot hold the cup, a flexible strap will be used to tie the cup to the palm.

Training Data

Twelve healthy subjects (five males, seven females, 25-59 years old) are recruited to perform those four exercises. One hundred repetitions per exercise per subject are recorded. All subjects are asked to mix slow, medium, and fast paces to their movements for better data variations. The speeds of the movements across multiple subjects do not need to be controlled to be the same. In total, each exercise contains around 60 minutes of record time or around 100,000 frames. The recorded data are fed into the training process without any segmentation.

Testing Data

Fourteen stroke survivors (six males, eight females, 44-86 years old) with their right arm affected are also recruited to perform those four exercises. All subjects, except one, are over six months post-stroke. Their Fugl-Meyer gross upper limb scores range from 19 to 41 while the full score is 42. Each subject is encourage to perform at least 20 repetitions per exercise if possible. However, their movement does not need to be perfect as we want to test the capability of the phase extraction method for those cases. The details of the subjects and the inclusion/exclusion criteria can be seen in Appendix B.2.2.

5.3.2 Data Preprocessing

All the repetitive movement records are processed by the pose estimation methods described in Chapter 4. For each frame, features are extracted according to Section 4.5. In this case, only the upper torso and the right limb are focused. In total, there are 13 dimensions of features to manage.

- right upper-arm pointing direction (three dimensions)
- right forearm pointing direction (three dimensions)
- right forearm pronation/supination turning angle
- upper torso left/right turning angle
- upper torso frontal flexion

- upper torso lateral flexion
- wrist position (three dimensions)

Azimuthal Equidistant Projection

Among those features, the first two are data on the surface of a unit sphere. This spherical representation causes a trouble later in the data reconstruction process in Chapter 6. As the reconstruction is based on dimensionality reduction by principal component analysis which is a linear method, the reconstruction of the spherical data in the 3D space is not likely to land on the unit sphere surface. Therefore, a new representation is proposed to ensure the validity of the reconstruction. At the same time, it must avoid the discontinuity of the data at the potential wrapping point to facilitate the learning process.

The solution is to rotate the spherical training data around the origin to make the mean of the training data stay on the positive side of Z-axis (the top side of the sphere). Then, the azimuthal equidistant project is applied to transform the data into a flat 2D space. By this projection, the point at the north pole will be at the origin of the new 2D space. The point at the south pole will be torn and stretched out to become a circle with a radius of π in the new 2D space. Because of this warping point at the south pole, the pre-rotation of the training data to the north pole needs to be done. By doing so, the chance that the data will touch the south pole in a normal scenario is low. The idea is illustrated in Figure 5.8.

Given a set of 3D points ($\mathbf{o}_i, i = 1..N_o$) on a sphere (from the training data), we want to find the rotation matrix R that move the average of those points to the north pole (positive side of Z-axis). To do so, three vectors to represent three rows of the rotation matrix are calculated.

- $\mathbf{r}_z = \mathcal{N}(\frac{1}{N_o} \sum_{i=1}^{N_o} \mathbf{o}_i)$ is a normalized average vector of all 3D points.
- $\mathbf{r}_y = \mathcal{N}(\mathbf{r}_z \times \text{PC}_1(\mathbf{o}_i - \mathbf{r}_z))$ is a normalized cross product between $\mathbf{r}_z$ and the first principal component from all 3D points subtracted by $\mathbf{r}_z$.
- $\mathbf{r}_x = \mathbf{r}_y \times \mathbf{r}_z$.

Then, we get the rotation matrix

$$R = [\mathbf{r}_x \ \mathbf{r}_y \ \mathbf{r}_z]^\top \quad (5.13)$$

The original point ($\mathbf{o}_i$) will be rotated by $\tilde{\mathbf{p}}_i = R\mathbf{o}_i$.

Figure 5.8: Trajectory of forearm pointing direction during a repetitive combing movement from multiple subjects. **Left:** Original 3D representation ($\mathbf{o}_i$) in the upper-torso reference frame. **Right:** Rotated to be around the north pole ($R\mathbf{o}_i$). **Below:** Result of azimuthal equidistant projection (x_{azeq}, y_{azeq}).

Given that the rotated point $\tilde{\mathbf{p}}_i = [\tilde{x} \ \tilde{y} \ \tilde{z}]^\top$, the azimuthal equidistant projection can be done by

$$x_{azeq} = \tilde{x} \cdot \frac{\arccos(\tilde{z})}{\sqrt{\tilde{x}^2 + \tilde{y}^2}} \quad (5.14)$$

$$y_{azeq} = \tilde{y} \cdot \frac{\arccos(\tilde{z})}{\sqrt{\tilde{x}^2 + \tilde{y}^2}} \quad (5.15)$$

The projection can also be inverted from 2D to 3D by the following equations.

$$\tilde{x} = x_{azeq} \cdot \frac{\sin(m)}{m} \quad (5.16)$$

$$\tilde{y} = y_{azeq} \cdot \frac{\sin(m)}{m} \quad (5.17)$$

$$\tilde{z} = \cos(m) \quad (5.18)$$

where $m = \sqrt{x_{azeq}^2 + y_{azeq}^2}$.

By applying this projection to the upper-arm pointing direction and forearm pointing direction, the total number of dimensions is reduced from 13 to 11. We name this pose representation as *AZEQ*.

Sample Standardization & Heading Direction Augmentation

Every training sample in the form of *AZEQ* representation (11 dimensions) will be used to calculate means and standard deviations for each dimension. These means and standard deviations are used to standardize the data so that all $K = 11$ dimensions of the training data have the mean of zero and the unit variance as explained in Section 5.1.3. Then, the heading direction will be augmented to the standardized data as described in Section 5.1.3.

5.3.3 Training Configuration

The neural network contains 5 hidden layers with 30 nodes in each layer ($Q = 5$, $H = 30$). Other parameters are $r = 3$, $w = 1$, $a = -2\pi/180$, $b = 15\pi/180$, $c = -0.75\pi$, $\alpha = 0.02$, $\gamma = 0.02$, $\sigma = 1$, $\lambda = 0.2$, and $\sigma_w = 0.5$. The training is done for 10,000 iterations.

The parameter a is set to a small negative value because those repetitive movements of human usually contain short pauses that have phase progression around zero or small negative values.

5.3.4 Results & Discussions

Observe the Learning from Pre-phase Layer

One way to observe the progress in the learning is by visualizing the distribution of the training data in the pre-phase layer. The shape of this distribution will evolve slowly from a random shape into a loop that flows around the origin (Figure 5.9).

Figure 5.9: Distribution in the pre-phase layer (multi-color point cloud) and the output layer (green unit circle) of samples from the training data at different iteration of the training process. The light grey points are mapped from the noise-added input used in the *marginal penalty* term. Each color in the multi-color point cloud represents data from one subject. In all four exercises, the learning always starts with random mapping (top row) and it slowly learns to form a loop around the origin (bottom row). **From left to right column:** COMB, RECT, TAP, and POUR exercise. The number of iteration **from the top to the bottom row** are 0 (initialization), 10, 50, 100, 1000, and 10000. See supplementary videos [102] for more details.

Pre-phase Data Distribution from Unseen Subjects

When the trained networks are used to extract unseen movements from unhealthy subjects, the data distribution in the pre-phase layer can also be used to tell the quality of the phase inference. If the distribution forms the loop around the origin without passing through the origin, the phase extraction result will work as intended. On the other hand, if the pre-phase distribution does not loop around the origin, there are two possibilities. First, the subject may not be able to complete the full cycle of that exercise. Second, the variation of the new data may be an outlier to the learned model.

In this experiment, the pre-phase distribution from all 14 unhealthy subjects are illustrated in Figure 5.10, 5.11, 5.12, and 5.13 for COMB, RECT, TAP, and POUR exercises respectively. For all four exercises in general, the shape of the pre-phase loops extracted from unhealthy subjects seems to match well with the shape of the final loop from the training data (Figure 5.9). For example, the POUR exercise forms a rounded diamond shape.

For the COMB exercise, movements of some subjects are unable to map into a well-distributed loop (*i.e.*, KKT, JSH, QSL, LKL, and LSL). This is because their movements did not go through the full cycle of the movement especially the section which the hand has to go over and behind the head.

For the RECT, TAP, and POUR exercises, most of the repetitions map well to the circular loop. However, some repetitions map closely to the region around the origin. These bad mapping can be a result of anomalies in the movement. As some types of anomalies can cause fluctuations in the phase sequence, it can be used for anomaly detection in Chapter 6.

Figure 5.10: Distribution in the pre-phase layer (red point cloud) and the output layer (green unit circle/arc) of samples from the COMB exercise performed by unhealthy subjects. As COMB is the most difficult movement among the four, many subjects are unable to complete the cycle of the movement. As a result, some of the extracted loops are noisy or incomplete.

Figure 5.11: Distribution in the pre-phase layer (red point cloud) and the output layer (green unit circle) of samples from the RECT exercise performed by unhealthy subjects. Subject QSL frequently lost attention in this movement and did not follow the instruction well causing the messy loop.

Figure 5.12: Distribution in the pre-phase layer (red point cloud) and the output layer (green unit circle) of samples from the TAP exercise performed by unhealthy subjects.

Figure 5.13: Distribution in the pre-phase layer (red point cloud) and the output layer (green unit circle) of samples from the POUR exercise performed by unhealthy subjects.

Phase Extraction in Unseen Subjects

With multiple subjects and intentional speed variations in the training data, the network has learned from them and is able to performed phase extractions from unseen healthy subjects despite the large variation in the pace of each movement. Some examples of those extraction can be seen in Figure 5.14, 5.15, 5.16, and 5.17). This robustness against speed variation fits well with a need in rehabilitative movement analysis. The model can be trained using movements from healthy subjects. Then, the trained model could be used to analyze rehabilitative patients who usually have relatively slower movement.

One important characteristic that emerges from this learning method is the responsiveness to the pause in the movement. As the movement slows down to a halt, the phase progression also slows down and stay around a particular phase value. This ability is important for repetitive exercise as all individual always has unique pace and pausing rhythm.

For a difficult movement such as COMB, it could be possible that some subjects are unable to complete the full movement cycle. As a result, the extracted phase may loop through a partial section of the whole phase range. An example of such case is illustrated in Figure 5.17.

Sometimes, the subject might not be able to follow the exercise instruction correctly. In such case, the phase extracted from the completely incorrect movement can be uninterpretable. An example is illustrated in Figure 5.18.

Computational Cost

The training speed depends largely on the size of the network and the size of the training data. For the human movement case which use around 100,000 pairs of training samples, it takes under 10 minutes on an Nvidia GTX 1080 Ti GPU to optimize for 10,000 iterations. The feed-forwarding process on the network to calculate an instantaneous phase takes less than 1 ms on a core of Intel Core i7-3770 CPU.

5.4 Additional Discussions

5.4.1 Flip Mapping Problem

When the parameter a is set to a negative value, the network can occasionally converge to the mapping where all the training data progress clockwise (backward) instead of counter-clockwise. When this situation occurs, the network is usually

Figure 5.14: Phase extraction of RECT exercise from unhealthy subjects, TKB and SGH. Forearm 1 and 2 are the two selected dimensions from all 11 dimensions (K) before the standardization. Although the movement from TKB is more than three times slower than the movement from SGH, the phase extraction can adapt to those speed variations. Moreover, the extracted phase sequences are able to capture the four long pauses in case of TKB and three short pauses in case of SGH which reflect different styles of motion at the corners of the rectangle drawn.

trapped forever in a local minimum. A way to fix this problem is to repeat the same training multiple times with newly random weight initialization until the average phase progression becomes positive.

5.4.2 Constant Phase Offset (Optional)

Because the proposed cost function (Eq. 5.2) is rotational invariant at the pre-phase layer and the output layer, the mapped data distribution on those layers can be rotated around the origin without increasing the cost function. This fact allows the user to choose a particular spot on the waveform to be mapped to the transition point between two consecutive periods (i.e., the phase of 2π and zero). For example, the user may need the table touching point to be the beginning of each round in the TAP exercise for an easier interpretation. This can be done by adding a constant rotation after the output layer of the trained network. The amount of rotation can be retrieved by feed-forwarding the signal from the target spot into the trained network to get a phase. The negative of that phase value is used in the additional rotation.

Figure 5.15: Phase extraction of TAP exercise from unhealthy subjects, JSH and NRH. Three possible pausing points are seen at phase values around 0π , 0.5π , and 1.25π for the nose, the shoulder, and the table respectively. While JSH seems to transit smoothly through those three taps, NRH's movement seems to contain some difficulties that are shown in the phase fluctuations.

Figure 5.16: Phase extraction of POUR exercise from unhealthy subjects, NAH and LSL. Forearm 1 and 2 are the two selected dimensions from all 11 dimensions (K) before the standardization. Two pausing points of this action are at the staring point and the moment of pouring.

Figure 5.17: Phase extraction of COMB exercise from unhealthy subjects, CST and KKT. The phase extraction from CST shows a smooth speed transition to the pausing point at the middle of every repetition. This is the point where the hand is resting on the table. The phase extraction from KKT is limited to a small range of phase (1.5π to 2π rad) because he can only move his arm to the head level but did not reach the top and the back of the head to complete the full cycle.

Figure 5.18: Phase extraction of RECT exercise from an unhealthy subject, QSL. Forearm 1 and 2 are the two selected dimensions from all 11 dimensions (K) before the standardization. The subject could not follow the instruction continuously due to her attention problem. As a result, she completes about two rounds of RECT movement and starts to move in wrong directions after that causing the phase extraction to be partially random.

Figure 5.19: With phase information assigned automatically to every sample in the training data, the data can be aligned directly from the record without any pre-segmentation or time-warping algorithm. **Grey points:** samples of forearm pointing direction (*AZEQ* representation) during repetitive combing movements from multiple subjects against the extracted phase. **Black lines:** average trajectory in each dimension as a by-product calculated from extracted phases.

5.4.3 Average Movement Extraction

An average trajectory for the trained movement can be extracted by plotting each dimension of *AZEQ* representation against the extracted phase from all samples in the training data. By separating the full range of phase into L equal bins, the average value from each bin can be calculated. This series of pose is one period of movement which can be converted all the way back to the original form of pose for exercise demonstration in 3D. An example of the result is illustrated in Figures 5.19 and 5.20.

5.5 Conclusion

A new universal framework for automatic phase learning and extraction is introduced to tackle some problems in repetitive movement modeling. With a unique advantage of being an unsupervised method with robustness to noise and movement speed variation in the data, this method is suitable for various applications that have access to a fair amount of target movement recorded in a repetitive manner. Our experiments on three synthetic signals and four upper-limb repetitive exercises show the generalization capability of the learning method. Nonetheless, this method could

Figure 5.20: **Grey lines:** Trajectory of forearm pointing direction in *AZEQ* representation recorded from multiple subjects. Zoom in to see the details in the digital version. **Colored dotted loop:** the average trajectory as a result from Figure 5.19.

potentially be useful in many applications as it works with any high-dimensional time-series signal. For example, the use of this technique to extract phase from rats' gait for synchronous robotic rehabilitation was also explored [103].

Chapter 6

Anomaly Detection in a Repetitive Movement

As described in the literature review (Section 2.3), all those kinematic features are designed specifically to detect some specific kinds of compensation or anomaly known by prior knowledge in those specific movements. For example, the range of motion of forearm pronation/supination in a page-turning task is significantly smaller when the subject has suffered from distal radius fracture [91]. Therefore, this knowledge can be pre-programmed to help the clinician in the diagnosis. For a reaching task with a known target position, it is known that a healthy trajectory of the hand will follow a straight line with a smooth single-peak velocity profile [44], [47]. An inter-joint coordination measure calculated from the correlation coefficient of the elbow flexion and shoulder abduction is designed only for the hand-to-mouth phase of a drinking task [52]. The tilt angle of the upper torso is detected to discourage the compensatory movement in a game that needs only shoulder abduction/adduction to control [58].

In contrast, there are more general features that are designed to assess other upper-limb movements selected from standard clinical assessments. There are studies which extracted features directly from three to six IMUs attached on their trunk and upper-limbs [94], [104], [105]. Some examples of these features are

- RMS of accelerometer data and its derivative
- Mean of accelerometer data
- Dominant frequency and energy ratio around the frequency
- Correlation coefficient between a pair of accelerometer time series at zero lag

- Lag time of the peak correlation between a pair of accelerometer
- RMS and maximum of velocity time series (integration of acceleration data)
- RMS of accelerometer data normalized by maximum velocity (jerk metric)
- Approximate entropy (ApEn)

Because these features are generic, they could not be interpreted directly to the quality of the movement like those features described in the first paragraph. Instead, they need to use some statistical or machine learning techniques to translate a set of features into a predicted clinical score such as Chedoke-McMaster, Wolf Motor Function Test, or Fugl-Meyer. With an automatic feature selection and learning with decision forest, Patel *et al.* are able to predict the Functional Ability Scale (WMFT-FAS) score accurately by measuring a subject performing six functional movements [94]. While this method is suitable for the standard clinical assessment, it might not be suitable for daily exercise as the summarized assessment score cannot give a clue of how the patient can improve their movement.

In this chapter, instead of focusing on the standard assessment score, our method is developed to assess the abnormality of the movement during a prescribed exercise. The norm of an exercise will not be defined by an expert knowledge but a collection of the movement from healthy subjects.

In this scenario, it is difficult to collect all kinds of possible anomalous movements from unhealthy people. On the other hand, it is much easier to collect data from general healthy people. Therefore, this problem can be classified as a semi-supervised anomaly detection problem [106] where only normal movements are used to build the model.

There are a few standard methods to model the norm and detect an anomaly. The simplest way is to look at each dimension of the data point independently and model each dimension with a mean and a standard deviation. They form a confident interval which indicates the degree of being an outlier in a dimension. For a cyclic movement data, time normalization is usually done before applying this method [91]. However, by ignoring the interaction between dimensions in the data, it could miss other types of anomalies that are not just the range of motion at the specific phase.

One way to detect an anomaly in the interactions between dimensions is to use Principal Component Analysis (PCA) [106]. This linear tool can be used to project high-dimensional data to a lower dimensional space. Then, the distance between

the reconstruction of the projection and the original point (PCA reconstruction error) can indicate if the data point violates the correlation structure of the normal population. To add more flexibility to the method, Kernel PCA is introduced as a non-linear extension of PCA [107].

The non-linearity in anomaly detection can also be applied through autoencoder [108]–[110]. Autoencoder is a kind of neural network that learns to reconstruct the input with some dimensional bottlenecks at some hidden layers. This limitation forces the autoencoder to learn to reconstruct the training data with high accuracy but cannot reconstruct other unseen patterns effectively. This could be used to measure the similarity of the new data against the norm (the training data).

Another type of anomaly detection techniques is called one-class Support Vector Machine [111]. The method projects the training data to a higher dimensional space with kernel functions. Then, the data and the origin in the feature space will be separated by a decision boundary which is a hyperplane. This plane is optimized to be farthest from the origin.

6.1 Anomaly Detection Method

The aim is to make every step of data pre-processing and training fully automated. These steps are designed to be invariant to all repetitive exercises. Therefore, a specific feature will not be designed just to detect an anomaly in an exercise. Instead, a set of general upper-body kinematic features from Section 4.5 will be used.

From a perspective of a medical person who wants to add a new exercise prescription to the system, the pose tracking algorithm described in Chapter 4 will be used to record that repetitive exercise from multiple healthy subjects. Without any need of manual pre-processing or pre-segmentation, the system will automatically build the model of that exercise. Then, this model can be deployed to detect anomalous movements from the actual rehabilitation exercise session.

The method detects the anomaly from two main clues.

1. The fluctuation in the phase progression
2. The deviation of kinematic features from the healthy norm

The method consists of two stages. The first stage is the modeling stage in which all the parameters (e.g., neural network weights and biases for phase extraction, dimensionality reduction matrices from principal component analysis, reconstruction

Figure 6.1: A data flow diagram of all the key steps in the anomaly detection. An anomaly can be detected by either an abnormality in a period waveform (flow on the left side) or a fluctuation of the extracted phase (flow on the right side).

error normalization parameters) will be calculated from records of unsegmented repeated normal movement. The second stage is the detection of anomalies in the actual exercise sequence which can be seen in Figure 6.1. This section will explain through all the components in this diagram.

6.1.1 Feature Extraction

This study focuses on repetitive unilateral gross upper-limb movements. Therefore, the selected features are limited to the upper torso, the right upper arm, and the right forearm. These features are

- right upper-arm pointing direction (3D unit vector)
- right forearm pointing direction (3D unit vector)
- right forearm pronation/supination angle
- upper torso left/right turning angle
- upper torso frontal flexion

- upper torso lateral flexion
- right wrist position (X component)
- right wrist position (Y component)
- right wrist position (Z component)

The forearm and upper-arm pointing direction are represented in the upper-torso reference frame. The wrist position is represented in the *core* reference frames. The details of these feature extractions and reference frames are described in Section 4.5.

6.1.2 Phase Learning & Extraction

The next step is to extract a sequence of phase from a multi-dimensional time-series sequence of the pose features.

The preprocessing procedures from Section 5.3.2 and the learning method from Section 5.1 are used to learn a way to transform a small time-window of data of all features into a phase value. The training is done in an unsupervised manner without the need of phase label or pre-segmentation in the training data. After the training, the weights and biases of the neural network are used for phase extraction via a standard feed-forwarding method. The network outputs a phase value represented by a point on a unit circle (x, y) .

In order to simplify the phase representation for illustration purpose in this chapter, it can be converted into 1D representation of phase by $\phi = \arctan 2(y, x) \% (2\pi)$. Since the *arctan2* operator has its range of $(-\pi, \pi]$, the modulo operator (%) is used to turn it to a positive range of $[0, 2\pi)$.

The output from the phase extraction is necessary for period segmentation and the anomaly detection in phase progression.

6.1.3 Phase Sequence Smoothing

Sometimes, noise and error from the pose extraction step could lead to an unreasonable phase warp. To mitigate this problem, the series of extracted phase (2D representation) will be smoothed by a simple moving average filter following by a normalization to adjust it back on the unit circle.

6.1.4 Monotonically Progressing Phase Assignment using Circular Dynamic Time Warping (CDTW)

By observation, the series of phase extracted from a healthy movement always progresses in the forward direction (i.e., counter-clockwise direction on a unit circle). A minor exception occurs only when the movement pauses. A pause can generate a tiny fluctuation around the same spot in the extracted phase sequence.

This observation leads to a key assumption, “if a series of phase in a period of movement is deviating from the closest monotonically progressing phase series, the source movement is also deviating from the normal movement”. According to the assumption, we need to find the closest monotonically progressing series and compare them to the extracted phase. Notice that the phrase *monotonically progressing* is used instead of *monotonically increasing* to agree with the wrapping nature of phase. If a phase that almost reaches 2π progresses forward, it will wrap around and start from zero again.

In order to search for the exactly closest monotonically progressing series, it requires too much computational power and slows down the process. Therefore, an approximation method is developed by limiting the value of the series to be discrete and evenly spread from 0 to 2π .

This problem can be solved by our proposed variance of Dynamic Time Warping (DTW) algorithm called *Circular Dynamic Time Warping (CDTW)*. The original DTW is designed to compare between two finite sequences that have similar waveform from the beginning to the end but may have different time-warping structure. It assumes both finite sequences to have the same beginning (at 0% progress) and the same ending (at 100% progress).

In our situation, there are two sequences to compare. The first sequence is the unsegmented smoothed phase sequence. This sequence can start and end at any phase value (from 0 to 2π). In between, it is likely to have continuous phase progression (ramps up to 2π , wraps to 0, and keep repeating) or some fluctuated phase progressions in the case of an anomaly. The number of rounds progressing in this sequence is unknown. The second sequence (template sequence), in the case of original DTW, should be a monotonically progressing sequence which starts and ends with the same phase as the first sequence, and it should contain the same number of progressing rounds in between to match perfectly with the first sequence. However, in the case of CDTW, because the number of repetition in the first sequence is unknown, the second sequence is replaced with a single period from 0 to 2π and

Figure 6.2: Comparison of the data structure (table of minimum accumulated distance) used in the original DTW and Circular DTW. Each black square represents a matching point between one sample in a template sequence and one sample in an unsegmented smoothed phase sequence (output of Algorithm 2). The original DTW is impractical as the proper number of rows is not known before the calculation. The Circular DTW lets one finite template sequence to be reused for multiple rounds as needed by bridging the first and the last row of the table.

the sequence is modified to have a wrapping point from 2π to 0.

This modification changes the structure of the minimum distance table in the original DTW algorithm from a rectangle to a cylinder tube. The length of the first sequence becomes the length of the tube. The length of the template sequence becomes the circumference of the tube. In every round of progress made in the first sequence, the warping result will go around the tube one time. Figure 6.2 illustrates the differences between the table of DTW and CDTW. Algorithm 1 describes how the table of CDTW is calculated given that distance function between two phase values is defined as

$$D(\phi_a, \phi_b) = \begin{cases} |(\phi_a - \phi_b) \% (2\pi)|, & \text{if } (\phi_a - \phi_b) \% (2\pi) \leq \pi \\ |(\phi_a - \phi_b) \% (2\pi) - 2\pi|, & \text{otherwise} \end{cases} \quad (6.1)$$

and *circularTemplate* is an array with length N_p where the i -th element contain value $2\pi i / N_p$ (Figure 6.3). The *circularTemplate* is the template sequence for our CDTW

Figure 6.3: The template sequence used in cyclic dynamic time warping (*CircularTemplate*). The range of 0 to 2π is evenly discretized to N_p steps. In this plot, $N_p = 50$.

calculation. This sequence is used to associate an index (0 to N_p-1) with its phase value.

After the table is filled with Algorithm 1, Algorithm 2 is used to trace back from the last column to the first column to find how the time is warped throughout the smoothed phase sequence. From the *timeWarpingPath* output, any specific column j will be matched with one or more consecutive rows. Given that i_{mid} is the index at the middle of those consecutive rows, the closest monotonically progressing phase assignment at the j -th sample is $2\pi i_{mid}/N_p$. An example of this assignment can be seen in Figure 6.4.

6.1.5 Anomaly Score from Phase Progression

The approximately closest monotonically progressing (ACMP) phase sequence is compared to the original extracted phase sequence using Eq.6.1. The magnitude of the resulting sequence is used to infer anomaly in phase progression.

If the movement is too different from the training data for the phase extraction network, it is likely that the phase output from the neural network will fluctuate and create movements in both increasing and decreasing direction. Therefore, this measure should be able to detect some types of anomaly.

Algorithm 1 CDTW: Table filling

Input: circularTemplate[0.. N_p-1], phaseSequence[0.. $m-1$]

Output: table[0.. N_p-1 , 0.. $m-1$]

```
1: Define d(i,j) as  $D(\text{circularTemplate}[i], \text{phaseSequence}[j])$ 
2: for i := 0 to  $N_p-1$  do
3:   table[i,0] := d(i,0)
4: end for
5: for j := 1 to  $m-1$  do
6:   for i := 0 to  $N_p-1$  do
7:     table[i,j] := d(i,j) + Min(table[i,j-1], table[(i-1)% $N_p$ , j-1])
8:   end for
9:   iMin := IndexOfMinimum(table[:,j]) //from j-th column
10:  for iRun := iMin+1 to iMin+ $N_p$  do
11:    i := iRun% $N_p$ 
12:    iPre := (iRun-1)% $N_p$ 
13:    table[i,j] := Min(table[i,j], table[iPre,j] + d(i,j))
14:  end for
15: end for
16: return table
```

Algorithm 2 CDTW: Table backtracing

Input: table[0.. N_p-1 , 0.. $m-1$]

Output: timeWarpingPath (list of tuples)

```
1: j := m-1 //start from last column
2: i := IndexOfMinimum(table[:,m-1])
3: append (i,j) to timeWarpingPath
4: while j > 0 do
5:   compare table[(i-1)% $N_p$ , j-1], table[(i-1)% $N_p$ , j], table[i,j-1]
6:   if table[(i-1)% $N_p$ , j-1] is the minimum then
7:     i := (i-1)% $N_p$ 
8:     j := j-1
9:   else if table[(i-1)% $N_p$ , j] is the minimum then
10:    i := (i-1)% $N_p$ 
11:   else if table[i,j-1] is the minimum then
12:     j := j-1
13:   end if
14:   append (i,j) to timeWarpingPath
15: end while
16: return timeWarpingPath
```

Figure 6.4: **Top:** An extracted phase sequence. **Middle:** Smoothed phase sequence. **Bottom:** Approximately closest monotonically progressing (ACMP) phase sequence (black) overlaid on top of the extracted phase sequence (light blue). The differences between them are obvious when the extracted phase is fluctuating.

6.1.6 Period Segmentation

Every point in the approximately closest monotonically progressing (ACMP) phase sequence that the value crosses 2π to a new round is considered as a separator between two consecutive periods. This segmentation breaks a long continuous exercise sequence into multiple sequences with unequal lengths. Each segmented sequence is expected to contain one round of exercise.

This time segmentation into each individual movement period is important because the following anomaly detection technique requires the availability of a whole movement period.

6.1.7 Detect Anomaly with Feature Reconstruction Error

Principal Component Analysis (PCA) is used as another core engine for anomaly detection. The idea is to reduce the dimension of a period to a lower dimensional space and reconstruct it back to the original form of the data to be compared for the error. However, the length of each segmented period is not always the same. There-

Figure 6.5: Multiple stages of data processing to calculate reconstruction error of a period. The selected kinematic features are described in Section 4.5. The conversion of the features to the AZEQ form is described in Section 5.3.2.

fore, a segmented period has to go through a few data processing stages (Figure 6.5) to make it has a fixed length (N_p) and become compatible with PCA.

Segment Standardization

First, all features (13 dimensions per sample) from Section 6.1.1 will be converted to the AZEQ form (11 dimensions per sample) by a method described in Section 5.3.2.

At this point, the length of one segment can be arbitrary. The time axis needs to be warped to change the segment length to a constant value of N_p . The time mapping result from CDTW (Section 6.1.4) is used to determine how the time is warping. If multiple consecutive samples from the original sequence are warped to the same slot in the standard length, the data will be averaged.

Now, we got a sequence of 11-dimensional vectors with the standard length N_p .

A training set of an exercise will generate a collection of such sequence. All period from the training data will be used to calculate standardization parameters which are $11N_p$ sets of means and standard deviations. For a new data, its sequence will be element-wise subtracted by the means and divided by the standard deviations. This standardization aims to make all the $11N_p$ dimensions of a period to have zero mean and unit variance for the PCA modeling and reconstruction.

PCA Reconstruction

A period with a total dimension of $11N_p$ will be separated into nine channels. Each channel will be flattened to a vector and reconstructed independently. Each channel corresponds to a feature in the following list.

1. Right upper-arm pointing direction (two dimensions per sample)
2. Right forearm pointing direction (two dimensions per sample)
3. Right forearm pronation/supination (one dimensions per sample)
4. Upper torso left/right turning (one dimensions per sample)
5. Upper torso frontal flexion (one dimensions per sample)
6. Upper torso lateral flexion (one dimensions per sample)
7. X-component of wrist position (one dimensions per sample)
8. Y-component of wrist position (one dimensions per sample)
9. Z-component of wrist position (one dimensions per sample)

If $11N_p$ dimensions are all together fed into a single PCA, an accidental correlation across feature channels and subsequent unreasonable reconstructions will appear. This problem usually occurs when the number of subjects in the training data is not very large. For practicality, the system should not require too many healthy subjects to build a model of an exercise prescription. Therefore, each feature channel is forced to reconstruct independently to keep the required number of subject low (10-12 subjects for our experiments).

Figure 6.6: Reconstruction of forearm pronation data channel during a segmented period. **Left:** A case which the movement is normal. The reconstruction will be similar to the recorded data. **Right:** A case which an anomaly occurs and the reconstruction does not align well.

Dimensionality Reduction Matrix: A channel of a period will be flattened to a N_c -dimensional vector (with $N_c = 2N_p$ or N_p depending on the channel). The standard Principal Component Analysis (PCA) is done to extract all the N_c principal components from the training data. The smallest subset of the principal components that explains at least 95% of the variance in the training data will be used to form a dimensionality reduction matrix (U_r) for that channel. If N_d components are selected, the matrix U_r will be N_d -by- N_c in size.

Reconstruction of the Standardized Form: Let $\mathbf{c}$ be a flattened N_c -dimensional vector from a channel of a period and let U_r be a dimensionality reduction matrix of the channel. The reconstruction of the vector $\mathbf{c}$ can be calculated by

$$\hat{\mathbf{c}} = U_r^\top U_r \mathbf{c} \quad (6.2)$$

From the right to the left, the first matrix multiplication is done to project a vector to a lower dimensional space (N_d dimensions). Then, the second multiplication is to reconstruct it back to N_c dimensions.

Reconstruction Errors: According to the right side of Figure 6.5, the channel reconstructions will be combined in the standardized form and will be converted all the way back to the original kinematic features. Here, every sample in the original data will have a reconstructed value to compare. Some examples of the reconstruction are shown in Figure 6.6.

Within the 13-dimensional vector of a sample, there are six dimensions from the upper arm pointing direction and the forearm pointing direction that lie on the

surface of the unit sphere. A reconstruction error (1D) is calculated as the angle between the recorded pointing direction (3D) and the reconstructed pointing direction (3D). Therefore, these six dimensions will generate two dimensions of the reconstruction error. The other seven dimensions will generate seven dimensions of the reconstruction error per sample calculated by element-wise absolute difference. In total, the reconstruction error of a sample will have nine dimensions.

Error Normalization Parameters

The CDTW map can be used to associate a data sample back to a phase slot (among N_p slots of the standard length sequence). Therefore, N_p bins are created to collect all the reconstruction errors (9-dimensional vectors) from the training sample associated with it. For each bin, a 9-dimensional vector of the RMS errors will be calculated and used as error normalization parameters. In total, there will be $9N_p$ parameters.

Normalized Reconstruction Errors

The reconstruction error of a period ($9N_p$ values in total) will be divided by the $9N_p$ normalization parameters. This is done to scale the errors of all values to a similar range. The normalized reconstruction errors in separated features allow the algorithm to detect anomalies at the feature level. When the reconstruction error peaks at a specific feature channel, the monitoring therapist gets a clue of what could be wrong in that movement section.

6.2 Quantitative Experiment on Healthy Subjects

The purpose of this experiment is to quantitatively evaluate the proposed anomaly detection method. This experiment has to be conducted on healthy subjects' movement with simulated anomalies because those anomalous sections can be labeled accurately for evaluation purpose. This anomaly labeling cannot be done on the actual anomaly from unhealthy subjects as it is difficult to draw a line on the non-discrete nature of anomaly.

In addition, this experiment also observes the effect of motion capture method on the anomaly detection accuracy. The comparison between a marker-based motion capture system (Qualisys), the Kinect V2 alone, and our motion capture system

(Chapter 4) can give an idea for the possibility of using our motion capture system as a more compact alternative of a marker-based system in rehabilitation applications.

6.2.1 Evaluation Method

Twelve healthy subjects (five males, seven females, 25-59 years old. see Appendix B for subject details) are recruited to perform the following four repetitive exercises.

- Rectangular drawing (**RECT**) : A subject slides his right hand to four spots on a table to draw a rectangle.
- Hair combing (**COMB**) : A subject starts with his hand on a table and lifts the hand to comb his hair from the top to the back of his head and puts the hand back on the table.
- Tap in a loop (**TAP**) : A subject starts with his right hand on a table and raises the hand to touch his nose and left shoulder. Then, puts the hand back on the table.
- Water pouring (**POUR**) : A subject holds an empty cup and perform a pouring action to a bucket on the table and moves the hand back to the starting position.

For the training data, 100 repetitions per exercise per subject are recorded. All subjects are asked to mix slow, medium, and fast paces to their movements for data variations. Those variations are mixed in the record without any label. The speeds of the movements across multiple subjects do not need to be controlled to be the same.

For the testing data, 40 normal repetitions per exercise per subject are recorded with six additional repetitions that contain simulated anomalies. Those six anomalous movements are designed according to our expectation about the physical limitations of actual stroke patients.

- **Simulated Anomalies for RECT:** 1) Try to move the fist by moving torso instead of arm joints. 2) Draw the rectangle with the palm facing to the left side instead of facing down (half supination). 3) Do not reach three corners of the rectangle in an undershooting way because of the limited range of motion. 4) Overshoot at two specific corners of the rectangle. 5) Add zigzag/shaky pattern to the trajectory. 6) Add jerky pattern to the trajectory.

- **Simulated Anomalies for COMB:** 1) Move the head down halfway to meet the hand. 2) Tilt torso sideways to help in lifting the right hand. 3) Move elbow to the shoulder level first before moving the hand up from the table. 4) Lift the hand to only jaw level and put it down. 5) Add zigzag/shaky pattern to the trajectory. 6) Add jerky pattern to the trajectory.
- **Simulated Anomalies for TAP:** 1) Move the nose down halfway to meet the thumb in the first tap. 2) Tilt torso sideways to help in lifting the right hand. 3) Move elbow to the shoulder level first before moving the wrist up from the table. 4) Overshoot and miss the nose and the shoulder. 5) Add zigzag/shaky pattern to the trajectory. 6) Add jerky pattern to the trajectory.
- **Simulated Anomalies for POUR:** 1) Transport the cup to the space above the bucket but do not pronate the forearm. 2) Try not to use arm joints but use torso to accomplish the task. 3) Move elbow to the shoulder level first before moving the hand up from the table. 4) Overshoot beyond the bucket and adjust back to the position above the bucket. 5) Add zigzag/shaky pattern to the trajectory. 6) Add jerky pattern to the trajectory.

Each anomalous event is labeled by marking the beginning and the end of the event for evaluation purpose only.

To test the generalization capability for the subject outside the training data, the six-fold cross-validation is used for the evaluation. In each round of validation, the normal data from 10 subjects will be used for training and the mixed data from the other two subjects will be used for testing. The evaluation is repeated six times with different partitionings to cover all 12 subjects in the testing step.

The testing output in a record will initially appear in two forms for any sample. The first one is the anomaly score (one dimension) from phase progression. One threshold is needed to flag anomalies from this measure. The second one is the anomaly score (nine dimensions) from nine feature channels. Since this measure is already normalized, one threshold will be used against all nine dimensions. The absolute value of all normalized reconstruction errors will be compared against this threshold to flag anomalies. If at least one sample in an anomaly event is flagged, that anomaly will be counted as detected (true positive). If at least one sample in a normal period is flagged, the period will be counted as a false anomaly (false positive).

To evaluate the capability of these proposed measures in distinguishing between normal and abnormal movements, a receiver operating characteristic (ROC) curve

will be plotted out by applying a grid of possible configurations of the two thresholds. Then, the area under the curve (AUC) is used to evaluate the anomaly detection method. If AUC is equal to one, it means that the anomaly can be perfectly separated from the normal movement.

For the phase training, all the parameters are set to be the same as Section 5.3.3. For the time warping and PCA reconstruction, we set $N_p = 50$.

In addition, the whole evaluation process is repeated three times with a variation in the source of input features. On the first round, the kinematic features are extracted from the proposed motion capture system with Kinect V2 and available IMUs. On the second round, the features are solely extracted from data provided by Kinect SDK. Without information from an accelerometer attached to the Kinect, the *FloorClipPlane* property from the SDK is used to estimate the direction of the gravity for the calculation of three upper-torso-related features. On the third round, the features are solely extracted from a multi-camera marker-based motion capture system (Qualisys). The system contains 16 infrared cameras. Plug-in Gait marker placement configuration [89] is chosen. All three motion capture systems recorded the subject movements simultaneously. This comparison between three sources of features is done to observe the effect of the proposed motion capture method in the anomaly detection task.

6.2.2 Results: Receiver Operating Characteristic (ROC) Curve

The overall result can be summarized in 12 ROC curves shown in Figure 6.7. For all four exercises, the ROC curves of our motion capture system are able to reach the zone of accuracy where the true positive rate is greater than 90% and the false positive rate is lesser than 10% at the same time. The method results in a near-perfect degree of separation on TAP and COMB exercises with the area under the curve (AUC) of 0.9985 and 0.9924 respectively. However, the performance slightly drops in the case of the POUR and RECT exercises with AUC of 0.9827 and 0.9753 respectively. For each exercise, a point on the ROC curve that maximizes its true positive rate subtracted by its false negative rate will be selected. These optimal points are associated with a pair of thresholds that generates its accuracy. These thresholds are used to draw horizontal gray dashed lines on the plots from this experiment. The AUC and the details of those optimal points can be seen in Table 6.1.

According to Figure 6.7, for all four exercises, the trends of the difference from three motion capture systems are the same. Data from Kinect SDK alone perform the worst. The data from the proposed motion capture system have improved the ROC curve and the area under the curve (AUC) to be comparable to the marker-based motion capture system. This result shows that the accuracy enhancement from the proposed motion capture system (Kinect + IMU) can provide benefits for the proposed anomaly detection system.

Table 6.1: Area under the curve (AUC) from four exercises and the accuracies at the optimal point. The optimal point is the point where the true positive rate subtracted by the false positive rate is maximized (the orange points in Figure 6.7).

Exercise	Feature source	AUC	Optimal point	
			True positive rate	False positive rate
RECT	Kinect SDK Alone	0.9299	0.9028	0.1948
	Proposed system	0.9753	0.9444	0.0803
	Marer-based system	0.9851	0.9028	0.0105
COMB	Kinect SDK Alone	0.9624	0.8904	0.0538
	Proposed system	0.9924	0.9589	0.0499
	Marer-based system	0.9928	0.9589	0.0483
TAP	Kinect SDK Alone	0.9733	0.9583	0.0954
	Proposed system	0.9985	0.9722	0.0132
	Marer-based system	0.9994	1.0000	0.0132
POUR	Kinect SDK Alone	0.9282	0.7222	0.0160
	Proposed system	0.9827	0.9306	0.0475
	Marer-based system	0.9882	0.9444	0.0453

Figure 6.7: Receiver Operating Characteristic (ROC) curve from four exercises (RECT, COMB, TAP, and POUR) retrieved from the six-fold cross-validation on 12 healthy subjects. By enhancing the accuracy of the Kinect skeleton tracking with our motion capture method, the accuracy of anomaly detection task is also improved to the level that is comparable to the marker-based motion capture system. Each orange point marks the point on the curve where the true positive rate subtracted by the false positive rate is maximized. Zoom in the digital version for more details.

6.2.3 Results: Dimensionality Bottleneck

As the dimensionality bottleneck is chosen by the minimum number of the principal components that can explain at least 95% of the variance in the training data, this number has shown an important difference between three sources of feature extraction. According to Table 6.2, the need of a much larger number of dimensions for the Kinect SDK source alone reflects the noisier nature of the source. Especially for those forearm-related and wrist-related features, having the IMU on the user's wrist has largely improved the forearm tracking quality and reduced the needed number of dimensions by two to five times.

On the other hand, the proposed tracking method needs around two to four times more dimensions for the torso turning and the torso lateral flexion feature. These results align well with the tracking results where the proposed tracking contains some jitter in the torso while the Kinect skeleton shows less of it.

The marker-based motion capture system always uses the smallest amount of dimensions at the PCA bottleneck in all feature channels. This is due to the high-accuracy nature of the marker-based system.

Table 6.2: Average numbers of principal components needed to explain 95% of variance in one period of each feature channel. Each value is an average from the six folds of the cross-validation. The numbers in the parentheses are the range (maximum - minimum) from the six folds of the cross-validation. Notice that the number of dimensions needed to explain 95% of variance in the data is highest in the case of Kinect SDK alone. This number reduces to about a half for the proposed motion capture. This number reduces to about a quarter for the marker-based motion capture system.

Feature [#Dimension]	RECT			COMB			TAP			POUR		
	Kinect Alone	Kinect +IMU	Marker- based	Kinect Alone	Kinect +IMU	Marker- based	Kinect Alone	Kinect +IMU	Marker- based	Kinect Alone	Kinect +IMU	Marker- based
Upper arm direction [100]	17.5 (3)	15.3 (3)	8.2 (2)	20.0 (5)	12.8 (3)	8.2 (1)	26.7 (8)	16.5 (1)	7.7 (1)	28.5 (1)	13.0 (2)	6.0 (0)
Forearm direction [100]	37.5 (4)	9.8 (2)	6.7 (1)	38.3 (5)	13.0 (0)	10.8 (2)	40.5 (1)	14.5 (1)	10.7 (2)	47.2 (5)	9.3 (1)	6.7 (2)
Torso frontal flexion [50]	2.8 (1)	3.3 (2)	2.3 (1)	2.3 (5)	6.8 (2)	2.5 (1)	1.8 (1)	11.2 (4)	1.2 (1)	2.5 (1)	3.3 (1)	1.5 (1)
Torso left/right turn [50]	7.7 (7)	9.5 (6)	3.7 (1)	3.2 (1)	7.5 (2)	2.3 (1)	3.8 (1)	10.0 (3)	1.8 (1)	2.2 (1)	8.2 (6)	2.0 (0)
Torso lateral flexion [50]	3.0 (0)	11.7 (3)	3.0 (0)	2.3 (4)	12.8 (3)	3.0 (0)	2.7 (1)	11.7 (2)	2.0 (0)	4.2 (1)	8.8 (2)	2.8 (1)
Forearm pronation [50]	29.8 (3)	3.2 (1)	2.2 (1)	38.8 (1)	6.5 (1)	6.0 (2)	38.7 (1)	5.3 (1)	4.3 (1)	33.8 (2)	6.2 (1)	4.8 (1)
Wrist X [50]	25.8 (4)	10.7 (1)	7.8 (2)	20.0 (4)	6.8 (1)	6.0 (0)	28.8 (3)	7.7 (1)	6.8 (1)	33.7 (3)	8.0 (2)	6.2 (1)
Wrist Y [50]	33.3 (1)	13.2 (2)	2.0 (2)	16.7 (4)	9.0 (0)	8.3 (1)	28.3 (4)	8.0 (2)	7.3 (1)	23.2 (2)	8.2 (2)	6.8 (1)
Wrist Z [50]	24.2 (2)	6.7 (1)	4.2 (1)	12.0 (4)	6.7 (1)	5.5 (1)	23.7 (4)	7.7 (2)	5.0 (0)	29.0 (4)	4.2 (1)	3.3 (1)
Total (average)	181.7	83.3	40.0	153.7	82.0	52.7	195.0	92.5	46.8	204.2	69.2	40.2
Dimension percentage after reduction	36.3%	16.7%	8.0%	30.7%	16.4%	10.5%	39.0%	18.5%	9.4%	40.8%	13.8%	8.0%

6.2.4 Examples of Detected Anomalies

According to the proposed method, an anomalous period could be detected by having a high value in the normalized reconstruction error or having a high phase fluctuation score. In most of the cases, it detects an anomaly from both measures such as the results shown in Figure 6.8. It is noticeable that not all the features are prominent in an anomaly event, but only a few of them do. This gives quite specific clues of what a therapist should pay attention to in that short segment of an exercise record.

The color code used in all the normalize reconstruction error plots are

- orange for upper arm pointing direction (UA Pnt.)
- red for forearm pointing direction (FA Pnt.)
- pink for torso frontal flexion (TS Ffx.)
- gray for torso left/right turn (TS Turn)
- lime green for torso lateral flexion (TS Lfx.)
- dark cyan for forearm pronation (FA Prn.)
- light blue for wrist position on left-right axis (WR X)
- black for wrist height (WR Y)
- violet for wrist position on front-back axis (WR Z).

All the plots are produced as vector images. They can be zoomed in for more details in the digital version.

Figure 6.8: Anomaly detection result from sequences of COMB and RECT exercises with six simulated anomaly events. These anomalous areas (highlighted in yellow) are manually labeled. For each sequence, **the lower plot** shows the extracted phase (without smoothing), and the approximately closest monotonically progressing (ACMP) phase sequence from CDTW algorithm. The difference between them (red) is the anomaly scores from the phase fluctuation. For each sequence, **the upper plot** shows the normalized reconstruction errors from different kinematic feature channels in different colors. Those are upper arm pointing direction (UA Pnt.), forearm pointing direction (FA Pnt.), torso frontal flexion (TS Ffx.), torso left/right turn (TS Turn), torso lateral flexion (TS Lfx.), forearm pronation (FA Prn.), wrist position on left-right axis (WR X), wrist height (WR Y), and wrist position on front-back axis (WR Z). The plots of the same kind in this chapter also use the same color code. Zoom in the digital version for more details.

Figure 6.9: Anomaly plots from a COMB exercise sequence. The subject was asked to use his torso to compensate in lifting the right arm (the highlighted section). While the detection from the phase fluctuation is not clear, the reconstruction errors of torso left/right turning (TS Turn, gray) jump up clearly during such period. Moreover, the wrist position on the left/right axis (WR X, light blue) is also detected as abnormal during the simulated anomaly.

In some cases, the anomaly does not cause much fluctuation in the extracted phase and is detected only by the feature reconstruction error. An example of these cases is shown in Figure 6.9.

On the other hand, there are cases where the anomaly only occurs in the phase progression. Figure 6.10 illustrates the case where an anomaly is not detected by the reconstruction error but is detected by the phase fluctuation.

Figure 6.10: Anomaly plots from a POUR exercise sequence. The subject was asked to do the pouring action without the forearm pronation. Although the reconstruction error in the anomaly area looks relatively high to the surrounding, those peaks are not large enough to touch the threshold. However, the fluctuation of the phase progression is clear enough to touch the threshold.

Leaking of period reconstruction error is a common artifact that can be seen in any sequence of reconstruction errors. An example is shown in Figure 6.11. These leaks happen because the reconstruction error is calculated per segmented period. If a part of the segmented period gets involved in an anomaly event, it can cause the reconstruction of that period to corrupt any frame in the same period. This is not a problem as every period that overlaps with an anomaly event will also be labeled as abnormal in this evaluation.

The approximation of the closest monotonically progressing (ACMP) phase sequence is an important component for the period segmentation and phase fluctuation calculation. With the proposed CDTW algorithm, the monotonically progressing phase sequence can be extracted robustly even under large fluctuations in phase sequence (Figure 6.12).

Figure 6.11: Anomaly plots from a TAP exercise sequence with some leaks of period reconstruction error. Notice that the high reconstruction error can leak out from an actual anomaly section to its neighbor area but only within the limit of its one-period boundary.

Figure 6.12: Anomaly plots from a POUR exercise sequence with high phase fluctuation. The subject was asked to add zigzag and jerky movements to her trajectory in the first and the second highlighted section respectively. Notice that the proposed CDTW algorithm is able to retrieve the approximately closest monotonically progressing (ACMP) phase sequence for a reasonable period segmentation despite the considerable fluctuation in the original extracted phase sequence.

Figure 6.13: Anomaly plots from a COMB exercise sequence with undetected anomalies. The subject was asked to add zigzag and jerky pattern to the first and the second highlighted section respectively. This is found to be too subtle in some subjects as the amplitude of those vibrations is not instructed. However, the normalized reconstruction errors in the highlighted areas are relatively higher than the surrounding and can be seen by manual inspection.

6.2.5 Undetected Anomaly

Some anomalous movements are not detected as they might accidentally align with the direction of a principal component. Some simulated anomalies such as tremors or jerks done by some subjects are too small to be detected (Figure 6.13). However, those could be seen manually from the plots with a careful inspection.

Figure 6.14: Anomaly plots from a POUR exercise sequence with false anomaly detection. The false anomalies are detected due to the reconstruction error of wrist height (WR Y, black). By comparing his movements to other healthy movements in the training data, his style is quite unique. His highest wrist height during the transportation to the bucket is about 10 cm higher than his wrist height above the bucket. This difference in the training data is just about 1 cm on average as the common strategy is to be more efficient and to lift the hand just to a necessary height. This kind of style difference can cause false positive in this evaluation.

6.2.6 False Anomaly (False Positive)

False anomalies can be caused by two issues. The first one is the large errors from skeleton tracking failure which rarely happens. The second one is about the personal style of movement which is more common. Sometimes, the healthy movements from an unseen subject (testing group) can be different from any of the training data in a way that the learned model cannot extrapolate to. An example of this style issue can be seen in Figure 6.14.

6.3 Qualitative Experiment on Unhealthy Subjects

6.3.1 Evaluation Method

Training Data from Healthy Subjects

The same set of four exercises are used. Unlike the evaluation method in Section 6.2.1, all the 100 repetitions (without simulated anomaly) per exercise per subject from all 12 healthy subjects are now used as training data.

Testing Data from Unhealthy Subjects

Fourteen stroke survivors (six males, eight females, 44-86 years old) with their right arm affected are recruited to perform those four exercises. All subjects, except one, are over six months post-stroke. Their Fugl-Meyer gross upper limb scores range from 19 to 41 while the full score is 42. Each subject is encourage to perform at least 20 repetitions per exercise if possible. However, their movements do not need to be perfect as we want to test the capability of the anomaly detection method for those cases. The details of the subjects and the inclusion/exclusion criteria can be seen in Appendix B.2.2.

6.3.2 Results & Discussions

Because of the non-binary nature of the actual anomaly, there is no clear way to manually label those anomalies as we did to the simulated anomaly. Therefore, this experiment focuses more on the qualitative analysis of the results.

The color code used in all the normalize reconstruction error plots are

- orange for upper arm pointing direction (UA Pnt.)
- red for forearm pointing direction (FA Pnt.)
- pink for torso frontal flexion (TS Ffx.)
- gray for torso left/right turn (TS Turn)
- lime green for torso lateral flexion (TS Lfx.)
- dark cyan for forearm pronation (FA Prn.)
- light blue for wrist position on left-right axis (WR X)

Figure 6.15: Anomaly plots from a RECT exercise sequence by an unhealthy subject STV. The exercise is done with ease. Notice that his pace is gradually faster overtime.

- black for wrist height (WR Y)
- violet for wrist position on front-back axis (WR Z).

All the plots are produced as vector images. They can be zoomed in for more details in the digital version.

RECT Exercise

RECT exercise is the easiest exercise among the four as the subjects do not need to lift the hand and resist the gravity. For a highly recovered subject, the resulting plots are as clean as the plots from healthy movements. Figure 6.15 shows an example from STV who can do the exercise without any difficulties.

Examples of anomalies seen in RECT exercise are shown in Figure 6.16 and 6.17. An anomaly could be detected when the movement trajectory is disturbed. A subject might lift her hand to adjust the plastic sliding plate underneath the hand (Figure 6.16). A subject may temporarily get confused about the direction of the movement and mix forward and backward direction (Figure 6.16). A subject might not have good control of his arm and generate a wavy path in some side of the rectangle (Figure 6.17). A subject excessively leans his torso forward to compensate for his limited elbow extension (Figure 6.17). Some types of anomaly can exist in every repetition of a subject but some types occur occasionally.

Figure 6.16: Anomaly plots from a RECT exercise sequence by an unhealthy subject KKT. She performs the exercise well in general but there are two areas that draw our attention. The first highlighted section is when her hand slips out from the plastic plate underneath her hand and she has to lift her hand to adjust it. The second highlighted section is when she gets confused and mixes forward and backward movement.

Figure 6.17: Anomaly plots from a RECT exercise sequence by an unhealthy subject LSL. His movement contains high reconstruction errors of the wrist position on the left-right axis (WR X, light blue) and the torso frontal flexion (TS Ffx., pink) in every repetition. The first issue occurs when he pulled his hand back closer to his body. He draws a wavy line instead of a straight line. The second issue is caused by his limited elbow extension. In order to reach the farthest corner of the rectangle, he has to excessively lean his torso forward.

Figure 6.18: Anomaly plots from a COMB exercise sequence by an unhealthy subject LKL. The extracted phase coverage is small and contains large phase jumps as the subject cannot complete the full movement cycle. This cause the smoothed phase sequence to be largely different from the original phase sequence. As a result, the comparison between the monotonically progressing phase sequence and the extracted phase provides large differences around the area of the phase jump.

COMB Exercise

COMB exercise is the most difficult exercise among the four as it requires a wide shoulder range of motion and enough strength to lift the hand against gravity. Some subjects might not be able to perform the full coverage of the COMB exercise. Some subjects raise their hands up to about the head level and the put their hands down without moving it above and behind the head. This cause a sudden jump in the phase extraction which results in the high difference between the extracted phase and the approximately closest monotonically progressing (ACMP) phase sequence. Example of phase skips can be seen in Figure 6.18 and 6.19.

Some other anomalies for COMB exercise includes hesitation (Figure 6.19 and 6.20), unusual hand trajectory (Figure 6.20), and tremors (Figure 6.21).

Figure 6.19: Anomaly plots from a COMB exercise sequence by an unhealthy subject TKB. The first and the last highlighted section are the moments when she hesitates in the middle of the movement and moves back a bit before continuing. The third highlighted section is an example of when she lifts her hand up to about the nose level and put it down. The act of skipping the section of hand at the head level causing phase to skip around 0.8π rad within one frame and the peak from the phase anomaly score is quite high. The second highlighted section is a more obvious case. She lifts her hand to only shoulder level and put it down. This case causes a phase jump that is larger than 1π . As a result, the CDTW is unable to see this section as two periods but as one period with high phase fluctuation in the middle.

Figure 6.20: Anomaly plots from a COMB exercise sequence by an unhealthy subject CMT. The first highlighted segment is a short pause at the midpoint between the table and the head which is an unusual pausing point. For the second and the third highlighted segments, the high reconstruction errors and the phase fluctuations are caused by an abnormal path of movement from the back of the head to the table. She moves her hand to the space in front of her belly first before moving to the table. However, in some repetitions (non-highlighted area), she did it normally without receiving any external feedback.

Figure 6.21: Anomaly plots from a COMB exercise sequence by an unhealthy subject NAH. This subject needs to put considerable effort to complete the full cycle of the COMB exercise. He takes a much longer time to complete one cycle of the movement and his movement contain a lot of arm shaking especially when the hand is above his head.

TAP Exercise

Abnormal patterns found in TAP exercise include inefficient hand trajectory (Figure 6.22, excessive elbow lift (Figure 6.23), arm shaking (Figure 6.24), torso leaning, and confusion of tapping order (Figure 6.25).

An interesting phenomenon observed in this exercise is the convergence towards a more efficient and common movement style after a number of repetitions without any external feedback. This can be seen among subjects who are almost fully recovered. An example of this case can be seen in Figure 6.22.

Figure 6.22: Anomaly plots from a TAP exercise sequence by an unhealthy subject CST. Those high reconstruction errors in the highlighted segment are caused by the inefficient hand trajectory from her shoulder to the table. Instead of moving the hand down directly to the table as healthy subjects do, her hand path curves up to the nose level before touching down on the table. After she repeats this exercise for a few iterations, this anomaly gradually disappears and her movements become normal at the end. This learning process can happen by just repeating the exercise without any external feedback.

Figure 6.23: Anomaly plots from a TAP exercise sequence by an unhealthy subject LSL. The reconstruction error from upper arm pointing direction (UA Pnt., orange) is high in all of his TAP repetitions. During the transportation phase from table to the nose, he lifts his elbow sideways to about the same level as his hand which is unusual.

Figure 6.24: Anomaly plots from a TAP exercise sequence by an unhealthy subject NRH. Her arm constantly shakes whenever the hand is lifted against gravity and has no anchor point to push against. This shaking kicks the wrist position out from the proper trajectory sometimes. In addition, her forearm pronation keeps rolling back and forth throughout the exercise resulting in high reconstruction errors (FA Prn., dark cyan).

Figure 6.25: Anomaly plots from a TAP exercise sequence by an unhealthy subject NHC. The first and the second highlighted segments consist of movements with wrong tapping order. Instead of moving to the nose first, she sometimes moves to the shoulder first. The third highlighted segment focuses on the torso lateral flexion (TS Lfx, lime green). She leans her torso to the left and the right side with the range from -5 to +12 degrees in one repetition which is larger than the healthy norm which ranges from -5 to +5 degrees.

Figure 6.26: Anomaly plots from a POUR exercise sequence by an unhealthy subject SGH. The level of anomaly score from this subject is low and the phase sequence is quite consistent throughout the record.

POUR Exercise

Most of the anomalies in POUR exercise is related to how high the subject lifts the cup during the transporting phase and the leaning of the torso. Some examples of the results are in Figure 6.26, 6.27, and 6.28.

Figure 6.27: Anomaly plots from a POUR exercise sequence by an unhealthy subject QSL. Her condition causes her to lean to her left side more than healthy subjects do even in a resting posture. Therefore, the excessive torso lateral flexion (TS Lfx., lime green) is detected in this case.

Figure 6.28: Anomaly plots from a POUR exercise sequence by an unhealthy subject NAV. He cannot extend his wrist to stay in a natural resting position. It causes his hand to always hang underneath the wrist level when he lifts the forearm. He has to compensate by lifting the wrist up to about 10-15 cm higher than healthy subjects' average. When the lift is too high, it is shown in the wrist height reconstruction error (WR Y, black). Phase fluctuations around the phase range of 1.0π to 1.6π that happen in most of his repetitions are caused by a small dip when the hand is above the bucket. By observation, a normal healthy subject usually lifts their hands just high enough to pour to the bucket without the dipping for the hand height adjustment. The long pause around the 500th frame is the moment when the cup is slipping out of his palm and he takes some time to adjust it.

How will it help a therapist in the monitoring an exercise?

By skimming through these kinds of plots in an exercise session, it is quite easy to quickly choose the best and the worst repetitions performed by a patient in a specific session. The therapist only needs to see the video record of those few repetitions per exercise to get an idea of what feedback he should give to the patient. Moreover, by looking at the extracted phase sequence, the therapist can get the idea of how difficult the exercise is to a patient from how often he takes a break and how fast the phase progress.

6.4 Conclusion

We have designed a statistical method to model an arbitrary repetitive exercise of unilateral gross upper limb movement for assisting a therapist in anomaly detection and feedback generation. The method is designed to learn from a small group of healthy subjects to create a model that later will be applied to the actual exercise from stroke patients. The method is evaluated quantitatively on healthy subjects and labeled simulated anomalies on four repetitive exercises. The method shows good potential in discriminating anomalous movements from normal ones with an average area under the curve (AUC) of 0.9872 using our proposed motion capture system which is close to the average AUC from the marker-based system at 0.9939. This result shows the possibility of replacing the marker-based motion capture system with our motion capture system which is more compact and affordable for community or home-based rehabilitation with a small sacrifice in the accuracy. An additional quantitative evaluation on stroke survivors is also conducted to observe the generalization of the method towards the target population.

Chapter 7

Rigid Object Tracking Using a Deep Signed Distance Network

While a number of vision-based free-hand tracking techniques has become mature enough to be commercialized in applications mainly in virtual reality and augmented reality products such as Leap Motion [112] and Fingo [113], the ability to track the hands while they are interacting with an arbitrary object is still a challenging research topic due to the occlusion and the additional ambiguity from the object. In the field of rehabilitation, a functional movement of distal upper limb parts *i.e.*, the wrist, the hand, and the fingers, usually comes with an objective of manipulating something. It is almost unavoidable that an object will be in the scene with a rehabilitative patient practicing the use of his hand. If the object is not tracked properly, it could blend in easily with some hand parts and prevent the fine details of the hand to be extracted. On the other hand, if the boundary of the tracked object is accurately known in the 3D space, the information could be used to constrain the hand pose for better hand tracking [114]. As the project is moving towards the hand tracking in the future, some learning techniques for the object representation in the tracking problem are explored in this chapter.

There are multiple approaches to achieve 3D tracking such as the use of inertial sensors or fiducial markers attached to the object. However, the readings of inertial sensors may drift with time and the marker-based approach can be intrusive. To overcome those limitations, vision-based tracking offers solutions that are non-invasive, practical, and cheap [115].

Since the introduction of commodity depth cameras, the availability of depth data extends RGB tracking methods by utilizing depth information in particle filter algorithm [116], [117], Gaussian filter algorithm [118], and also the well-established Iterative Closest Point (ICP) algorithm [119], [120]. The ICP aims to find the best

pose estimate to minimize the distance between two sets of depth data and while there are several variants of ICP with more efficient and robust solutions [121], [122], the main process of searching for point correspondences can be computationally expensive and error-prone.

Thus, to avoid the time-consuming point-to-point correspondence search, depth data could be modeled implicitly to allow point-to-model distance minimization [123]. For example, a minimal set of primitive shapes is used to model a simple industrial part [124], but for a more complex model, implicit functions such as implicit B-Spline can be employed to provide a richer data representation for a better registration [125]. The signed distance function (SDF) is another representation that implicitly encodes 3D surfaces and can be used directly to define a cost function for accurate registration [126]. As such, SDF has been applied in recent works on scene reconstruction [127], [128], camera tracking [127]–[129], and rigid object tracking [130]. The SDF of basic geometric shapes such as spheres, cubes, and ellipsoids can be represented implicitly. For example, the SDF of a sphere of radius r centred at the origin can be written using the implicit expression $\sqrt{x^2 + y^2 + z^2} - r$.

However, the SDF of more complex shapes are much more difficult to define and are thus often represented as sampled volumes [131]. To obtain a continuous representation of a complex object surface, there are a few works that incorporate machine learning techniques. For instance, Radial Basis Function (RBF) neural network can be used to classify a 3D point into three classes, namely internal, on-surface, and external [132]. To date, the use of a multi-layer neural network to model the SDF of an object has only been done for brain structures segmentation [133]. To the best of our knowledge, our work is the first to model the SDF of an object using a deep neural network for object tracking purpose.

Recently, learning-based methods have revolutionized many areas of computer vision including object tracking. Since Tan and Ilic [134] have proposed a random-forest-based method to regress the movement of the object from the change in the observed point cloud, several improvements [135]–[137] have been made to advance the state-of-the-arts in temporal 3D object tracking without GPU. On the other hand, Garon and Lalonde [138] claimed to develop the first end-to-end deep learning for temporal object tracking. However, it needs GPU for real-time tracking due to the large network size.

The contribution of this chapter is the adoption of a learning-based method to train a deep neural network, which we term as SDF-Net, to approximate the SDF of an object. In addition, we also propose two methods to utilize the trained SDF-Net

for rigid object tracking. Furthermore, a quantitative comparison on a public dataset is carried out to compare our approach against the state-of-the-arts. Our methods are also tested with real depth data from two different commodity depth cameras to demonstrate the real-time object tracking capability in different scenarios.

This chapter is organized as follows. Section 7.1 details the methodology to train a deep neural network that models the SDF of an object and the two different ways to use the network for object tracking. Section 7.2 outlines the evaluation methods and Section 7.3 discusses the results. Finally, Section 7.6 presents the conclusion, as well as the limitation and future work.

7.1 Rigid Object Tracking Method

Our goal is to estimate the current 6-DOF pose θ_t , which contains an orientation R_t and a translation t_t , of the given object in the camera reference frame $\{C\}$.

A depth camera is used to provide a sequence of depth images. For each depth frame, the following inputs are used in our method:

1. The depth image of the current frame t and the previous frame $t - 1$.
2. Pose estimation results from the previous frames $(\theta_{t-h}, \dots, \theta_{t-1})$ where $h > 1$

Also, the method has access to the camera intrinsic parameters and the triangular mesh model of the target object in the object reference frame $\{H\}$.

7.1.1 Method Overview

The SDF of an object is simply a function that takes a 3D point $\mathbf{p}$ and returns a signed Euclidean distance to the closest point on the object surface. In this chapter, the signed distance is defined to be negative when $\mathbf{p}$ is inside the object, and positive when $\mathbf{p}$ is outside the object. The intuition behind our fitting approaches is to use SDF in a form of trained neural network to guide the pose update in moving sampled points from the observation towards zero signed distance value. This idea is illustrated in Figure 7.1.

Our method can be divided into two stages. The first stage is to prepare an approximation of the SDF of the object by training a neural network, while the second stage utilizes the network for pixel sampling and pose tracking. Two different methods of pose tracking are proposed. The first method is based on the conventional ICP approach with an adaptation at the correspondence search. The second method is

Figure 7.1: **Left:** The model contain a continuous signed-distance field. The contour line of zero (blue line) is the place where object surface should be. The signed distance is negative inside the contour and positive outside the contour. **Middle:** When the surface from the observation (yellow line) is not aligned on the blue contour, the signed-distance field and its gradient will be used to push the observed surface towards the nearest point on the zero contour (green arrows). **Right:** When the observation aligns with the model, the system will reach an equilibrium and the pose become stable.

based on an optimization approach. Both methods share the same sparse sampling mechanism which is designed to increase the robustness of tracking when there is an occlusion. Nonetheless, both methods depend heavily on the quality of learned SDF-Net.

7.1.2 Building a Signed Distance Network

Training Data Preparation.

The 3D model of the object to be tracked is translated to make its centroid stays at the origin. To facilitate the learning process, the model is scaled with a scaling factor $s = 1/d$, where d is the maximum diameter of the object; one unit in this scaled world $\{S\}$ is equivalent to d .

Three sets of 100,000 points, namely A_I , A_S and A_O , are sampled from the region inside the object, on the surface, and outside the object respectively. The points in A_O are only sampled from the space outside object up to 1.5 units from the surface. The space is divided equally into 100 sections with a thickness of 0.015 units each. One thousand points are then randomly sampled from each section to ensure that the training data obtained in A_O are evenly distributed.

Let $K : \mathbb{R}^3 \mapsto \mathbb{R}^3$ maps a 3D position to the closest point on the object surface and $\Phi' : \mathbb{R}^3 \mapsto \mathbb{R}^3$ be the expected gradient that returns a unit direction, which points towards the closest point on the surface when the input lies inside the object. In contrast, the direction will point away from the closest point on the surface if the input falls outside the object. All the associated expected gradients and closest

surface mapping at all points in A_I and A_O are pre-calculated.

Network Structure.

Our neural network has three input nodes for a 3D position and one output node for the signed distance. The hidden layers are fully-connected. All the hidden nodes use tanh activation function, except for the output node which uses linear activation function.

In this study, two different diamond-shaped networks are tested. Network $\mathcal{A}$ is a smaller network with 10 hidden layers, while network $\mathcal{B}$ is a bigger network with 12 hidden layers. The number of nodes in all layers are 3-6-9-12-15-18-15-12-9-6-3-1 for network $\mathcal{A}$ and 3-6-9-12-15-18-21-18-15-12-9-6-3-1 for network $\mathcal{B}$, with total tunable parameters of 1,369 and 2,162 respectively. The diamond structure allows a gradual projection of the 3D data to a higher dimensional space before being slowly reduced to one dimension. Given the same number of tunable parameters, shallower networks with equal numbers of hidden nodes for all layers seems to be less effective in learning.

Cost Function.

Let Θ represent all weights and biases of a target network. Function $N_\Theta : \mathbb{R}^3 \mapsto \mathbb{R}$ does a feed-forwarding that maps a 3D position (input layer) to a single number at the output node. Function $N'_\Theta : \mathbb{R}^3 \mapsto \mathbb{R}^3$ maps a 3D position to the gradient of N_Θ at that spot. The cost function is defined as:

$$\begin{aligned} Cost(\Theta) = & \left(\frac{1000}{s^2|A_S|} \sum_{\mathbf{p}_i \in A_S} N_\Theta(\mathbf{p}_i)^2 \right) \\ & + \left(\frac{1000}{s|A_O|} \sum_{\mathbf{p}_i \in A_O} \max(0, -N_\Theta(\mathbf{p}_i)) \right) + \left(\frac{1000}{s|A_I|} \sum_{\mathbf{p}_i \in A_I} \max(0, N_\Theta(\mathbf{p}_i)) \right) \\ & + \left(\frac{1}{|A_I| + |A_O|} \sum_{\mathbf{p}_i \in A_I \cup A_O} \|N'_\Theta(\mathbf{p}_i) - \Phi'(\mathbf{p}_i)\| + \|N'_\Theta(K(\mathbf{p}_i)) - \Phi'(\mathbf{p}_i)\| \right) \end{aligned} \quad (7.1)$$

The first (*SurfaceDistancePenalty*) term penalizes the network when some outputs at surface points deviate from zero. The second term penalizes the network when some outputs at outsider points are negative. The third term penalizes the network when some outputs at insider points are positive. Since the gradient is applied directly in our tracking methods, it is also considered in the training. Thus, the last (*GradientPenalty*) term is introduced to penalize the deviation of gradient from the

expectation. This term also applies gradient constraints on the object surface using the same expected gradient from its correspondence.

All the weights and biases (Θ) are trained to minimize the cost function in Eq.7.1. The training is done on TensorFlow [100] using ADAM optimizer [101] with a learning rate of 0.001 for 100,000 iterations.

Notation.

After the training, the network is ready to be used in the scaled reference frame $\{S\}$. To make the notation more compact, the network is augmented so that it can work in the object reference frame $\{H\}$ and output the signed distance that relates to the real world scale. We define our learned signed distance function $D : \mathbb{R}^3 \mapsto \mathbb{R}$ and its gradient function $\mathbf{G} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ as

$$D({}_H\mathbf{p}) = N_{\Theta}(s \cdot {}_H\mathbf{p})/s \quad (7.2)$$

$$\mathbf{G}({}_H\mathbf{p}) = N'_{\Theta}(s \cdot {}_H\mathbf{p}) \quad (7.3)$$

given that ${}_H\mathbf{p}$ is the input position in the object reference frame $\{H\}$. The function D and $\mathbf{G}$ are implemented in a closed-form using the standard feed-forwarding and back-propagation algorithm. This technique allows the computation to be vectorized and to run efficiently on a CPU with SIMD capability.

The state of orientation R_t and translation $\mathbf{t}_t$ represents the transformation of the object with respect to the camera reference frame at time t :

$${}_C\mathbf{p} = R_t {}_H\mathbf{p} + \mathbf{t}_t \quad (7.4)$$

7.1.3 Current Frame Pose Prediction

At the current frame t , when the θ_t is not yet calculated, the predicted pose $\hat{\theta}_t$ will be calculated using a short series of pose estimation results from the previous frames $(\theta_{t-h}, \dots, \theta_{t-1})$. The predicted translation $\hat{\mathbf{t}}_t$ and the predicted orientation $\hat{R}_t$ are calculated independently. For translation, a weighted linear regression is used to extrapolate $\hat{\mathbf{t}}_t$ with weights $v_1, \dots, v_h$. For orientation, all the past orientations are expressed in quaternions. Then, we predict the current orientation $\hat{q}_t$ with the fol-

lowing equation:

$$\hat{q}_t = \vartheta \left(\sum_{i=1}^{h-1} w_i \cdot (q_{t-1} \times (\sigma_{t-2}(t-1-i, t-1))^{-1} \times q_{t-1}) \right) \quad (7.5)$$

given that $\sigma_c(a, b)$ is a quaternion spherical linear interpolation (Slerp) between q_a and q_b at frame c . The function $\vartheta(\cdot)$ represents quaternion normalization. The weights $v_1, \dots, v_h$ and $w_1, \dots, w_{h-1}$ allow the adjustment of the responsiveness of the prediction. An incremental geometric series 2^{i-1} , with $i = 1, 2, \dots, h$, is used for both weight series.

7.1.4 Object Pixel Sampling

Among all the depth pixels in the current frame, a number of pixels will be sampled and used in object tracking. If a few non-object pixels are sampled to fit with the object, it could reduce the tracking accuracy and lead to a loss of tracking eventually. Therefore, it is important to ensure that all the pixels used are sampled from the object surface.

As the pose estimation from the previous frame ($\theta_{t-1} = [R_{t-1}, \mathbf{t}_{t-1}]$) is known, we can transform the 3D associated position of every pixel (from the previous frame) into the object reference frame and use the learned SDF to classify whether the pixel belongs to the tracked object. If $D(R_{t-1}^T({}_C\mathbf{p}_i - \mathbf{t}_{t-1}))$ is less than a small positive value (e.g. 0.004 m), the pixel i of previous frame will be classified as object surface. Otherwise, it will be classified as non-object.

To accelerate this process, we introduce a classification interval k . Then, we only consider pixels every k -th row and k -th column with distance from $\mathbf{t}_{t-1}$ less than the maximum object radius r_o . All the classified object and non-object points are kept in the object reference frame as ${}_H\mathbf{p}_i = R_{t-1}^T({}_C\mathbf{p}_i - \mathbf{t}_{t-1})$.

In the current frame t , the collected non-object points ${}_H\mathbf{p}_i$ from the previous frame will be transformed by the previous pose θ_{t-1} and the predicted pose $\hat{\theta}_t$ separately to represent both non-object points from the previous frame and predicted non-object points in the current frame. The transformed points are then projected onto the image plane. When a transformed non-object point is projected to a pixel, that pixel and its neighbor within Chebyshev distance of k will record the shallowest depth from all non-object projections. All these records can be considered as potential occluders, meaning that any point in the volume behind them should not be sampled.

Then, the collected object points from the previous frame will be transformed by θ_{t-1} and $\hat{\theta}_t$ and then projected to the depth image. If the projected depth is at least 10 cm shallower than the occluder at that pixel and if the current observed depth is within 5 cm from the projected depth, we consider that pixel to be safe to sample. Among those survivals, m pixels will be sampled randomly.

The rationale of this method is to cover both static occlusions and those which move together with the object such as hand and fingers. Therefore, θ_{t-1} and $\hat{\theta}_t$ are used to represent the two kinds of occlusions. Moreover, the expansion of the occlusions after projection will give some margins for the error in pose prediction and unpredicted movements of those occlusions.

7.1.5 ICP-based Fitting Approach

This approach is similar to the well-established ICP except for the correspondence searching step. Instead of finding the exact closest point on the object surface, the trained SDF is used to infer the correspondence. The latest state of the pose $\tilde{\theta} = [\tilde{R}, \tilde{\mathbf{t}}]$, initialized as $\tilde{\theta} := [R_{t-1}, \hat{\mathbf{t}}_t]$, will be updated iteratively until a stopping condition is met.

Pseudo-correspondence Inference.

Given ${}_H\mathbf{p}_i = \tilde{R}^T({}_C\mathbf{p}_i - \tilde{\mathbf{t}})$, with $i = 1, 2, \dots, m$, as the transformed sampled 3D points from Section 7.1.4 using the latest state of the pose $\tilde{\theta}$, their correspondences will be

$${}_H\mathbf{e}_i = {}_H\mathbf{p}_i - D({}_H\mathbf{p}_i) \cdot \mathcal{N}(\mathbf{G}({}_H\mathbf{p}_i)) \quad (7.6)$$

where $\mathcal{N}(\cdot)$ is the vector normalization.

Pose Update.

Let ${}_H\mathbf{c}_p$ be the centroid location calculated from all ${}_H\mathbf{p}_i$ and ${}_H\mathbf{c}_e$ be the centroid location calculated from all ${}_H\mathbf{e}_i$. The optimal orientation ΔR can be found by minimizing the sum of pair-wise distances:

$$E(\Delta R) = \sum_{i=1}^m \|({}_H\mathbf{c}_e + \Delta R({}_H\mathbf{p}_i - {}_H\mathbf{c}_p)) - {}_H\mathbf{e}_i\|^2. \quad (7.7)$$

Let P be a matrix whose i -th column is a vector ${}_H\mathbf{p}_i - {}_H\mathbf{c}_p$ and Q be a matrix whose i -th column is a vector ${}_H\mathbf{e}_i - {}_H\mathbf{c}_e$. E is minimized by performing a singular value

decomposition (SVD) to the cross-covariance matrix $M = PQ^\top$ which results in $M = UWV^\top$. The rotation matrix is obtained from $\Delta R = VU^\top$. As a consequence, the optimal translation will be $\Delta \mathbf{t} = {}_H\mathbf{c}_e - \Delta R {}_H\mathbf{c}_p$.

While ΔR and $\Delta \mathbf{t}$ moves the sampled observation points to fit with the signed distance field in the object reference frame, the state $\tilde{R}$ and $\tilde{\mathbf{t}}$ perform the opposite. Therefore, an inversion will be applied to update the parameters in the following order:

$$\tilde{\mathbf{t}} := \tilde{\mathbf{t}} - \tilde{R}\Delta \mathbf{t} \quad \text{then} \quad \tilde{R} := \tilde{R}(\Delta R)^T \quad (7.8)$$

Stopping Condition.

The iteration will stop when the number of iterations has reached n_{max} or the update is small. For the latter criterion, we set the condition to be

$$(3 - \text{Trace}(\Delta R))/3 < 10^{-6} \quad \text{and} \quad \Delta \mathbf{t} < 10^{-5}. \quad (7.9)$$

Then, the latest state will be assigned to the final pose estimation $\theta_t := \tilde{\theta}$.

7.1.6 Optimization-based Fitting Approach

Cost Function.

This approach makes use of an optimization algorithm to perform the fitting for current frame t . In our application, the Levenberg-Marquardt algorithm (LM) is used to minimize a cost function. The cost function F and all m residues F_i are defined as

$$F(\Delta R, \Delta \mathbf{t}) = \sum_{i=1}^m F_i(\Delta R, \Delta \mathbf{t})^2 \quad (7.10)$$

$$F_i(\Delta R, \Delta \mathbf{t}) = D\left(\Delta R \tilde{R}^\top [{}_C\mathbf{p}_i - (\tilde{\mathbf{t}} + \Delta \mathbf{t})]\right) \quad (7.11)$$

where ΔR and $\Delta \mathbf{t}$ are the changes of orientation and translation respectively, and m is the number of sampled points used in the optimization.

In contrast to Section 7.1.5, the current state $\tilde{\theta} = [\tilde{R}, \tilde{\mathbf{t}}]$ is initialized as $\tilde{\theta} := [R_{t-1}, \mathbf{t}_{t-1}]$, while ΔR and $\Delta \mathbf{t}$ are initialized to be the identity rotation and the zero vector respectively.

Formulation of Jacobian.

For the cost function in Eq.7.10, the rotation ΔR is represented using a quaternion $\mathbf{q} = w + x\hat{i} + y\hat{j} + z\hat{k}$.

Given that $\omega = [x \ y \ z]^\top$, a point $\mathbf{p}$ rotated by the quaternion $\mathbf{q}$ is

$$\mathbf{p}_{\text{rotated}} = \mathbf{p} + 2w(\omega \times \mathbf{p}) + 2[\omega \times (\omega \times \mathbf{p})] \quad (7.12)$$

For each point ${}_C\mathbf{p}_i$, we define $\mathbf{p}'_i = \tilde{R}^\top [{}_C\mathbf{p}_i - (\tilde{\mathbf{t}} + \Delta\mathbf{t})]$, $D_i = D(\Delta R\mathbf{p}'_i)$ and $\mathbf{G}_i = \mathbf{G}(\Delta R\mathbf{p}'_i)$. All gradient components in the Jacobian J can be derived as follow:

$$\frac{\partial F_i}{\partial \Delta\mathbf{t}} = -\mathbf{G}_i^\top (\Delta R \tilde{R}^\top) \quad (7.13)$$

$$\frac{\partial F_i}{\partial w} = 2\mathbf{G}_i^\top ([\omega]_\times \mathbf{p}'_i) \quad (7.14)$$

$$\frac{\partial F_i}{\partial v} = 2\mathbf{G}_i^\top \left[(w[\mathbf{e}_v]_\times + [\mathbf{e}_v]_\times [\omega]_\times + [\omega]_\times [\mathbf{e}_v]_\times) \mathbf{p}'_i \right] \quad (7.15)$$

where $v \in \{x, y, z\}$, $\mathbf{e}_v$ are the respective standard basis vectors, and $[\cdot]_\times$ represents the skew symmetric matrix.

When ΔR and $\Delta\mathbf{t}$ are the identity rotation and the zero vector respectively, we have $\mathbf{p}'_i = \tilde{R}^\top ({}_C\mathbf{p}_i - \tilde{\mathbf{t}})$ and the Jacobian calculations are simplified to

$$\frac{\partial F_i}{\partial \Delta\mathbf{t}} = -\mathbf{G}_i^\top \tilde{R}^\top \quad (7.16)$$

$$\frac{\partial F_i}{\partial w} = 0 \quad (7.17)$$

$$\frac{\partial F_i}{\partial v} = 2\mathbf{G}_i^\top ([\mathbf{e}_v]_\times \mathbf{p}'_i) \quad (7.18)$$

Pose Update.

The change in pose parameters δ is computed from

$$[J^\top J + \lambda K] \delta = J^\top \mathbf{r} \quad (7.19)$$

where $\mathbf{r} = -[F_1 \ F_2 \ \dots \ F_m]^\top$, λ is the damping factor of the optimization. K can be chosen as the identity matrix or $\text{diag}(J^\top J)$.

Adding the respective components of δ to the zero translation and the identity quaternion, we have $\Delta\mathbf{t} = [\Delta t_x \ \Delta t_y \ \Delta t_z]^\top$ and $\Delta q = 1 + \Delta x\hat{i} + \Delta y\hat{j} + \Delta z\hat{k}$, where

the latter is normalized before being converted to the rotation matrix ΔR .

The current pose $\tilde{R}$ and $\tilde{\mathbf{t}}$ are updated with $\tilde{R} := \tilde{R}(\Delta R)^\top$ and $\tilde{\mathbf{t}} := \tilde{\mathbf{t}} + \Delta \mathbf{t}$. The terms ΔR and $\Delta \mathbf{t}$ are always reset to the identity rotation and the zero vector for the next iteration.

The optimization process is iterated until the cost function (Eq. 7.10), its change, or the magnitude of δ falls below their respective thresholds, or the number of iterations has reached n_{max} .

7.2 Evaluation

7.2.1 Quantitative Evaluation

Our proposed method is evaluated using synthetic dataset from Choi and Christensen [116] which consists of four RGB-D sequences with ground truth object trajectories. The parameters used in the ICP-based method are $m = 50$, $k = 5$, and $n_{max} = 20$. The parameters used in the LM-based method are $m = 30$, $k = 11$, $\lambda_{init} = 0.1$, $\lambda_{scaling} = 2$, $n_{max} = 20$, and $K = I$. The pose is initialized using the ground truth in the first frame for each sequence. The algorithms are run on a single CPU thread on Intel Core i7-4770 @ 3.4 GHz.

7.2.2 Qualitative Evaluation

The tracking methods are tested on two different real objects, namely the Stanford Bunny and a detergent bottle. The bunny is 3D-printed using the mesh model obtained from the Stanford 3D Scanning Repository [139]. Meanwhile, the detergent bottle is scanned using Microsoft Kinect V2 and *3D Scan* software. In the evaluation, we demonstrate our dynamic tracking with occlusions by nearby objects and by the hand holding the object. We have also tested the algorithms on two types of depth cameras: Intel RealSense SR300 that uses a structured-light sensor and Microsoft Kinect V2 that uses a time-of-flight (ToF) sensor.

7.3 Results & Discussion

7.3.1 The Trained Networks

Multiple Stages in the Training.

All the networks, despite the randomized weight initializations, have evolved automatically through a few stages in a surprisingly systematic way. During the initial few thousands of iterations, the network would flatten the value of the output node to a very small value as it tries to satisfy the *SurfaceDistancePenalty* term in Eq.7.1. As a side effect, all the gradient magnitudes in the whole volume would be almost close to zero. Most of the gradient errors in the *GradientPenalty* term will stay slightly below one as we expect all the gradients to have a magnitude of one. Next, the network will slowly shape the terrain to satisfy the *GradientPenalty* term while the surface points are already loosely nailed at the near zero output value.

Single Point Convergence.

To verify that the SDF-Net and its gradient can be used to guide any point in the surrounding region towards the object surface, we try to reconstruct the surface using random points sampled from the region. Each point will get updated by $\mathbf{p} := \mathbf{p} - 0.01\mathbf{G}(\mathbf{p}) \cdot \text{sign}(D(\mathbf{p}))$ iteratively until $D(\mathbf{p})$ passes a zero-crossing point. Then, the zero-crossing position is linearly interpolated from the two latest positions. The estimated zero-crossing points are collectively shown in Figure 7.2. All the random points are able to converge on the surface in a virtually straight trajectory without being trapped in any local minimum. Refer to the interactive results in the supplementary materials [140].

Surface Smoothing Error.

One inherent nature of the neural network is to smooth out noises in its training data. Thus, some small regions on the object surface may not be modeled accurately, especially protruding volumes such as the corner of a box or a deep concave surface that is too small. As they are small relative to the whole surface area, it should not largely affect the tracking.

Figure 7.2: Convergence of random points. **Grey cloud**: points on the actual model surface. **Red cloud**: points converged from random positions. **Green lines**: some examples of point travelling paths from a random position to the surface. 3D interactive interface of all trained network can be found in the supplementary repository [140].

7.3.2 Object Tracking

Quantitative Results.

For the ICP-based method, we obtain the best average accuracy with a 0.85 mm translation error and a 0.27° orientation error with an average processing time of 5.11 ms. The accuracy is better than the LM-based method, which has average accuracies of 0.93 mm and 0.48° for translation and rotation errors respectively. However, the LM-based method runs at a faster rate of 1.29 ms. In terms of memory usage, our method requires the least memory footprint among the state-of-the-arts at 9 kB or lower if a smaller network is used.

Among the tracking methods that can run at sub-degree accuracy with a real-time speed on CPU, most of them [135]–[137] are developed on a decision-forest-based regression method [134]. Some methods are learning-free [141], [142] but developed on a well-established PWP3D method [143] which relies heavily on the color data. Our new branch of methods has reached a similar level of accuracy and calculation time on one CPU core without relying on any decision forest or color-based method. Refer to Table 7.1 for detailed results.

According to Table 7.1, the translation errors perpendicular to the optical axis (t_x and t_y) are two to three times larger than the translation errors along the depth axis (t_z) in our methods and in Tan et al.’s method [135]. However, the drifts (t_x and t_y) are minimized in Tan et al.’s later method [137] as they utilize color to extract edge

points, which constrain the whole model from sliding along the plane perpendicular to the optical axis; hence, the overall error is reduced significantly.

The comparison between the ICP-based and the LM-based methods is presented in Table 7.2. The LM-based method is found to take much lesser time when compared with the ICP-method. This is because of the fast convergence rate due to the low damping factor λ , which results in lower iteration count. In addition, a sparser classification interval k and a lower number of sampled points m are used. This configuration greatly reduces the computational time without compromising much accuracy.

Table 7.2 also compares the result from the small and the large neural network. As expected, network $\mathcal{B}$ with more memory capacity has a better overall accuracy but requires additional 20% of calculation time when compared with network $\mathcal{A}$.

In terms of memory footprint, our representation is the most compact, with 6 kB and 9 kB for the network $\mathcal{A}$ and $\mathcal{B}$ respectively. With this size, a discrete representation of SDF can only represent a crude model of the surface.

Qualitative Results.

The results show that the object remains tracked even when more than half of it is occluded (Figure 7.3b,7.3f). Also, our methods are able to track the object when it is toppled and thrown (Figure 7.3d). Videos of our tracking results can be found in the supplementary repository [140].

When the object is static, the fitted model generally shows slight jitter when the ICP-based method is used; this is less likely to be seen for the LM-based method. This is because the LM algorithm always verifies if a particular update will lower the cost function (Eq.7.10), making it tend to stay at its pose from the previous frame; this feature is not present in the ICP-based method.

The comparison between the tracking results on both short-ranged and long-ranged cameras shows that the latter generally yields sub-par performance. This is mainly due to a larger depth error at the far distance together with the *multi-path effect* which only occurs in ToF camera.

After the fitting process is done in each frame, as a by-product, SDF-Net can be used to differentiate object and non-object pixels without the use of color. Hands and fingers on the object are also segmented out (Figure 7.3c,7.3f). This feature could be useful in hand tracking applications with object interaction.

Table 7.1: Benchmark comparison of the RMS errors in translation (mm), orientation (degree) and the runtime (ms) on a synthetic dataset [116]. The required memory footprints used to store pre-calculated content are approximated. The results from our two methods, ICP-based and LM-based, are from the bigger network $\mathcal{B}$.

		PCL	C&C	Krull	Tan15	Kehl	S.A	Tan17	Ours	
Error		[144]	[116]	[117]	[135]	[141]	[145]	[137]	ICP	LM
<i>Kinect Box</i>	t_x	43.99	1.84	0.83	1.54	0.76	0.30	0.15	1.27	1.12
	t_y	42.51	2.23	1.67	1.90	1.09	0.49	0.19	1.22	1.81
	t_z	55.89	1.36	0.79	0.34	0.38	0.31	0.09	0.50	0.56
	<i>Roll</i>	7.62	6.41	1.11	0.42	0.17	0.21	0.09	0.16	0.16
	<i>Pitch</i>	1.87	0.76	0.55	0.22	0.18	0.27	0.06	0.17	0.30
	<i>Yaw</i>	8.31	6.32	1.04	0.68	0.20	0.23	0.04	0.10	0.17
	<i>Time</i>	4539	166	143	1.5	8.10	1.71	2.2	5.62	1.37
<i>Milk</i>	t_x	13.38	0.93	0.51	1.23	0.64	0.63	0.09	0.98	1.01
	t_y	31.45	1.94	1.27	0.74	0.59	1.19	0.11	0.80	0.76
	t_z	26.09	1.09	0.62	0.24	0.24	0.48	0.08	0.32	0.36
	<i>Roll</i>	59.37	3.83	2.19	0.50	0.41	0.19	0.07	0.20	0.34
	<i>Pitch</i>	19.58	1.41	1.44	0.28	0.29	0.28	0.09	0.27	0.76
	<i>Yaw</i>	75.03	3.26	1.90	0.46	0.42	0.27	0.06	0.15	0.22
	<i>Time</i>	2205	134	135	1.5	8.54	1.70	2.1	4.99	1.26
<i>Orange Juice</i>	t_x	2.53	0.96	0.52	1.10	0.50	0.39	0.11	1.12	1.15
	t_y	2.20	1.44	0.74	0.94	0.69	0.37	0.09	0.88	0.96
	t_z	1.91	1.17	0.63	0.18	0.17	0.37	0.09	0.67	0.67
	<i>Roll</i>	85.81	1.32	1.28	0.35	0.12	0.12	0.08	0.41	0.44
	<i>Pitch</i>	42.12	0.75	1.08	0.24	0.20	0.17	0.08	0.25	0.65
	<i>Yaw</i>	46.37	1.39	1.20	0.37	0.19	0.15	0.08	0.39	0.65
	<i>Time</i>	1637	117	129	1.5	8.79	1.69	2.2	4.79	1.25
<i>Tide</i>	t_x	1.46	0.83	0.69	0.73	0.34	0.42	0.08	0.99	1.01
	t_y	2.25	1.37	0.81	0.56	0.49	0.51	0.09	1.03	1.05
	t_z	0.92	1.20	0.81	0.24	0.18	0.64	0.07	0.28	0.30
	<i>Roll</i>	5.15	1.78	2.10	0.31	0.15	0.22	0.05	0.09	0.24
	<i>Pitch</i>	2.13	1.09	1.38	0.25	0.39	0.29	0.12	0.54	1.03
	<i>Yaw</i>	2.98	1.13	1.27	0.34	0.37	0.30	0.05	0.13	0.34
	<i>Time</i>	2762	111	116	1.5	9.42	1.70	2.2	5.04	1.30
Mean	Transl.	18.72	1.36	0.82	0.81	0.51	0.50	0.10	0.85	0.93
	Rot.	29.70	2.45	1.38	0.37	0.26	0.22	0.07	0.27	0.48
	Time	2786	132	131	1.5	8.71	1.7	2.2	5.11	1.29
Requires	Mem.	-	-	-	7.4MB	10MB	123kB	13MB	9 kB	
	GPU	-	✓	✓	-	-	-	-	-	
	Color	✓	✓	✓	-	✓	-	✓	-	
	Depth	✓	✓	✓	✓	✓	✓	✓	✓

Figure 7.3: **a)** An example of fitting results from the dataset [116] with partial occlusion. **b-g)** Examples of post-fitting segmentation of objects with occlusions; green and red regions represent pixels classified as object and non-object points respectively. **g-h)** When the object is lifted abruptly, the tracking fails as the predicted pose $\hat{\theta}$ lags behind the actual observation, causing the failure in sampling some key features, e.g. the handle of the detergent, which are critical in determining the object pose.

Table 7.2: Comparison of our methods using two different networks on the synthetic dataset [116]. Note that the translation and orientation errors in this table are different from those in Table 7.1; we use the Euclidean distance and the angle difference between the estimated and the ground truth orientation to compute the errors as these measures are invariant under coordinate transformation.

Network	ICP-based method		LM-based method	
	$\mathcal{A}$ (smaller)	$\mathcal{B}$ (bigger)	$\mathcal{A}$ (smaller)	$\mathcal{B}$ (bigger)
Avg. translation error (mm)	1.62	1.49	1.70	1.55
Max. translation error (mm)	15.75	4.82	10.02	8.27
Avg. orientation error (deg)	0.43	0.33	0.77	0.74
Max. orientation error (deg)	3.59	3.06	6.80	4.30
Avg. Number of iterations	13.84	12.89	2.32	2.05
Time per frame (ms)	4.03	5.11	1.11	1.29

7.4 Limitation

As the proposed tracking method works by finding the transformation parameters between consecutive frames, the initial pose of the object must be provided. Also, in case of tracking loss, the object pose has to be reinitialized manually. Hence, initialization and detection of the object in real-time will be investigated in the future.

7.5 Towards Real-time Articulated Object Tracking

In the recent literature, there are multiple ways to model the signed distance field of articulated objects for pose tracking. A method proposed by Schmidt *et al.* [146] pre-compute SDF for each rigid segment in its local volume. For a given pose, those local SDF volumes will be transformed and overlap one another. Global SDF of a point is found by taking the minimal SDF from all overlapping volumes at the point. Taylor *et al.* [147] take a different approach using pre-computed SDF in the volume surrounding the whole object in the resting pose. This volume of SDF will be articulated together with the model using a mesh of tetrahedral volumes that forms a thick layer on the object skin. Both methods are carefully designed to allow a highly parallelizable implementation on a general purpose graphics processing unit (GPGPU). Still, their pre-calculated SDFs are in the form of a discrete grid with a limited resolution and need trilinear or tricubic interpolation to approximate a continuous SDF.

The signed distance function of an articulated object can also be seen as a dynamical field that the signed distance value at a 3D position will gradually change when a pose parameter gradually changes. Therefore, we believe that the SDF-Net could

Figure 7.4: Comparisons between SDF-Net and ASDF-Net. ASDF-Net has additional pose parameter nodes at the input layer.

be extended and used to learn how the field will change according to the changing pose parameters. This section presents an early design, an implementation, and a result of this idea for a full-body human tracking application.

7.5.1 Articulated Signed Distance Field Network (ASDF-Net)

The SDF-Net is simply modified by adding pose parameter nodes to the input layers of the neural network. Moreover, the number of hidden node in each hidden layer is 13 times larger to give more learning capacity to the network.

In this implementation, the full-body SMPL model is simplified to 28 DOFs with extra six DOF from global translation and orientation. All the pose parameters must be selected carefully to ensure a few things.

1. A small change in a pose parameter will not create a large jump in the signed distance value.
2. All possible poses should not generate any twist in a joint which can lead to a singularity in tracking (multiple different poses have the same or very similar SDF) or candy wrapper artifacts.

7.5.2 Data Generation and Training

With 28 DOF, it is not possible to generate all the combination of poses. Even the crudest sampling grid of two samples per DOF will take 2^{28} seconds or over eight years to generate. Therefore, the strategy is to keep generating new training data all the time while the training process is running. Both the data generation module and the training module have its own dedicated GPU to accelerate their own work.

The duty of the data generation module is to generate random poses, random points, and their signed distances to be expected at the output layer. It also has to screen out the pose that has self-penetration. It will generate a set of 500 random poses with 40,000 points per pose which will be sent to the training module every 10 epochs of training.

The training module will only keep 1,000 poses at a time for training. After 10 epochs, it will evaluate all the 1,000 poses and throw away 500 poses that it can fit well. Then, it will receive 500 new poses from the data generation module and mix them with 500 survival poses from the last round and start training again. With this strategy, the poses that are not well fitted by the network will stay longer in the training data.

Starting from a set of random weights, the training on an Nvidia GTX 1080 Ti GPU takes around 40 hours to converge to a workable state.

7.5.3 Notations

Similar to SDF-Net, a function $N_{\Theta} : \mathbb{R}^{28} \times \mathbb{R}^3 \mapsto \mathbb{R}$ does a feed-forwarding that maps a pose and a 3D position to a single number at the output node. To simplify the calculation, the scaling factor s is set to one. Therefore the signed distance function D is equivalent to N_{Θ} .

$$D(\mathfrak{X}, {}_H\mathbf{p}) = N_{\Theta}(\mathfrak{X}, {}_H\mathbf{p}) \quad (7.20)$$

The notation ${}_H\mathbf{p}$ is the input position in the object reference frame $\{H\}$ and $\mathfrak{X}$ is a 28-dimensional vector that represents a pose.

There are two types of gradient used in this method, pose gradient $\mathbf{G}_{\mathfrak{X}} : \mathbb{R}^{28} \times \mathbb{R}^3 \mapsto \mathbb{R}^{28}$ and position gradient $\mathbf{G}_{\mathbf{p}} : \mathbb{R}^{28} \times \mathbb{R}^3 \mapsto \mathbb{R}^3$.

$$\mathbf{G}_{\mathbf{p}}(\mathfrak{X}, {}_H\mathbf{p}) = \frac{\partial D}{\partial \mathbf{p}} \quad (7.21)$$

Figure 7.5: Convergence of random points in random poses. The converged surfaces overlap very well with the rendered surfaces even though it is very unlikely that those random poses have ever been sampled as a part of the training data. **Grey cloud:** points on the actual model surface rendered from the SMPL model. **Red cloud:** points converged from random positions. **Light blue cloud:** Starting random positions.

$$\mathbf{G}_{\mathfrak{X}}(\mathfrak{X}, {}_H\mathbf{p}) = \frac{\partial D}{\partial \mathfrak{X}} \quad (7.22)$$

These signed distance function and gradients can be calculated using a standard forward and backward propagation algorithm which can be highly parallelized on a GPU. Our implementation on a GPU (Nvidia GTX 1080) with CuBLAS library and CUDA kernels is about 100 times faster than a vectorized implementation with BLAS/LAPACK on one CPU thread (Intel Core i7-3770 @ 3.4GHz).

7.5.4 Network Validation

To validate that the network has learned the signed distance field of the articulated human body, the trained network is tested using the same technique of surface reconstruction from SDF-Net. A pose ($\mathfrak{X}$) is randomly sampled and kept constant. Four thousand 3D points are randomly sampled. Each point will get updated by $\mathbf{p} := \mathbf{p} - 0.01 \mathbf{G}_{\mathbf{p}}(\mathfrak{X}, \mathbf{p}) \cdot \text{sign}(D(\mathfrak{X}, \mathbf{p}))$ iteratively until $D(\mathfrak{X}, \mathbf{p})$ passes a zero-crossing point. Those zero crossing points from different random poses are shown in Figure 7.5. The neural network also learns the deformation of the surface surrounding each joint produced by the SMPL model.

7.5.5 Tracking Method

The tracking algorithm is similar to the LM-based technique used in the rigid object case (Section 7.1.6). The key difference is the additional states of 28 pose parameters on top of the global translation and orientation. Therefore, the current state becomes $\tilde{\theta} = [\tilde{\mathfrak{X}}, \tilde{R}, \tilde{t}]$ and the cost function to be optimized in a frame is modified from

Eq.7.10 and Eq.7.11 to

$$F(\Delta\mathfrak{X}, \Delta R, \Delta\mathbf{t}) = \sum_{i=1}^m F_i(\Delta\mathfrak{X}, \Delta R, \Delta\mathbf{t})^2 \quad (7.23)$$

$$F_i(\Delta\mathfrak{X}, \Delta R, \Delta\mathbf{t}) = D\left(\tilde{\mathfrak{X}} + \Delta\mathfrak{X}, \Delta R \tilde{R}^\top [{}_C\mathbf{p}_i - (\tilde{\mathbf{t}} + \Delta\mathbf{t})]\right) \quad (7.24)$$

The pose state $\tilde{\mathfrak{X}}$ is taken from the tracking result from the previous frame. For the first frame of the tracking, the tracked skeleton Kinect SDK is used to approximate the current state $\tilde{\theta}$. In each frame, $m = 512$ pixels will be sampled randomly from the human silhouette area retrieved from Kinect SDK.

The calculation of the Jacobian components for the global translation and the global orientation (Eq.7.16 - Eq.7.18) are the same with $\mathbf{G}_i = \mathbf{G}_p(\tilde{\mathfrak{X}}, \Delta R \mathbf{p}'_i)$.

The calculation of the Jacobian components for the pose update $\Delta\mathfrak{X}$ is

$$\frac{\partial F_i}{\partial \Delta\mathfrak{X}} = \mathbf{G}_x(\tilde{\mathfrak{X}}, \mathbf{p}'_i) \quad (7.25)$$

The optimization with LM iterations is done in the same way and the pose parameters are also updated by $\tilde{\mathfrak{X}} := \tilde{\mathfrak{X}} + \Delta\mathfrak{X}$. The iteration stops when the cost reduction from an iteration is under 2%.

7.5.6 Preliminary Evaluation

To evaluate the early stage of this method, the personalization method from Section 4.2 is used to find shape parameters of a subject to create an articulated SMPL model. The model is used to synthesize the training data to train an ASDF-Net. The number of nodes in all layers is 31-78-117-156-195-234-273-243-195-156-117-78-39-1. The trained network is used to track a few sequences of full-body movements recorded from Kinect V2.

7.5.7 Preliminary Results

Calculation Time.

The average calculation time is 15 ms/frame with the neural-network related functions D , $\mathbf{G}_x$, and $\mathbf{G}_p$ running on a GPU (Nvidia GTX 1080). As the current neural network architecture might not be optimal, this calculation could be faster with a smaller network.

Figure 7.6: Full-body tracking results using ASDF-Net. The yellow skeleton and the blue mesh are rendered from the final pose parameters. The model is able to follow the observed point cloud as expected.

Qualitative Tracking Result.

The pose parameters can converge to the point cloud as expected for slow and moderate speed movements. However, some frames of high-speed movements can create a point cloud region that is too far from the training. This situation makes the pose converge in a local minimum of an untrained region instead of a correct pose. Without a penalty for the unmatched surface normal between the observation and the model, an upper limb will sometimes converge by sticking one side of the model surface to the other side of the observed surface and produces a sequence of wrong results. Some results are shown in Figure 7.6. Videos of the tracking result are also available¹.

7.5.8 Future Research Direction

This simplest form of ASDF-Net is found to be capable of articulated object tracking. However, the robustness still needs to be improved. We plan to experiment with extra constraints such as joint range limitations, the use of surface normal, or the information from the inertial measurement units.

¹<https://goo.gl/z9qHQL>

7.6 Conclusion

It is shown that a deep neural network trained with our proposed method could be used to learn an approximated signed distance field of a rigid object which is accurate enough for an object tracking purpose. Our results have been shown to reach up to sub-millimeter and sub-degree accuracy when evaluated on a public dataset. The real-time capability of our rigid object tracking method has been demonstrated using the depth data from commodity depth cameras, and the algorithm could run on a single CPU thread.

If the network is scaled up to have a much larger capacity, it is able to learn the pose-dependent signed distance field of an articulated object with high DOF such as a whole human body. The algorithm can run in real-time as the neural network can be highly parallelized on a GPU. However, its robustness still needs to be improved.

Chapter 8

Conclusion

8.1 Contributions

An existing upper limb sensing system rehabilitation always has limitations in some ways that make the whole stack of implementation unscalable. From the motion capturing perspective, researchers usually use a marker-based multi-camera system for a highly accurate analysis of the movement, but the large required space and the difficult marker attachments prevent such system from being used at a patient home. An alternative solution such as Microsoft Kinect requires a practical amount of space, but the tracking capability is limited as it cannot reliably track forearm pronation/supination which is an important feature in upper limb rehabilitation. From the perspective of a therapist, giving feedback by looking through the whole exercise session is possible but not scalable. Among existing tele-rehabilitation platforms, none of them is able to automatically highlight the section of the upper limb exercise session that should be inspected to improve the efficiency of feedback generation. Moreover, the pool of prescribable movements in those platforms is limited as each of them needs to be preprogrammed specifically by an engineer of the platform. The therapists are not able to add a new patient-specific prescription as they need. Our contributions are a stack of techniques that work together to solve these issues. The contributions can be described as follows.

- In Chapter 3, we have designed and implemented a markerless sensing system that integrates an IMU-augmented depth camera with two wrist-worn IMUs which enable a reliable tracking of forearm orientation and the pronation/supination feature.
- In Chapter 4, we introduce a personalized marker-based upper-body motion

capture technique which improves the shoulder, elbow, and wrist tracking accuracy over the Kinect SDK 2.0. The tracking method also improves the robustness against occlusions due to the wrist-worn IMUs.

- In Chapter 5, an unsupervised technique to learn and extract phase from a repetitive movement is innovated as a key that will unlock the full automation in the exercise learning system for evaluation as it allows the training data to be unsegmented and unlabelled.
- In Chapter 6, the unsupervised phase extraction technique and the principal component analysis are utilized for automated modeling of repetitive exercises for automated anomaly score generation. Its flexibility allows a therapist to add a new exercise prescription to the system by just showing healthy examples to it.
- In Chapter 7, we discover a way to implicitly represent a signed distance field of a rigid object using just a deep neural network. The technique can be used in real-time object tracking from a depth camera requiring very low memory consumption. This contribution will pave a way toward the next stage of the project where the sensing of hand-object interaction is needed.

8.2 Future Research Direction

To push this project closer towards the real-world application, the research directions are as follows.

- A deployment trial should be conducted to evaluate the effectiveness of therapists' feedback generation with and without assistance from our system. The effectiveness can be judged from the time spent on the feedback generation and the specificity of the feedback. For the developed techniques to be deployed effectively, it needs proper designs of graphical user interfaces together with a client-server infrastructure.
- The articulated signed distance function network (ASDF-Net) discovered in Chapter 7 could be improved and integrated into the human tracking method in Chapter 4 to accelerate the calculation to the level of real-time tracking. With the real-time speed and better accuracy than Kinect V2, it can support a wider range of applications which requires sudden feedback such as rehab games, and remote physical assessment during a video call.

- A long-term plan is to expand this capability towards the wrist, hand, and finger movements, and the object manipulation. The fully-articulated hand tracking capability will allow the system to model a more complete coverage of upper-limb functional tasks.

Circuit Design

Figure A.1: Schematic diagram page 1. Main microcontroller, wireless module, and FRAM.

Figure A.2: Schematic diagram page 2. Accelerometer, gyroscope, magnetometer, and barometer.

Figure A.3: Schematic diagram page 3. Battery charging module and voltage regulator.

Figure A.4: 3D-rendering of the designed PCB (front)

Figure A.5: 3D-rendering of the designed PCB (back)

Appendix B

Subject-Related Information

B.1 Subject Group 1

Six healthy human subjects are recruited for the data used in Chapter 4. These data collection was approved by Nanyang Technological University Institutional Review Board (IRB-2017-08-003).

Subject ID	Gender (M/F)	Height (cm)	Weight (kg)
MA	M	165	86
MB	M	177	64
MC	M	178	55
FA	F	160	61
FB	F	160	52
FC	F	161	53

B.2 Subject Group 2

Twelve healthy subjects and fourteen unhealthy subjects are recruited for the data used in Chapter 5 and 6. These data collection was approved by Nanyang Technological University Institutional Review Board (IRB-2018-03-036).

B.2.1 Healthy Subjects

Inclusion Criteria

- Ages above 21 years old
- Has no difficulty with upper-limb movements

Recruited Subject Details

Subject ID	Gender (M/F)	Age (year)	Height (cm)	Weight (kg)
AL	M	42	174	85
NT	F	25	161	60
ED	F	59	164	58
JK	M	30	178	69
SS	F	50	156	55
JY	M	27	175	70
LP	F	27	160	54
MJ	M	25	178	54
AR	F	29	160	58
PG	F	40	160	59
JX	M	31	171	68
YC	F	50	155	58

B.2.2 Unhealthy Subjects

Inclusion Criteria

- Ages above 21 years old
- Has right arm weakness due to stroke
- Has no difficulty in communication
- Can balance by himself/herself in a sitting pose without any form of assistance
- Does not experience pain in the right arm at rest and when moving
- Can touch their own head, nose, and shoulder using right arm without any assistance

Exclusion Criteria

- Cannot balance on a chair with a low backrest
- Has Fugl-Meyer upper-limb score under 20

Recruited Subject Details

The data in the following table are sorted by the gross upper-limb score of the Fugl-Meyer assessment. The gross upper-limb part of Fugl-Meyer does not include the wrist, hand, and finger assessments. All of the subjects are in the chronic stage (more than six months post-stroke) at the moment of the data collection except NAH. The weight and height of some subjects have not been retrieved.

Subject ID	Gender (M/F)	Age (year)	Height (cm)	Weight (kg)	Fugl-Meyer Score	
					Upper-limb (0-66)	Gross Upper-limb (0-42)
CMT	F	70	-	-	62	41
STV	M	63	169	61	61	40
CST	F	61	-	-	61	38
NHC	F	57	-	-	57	37
SGH	F	85	-	-	53	36
KKT	M	86	165	-	50	34
JSH	M	44	160	69	40	30
QSL	F	84	-	-	46	27
LKL	F	62	148	61	39	27
NAV	M	44	173	72	37	27
NRH	F	67	-	-	36	26
NAH	M	57	180	75	27	26
TKB	F	77	-	-	34	23
LSL	M	56	-	-	21	19

Bibliography

- [1] K. Strong, C. Mathers, and R. Bonita, "Preventing stroke: Saving lives around the world," *The Lancet Neurology*, vol. 6, no. 2, pp. 182–187, 2007. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1474442207700315>.
- [2] (2015). Population ageing and sustainable development, Department of Economic and Social Affairs, United Nations, [Online]. Available: http://www.un.org/en/development/desa/population/publications/pdf/popfacts/PopFacts_2014-4Rev1.pdf.
- [3] J. D. Schaechter, "Motor rehabilitation and brain plasticity after hemiparetic stroke," *Progress in Neurobiology*, vol. 73, no. 1, pp. 61–72, 2004. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0301008204000565>.
- [4] L. von Koch, L. W. Holmqvist, A. W. Wottrich, K. Tham, and J. de Pedro-Cuesta, "Rehabilitation at home after stroke: A descriptive study of an individualized intervention," *Clinical Rehabilitation*, vol. 14, no. 6, pp. 574–583, 2000, PMID: 11128731. DOI: 10.1191/0269215500cr364oa.
- [5] G. C. H. Koh, S. K. Saxena, T. P. Ng, D. Yong, and N. P. Fong, "Effect of duration, participation rate, and supervision during community rehabilitation on functional outcomes in the first poststroke year in singapore," *Archives of physical medicine and rehabilitation*, vol. 93, no. 2, pp. 279–286, 2012. DOI: 10.1016/j.apmr.2011.08.017.
- [6] P. M. van Vliet and G. Wulf, "Extrinsic feedback for motor learning after stroke: What is the evidence?" *Disability and Rehabilitation*, vol. 28, no. 13-14, pp. 831–840, 2006, PMID: 16777770. DOI: 10.1080/09638280500534937.
- [7] Xsens. (2015). Xsens 3d motion tracking, [Online]. Available: <https://www.xsens.com/products/xsens-mvn/>.

- [8] Noitom. (2015). Perception neuron, [Online]. Available: https://neuronmocap.com/products/perception_neuron.
- [9] APDM. (2015). Apdm - wearable sensors, [Online]. Available: <http://www.apdm.com/wearable-sensors/>.
- [10] soft2tec GmbH. (2015). Nexonar - next generation tracking technology, [Online]. Available: <http://nexonar.com/en/products/>.
- [11] Aseccion Technology Corporation. (2015). Aseccion technology corp electromagnetic tracking - aseccion technology products, [Online]. Available: <http://www.ascension-tech.com/products/#3d-guidance>.
- [12] J. A. Corrales, F. A. Candelas, and F. Torres, "Hybrid tracking of human operators using imu/uwb data fusion by a kalman filter," in *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Amsterdam, Netherlands, 2008, pp. 193–200. DOI: 10.1145/1349822.1349848.
- [13] M. Yavari and B. G. Nickerson, "Ultra wideband wireless positioning systems," *Dept. Faculty Comput. Sci., Univ. New Brunswick, Fredericton, NB, Canada, Tech. Rep. TR14-230*, 2014.
- [14] V. C. Coffey, "Imaging in 3-d: Killer apps coming soon to a device near you!" *Optics and Photonics News*, vol. 25, no. 6, pp. 36–43, 2014. [Online]. Available: <http://www.osa-opn.org/abstract.cfm?URI=opn-25-6-36>.
- [15] NaturalPoint. (2015). Build your own motion capture system, [Online]. Available: <http://www.optitrack.com/systems/#motive-body/flex-3/6>.
- [16] Vicon Motion Systems Ltd. (2015). Bonita, [Online]. Available: <http://www.vicon.com/products/camera-systems/bonita>.
- [17] M. Windolf, N. Götzen, and M. Morlock, "Systematic accuracy and precision analysis of video motion capturing systems-exemplified on the vicon-460 system," *Journal of Biomechanics*, vol. 41, no. 12, pp. 2776–2780, 2008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0021929008003229>.
- [18] R. Baker, "Gait analysis methods in rehabilitation," English, *Journal of NeuroEngineering and Rehabilitation*, vol. 3, no. 1, pp. 1–10, 2006. [Online]. Available: <http://dx.doi.org/10.1186/1743-0003-3-4>.

- [19] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, “Real-time human pose recognition in parts from single depth images,” *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, Jan. 2013, ISSN: 0001-0782. DOI: 10.1145/2398356.2398381.
- [20] K. Khoshelham and S. O. Elberink, “Accuracy and resolution of kinect depth data for indoor mapping applications,” *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.
- [21] M. Zhao, F. Tan, C. W. Fu, C. K. Tang, J. Cai, and T. J. Cham, “High-quality kinect depth filtering for real-time 3d telepresence,” in *2013 IEEE International Conference on Multimedia and Expo (ICME)*, San Jose, CA, USA, 2013, pp. 1–6. DOI: 10.1109/ICME.2013.6607501.
- [22] T. Dutta, “Evaluation of the kinect™ sensor for 3-d kinematic measurement in the workplace,” *Applied Ergonomics*, vol. 43, no. 4, pp. 645–649, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0003687011001529>.
- [23] A. Pfister, A. M. West, S. Bronner, and J. A. Noah, “Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis,” *Journal of medical engineering & technology*, vol. 38, no. 5, pp. 274–280, 2014.
- [24] H. Mokhtarzadeh, A. Emmens, L. V. S. Peter, and O. Denny, “Feasibility evaluation of the microsoft kinect as a low cost alternative to motion capture systems,” in *Australian & New Zealand Orthopaedic Research Society, 17th Annual meeting*, Brisbane, Australia, 2012.
- [25] Š. Obdržálek, G. Kurillo, F. Ofli, R. Bajcsy, E. Seto, H. Jimison, and M. Pavel, “Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population,” in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Diego, CA, USA, 2012, pp. 1188–1193. DOI: 10.1109/EMBC.2012.6346149.
- [26] D. Webster and O. Celik, “Experimental evaluation of microsoft kinect’s accuracy and capture rate for stroke rehabilitation applications,” in *2014 IEEE Haptics Symposium (HAPTICS)*, Houston, TX, USA, 2014, pp. 455–460. DOI: 10.1109/HAPTICS.2014.6775498.

- [27] B. Galna, G. Barry, D. Jackson, D. Mhiripiri, P. Olivier, and L. Rochester, "Accuracy of the microsoft kinect sensor for measuring movement in people with parkinson's disease," *Gait & Posture*, vol. 39, no. 4, pp. 1062–1068, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0966636214000241>.
- [28] C. Culbertson. (2015). Introducing the intel® realsense™ 3d r200 camera, [Online]. Available: <https://software.intel.com/en-us/articles/realsense-r200-camera>.
- [29] H. Fürntratt and H. Neuschmied, "Evaluating pointing accuracy on kinect v2 sensor," in *International Conference on Multimedia and Human-Computer Interaction (MHCI)*, Prague, Czech Republic, 2014.
- [30] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. E. Saddik, "Evaluating and improving the depth accuracy of kinect for windows v2," *Sensors Journal, IEEE*, vol. 15, no. 8, pp. 4275–4285, 2015.
- [31] R. A. Clark, Y. H. Pua, C. C. Oliveira, K. J. Bower, S. Thilarajah, R. McGaw, K. Hasanki, and B. F. Mentiplay, "Reliability and concurrent validity of the microsoft xbox one kinect for assessment of standing balance and postural control," *Gait & Posture*, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0966636215000740>.
- [32] B. F. Mentiplay, L. G. Perraton, K. J. Bower, Y. H. Pua, R. McGaw, S. Heywood, and R. A. Clark, "Gait assessment using the microsoft xbox one kinect: Concurrent validity and inter-day reliability of spatiotemporal and kinematic variables," *Journal of Biomechanics*, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0021929015002985>.
- [33] X. Xu and R. W. McGorry, "The validity of the first and second generation microsoft kinect™ for identifying joint center locations during static postures," *Applied Ergonomics*, vol. 49, pp. 47–54, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0003687015000149>.
- [34] A. Kolb and F. Pece, "Range imaging," in, 1st ed., ser. Digital Representations of the Real World: How to Capture, Model, and Render Visual Reality. CRC Press, 2015, ch. 4, pp. 51–64.

- [35] A. de los Reyes-Guzmán, I. Dimbwadyo-Terrer, F. Trincado-Alonso, F. Monasterio-Huelin, D. Torricelli, and A. Gil-Agudo, "Quantitative assessment based on kinematic measures of functional impairments during upper extremity movements: A review," *Clinical Biomechanics*, vol. 29, no. 7, pp. 719–727, 2014.
- [36] E. Jaspers, K. Desloovere, H. Bruyninckx, K. Klingels, G. Molenaers, E. Aertbeliën, L. V. Gestel, and H. Feys, "Three-dimensional upper limb movement characteristics in children with hemiplegic cerebral palsy and typically developing children," *Research in Developmental Disabilities*, vol. 32, no. 6, pp. 2283–2294, 2011, ISSN: 0891-4222. DOI: 10.1016/j.ridd.2011.07.038.
- [37] K. Kim, W. K. Song, J. Lee, H. Y. Lee, D. S. Park, B. W. Ko, and J. Kim, "Kinematic analysis of upper extremity movement during drinking in hemiplegic subjects," *Clinical Biomechanics*, vol. 29, no. 3, pp. 248–256, 2014, ISSN: 0268-0033. DOI: 10.1016/j.clinbiomech.2013.12.013.
- [38] Kinetisense. (2015). Kinetisense range of motion analysis software, [Online]. Available: <http://www.kinetisense.com/>.
- [39] Jintronix. (2015). Kinect rehabilitation software - jintronix, [Online]. Available: <http://www.jintronix.com/>.
- [40] E. R. Technologies. (2018). Virtualrehab - play enjoy improve! [Online]. Available: <https://evolvrehab.com/virtualrehab/>.
- [41] J. Zariffa, N. Kapadia, J. L. K. Kramer, P. Taylor, M. Alizadeh-Meghrai, V. Zivanovic, U. Albisser, R. Willms, A. Townson, A. Curt, M. R. Popovic, and J. D. Steeves, "Relationship between clinical assessments of function and measurements from an upper-limb robotic rehabilitation device in cervical spinal cord injury," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 3, pp. 341–350, 2012, ISSN: 1534-4320. DOI: 10.1109/TNSRE.2011.2181537.
- [42] Brontes Processing. (2018). Kinect rehabilitation with biofeedback, [Online]. Available: <http://www.virtual-reality-rehabilitation.com/products/seeme/what-is-seeme>.
- [43] S. Balasubramanian, A. Melendez-Calderon, and E. Burdet, "A robust and sensitive metric for quantifying movement smoothness," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 8, pp. 2126–2136, 2012, ISSN: 0018-9294. DOI: 10.1109/TBME.2011.2179545.

- [44] E. W. A. Cacho, R. de Oliveira, R. L. Ortolan, R. Varoto, and A. Cliquet, "Upper limb assessment in tetraplegia: Clinical, functional and kinematic correlations," *International journal of rehabilitation research*, vol. 34, no. 1, pp. 65–72, 2011.
- [45] C. E. Lang, J. M. Wagner, A. J. Bastian, Q. Hu, D. F. Edwards, S. A. Sahrman, and A. W. Dromerick, "Deficits in grasp versus reach during acute hemiparesis," *Experimental Brain Research*, vol. 166, no. 1, pp. 126–136, 2005, ISSN: 1432-1106. DOI: 10.1007/s00221-005-2350-6.
- [46] A Merlo, M Longhi, E Giannotti, P Prati, M Giacobbi, E Ruscelli, A Mancini, M Ottaviani, L Montanari, and D Mazzoli, "Upper limb evaluation with robotic exoskeleton. normative values for indices of accuracy, speed and smoothness," *NeuroRehabilitation*, vol. 33, no. 4, pp. 523–530, 2013.
- [47] J. J. Chang, T. I. Wu, W. L. Wu, and F. C. Su, "Kinematical measure for spastic reaching in children with cerebral palsy," *Clinical Biomechanics*, vol. 20, no. 4, pp. 381–388, 2005.
- [48] B. Rohrer, S. Fasoli, H. I. Krebs, R. Hughes, B. Volpe, W. R. Frontera, J. Stein, and N. Hogan, "Movement smoothness changes during stroke recovery," *The Journal of Neuroscience*, vol. 22, no. 18, pp. 8297–8304, 2002.
- [49] E. Vergaro, M. Casadio, V. Squeri, P. Giannoni, P. Morasso, and V. Sanguineti, "Self-adaptive robot training of stroke survivors for continuous tracking movements," *Journal of NeuroEngineering and Rehabilitation*, vol. 7, no. 1, p. 13, 2010, ISSN: 1743-0003. DOI: 10.1186/1743-0003-7-13.
- [50] N. Hogan and D. Sternad, "Sensitivity of smoothness measures to movement duration, amplitude, and arrests," *Journal of Motor Behavior*, vol. 41, no. 6, pp. 529–534, 2009, PMID: 19892658. DOI: 10.3200/35-09-004-RC.
- [51] J. Aizawa, T. Masuda, T. Koyama, K. Nakamaru, K. Isozaki, A. Okawa, and S. Morita, "Three-dimensional motion of the upper extremity joints during various activities of daily living," *Journal of biomechanics*, vol. 43, no. 15, pp. 2915–2922, 2010.
- [52] P. S. Lum, S. Mulroy, R. L. Amdur, P. Requejo, B. I. Prilutsky, and A. W. Dromerick, "Gains in upper extremity function after stroke via recovery or compensation: Potential differential effects on amount of real-world limb use," *Topics in Stroke Rehabilitation*, vol. 16, no. 4, pp. 237–253, 2009. DOI: 10.1310/tsr1604-237.

- [53] S. Balasubramanian, R. Colombo, I. Sterpi, V. Sanguineti, and E. Burdet, "Robotic assessment of upper limb motor function after stroke," *American Journal of Physical Medicine & Rehabilitation*, vol. 91, no. 11, S255–S269, 2012.
- [54] M. A. Murphy, C. Willén, and K. S. Sunnerhagen, "Kinematic variables quantifying upper-extremity performance after stroke during reaching and drinking from a glass," *Neurorehabilitation and Neural Repair*, vol. 25, no. 1, pp. 71–80, 2011. DOI: 10.1177/1545968310370748.
- [55] C. A. Trombly and C. Y. Wu, "Effect of rehabilitation tasks on organization of movement after stroke," *American Journal of Occupational Therapy*, vol. 53, no. 4, pp. 333–344, 1999. DOI: 10.5014/ajot.53.4.333.
- [56] S. K. Subramanian, J. Yamanaka, G. Chilingaryan, and M. F. Levin, "Validity of movement pattern kinematics as measures of arm motor impairment poststroke," *Stroke*, vol. 41, no. 10, pp. 2303–2308, 2010.
- [57] G. Alankus, "Motion-based video games for stroke rehabilitation with reduced compensatory motions," English, PhD thesis, Washington University in St. Louis, 2011, p. 238, ISBN: 9781267098078. [Online]. Available: <https://search.proquest.com/docview/916383077?accountid=12665>.
- [58] G. Alankus and C. Kelleher, "Reducing compensatory motions in video games for stroke rehabilitation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12, Austin, Texas, USA: ACM, 2012, pp. 2049–2058, ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208354.
- [59] L. E. Sucar, R. Luis, R. Leder, J. Hernández, and I. Sánchez, "Gesture therapy: A vision-based system for upper extremity stroke rehabilitation," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, Buenos Aires, Argentina, 2010, pp. 3690–3693. DOI: 10.1109/IEMBS.2010.5627458.
- [60] Reflexion Health. (2015). Reflexion health | prescriptive software, [Online]. Available: <http://reflexionhealth.com/>.
- [61] R. Komatireddy, A. Chokshi, J. Basnett, M. Casale, D. Goble, and T. Shubert, "Quality and quantity of rehabilitation exercises delivered by a 3-d motion controlled camera: A pilot study," *International journal of physical medicine & rehabilitation*, vol. 2, no. 4, 2014.

- [62] K. H. Leo and W. T. Ang, “Nonlinear time normalisation for accurate characterization of normative upper extremity kinematics during reach-to-touch and reach-to-grasp activities,” *Journal of Medical Imaging and Health Informatics*, vol. 2, no. 4, pp. 411–418, 2012.
- [63] Y. Petrov. (2009). Ellipsoid fit, [Online]. Available: <http://www.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit>.
- [64] D. Comotti, M. Ermidoro, M. Galizzi, and A. Vitali, “Development of a wireless low-power multi-sensor network for motion tracking applications,” in *2013 IEEE International Conference on Body Sensor Networks*, Cambridge, MA, USA, 2013, pp. 1–6. DOI: 10.1109/BSN.2013.6575466.
- [65] J. Van Waveren, *From quaternion to matrix and back*, Id Software, Inc., 2005. [Online]. Available: http://fabiansanglard.net/doom3_documentation/37726-293748.pdf.
- [66] P. Jatesiktat and W. T. Ang, “Recovery of forearm occluded trajectory in kinect using a wrist-mounted inertial measurement unit,” in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Seogwipo, South Korea, 2017, pp. 807–812. DOI: 10.1109/EMBC.2017.8036947.
- [67] F. Bogo, M. J. Black, M. Loper, and J. Romero, “Detailed full-body reconstructions of moving people from monocular rgb-d sequences,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 2300–2308. DOI: 10.1109/ICCV.2015.265.
- [68] M. Ye, Y. Shen, C. Du, Z. Pan, and R. Yang, “Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1517–1532, 2016, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2557783.
- [69] M. Ding and G. Fan, “Articulated and generalized gaussian kernel correlation for human pose estimation,” *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 776–789, 2016, ISSN: 1057-7149. DOI: 10.1109/TIP.2015.2507445.
- [70] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, “Accurate 3d pose estimation from a single depth image,” in *2011 International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 731–738. DOI: 10.1109/ICCV.2011.6126310.

- [71] A. Baak, M. Müller, G. Bharaj, H. P. Seidel, and C. Theobalt, “A data-driven approach for real-time full body pose reconstruction from a depth camera,” in *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*, A. Fossati, J. Gall, H. Grabner, X. Ren, and K. Konolige, Eds., London: Springer London, 2013, ch. 5, pp. 71–98, ISBN: 978-1-4471-4640-7. DOI: 10.1007/978-1-4471-4640-7_5.
- [72] T. Helten, M. Müller, H. P. Seidel, and C. Theobalt, “Real-time body tracking with one depth camera and inertial sensors,” in *2013 IEEE International Conference on Computer Vision*, Sydney, NSW, Australia, 2013, pp. 1105–1112. DOI: 10.1109/ICCV.2013.141.
- [73] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, “Efficient human pose estimation from single depth images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2821–2840, 2013, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.241.
- [74] G. Pons-Moll, J. Taylor, J. Shotton, A. Hertzmann, and A. Fitzgibbon, “Metric regression forests for correspondence estimation,” *International Journal of Computer Vision*, vol. 113, no. 3, pp. 163–175, 2015, ISSN: 1573-1405. DOI: 10.1007/s11263-015-0818-9.
- [75] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy, “Evaluation of pose tracking accuracy in the first and second generations of microsoft kinect,” in *2015 International Conference on Healthcare Informatics*, Dallas, TX, USA, 2015, pp. 380–389. DOI: 10.1109/ICHI.2015.54.
- [76] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton, “Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences,” *ACM Transactions on Graphics*, vol. 35, no. 4, 143:1–143:12, 2016, ISSN: 0730-0301. DOI: 10.1145/2897824.2925965.
- [77] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, “SMPL: A skinned multi-person linear model,” *ACM Transactions on Graphics*, vol. 34, no. 6, 248:1–248:16, Oct. 2015, ISSN: 0730-0301. DOI: 10.1145/2816795.2818013.

- [78] G. Pons-Moll, A. Baak, T. Helten, M. Müller, H. P. Seidel, and B. Rosenhahn, "Multisensor-fusion for 3d full-body human motion capture," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, 2010, pp. 663–670. DOI: 10.1109/CVPR.2010.5540153.
- [79] G. Pons-Moll, A. Baak, J. Gall, L. Leal-Taixé, M. Müller, H. P. Seidel, and B. Rosenhahn, "Outdoor human motion capture using inverse kinematics and von mises-fisher sampling," in *2011 International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 1243–1250. DOI: 10.1109/ICCV.2011.6126375.
- [80] T. v. Marcard, G. Pons-Moll, and B. Rosenhahn, "Human pose estimation from video and imus," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1533–1547, 2016, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2522398.
- [81] K. M. G. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1, Madison, WI, USA, 2003, pp. 77–84. DOI: 10.1109/CVPR.2003.1211340.
- [82] S. Corazza, L. Mündermann, E. Gambaretto, G. Ferrigno, and T. P. Andriacchi, "Markerless motion capture through visual hull, articulated icp and subject specific model generation," *International Journal of Computer Vision*, vol. 87, no. 1, p. 156, 2009, ISSN: 1573-1405. DOI: 10.1007/s11263-009-0284-3.
- [83] T. Helten, A. Baak, G. Bharaj, M. Müller, H. P. Seidel, and C. Theobalt, "Personalization and evaluation of a real-time depth-based full body tracker," in *2013 International Conference on 3D Vision*, Seattle, WA, USA, 2013, pp. 279–286. DOI: 10.1109/3DV.2013.44.
- [84] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: Shape completion and animation of people," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 408–416, Jul. 2005, ISSN: 0730-0301. DOI: 10.1145/1073204.1073207.
- [85] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97, New York, NY, USA: ACM

- Press/Addison-Wesley Publishing Co., 1997, pp. 209–216, ISBN: 0-89791-896-7. DOI: 10.1145/258734.258849.
- [86] D. Ebery. (May 2016). Mesh differential geometry. pp. 9, [Online]. Available: <https://www.geometrictools.com/Documentation/MeshDifferentialGeometry.pdf>.
 - [87] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic differentiation in machine learning: A survey,” *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018. [Online]. Available: <http://www.jmlr.org/papers/volume18/17-468/17-468.pdf>.
 - [88] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. Fitzgibbon, “User-specific hand modeling from monocular depth sequences,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014, pp. 644–651. DOI: 10.1109/CVPR.2014.88.
 - [89] Vicon. (2018). Plug-in gait model details, [Online]. Available: <https://www.vicon.com/downloads/documentation/plug-in-gait-model-details>.
 - [90] P. Jatesiktat. (2018). NTU motion capture dataset, [Online]. Available: https://koonyook.github.io/mocap_dataset/.
 - [91] A. Murgia, P. Kyberd, and T. Barnhill, “The use of kinematic and parametric information to highlight lack of movement and compensation in the upper extremities during activities of daily living,” *Gait & Posture*, vol. 31, no. 3, pp. 300–306, 2010, ISSN: 0966-6362. DOI: 10.1016/j.gaitpost.2009.11.007.
 - [92] W. Timoszyk, R. De Leon, N London, R. Roy, V. Edgerton, and D. Reinkensmeyer, “The rat lumbosacral spinal cord adapts to robotic loading applied during stance,” *Journal of Neurophysiology*, vol. 88, no. 6, pp. 3108–3117, 2002.
 - [93] D. Holden, T. Komura, and J. Saito, “Phase-functioned neural networks for character control,” *ACM Transactions on Graphics*, vol. 36, no. 4, 42:1–42:13, Jul. 2017, ISSN: 0730-0301. DOI: 10.1145/3072959.3073663.
 - [94] S. Patel, R. Hughes, T. Hester, J. Stein, M. Akay, J. G. Dy, and P. Bonato, “A novel approach to monitor rehabilitation outcomes in stroke survivors using wearable technology,” *Proceedings of the IEEE*, vol. 98, no. 3, pp. 450–461, 2010, ISSN: 0018-9219. DOI: 10.1109/JPROC.2009.2038727.

- [95] S. B. Thies, P. A. Tresadern, L. P. Kenney, J. Smith, D. Howard, J. Y. Goulermas, C. Smith, and J. Rigby, "Movement variability in stroke patients and controls performing two upper limb functional tasks: A new assessment methodology," *Journal of NeuroEngineering and Rehabilitation*, vol. 6, no. 1, p. 2, 2009, ISSN: 1743-0003. DOI: 10.1186/1743-0003-6-2.
- [96] S. L. Hahn, *Hilbert transforms in signal processing*. Artech House, 1996.
- [97] A. Pikovsky, M. Rosenblum, J. Kurths, and J. Kurths, "Instantaneous phase and frequency of a signal," in *Synchronization: a universal concept in nonlinear sciences*, vol. 12, Cambridge university press, 2003, ch. A2, pp. 362–370.
- [98] J.-P. Lachaux, E. Rodriguez, J. Martinerie, and F. J. Varela, "Measuring phase synchrony in brain signals," *Human Brain Mapping*, vol. 8, no. 4, pp. 194–208, 1999. DOI: 10.1002/(SICI)1097-0193(1999)8:4<194::AID-HBM4>3.0.CO;2-C.
- [99] R. Quian Quiroga, A. Kraskov, T. Kreuz, and P. Grassberger, "Performance of different synchronization measures in real data: A case study on electroencephalographic signals," *Phys. Rev. E*, vol. 65, p. 041903, 4 2002. DOI: 10.1103/PhysRevE.65.041903.
- [100] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *Computing Research Repository (CoRR)*, vol. abs/1603.04467, 2016. [Online]. Available: <http://arxiv.org/abs/1603.04467>.
- [101] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computing Research Repository (CoRR)*, vol. abs/1412.6980, 2014. arXiv: 1412.6980. [Online]. Available: <http://arxiv.org/abs/1412.6980>.
- [102] P. Jatesiktat, *Thesis supplementary materials*, <https://koonyook.github.io/ntu-thesis-materials/>.

- [103] P. Jatesiktat, D. Anopas, and W. T. Ang, "Unsupervised phase learning and extraction from repetitive movements," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Honolulu, HI, USA, 2018, pp. 227–230. DOI: 10.1109/EMBC.2018.8512196.
- [104] B. Knorr, R. Hughes, D. Sherrill, J. Stein, M. Akay, and P. Bonato, "Quantitative measures of functional upper limb movement in persons after stroke," in *Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005.*, Arlington, VA, USA, 2005, pp. 252–255. DOI: 10.1109/CNE.2005.1419604.
- [105] T. Hester, R. Hughes, D. M. Sherrill, B. Knorr, M. Akay, J. Stein, and P. Bonato, "Using wearable sensors to measure motor abilities following stroke," in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, Cambridge, MA, USA, 2006, pp. 4–8. DOI: 10.1109/BSN.2006.57.
- [106] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, 15:1–15:58, Jul. 2009, ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.
- [107] H. Hoffmann, "Kernel pca for novelty detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863–874, 2007, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2006.07.009.
- [108] B. B. Thompson, R. J. Marks, J. J. Choi, M. A. El-Sharkawi, M.-Y. Huang, and C. Bunje, "Implicit learning in autoencoder novelty assessment," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, vol. 3, Honolulu, HI, USA, 2002, 2878–2883 vol.3. DOI: 10.1109/IJCNN.2002.1007605.
- [109] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with non-linear dimensionality reduction," in *Proceedings of the MLSDA 2014 2Nd Workshop on Machine Learning for Sensory Data Analysis*, ser. MLSDA'14, Gold Coast, QLD, Australia: ACM, 2014, 4:4–4:11, ISBN: 978-1-4503-3159-3. DOI: 10.1145/2689746.2689747.
- [110] P. Jatesiktat and W. T. Ang, "Unsupervised anomalous movement detection using autoencoder reconstruction error," in *Proceedings of AUN/SEED-NET Regional Conference on Computer and Information Engineering*, Yangon, Myanmar, 2016, pp. 198–203.

- [111] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99, Denver, CO: MIT Press, 1999, pp. 582–588. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3009657.3009740>.
- [112] L. M. Inc. (2018). Leap motion controller, [Online]. Available: <https://www.leapmotion.com/>.
- [113] uSens Inc. (2018). Fingo, [Online]. Available: <https://www.usens.com/fingo>.
- [114] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints," in *2011 International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 2088–2095. DOI: 10.1109/ICCV.2011.6126483.
- [115] V. Lepetit and P. Fua, "Monocular model-based 3D tracking of rigid objects," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2005, ISSN: 1572-2740. DOI: 10.1561/06000000001.
- [116] C. Choi and H. I. Christensen, "Rgb-d object tracking: A particle filter approach on gpu," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013, pp. 1084–1091. DOI: 10.1109/IR0S.2013.6696485.
- [117] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother, "6-dof model based tracking via object coordinate regression," in *2014 Asian Conference of Computer Vision (ACCV)*, D. Cremers, I. Reid, H. Saito, and M.-H. Yang, Eds., Singapore: Springer International Publishing, 2014, pp. 384–399, ISBN: 978-3-319-16817-3. DOI: 10.1007/978-3-319-16817-3_25.
- [118] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, and S. Schaal, "Depth-based object tracking using a robust gaussian filter," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016, pp. 608–615. DOI: 10.1109/ICRA.2016.7487184.
- [119] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992. DOI: 10.1109/34.121791.

- [120] Z. Zhang, "On local matching of free-form curves," in *Proceedings of the British Machine Vision Conference*, Leeds, England: Springer London, 1992, pp. 347–356. DOI: 10.1007/978-1-4471-3201-1_36.
- [121] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, Quebec City, Canada, 2001, pp. 145–152. DOI: 10.1109/IM.2001.924423.
- [122] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013. DOI: 10.1007/s10514-013-9327-2.
- [123] M. Rouhani and A. D. Sappa, "Correspondence free registration through a point-to-model distance minimization," in *2011 International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 2150–2157. DOI: 10.1109/ICCV.2011.6126491.
- [124] N. Somani, C. Cai, A. Perzylo, M. Rickert, and A. Knoll, "Object recognition using constraints from primitive shape matching," in *International Symposium on Visual Computing*, Las Vegas, NV, USA, 2014, pp. 783–792. DOI: 10.1007/978-3-319-14249-4_75.
- [125] M. Rouhani and A. D. Sappa, "The richer representation the better registration," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5036–5049, 2013. DOI: 10.1109/TIP.2013.2281427.
- [126] D. R. Canelhas, T. Stoyanov, and A. J. Lilienthal, "SDF tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013, pp. 3671–3676. DOI: 10.1109/IRoS.2013.6696880.
- [127] W. Kehl, N. Navab, and S. Ilic, "Coloured signed distance fields for full 3d object reconstruction," in *Proceedings of the British Machine Vision Conference*, Nottingham, England: BMVA Press, 2014. DOI: <http://dx.doi.org/10.5244/C.28.41>.
- [128] M. Slavcheva, W. Kehl, N. Navab, and S. Ilic, "SDF-2-SDF: Highly accurate 3D object reconstruction," in *European Conference on Computer Vision*, Amsterdam, Netherlands, 2016. DOI: 10.1007/978-3-319-46448-0_41.

- [129] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, “Real-time camera tracking and 3D reconstruction using signed distance functions,” in *Robotics: Science and Systems*, Berlin, Germany, 2013. DOI: 10.15607/RSS.2013.IX.035.
- [130] C. Y. Ren, V. Prisacariu, D. Murray, and I. Reid, “Star3d: Simultaneous tracking and reconstruction of 3d objects using rgb-d data,” in *2013 IEEE International Conference on Computer Vision*, Sydney, NSW, Australia, 2013, pp. 1561–1568. DOI: 10.1109/ICCV.2013.197.
- [131] S. F. Frisken, “Designing with distance fields,” in *International Conference on Shape Modeling and Applications 2005 (SMI’05)*, Cambridge, MA, USA, 2005, pp. 58–59. DOI: 10.1109/SMI.2005.16.
- [132] G. Lu, L. Ren, A. Kolagunda, X. Wang, I. B. Turkbey, P. L. Choyke, and C. Kambhamettu, “Representing 3D shapes based on implicit surface functions learned from RBF neural networks,” *Journal of Visual Communication and Image Representation*, vol. 40, pp. 852–860, 2016. DOI: 10.1016/j.jvcir.2016.08.014.
- [133] M. Jabarouti Moghaddam and H. Soltanian-Zadeh, “Automatic segmentation of brain structures using geometric moment invariants and artificial neural networks,” in *21st International Conference on Information Processing in Medical Imaging*, Williamsburg, VA, USA, 2009, pp. 326–337. DOI: 10.1007/978-3-642-02498-6_27.
- [134] D. J. Tan and S. Ilic, “Multi-forest tracker: A chameleon in tracking,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014, pp. 1202–1209. DOI: 10.1109/CVPR.2014.157.
- [135] D. J. Tan, F. Tombari, S. Ilic, and N. Navab, “A versatile learning-based 3d temporal tracker: Scalable, robust, online,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 693–701. DOI: 10.1109/ICCV.2015.86.
- [136] S. Akkaladevi, M. Ankerl, C. Heindl, and A. Pichler, “Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016, pp. 5644–5649. DOI: 10.1109/ICRA.2016.7487784.

- [137] D. J. Tan, N. Navab, and F. Tombari, “Looking beyond the simple scenarios: Combining learners and optimizers in 3D temporal tracking,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 11, pp. 2399–2409, 2017. DOI: 10.1109/TVCG.2017.2734539.
- [138] M. Garon and J. Lalonde, “Deep 6-dof tracking,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 11, pp. 2410–2418, 2017, ISSN: 1077-2626. DOI: 10.1109/TVCG.2017.2734599.
- [139] M. Levoy, *The stanford 3d scanning repository*, <http://graphics.stanford.edu/data/3Dscanrep/>.
- [140] P. Jatesiktat, M. J. Foo, and G. M. Lim, *SDF-Net: Supplementary materials repository*, <https://koonyook.github.io/SDF-Net-materials/>.
- [141] W. Kehl, F. Tombari, S. Ilic, and N. Navab, “Real-time 3d model tracking in color and depth on a single cpu core,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 465–473. DOI: 10.1109/CVPR.2017.57.
- [142] C. Y. Ren, V. A. Prisacariu, O. Kähler, I. D. Reid, and D. W. Murray, “Real-time tracking of single and multiple objects from depth-colour imagery using 3D signed distance functions,” *International Journal of Computer Vision*, vol. 124, no. 1, pp. 80–95, 2017. DOI: 10.1007/s11263-016-0978-2.
- [143] V. Prisacariu and I. Reid, “PWP3D: Real-time segmentation and tracking of 3D objects,” *International Journal of Computer Vision*, pp. 1–20, 2012. DOI: 10.1007/s11263-011-0514-3.
- [144] R. B. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 1–4. DOI: 10.1109/ICRA.2011.5980567.
- [145] S. C. Akkaladevi, M. Ankerl, G. Fritz, and A. Pichler, “Real-time tracking of rigid objects using depth data,” in *Proceedings of OAGM&ARW Joint Workshop on Computer Vision and Robotics*, Wels, Austria, 2016. DOI: 10.3217/978-3-85125-528-7-14.
- [146] T. Schmidt, R. Newcombe, and D. Fox, “Dart: Dense articulated real-time tracking with consumer depth cameras,” *Autonomous Robots*, vol. 39, no. 3, pp. 239–258, 2015, ISSN: 1573-7527. DOI: 10.1007/s10514-015-9462-z.

- [147] J. Taylor, V. Tankovich, D. Tang, C. Keskin, D. Kim, P. Davidson, A. Kowdle, and S. Izadi, “Articulated distance fields for ultra-fast tracking of hands interacting,” *ACM Transactions on Graphics*, vol. 36, no. 6, 244:1–244:12, Nov. 2017, ISSN: 0730-0301. DOI: 10.1145/3130800.3130853.