Tracking Hands in Action for Gesture-based Computer Input

UNIVERSITÄT DES SAARLANDES

Thesis for obtaining the title of
Doctor of Engineering

of the Faculty of Mathematics and Computer Science
of Saarland University

Srinath Sridhar

Saarbrücken, October 2016

Dean of the Faculty

Univ.-Prof. Dr. Frank-Olaf Schreyer
Saarland University
Saarbrücken, Germany

Defense

December 16, 2016, in Saarbrücken

Chair of the Committee

Prof. Dr. Hans-Peter Seidel

Examiners

Prof. Dr. Christian Theobalt

Prof. Dr. Antti Oulasvirta

Prof. Dr. Bernt Schiele

Prof. Dr. Hao Li

Academic Assistant

Dr. Avishek Chatterjee

To my parents, Bhuvaneswari and Sridhar.

Acknowledgements

First and foremost, I would like to thank my advisors Christian Theobalt and Antti Oulasvirta. During the course of my PhD, I estimate that we have spent over 500 hours (or 20 full days) in meetings and exchanged over 2600 emails. In this process, they provided me with the best possible advice and guidance I could have asked for. In addition, they provided me with support, encouragement, motivation and stood by me at times of crises and success. I am truly lucky to have had the opportunity to work with them.

The Computer Graphics Department at MPI provides such a unique and creative environment that fosters high quality research and excellence. I would like to thank Hans-Peter Seidel for making this possible, and also for his advice and support. I am also grateful to Bernt Schiele and Hao Li for kindly agreeing to be examiners for my thesis.

When I first started as a PhD student, many people helped me get up to speed. In particular, I would like to thank Thomas Helten, Nils Hasler, Carsten Stoll, and Ahmed Elhayek. Present and past members of the GVV Group at MPI, and the HCI/UI Group at MPI/Aalto University have made my stay such an exciting and pleasant experience. There are too many to name here, but I would like to mention Kiran Varanasi, James Tompkin, Levi Valgaerts, Chenglei Wu, Kwang In Kim, Gilles Bailly, Myroslav Bachynskyi, Arttu Modig, Nadia Robertini, Pablo Garrido, Hyeongwoo Kim, Weipeng Xu, Avishek Chatterjee, and many more.

My wonderful experience has been shaped by my amazing collaborators and friends: Helge Rhodin, Anna Feit, Franziska Mueller, Michael Zollhöfer, Dan Casas, Anders Markussen, and Sebastian Boring. We have spent many long days (and nights) debugging code, brainstorming ideas, computing gradients, arguing about Gaussians, and in general having a lot of fun. It is an honor to have worked with such talented people. I had an amazing time at Microsoft Research thanks to the mentorship and guidance of Shahram Izadi. Other talented colleagues made my internship experience great: Julien Valentin, Sean Fanello, Cem Keskin, Pushmeet Kohli, David Kim, Sarah Mennicken, Jonathan Taylor, Danny Tang, and many others.

I am indebted to the funding agencies that have supported me: the Max Planck Society, and the Max Planck Center for Visual Computing and Communication (MPC-VCC).

Many people helped prepare supplementary videos, annotate datasets, and provided support for the work presented in this thesis. In particular, I would like to thank Han Du, Monika Singh, Perttu Lähteenlahti, Danny Tang, Christian Richardt, James Tompkin, and Sebastian Schuon. My sincere thanks goes out to administrative staff members, Sabine Budde and Ellen Fries for being so helpful and putting up with many of my last minute requests. I would also like to thank the Information Services and Technology (IST) and AG4 Admins at MPI for their top quality technical support. Thomas Helten and Franziska Mueller graciously shared their office with me without complaining about my deadline driven craziness.

Doing a PhD can at times be hard on health and social life. Special thanks goes out to my gym buddies, Abhimitra Meka and Dushyant Mehta, for making each other push hard. My stay at MPI was enjoyable thanks to water cooler converstations with many smart people such as Christian Richardt, Rhaleb Zayer, Silke Jansen, Shida Beigpour, Ayush Tewari, and many more.

Finally, I would like to thank my friends, too many to name here, for their friendship and support. Last but certainly not the least, I would like to thank my parents, Bhuvaneswari and Sridhar for their unwavering support and empowering me to pursue my interests. Without them this thesis would have never happened.

Abstract

The ubiquity of modern computers in the form of smartphones, smartwatches, and virtual and augmented reality glasses has lead to the need to create new ways of computer input. Conventional input devices like the keyboard and the mouse can no longer be used for such emerging devices. The human hands are highly dexterous and could provide an *always-on* input capability through the use of gestures. In this thesis, we introduce new methods for markerless tracking of the full articulated motion of hands and using tracked motion for gesture-based computer input.

First, we contribute to computer vision-based markerless tracking of hands for use in computer input. This is a hard problem due to occlusions, uniform skin color, fast motions, and scene clutter. We show that combining novel representations for model-based tracking with discriminative learning techniques can result in mutually exclusive failure modes that help overcome some of the challenges. We show the benefit of our contributions in a variety of scenarios including varying number of cameras, viewpoints, and run-time requirements. We also show that our contributions can scale with scene complexity—it can be used, to our knowledge for the first time, to jointly track hands interacting with objects.

Second, we contribute to gesture-based input driven by markerless hand tracking. The design of appropriate interaction techniques and gestures is a hard problem because of the large design space, and human factors such as ergonomics. We show that gestures elicited from users can be used to develop interaction techniques for 3D navigation tasks. We then identify limitations with elicitation studies and propose a novel method for computational gesture design. This allows designers, for the first time, to automatically generate gestures satisfying criteria such as speed or accuracy. Finally, we show that even limiting hand tracking to only fingertips can enable new input methods for small form factor devices such as smartphones. We conclude the thesis with a critical discussion about limitations and directions for future work.

Kurzfassung

Die Omnipräsenz von modernen Computersystemen wie etwa Smartphones, Smartwatches und Head-Mounted Displays zum Eintauchen in die virtuelle und erweiterte Realiťt führt dazu, dass neue Eingabemodalitäten benötigt werden. Konventionelle Eingabegeräte, wie etwa die Tastatur oder Maus, können nicht länger in Verbindung mit diesen neuen Geräten verwendet werden. Die menschlichen Hände hingegen sind sehr ausdrucksstark und stehen dem Benutzer immer zur Verfügung. In dieser Arbeit werden Verfahren vorgestellt, welche die markerlose Bewegungserfassung der vollständigen artikulierten Hand und der Verwendung dieser Informationen für die Gestensteuerung von Computern ermöglichen.

Zuerst stellen wir neue markerlose Bewegungsschätzungsverfahren für Hände vor, welche zur Computersteuerung verwendet werden können. Das zugrundeliegende Problem ist aufgrund von starken Verdeckungen, der gleichmäßigen Farbe der Hand, schnellen Bewegungen und einer hohen Variabilität des Hintergrundes sehr anspruchsvoll. Zusätzlich zeigen wir, dass eine Kombination von modellbasierten Verfahren und datenbasierten Lernverfahren, aufgrund von unterschiedlichen Stärken und Schwächen dieser beiden Ansätze, einige dieser Hürden meistern können. Wir zeigen die Vorteile unserer Verfahren anhand einer Vielzahl von Beispielen, unter anderem für die Bewegungsschätzung mittels unterschiedlich vieler Kameras, aus verschieden Blickwinkeln und unter Laufzeitbeschränkungen. Wir zeigen auch, dass unsere Verfahren sich an die Szenenkomplexität anpassen lassen. So können diese zum Beispiel auch dazu verwendet werden, gleichzeitig sowohl eine Hand als auch die Bewegung eines manipulierten Objektes zu schätzen.

Als nächstes stellen wir ein Verfahren vor, dass die Gestensteuerung von Computern basierend auf den rekonstruierten Bewegungsabläufen realisiert. Die Erstellung von gut geeignet Interaktionstechniken und Gesten ist ein anspruchsvolles Problem, da der Raum der in Frage kommenden Gesten sehr groß ist und Ansprüche an die Ergonomie mit in Betracht gezogen werden müssen. Zusätzlich erlaubt es die Gestensteuerung dem Benutzer, im dreidimensionalen Raum zu navigieren. Wir zeigen Limitierungen von aktuellen Benutzerstudien auf und stellen ein neues Verfahren vor, dass es erlaubt, Gesten zu entwerfen. Dieses ermöglicht es zum ersten Mal, neue Gesten unter Berücksichtigung bestimmter Kriterien, wie zum Beispiel Geschwindigkeit oder Genauigkeit, automatisch zu entwerfen. Schließlich zeigen wir, dass die Analyse der Bewegung der Fingerspitzen es ermöglicht, kleine Geräte, wie zum Beispiel Smartwatches, zu bedienen. Abschließend werfen wir einen kritischen Blick auf die verbleibenden Limitierungen der vorgestellten Verfahren und Möglichkeiten für zukünftige Forschungsprojekte.

Glossary

BOH back of the hand. 7

CNN convolutional neural network. 146

DIP distal interphalangeal. 2

DOF degree of freedom. xxi, 2, 3, 12, 86, 88

FPS frames per second. 5, 6, 11, 33, 47, 63, 86, 145

GUI graphical user interface. 1

HCI human–computer interaction. 1, 2, 6, 11, 47

IoT internet of things. 1

MCP metacarpophalangeal. 2

PIP proximal interphalangeal. 2

RGB Red Green Blue (Color). 11, 12

VR virtual reality. 147

WPM words per minute. 6

Contents

Glossary **xiii**

List of Figures **xxi**

List of Tables **xxv**

1 Introduction **1**

1.1 The Human Hand 2

1.2 Challenges 3

1.3 Research Problem 4

1.4 Contributions and Structure 4

1.4.1 Part I: Tracking Hands in Action 4

1.4.2 Part II: Gesture-based Computer Input 6

1.5 List of Publications 7

I Tracking Hands in Action **9**

2 Problem Definition and Preliminaries **11**

2.1 Problem Definition and Terminology 11

2.1.1 Shape Representations for Tracking 13

2.1.2 Properties of Gaussian Distributions 14

2.1.3 Perspective Projection of Gaussian Functions 15

2.1.4 Random Forests for Per-Pixel Classification 17

3 Interactive Multi-Camera Hand Tracking **19**

3.1 Introduction 20

3.2 Related Work 21

3.3 Input Data and Hand Modeling 22

3.4 Generative Hand Pose Estimation 23
3.4.1 Model-to-Image Similarity Measure 24
3.4.2 Optimization 25
3.5 Part-based Pose Retrieval 26
3.5.1 Multiple Finger Pose Database Generation 26
3.5.2 Palm and Hand Orientation Estimation 27
3.5.3 Fingertip Detection 28
3.5.4 Finger Pose Estimation 28
3.6 Pose Candidate Fusion 29
3.7 Results 29
3.8 Discussion 31
3.9 Conclusion 32

4 Real-time Hand Tracking with Multiple RGB Cameras 33
4.1 Introduction 33
4.2 Previous Work 35
4.3 Tracking Overview 36
4.4 SAG-based Generative Tracking 37
4.4.1 Fundamentals of SAG Model 37
4.4.2 Projection of 3D SAG to 2D SAG 39
4.4.3 Pose Fitting Energy 40
4.4.4 Pose Optimization 41
4.5 Experiments 43
4.6 Discussion 45
4.7 Conclusion 45

5 Real-time Hand Tracking from a Single Depth Camera 47
5.1 Introduction 47
5.2 Related Work 49
5.3 Input and Model Representation 51
5.3.1 Depth Data Representation 51
5.3.2 Hand Model 52
5.4 Hand Pose Optimization 52
5.4.1 Depth-Only Pose Optimization 53
5.4.2 Objective Function 53
5.4.3 Detection-Guided Pose Optimization 56
5.5 Late Fusion 57

5.6 User Specific Hand Modeling . . . 58
5.7 Results and Evaluation . . . 59
5.7.1 Quantitative Evaluation . . . 59
5.7.2 Qualitative Results . . . 61
5.8 Discussion . . . 63
5.9 Conclusion . . . 63

6 Real-time Joint Tracking of a Hand Manipulating an Object 65
6.1 Introduction . . . 65
6.2 Related Work . . . 67
6.3 Discriminative Hand Part Classification . . . 69
6.4 Gaussian Mixture Model Representation . . . 70
6.5 Unified Density Representation . . . 71
6.6 Multiple Proposal Optimization . . . 72
6.7 Hand-Object Tracking Objectives . . . 73
6.8 Experiments and Results . . . 76
6.9 Discussion . . . 80
6.10 Conclusion . . . 80

II Gesture-based Computer Input 83

7 Continuous Computer Input 85
7.1 Introduction . . . 85
7.2 Related Work . . . 87
7.3 Hand Motion Tracking . . . 88
7.3.1 Physical Setup . . . 89
7.3.2 Tracking Algorithm . . . 89
7.3.3 Fast Bimanual Tracking in a Tabletop Setting . . . 91
7.4 Gesture Elicitation Study and Accuracy Assessment . . . 92
7.4.1 Method . . . 92
7.4.2 Results . . . 93
7.5 Designing Free-Hand Interactions . . . 94
7.6 Free-Hand Interaction Applications . . . 95
7.6.1 Navigation + Selection: Space Invaders . . . 96
7.6.2 Two-Handed Interaction: Menu Selection . . . 96
7.6.3 Emulation of Input Devices: Mouse . . . 96

7.6.4 3D Navigation: Virtual Globe . . . 97
7.6.5 Multiple Controls: First-person Shooter . . . 98
7.7 Study of Virtual Globe Navigation . . . 99
7.7.1 Method . . . 99
7.7.2 Results . . . 100
7.8 Discussion . . . 101
7.9 Conclusion . . . 101

8 Computational Gesture Design 103
8.1 Introduction . . . 104
8.2 Background: Characteristics of Finger Motion . . . 106
8.2.1 The Kinematic Skeleton . . . 106
8.2.2 Movement Performance . . . 106
8.2.3 Inter-Finger Dependencies . . . 107
8.3 Experiment: Finger Dexterity . . . 107
8.3.1 Participants . . . 109
8.3.2 Experimental Design . . . 109
8.3.3 Task, Materials, and Procedure . . . 109
8.3.4 Apparatus . . . 110
8.3.5 Analysis . . . 110
8.4 Results . . . 111
8.4.1 Performance: Fitts' Law Models . . . 111
8.4.2 Individuation: Schieber Indices . . . 112
8.4.3 Comfortable Movement Ranges . . . 113
8.4.4 Observations on Individual Differences . . . 113
8.5 Application to Text Entry . . . 114
8.5.1 Step 1: Discretizing Joint Angles . . . 116
8.5.2 Step 2: Generalizing to Multi-Joint Gestures . . . 116
8.5.3 Step 3: Objective Function Formulation . . . 117
8.5.4 Step 4: Optimization . . . 119
8.6 Design Cases . . . 119
8.6.1 First Observations on User Performance: FastType . . . 121
8.7 Discussion . . . 122
8.8 Conclusion and Future Work . . . 123

9 On- and Above-Skin Sensing for Continuous and Discrete Input 125
9.1 Introduction . . . 125
9.2 WatchSense . . . 127
9.2.1 Embedded Depth Sensors . . . 128
9.2.2 Input Capabilities . . . 128
9.2.3 Resulting Challenges . . . 129
9.3 Related Work . . . 130
9.4 Implementation . . . 131
9.4.1 Prototype System . . . 132
9.4.2 Algorithm Description . . . 132
9.5 Technical Evaluation . . . 135
9.5.1 Runtime Performance . . . 136
9.5.2 Touch Accuracy . . . 136
9.5.3 Touch Tolerance . . . 138
9.5.4 Random Forest Classification Accuracy . . . 139
9.6 WatchSense-Enabled Applications . . . 139
9.7 Discussion . . . 142
9.8 Conclusion . . . 142

10 Conclusions and Future Work 145
10.1 Future Directions and Outlook . . . 146
10.1.1 Single RGB Camera . . . 146
10.1.2 Tracking in Cluttered Scenes from Different Viewpoints . . . 146
10.1.3 Tracking Hands in Conjunction with Full Body . . . 147
10.1.4 Tracking Multiple Strongly Interacting Hands . . . 147
10.1.5 Computational Gesture Design for Continuous Input . . . 147
10.1.6 Application Specific Tracking for Interaction . . . 147
10.1.7 Tangible Computer Input . . . 148

Bibliography 149

List of Figures

1.1 The human–computer communication system (input only). 2
1.2 Simplified illustration of the bones and joints in the hand. 2
1.3 Range of human hand motion 3

2.1 The goal of markerless *hand tracking*. 12
2.2 Kinematic skeleton of the hand (red). We use 32 joints and 26 degrees of freedom (DOFs). 12
2.3 Different shape representations used in tracking. 13
2.4 Perspective projection of ellipsoids. 15
2.5 An ensemble of random decision trees forms a random forest. 17
2.6 A depth image of the hand (left) is segmented into 12 hand parts with a depth classification forest. 18

3.1 Interactive multi-camera hand tracking. 19
3.2 Overview of our interactive multi-camera hand tracking approach. 20
3.3 Input Data and Hand Modeling. 23
3.4 Palm and Hand Orientation Estimation. 27
3.6 Qualitative results of our method as seen from two camera views. 31

4.1 Qualitative results from our SAG-based tracking method. 33
4.2 Overview of the our SAG-based tracking framework. 36
4.3 Comparison of SAG and SoG models. 39
4.4 Sketch of the perspective projection of ellipsoids. 40
4.5 Tracking error comparison of SAG with SoG. 43
4.6 Average errors for all sequences in the Dexter 1 dataset. 44
4.7 Qualitative tracking results. 45

5.1 Overview of our detection-guided tracking method. 48
5.2 Depth Similarity Term. 55

5.3 Automatic fitting of user specific hand model for 4 subjects, one of whom is wearing a thick glove to simulate variability in hand dimension. The red spheres denote 3D Gaussians. 58
5.4 Results on Dexter 1 dataset. 60
5.5 Effect of varying the number of particle and iterations. 61
5.6 Plot of the error for the depth-only tracking and late fusion approach. 61
5.7 Qualitative results from our tracking approach. 62
5.8 Tracking from egocentric viewpoint. 63

6.1 Proposed approach for joint hand and object tracking. 66
6.2 Overview of joint hand and object tracking. 67
6.3 Three stage hand part classification. 69
6.4 Quantitative hand-object tracking evaluation on ground truth data. The object contributes a higher error. 77
6.6 Ablative analysis. 77
6.5 Qualitative results of joint hand and object tracking. 78
6.7 (a, b) show tracking results on our dataset. (c) shows real-time results with different object shapes and colors. (d) shows results on a public dataset . . . 79
6.8 Subset of tracked frames on the dataset of [146]. 79
6.9 Occlusion error and recovery. 80

7.1 FullHand can track one or both hands with all fingers. 86
7.2 Our tabletop setup requires 5 RGB cameras and 1 depth sensor. 89
7.3 The tracking algorithm is a combination of a generative and discriminative method. 90
7.4 Plot of the computation times for one and two hands. 91
7.5 Plot of accuracy defined as the percentage of frames with error <15 mm. . . 93
7.6 Examples of interaction applications made possible by FullHand. 97
7.7 Interaction techniques for virtual globe in *space viewpoint*. 97
7.8 Interaction techniques for virtual globe in *terrain viewpoint*. 98
7.9 Tasks used in the Virtual Globe study. 99
7.10 Development of task performance for mouse vs. free-hand interactions in four navigation tasks with the Virtual Globe. 100

8.1 We investigate the dexterity of using multiple fingers for mid-air input. . . . 103
8.3 The experiment investigates the dexterity of six joints that can be reliably tracked with the Leap Motion sensor. 108
8.4 Performance models for each joint as given by Fitts' law. 111

8.5 Raw data for movement of Index relative to instructed movement of Thumb-Down. 114
8.6 Average coactivation of all joints relative to the instructed movement of the middle finger. The slopes are the average of the absolute values over all users. 115
8.7 Differences among users (denoted by four digit user ID) in the movement of the index finger relative to the middle finger. 116

9.1 *WatchSense* enables on- and above-skin input on the back of the hand (BOH) through a wrist-worn depth sensor. 126
9.2 Interaction techniques enabled by *WatchSense*. 127
9.3 Overview of *WatchSense* implementation. 133
9.4 Method for fingertip detection. 134
9.5 Method for touch detection. 135
9.6 Evaluation of touch accuracy on the BOH. 136
9.7 Some interactive applications enabled by *WatchSense*. 139
9.8 *CardboardBoxes* game for VR/AR. 140
9.9 *WatchSense* allows for joystick-like directional control for gaming. 142

List of Tables

4.1 Percentage of total frames in a sequence that have an error of less x mm. We observe that `SAG` outperforms `SoG` in all sequences and error bounds. The values in bold face indicate the best values for a given error bound. 44

5.1 Percentage of total frames in a sequence that have an error of less x mm. . . 60

6.1 Average error (mm) for hand and object tracking in our dataset 77

7.1 Results from the elicitation study showing number of participants who used one or two hands. 93
7.2 Guidelines for free-hand interaction design from previous literature. 95
7.3 Comparison of different applications based on the sub-tasks involved. . . . 96

8.1 Fitts' Law models for each joint, given by intercept and slope. 111
8.2 Individuation index and relative coactivation describe the involuntary motion of joints. The individuation index is an aggregate that describes the independence of a finger when averaged over all other fingers (1 = perfect individuation). Relative coactivation denotes the movement of a non-instructed joint when the instructed joint (each row) is moving. A value of 1 denotes that the two joints always move together. 112
8.3 Angular limits and movement range of each joint. The table shows values averaged over all users together with standard deviations. 112
8.4 FastType was optimized favoring Performance. The bin addresses describe each gesture, see text for explanation. Observe how commonly occurring letters like 'a' are assigned to easy postures such as flexing the thumb. . . . 120
8.5 An overview of optimized mappings and predicted WPM. The bottom part shows predictions for two existing methods. 121

Chapter 1

Introduction

Computing devices are becoming increasingly ubiquitous in human society. Until a decade ago, the most common computers were servers, workstations, PCs, or laptops. Recent advances in electronics miniaturization, display technology, and battery capacity, however, have lead to a profound change in the form factors of computers. Today smartphones, smartwatches, smartglasses (for virtual or augmented reality), and smart-televisions have become or are becoming common sight. This trend is expected to continue eventually leading to a world where every physical object has compute capability and is interconnected—the so called *internet of things (IoT)*.

In order to benefit from this explosion of ubiquitous computers, humans need to be able to effectively interchange information with these devices. First, computers need to be capable of efficient information **output**. Typically electronic displays (visual), speakers (auditory), or motorized feedback devices (tactile) are used for this purpose. Users also need to be able to **input** information for storage or for instructions. For example, the keyboard and the mouse are widely adopted for text entry and graphical user interface (GUI) interaction. It is useful to view these input and output channels from an information theoretic perspective. In Figure 1.1, the black hourglass shape represents the human–computer interface channel which is affected by the design of the sensor (input device) and its sources of noise. One of the grand challenges in human–computer interaction (HCI) is to increase the throughput of this channel (e.g., green dotted lines) to allow faster and more efficient communication with computers. This thesis presents some ways of increasing throughput by exploiting the dexterity of the human hand and fingers for computer input.

Increasing the throughput of human–computer interaction is critical not only for improving efficiency in existing devices but also for effective computer input in emerging devices. As users adopt smartphones, smartwatches, and smartglasses, traditional notions of input are challenged. For example, multitouch interaction on smartwatches is hard due to small

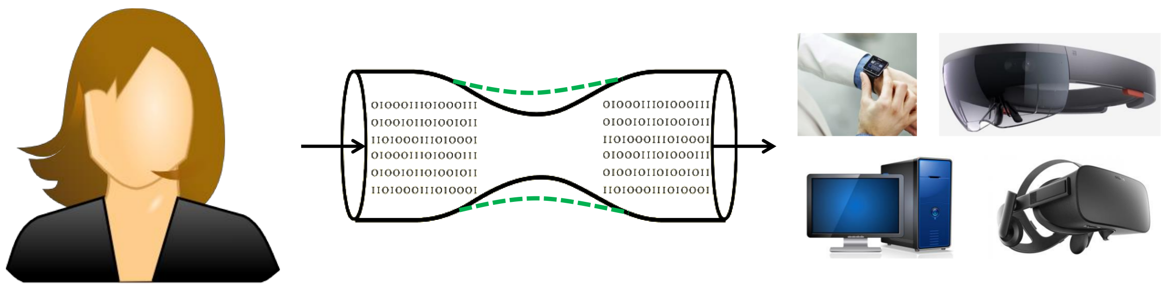


Figure 1.1 The human–computer communication system (input only). The black hourglass shape represents the currently available throughput of the human–computer input channel. This thesis presents some ways of increasing the throughput and expressiveness of input (green dotted line). We present sensing methods, interaction techniques, empirical data, user studies, and working examples to support our claims.

size of the display. Traditional input devices such as the keyboard or the mouse cannot be used in such mobile scenarios.

In this thesis, we ask the question of whether we could use the hand, together with its interaction with the environment, as the input device. Doing so has many advantages such as increased dexterity due to the many DOFs, and mobility. However, it also requires solving several hard, unsolved problems in computer vision and HCI. In this thesis, we present, to our knowledge, the first of their kind solutions to hard computer vision problems such as markerless hand tracking with a single camera, and hard HCI problems such as automatic gesture design. We present sensing methods, interaction techniques, empirical data, user studies, and working examples that support our thesis that the hand can indeed be used as an input device.

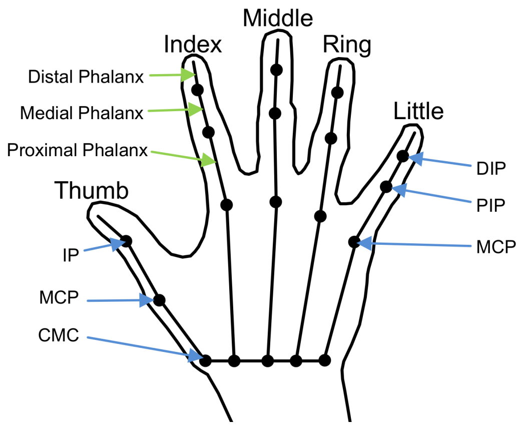


Figure 1.2 Simplified illustration of the bones and joints in the hand.

1.1 The Human Hand

The human hand is the most dexterous of the human extremities with over 26 DOFs [59]. Figure 1.2 is a simplified illustration of bones and joints in the hand. The metacarpophalangeal (MCP) joints in each finger have 2 DOFs while the proximal interphalangeal (PIP) and distal interphalangeal (DIP) joints have 1 DOF each. These DOFs are controlled by 38

Figure 1.3 Human hands can convey information through gestures, dexterously employ tools (e.g., for writing), perform fine motor movements, and manipulate objects with skill.

muscles in the hand and the forearm allowing the hand to **articulate**, i.e., move the 27 bones in a coordinated manner. Due to biomechanical constraints, each DOF has a limited range of motion. The motion of fingers is often expressed as **flexion**, i.e., movement that decreases the angle between a finger segment and its proximal segment (and vice versa for **extension**). **Abduction** refers to a motion that pulls the fingers away from the center of the hand while **adduction** refers to the opposite.

The DOFs of the hand cannot all be independently controlled. Inspite of this limitation, hands are capable of dexterous movements such as gesturing, and tool manipulation. Part of this dexterity comes from the high cortical sensorimotor capacity allocated to the hand [59]. This allows the hand to perform motor movements such as **prehension** (grasping), and non-prehensile skilled movements. Together, these movements lead to an immense range of everyday actions such as gesturing for communication, tool use for building, and sleight of hand (see Figure 1.3 for examples).

1.2 Challenges

Using the expressive capacity of the hand for computer input has been a prime goal for research on input devices and interaction techniques. However, sensing and design challenges have prevented extensive research and use of hand gesture-based computer input. In this thesis, we address these two challenges.

First, sensing or tracking the movement of hands in action poses difficulty due to fast motions. Many sensing technologies such as wired gloves, miniature radar[1], mechanical exoskeletons[2], and camera-based methods have been proposed. Methods that require users to wear gloves or markers prevent use in everyday scenarios because they hinder free hand motion. Non-contact sensing methods such as camera-based computer vision methods work best because they do not require users to wear markers or gloves. Sensing hand motion using cameras is a hard problem because of fast motions, uniform skin color, self-occlusions, and

[1]https://atap.google.com/soli
[2]http://www.dextarobotics.com

environmental clutter. In addition, for use in computer input, high accuracy, and low latency and runtime are crucial.

Given high accuracy and low latency tracking of hands, the second challenge is the design of appropriate gestures that are fast, accurate, and intuitive for users. Literature that uses markerless camera-based hand tracking for interaction is sparse. Thus, an understanding of what hand motions are fast, accurate, and intuitive are missing. Furthermore, designing gestures based on these criteria is a much harder problem because of the immense size of the interaction space.

1.3 Research Problem

Our main research problem is the design of high throughput gesture-based computer input using markerless hand tracking. Previous work on markerless tracking often does not consider the implications and requirements that their use in gesture-based input entails. To be useful for input, methods have to be robust, accurate, have low latency, and high speed. Conversely, the lack of markerless tracking methods suitable for gesture-based input has made it hard to investigate high throughput gestures. Gesture design must take the benefits and limitations of hand trackers into account, be intuitive, memorable, fast, accurate, and also easily trackable by hand trackers. In this thesis, we aim to advance the state of the art in both markerless tracking of hands and gesture-based input by informing the development of one by the other.

1.4 Contributions and Structure

This thesis contributes to both computer vision-based tracking and gesture-based human–computer interaction research. We list the contributions in detail by dividing them into two categories: (1) tracking hands in action, and (2) gesture-based computer input. Please see Section 1.5 for a full list of publications where some of these contributions were originally reported.

1.4.1 Part I: Tracking Hands in Action

In Part I, we contribute to **computer vision** research by presenting new non-contact, markerless algorithms for tracking hands in action. In Chapter 2 we define the problem and introduce basic terminology and concepts that are essential to understanding our contributions.

Chapters 3–6 present four different tracking algorithms each suited for a particular scenario. The supported tracking scenarios can be identified based on three criteria:

- **No. of Cameras:** Multiple cameras (Chapter 3, 4) or single camera (Chapters 5, 6)
- **Run-time:** Interactive (Chapter 3) or real-time (Chapter 5)
- **Scene Complexity:** Hands-only (Chapter 5) vs. hands and objects (Chapter 6)

Together these methods support a range of tracking scenarios previously not supported by other methods: (1) we can track hands in static desktop-based settings more accurately and robustly than previous approaches, (2) we can track hands in real-time from a single depth camera thereby allowing moving egocentric setups, (3) we can, to our knowledge for the first time, also jointly track hands interacting with objects in real-time from a single depth sensor.

In Chapter 3 we focus on multi-camera tracking of only hands at interactive frame rates. We first discuss a traditional pose optimization framework that uses special representations for **generative** tracking. We show that using only this approach for tracking hands results in catastrophic failure. We propose a hybrid approach that combines generative tracking with a novel part-based, **discriminative** pose retrieval strategy. We further improve accuracy of this method by presenting a new shape representation called the 3D Sum of Anisotropic Gaussians (SAG) in Chapter 4. To evaluate these contributions, we introduce an extensive, annotated benchmark dataset consisting of challenging hand motion sequences. Results from validation on this dataset shows that our new shape representation together with the hybrid approach is superior to previous work and allows robust and accurate real-time hand tracking.

In Chapter 5, we shift our attention to tracking hands using a single depth camera. We contribute by proposing a novel shape representation for depth that allows efficient, accurate, and robust tracking of a hand at real-time frame rates. This representation is compact, mathematically smooth and allows us to formulate pose estimation as a 2.5D generative optimization problem in depth. While pose tracking on this representation could run in excess of 120 frames per second (FPS) using gradient-based local optimization, this often results in a wrong local pose optimum. For added robustness we incorporate evidence from trained randomized decision forests that label depth pixels into predefined parts of the hand. The part labels include discriminative detection evidence into generative pose estimation. This enables the tracker to better recover from erroneous local pose optima and prevents temporal jitter common to detection-only approaches. The robustness of this approach allows to track the full articulated 3D pose of the hand under different poses such as pinching and those with self-occlusions. Because it uses only a single depth camera, our approach is one of the first methods to track from moving head-mounted cameras and other similar egocentric viewpoints.

Finally, in Chapter 6 we present a first-of-its-kind method to address the harder problem of jointly tracking hands and objects using a single RGB-D camera at real-time frame rates. Jointly tracking hands and objects poses new challenges due to the difficulty in segmenting hands from objects, and handling additional occlusions due to objects. We propose a multi-layered random forest architecture to address the segmentation problem and incorprate additional energy terms specific to the hand grasping objects. Once again, extensive evaluation and comparisons show that our method achieves high accuracy in spite of running at 30 FPS. To our knowledge, this is the first method to support *real-time* joint tracking of hands and objects.

1.4.2 Part II: Gesture-based Computer Input

In Part II, we contribute to **HCI** research by presenting new forms of gesture-based computer input enabled by markerless hand and finger tracking. In Chapter 7, we present our first approach to continuous gesture-based computer input. We show how gestures elicited from users (i.e., through **elicitation studies**) can be used to create interaction techniques suitable for 3D navigation tasks using purely freehand gestures. User studies indicated that our interaction techniques were comparable to existing techniques supported by devices like the mouse. Elicitation studies, however, have limitations which we discuss.

Informed by the lessons learned in creating continuous freehand gestures, we present an approach for **computational gesture design** in Chapter 8. Computational gesture design refers to the process of automatically designing gestures for an interaction task to suit designer-specified criteria. We present one of the first approaches for computational gesture design which is informed by the characteristics of hand trackers such as the those presented in Part I. We base our computational approach on data about the dexterity of the hand which includes speed and accuracy of the movement of fingers, comfortable motion ranges of fingers, and individuation of fingers. Our investigation was informed by an extensive user study that measured the components of dexterity in the context of markerless hand tracking. We present design recommendations based on the data we collected. We show how the data on dexterity can be used to inform the computational design of mid-air gestures. In particular, we focus on mid-air text entry and show that an approach similar to fingerspelling can lead to predicted text entry rates of over 50 words per minute (WPM). We formulate mid-air text entry as a *combinatorial optimization problem* and show that our data can drive the optimization of gestures based on criteria chosen by the designer. We finally present validation of the approach on users. Although we applied our approach to a discrete input task (i.e., text entry) our dexterity model is broadly applicable to continuous input tasks such as 3D navigation or pointing.

Finally, Chapter 9 discusses combining on- and above-skin input in the context of small form factor devices such as smartwatches. Interaction on devices with small displays poses problems because of small touch area and occlusions, the so called *fat finger* problem. We present an approach, called *WatchSense*, that supports extending the input space to areas around wearable devices like smartwatches. Our prototype takes a lightweight approach to hand tracking eschewing full hand pose estimation and instead relying on fingertip and touch detection. *WatchSense* enables tracking fingertip positions near the back of the hand (BOH) close to a smartwatch. We also support detection of touch points on the BOH which can be used to create a rich set of expressive gesture-based interaction. This enables, to our knowledge for the first time, simultaneous mid-air and multitouch gestures on the BOH. We show through technical evaluations and applications that our approach is accurate, robust, and does indeed provide benefits for more expressive interaction.

Chapter 10 concludes the thesis with a critical discussion of the limitations of our tracking and gesture-based input contributions, several directions for future work, and concludes the thesis with some final thoughts.

1.5 List of Publications

Some of the contributions in Part I of the thesis were originally reported in the following publications.

1. **Srinath Sridhar**, Antti Oulasvirta, Christian Theobalt. *Interactive Markerless Articulated Hand Motion Tracking using RGB and Depth Data.* International Conference on Computer Vision 2013 (**ICCV** 2013 [131]).
2. **Srinath Sridhar**, Helge Rhodin, Hans-Peter Seidel, Antti Oulasvirta, Christian Theobalt. *Real-time Hand Tracking Using a Sum of Anisotropic Gaussians Model.* International Conference on 3D Vision 2014 (**3DV** 2014 [132]).
3. **Srinath Sridhar**, Franziska Mueller, Antti Oulasvirta, Christian Theobalt. *Fast and Robust Hand Tracking Using Detection-Guided Optimization.* Conference on Computer Vision and Pattern Recognition 2015 (**CVPR** 2015 [129]).
4. **Srinath Sridhar**, Franziska Mueller, Michael Zollhöfer, Dan Casas, Antti Oulasvirta, Christian Theobalt. *Real-time Joint Tracking of a Hand Manipulating an Object from RGB-D Input.* European Conference on Computer Vision 2016 (**ECCV** 2016 [130]).

Some of the contributions in Part II of the thesis were originally reported in the following publications.

1. **Srinath Sridhar**, Anna Maria Feit, Christian Theobalt, Antti Oulasvirta. *Investigating the Dexterity of Multi-Finger Input for Mid-Air Text Entry*. SIGCHI Conference on Human Factors in Computing Systems 2015 (**CHI** 2015 [127]).
2. **Srinath Sridhar**, Gilles Bailly, Elias Heydrich, Antti Oulasvirta, Christian Theobalt. *FullHand: Markerless Skeleton-based Tracking for Free-Hand Interaction*. MPI-I-2016-4-002. Saarbrücken: Max-Planck-Institut für Informatik 2016 ([126]).
3. **Srinath Sridhar**, Anders Markussen, Antti Oulasvirta, Christian Theobalt, Sebastian Boring. *On- and Above-Skin Input Sensing through a Wearable Depth Sensor*. MPI-I-2016-4-003. Saarbrücken: Max-Planck-Institut für Informatik 2016 ([128]).

Part I

Tracking Hands in Action

Chapter 2

Problem Definition and Preliminaries

In Part I of this thesis, we deal exclusively with markerless computer vision-based tracking of hands in action. The ultimate goal of hand tracking is to be able to detect and track hands for different users, under general conditions using a single camera. This is an extremely challenging problem due to fast motions, occlusions, changing lighting conditions, scene clutter, uniform skin color, and the relatively small size of the hand in images. Additionally, methods have to run in real-time with low latency for use in HCI.

Given the challenges, we start by solving a relatively less challenging problem of *interactive* hand tracking from multiple RGB cameras in Chapter 3. In subsequent chapters we address increasingly harder problems by improving accuracy, imposing runtime constraints, reducing the number of cameras, and increasing scene complexity. Chapter 4 addresses more accurate multi-camera tracking in real-time using a new input and model representation. Chapter 5 further adds the single camera constraint and shows real-time results at 50 FPS. Finally, Chapter 6 looks into tracking hands together with objects in cluttered environment.

In this chapter, we formally define the hand tracking problem, introduce mathematical concepts and terminology that are used in the rest of the thesis.

2.1 Problem Definition and Terminology

We assume that the term *hand tracking* implies *markerless* tracking of the **full articulated 3D posture** of the hand and fingers without the use of gloves or reflective markers. This explicit definition is essential since many previous work that track only the position of the hand as a whole (a much easier problem) also use the term hand tracking. We also assume that a non-contact vision-based aproach is preferable to a contact-based approach (e.g., exoskeletons) for convenient gesture-based input. The input to such a hand tracker can be in

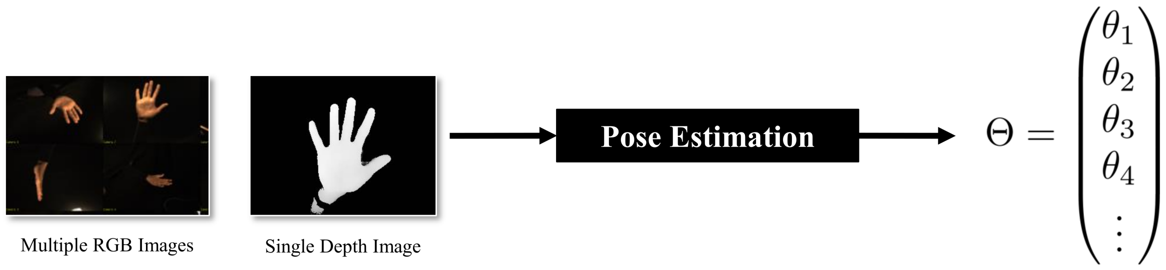

Multiple RGB Images
Single Depth Image
Pose Estimation
$\Theta = \begin{pmatrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\\ \vdots\end{pmatrix}$

Figure 2.1 The goal of markerless *hand tracking* is to estimate the position and orientation (pose) of the hand given input in the form of RGB and/or depth images. Hand pose is usually parametrized by joint angles and can be represented as a vector Θ.

the form of many color and/or a single depth image. The goal is to estimate the position and orientation (**pose**) of the hand given input images (see Figure 2.1). In order to denote hand pose, we need a formal representation that can be used mathematically as well as to represent the articulations of the hand. To this end, we use a hierarchical **kinematic skeleton** representation [90].

Kinematic Skeleton: A kinematic skeleton is a hierarchy of 3D rigid transforms where each transform represents an equivalent to a joint in the human hand. In order to capture the articulations of the hand we model it as a kinematic chain consisting of 32 joints (see Figure 2.2). We model the 26 DOFs of the hand using parameters $\Theta = \{\theta_i\}$, where $0 \leq i \leq 25$ (20 joint angles, 3 global rotations, and 3 global translations). Each joint angle is limited to a fixed range, $\theta_i \in [l^i_{min}, l^i_{max}]$, taken from studies of the hand [123].

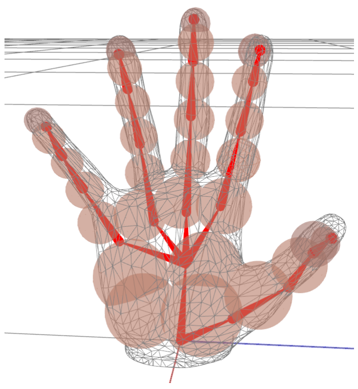


Figure 2.2 Kinematic skeleton of the hand (red). We use 32 joints and 26 DOFs.

Given this formal representation for hand pose, we now have the tools for pose estimation. Pose estimation methods in literature can be broadly classified into **generative** and **discriminative** approaches.

Generative Methods: Generative methods employ a hand model (e.g., kinematic skeleton) and synthesize a pose for the model that best explains the input (e.g., [86, 96]). For instance, Oikonomidis et al. [99] used a depth sensor and a model of the hand for tracking. Generative methods usually employ **pose optimization** techniques for convergence to the correct pose (e.g., particle swarm optimization in [99]). In this thesis, we use a Gaussian mixture formulation [57] and gradient-based methods for pose optimization.

Discriminative Methods: Discriminative methods use prior knowledge about hands (e.g., pose database) and find the closest example in this knowledge base. There are several

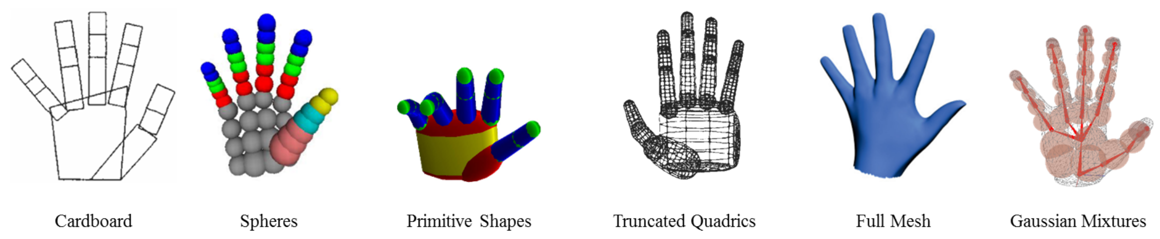


Figure 2.3 Different shape representations used in tracking. **Left to right**: 2D cardboard [163], spheres [108], collection of primitive shapes [96], truncated quadrics [133], full mesh [13], Gaussian mixtures (Chapters 3–6).

ways of achieving the closest search such as using a database lookup (e.g., [153]) or using machine learning (e.g., [119]). More detailed reviews of related work can be found within every chapter in Part I.

2.1.1 Shape Representations for Tracking

Generative methods usually employ a *model* of the hand which includes the kinematic skeleton to control articulation and, additionally, a shape attached to the skeleton. In computer graphics, it is common to use a *mesh* that is attached to the skeleton and deforms with it (i.e., a rigged mesh). However, using a full mesh for generative tracking can be computationally expensive, so many methods resort to simpler representations. Figure 2.3 shows some shape representations that have been used in previous work.

Some of the earliest works in hand tracking used simple 2D shapes to represent hand shape [163]. Using a full mesh [13] or superquadrics [133] could lead to better model fitting, but computational overheads prevent their widespread adoption. Many methods use approximations of the hand volume with primitive shapes such as spheres [108] or cylinders [96].

In this thesis, we use a Gaussian mixtures representation that has several advantages compared to previous work. As shown in Figures 2.2 and 2.3, we approximate the volumetric extent of the hand with a collection of un-normalized volumetric Gaussians. Each Gaussian in the collection can be modeled as an isotropic (i.e., uniform variance, see Chapters 3, 5, 6) or anisotropic Gaussian (see Chapter 4). Together this collection can be represented as a Gaussian mixture, $\mathcal{C}$. The Gaussian mixture is rigidly attached to the underlying kinematic skeleton and moves with it. Although we visualize the Gaussians as spheres or ellipsoids (see Chapter 4), they have infinite support. In some parts of the thesis, we use the terms Sum of Gaussians (SoG) or Sum of Anisotropic Gaussians (SAG) to refer to isotropic or anisotropic Gaussian mixtures, respectively.

Gaussian mixtures are well suited for pose estimation because they have a mathematically continuous representation that makes optimization more convenient i.e., we can compute analytical gradients for an appropriately defined objective function. Additionally, only a few Gaussians are sufficient to represent shape as opposed to thousands of vertices for a full mesh model. Finally, as we show next, Gaussian functions have favorable mathematical properties that make them well suited for transformations such as perspective projection, and comparison with other similar functions.

2.1.2 Properties of Gaussian Distributions

In this section, we provide some basic properties of Gaussian functions useful for better understanding of the different methods we present in Part I of this thesis.

Product of Two Gaussian Distributions: Let normalized Gaussian functions in k-dimensions be represented as

$$N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right], \tag{2.1}$$

with $k = \dim(\boldsymbol{\Sigma})$. Here the mean is $\boldsymbol{\mu}$ and the covariance matrix is $\boldsymbol{\Sigma}$. This Gaussian can be visualized as a k-dimensional ellipsoid.

The product of two Gaussians $N(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p) \cdot N(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ with means $\boldsymbol{\mu}_p, \boldsymbol{\mu}_q$ and covariance matrices $\boldsymbol{\Sigma}_p, \boldsymbol{\Sigma}_q$, respectively, is given as [3]

$$N(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p) \cdot N(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) = c \cdot N(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{2.2}$$

where $\boldsymbol{\Sigma}_c = \left(\boldsymbol{\Sigma}_p^{-1} + \boldsymbol{\Sigma}_q^{-1}\right)^{-1}$ and $\boldsymbol{\mu}_c = \boldsymbol{\Sigma}_c(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\mu}_p + \boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q)$ and normalization constant

$$c = \frac{1}{\sqrt{|2\pi(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q)|}} \exp\left[-\frac{1}{2}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q)^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)\right]. \tag{2.3}$$

Integration of the Product of Two Gaussian Distributions: Using the above result and $\int_\Omega N(\boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\mathbf{x} = 1$, it follows that the integral of two normalized Gaussians is $\int_\Omega N(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p) \cdot N(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)\, d\mathbf{x} = \int_\Omega c N(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)\, d\mathbf{x} = c$.

If we need to compute the same similarity measure for un-normalized Gaussians of the form

$$G(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p) := \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right], \tag{2.4}$$

we need to account for the missing constant. Considering this and using the fact $|2\pi\,\boldsymbol{\Sigma}| = (2\pi)^k|\boldsymbol{\Sigma}|$, the integral of two un-normalized Gaussians is given as

$$\int_\Omega G(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p) \cdot G(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)\, d\mathbf{x} = c\sqrt{(2\pi)^k|\boldsymbol{\Sigma}_p|}\sqrt{(2\pi)^k|\boldsymbol{\Sigma}_q|} = c\sqrt{(2\pi)^{2k}|\boldsymbol{\Sigma}_p\,\boldsymbol{\Sigma}_q|}, \quad (2.5)$$

where c is as defined in the normalized Gaussians case and $k = \dim(\boldsymbol{\Sigma}_p) = \dim(\boldsymbol{\Sigma}_q)$. This provides an efficient formula for measuring the similarity of two general Gaussians which is given as

$$E_{pq} = \frac{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}_p\,\boldsymbol{\Sigma}_q|}}{\sqrt{|(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q)|}} \exp\left[-\frac{1}{2}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q)^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)\right]. \quad (2.6)$$

The similarity measure and properties described above will find use in Chapters 3–6.

2.1.3 Perspective Projection of Gaussian Functions

In many of the methods presented in this thesis, we project Gaussian functions from 3D to 2D using a perspective camera projection model. Without loss of generality, we assume that this projection operation on Gaussian functions is equivalent to projection of a *general ellipsoid*, which represents the isosurface of a general Gaussian function at 1 standard deviation along each dimension. In Chapters 3, 5, and 6, we assume that the Gaussian functions are isotropic (i.e., can be visualized as spheres instead of ellipsoids). In Chapter 4, we use anisotropic Gaussians (i.e., can be visualized as ellipsoids). In this section, we show the general case of ellipsoid projection which subsumes sphere projection as well.

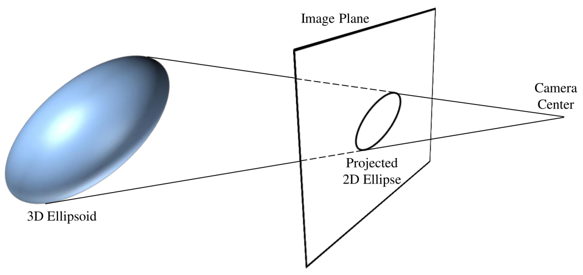


Figure 2.4 Sketch of the perspective projection of ellipsoids as the intersection of the image plane with the cone formed by the camera center and the ellipsoid.

The perspective projection of an ellipsoid is an ellipse defined by the intersection of the elliptical cone, formed by the rays originating from the camera center and tangential to the ellipsoid, with the image plane (see Figure 2.4). The projection equation is best explained in four separate steps. We assume a perspective pin hole camera model for this projection.

World–Camera Transformation: The extrinsic camera parameters are the orientation $\mathbf{R}_{wc}$ and position $\mathbf{c}$ of the camera. They transform the ellipsoid $(\boldsymbol{\Sigma}_h, \boldsymbol{\mu}_h)$ to the camera coordinate system by

$$\begin{aligned} \boldsymbol{\Sigma}_c &= \mathbf{R}_{wc}\, \boldsymbol{\Sigma}_h\, \mathbf{R}_{wc}^T \\ \boldsymbol{\mu}_c &= \mathbf{R}_{wc}(\boldsymbol{\mu}_h - \mathbf{c}), \end{aligned} \tag{2.7}$$

such that the origin is at the camera center and the z direction is aligned with the camera view direction.

Construction of Elliptical Cone: We are interested in a mathematical expression for the elliptical cone that is formed by the rays originating at the camera center **c** and tangential to the ellipsoid (see Figure 2.4). According to [35] all points on this cone satisfy

$$\mathbf{x}^\top \mathbf{M} \mathbf{x} = 0, \tag{2.8}$$

where the *cone matrix* **M** is

$$\mathbf{M} = \boldsymbol{\Sigma}_c^{-1} \left(\boldsymbol{\mu}_c - \mathbf{c}\right) \boldsymbol{\mu}_c^\top\, \boldsymbol{\Sigma}_c^{-1} - \left(\boldsymbol{\mu}_c^\top\, \boldsymbol{\Sigma}_c^{-1}\, \boldsymbol{\mu}_c - 1\right)\, \boldsymbol{\Sigma}_c^{-1}. \tag{2.9}$$

Intersection of the Elliptical Cone with the Image Plane: The points that form the projected ellipsoid on the canonical image plane I are those points that satisfy both Equation 2.8 and the image plane equation (see Figure 2.4). For a canonical image plane, the image plane equation is $z = 1$. We can derive an expression for the intersection of I and Equation 2.8 as follows.

The second degree polynomial representation of a conic section is given as [6]

$$px^2 + qxy + ry^2 + sx + ty + u = 0,$$

where $\mathbf{x} = [x, y, 1]^T$. The above equation is equivalent to Equation 2.8 where **M** can be written as

$$\mathbf{M} = \begin{bmatrix} p & q/2 & s/2 \\ q/2 & r & t/2 \\ s/2 & t/2 & u \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_2 & m_4 & m_5 \\ m_3 & m_5 & m_6 \end{bmatrix}. \tag{2.10}$$

Here m_k represent the elements of the symmetric matrix **M**. Let $\mathbf{M}_{33}$ represent the 2×2 submatrix of **M** excluding the 3^{rd} row and 3^{rd} column. The canonical parameters of the ellipse are given by

$$\tilde{\boldsymbol{\mu}}_p = \frac{1}{(4pr - q^2)} \begin{bmatrix} (qt - 2rs) \\ (sq - 2pt) \end{bmatrix} = \frac{1}{|\mathbf{M}_{33}|} \begin{bmatrix} |\mathbf{M}_{31}| \\ -|\mathbf{M}_{23}| \end{bmatrix}, \tag{2.11}$$

$$\widetilde{\boldsymbol{\Sigma}}_p = -\frac{|\mathbf{M}|}{|\mathbf{M}_{33}|}\mathbf{M}_{33}^{-1}. \tag{2.12}$$

For a general camera with intrinsics matrix **K** (as defined in [50]), the projected ellipse $(\widetilde{\boldsymbol{\Sigma}}_p, \tilde{\boldsymbol{\mu}}_p)$ from the canonical image plane is transformed to a general image plane. The transformed ellipse parameters are

$$\begin{aligned} \boldsymbol{\mu}_p &= \mathbf{K}_{33}\tilde{\boldsymbol{\mu}}_p + \begin{bmatrix} k_{13} \\ k_{23} \end{bmatrix}, \\ \boldsymbol{\Sigma}_p &= \mathbf{K}_{33}\, \widetilde{\boldsymbol{\Sigma}}_p\, \mathbf{K}_{33}^T. \end{aligned} \tag{2.13}$$

We utilize this ellipsoid projection formulation to project the 3D Gaussian mixture model to a 2D image Gaussian mixture.

2.1.4 Random Forests for Per-Pixel Classification

Several methods reported in this thesis rely on per-pixel classification of the input image. For this segmentation problem, we use per-pixel *classification forests* which have been shown to produce state-of-the-art results in human pose estimation and other segmentation problems [122, 64, 129]. We provide a brief overview and refer the reader to [30] for further details.

Figure 2.5 illustrates a sample *random decision forest* (or random forest). A random forest consists of many binary decision trees, each of which is trained on a random subset of the input data (hence the name random decision trees). Having an ensemble of decision trees helps improve generalization to unseen examples. At test time, input data points are passed from

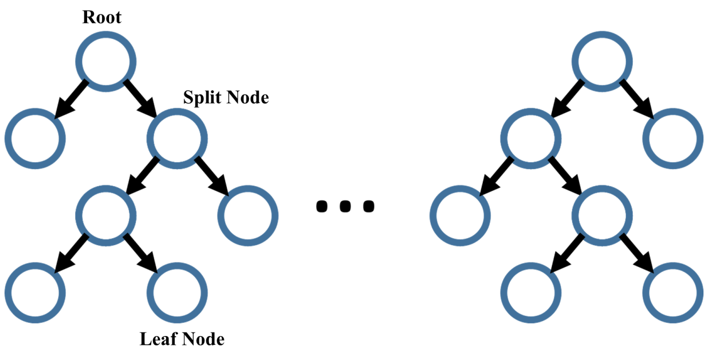


Figure 2.5 An ensemble of random decision trees forms a random forest.

the root node to a leaf node of a tree. At each split node, a decision is made about which child the data point must pass through. Therefore, at train time, decisions that need to be made at the split nodes are optimized. This binary decision made at a split node is called a *feature response* and a *weak learner* is employed to prevent overfitting. Arbitrary information about the data points can be stored at a leaf node. Typically, an empirical distribution about all the data points that reach a leaf node are stored.

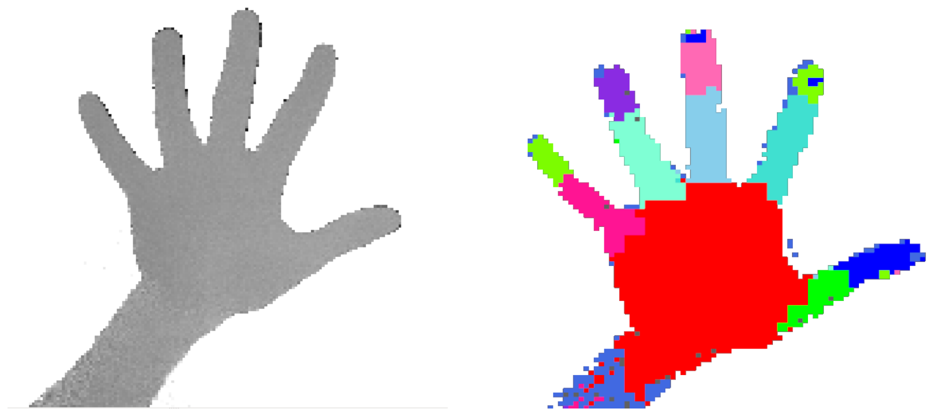


Figure 2.6 A depth image of the hand (left) is segmented into 12 hand parts with a depth classification forest.

In per-pixel classification forests, the goal is to train a forest to label each input pixel into a class label (e.g., part of a human body). At train time, the decisions at the split nodes are optimized based on thousands of training examples. For the task of depth-based classificaion we use the feature response function

$$f(I,\mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right),$$

where I is the input depth image, $\mathbf{x}$ is the pixel location, $\mathbf{u}$ and $\mathbf{v}$ are randomly chosen offsets from the current pixel location, and $d(.)$ denotes the depth at a certain location on the image. At test time, for each input pixel, a tree in the forest makes a prediction about which part it likely belongs to (see Figure 2.6). The output from all trees in the forest is aggregated to provide a final prediction about the pixel's class as $p(c\,|\,I,\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c\,|\,I,\mathbf{x})$, where p is the predicted class distribution for the pixel $\mathbf{x}$ and T is the number of random trees that makes a prediction p_t.

Chapter 3

Interactive Multi-Camera Hand Tracking

Tracking hands in action has several applications in human–computer interaction, teleoperation, sign language recognition, and virtual character control among others. An ideal hand tracker that can be used for these applications is a markerless method that tracks hand motion in real-time, using a single camera under changing lighting and scene clutter. As a first step towards solving this hard problem, we address the relatively less difficult problem of markerless, **interactive** (i.e., tracking at near-real-time framerates), *multi-camera* hand tracking in this chapter. Parts of this chapter appeared in a previous publication [131]. In subsequent chapters, we show how to solve hand tracking under progressively harder scenarios such as faster runtime, less cameras, and more complex scenes.

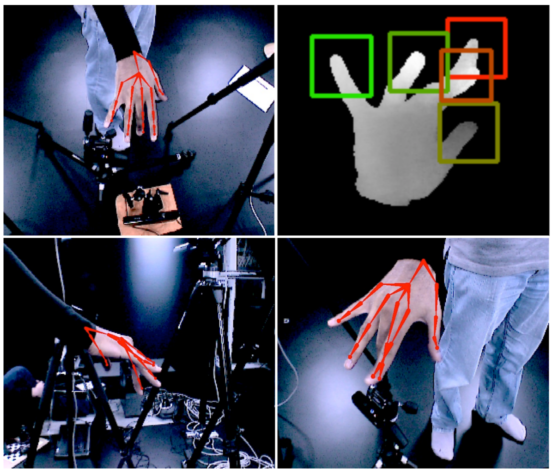


Figure 3.1 Our approach combines two methods: (1) Generative pose estimation on multiple RGB images using local optimization (bottom row and top left) (2) Part-based pose retrieval on five finger databases indexed using detected fingertips on a single depth image (top right).

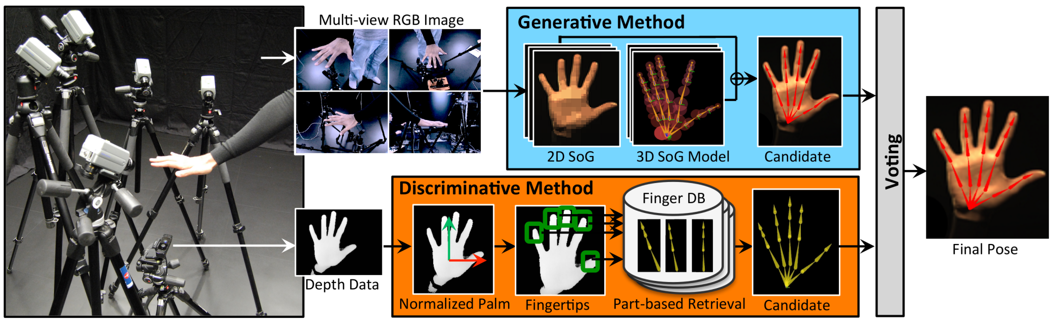


Figure 3.2 Overview of our interactive multi-camera hand tracking approach. SoG stands for Sum of Gaussians.

3.1 Introduction

Interactive markerless **tracking** of *articulated hand motion* is an important problem with a wide range of applications. *Marker* or *glove*-based solutions exist for tracking the articulations of the hand [153], but they constrain natural hand movement and require extra user effort. Recently, many commercial sensors have been developed that detect 3D fingertip locations without using markers but these sensors do not recover a semantically meaningful skeleton model of the hand. In this chapter, we describe a novel *markerless* hand motion tracking method that captures a broad range of articulations in the form of a *kinematic skeleton* at near-realtime frame rates.

Hand tracking is inherently hard because of the large number of degrees of freedom (DoF) [59], fast motions, self-occlusions, and the homogeneous color distribution of skin. Most previous realtime markerless approaches (see Section 3.2) capture slow and simple articulated hand motion since reconstruction of a broader range of complex motions requires offline computation. Our algorithm follows a hybrid approach that combines a generative pose estimator with a discriminative one (Figure 3.1). The input to our method are RGB images from five calibrated cameras, depth data from a monocular time-of-flight (ToF) sensor and a user-specific hand model (Section 3.3). The output of our method are the global pose and joint angles of the hand represented using 26 parameters.

Our approach is inspired by the robustness and accuracy of recent hybrid methods for realtime full-body tracking [7]. However, using the same strategy for hand tracking is challenging because of the absence of sufficiently discriminating image features, self-occlusions caused by fingers, and the large number of possible hand poses.

Figure 3.2 gives an overview of our algorithm. We use multiple co-located RGB cameras and a depth sensor as input to our method. Similar to previous work in full-body motion tracking [7, 156, 167], we instantiate two pose estimators in parallel. First, the generative pose estimator uses local optimization and a similarity metric based on the Sum of Gaussians (SoG) model [135] to find the pose that best explains the input RGB images (Section 3.4). Second, the discriminative pose estimator is a *part-based retrieval technique* that allows us to recover poses spanning a large hand articulation space while dealing with self-occlusions. Our discriminative pose estimation method first detects fingertips on the depth image from a single depth sensor using a linear SVM classifier (Section 3.5.3). The detected fingertips are then used in a hypothesize-and-test framework along with five finger pose databases to obtain multiple pose hypotheses, each of which is tested using two criteria (Section 3.5.4). The final (complete or partial) hand pose is the pose that has the least error between the estimated and observed fingertip positions. This is then used as initialization for local optimization in the generative pose estimator. This part-based approach reduces the database size dramatically as only the articulations of each finger need to be indexed. The evidence from both pose estimators are fused using an error metric to obtain a final hand pose (Section 3.6).

To critically assess our method, we report evaluations using challenging, kinesiologically motivated datasets. While there are numerous benchmark datasets for full-body pose estimation, we know of none for hand motion tracking. We therefore created seven annotated datasets recorded using multiple calibrated sensors. The motions cover the full abduction–adduction and flexion–extension ranges of the hand. Quantitative results show that we can cover a broad range of motions with an average error of around 13 mm. Our approach compares favorably in terms of accuracy and computational cost to a previous state-of-the-art approach [99]. To sum up, the primary contributions of this chapter are:

- A hybrid approach that combines a generative pose estimator based on local optimization with a novel part-based pose retrieval strategy.
- A near-real-time framework that captures hand motions (from multiple RGB cameras and a depth sensor) with a level of precision and speed necessary for interactive applications.
- An extensive, annotated benchmark dataset consisting of general hand motion sequences.

3.2 Related Work

One of the first kinematics-based hand motion tracking methods was presented by Rehg and Kanade [111]. The first study of size of the motion space of hand articulations when using kinematic skeletons was done by Lin et al. [78, 163]. They identified three types of constraints: joint angle limits (type I), intra-finger constraints (type II) and naturalness of hand motion (type III). Subsequent surveys of vision-based hand tracking methods [37] have divided methods into two categories—generative methods based on local or global optimization and discriminative methods based on learning from exemplars or exemplar pose retrieval.

Generative Methods: Oikonomidis et al. [99] presented a method based on particle swarm optimization for full DoF hand tracking using a depth sensor. They reported a frame rate of 15 fps with GPU acceleration. Other generative approaches have been proposed that use objects being manipulated by the hand as constraints [46, 47, 96, 114]. One such approach by Ballan et al. [13] used discriminatively learned salient features on fingers along with edges, optical flow, and collisions in an optimization framework. However, this method is unsuitable for interactive applications due to its large computation time. Other model-based global optimization approaches suffer from the same runtime performance problem [80, 133].

Discriminative Methods: A method for 3D hand pose estimation framed as a database indexing problem was proposed by Athitsos and Sclaroff [5]. Their method used a database of 26 hand shapes and a chamfer distance metric to find the closest match of a query in the database. The idea of using a global pose retrieval from a database of hand poses was explored by Wang et al. [152, 153]. However, in order to cover the whole range of hand motions the size of the database required would be large. Keskin et al. [63] proposed a method for hand pose estimation by hand part labeling but not as a kinematic skeleton.

Full-Body Motion Tracking: Given the similarity, volume, and success of existing research in full-body tracking, it would be natural to adopt one of those techniques for hand motion tracking. Several methods produce a 3D mesh and/or kinematic skeleton as their output [88, 107]. Some techniques, such as Stoll et al. [135], rely on multiple RGB cameras while many others use depth information from time-of-flight (ToF) or structured light depth cameras [7, 41, 121]. However, direct application of these methods to hand tracking is not straightforward because of homogeneous skin color, fast motions, and self-occlusions.

Our approach takes inspiration from hybrid approaches to full-body pose estimation, such as Ye et al. [167], Baak et al. [7], and Wei et al. [156]. However, our discriminative pose estimator uses a *part-based* pose retrieval technique as opposed to global pose retrieval.

3.3 Input Data and Hand Modeling

Figure 3.2 shows our setup consisting of multiple RGB cameras and a monocular ToF depth sensor. The image data from RGB cameras provides high visual accuracy for tracking. The complementary single-view depth data helps us to retrieve poses effectively, as we can resolve depth ambiguities and detect fingertip features in the 2.5D data. Retrieval efficiency is also supported by having to consider monocular image data only.

RGB Images: We use multiple, synchronized, and calibrated cameras to obtain RGB image data. We position n_k cameras in an approximate hemisphere such that typical hand motions within this hemispherical space would be visible in multiple cameras. All cameras are calibrated to obtain both the intrinsic and extrinsic parameters. We denote the RGB image produced by each camera as I_r^k. In all our experiments we used five Sony DFW-V500 cameras set at a resolution of 320 $\times$ 240 and a frame rate of 30 fps.

Depth Data: The other input to our method comes from a single time-of-flight (ToF) depth camera. The ToF camera is placed such that the hand motion space is within its range and is extrinsically calibrated along with the RGB cameras. We denote the depth image produced by the ToF camera as I_d and the unprojected point cloud representation of the scene as C_d. We used the Creative Interactive Gesture Camera as our ToF depth data sensor.

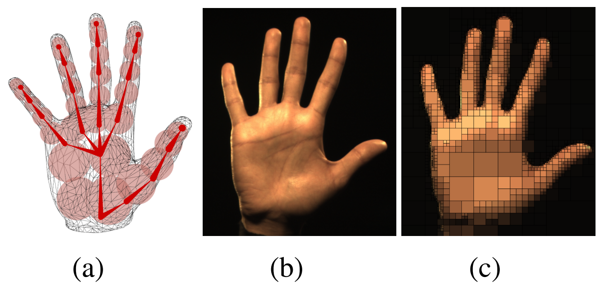


Figure 3.3 (a) Hand model consisting of a kinematic skeleton and attached 3D Gaussians visualized as spheres with a radius of 1 standard deviation. (b, c) Quadtree clustering of input image into 2D SoG.

Hand Modeling: In order to capture the articulations of the hand, we model it as a kinematic chain consisting of 32 joints (see Figure 3.3). We model the 26 degrees-of-freedom (DoF) of the hand using parameters $\Theta = \{\theta_i\}$, where $0 \leq i \leq 25$ (20 joint angles, 3 global rotations, and 3 global translations). Each joint angle is limited to a fixed range, $\theta_i \in [l^i_{min}, l^i_{max}]$, taken from studies of the hand [123]. Since we use a SoG model based generative tracking approach, we also augment the kinematic skeleton with 30 uniform 3D Gaussians with a fixed mean, variance, and color (c.f. [135]). Finally, we attach a 3D mesh, $\mathcal{M}$, consisting of 1774 vertices to the skeleton. The final output of our method are the parameters Θ of the kinematic skeleton.

3.4 Generative Hand Pose Estimation

Generative tracking estimates the hand pose parameters Θ_G that best match a given set of n_k input RGB images according to a consistency energy. We adopt a local energy maximization approach similar to that of Stoll et al. [135] which we modified to account for hand motions which are different from full-body motion. In this approach both the hand

and the input measurements are modeled using a *Sum of Gaussians* (SoG) representation. SoGs are mathematically smooth, yield analytical expressions for the energy functional and its derivative thereby facilitating fast pose optimization. Our consistency energy is given as

$$\mathcal{E}(\Theta) = E(\Theta) - w_l E_{lim}(\Theta), \tag{3.1}$$

where $E(\Theta)$ is a model-to-image similarity measure (Section 3.4.1). The second term, $w_l E_{lim}(\Theta)$, is a soft constraint on skeleton joint limits and has the same formulation as Stoll et al.. The weight parameter w_l was set to be 0.1 in all of our experiments.

3.4.1 Model-to-Image Similarity Measure

Given a 3D SoG based model of the hand and multiple input RGB images, we want to have a measure of similarity between the model and the images. We approximate each image with a 2D SoG model by performing quadtree clustering into regions of similar color, and fitting a 2D Gaussian with an average color to each region (Figure 3.3). Given two $2D$ SoGs $\mathcal{K}_a$ and $\mathcal{K}_b$ with associated colors $\mathbf{c}$, their similarity is defined as [135],

$$\begin{aligned} E(\mathcal{K}_a, \mathcal{K}_b) &= \int_\Omega \sum_{i \in \mathcal{K}_a} \sum_{j \in \mathcal{K}_b} d(\mathbf{c}_i, \mathbf{c}_j) \mathcal{B}_i(\mathbf{x}) \mathcal{B}_j(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &= \sum_{i \in \mathcal{K}_a} \sum_{j \in \mathcal{K}_b} E_{ij}, \end{aligned} \tag{3.2}$$

where $\mathcal{B}(\mathbf{x})$ is an un-normalized Gaussian basis function

$$\mathcal{B}(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \boldsymbol{\mu}\|^2}{2\sigma^2} \right). \tag{3.3}$$

E_{ij} is the similarity between a pair of Gaussians $\mathcal{B}_i$ and $\mathcal{B}_j$ given their colors $\mathbf{c}_i$ and $\mathbf{c}_j$ and is defined as

$$\begin{aligned} E_{ij} &= d(\mathbf{c}_i, \mathbf{c}_j) \int_\Omega \mathcal{B}_i(\mathbf{x}) \mathcal{B}_j(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &= d(\mathbf{c}_i, \mathbf{c}_j) 2\pi \frac{{\sigma_i}^2 {\sigma_j}^2}{{\sigma_i}^2 + {\sigma_j}^2} \exp\left(-\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{2\left({\sigma_i}^2 + {\sigma_j}^2\right)} \right). \end{aligned} \tag{3.4}$$

The color similarity function $d(\mathbf{c}_i, \mathbf{c}_j)$ measures the Euclidean distance between $\mathbf{c}_i$ and $\mathbf{c}_j$ in the HSV color space and feeds the result into a Wendland function [160]. This renders d, a smooth function bounded in [0,1] (0 for dissimilar input and 1 for similar input).

Using the above defined similarity measure, we can find how similar a particular pose of the 3G SoG hand model is to the observed RGB images. To this end, the 3D Gaussians are projected onto the images using a projection operator $\Psi(\mathcal{K}_m)$ [135]. We now define the final similarity measure as

$$E_{sim}(\mathcal{K}_I, \mathcal{K}_m(\Theta)) = \sum_{i \in \mathcal{K}_I} \min \left(\left(\sum_{j \in \Psi(\mathcal{K}_m)} w_j^m E_{ij} \right), E_{ii} \right),$$

where w_j^m is a weighting factor for each projected 3D Gaussian $\Psi(\mathcal{K}_m)$. With this parameter we control the relative influence of each 3D Gaussian on the final similarity.

To prevent overlapping projected 3D Gaussians from contributing multiple times in the above sum and distorting the similarity function, we clamp the similarity to be at most E_{ii}, which is the similarity of the image Gaussian with itself. This can be seen as a simple approximation of an occlusion term.

The offline step in this optimization method is to perform person-specific customization of the hand model's shape and dimensions, once for each actor. We adopt the *semi-automatic* process described by Stoll et al. [135] to our default hand skeleton template. We captured four static hand poses in which joints were clearly visible, and manually positioned our default hand skeleton to fit the poses. After this step, the position, variance, and color of the 3D Gaussians and bone lengths are optimized. This hand model is used in all stages of our method.

3.4.2 Optimization

The goal of the optimization step is to estimate the pose parameters Θ_t at every time instant. We adapted the gradient ascent local optimization method proposed by Stoll et al. which enables realtime estimation of the pose parameters at every time instant t, as analytical gradients can be computed for our energy function. Each iteration of the optimization is initialized by extrapolating the estimated pose from two previous times steps as

$$\Theta_0^t = \Theta^{t-1} + \alpha(\Theta^{t-1} - \Theta^{t-2}), \tag{3.5}$$

where α is set to 0.5. In Section 3.5, we describe how our part-based pose retrieval strategy can be used to initialize the optimization.

Even though the generative pose optimization method is fast and proven to be reliable for full-body tracking, it quickly reaches its limits during hand tracking and fails by converging to local pose optima from which it cannot recover. This is because the hand exhibits a higher articulation complexity than the body (thus allowing for a much wider range of poses in a small space), faster motions, and homogeneous color. The consequences are frequent self-occlusions and large visible displacements of the hand between two frames which challenge a local pose optimizer. Furthermore, the uniform skin color of the bare hand makes model-to-image associations much more ambiguous than in the case of humans wearing colored clothing. We therefore complement our generative tracker with an efficient discriminative hand pose estimation algorithm described in the following sections. It generates hand pose hypotheses in parallel to the generative method and is able to re-initialize it in case of convergence to a wrong pose.

3.5 Part-based Pose Retrieval

The goal of our discriminative pose estimation method is to estimate a complete or *partial* pose, $\widetilde{\Theta}_D$, of the hand from a single depth image I_d. We do this by adopting a part-based strategy i.e., instead of trying to recover the full hand pose, we separately recover the pose of each finger Θ^f_D. This is achieved by extracting fingertips on the depth image using a linear SVM classifier, and by using the detected positions to find the closest match in multiple exemplar *finger pose databases*. Having separate databases for each finger has several advantages. First, for combinatorial reasons, the articulation space that we are able to represent in a pose database of necessarily limited size is much larger than when using a single pose database with exemplars for the entire hand (Section 3.5.1). Second, our approach has the advantage of being able to recover a partial hand pose (i.e., missing some finger poses) even when some of the fingers are occluded. The recovered finger poses are then assembled using a hypothesize-and-test framework to form a complete or partial pose $\widetilde{\Theta}_D$.

3.5.1 Multiple Finger Pose Database Generation

We briefly motivate the need for using multiple finger databases as opposed to a single global pose database. The global pose retrieval method of Wang and Popović [153] uses $18,000$ poses sampled from real hand motion. Although one of their goals was to avoid oversampling, the size of their database is still insufficient to span the range of articulations that can occur in natural motion. One way to quantitatively assess the relationship between the range of articulations and the size of the database is to consider discretizations of joint angles within allowable joint limits. Ignoring global motion, we model the hand using 21 joint angles (DoFs). If each joint angle were discretized into 3, then for global pose retrieval the size of the database would be of the order of 10^{10}. On the other hand, part-based pose retrieval would need five databases, each with a size of 81. Thus, part-based pose retrieval results in much smaller databases for the hand than global pose retrieval. This prevents oversampling while still keeping the articulation space large.

Previous approaches [7, 153] that use global pose retrieval capture real data using motion capture systems for generating a pose database. However, complex hand motions are difficult to capture using mocap systems because of self-occlusions and glove constraints. We therefore obtain our finger pose database by synthetically generating the poses over discretizations of all joint angles for each finger. To this end we use the person-specific model of the hand obtained earlier (Section 3.4.1)

For each synthetic pose generated per finger, Θ_S^f, we compute the end effector position $\mathbf{x}_s^f$ with respect to a local skeleton coordinate system (see Section 3.5.2). We use the computed 3D end effector position as our database indexing feature since it uniquely identifies a pose of the finger and can be detected comparatively easily on depth data. We use a k-d tree for indexing the features. In all our experiments we used a database size of 4096 corresponding to a joint discretization of 8 levels per DoF.

3.5.2 Palm and Hand Orientation Estimation

Since our finger pose databases are indexed based on features relative to the hand model, we need to normalize the detected query features so that they lie in the same frame of reference. To this end, we extract the palm and its orientation from the depth data. We first apply a box filter on the depth image I_d to extract the depth image, I_b, and unprojected point cloud, C_b, corresponding to the hand only. We use the morphological operations erode and dilate on I_b to remove fingers but retain the palm. The result is a binary mask of the palm which is used to obtain a basic segmented point cloud of the palm, C_s. However, C_s might contain fingers that lie on the line of sight between the sensor and the palm. We therefore fit a plane, P, to C_s using RANSAC with a consensus threshold of 5 mm to obtain the final segmented point cloud of the palm, C_p. We compute the center of the palm as the point that lies on P and is the centroid of the axis aligned bounding box of C_p. We then perform principal component analysis (PCA) of C_b projected onto the plane P to find the principal directions of the hand and palm. As a final step, we use a Kalman filter in order to

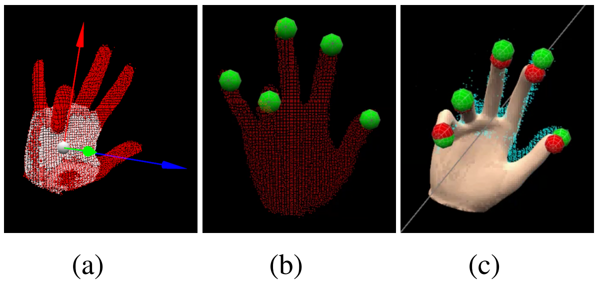


Figure 3.4 (a) Palm extracted from the point cloud (white) and hand orientation normalization (arrows). (b) Fingertips detected using a linear SVM classifier. (c) Estimated partial or complete hand pose.

reduce jitter in the estimated orientation. The detected palm center and orientations serve to stabilize the results of the finger pose database look up (see Figure 3.4).

3.5.3 Fingertip Detection

For our part-based pose retrieval strategy, we need to reliably detect the end effector positions in the depth data. Previous work in full-body pose estimation has used features such as Geodesic extrema [7, 105] which do not work well for the hand and result in spurious extrema which are difficult to disambiguate from the real extrema. In order to overcome this problem, we use a machine learning approach to detect fingertips using a linear SVM classifier and HOG descriptors as features. We follow the object detection framework of Dalal and Triggs [32] on depth images instead of RGB images. For training our linear SVM we used a combination of manually annotated real sequences, annotated synthetic sequences, and rotated versions of both (4 orientations). We use a fingertip detection window size of 32×32. Because of the high cost of not detecting a fingertip in the pose retrieval step we adjusted the parameters of the linear SVM for higher recall rates. We found that most false positives could be eliminated using assumptions about the position of the finger i.e., a fingertip cannot lie far away or too close to the center of the palm. After elimination, we obtain five or less fingertip candidate points $\mathbf{x}_c^f$. Figure 3.1 shows one depth frame with detected fingertips overlaid and Figure 3.4 shows the filtered fingertips on the point cloud.

3.5.4 Finger Pose Estimation

The final step of discriminative pose estimation is to find the complete or partial pose of the hand, $\widetilde{\Theta}_D$. However, in order to query the finger pose databases we would need to label each detected fingertip. This is a hard problem since there is tremendous variation in fingertip appearance in depth or RGB images. We instead adopt a hypothesize-and-test framework to test all elements in the set of permutations of labels, Σ, using two criteria. First, for each permutation $\sigma_i \in \Sigma$ we reject a hypothesized pose early based on the distance of each detected fingertip to the nearest neighbor in the finger pose database corresponding to the current labeling for that fingertip. We set a distance threshold $\mu = 20$ mm in all our experiments. Only those hypotheses that pass the first stage are tested with the distance measure which is given as

$$\delta(\sigma_i, \widetilde{\Theta}) = \frac{1}{r} \|\mathbf{x}_i - \mathbf{x}_c^f\|_2, \tag{3.6}$$

where r is the number of detected fingertips, $\mathbf{x}_i$ is the position of a fingertip corresponding to a candidate fingertip $\mathbf{x}_c^f$ and $\widetilde{\Theta}$ is the current hypothesis pose. The pose that has the lowest distance measure is selected as the best pose $\widetilde{\Theta}_D$. In the case of less than five detected fingertip locations, a partial pose with the lowest distance is still recovered since partial poses are also part of the permutations set Σ.

3.6 Pose Candidate Fusion

At this stage, we have two hand pose candidates, Θ_G and $\widetilde{\Theta}_D$, from the generative and discriminative methods. In order to combine them together to find the best pose, we first initialize a second instance of the generative tracker with $\widetilde{\Theta}_D$ instead of extrapolation. Those pose parameters that are not part of $\widetilde{\Theta}_D$ are extrapolated using Equation 3.5. Upon optimization we obtain the pose Θ_D and an associated optima energy $\mathcal{E}(\Theta_D)$. The final pose, Θ_F, is the pose that has the higher energy given by

$$\Theta_F = \underset{\Theta \in \{\Theta_G, \Theta_D\}}{\operatorname{argmax}} \{\mathcal{E}(\Theta_G), \mathcal{E}(\Theta_D)\}. \tag{3.7}$$

3.7 Results

We implemented and tested our method, algorithmic variants of it, and a related algorithm from literature [99] on a computer with a clock speed of 3.30 GHz, 8 GB of RAM, and an Nvidia NVS 300 GPU. On this machine, our method achieved an interactive frame rate of 10 fps. With our unoptimized C++ code, the most time consuming components were the local optimization for generative pose estimation (53 ms) and multiscale fingertip detection (40 ms).

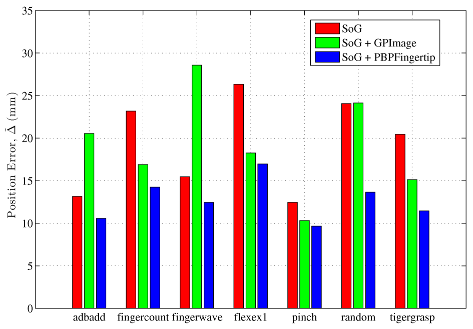
We will now present results from extensive experimental evaluation that we conducted using our method on a variety of sequences. Unlike previous approaches that used a combination of synthetic and real data for evaluation, we used a large corpus of real data. We collected seven real sequences consisting of synchronized and calibrated multi-view RGB images, as well as monocular ToF and Kinect data (see Figure 3.2). All sequences were manually annotated to mark fingertip and palm center positions in the depth data. In total, our test sequences consist of 2137 frames of data containing both slow and fast motions with a static background and general illumination conditions. The sequences that we captured span a range of hand movement from flexion–extension (e.g., `fingerwave`, `flexex1`, `pinch`, `fingercount`), abduction–adduction (e.g., `abdadd`), and included random motions (e.g., `random`, `fingerwave`).

Overall quantitative results from our experiments show that our approach of combining a generative pose estimation method with a discriminative *part-base pose retrieval* technique (**SoG + PBPFingertip**) performs better than other alternatives in most cases. Our algorithm is stable and all results were recorded using the same parameters. We compared our approach with several algorithmic alternatives—(a) generative pose estimation with SoG model only (**SoG**), (b) generative pose estimation method combined with a global pose retrieval technique based on normalized depth images (**SoG + GPImage**), and (c) publicly available implementation of the method proposed by Oikonomidis et al. [99] (**FORTH**, one sequence only).

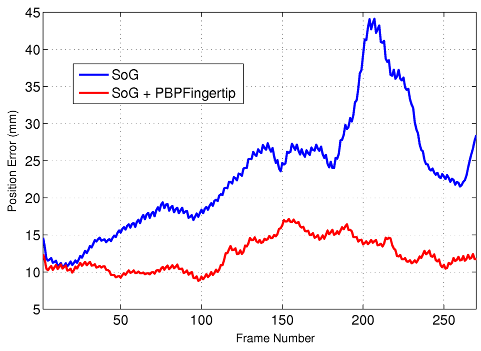
Evaluation Metric: To enable relative comparison with other methods we adopted an error metric similar to that used by Oikonomidis et al. [96]. The Euclidean distance between the estimated and ground truth fingertip positions and palm center positions are computed for each frame for all datasets. We find the average error, $\widetilde{\Delta}$, over all frames within each dataset.

Quantitative Results: Figure 3.5a compares our result (SoG + PBPFingertips) with using only SoG and SoG + GPImage. Our hybrid method produces better results and achieves an accuracy of 13.24 mm on average which is close to the best offline methods [13]. This can be attributed to the fact that each time generative pose estimation fails, the discriminative part-base pose retrieval strategy re-initializes it appropriately. This becomes clear in Figure 3.5b which shows the error as a function of the frame number. The error starts accumulating in the generative method at about frame 25 and never goes down. But our method periodically re-initializes so as to maintain a constant error rate even in long sequences. Most notably, towards the end of the sequence our method produces errors that are not too different from the first few frames. One surprising result here is that SoG + GPImage produces a higher error than SoG only which indicates that image based global retrieval is sensitive to noise in the depth data.

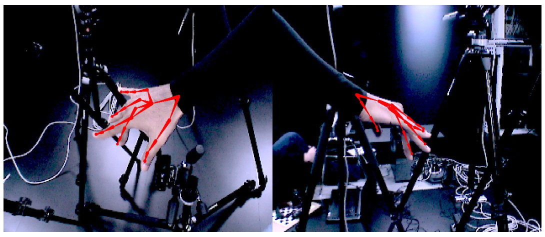
We also tested the FORTH method on a dataset containing motions similar to `fingerwave` but under different illumination conditions as we were unable to get their method to work on any one of our seven sequences. Their method is based on GPU acceleration and runs at only 2.5 fps compared to our 10 fps on the same machine indicating that optimization of our code could lead to faster frame rates. We then computed the error measure, $\widetilde{\Delta}$, for the FORTH method over the entire (similar) sequence and found it to be 10.31 mm. This compares favorably with the mean error of our method which was 13.24 mm. Thus, our method performs well for similar datasets while using less computational budget than FORTH.



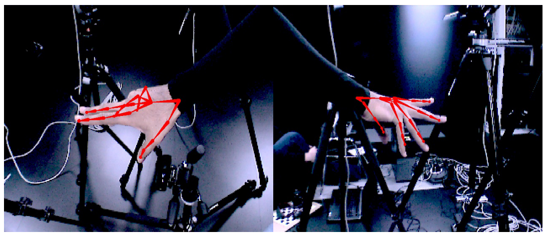
(a) The average position error over the entire dataset for SoG only, SoG + GPImage and SoG + PBPFingertips (ours).



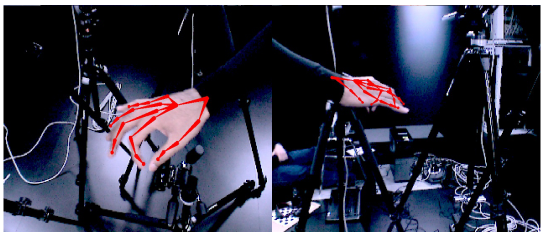
(b) The average position error over the `fingerwave` dataset for SoG only and SoG + PBPFingertip (ours).



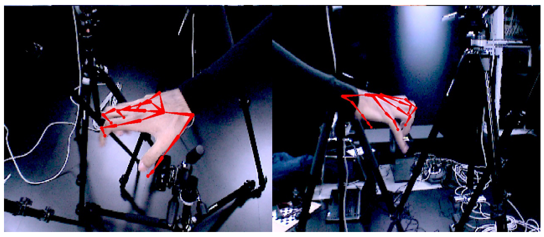
(a)



(b)



(c)



(d)

Figure 3.6 Qualitative results of our method as seen from two camera views. Results in (a), (b) show general slow motion. Results in (c) show successful tracking even in the presence of fast motion. Result (d) shows a failure case due to fast motion.

3.8 Discussion

In this chapter, we presented a novel method for tracking the full articulated 3D motion of the human hand using a hybrid method from multiple RGB and depth cameras. Our method advances the state of the art by demonstrating high accuracy across a large corpus of motions with a runtime that is sufficient for many interactive applications. Our main contribution was the use of a new method for part-based pose retrieval in conjunction with image-based pose optimization. Part-based pose retrieval enables recovery and stable tracking of poses with self-occlusions that are characteristic of hand motion, and enables a dramatic reduction of the pose database size. This allows our method to track difficult hand poses more accurately and robustly than previous approaches while running at interactive speeds.

3.9 Conclusion

Although our method achieves good performance on real sequences there is still room for improvement. In particular the accuracy and runtime of our method can be improved considerably. To improve accuracy, we will explore the use of new model and input representations in Chapter 4. Later in Chapter 7, we show how our method can also run considerably faster beause the SoG representation lends itself well for parallelization. Our calibrated multi-camera setup requires time to setup. Therefore, in Chapter 5, we explore reducing the number of cameras and incorporating depth data into generative pose optimization. In the next chapter, we discuss the use of novel representations for the input data and the hand model to achieve real-time tracking with more accuracy.

Chapter 4

Real-time Hand Tracking with Multiple RGB Cameras

In order for hand tracking to be useful for gesture-based computer input some key requirements are accuracy, real-time performance and low latency. Most approaches from the literature frequently fail to track even moderately fast and complex hand motion in real-time. High accuracy approaches tend to be slow. In this chapter, we present an approach that achieves accurate, real-time (>25 FPS) tracking for complex hand motion. Parts of this chapter appeared in a previous publication [132].

4.1 Introduction

Previous methods for hand tracking can be broadly classified into either *generative* methods [99, 133, 80] or *discriminative* methods [5, 153, 152, 63]. Generative methods usually employ a dedicated model of hand shape and articulation whose pose parameters are optimized to fit image data. While this yields temporally smooth solutions, real-time performance necessitates fast local optimization strategies which may converge to erroneous local pose optima of the non-convex objective function. In contrast, discriminative methods detect hand pose

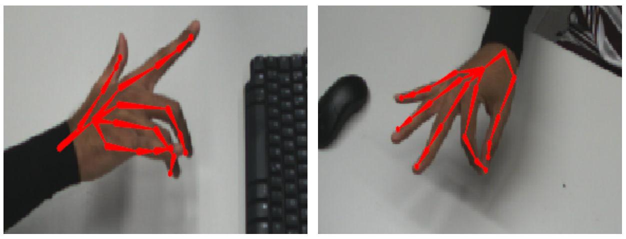


Figure 4.1 Qualitative results from our SAG-based tracking method. We achieve a framerate of 25 fps which is suitable for interactive applications.

from image features, e.g., by retrieving a plausible hand configuration from a learned space of poses, but the results are usually temporally less stable.

In the previous chapter, we presented a hybrid method that combines generative and discriminative pose estimation for hand tracking from multi-view video and a single depth camera (Chapter 3). In that work, generative tracking is based on an implicit Sum of Gaussians (SoG) representation of the hand, and discriminative tracking uses a linear SVM classifier to detect fingertip locations. This approach showed increased tracking robustness compared to prior work but was limited to using isotropic Gaussian primitives to model the hand.

In this chapter, we build on this previous method and further develop it to enable fast, more accurate, and robust articulated hand tracking at real-time rates of 25 fps using only multi-view RGB images. We contribute a fundamentally extended generative tracking algorithm based on an augmented implicit shape representation.

The original SoG model is based on the simplifying assumption that all Gaussians in 3D have **isotropic covariance**, facilitating simpler projection and energy computation. However, in the case of hand tracking this isotropic 3D SoG model reveals several disadvantages. Therefore we introduce a new 3D *Sum of Anisotropic Gaussians* (SAG) representation (Figure 4.3) that uses **anisotropic** 3D Gaussian primitives attached to a kinematic skeleton to approximate the volumetric extent and motion of the hand. This step towards a more general class of 3D functions complicates the projection from 3D to 2D and thus the computation of the pose fitting energy. However, it maintains important smoothness properties and enables a better approximation of the hand shape with less primitives (visualized as ellipsoids in Figure 4.3). Our approach, in contrast to previous methods ([135, 131], Chapter 3), models the full perspective projection of 3D Gaussians. To summarize, the primary contributions of this chapter are:

- An advancement of the method presented in Chapter 3 that generalizes the SoG-based tracking to one based on a new 3D Sum of Anisotropic Gaussians (SAG) model, thus enabling tracking using fewer primitives.
- Utilization of a full perspective projection model for projection of 3D Gaussians to 2D in matrix-vector form.
- Analytic derivation of the gradient of our pose fitting energy, which is smooth and differentiable, to enable real-time optimization.

We evaluate the improvements enabled by SAG-based generative tracking over previous work. Our contributions not only lead to more accurate and robust real-time tracking but also allow tracking of objects in addition to the hand.

4.2 Previous Work

Following the survey of Erol et al. [37], we review previous work by categorizing them into either *model-based tracking* methods or *single frame pose estimation* methods. Model-based tracking methods use a hand model, usually a kinematic skeleton with additional surface modeling, to estimate the parameters that best explain temporal image observations. Single frame methods are more diverse in their algorithmic recipes, they make fewer assumptions about temporal coherence and often use non-parametric models of the hand. Hand poses are inferred by exploiting some form of inverse mapping from image features to a space of hand configurations.

Model-based Tracking: Rehg and Kanade [111] were among the first to present a kinematic model-based hand tracking method. Lin et al. [78, 163] studied the constraints of hand motion and proposed feasible base states to reduce the search space size. Oikonomidis et al. [99] presented a method based on particle swarm optimization for full DoF hand tracking using a depth sensor and achieved a frame rate of 15 fps with GPU acceleration. Other model-based methods using global optimization for pose inference fail to perform at real-time frame rates [133, 80].

Primitive shapes such as spheres and (super-)quadrics have been explored for tracking objects [69], and, recently, for tracking hands [108]. However, perspective projection of complex shapes is hard to represent analytically and therefore fast optimization is hard. In this work we use anisotropic Gaussian primitives with analytical expression for perspective projection. An overview of perspective projection of spheroids, which are conceptually similar to anisotropic Gaussians, can be found in [35].

Tracking hands with objects imposes additional constraints on hand motion. Methods proposed by Hamer et al. [47, 46], and others [96, 114, 13] model these constraints. However, these methods require offline computation and are unsuitable for interaction applications.

Single Frame Pose Estimation: Single frame methods estimate hand pose in each frame of the input sequence without taking temporal information into account. Some methods build an exemplar pose database and formulate pose estimation as a database indexing problem [5]. The retrieval of the whole hand pose was explored by Wang and Popović [153, 152]. However, the hand pose space is large and it is difficult to sample it with sufficient granularity for jitter-free pose estimation. Sridhar et al. [131] proposed a part-based pose retrieval method to reduce the search space. Decision and regression forests have been successfully used in full body pose estimation to learn human pose from a large synthetic dataset [121]. This approach has been recently adopted for hands [63, 142, 165, 140]. These methods gener-

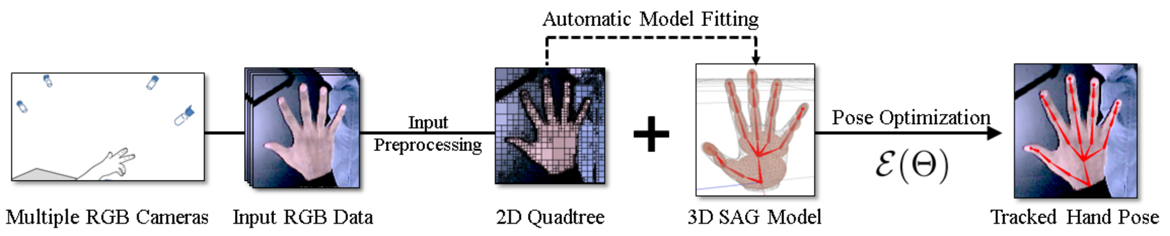


Figure 4.2 Overview of the our tracking framework. We present a novel generative tracking method that models the hand as a *Sum of Anisotropic Gaussians*. We obtain better accuracy and robustness than previous work.

ally lack temporal stability and recover only joint positions or part labels instead of a full kinematic skeleton.

Hybrid Tracking: Hybrid frameworks that combine the advantages of model-based tracking and single frame pose estimation can be found in full body pose estimation [167, 7, 156] and early hand tracking [115]. A hybrid method that uses color and depth data for hand tracking was proposed in Chapter 3 and [131]. However, this method is limited to studio conditions and uses isotropic Gaussian primitives. In this chapter, we extend this method by introducing an improved (model-based) generative tracker. This new tracker alone leads to higher tracking accuracy and robustness than the baseline method it extends while running in real-time.

4.3 Tracking Overview

Figure 4.2 shows an overview of our tracking framework. The goal is to estimate hand pose robustly by maximizing the similarity between the hand model and the input images. The tracker developed in this chapter extends the generative pose estimation method of the tracking algorithm in [131].

The input to our method are a set of RGB images from 5 video cameras (Point Grey Flea 3) of resolution 320×240 (see Figure 4.2). The cameras are calibrated and run at 60 fps. The hand is modeled as a full kinematic skeleton with 26 degrees-of-freedom (DOF), and unlike other methods that deliver only joint locations or part labels in the images, our approach computes full kinematic joint angles $\Theta^* = \{\theta_j^*\}$.

Our method maximizes the similarity between the 3D hand model projected into all RGB images, and the images themselves, by means of a fast iterative local optimization algorithm. The main novelty and contribution of this work is the new shape representation and pose optimization framework used in the tracker (Section 4.4). Our pose fitting energy is smooth, differentiable and allows fast gradient-based optimization. This enables real-time hand tracking with higher accuracy and robustness while using fewer Gaussian primitives.

4.4 SAG-based Generative Tracking

The generative tracker from Chapter 3 is based on a representation called a Sum of Gaussians (SoG) model, that was originally proposed in [135] for full body tracking (also see Chapter 2). The basic concept of the SoG model is to approximate the 3D volumetric extent of the hand by **isotropic Gaussians** attached to the bones of the skeleton, with a color associated to each Gaussian (Figure 4.3 (c)). Similarly, input images are segmented to regions of coherent color, and each region is approximated by a 2D SoG (Figure 4.4 (b-c)). A SoG-based pose fitting energy was defined by measuring the overlap (in terms of spatial support and color) between the projected 3D Gaussians and all the image Gaussians. This energy (Equation 4.9) is maximized with respect to the degrees of freedom to find the correct pose.

Unfortunately, a faithful approximation of the hand volume with a collection of isotropic 3D Gaussians often requires many Gaussian primitives with small standard deviation, a problem akin to packing a volume with spherical primitives. With SoG, this leads to sub-optimal hand shape approximation and increased computational complexity due to a high number of primitives in 3D (Figure 4.3). In this chapter, we extend the SoG model and represent the hand shape in 3D with **anisotropic Gaussians**, yielding a *Sum of Anisotropic Gaussians* model (see Figure 4.1). This not only enables a better approximation of the hand shape with less 3D primitives (Figure 4.3), but also leads to higher pose estimation accuracy and robustness. The move to anisotropic 3D Gaussians complicates their projection into 2D where scaled orthographic projection [135, 131] cannot be used. But we show that the numerical benefits of the SoG representation hold equally for the SAG model: 1) We derive a pose fitting energy that is smooth and analytically differentiable for the SAG model under perspective projection that allows efficient optimization with a gradient-based iterative solver; 2) We show that occlusions can be efficiently approximated with the SAG model within our energy formulation. This is in contrast to many other generative trackers where occlusion handling leads to discontinuous pose fitting energies.

4.4.1 Fundamentals of SAG Model

We represent both the volume of the hand in 3D, as well as the RGB images with a collection of anisotropic Gaussian functions. A *Sum of Anisotropic Gaussians* (SAG) model thus takes the form:

$$\mathcal{C}(\mathbf{x}) = \sum_{i=1}^{n} \mathcal{G}_i(\boldsymbol{\mu}_\mathbf{i}, \boldsymbol{\Sigma}_\mathbf{i}), \tag{4.1}$$

where $\mathcal{G}_i(.)$ denotes an un-normalized, anisotropic Gaussian

$$\mathcal{G}(\boldsymbol{\mu}_\mathbf{i}, \boldsymbol{\Sigma}_\mathbf{i}) := \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T {\boldsymbol{\Sigma}_i}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right], \tag{4.2}$$

with mean $\boldsymbol{\mu}_i$ and covariance matrix is $\boldsymbol{\Sigma}_i$ for the i^{th} Gaussian. Each Gaussian also has an associated average color vector $\mathbf{c}_i$ in HSV color space.

Using the above representation, we model the hand surface as a sum of 3D anisotropic Gaussians (3D SAG), where $\mathbf{x} \in \mathbb{R}^3$. We also approximate the input RGB images as a sum of 2D isotropic Gaussians (2D SoG), where $\mathbf{x} \in \mathbb{R}^2$. This is an extension of the SoG representation proposed earlier in [135, 131], which was limited to isotropic Gaussians.

3D Hand Modeling: We model the volumetric extent of the hand as a 3D sum of anisotropic Gaussians model (3D SAG), where $\mathbf{x} \in \mathbb{R}^3$. Each $\mathcal{G}_i$ in the 3D SAG is attached to one bone of the skeleton, and thus moves with the local frame of the bone (Figure 4.3). A linear mapping between skeleton joint angles and a pose parameter space, $\Theta = \{\theta_j\}$, is constructed. The skeleton pose parameters are further constrained to a predefined range of motion reflecting human anatomy, $\theta_j \in [l^j_{min}, l^j_{max}]$. The use of anisotropic Gaussians, whose spatial density is controlled by the covariances, enables us to approximate the general shape of the hand with less primitives than needed with the original isotropic SoG model (Figure 4.3). This is because we can create a better *packing* of the hand volume with more generally elongated Gaussians, particularly for approximating cylindrical structures like the fingers. The Gaussians in 3D have infinite spatial support, which is an advantageous property for pose fitting, as explained later, but also means that the SAG does not represent a finite volume ($\mathcal{C}(\mathbf{x}) > 0$ everywhere). We therefore assume that the hand is well modeled by a 3D SAG if the surface passes through each Gaussian at a distance of 1 standard deviation from the mean.

Hand Model Initialization: The hand model for tracking requires initialization of the skeleton dimensions, Gaussian covariances that control their shapes, and associated colors for an actor before it can be used for tracking. Our method accepts manually created hand models which could be obtained from a laser scan. Alternatively, we also provide a fully automatic procedure to obtain a hand model to fit each person. This method uses a greedy optimization strategy to optimize for a total of 3 global hand shape parameters and 3 independent scaling parameters (along the local principal axes) for each of the 17 Gaussians in

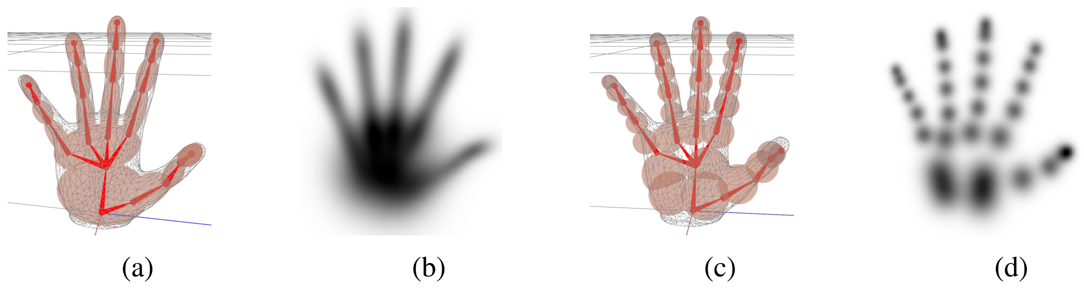


Figure 4.3 (a) Our 3D SAG hand model with 17 **anisotropic Gaussians** visualized as ellipsoids with radii equal to the standard deviation. (c) A 3D SoG hand model with 30 **isotropic Gaussians** visualized as spheres. (b) Visualizes the SAG model density when projected into 2D; with less primitives, the shape of the hand is much better approximated than with the SoG model (d) Visualization of SoG model density.

the hand model. We observed that starting with a manual model and then using the greedy fitting algorithm works best.

2D RGB Image Modeling: We approximate the input RGB images using a 2D SoG, $\mathcal{C}_I$, by quad-tree clustering of regions of similar color. While it would also be possible to approximate the image as 2D SAG, the computational expense of the non-uniform region segmentation would prohibit realtime performance. We found in our experiments that around 500 2D image Gaussians were generated for each camera image.

4.4.2 Projection of 3D SAG to 2D SAG

Pose optimization (Section 4.4.3) requires the comparison of the projections of the 3D SAG into all camera views, with the 2D SoG of each RGB image. Intuitively (for a moment ignoring infinite support), SAG and SoG can be visualized as ellipsoids and spheres, respectively.

The perspective projections of spheres and ellipsoids both yield ellipses in 2D [35]. For the case of isotropic Gaussians in 3D, like in the earlier SoG model, projection of a 3D Gaussian can be approximated as a 2D Gaussian with a standard deviation that is a scaled orthographic projection of the 3D standard deviation (see Chapter 2 and [131, 135]). This simple approximation does not hold for our anisotropic Gaussians.

Therefore, we utilize an exact perspective projection Π of ellipsoids, in order to model the projection of the 3D SAG hand model, $\mathcal{C}_H$, into its 2D SAG equivalent, $\mathcal{C}_P$. Given an ellipsoid in $\mathbb{R}^3$ with associated mean, $\boldsymbol{\mu}_h$ and covariance matrix, $\boldsymbol{\Sigma}_h$, its perspective projection can be visualized as an ellipse in $\mathbb{R}^2$ with parameters $\boldsymbol{\Sigma}_p$ and mean $\boldsymbol{\mu}_p$. Figure 4.4 (a) sketches the perspective projection, intuitively, Π can be thought of as the intersection of the elliptical cone (formed by the ellipsoid and the camera center) with the image plane. Without loss of generality, we assume the camera to be at the origin with a camera matrix,

$\mathbf{P} = \mathbf{K}\,[\,\mathbf{I}\,|\,\mathbf{0}\,]$. The parameters of the projected Gaussian are given by

$$\boldsymbol{\mu}_p = \frac{1}{|\mathbf{M}_{33}|}\mathbf{K}_{33}\begin{bmatrix} |\mathbf{M}_{31}| \\ -|\mathbf{M}_{23}| \end{bmatrix} + \begin{bmatrix} k_{13} \\ k_{23} \end{bmatrix}, \tag{4.3}$$

$$\boldsymbol{\Sigma}_p = -\frac{|\mathbf{M}|}{|\mathbf{M}_{33}|}\mathbf{K}_{33}\,\mathbf{M}_{33}^{-1}\,\mathbf{K}_{33}^{T}, \tag{4.4}$$

where

$$\mathbf{M} = \boldsymbol{\Sigma}_h^{-1}\,\boldsymbol{\mu}_h\boldsymbol{\mu}_h^T\boldsymbol{\Sigma}_h^{-\top} - \left(\boldsymbol{\mu}_h^\top\boldsymbol{\Sigma}_h^{-1}\boldsymbol{\mu}_h - 1\right)\boldsymbol{\Sigma}_h^{-1}, \tag{4.5}$$

$|\mathbf{M}|$ is the determinant of $\mathbf{M}$, $\mathbf{A}_{ij}$ is a matrix $\mathbf{A}$ with its i^{th} row and j^{th} column removed, and k_{ij} is the element at the i^{th} row and j^{th} column of $\mathbf{K}$. This more general projection also leads to a more involved pose fitting energy with more involved derivatives than for the SoG model, as explained in the next section.

4.4.3 Pose Fitting Energy

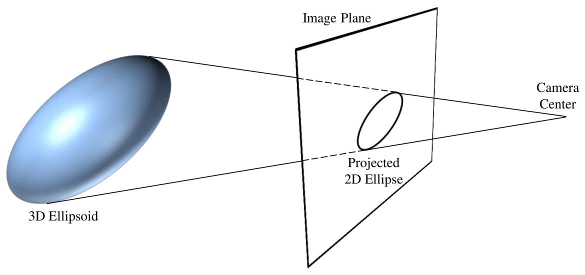


Figure 4.4 Sketch of the perspective projection of ellipsoids as the intersection of the image plane with the cone formed by the camera center and the ellipsoid.

We now define an energy that measures the quality of overlap between the projected 3D SAG $\mathscr{C}_P = \Pi(\mathscr{C}_H)$, and the image SoG $\mathscr{C}_I$, and that is optimized with respect to the pose parameters Θ of the hand model. Our overlap measure is an extension of the SoG overlap measure [135] to a SAG (see also Chapter 2). Intuitively, we assume that two Gaussians in 2D match well if their spatial support aligns, and their color matches. This criterion can be expressed by the spatial integral over their product, weighted by a color similarity term. The similarity of any two sets, $\mathbf{C}_a$ and $\mathbf{C}_b$, of SAG or SoG in 2D (including combined models with both isotropic and anisotropic Gaussians) can thus be defined as

$$\begin{aligned} E(\mathbf{C}_a, \mathbf{C}_b) &= \sum_{p\in\mathbf{C}_a}\sum_{q\in\mathbf{C}_b} d(\mathbf{c}_p, \mathbf{c}_q)\int_\Omega \mathscr{G}_p(\mathbf{x})\mathscr{G}_q(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &= \sum_{p\in\mathbf{C}_a}\sum_{q\in\mathbf{C}_b} d(\mathbf{c}_p, \mathbf{c}_q)\,D_{pq} = \sum_{p\in\mathbf{C}_a}\sum_{q\in\mathbf{C}_b} E_{pq} \end{aligned} \tag{4.6}$$

where E_{pq} is the integral overlap measure mentioned earlier, $d(\mathbf{c}_p, \mathbf{c}_q)$ measures color similarity using the Wendland function [160], and $E_{pq} = d(\mathbf{c}_p, \mathbf{c}_q)\, D_{pq}$. Unlike the SoG model, for the general case of potentially anisotropic Gaussians, the term D_{pq} evaluates to

$$D_{pq} = \frac{\sqrt{(2\pi)^2 |\boldsymbol{\Sigma}_p \, \boldsymbol{\Sigma}_q|}}{\sqrt{|(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q)|}} \mathbf{e}^{-\frac{1}{2}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T (\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q)^{-1} (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)}. \tag{4.7}$$

Using this Gaussian similarity formulation allows us to compute the similarity between the image SoG $\mathcal{C}_I$ and the projected hand SAG $\mathcal{C}_P$.

We also need to consider occlusions of Gaussians from a camera view. Computing a function that indicates occlusion analytically independent of pose parameters is generally difficult and may lead to a discontinuous similarity function. Thus, we use a heuristic approximation of occlusion [135] that yields a continuous fitting energy defined as follows

$$E_{sim}\left[\mathcal{C}_I, \mathcal{C}_H\right] = \sum_{q \in \mathcal{C}_I} \min \left(\sum_{p \in \Pi(\mathcal{C}_H)} w_p^h \, E_{pq}, E_{qq} \right), \tag{4.8}$$

where w_p^h is a weighting factor for each projected 3D Gaussian of the hand model. E_{qq} is the overlap of an image Gaussian with itself. With this formulation, an image Gaussian cannot contribute more to the overlap similarity than by its own footprint in the image. To find the hand pose, E_{sim} is optimized with respect to Θ, as described in the following section. Note that the infinite support of the Gaussians is advantageous as it leads to an *attracting force* between the projected model and the image of the hand, even if they do not overlap in a camera view.

4.4.4 Pose Optimization

The final energy that we maximize to find the hand pose takes the form

$$\mathcal{E}(\Theta) = E_{sim}(\Theta) - w_l \, E_{lim}(\Theta), \tag{4.9}$$

where $E_{lim}(\Theta)$ penalizes motions outside of parameter limits quadratically, and weight w_l is empirically set to 0.1. With the SoG formulation, it was possible to express the energy function (with a scaled orthographic projection) in a closed form analytic expression, and to derive the analytic gradient. We have found that $E_{sim}(\Theta)$ in our SAG-based, even with its full perspective projection model, can still be written in closed form with analytic gradient.

We derive the analytical gradient of E_{sim} with respect to the degrees of freedom Θ in three steps. For each Gaussian pair (h,q) and parameter θ_j we compute

$$\left(\frac{\partial \boldsymbol{\Sigma}_h}{\partial \theta_j}, \frac{\partial \boldsymbol{\mu}_h}{\partial \theta_j}\right) \overset{a)}{\longrightarrow} \left(\frac{d\mathbf{M}}{d\theta_j}\right) \overset{b)}{\longrightarrow} \left(\frac{d\boldsymbol{\Sigma}_p}{d\theta_j}, \frac{d\boldsymbol{\mu}_p}{d\theta_j}\right) \overset{c)}{\longrightarrow} \left(\frac{dD_{pq}}{d\theta_j}\right). \tag{4.10}$$

We exemplify the computation at hand of step a); the input to a) is the change of the ellipsoid covariance matrix $\partial \boldsymbol{\Sigma}_h^{-1}$ and the change of position $\partial \boldsymbol{\mu}_h$ with respect to the DOF θ_j. In this step we are interested in the total derivative

$$\begin{aligned}\frac{d\mathbf{M}}{d\theta_j} = &\sum_{i\in\{1,2,3\}} \frac{\partial}{\partial \boldsymbol{\mu}_{hi}} \frac{\partial \boldsymbol{\mu}_{hi}}{\partial \theta_j} \\ &+ \sum_{k\in\{1,\cdots,6\}} \frac{\partial}{\partial \boldsymbol{\Sigma}_{h_k}^{-1}} \frac{\partial \boldsymbol{\Sigma}_{h_k}^{-1}}{\partial \theta_j}.\end{aligned} \tag{4.11}$$

The partial derivatives of the cone matrix $\mathbf{M}$ with respect to $\boldsymbol{\mu}_h$, $\boldsymbol{\Sigma}_h$ are

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\mu}_{hi}} &= \boldsymbol{\Sigma}_h^{-1}(\mathbf{e}^i \boldsymbol{\mu}_h^\top + \mathbf{e}^i \boldsymbol{\mu}_h^\top)^\top \boldsymbol{\Sigma}_h^{-1} \\ &\quad - \left((\mathbf{e}^{i\top} \boldsymbol{\Sigma}_h^{-1} \boldsymbol{\mu}_h) + (\mathbf{e}^{i\top} \boldsymbol{\Sigma}_h^{-1} \boldsymbol{\mu}_h)^\top\right) \boldsymbol{\Sigma}_h^{-1}, \\ \frac{\partial}{\partial \boldsymbol{\Sigma}_{h_k}^{-1}} &= H + H^\top - \boldsymbol{\mu}_h{}^\top \mathbf{S}^k \boldsymbol{\mu}_h \boldsymbol{\Sigma}_h^{-1} \\ &\quad - (\boldsymbol{\mu}_h^\top \boldsymbol{\Sigma}_h^{-1} \boldsymbol{\mu}_h - 1)\mathbf{S}^k,\end{aligned} \tag{4.12}$$

with e^i the i^{th} unit vector, $\boldsymbol{\mu}_{h_i}$ the i^{th} entry of $\boldsymbol{\mu}_h$, $\mathbf{H} = \mathbf{S}^k \boldsymbol{\mu}_h \boldsymbol{\mu}_h^\top \boldsymbol{\Sigma}_h^{-\top}$, where k indexes the unique elements of the symmetric matrix $\boldsymbol{\Sigma}_h^{-1}$, and $\mathbf{S}^k$ is the symmetric structure matrix with the k^{th} elements equal to one, and zero otherwise. The total similarity energy E_{sim} is the weighted sum over all θ_j and D_{pq} according to Equation 4.8. Combined with the analytical gradient of $E_{lim}(\Theta)$ we obtain an analytic formulation for $\frac{\partial}{\partial\Theta}\mathcal{E}(\Theta)$. As sums of independent terms, both $\mathcal{E}$ and $\frac{\partial}{\partial\Theta}\mathcal{E}$ lend themselves to parallel implementation.

Even though evaluation of fitting energy and gradient is much more involved than for the SoG model, both share the same smoothness properties and can be evaluated efficiently, and thus an optimal pose estimate can be computed effectively using a standard gradient-based optimizer. The optimizer is initiliazed with an extrapolation of the pose parameters from the two previous time steps. The SAG framework leads to much better accuracy and robustness and requires far fewer shape primitives to be compared, as validated in Section 4.5.

4.5 Experiments

We conducted extensive experiments to show that our SAG-based tracker outperforms the SoG based method it extends. We also compare with another state-of-the-art method that uses a single depth camera [140]. We ran our experiments on the publicly available **Dexter 1** dataset (see Chapter 3 and [131]) which has ground truth annotations. This dataset contains challenging, slow and fast motions. We processed all 7 sequences in the dataset and, while [131] evaluated their algorithm only on the slow motions, we evaluated our method on fast motions as well.

For all results we used 10 gradient ascent iterations. Our method runs at a framerate of 25 fps on an Intel Xeon E5-1620 running at 3.60 GHz with 16 GB RAM. Our implementation of the SoG-based tracker of [131] runs slightly faster at 40 fps.

Accuracy: Figure 4.6 shows a plot of the average error for each sequence in our dataset. Over all sequences, SAG had an error of 24.1 mm, SoG had an error of 31.8 mm, and [140] had an error of 42.4 mm (only 3 sequences). The mean standard deviations were 11.2 mm for SAG, 13.9 mm for SoG, and 8.9 mm for [140] (3 sequences only). Our errors are higher than those reported in Chapter 3 because we performed our experiments on both the slow and fast motions as opposed to slow motions only. Additionally, we discarded the palm center used by [131] since this is not clearly defined. We would like to note that [140] perform their tracking on the depth data in Dexter 1, and use no temporal information. In summary, SAG achieves the lowest error and is 7.7 mm better than SoG. This improvement nearly covers the width of a finger thus making it a significant gain in accuracy.

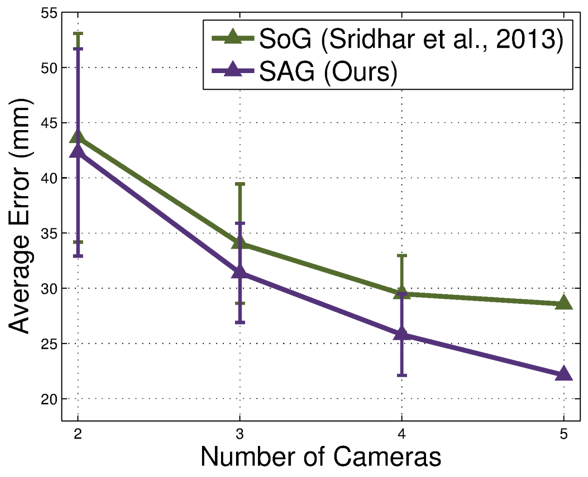


Figure 4.5 This figure shows a comparison of tracking error for SAG and SoG with 2 to 5 cameras. A total of 156 runs were required for SAG and SoG with different camera combinations. The results show that SAG outperforms SoG. Best viewed in color.

Error Frequency: Table 4.1 shows an alternative view of the accuracy and robustness improvement of SAG. We calculated the percentage of frames of each sequence in which the tracking error is less than x mm, where $x \in \{15, 20, 25, 30, 45\}$. This experiment shows clearly that SAG outperforms SoG in almost all sequences and error bounds. In particular, the improvement in **accuracy** is measured by the increased number of frames with error smaller than 15 mm, and the **robustness to fast motions** by the smaller number of dramatic failures

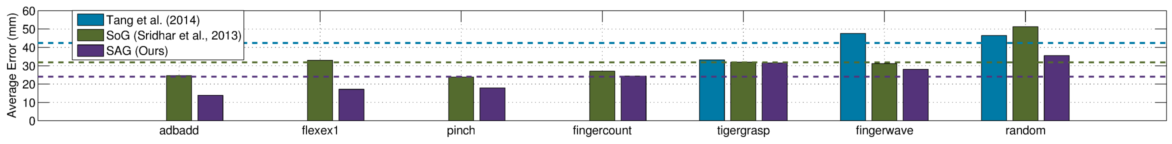


Figure 4.6 Average errors for all sequences in the Dexter 1 dataset. Our method has the lowest average error of 24.1 mm compared to SoG (31.8 mm) and [140] (42.4 mm). The dashed lines represent average errors over all sequences. Best viewed in color.

Error <	**adbadd**		**fingercount**		**fingerwave**		**flexex1**		**pinch**		**random**		**tigergrasp**	
	SoG	SAG	SoG	SAG	SoG	SAG	SoG	SAG	SoG	SAG	SoG	SAG	SoG	SAG
15	34.5	**70.7**	**13.1**	8.7	11.0	**16.7**	5.2	**50.0**	10.8	**34.0**	3.3	**10.5**	**11.2**	10.2
20	48.1	**97.5**	**35.2**	33.4	31.0	**34.3**	12.1	**79.4**	30.78	**66.3**	5.3	**21.4**	**25.6**	**25.6**
25	61.0	**99.4**	54.1	**61.0**	45.8	**47.0**	29.7	**91.0**	50.3	**89.9**	6.9	**34.7**	43.8	**51.7**
30	70.7	**99.4**	65.4	**79.4**	58.5	**59.4**	45.0	**96.5**	81.0	**98.7**	10.9	**46.4**	50.2	**58.7**
45	93.4	**99.7**	90.1	**99.1**	82.0	**90.4**	86.6	**98.9**	**100.0**	**100.0**	40.2	**72.1**	**83.3**	82.3

Table 4.1 Percentage of total frames in a sequence that have an error of less x mm. We observe that `SAG` outperforms `SoG` in all sequences and error bounds. The values in bold face indicate the best values for a given error bound.

of errors larger than 30 mm. For example, in the `adbadd` sequence 70.7% of frames are better than 15 mm for `SAG` while only 34.5% of frames for `SoG`. Note that when $x = 100$ mm, the percentage of frames $< x$ mm is 100% for `SAG`.

Influence of Number of Cameras: To evaluate the scalability of our method to the number of cameras we conducted an experiment where each camera was progressively disabled with total active cameras ranging from 2 to 5. This leads to 26 possible camera combinations for each sequence and a total of 156 runs for both the `SAG` and `SoG` methods. We excluded the `random` sequence as it was too challenging for tracking with 3 or less cameras.

Figure 4.5 shows the average error over all runs for varying cameras. Clearly, `SAG` produces lower errors and standard deviations for all camera combinations. We also observe a diverging trend and hypothesize that as the number of cameras is increased the gap between `SAG` and `SoG` will also increase. This may be important for applications requiring very precise tracking such as motion capture for movies. We associate the improvements in accuracy of `SAG` with its ability to approximate the users' hand better than `SoG`. Figure 4.3 (b, d) visualizes the projected model density and reveals a better approximation for `SAG`.

Qualitative Tracking Results: Finally, we show several qualitative results of tracking in Figure 4.7 comparing `SAG` and `SoG`. Since our tracking approach is flexible we are also able to track additional simple objects such as a plate using only a few primitives.

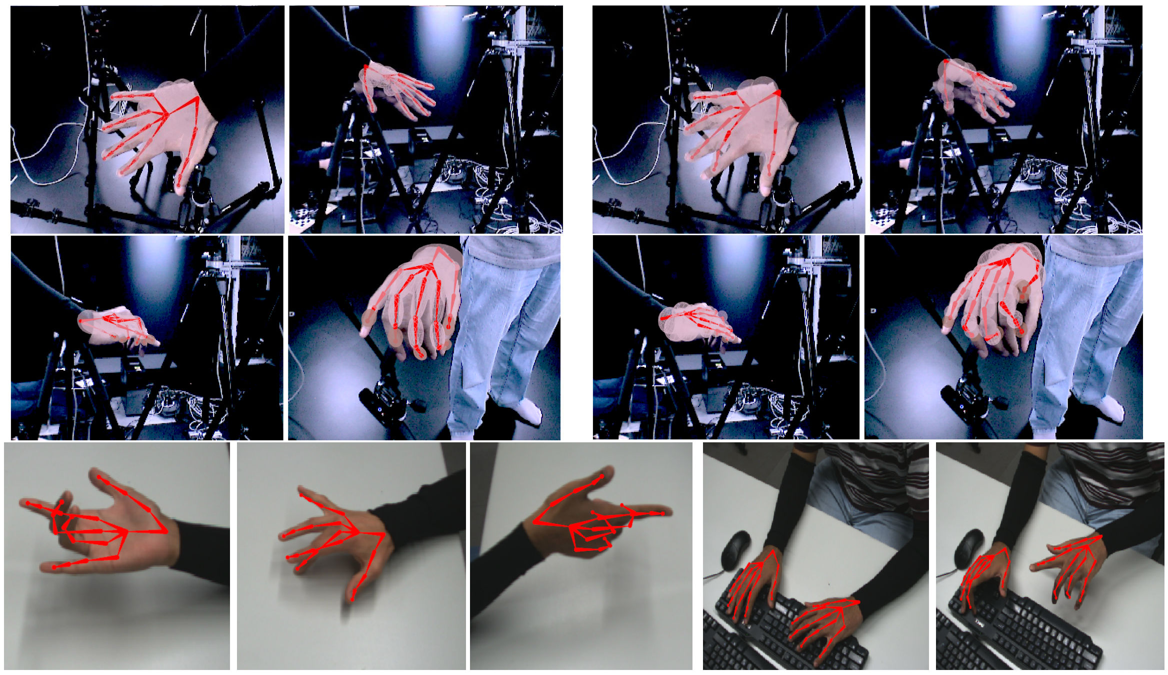


Figure 4.7 **First Two Rows**: Comparison of SAG (left) and SoG (right) for two frames in the Dexter 1 dataset. In the first row, SAG covers the hand much better during a fast motion of the hand in spite of using fewer primitives. In the second row, a challenging motion is performed for which SAG performs better. **Bottom Row**: Realtime tracking results for one hand with different actors, and two hands.

4.6 Discussion

We presented a method for articulated hand tracking that uses a novel Sum of Anisotropic Gaussians (SAG) representation to track hand motion. Our SAG formulation uses a full perspective projection model and uses only a few Gaussians to model the hand. Because of our smooth and differentiable pose fitting energy, we are able to perform fast gradient-based pose optimization to achieve real-time frame rates. Our approach produces more robust and accurate tracking than previous methods while featuring advantageous numerical properties and comparable runtime. As demonstrated in the above experiments, our method advances state of the art methods in accuracy and is suitable for real-time applications.

4.7 Conclusion

The method presented in this chapter is a purely generative approach that could lose tracking because of fast hand motions. Like other hybrid methods, we could augment our method with a discriminative tracking strategy similar to the approach presented in Chapter 3. The generality of our method allows easy integration into such a hybrid framework.

Our method uses multiple calibrated cameras which could hinder adoption by users for interactive applications. These limitations could be overcome if we rely only on the depth data from a single camera. In the next chapter, we present a method that uses purely depth data from a single camera for real-time hand tracking.

While we track the hand at more than 25 fps, it might be insufficient for applications in gesture-based input. In the next chapter, we also show how we can achieve tracking speeds of 50 fps which make it more useful for interactive applications.

Chapter 5

Real-time Hand Tracking from a Single Depth Camera

In this chapter, we show how we can add an important constraint to the problem—the use of only a single camera for hand tracking. Because of the ambiguities inherent in single RGB images, we use a depth camera which provides a depth value at every pixel. Our algorithm is one of the first methods that uses only the depth channel to track complex hand poses at 50 FPS while running completely on the CPU. This enables new HCI applications that require real-time user interaction and low latency. Parts of this chapter appeared in a previous publication [129].

5.1 Introduction

There is increasing interest in using markerless hand tracking in human-computer interaction, for instance when interacting with 3D applications, augmented reality, smart watches, and for gestural input [65, 75, 152]. However, flexible, realtime markerless tracking of hands presents several unique challenges. First, natural hand movement involves simultaneous control of several (≥ 25) degrees-of-freedom (DOFs), fast motions with rapid changes in direction, and self-occlusions. Tracking fast and complex *finger articulations* combined with global motion of the hand at *high framerates* is critical but remains a challenging problem. Second, many methods use dense camera setups [96, 131] or GPU acceleration [99], i.e., have high *setup costs* which limits deployment. Finally, applications of hand tracking demand tracking across many camera-to-scene configurations including desktop, egocentric and wearable settings. This chapter presents a novel method for hand tracking with a single depth camera that aims to address these challenges. Our method is extremely fast (nearly equalling the capture rate of the camera), reliable, and supports varying close-range camera-to-hand arrangements including desktop, and moving egocentric (camera mounted to the head).

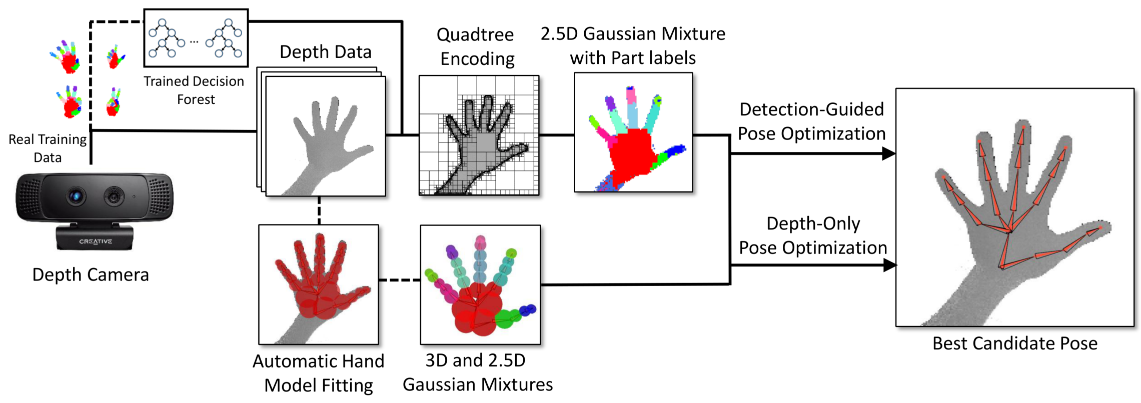

Trained Decision Forest
Real Training Data
Depth Camera
Depth Data
Quadtree Encoding
2.5D Gaussian Mixture with Part labels
Detection-Guided Pose Optimization
Depth-Only Pose Optimization
Best Candidate Pose
Automatic Hand Model Fitting
3D and 2.5D Gaussian Mixtures

Figure 5.1 Overview of our detection-guided tracking method. We develop a novel representation for depth data and hand model as a mixture of 2.5D Gaussians. This representation allows us to combine the benefits of model-based generative tracking and discriminative part detection. Pixels classified using a trained decision forest are directly incorporated as evidence in detection-guided pose optimization. Dashed lines indicate offline computation. Best viewed in color.

The main novelty in our work is a new **detection-guided optimization** strategy that combines the benefits of two common strands in hand tracking research—model-based generative tracking and discriminative hand pose detection—into a unified framework that yields high efficiency and robust performance and minimizes their mutual failures (see Figure 5.1). The first contribution in this strategy is a novel, efficient representation of both the input depth and the hand model shape as a mixture of Gaussian functions. While previous work used primitive shapes like cylinders [99, 96] or spheres [108] to represent the hand model, we use Gaussian mixtures for both the depth data and the model (see Chapter 2). This compact, mathematically smooth representation allows us to formulate pose estimation as a 2.5D generative optimization problem in depth. We define a new **depth-only** energy, that optimizes for the similarity of the input depth with the hand model. It uses additional prior and data terms to avoid finger collisions and to preserve the smoothness of reconstructed motions. Importantly, since the energy is smooth, we can obtain analytic gradients and perform rapid optimization. While pose tracking on this energy alone could run in excess of 120 fps using gradient-based local optimization, this often results in a wrong local pose optimum.

The second contribution in our strategy is thus to incorporate evidence from trained randomized decision forests that label depth pixels into predefined parts of the hand. Unlike previous purely detection-based approaches [38, 121], we use the part labels as additional constraints in an augmented version of the aforementioned depth-only energy, henceforth termed **detection-guided** energy. The part labels include discriminative detection evidence into generative pose estimation. This enables the tracker to better recover from erroneous local pose optima and prevents temporal jitter common to detection-only approaches. The precondition for recovery is reliability of the part labels. However, even with large training sets it is hard to obtain perfect part classification (per-pixel accuracy is usually around 60%). Thus, pose estimation based on this additional discriminative evidence is also not sufficient.

Our third contribution therefore, is a new **late fusion approach** that combines particle-based multi-hypothesis optimization with an efficient local gradient-based optimizer. Previous work has used particle-based optimizers, but they tend to be computationally expensive [99, 96]. Our approach is fast because we combine the speed of local gradient-based optimization with the robustness of particle-based approaches. At each time step of depth video, a set of initial pose hypotheses (particles) is generated, from which a subsequent local optimization is started. Some of these local optimizers use the depth-only pose energy, some others use the detection-guided energy. In a final late fusion step the best pose is chosen based on the pose fitting energy.

Our approach results in a temporally stable and efficient tracker that estimates full articulated joint angles of even rapid and complex hand motions at previously unseen frame rates in excess of 50 fps, even with a CPU implementation. Our tracker is resilient to erroneous local convergence by resorting to the detection-guided solution when labels can be trusted, and it is not misled by erroneous detections as it can then switch to the depth-only tracking result.

We show these improvements with (1) qualitative experiments, (2) extensive evaluation on public datasets, and (3) comparisons with other state-of-the-art methods.

5.2 Related Work

In this review, we focus on previous approaches to markerless hand tracking from depth images. First, we briefly discuss marker-based and multi-camera techniques. Gloves fitted with retro-reflective markers or color patches were used to estimate the kinematic skeleton using inverse kinematics [137, 153, 172]. Research on *markerless* tracking was made popular in the early 2000s (e.g., [5, 163]). Some recent solutions assume a multi-camera setup with offline processing [13, 96, 154], while others track at interactive rates [131, 152] of up to 30 fps [132]. However, calibrated multi-camera setups make these methods difficult to adopt for practical applications. The recent introduction of consumer depth sensors has resulted in a number of methods that require only a single depth camera. Some commercial solutions exist, such as the Leap Motion[1]. Although Leap Motion is fast, the approach uses strong priors and fails with complex self-occlusions and non-standard motions (we show an example in Section 5.7).

The main approaches to real-time hand tracking can be divided into two classes: (1) generative and (2) discriminative methods.[2] First, a method to track a hand manipulating an object that takes 6.2 s/frame was proposed in [47]. Oikonomidis et al. [99] proposed a model-based method that made use of particle-swarm optimization. This method requires GPU acceleration to achieve 15 fps and uses skin color segmentation which is sensitive to lighting. They showed an extension to interacting hands, although only offline [97, 98]. Melax et al. [86] proposed a tracking method directly in depth by efficient parallel physics simulations. While this method is fast, finger articulations are often incorrectly tracked, as we demonstrate later. Recent real-time surface tracking methods from depth [173] were applied to hands, but are limited to simple motions with no occlusions.

Second, decision forests were used with great success for full body tracking [42, 121] and later adopted to hand tracking with varying success. Keskin et al. [62] proposed a method for recognizing finger spelling in depth data using classification forests. Many others [38, 140, 142, 165] also proposed methods based on variants of random forests. Tompson et al. [144] track hand motion from depth at $\leq$ 25 fps using feature detections from a convolutional network and further pose refinement through inverse kinematics. However, a common problem with these approaches is jitter due to missing temporal information at each time step. We provide a direct comparison with one recent method [140] to demonstrate this. Moreover, most methods estimate joint positions with temporally varying bone lengths, limiting applicability.

In Chapter 3, we proposed combining discriminative and generative hand pose estimation. This approach detected only fingertips, which could easily be occluded or misdetected. Offline tracking in RGB-D using a combination of discriminative and generative pose estimation was shown in [147]. Qian et al. [108] proposed a method based on optimization in combination with discriminative fingertip detection, achieving 25 fps. However, tracking would be hard with this method when one or more of the fingertips are occluded.

In this chapter we present a method that combines decision forests and pose estimation in a unified optimization framework. To our knowledge, ours is the first method to track rapid articulations at 50 fps using a single depth camera and yet achieve state-of-the-art accuracy.

[1]https://www.leapmotion.com/

[2]There are algorithmic parallels to full-body tracking [7, 41, 70, 121].

5.3 Input and Model Representation

In the past, representations such as spheres or cylinders have been used to represent the hand model [99, 108]. Similarly, downsampled images [152, 153] or silhouettes [13] have been used as representations of input data. However, such representations make pose optimization energies discontinuous and difficult to optimize. Our novel representation of depth and 3D model data uses a mixture of *weighted* Gaussian functions to represent both depth data and the hand shape. We were inspired by [135] who use multiple 2D RGB images and [70] who use depth data. Both methods rely on a uniformly weighted Gaussian mixture, and a 2D or 3D error metric for pose estimation. However, we make important modifications that allows representing 3D depth data using a 2.5D formulation since data from depth sensors contains information only about the *camera-facing* parts of the scene. Thus, we enable pose estimation based on alignment to a single depth image using a 2.5D error metric.

An instance of the input depth or the hand model can be represented as a mixture of Gaussian functions

$$\mathbf{C}(\mathbf{x}) = \sum_{i=1}^{n} w_i \, \mathbf{G}_i(\mathbf{x}; \sigma, \boldsymbol{\mu}), \tag{5.1}$$

where $\mathbf{G}_i(.)$ denotes a unnormalized Gaussian function with isotropic variance, σ^2, in all dimensions of $\mathbf{x} \in \mathcal{R}^n$, and mean $\boldsymbol{\mu}$. The Gaussian mixture representation has many advantages. First, it enables a mathematically smooth pose estimation energy which is analytically differentiable. Second, only a few Gaussians are needed for representing the input depth and the hand model, an implicit data reduction which makes optimization extremely fast. Finally, it provides a natural way to compute collisions using an analytically differentiable energy. We show later that collisions are important for pose estimation (Section 5.4). To aid visualization, we henceforth represent each Gaussian in the mixture as a sphere ($\mathbf{x} \in R^3$) or circle ($\mathbf{x} \in R^2$) with a radius of $1\,\sigma$. However, Gaussians have infinite support ($\mathbf{C}(\mathbf{x}) > 0$ everywhere) and can produce long range attractive or repulsive *force* during pose optimization.

5.3.1 Depth Data Representation

The depth camera outputs depth maps, i.e., each pixel has an associated depth value. Depth maps contain only the camera-facing parts of the scene and information about occluded parts is unavailable. We therefore only represent the camera-facing pixels using Gaussian mixtures, which are computed in real-time (see also Figure 5.2).

First, we decompose the depth image into regions of homogeneous depth using a quadtree. The quadtree recursively decomposes depth image regions further, until the depth difference between the furthest and nearest point in a region is below a threshold ϵ_c ($\epsilon_c = 20$ mm in all experiments). To each quad in the tree, we fit a Gaussian function with $\boldsymbol{\mu}$ set to the center of the quad, and $\sigma = a/\sqrt{2}$, where a is the side length of the quad. We also set each Gaussian function to have unit weight w_i since we consider all input data to be equally important. This leads us to an analytic representation of the *camera-facing surface* of the input depth, $\mathbf{C}_I(\mathbf{x}) = \sum_{q=1}^{n} \mathbf{G}_q(\mathbf{x})$, where $\mathbf{x} \in \mathcal{R}^2$ and n is the number of leaves in the quadtree. Additionally, each quad has an associated depth value, d_q, which is the mean of all depth pixels within the quad. Figure 5.1 illustrates the process of converting input depth to a Gaussian mixture.

5.3.2 Hand Model

We model the volumetric extent of the hand analytically using a mixture of 3D Gaussian functions, $\mathbf{C}_h(\mathbf{x}) = \sum_{h=1}^{m} w_h \mathbf{G}_h(\mathbf{x})$ where $\mathbf{x} \in \mathcal{R}^3$ and m is the number of Gaussians. We assume that the best fitting model has Gaussians whose isosurface at $1\,\sigma$ coincides with the surface of the hand. In Section 5.6 we present a fully automatic procedure to fit such a hand model to a user. Additionally, $\mathbf{C}_h$, is attached to a *parametric*, kinematic skeleton similar to that of [123], i.e., each 3D Gaussian is attached to a bone which determines its mean position in 3D. We use $|\Theta| = 26$ skeletal pose parameters in twist representation, including 3 translational DOFs, 3 global rotations, and 20 joint angles.

Model Surface Representation: $\mathbf{C}_I$ is a representation of the *camera-facing surface* while $\mathbf{C}_h$ represents the full volumetric extent of the hand. In order to create an equivalent representation of the hand model that approximates the camera-facing parts, which we later use in pose optimization (Section 5.4). For each model Gaussian in $\mathbf{C}_h$, we create a new projected Gaussian such that the projected hand model has the form $\mathbf{C}_p = \sum_{p=1}^{m} w_p \mathbf{G}_p(\mathbf{x})$ where $\mathbf{x} \in R^2$ and $w_p = w_h \,\forall\, h$. $\mathbf{C}_p$ is a representation of the hand model as seen from the perspective of the depth camera and is defined over the depth image domain. The parameters of each Gaussian $\mathbf{G}_p$ are set to be $(\boldsymbol{\mu}_p, \sigma_p)$, where $\boldsymbol{\mu}_p = \mathbf{K}\,[\,\mathbf{I}\,|\,\mathbf{0}\,]\,\boldsymbol{\mu}_h$. Like [135] we approximate the perspective projection with a scaled orthographic projection, yielding 2D Gaussians with $\sigma_p = \sigma_h\, f / \left[\boldsymbol{\mu}_p\right]_z$. Here f is the focal length of the camera, and $\left[\boldsymbol{\mu}_p\right]_z$ denotes the z-coordinate of the Gaussian mean.

5.4 Hand Pose Optimization

In this section we describe our new formulation of pose estimation as an optimization problem using the Gaussian mixture representation of 2.5D depth data (See Figure 5.1). Our algorithm uses two variants of a model-to-image similarity energy, one that is only based on depth data (Section 5.4.1), and another that is guided by decision forest-based part detection (Section 5.4.3). Pose estimates obtained with each energy are used by a late fusion approach to find the final pose estimate (Section 5.5). Input to pose optimization at each time step of depth video is the 2.5D mixture of Gaussians representation of a depth image $\mathbf{C}_I$. The latter is computed after median filtering the depth (to remove flying pixels in time-of-flight data), and for a constrained working volume in depth between 150 mm and 600 mm from the camera. The 3D Gaussian mixture of the hand model is denoted by $\mathbf{C}_h$ and its projected version is denoted by $\mathbf{C}_p$.

5.4.1 Depth-Only Pose Optimization

Our goal is to optimize for the skeleton pose parameters Θ that best explain the input data and are anatomically plausible. We frame an energy that satisfies our goal while being mathematically smooth and differentiable. These properties make the energy ideal for fast optimization.

5.4.2 Objective Function

Our new energy has the following general form:

$$\mathcal{E}(\Theta) = E_{sim} - w_c \, E_{col} - w_l \, E_{lim} - w_s \, E_{smo}, \tag{5.2}$$

where E_{sim} is a measure of 2.5D similarity between $\mathbf{C}_I$ and $\mathbf{C}_p$, E_{col} is a penalty for collisions between Gaussians in $\mathbf{C}_h$, E_{lim} enforces a soft constraint on the skeleton joint limits, E_{smo} enforces smoothness in the tracked motion. In all our experiments, we used fixed weighting factors chosen by searching for the best accuracy over the dataset: $w_c = 1.0$, $w_l = 0.2$, and $w_s = 1.0$. Before describing each of the terms in detail we first introduce a measure of similarity between two Gaussian mixtures which is the basis for many of the terms in the objective.

Gaussian Similarity Measure: We define a similarity measure between any two pairs of Gaussian mixtures $\mathbf{C}_a$ and $\mathbf{C}_b$ as,

$$E(\mathbf{C}_a, \mathbf{C}_b) = \sum_{p \in \mathbf{C}_a} \sum_{q \in \mathbf{C}_b} D_{pq}, \tag{5.3}$$

$$\text{where, } D_{pq} = w_p \, w_q \int_{\Omega} \mathbf{G}_p(\mathbf{x}) \, \mathbf{G}_q(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{5.4}$$

Ω denotes the domain of integration of $\mathbf{x}$. This Gaussian similarity measure has a high value if the spatial support of the two Gaussian mixtures aligns well. It bears resemblance to the Bhattacharyya Coefficient [16] used to measure the similarity of probability distributions while being computationally less expensive.

Depth Similarity Term (E_{sim}):

The 2.5D depth similarity term measures the quality of overlap between the projected model Gaussian mixture $\mathbf{C}_p$ and the image Gaussian mixture $\mathbf{C}_I$. Additionally, this measure also incorporates the depth information available for each Gaussian in the mixture. Figure 5.2 explains this term intuitively. Two Gaussians that are close (in 2D pixel distance) in the depth image obtain a high value if their depth values are also close. On the other hand, the same Gaussians obtain a low value if their depths are too far apart. Formally, this term is defined as,

$$E_{sim}(\mathbf{C}_p, \mathbf{C}_I) = \frac{1}{E(\mathbf{C}_I, \mathbf{C}_I)} \sum_{p \in \mathbf{C}_p} \sum_{q \in \mathbf{C}_I} \Delta(p, q)\, D_{pq} \tag{5.5}$$

where D_{pq} is as defined in Equation 5.4 and the *depth similarity factor* is

$$\Delta(p, q) = \begin{cases} 0, & \text{if } |d_p - d_q| \geq 2\sigma_h \\ 1 - \frac{|d_p - d_q|}{2\sigma_h}, & \text{if } |d_p - d_q| < 2\sigma_h \end{cases}. \tag{5.6}$$

Here, d_p and d_q are the depth values associated with each Gaussian in $\mathbf{C}_p$ and C_q respectively, and σ_h is the standard deviation of the *backprojected* model Gaussian $\mathbf{G}_h$. The surface depth value of each Gaussian in C_p is computed as $d_p = [\boldsymbol{\mu}_h]_z - \sigma_h$. The factor $E(\mathbf{C}_I, \mathbf{C}_I)$ is the similarity measure from Equation 5.3 of the depth image with itself and serves to normalize the similarity term. The Δ factor has a support in the interval $[0, 1]$ thus ensuring the similarity between a projected model Gaussian and an image Gaussian is 0 if they lie too far apart in depth.

Collision Penalty Term (E_{col}): The fingers of a hand are capable of fast motions and often come in close proximity with one another causing aliasing of corresponding depth pixels in the input. Including a penalty for collisions avoids fingers *sticking* with one another and Gaussian interpenetration. The 3D Gaussian mixture representation of the hand model ($\mathbf{C}_h$) offers an efficient way to penalize collisions because they implicitly act as collision

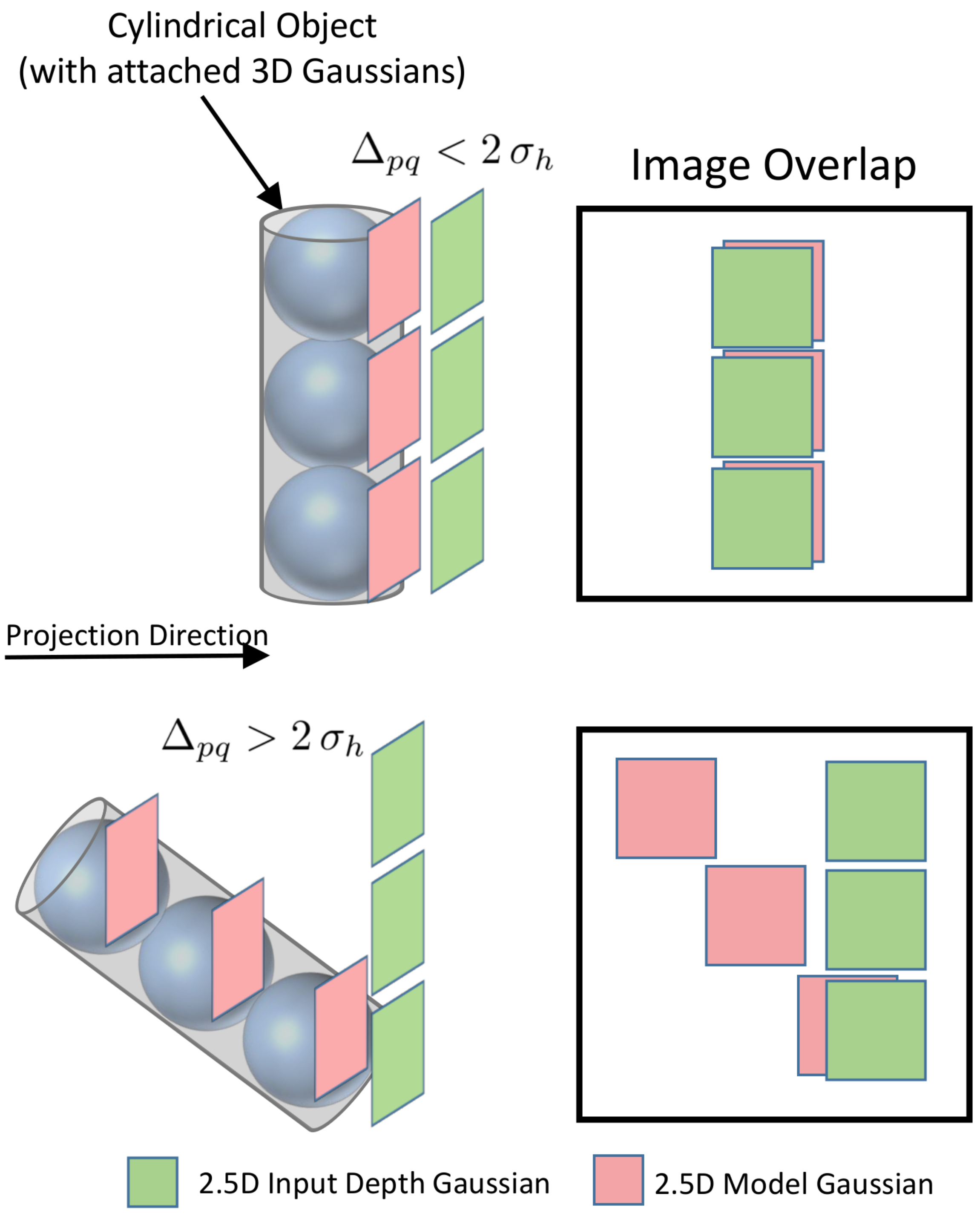

Cylindrical Object
(with attached 3D Gaussians)
$\Delta_{pq} < 2\,\sigma_h$
Image Overlap
Projection Direction
$\Delta_{pq} > 2\,\sigma_h$
2.5D Input Depth Gaussian
2.5D Model Gaussian

Figure 5.2 **Depth Similarity Term**: Consider the similarity value (E_{sim}) for a cylindrical shape represented by 3 Gaussians ($x \in \mathcal{R}^3$). The top figure shows a case where the value of E_{sim} is high since the image overlap is high and the depth difference Δ_{pq} is low. The bottom figure shows a case where the image overlap is moderate but $\Delta > 2\,\sigma_h$ thus making $E_{sim} = 0$.

proxies. We define the penalty for collisions as,

$$E_{col}(\Theta) = \frac{1}{E(\mathbf{C}_h, \mathbf{C}_h)} \sum_{p \in \mathbf{C}_h} \sum_{\substack{q \in \mathbf{C}_h \\ q > p}} D_{pq}, \tag{5.7}$$

where $E(\mathbf{C}_h, \mathbf{C}_h)$ is the similarity measure from Equation 5.3 for the hand model and serves to normalize the collision term. The collision term penalizes model Gaussians that collide with others but not if they collide with themselves. As we show in the results, the collision term has a large impact on tracking performance.

Joint Limit Penalty Term (E_{lim}): We add a penalty for poses that exceed predefined joint angle limits. This forces biomechanically plausible poses to be preferred over other poses. The joint limit penalty is given as,

$$E_{lim}(\Theta) = \sum_{\theta_j \in \Theta} \begin{cases} 0, & \text{if } \theta_j^l \leq \theta_j \leq \theta_j^h \\ ||\theta_j^l - \theta_j||^2, & \text{if } \theta_j < \theta_j^l \\ ||\theta_j - \theta_j^h||^2, & \text{if } \theta_j > \theta_j^h \end{cases} \tag{5.8}$$

where θ_j^l and θ_j^h are the lower and higher limits of the parameter θ_j which is defined based on anatomical studies of the hand [123]. The result is a tracked skeleton that looks biomechanically plausible.

Smoothness Penalty Term (E_{smo}): During frame-by-frame pose optimization, noise is introduced which manifests as jitter in tracking. To prevent this we penalize fast motions by adding a penalty as done by [135]. This term is given as,

$$E_{smo}(\Theta) = \sum_{j=0}^{|\Theta|-1} \left(0.5 \left(\Theta_j^{t-2} + \Theta_j^t\right) - \Theta_j^{t-1}\right)^2 \tag{5.9}$$

where, Θ^t denotes the pose at time t. This term acts as a regularizer and prevents jitter in the tracked pose.

5.4.3 Detection-Guided Pose Optimization

To increase chances of recovery when the estimated pose is at a wrong local pose optima, we use a second pose optimization energy that includes evidence from hand part detection. In particular, we use pixel labels computed with a trained random forest [30]. Decision forests have been used before for 3D pose and joint position detection [62, 140, 142, 165]. We are interested in part labels and therefore follow an approach similar to [121] and [62]. The evidence from the part labels is incorporated in our tracking.

We use 12 part labels for the hand (see Figure 5.1) and found this to be an ideal trade-off between classification accuracy and sufficient evidence for detection-guided optimization. We adopt the same depth features as [121]. We use 50,000 labeled training images spanning the hand pose space. As opposed to previous work [62] that use synthetic data, we use **real** hand motions with part labels which were obtained using the depth-only version of our method and tracking motions slowly without causing tracking failure. During training we trained 3 trees, each with a maximum depth of 22. For each training image, we sampled 2000 random, foreground pixels, and evaluated 4000 candidate threshold-feature response pairs.

During quadtree clustering of the depth (Section 5.3.1) each quad is endowed with a part label, l_q which is the label with the highest number of votes among all pixels in the quad. We can now tightly integrate the part labels in the optimization by defining a pose fitting energy identical to Equation 5.2 with one exception: the depth similarity factor from Equation 5.6 is replaced by the following *label similarity factor*.

$$\Delta_l(p,q) = \begin{cases} 0, & \text{if } l_p \neq l_q \text{ or } |d_p - d_q| \geq 2\,R_i \\ 1 - \frac{|d_p - d_q|}{2\,R_i}, & \text{if } l_p = l_q \end{cases},$$

where l_p and l_q are the part labels, d_p and d_q are the depth values, R_i refers to the radius of influence which is set to 200 mm in all our experiments. Intuitively, Δ_l has a value of zero if the labels are different and a value of one if the two Gaussians have identical labels and are perfectly aligned in 2.5D. The labels l_p are obtained from preassigned labels of each Gaussian in the hand model.

5.5 Late Fusion

The goal of optimization is to find the pose Θ such that $\mathcal{E}(\Theta)$ is maximized. Our energies—both with and without detection—are well suited for gradient based optimization because we can derive the analytic gradient with respect to the DOFs Θ. For efficiency, we adopt the fast gradient-based optimizer with adaptive step length proposed by [135].

To improve robustness, especially with changes in direction and global rotation, we use multiple *pose particles* for optimizing each frame. Multiple particles improve the chances of a good initialization for optimization. Each particle P_i is initialized using the pose parameters from two previous time steps Θ^{t-1} and Θ^{t-2} with different extrapolation factors

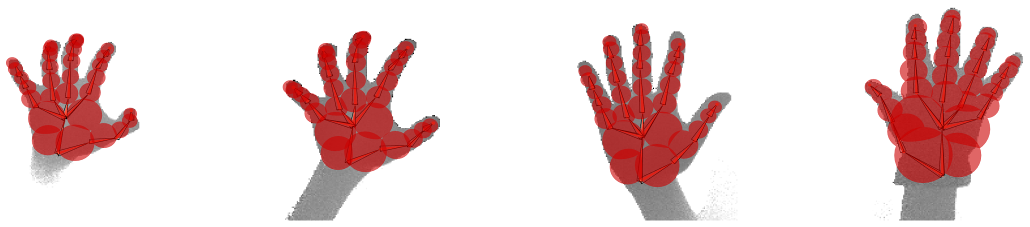


Figure 5.3 Automatic fitting of user specific hand model for 4 subjects, one of whom is wearing a thick glove to simulate variability in hand dimension. The red spheres denote 3D Gaussians.

α_{ij} for each DOF j. This is given as $P_i = \theta_j^{t-1} + \alpha_{ij}\theta_j^{t-2}, \forall j$. We sample α_{ij} from a normal distribution with mean fixed at the initial value of θ_j. All but one of these particles is optimized using the depth-only energy. Finally, the pose particle which converges with the best energy value is chosen as the winning pose. In all our experiments, we found that 2–3 particles were sufficient to obtain more robust results. Increasing the number of particles had a negative effect and caused jitter in the final pose. Each particle used 10–30 iterations per frame. We justify the choice of these parameters in Section 5.7.

5.6 User Specific Hand Modeling

Accounting for the fact that there are large variations in anthropometric dimensions, our pose optimization method works best with a customized hand model for each user. Our method does not necessitate laser scans, manual tuning of the model, or semi-automatic bone model optimization as used by existing methods [135, 131].

We observed in our experiments that the primary variations in hand dimensions are finger thickness, hand length and width. We developed a simple strategy where a default hand model is scaled using three parameters: hand length, width, and variance of Gaussians. To find the scaling parameters for a user, we perform a greedy search over a fixed range for each scaling parameter. At each point on this parameter grid we evaluate the energy function value from Equation 5.2. The parameters that obtain the best energy are selected as the model scaling parameters. This method is fast and takes less than a second to find a user-specific hand model. Figure 5.3 shows some qualitative results from our model fitting strategy for different users.

5.7 Results and Evaluation

We provide quantitative and qualitative evidence for performance with fast motions and finger articulations. Evaluation of hand tracking algorithms is challenging because ground truth data is difficult to obtain. Marker-based motion capture is often problematic due to self-occlusions. Many methods have therefore resorted to evaluation on synthetic data [99, 96] which, however, is not representative of real hand motions. There are also no established benchmark datasets with accepted error metrics, and only a few implementations have been made public.

We use the dataset from [131] which consists of seven challenging sequences (abduction–adduction, finger counting, finger waving, flexion–extension, pinching, random motions, grasping) that are further split into slow and fast parts. The fingertips are annotated manually in the depth data thus making it possible to compare with the multi-view approaches of [131] and [132]. Additionally, we also compare with the discriminative method of [140] on 3 sequences. We also motivate the need for our fusion strategy, parameter selection in optimization, and analyze the effects of different components of our objective function. We also provide details about our framerate and qualitative evidence of improvements over [86] and the Leap Motion.

Error Metrics: Our evaluations concern the average fingertip localization error which correlates well with overall pose accuracy. For each sequence, we compute Euclidean error of the 5 fingertip positions averaged over all frames. Additionally, we use a second error metric [108] which is the percentage of frames that have an error of less than x mm where $x \in \{15, 20, 25, 30\}$. This is a stricter measure that highlights reliability.

5.7.1 Quantitative Evaluation

Accuracy: Figure 5.4 shows our average error compared with that of [131], [132], and [140]. Our method produces the lowest average error of **19.6** mm while using only a single depth camera. The multi-view approaches of [132] and [131] have errors of **24.1** mm and **31.8** mm respectively. The detection-based discriminative method of [140] has an error of **42.4** mm (3 sequences only) highlighting the need for using temporal information. We observe that our method does particularly well for motions that involve articulation of fingers such as `flexex1`. Our worst performance was on the `random` sequence involving fast global hand rotation.

Error Frequency: Table 1 confirms the trend that our method performs well for finger articulations. In 6 out of 7 sequences, our method results in tracking errors of less than

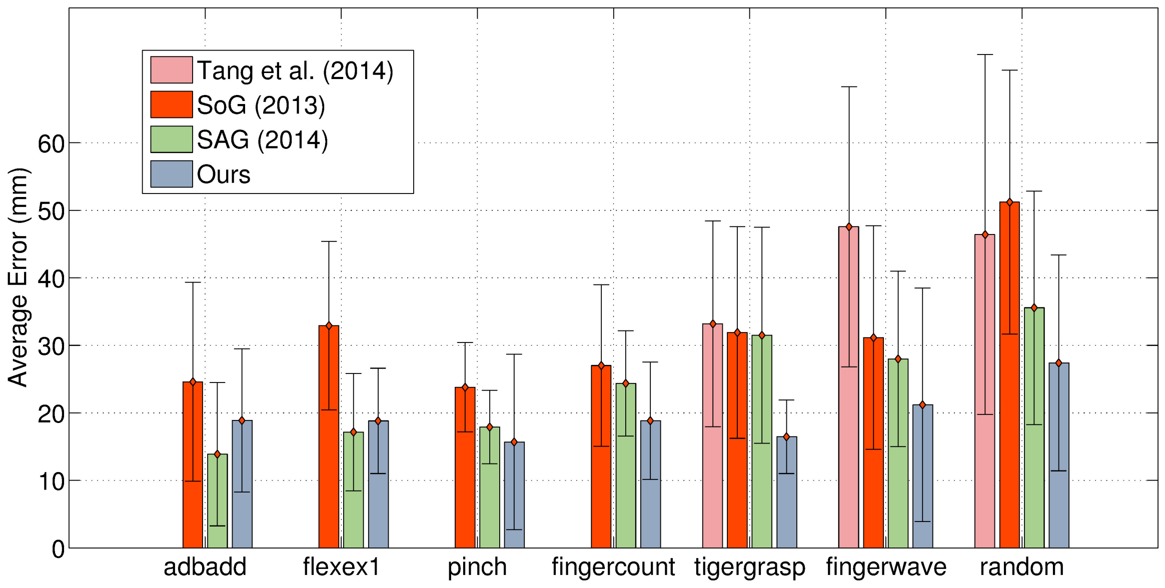


Figure 5.4 Average error over the 7 sequences in Dexter 1 and comparison with the **multi-view** methods of [131] and [132], and the **detection-based** method of [140]. Our method achieves the lowest error on 5 of the 7 sequences and the best average error (**19.6** mm).

Error < (mm)	adbadd	fingercount	fingerwave	flexex1	pinch	random	tigergrasp
15	56.6	50.0	56.2	53.7	56.7	19.1	62.9
20	70.6	66.5	71.2	68.1	83.9	40.7	80.6
25	76.2	77.7	78.3	76.7	93.1	59.0	87.3
30	84.9	85.8	85.0	85.5	97.4	70.6	91.8

Table 5.1 Percentage of total frames in a sequence that have an error of less x mm.

30 mm in 85% of the frames. A closer examination shows that these sequences contain complex finger articulations.

Robustness: We measure robustness as the ability of a tracker to recover from tracking failures. To demonstrate how our late fusion strategy and the different terms in the energy help achieve this, we show the frame-wise error over the `flexex1` sequence (Figure 5.6). Using the depth-only energy with all terms except E_{sim} disabled (2 particles) results in catastrophic tracking failure as shown by the accumulating error. Adding the other terms, especially the collision penalty (E_{col}) term, improves accuracy but results are still unsatisfactory. The results from the late fusion approach show large gains in accuracy. The errors also remain more uniform which results in temporally stable tracking with less jitter.

Number of Particles and Iterations: Figure 5.5 shows the effect of varying the number of particles and iterations during optimization. As the number of particles increased we noticed very little increase in accuracy. In fact, the best accuracy was with 2 particles which we use throughout. We noticed a reduction in error when using more number of iterations

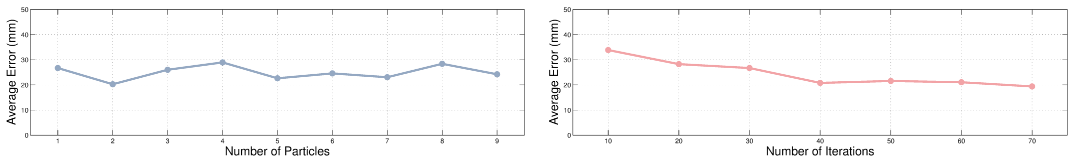


Figure 5.5 Effect of varying the number of particles and iterations during optimization. We found that increasing the number of particles resulted in diminishing returns.

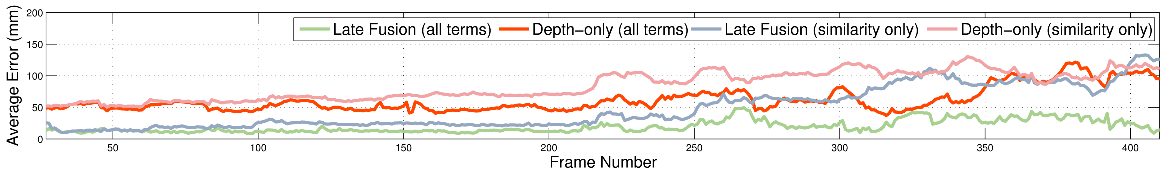


Figure 5.6 Plot of the error for the depth-only tracking and late fusion approach. Each approach was run with only the similarity term E_{sim} and with all terms. Notice the catastrophic tracking failure with the depth-only energy. The late fusion strategy is robust and prevents error accumulation. The collision penalty term also results in large accuracy gains. Best viewed in color.

per particle but at the expense of runtime. We therefore fixed the number of iterations to 10.

Tracking Speed: We tested the tracking speed of different variants of our method on a 3.6 GHz Intel Processor with 16 GB of RAM. Our method was parallelized using OpenMP but no GPU was used. All tests were done with the Intel Senz3D depth camera with a depth resolution of 320×240 and capture rate of 60 fps. The decision forest when loaded in memory used 1 GB because the trees were stored as full trees. This can be avoided by loading only nodes that are valid. The depth-only energy when used with 1 particle, and 10 iterations per particle ran at 120 fps. When 2 particles were used, the speed came down to 60 fps. The late fusion approach, when used with 2 particles (10 iterations per particle), achieved a framerate of **50** fps. Image acquisition, part labeling, preprocessing, and creating the Gaussian mixture representation took 2 ms. The optimization took between 18 and 20 ms.

5.7.2 Qualitative Results

We present several qualitative results from realtime sequences in Figure 5.7. The examples show motions with a wide range of finger articulations involving abduction, adduction, flexion, extension, and considerable occlusions. They also include common gestures such as the *v-sign* and pointing. In the boxes, we also show comparison with [86] and the Leap Motion on similar poses. We observe a finger sliding effect in both these methods. Pinching is an important gesture, but the Leap Motion produces sliding fingertips which makes it hard to

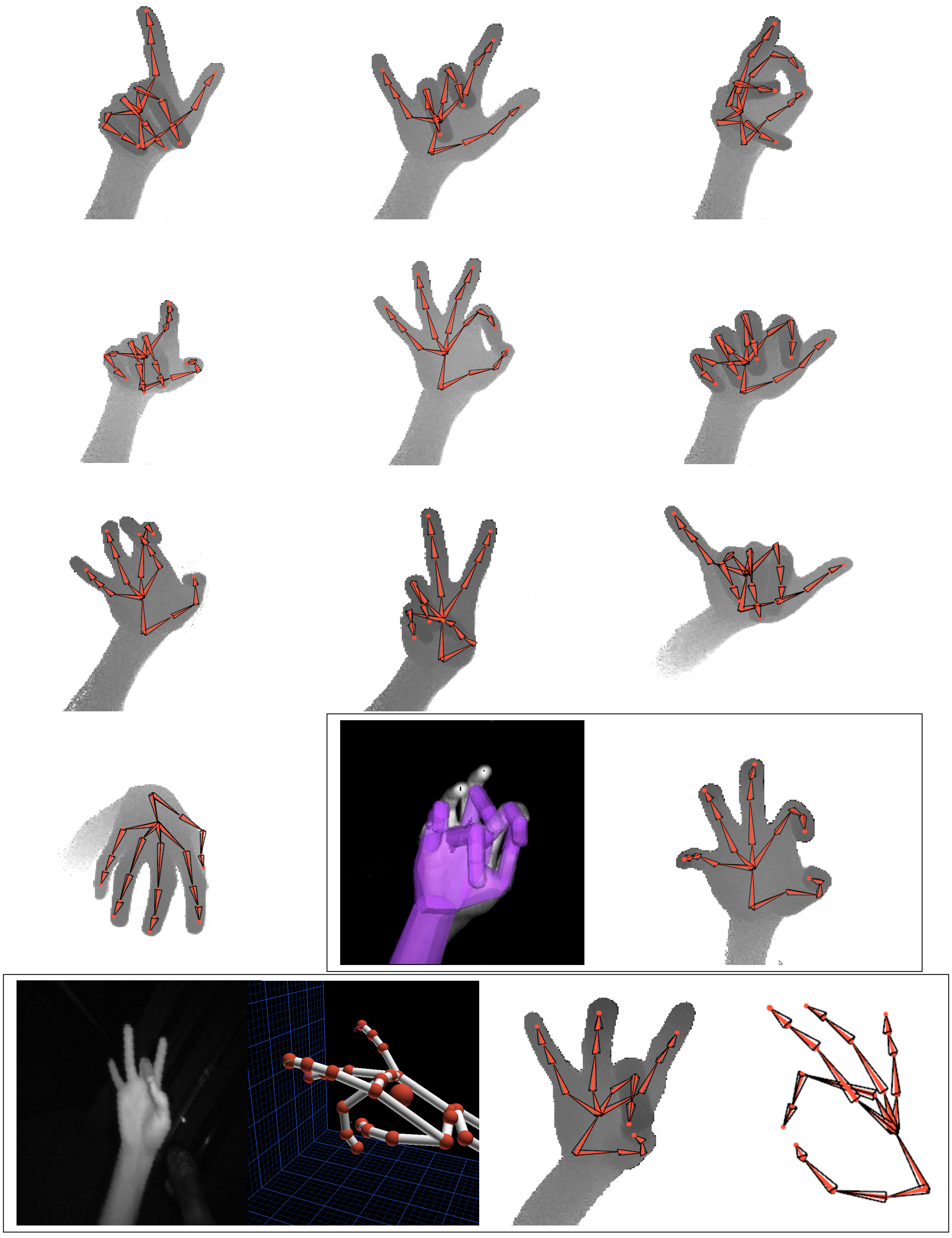


Figure 5.7 Qualitative results from our tracking approach (top row and four leftmost in the second row). The highlighted boxes show comparison with [86] and the Leap Motion both of which produce a finger sliding effect. Our method tracks the pinch faithfully.

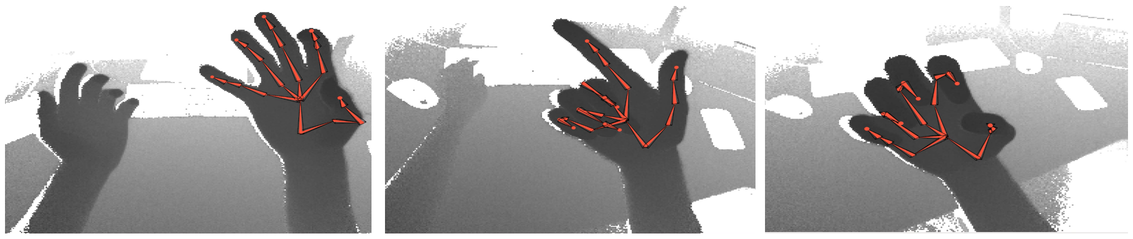


Figure 5.8 Tracking from egocentric viewpoint. Our method supports egocentric hand tracking of gestures like pointing and grasping.

detect pinching gestures from the tracked skeleton. Our method reproduces pinching faithfully as is evident from the skeleton overlaid on the depth image. Occasional tracking failures occur with large global rotations but the detection-guided energy eventually reinitializes to the correct pose.

5.8 Discussion

In this chapter, we presented a method for realtime hand tracking using detection-guided optimization. Our method is robust and tracks the hand at 50 FPS without using a GPU. We contribute to vision-based hand tracking research by proposing a novel representation of the input data and hand model using a mixture of Gaussians. This representation allows us to formulate pose estimation as an optimization problem and efficiently optimize it using analytic gradient. We also showed how additional evidence from part detection can be incorporated into our tracking framework to increase robustness. We evaluated our method on a publicly available dataset and compared with other state-of-the-art methods.

5.9 Conclusion

While our method can track hands from a single depth camera under reasonable background conditions, it would fail under heavy background clutter. There are two approaches to addressing this issue: (1) model clutter in the background and track them jointly with the hand, (2) segment background clutter completely, perhaps using multiple, layered random forests.

In the next chapter, we show how the first approach can be used to jointly track both objects and the hand. This brings the additional advantage of being able to use tracked object motion in interactive applications in addition to full 3D hand pose. We will also show that the strong analytic formulation offered by our method naturally extends to the hand-object tracking problem.

Chapter 6

Real-time Joint Tracking of a Hand Manipulating an Object

Thus far in this thesis, we have focussed on tracking only free hand motions. However, the human hand evolved not only for free hand motions but also for tool manipulation and interaction with the environment. Being able to track both hand motions and that of objects that it interacts with has the potential to create new interaction opportunities.

In this chapter, we solve the considerably harder problem of tracking objects in conjuction with the hand. To our knowledge, this is the first approach to achieve this under real-time tracking conditions. Parts of this chapter appeared in a previous publication [130].

6.1 Introduction

The human hand exhibits incredible capacity for manipulating external objects via gripping, grasping, touching, pointing, caging, and throwing. We can use our hands with apparent ease, even for subtle and complex motions, and with remarkable speed and accuracy. However, this dexterity also makes it hard to track a hand in close interaction with objects. While a lot of research has explored real-time tracking of hands or objects in isolation, real-time hand-object tracking remains unsolved. It is inherently more challenging due to the higher dimensionality of the problem, additional occlusions, and difficulty in disambiguating hand from object. A fast, accurate, and robust solution based on a minimal camera setup is a precondition for many new and important applications in vision-based input to computers, including virtual and augmented reality, teleoperation, tangible computing, and wearable computing. In this chapter, we present a **real-time** method to **simultaneously track** a hand and the manipulated object. We support tracking objects of **different shapes, sizes**, and

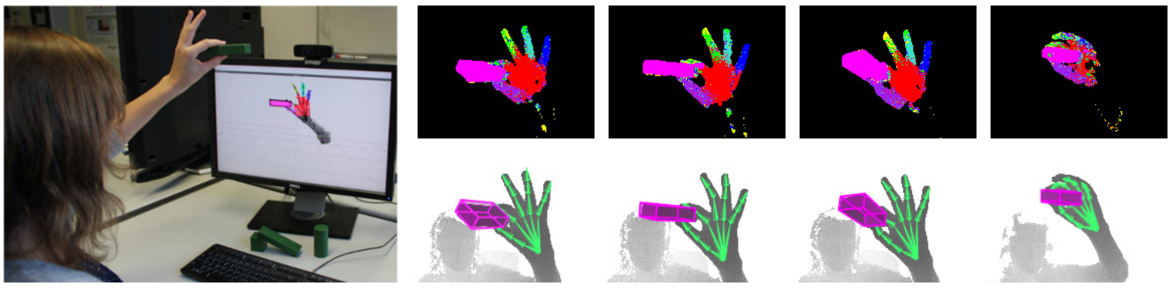


Figure 6.1 Proposed real-time hand-object tracking approach: we use a single commodity depth camera (*left*) to classify (*top*) and track the articulation of a hand and the rigid body motion of a manipulated object (*bottom*)

colors. Previous work has employed setups with multiple cameras [13, 96] to limit the influence of occlusions which restricts use to highly controlled setups. Many methods that exploit dense depth and color measurements from commodity RGB-D cameras [47, 72, 73] have been proposed. However, these methods use expensive segmentation and optimization steps that make interactive performance hard to attain. At the other end of the spectrum, discriminative one-shot methods (for tracking only hands) often suffer from temporal instability [62, 140, 166]. Such approaches have also been applied to estimate hand pose under object occlusion [114], but the object is not tracked simultaneously. In contrast, the approach proposed here is the first to track hand and object motion simultaneously at real-time rates using only a single commodity RGB-D camera (see Fig. 6.1). Building on recent work in single hand tracking and 3D point set registration, we propose a 3D articulated Gaussian mixture alignment strategy tailored to hand-object tracking. Gaussian mixture alignment aligns two Gaussian mixtures and has been successfully used in 3D pointset registration [57]. It can be interpreted as a generalization of ICP and does not require explicit, error-prone, and computationally expensive correspondence search [22]. Previous methods, such as those presented in Chapter 5, have used articulated 2.5D Gaussian mixture alignment formulations that are discontinuous. This leads to tracking instabilities because 3D spatial proximity is not considered. We also introduce additional novel regularizers that consider occlusions and enforce contact points between fingers and objects analytically. Our combined energy has a closed form gradient and allows for fast and accurate tracking. For an overview of our approach, see Figure 6.2. To further increase robustness and allow for recovery of the generative tracker, we guide the optimization using a multi-layer random forest hand part classifier. We use a variational optimization strategy that optimizes two different hand-object tracking energies simultaneously (multiple proposals) and then selects the better solution. The main contributions are:

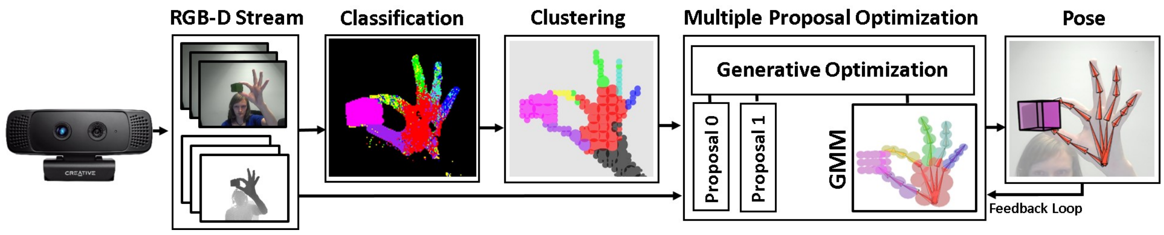


Figure 6.2 We perform classification of the input into object and hand parts. The hand and object are tracked using 3D articulated Gaussian mixture alignment.

- A 3D articulated Gaussian mixture alignment approach for jointly tracking hand and object accurately.
- Novel contact point and occlusion objective terms that were motivated by the physics of grasps, and can handle difficult hand-object interactions.
- A multi-layered classification architecture to segment hand and object, and classify hand parts in RGB-D sequences.
- An extensive evaluation on public datasets as well as a new, fully annotated dataset consisting of diverse hand-object interactions.

6.2 Related Work

Single Hand Tracking Single hand tracking has received a lot of attention in recent years with discriminative and generative methods being the two main classes of methods. Discriminative methods for monocular RGB tracking index into a large database of poses or learn a mapping from image to pose space [5, 162]. However, accuracy and temporal stability of these methods are limited. Monocular generative methods optimize pose of more sophisticated 3D or 2.5D hand models by optimizing an alignment energy [52, 20, 33]. Occlusions and appearance ambiguities are less problematic with multi-camera setups [13]. In [154], a physics-based approach to optimize the pose of a hand using silhouette and color constraints at slow non-interactive frame rates is presented. In Chapter 3 we showed how multiple RGB cameras and a single depth camera can be used to track single hand poses in near real-time by combining generative tracking and finger tip detection. More lightweight setups with a single depth camera are preferred for many interactive applications. Among single camera methods, examples of discriminative methods are based on decision forests for hand part labeling [62], on a latent regression forest in combination with a coarse-to-fine search [140], fast hierarchical pose regression [138], or Hough voting [166]. Real-time performance is feasible, but temporal instability remains an issue. Oikonomidis et al. [99] generatively track a hand by optimizing a depth and appearance-based alignment metric with particle swarm optimization (PSO). A real-time generative tracking method with a physics-based solver was proposed in [86]. The stabilizaton of real-time articulated ICP based on a learned subspace prior on hand poses was used in [139]. Template-based non-rigid deformation tracking of arbitrary objects in real-time from RGB-D was shown in [173], very simple unoccluded hand poses can be tracked. Combining generative and discriminative tracking enables recovery from some tracking failures [119, 147, 131]. In Chapter 5, we showed real-time single hand tracking from depth using generative pose optimization under detection constraints. Similarly, reinitialization of generative estimates via finger tip detection [108], multi-layer discriminative reinitialization [119], or joints detected with convolutional networks is feasible [144]. Tang et al. [141] employ hierarchical sampling from partial pose distributions and a final hypothesis selection based on a generative energy. None of the above methods is able to track interacting hands and objects simultaneously and in non-trivial poses in real-time.

Tracking Hands in Interaction Tracking two interacting hands, or a hand and a manipulated object, is a much harder problem. The straightforward combination of methods for object tracking, e.g. [8, 143], and hand tracking does not lead to satisfactory solutions, as only a combined formulation can methodically exploit mutual constraints between object and hand. Wang et al. [152] track two well-separated hands from stereo by efficient pose retrieval and IK refinement. In [97] two hands in interaction are tracked at 4 Hz with an RGB-D camera by optimizing a generative depth and image alignment measure. Tracking of interacting hands from multi-view video at slow non-interactive runtimes was shown in [13]. They use generative pose optimization supported by salient point detection. The method in [139] can track very simple two hand interactions with little occlusion. Commercial solutions, e.g. Leap Motion [1] and NimbleVR [2], fail if two hands interact closely or interact with an object. In [96], a marker-less method based on a generative pose optimization of a combined hand-object model is proposed. They explicitly model collisions, but need multiple RGB cameras. In [47] the most likely pose is found through belief propagation using part-based trackers. This method is robust under occlusions, but does not explicitly track the object. A temporally coherent nearest neighbor search tracks the hand manipulating an object in [114], but not the object, in real time. Results are prone to temporal jitter. Kyriazis et al. [72] perform frame-to-frame tracking of hand and objects from RGB-D using physics-based optimization. This approach has a slow non-interactive runtime. An ensemble of Collaborative Trackers (ECT) for RGB-D based multi-object and multiple hand tracking is used in [73]. Their accuracy is high, but runtime is far from real-time. Pham et al. [103] infer contact forces from a tracked hand interacting with an object at slow non-interactive runtimes. In [100] and [146],

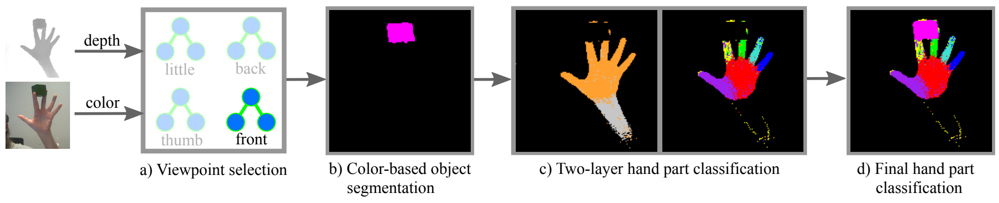

depth
color
little
back
thumb
front
a) Viewpoint selection
b) Color-based object segmentation
c) Two-layer hand part classification
d) Final hand part classification

Figure 6.3 Three stage hand part classification. **Stage 1**: Viewpoint selection, **Stage 2**: color-based object segmentation, **Stage 3**: two-layer hand part classification.

methods for in-hand RGB-D object scanning are proposed. Both methods use known generative methods to track finger contact points to support ICP-like shape scanning. Recently, [145] introduced a method for tracking hand-only, hand-hand, and hand-object (we include a comparison with this method). None of the above methods can track the hand and the manipulated object in *real-time* in non-trivial poses from a *single depth camera* view, which is what our approach achieves.

Model-based Tracking Approaches A common representation for model tracking are meshes [13, 139]. Other approaches use primitives [73, 108], quadrics [134], 2.5D Gaussians (see Chapter 5), or Gaussian mixtures [57]. Gaussian mixture alignment has been successfully used in rigid point set registration [57]. In contrast, we propose a 3D *articulated* Gaussian mixture alignment strategy. [168] relate template and data via a probabilistic formulation and use EM to compute the best fit. Different from our approach, they only model the template as a Gaussian mixture.

6.3 Discriminative Hand Part Classification

As a preprocessing step, we classify depth pixels as hand or object, and further into hand parts. The obtained labeling is later used to guide the generative pose optimization. Our part classification strategy is based on a two-layer random forest that takes occlusions into account. Classification is based on a three step pipeline (see Fig. 6.3). Input is the color $\mathcal{C}_t$ and depth $\mathcal{D}_t$ frames captured by the RGB-D sensor. We first perform hand-object segmentation based on color cues to remove the object from the depth map. Afterwards, we select a suitable two-layer random forest to obtain the classification. The final output per pixel is a part probability histogram that encodes the class likelihoods. Note, object pixel histograms are set to an object class probability of 1. The forests are trained based on a set of training images that consists of real hand motions re-targeted to a virtual hand model to generate synthetic data from multiple viewpoints. A virtual object is automatically inserted in the scene to simulate occlusions. To this end, we randomly sample uniform object positions between the thumb and one other finger and prune implausible poses based on intersection tests.

Viewpoint Selection We trained two-layer forests for hand part classification from different viewpoints. Four cases are distinguished: observing the hand from the front, back, thumb and little finger sides. We select the forest that best matches the hand orientation computed in the last frame. The selected two-layer forest is then used for hand part classification.

Color-Based Object Segmentation As a first step, we segment out the object from the captured depth map $\mathcal{D}_t$. Similar to many previous hand-object tracking approaches [99], we use the color image $\mathcal{C}_t$ in combination with an HSV color segmentation strategy. As we show in the results, we are able to support objects with different colors. Object pixels are removed to obtain a new depth map $\hat{\mathcal{D}}_t$, which we then feed to the next processing stage.

Two-Layer Hand Part Classification We use a two-layer random forest for hand part classification. The first layer classifies hand and arm pixels while the second layer uses the hand pixels and further classifies them into one of several distinct hand parts. Both layers are per-pixel classification forests [121]. The hand-arm classification forest is trained on $N = 100k$ images with diverse hand-object poses. For each of the four viewpoints a random forest is trained on $N = 38k$ images. The random forests are based on three trees, each trained on a random distinct subset. In each image, 2000 example foreground pixels are chosen. Split decisions (see Chapter 2) at nodes are based on 100 random feature offsets and 40 thresholds. Candidate features are a uniform mix of unary and binary depth difference features [121]. Nodes are split as long as the information gain is sufficient and the maximum tree depth of 19 (21 for hand-arm forest) has not been reached. On the first layer, we use 3 part labels: 1 for hand, 1 for arm and 1 to represent the background. On the second layer, classification is based on 7 part labels: 6 for the hand parts, and 1 for the background. We use one label for each finger and one for the palm, see Fig. 6.3c. We use a cross-validation procedure to find the best hyperparameters. On the disjoint test set, the hand-arm forest has a classification accuracy of 65.2%. The forests for the four camera views had accuracies of 59.8% (front), 64.7% (back), 60.9% (little), and 53.5% (thumb).

6.4 Gaussian Mixture Model Representation

Joint hand-object tracking requires a representation that allows for accurate tracking, is robust to outliers, and enables fast pose optimization. Gaussian mixture alignment, initially proposed for rigid pointset alignment (e.g. [57]), satisfies all these requirements. It features the advantages of ICP-like methods, without requiring a costly, error-prone correspondence search. We extend this approach to 3D articulated Gaussian mixture alignment tailored to hand-object tracking. Compared to this 3D formulation, the 2.5D formulation in Chapter 5 is discontinuous. This causes instabilities, since the spatial proximity between model and data is not fully considered. We quantitatively show this for hand-only tracking (Section 6.8).

6.5 Unified Density Representation

We parameterize the articulated motion of the human hand using a kinematic skeleton with $|\mathcal{X}_h| = 26$ degrees of freedom (DOF). Non-rigid hand motion is expressed based on 20 joint angles in twist representation. The remaining 6 DOFs specify the global rigid transform of the hand with respect to the root joint. The manipulated object is assumed to be rigid and its motion is parameterized using $|\mathcal{X}_o| = 6$ DOFs. In the following, we deal with the hand and object in a unified way. To this end, we refer to the vector of all unknowns as $\mathcal{X}$. For pose optimization, both the input depth as well as the scene (hand and object) are expressed as 3D Gaussian Mixture Models (GMMs). This allows for fast and analytical pose optimization. We first define the following generic probability density distribution $\mathcal{M}(\mathbf{x}) = \sum_{i=1}^{K} w_i \mathcal{G}_i(\mathbf{x}|\boldsymbol{\mu}_i, \sigma_i)$ at each point $\mathbf{x} \in \mathbb{R}^3$ in space. This mixture contains K unnormalized, isotropic Gaussian functions $\mathcal{G}_i$ with mean $\boldsymbol{\mu}_i \in \mathbb{R}^3$ and variance $\sigma_i^2 \in \mathbb{R}$. In the case of the model distribution, the positions of the Gaussians are parameterized by the unknowns $\mathcal{X}$. For the hand, this means each Gaussian is being rigidly rigged to one bone of the hand. The probability density is defined and non-vanishing over the whole domain $\mathbb{R}^3$.

Hand and Object Model The three-dimensional shape of the hand and object is represented in a similar fashion as probability density distributions $\mathcal{M}_h$ and $\mathcal{M}_o$, respectively. We manually attach $N_h = 30$ Gaussian functions to the kinematic chain of the hand to model its volumetric extent. Standard deviations are set such that they roughly correspond to the distance to the actual surface. The object is represented by automatically fitting a predefined number N_o of Gaussians to its spatial extent, such that the one standard deviation spheres model the object's volumetric extent. N_o is a user defined parameter which can be used to control the trade-off between tracking accuracy and runtime performance. We found that $N_o \in [12, 64]$ provides a good trade-off between speed and accuracy for the objects used in our experiments. We refer to the combined hand-object distribution as $\mathcal{M}_s$, with $N_s = N_h + N_o$ Gaussians. Each Gaussian is assigned to a class label l_i based on its semantic location in the scene. Note, the input GMM is only a model of the visible surface of the hand/object. Therefore, we incorporate a visibility factor $f_i \in [0, 1]$ (0 completely occluded, 1 completely visible) per Gaussian. This factor is approximated by rendering an occlusion map with each Gaussian as a circle (radius equal to its standard deviation). The GMM is restricted to the visible surface by setting $w_i = f_i$ in the mixture. These operations are performed based on the solution of the previous frame $\mathcal{X}_{old}$.

Input Depth Data We first perform bottom-up hierarchical quadtree clustering of adjacent pixels with similar depth to convert the input to the density based representation. We cluster at most $(2^{(4-1)})^2 = 64$ pixels, which corresponds to a maximum tree depth of 4. Clustering is performed as long as the depth variance in the corresponding subdomain is smaller than $\epsilon_{cluster} = 30$ mm. Each leaf node is represented as a Gaussian function $\mathcal{G}_i$ with $\boldsymbol{\mu}_\mathbf{i}$ corresponding to the 3D center of gravity of the quad and $\sigma_i^2 = (\frac{a}{2})^2$, where a is the backprojected side length of the quad. Note, the mean $\boldsymbol{\mu}_\mathbf{i} \in \mathbb{R}^3$ is obtained by backprojecting the 2D center of gravity of the quad based on the computed average depth and displacing by a in camera viewing direction to obtain a representation that matches the model of the scene. In addition, each $\mathcal{G}_i$ stores the probability p_i and index l_i of the best associated semantic label. We obtain the best label and its probability by summing over all corresponding per-pixel histograms obtained in the classification stage. Based on this data, we define the input depth distribution $\mathcal{M}_{d_h}(\mathbf{x})$ for the hand and $\mathcal{M}_{d_o}(\mathbf{x})$ for the object. The combined input distribution $\mathcal{M}_d(\mathbf{x})$ has $N_d = N_{d_o} + N_{d_h}$ Gaussians. We set uniform weights $w_i = 1$ based on the assumption of equal contribution. N_d is much smaller than the number of pixels leading to real-time hand-object tracking.

6.6 Multiple Proposal Optimization

We optimize for the best pose $\mathcal{X}^*$ using two proposals $\mathcal{X}_i^*,\ i \in \{0, 1\}$ that are computed by minimizing two distinct hand-object tracking energies:

$$\mathcal{X}_0^* = \operatorname*{argmin}_{\mathcal{X}} E_{align}(\mathcal{X}),\ \mathcal{X}_1^* = \operatorname*{argmin}_{\mathcal{X}} E_{label}(\mathcal{X})\ . \tag{6.1}$$

E_{align} leverages the depth observations and the second energy E_{label} incorporates the discriminative hand part classification results. In contrast to the optimization of the sum of the two objectives, this avoids failure due to bad classification and ensures fast recovery. For optimization, we use analytical gradient descent (10 iterations per proposal, adaptive step length) [135]. We initialize based on the solution of the previous frame $\mathcal{X}_{old}$. Finally, $\mathcal{X}^*$ is selected as given below, where we slightly favor ($\lambda = 1.003$) the label proposal to facilitate fast pose recovery:

$$\mathcal{X}^* = \begin{cases} \mathcal{X}_1^* & \text{if } (E_{val}(\mathcal{X}_1^*) < \lambda E_{val}(\mathcal{X}_0^*)) \\ \mathcal{X}_0^* & \text{otherwise} \end{cases} . \tag{6.2}$$

The energy $E_{val}(\mathcal{X}) = E_a(\mathcal{X}) + w_p E_p(\mathcal{X})$ is designed to select the proposal that best explains the input, while being anatomically correct. Therefore, it considers spatial alignment to the input depth map E_a and models anatomical joint angle limits E_p, see Section 6.7. In the following, we describe the used energies in detail.

6.7 Hand-Object Tracking Objectives

Given the input depth distribution $\mathcal{M}_d$, we want to find the 3D model $\mathcal{M}_s$ that best explains the observations by varying the corresponding parameters $\mathcal{X}$. We take inspiration from methods with slow non-interactive runtimes that used related 3D implicit shape models for full-body pose tracking [106, 71], but propose a new efficient tracking objective tailored for real-time hand-object tracking. In contrast to previous methods, our objective operates in 3D (generalization of ICP), features an improved way of incorporating the discriminative classification results, and incorporates two novel regularization terms. Together, this provides for a better, yet compact, representation that allows for fast analytic pose optimization on the CPU. To this end, we define the following two objective functions. The first energy E_{align} measures the alignment with the input:

$$E_{align}(\mathcal{X}) = E_a + w_p E_p + w_t E_t + w_c E_c + w_o E_o . \tag{6.3}$$

The second energy E_{label} incorporates the classification results:

$$E_{label}(\mathcal{X}) = E_a + w_s E_s + w_p E_p . \tag{6.4}$$

The energy terms consider spatial alignment E_a, semantic alignment E_s, anatomical plausibility E_p, temporal smoothness E_t, contact points E_c, and object-hand occlusions E_o, respectively. The priors in the energies are chosen such that they do not hinder the respective alignment objectives. All parameters $w_p = 0.1, w_t = 0.1, w_s = 3 \cdot 10^{-7}, w_c = 5 \cdot 10^{-7}$ and $w_o = 1.0$ have been empirically determined and stay fixed for all experiments. We optimize both energies simultaneously using a multiple proposal based optimization strategy and employ a winner-takes-all strategy (see Section 6.6). We found empirically that using two energy functions resulted in better pose estimation and recovery from failures than using a single energy with all terms. In the following, we give more details on the individual components.

Spatial Alignment We measure the alignment of the input density function $\mathcal{M}_d$ and our scene model $\mathcal{M}_s$ based on the following alignment energy:

$$E_a(\mathcal{X})=\int_{\Omega}\Big[\big(\mathcal{M}_{d_h}(\mathbf{x})-\mathcal{M}_h(\mathbf{x})\big)^2+\big(\mathcal{M}_{d_o}(\mathbf{x})-\mathcal{M}_o(\mathbf{x})\big)^2\Big]d\mathbf{x}\,. \tag{6.5}$$

It measures the alignment between the two input and two model density distributions at every point in space $\mathbf{x} \in \Omega$. Note, this 3D formulation leads to results of higher accuracy (see Section 6.8) than the 2.5D formulation presented in Chapter 5.

Semantic Alignment In addition to the alignment of the distributions, we also incorporate semantic information in the label energy E_{label}. In contrast to Chapter 5, we incorporate uncertainty based on the best class probability. We use the following least-squares objective to enforce semantic alignment:

$$E_s(\mathcal{X}) = \sum_{i=1}^{N_s}\sum_{j=1}^{N_d} \alpha_{i,j} \cdot ||\boldsymbol{\mu}_\mathbf{i} - \boldsymbol{\mu}_\mathbf{j}||_2^2\,. \tag{6.6}$$

Here, $\boldsymbol{\mu}_\mathbf{i}$ and $\boldsymbol{\mu}_\mathbf{j}$ are the mean of the i^{th} model and the j^{th} image Gaussian, respectively. The weights $\alpha_{i,j}$ switch attraction forces between similar parts on and between different parts off:

$$\alpha_{i,j} = \begin{cases} 0 & \text{if } (l_i \neq l_j) \text{ or } (d_{i,j} > r_{max}) \\ (1 - \frac{d_{i,j}}{r_{max}}) \cdot p_i & \text{else} \end{cases}\,. \tag{6.7}$$

Here, $d_{i,j} = ||\boldsymbol{\mu}_\mathbf{i} - \boldsymbol{\mu}_\mathbf{j}||_2$ is the distance between the means. l_i is the part label of the most likely class, p_i its probability and r_{max} a cutoff value. We set r_{max} to 30cm. l_i can be one of 8 labels: 6 for the hand parts, 1 for object and 1 for background. We consider all model Gaussians, independent of their occlusion weight, to facilitate fast pose recovery of previously occluded regions.

Anatomical Plausibility The articulated motion of the hand is subject to anatomical constraints. We account for this by enforcing soft-constraints on the joint angles $\mathcal{X}_h$ of the hand:

$$E_p(\mathcal{X}) = \sum_{x_i \in \mathcal{X}_h} \begin{cases} 0 & \text{if } x_i^l \leq x_i \leq x_i^u \\ ||x_i - x_i^l||^2 & \text{if } x_i < x_i^l \\ ||x_i^u - x_i||^2 & \text{if } x_i > x_i^u \end{cases}\,. \tag{6.8}$$

Here, $\mathcal{X}_h$ are the DOFs corresponding to the hand, and x_i^l and x_i^u are the lower and upper joint limit that corresponds to the i^{th} DOF of the kinematic chain.

Temporal Smoothness We further improve the smoothness of our tracking results by incorporating a temporal prior into the energy. To this end, we include a soft constraint on parameter change to enforce constant speed:

$$E_t(\mathcal{X}) = ||\nabla\mathcal{X} - \nabla\mathcal{X}^{(t-1)}||_2^2 \, . \tag{6.9}$$

Here, $\nabla\mathcal{X}^{(t-1)}$ is the gradient of parameter change at the previous time step.

Contact Points We propose a novel contact point objective, specific to the hand-object tracking scenario:

$$E_c(\mathcal{X}) = \sum_{(k,l,t_d)\in\mathcal{T}} \left(||\boldsymbol{\mu}_\mathbf{k} - \boldsymbol{\mu}_\mathbf{l}||^2 - t_d^2 \right)^2 . \tag{6.10}$$

Here, $(k, l, t_d) \in \mathcal{T}$ is a detected touch constraint. It encodes that the fingertip Gaussian with index k should have a distance of t_d to the object Gaussian with index l. We detect the set of all touch constraints $\mathcal{T}$ based on the last pose $\mathcal{X}_{old}$. A new touch constraint is added if a fingertip Gaussian is closer to an object Gaussian than the sum of their standard deviations. We then set t_d to this sum. This couples hand pose and object tracking leading to more stable results. A contact point is active until the distance between the two Gaussians exceeds the release threshold δ_R. Usually $\delta_R > t_d$ to avoid flickering.

Occlusion Handling No measurements are available in occluded hand regions. We stabilize the hand movement in such regions using a novel occlusion prior:

$$E_o(\mathcal{X}) = \sum_{i=0}^{N_h} \sum_{j\in\mathcal{H}_i} (1 - \hat{f}_i) \cdot ||x_j - x_j^{old}||_2^2 \, . \tag{6.11}$$

Here, $\mathcal{H}_i$ is the set of all DOFs that are influenced by the i-th Gaussian. The global rotation and translation is not included. The occlusion weights $\hat{f}_i \in [0, 1]$ are computed similar to f_i (0 occluded, 1 visible). This prior is based on the assumption that occluded regions move consistently with the rest of the hand.

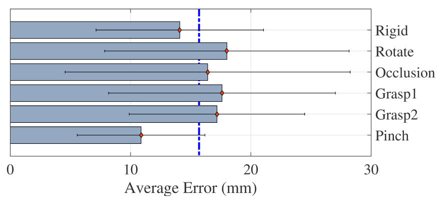
6.8 Experiments and Results

We evaluate and compare our method on more than **15 sequences** spanning 3 public datasets, which have been recorded with 3 different RBG-D cameras. Additional live sequences (see Fig. 6.8) show that our method handles fast object and finger motion, difficult occlusions and fares well even if two hands are present in the scene. Our method supports commodity RGB-D sensors like the *Creative Senz3D*, *Intel RealSense F200*, and *Primesense Carmine*. We rescale depth and color to resolutions of 320×240 and 640×480 respectively, and capture at 30 Hz. Furthermore, we introduce a new hand-object tracking benchmark dataset with ground truth fingertip and object annotations.

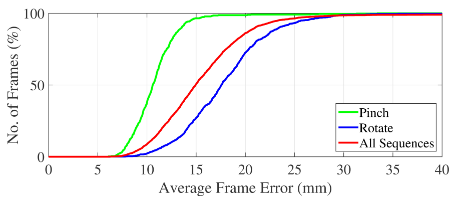
Comparison to the State-of-the-Art We quantitatively and qualitatively evaluate on two publicly available hand-object datasets [145, 146] (see Fig. 6.8). Only one dataset (IJCV [145]) contains ground truth joint annotations. We test on 5 rigid object sequences from IJCV. We track the right hand only, but our method works even when multiple hands are present. Ground truth annotations are provided for 2D joint positions, but not object pose. Our method achieves a fingertip pixel error of **8.6px**, which is comparable (difference of only 2px) to that reported for the slower method of [145]. This small difference is well within the uncertainty of manual annotation and sensor noise. Note, our approach runs over 60 times faster, while producing visual results that are on par (see Fig. 6.8). We also track the dataset of [146] (see also Fig. 6.8). While they solve a different problem (offline in-hand scanning), it shows that our real-time method copes well with different shaped objects (e.g., bowling pin, bottle, etc.) under occlusion.

New Benchmark Dataset With the aforementioned datasets, evaluation of object pose is impossible due to missing object annotations. We therefore introduce, to our knowledge, the first dataset[1] that contains ground truth for **both** fingertip positions and object pose. It contains 6 sequences of a hand manipulating a cuboid (2 different sizes) in different hand-object configurations and grasps. We manually annotated pixels on the depth image to mark 5 fingertip positions, and 3 cuboid corners. In total, we provide 3014 frames with ground truth annotations. As is common in the literature [119, 140, 129, 108, 139], we use the average 3D Euclidean distance E between estimated and ground truth positions as the error measure. Occluded fingertips are excluded on a per-frame basis from the error computation. If one of the annotated corners of the cuboid is occluded, we exclude it from that frame. In Fig. 6.4a we plot the average error over all frames of the 6 sequences. Our method has an average

[1]http://handtracker.mpi-inf.mpg.de/projects/RealtimeHO/



(a) We achieve low errors on each of the 6 sequences in our new benchmark dataset.



(b) Tracking consistency of the best, worst and average case.

Figure 6.4 Quantitative hand-object tracking evaluation on ground truth data. The object contributes a higher error.

Table 6.1 Average error (mm) for hand and object tracking in our dataset

	Rigid	*Rotate*	*Occlusion*	*Grasp1*	*Grasp2*	*Pinch*	**Overall (mm)**
Fingertips	14.2	16.3	17.5	18.1	17.5	10.3	**15.6**
Object	13.5	26.8	11.9	15.3	15.7	13.9	**16.2**
Combined (E)	14.1	18.0	16.4	17.6	17.2	10.9	**15.7**

error (for both hand and object) of **15.7 mm**. Over all sequences, the average error is always lower than 20 mm with standard deviations under 12 mm. Average error is an indicator of overall performance, but does not indicate how consistent the tracker performs. Fig. 6.4b shows that our method tracks almost all frames with less than 30 mm error. *Rotate* has the highest error, while *Pinch* performs best with almost all frames below 20 mm. Table 6.1 shows the errors for hand and object separately. Both are in the same order of magnitude.

Ablative Analysis Firstly, we show that the articulated 3D Gaussian mixture alignment formulation is superior (even for tracking only hand) to the 2.5D formulation described in Chapter 5. On the Dexter dataset [131], [129] report an average fingertip error of **19.6 mm**. In contrast, our method (**without** any hand-object specific terms) is consistently better with an average of **17.2 mm** (maximum improvement is **5 mm** on 2 sequences). This is a result of the continuous articulated 3D Gaussian mixture alignment energy, a generalization of ICP, which considers 3D spatial proximity between Gaussians.

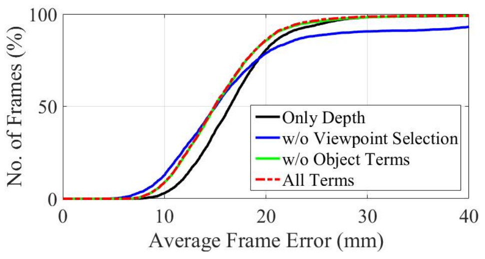


Figure 6.6 Ablative analysis.

Secondly, we show that the average error on our hand-object dataset is worse without viewpoint selection, semantic alignment, occlusion handling, and contact points term. Fig. 6.6 shows a consistency plot with different components of the energy disabled. Using

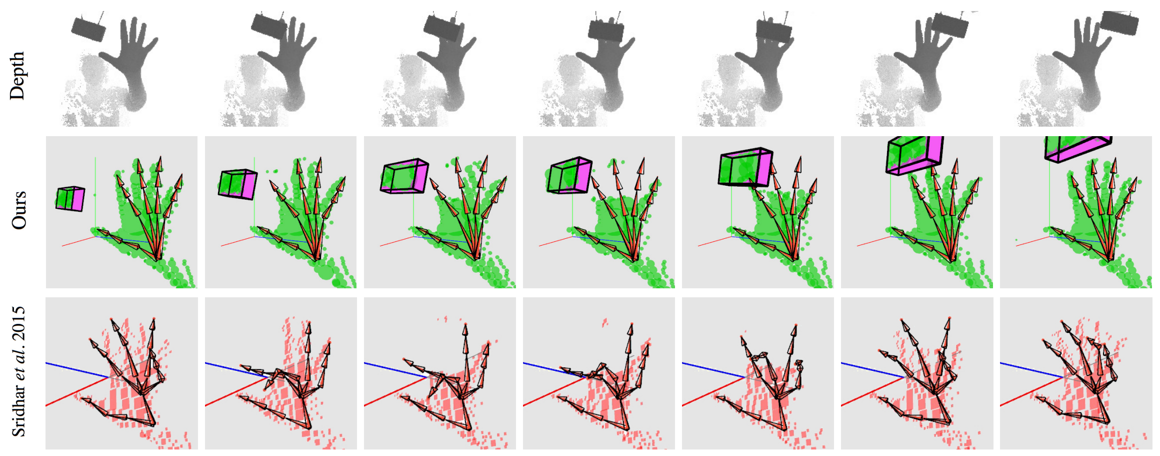
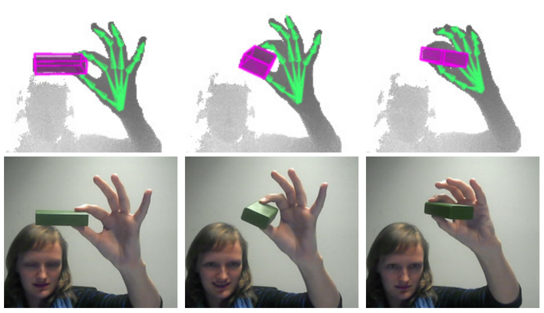


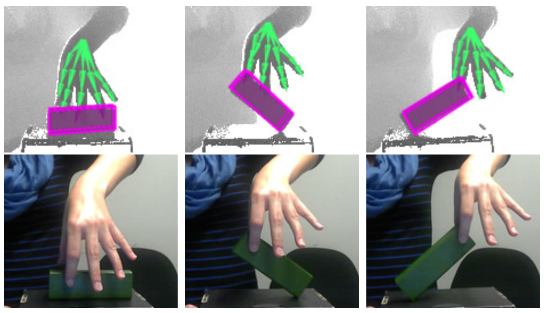
Figure 6.5 *Top row:* Input depth, an object occludes the hand. *Middle row:* Result of our approach (different viewpoint). Our approach succesfully tracks the hand under heavy occlusion. *Bottom row:* Result of [129] shows catastrophic failure (object pixels were removed for fairness)

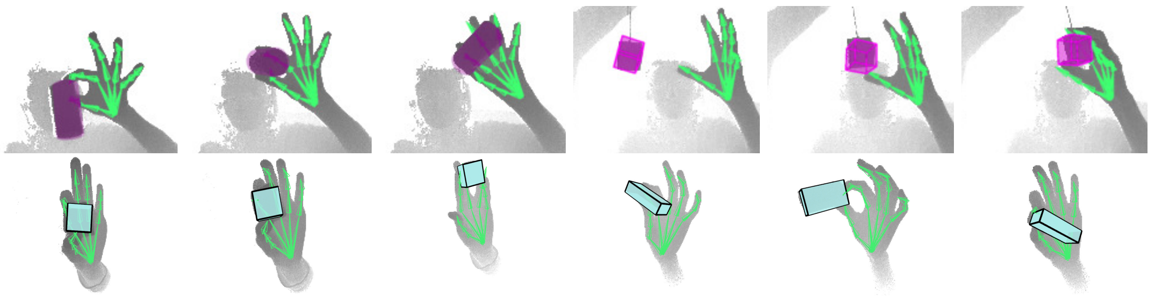
only the data term often results in large errors. The errors are even larger without viewpoint selection. The semantic alignment, occlusion handling, and contact points help improve **robustness** of tracking results and **recovery** from failures. Fig. 6.5 shows that [129] clearly fails when fingers are occluded. Our hand-object specific terms are more robust to these difficult occlusion cases while achieving real-time performance.

Runtime Performance All experiments were performed on an Intel Xeon E5-1620 CPU with 16 GB memory and an NVIDIA GTX 980 Ti. The stages of our approach take on average: 4 ms for preprocessing, 4 ms for part classification, 2 ms for depth clustering, and 20-30 ms for pose optimization using two proposals. We achieve real-time performance of 25-30 Hz. Multi-layer random forests ran on the GPU while all other algorithm parts ran multithreaded on a CPU.

Limitations Although we demonstrated robustness against reasonable occlusions, situations where a high fraction of the hand is occluded for a long period are still challenging. This is mostly due to degraded classification performance under such occlusions. Misalignments can appear if the underlying assumption of the occlusion heuristic is violated, i. e. occluded parts do not move rigidly. Fortunately, our discriminative classification strategy enables the pose optimization to recover once previously occluded regions become visible again as shown in Fig. 6.9. Further research has to focus on better priors for occluded regions, for example grasp and interaction priors learned from data. Also improvements to hand part


(a) *Rotate* sequence from our dataset


(b) *Grasp2* sequence from our dataset


(c) Real-time tracking results with various object shapes and different users

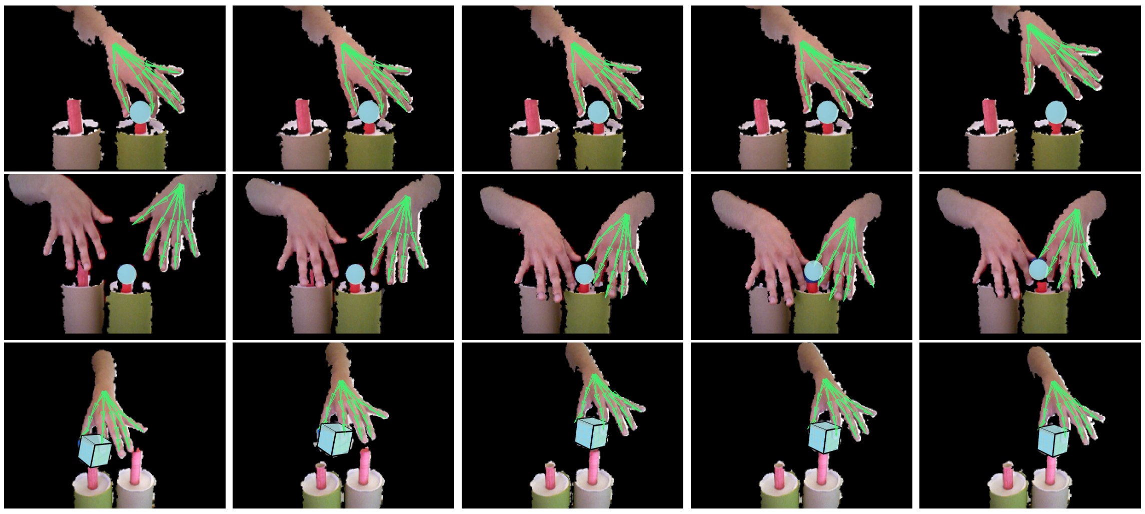

(d) Results on the IJCV dataset [145]. Notice how our method tracks the hand even if multiple hands are in view. Tracked skeleton in green and object in light blue

Figure 6.7 (a, b) show tracking results on our dataset. (c) shows real-time results with different object shapes and colors. (d) shows results on a public dataset

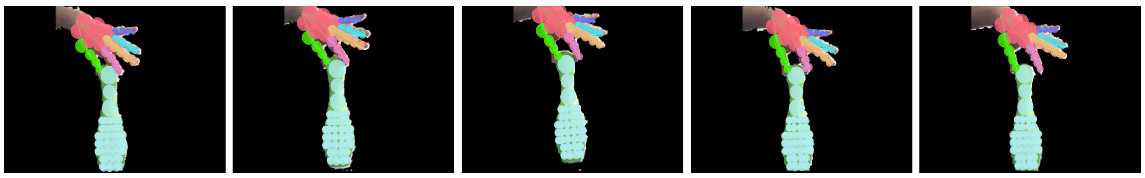


Figure 6.8 Subset of tracked frames on the dataset of [146]. Our method can handle objects with **varying sizes, colors, and different hand dimensions**. Here we show how even a complex shape like a bowling pin can be approximated using only a few tens of Gaussians

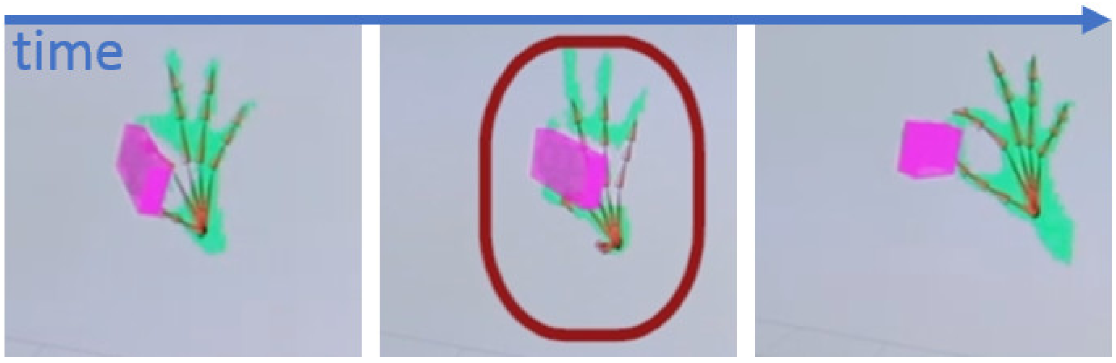


Figure 6.9 Occlusion error and recovery.

classification using different learning approaches or the regression of dense correspondences are interesting topics for future work. Another source of error are very fast motions. While the current implementation achieves 30 Hz, higher frame rate sensors in combination with a faster pose optimization will lead to higher robustness due to improved temporal coherence. We show diverse object shapes being tracked. However, increasing object complexity (shape and color) affects runtime performance. We would like to further explore how multiple complex objects and hands can be tracked.

6.9 Discussion

In this chapter, we have presented the first real-time approach for simultaneous hand-object tracking based on a single commodity depth sensor. Our approach combines the strengths of discriminative classification and generative pose optimization. Classification is based on a multi-layer forest architecture with viewpoint selection. We use 3D articulated Gaussian mixture alignment tailored for hand-object tracking along with novel analytic occlusion and contact handling constraints that enable successful tracking of challenging hand-object interactions based on multiple proposals. Our qualitative and quantitative results demonstrate that our approach is both accurate and robust. Additionally, we have captured a new benchmark dataset (with hand and object annotations) and make it publicly available.

6.10 Conclusion

Future work in joint hand and object tracking needs to address several issues. First, we track only a limited set of objects with known shapes. For any approach to have practical applications a wide variety of object classes need to be tracked. Future work also needs to address difficult occlusions that occur when objects occlude fingers. We feel that strong priors for occluded hand parts will play an important role.

Part I of this thesis has presented methods for hand tracking from multiple RGB cameras and single depth sensors. We presented one of the first methods to track hands in real-time from a single depth camera in Chapter 5. In this chapter, we showed how real-time joint hand and object tracking is achievable. These methods form the basis for the gesture-based input techniques that will be described in Part II of this thesis.

Part II

Gesture-based Computer Input

Chapter 7

Continuous Computer Input

Part I of the thesis dealt with the computer vision problem of tracking hands in action under different conditions and run-time requirements. We showed how multiple cameras or a single depth sensor can be used to track the fine motion of hands and also objects that it interacts with. The solution to the hand tracking problem encompasses only half of the challenges involved in gesture-based computer input. In order to realize the final goal of using hand motion for computer input, we need to devise methods to transform tracking data (e.g., parameters of the kinematic skeleton) into meaningful interactions.

In Part II of this thesis, we introduce ways of gesture-based computer input. We do so by introducing new interaction techniques, and presenting empirical data, user studies and working examples. We divide this part into 3 chapters loosely based on the type of input: **continuous**, **discrete**, and **combined**.

In this chapter, we present, *FullHand*, a method to map hand motions to continuous computer input for use in games and virtual globe navigation. We assume that tracked hand motion data in the form of pose parameters of a kinematic skeleton are available. Any of the methods described in Part I can be used for this purpose. In this chapter, we use and further develop the method presented in Chapter 3. We show how elicitation studies, i.e., asking several users to perform gestures that occur to them naturally for a certain task, can be used to design interaction techniques for 3D navigation tasks. Parts of this chapter appeared in an earlier publication [126].

7.1 Introduction

Exploiting the exceptional dexterity of the human hand for computer input has been a prime goal for research on input devices and interaction techniques. *Hand articulation* refers to the coordinated movement of the 27 bones controlled by 38 muscles in the hand and the

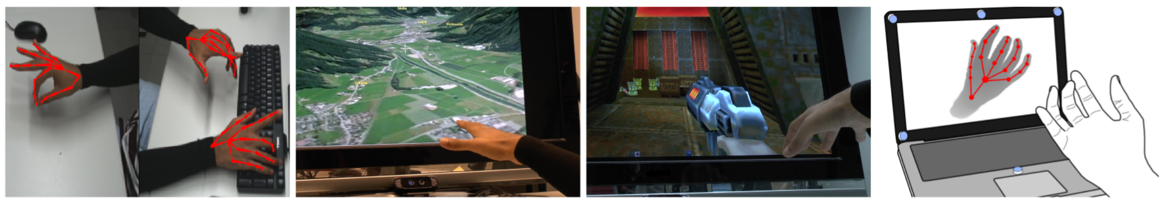


Figure 7.1 (a) FullHand can track one or both hands with all fingers. (b, c) FullHand enables free-hand interactions for many applications such as virtual globes and first-person shooters. (d) We envision miniature multi-camera setups for hand tracking in the future. The blue cylinders represent cameras.

forearm [59]. Fingers are the most precisely controllable parts of the body in spite of high angular velocity in their movement. Although all DOFs cannot be independently controlled, individuation of finger control becomes virtually perfect with practice [59]. However, common input devices used today, such as the mouse, tap only into a fraction of the hand's capacity.

Several tracking methods have been proposed to capture the articulation of the hand for interactive applications. They can be classified into two categories. (1) *Contact-based* methods measure joint angles with instrumented gloves, or they use fiducial markers on the skin tracked by cameras [172, 137]. However, these methods restrict free motion of the hand, and they can be uncomfortable and unpractical for users. (2) *Non-contact* methods, typically based on computer vision, do not require contacting sensors. However, existing methods have limitations related to the set of DOFs they capture or interactive performance. For instance, the Leap Motion tracks only salient points like fingertips, and only succeeds under a constrained range of hand orientations. This restricts designers to a narrow set of free-hand interactions.

In this chapter, we present *FullHand*, a system for hand motion tracking and interaction. FullHand tracks the motion of the hand using a *kinematic skeleton* that captures the major rotational and translational degrees of freedom of the hand using a modified version of the method presented in Chapter 3. FullHand has several advantages over previous methods: (1) it captures the motion of the hand with all fingers, (2) achieves a framerate of 50 FPS for one hand, (3) has a low latency, (4) achieves high levels of precision, (5) can reliably recover from tracking errors, (6) supports two-handed interaction and (7) enables rapid development of interaction techniques by offering an abstraction (skeleton).

Markerless tracking of finger motion (articulations) for HCI is a challenging problem because of the absence of discriminating image features, rapid motions, self-occlusions, the large number of possible poses and homogeneous colour distribution. At the same time, tracking fingers is essential for enabling free-hand interactions. Previous approaches have avoided this problem by using ad-hoc solutions to directly detect gestures without tracking fingers [161]. The input to our method are RGB images from a calibrated camera video setup, monocular time-of-flight depth data and a hand model adapted to a person. The output are the global pose and joint angles of the hand as a skeleton. The kinematic skeleton provides a means for rapid design of free-hand interactions.

We describe how the tracker from Chapter 3 was developed to allow (1) low latency, (2) high precision, (3) coverage of typical motions in HCI, and (4) two-handed interaction. Results from a technical evaluation show an accuracy of 87% on a dataset of 19 annotated video sequences. Results from a *gesture elicitation* study to confirm the usefulness of FullHand for interaction tasks. FullHand allows users to perform gestures and multi-finger controls that were not possible with previous systems [152].

After presenting the technical contribution, we discuss the design of free-hand interactions using the kinematic skeleton. We build on previous work in 3D interaction and human factors to derive guidelines for free-hand interactions that exploit finger articulations. We designed and implemented free-hand interactions for navigation in 3D scenes, simulation of input devices, mid-air menu techniques and games. The designed interactions explore different capabilities of hand motion including finger articulations (upto 8 fingers) and global hand motion. For example, we demonstrate a mid-air menu selection technique that uses several fingers and terrain level flying with global hand motion (Figure 7.1). To critically assess if such interactions can be tracked and be beneficial for user performance, we conducted a study of virtual globe navigation.

To sum up, the primary contributions of this chapter are:

- An extension to a previous hybrid approach (Chapter 3) for skeleton-based hand tracking for interactive applications.
- The design and implementation of interactive applications demonstrating the use of FullHand for hand and finger motion controls.
- A user study and a gesture elicitation study to validate the proposed approach.

7.2 Related Work

Free-hand tracking for interaction is an old topic dating back to as early as 1979 [14, 18]. Several initial approaches, e.g., for virtual reality or robotics, were based on gloves [172, 137] to ease the problem of hand tracking. However, users may be reluctant to put gloves, especially during long work sessions or when they have to switch with other devices such as the mouse or the keyboard.

Markerless capture of free-hand motion and gestures with non-contact tracking is more challenging. As a result, only a few gesture sets have been proposed and most of the interaction techniques [10, 40, 161] are limited to pinching with one or two hands [161, 40]. This posture can easily be recognized even with RGB cameras but is sensitive to hand orientations and occlusion and does not exploit rich finger coordination.

With the introduction of infrared-based depth sensors like the Kinect, it has become easier and more robust to detect hand gestures. It has been used in large variety of applications such as tabletops [53], distant displays [11], and 3D desktops [75]. For instance, Keskin et al. [62] proposed a method for recognizing finger spelling in depth data. While these methods work well for application-specific hand interactions, they do not generalize and capture the full range of hand motions.

Markerless high DOF free-hand motion tracking for interaction has only recently been explored by Wang et al. [152] for 3D CAD modelling. However, this method uses only part of the hand motion space for interaction (6 DOF). Articulated hand motion tracking continues to be a challenging computer vision problem which has restricted its application in interaction scenarios. Techniques for hand tracking can be divided into *generative* and *discriminative* methods [37].

Generative methods employ a hand model (e.g., kinematic skeleton) and synthesize a *pose* for the model that best explains the input (e.g. [86, 96]). For instance, Oikonomidis et al. [99] used a depth sensor and a method based on particle swarm optimization to achieve a frame rate of 15 fps. Other generative approaches [13, 133, 80] suffer from large computations times and are thus unsuitable for interaction.

Discriminative methods use prior knowledge about hands (e.g., pose database) and try to explain the input images based on this knowledge. One such method that uses a pose database was proposed by Athitsos and Sclaroff [5]. This idea was further explored by Wang and et al. in both color glove-based [153] and markerless variants [152].

Recently, a hybrid method for single hand tracking in a motion capture setting was proposed by Sridhar et al. [131] (see also Chapter 3). This method was able to track one hand at 10 fps which is insufficient for interactive applications. In this chapter, we extend their hybrid method to realtime (50 fps), single and bimanual hand motion tracking. We also demonstrate our method for interaction on a wide range of applications.

7.3 Hand Motion Tracking

In this section, we describe our method for articulated hand tracking that is inspired by the hybrid approach presented in Chapter 3. We chose this particular hybrid approach because it is well suited for interaction applications. The generative component of the hybrid method lends itself for fast optimization which is suitable for interaction but prone to local optima issues leading to wrong hand and finger pose. But when combined with a discriminative component this issue is alleviated leading to better hand and finger pose. We now describe our setup, briefly summarize the hybrid method and explain specific extensions that we have made to enable fast bimanual tracking.

7.3.1 Physical Setup

Figure 7.2 shows the physical setup for hand motion tracking and interaction. It consists of 5 RGB cameras and 1 depth sensor. The image data from RGB cameras provides high visual accuracy for tracking. The complementary single-view depth data helps us to retrieve poses effectively. The setup also consists of a large television screen for interaction and visual feedback. The setup requires calibration of the cameras for intrinsic and extrinsic camera parameters.

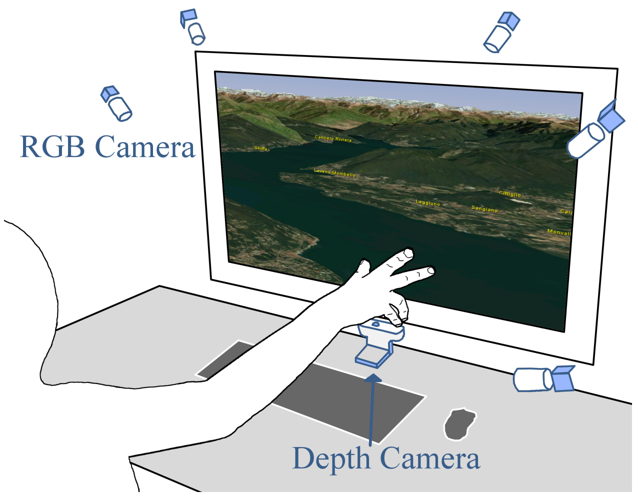


Figure 7.2 Our tabletop setup requires 5 RGB cameras and 1 depth sensor.

While we realize that such a setup is currently cumbersome to setup, we believe that in the future, miniature cameras (see Figure 7.1) and ambient cameras in homes and offices will become widely available. Moreover, as we show in Chapter 5, the number of required cameras can be reduced removing the need for camera calibration completely.

7.3.2 Tracking Algorithm

Markerless optical hand tracking is our approach of choice as it requires no interference with or instrumentation of the hand in any form. However, it is an inherently hard problem because of the large number of DOFs, fast motions, homogeneous skin color distribution and self-occlusions. In the past, numerous approaches for hand tracking have been proposed, which can be roughly classified into generative and discriminative methods. However, both classes of methods in isolation suffer from issues that make them unsuitable for interaction

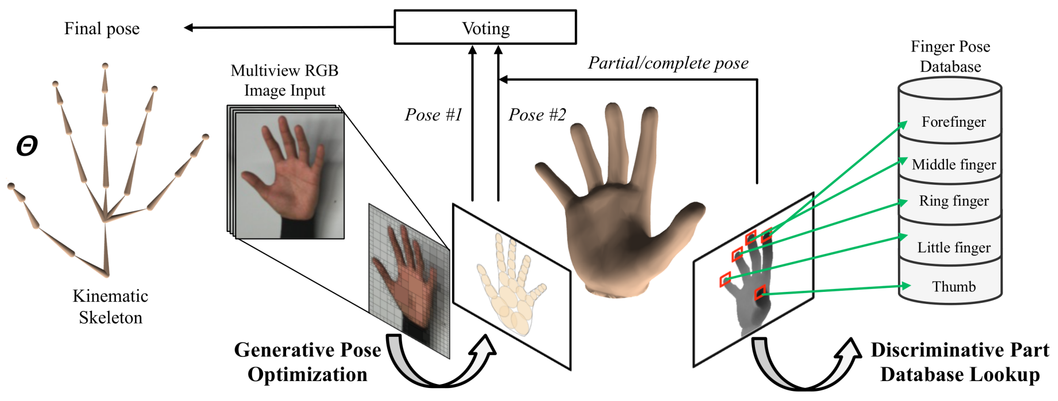


Figure 7.3 The tracking algorithm is a combination of a generative and discriminative method.

tasks. Generative methods optimize a 3D model-to-image consistency measure, E. Fast generative trackers use local optimization of this energy that tends to converge to erroneous local pose optima, e.g. leading to *sticky fingers* – two fingers overlapping each other on the image. Discriminative methods aim to infer hand poses from a learned space of plausible poses by means of extracted features. In this context, many approaches index into the hand pose space, and suffer from scaling problems due to exponential database sizes for high DOF models. In this chapter, we adopt the hybrid approach described in Chapter 3 which combines generative and discriminative tracking, and which exploits their non-congruent failure modes for mutual benefit.

Estimation of the hand pose parameters (see Figure 7.3), Θ, at a time step of video is performed by running two tracking strategies in parallel. The first strategy is a generative tracker that uses multi-view color images, and that relies on a Sum-of-Gaussians scene representation, originally introduced by Stoll et al. [135]. It represents the hand in 3D by a kinematic bone skeleton, to the bones of which a discrete set 3D Gaussian functions are attached. Each Gaussian function is assigned a color, too. Similarly, each 2D image is decomposed into regions of similar color by means of a quad-tree decomposition, and to each region a 2D Gaussian with associated average color is fitted. The hand pose is found by optimizing the overlap between the 3D hand SoG model with all 2D image SoG models. The SoG representation enables the definition of a 3D-2D consistency measure that has analytic derivatives. In addition, the consistency measure can be defined as a smooth function, lends itself to efficient parallelization, and can be effectively optimized with a fast conditioned gradient ascent solver that is initialized with an extrapolated solution from preceding time steps.

The second strategy is a discriminative pose estimation algorithm that uses images from the depth camera. It relies on a part-based strategy that estimates the pose of each finger separately rather than the full pose simultaneously. This is achieved by extracting fingertips on the depth image using a linear support vector machine (SVM) classifier, and by using the detected positions to find the closest match in multiple exemplar finger pose databases. Having separate databases for each finger has several advantages. The part-based strategy enables compartmentalization of the database and effective indexing into a much more densely sampled pose space than with a database storing full hand poses. Further on, with our method even partial hand poses can be found, for instance if some fingers are occluded.

Both tracking strategies yield a pose hypothesis for the hand. The final pose hypothesis is either (1) the solution from generative tracking, or (2) the solution from generative tracking initialized with the outcome of discriminative pose estimation. A final voting step selects the best solution based on the generative consistency measure, E.

7.3.3 Fast Bimanual Tracking in a Tabletop Setting

We have improved the above tracking strategy in several ways to enable fast one and two handed tracking. First, we enable realtime, low latency tracking by exploiting the algorithmic design of the tracking. Second, we enable two handed tracking which captures the articulations of all fingers. Finally, we show that the hybrid method can be optimized to work well in a tabletop setting instead of a controlled studio environment that was used in Chapter 3.

Both the generative and discriminative components of the algorithm lend themselves well for parallelization which we exploit. For the generative method, we use the structure of the consistency measure that allows parallel computation during pose optimization. The discriminative method detects fingertips on the depth image using the *sliding window* technique. We run multiple sliding windows on non-overlapping parts of the image in parallel which leads to lower computations times. Moreover, the two instances of the generative method run in parallel for even more gains. Overall, our average computation times were 3 to 4 times better than those reported in Chapter 3. Figure 7.4 shows a plot of the computation times of the tracker averaged over 3, 1000 frame runs with a user performing slow and fast hand motions. The average time to process one frame was 19 ms (50 fps).

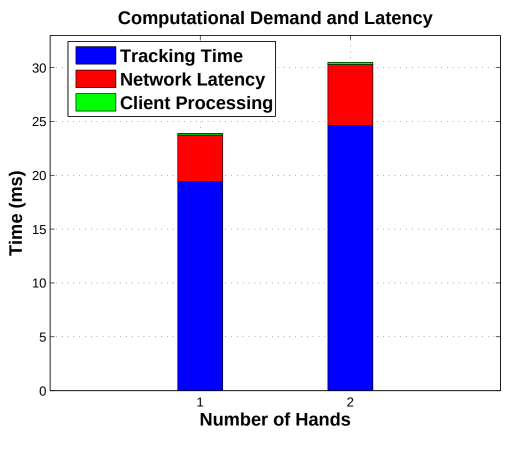


Figure 7.4 Plot of the computation times for one and two hands.

For bimanual tracking, we created a kinematic skeleton for both hands which together consist of 65 joints and 53 DOFs. Since the computation times are proportional to the number of DOFs of the hands and the fingertips to be detected on the depth image, our computational performance reduces to 20-40 fps. However, this is still sufficient for realtime interaction. Figure 7.4 shows a comparison of our computational performance for both single and two hands. For interacting with applications we send the tracked hand (along with gestures which are described later) over the network on a WebSocket protocol. Figure 7.4 shows the network latencies along with the tracking performance.

Finally, we setup our cameras in a tabletop setting (Figure 7.2) to match real world conditions. By tuning the parameters of the Sum-of-Gaussians representation we were able to achieve tracking performance comparable to that reported in Chapter 3. et al. Section 7.4 shows a plot of tracking accuracy for our gesture elicitation study. Figure 7.5 show sample tracking results with one and two hands.

7.4 Gesture Elicitation Study and Accuracy Assessment

In order to understand the kind of gestures that users prefer for interaction and the tracker's capability in covering these, we conducted a gesture elicitation study. In this context we define a gesture to be a semantically meaningful motion of the hand within a given temporal period (e.g. pinching). We chose 6 student volunteers to participate in this study. All participants were right-handed males with a mean age of 29.2 (SD = 5.0). None of the participants had prior experience using or developing free-hand gestures.

7.4.1 Method

We prepared static images of interaction scenarios representative of the four interaction subtasks.

1. Navigation: Participants were presented with images of a virtual globe in both space and terrain viewpoints. They were asked to visualize navigating to cities and flying through buildings and valleys.
2. Selection: Participants were presented with images of a grid menu with 16 items and instructed to simulate selection of three highlighted items.
3. Manipulation: Three primitive objects were shown at random positions on the screen. The participants were instructed to simulate selecting and moving these objects so that they aligned vertically.

4. System Control: Participants were shown images of window switching and photo flipping and were asked to simulate this using hand gestures.

We presented static images instead of video sequences because we found in a pilot study that the interaction technique used in the video (eg. mouse for navigation) biased the kind of gestures that participants elicited. We gave participants 3–5 minutes to think of the gesture that they wanted to perform for each task. They were then asked to orally explain their gesture. Finally, we recorded them performing that gesture using our multi-camera setup.

7.4.2 Results

Participants were allowed to use global hand motion, all finger motion and both hands. When participants repeated the same gesture for two tasks they were asked to perform a different one. In all, we recorded a total of 28 sequences consisting of 22061 multi-view image frames. Table 7.1 shows a classification of the elicited gestures based on the type, number of hands and fingers that participants used for each task. In Table 7.1 we summarize the results of the elicitation study based on the number of hands and fingers that participants used. Users performed gestures that included pointing for navigation, finger waving for the selection, *swipe*-like gesture for manipulation and wrist rotation for system control.

Task	**One Hand**	**Two Hands**	**Avg. No. of Active Fingers**
Navigaton	3	3	1.5
Selection	6	0	2.3
Manipulation	5	1	2.0
System Control	5	1	1.2

Table 7.1 Results from the elicitation study showing number of participants who used one or two hands.

Since we recorded all elicited gestures, we also gained a large multi-view image sequence corpus as a dataset for evaluating the accuracy of hand tracking. While a few datasets exist for measing hand tracking performance, our dataset is specifically of users performing gestures for interaction tasks. In order to show that our tracking method is able to track the gestures elicited, we manually annotated (fingertip and palm locations) the

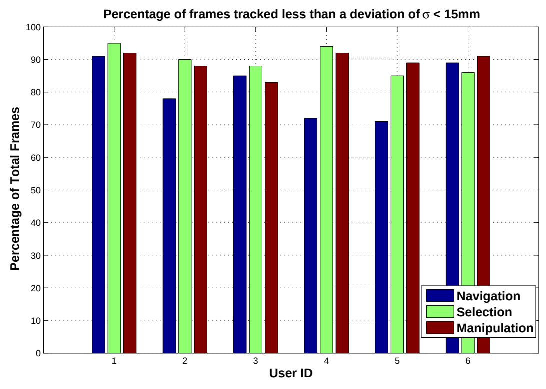


Figure 7.5 Plot of accuracy defined as the percentage of frames with error <15 mm.

elicited gestures for 3 out of the 4 tasks. Because of the large size of our dataset, we subsampled the data by annotating once every 10 frames. We adopted the tracking error of Oikonomidis [99] which measures average fingertip error. We then measured the tracking accuracy to be the percentage of total frames in a sequence that had an error of less than 15 mm. A plot of this measure averaged over all datasets for each user is given in Figure 7.5. We were able to track an average of 86% of the total frames at an accuracy 15 mm or better (after subsampling). The dataset that we have recorded is useful both from the user perspective and the tracking perspective. To our knowledge, such a large dataset with specific free-hand gestures for markerless free-hand tracking is not currently available.

7.5 Designing Free-Hand Interactions

Skeletal representation of hand motion provides a rich and flexible means for designing free-hand interactions. This section outlines the design problem and collects guidelines from previous literature. We then present multiple examples of interactions designed for FullHand using these heuristics and guidelines to demonstrate the capability of hand tracking and the effectiveness of the skeleton-based approach.

The problem in designing free-hand interactions is that the motion space is large and there are multiple ways to map them. Based on previous literature, the design problem can be split into four sub problems: Task Description, Gesture Definition, Gesture Mapping (assigning of a gesture to a task) and parameter optimization.

First, a *task* can be split into multiple sub-tasks. Previous work suggests splitting each sub-task into two- or three-dimensional tasks [61, 92]. Each sub-task, in turn, can address Navigation, Selection, Manipulation, or System control [19]. Second, designers should define the set of *gestures* they want to use. Selection of gesture sets depends on many factors including ergonomic considerations and technical constraints of the gesture recognizer. Third, the designer has to *map* appropriate gestures to a UI control task. Finding the right assignment of gestures to tasks and sub-tasks is not an easy problem. Different users use different kinds of interactions for the same task and one way to find commonality is through elicitation studies such as the one we conducted 7.4.

Finally, once a mapping has been defined, designers need to optimize the technique and choose appropriate transfer functions between hand motion and virtual motion for each sub-task. A small amplitude gesture can trigger a small or large displacement on the screen. This requires user trials and constant improvement by the designer. To further inform design choices, we collected several guidelines from previous literature on hand interaction,

Finger individuation
F1. The principal motions of the digits of the hand are extension/flexion, apposition/opposition of the index and the thumb, and the abduction/adduction of digits [94]
F2. Use index finger and thumb for independent controls [59]
F3. Avoid simultaneous control by middle, ring, and little finger [59]
F4. Allow tremor [59]
Motor control
L1. For higher skill, favor motions that are familiar [59]
L2. Only use the necessary maximum of degrees of freedom [152]
L3. Choose memorable gestures [152]
L4. Directions of motion should be congruous between hand and VE
L5. Performance increases when shoulder muscles can contribute to control [95]
Ergonomics
E1. Avoid hyperextension of fingers
E2. To minimize muscular loading, reduce global motion [152]
E3. Avoid continuous isometric tension of large muscles [23]
E4. Provide a rest for elbow and forearm [152]
E5. Elbow angle should be around 90 degrees [23]
E6. Place the display for comfortable body posture [152]

Table 7.2 Guidelines for free-hand interaction design from previous literature.

human hand functioning, and motor control. Table 7.2 presents several guidelines under these categories.

In the above discussion we have not mentioned the effect of the hand tracking or gesture recognition component in designing interactions. Often, limitations in hand motion tracking or gesture recognition leads designers to come up with gestures that are easier to detect rather than easier for users. In this context, FullHand offers more flexibility because we track the continuous skeleton motion of the hand and detect gestures on the tracked skeleton. In our current work we adopt a heuristics-based approach which is quick to implement and robust enough to enable interactions. For instance, to detect pinch gestures, we use the position of the thumb tip and the fore finger tip as a measure.

7.6 Free-Hand Interaction Applications

In order to demonstrate the capability of the tracker and the skeleton-based approach for interaction, we show applications that (1) span different kinds of tasks (navigation, manipulation, selection and system control) and (2) employ fingers, one hand and bimanual input for interaction. Table 7.3 lists the applications based on the type of control task and the number of hands and fingers involved. We now discuss each in turn.

7.6.1 Navigation + Selection: Space Invaders

Space Invaders, a popular arcade game, combines a one dimensional navigation (maneuvering) and discrete selection (shooting) task. We use a pinch gesture similar to that shown in Figure 7.7 where it is used as a discrete selection event to shoot. To move the spaceship on the screen, we use the raw hand position data we receive from the tracker. Qualitative tests of this interaction technique showed that users were able to successfully complete the game which meant that users destroyed all enemy spaceships.

7.6.2 Two-Handed Interaction: Menu Selection

In this application we show that users are able use both their hands for interacting for a *menu selection* task. We simulate a menu consisting of 8 items and use a pinch gesture recognizer to detect pinching of all fingers with the thumb. Each pinch gesture is a discrete event and is mapped to one item on the menu. The technique demonstrates two-handed interaction for selecting commands without requiring the visual modality.

7.6.3 Emulation of Input Devices: Mouse

FullHand can also be used to emulate existing input devices such as the keyboard or the mouse which capture less DOFs. Virtual input devices have the advantage of reducing the cost for switching from one device to another one. By capturing slightly exaggerated versions of typical hand and finger motions required for e.g., a mouse, FullHand is able to stand-in for that device's functionality. Moreover, FullHand provides more degrees of freedom than existing hand trackers such as the Leap motion, making it possible to emulate this input device.

Interactive Application	Navigation	Selection	Manipulation	System Control	No. of Hands	No. of Fingers
Space Invaders	●	●	○	○	1	2
Menu Selection	○	●	○	○	2	8
Mouse	●	●	●	●	1	2
Virtual Globe	●	○	○	○	1	3
FPS	●	●	◒	○	1	3

● fully covered ◒ partially covered ○ not covered

Table 7.3 Comparison of different applications based on the sub-tasks involved.

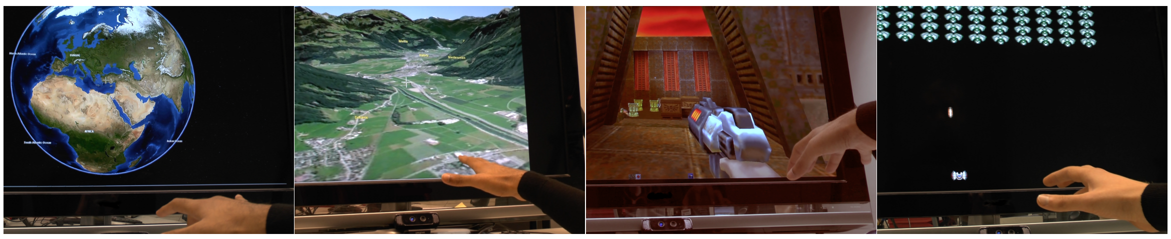


Figure 7.6 Examples of interaction applications made possible by FullHand.

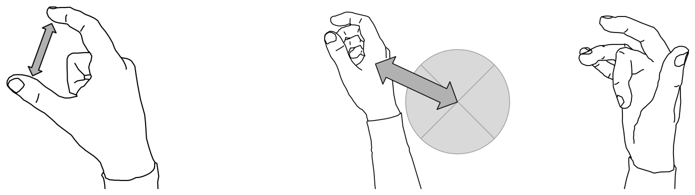


Figure 7.7 Interaction techniques for virtual globe in *space viewpoint*.

7.6.4 3D Navigation: Virtual Globe

Virtual globes, such as Google Earth or NASA WorldWind[1], are an example of a 3D navigation task. They benefit from free-hand control because of the nature of the task involving multiple degrees of freedom. In this example, we used NASA WorldWind and connected it using WebSocket to obtain the raw joint angle parameters and recognized gestures.

We divide virtual globe navigation into two distinct *viewpoints* and propose two techniques to control navigation in each viewpoint. Although they are different techniques, they are compatible with each other.

Space Viewpoint: This mode is active when the camera is 4 km or above the globe's surface. In this viewpoint, there are three parameters that are controllable – the latitude, longitude and altitude. To control *altitude* (zooming) users perform a *pinch gesture* as shown in Figure 7.7. The distance between the thumb and the forefinger on the tracked skeleton defines a rate based control of zooming. A *dead zone* (a region where motion is ignored) of 30 mm centered around the natural arched distance between thumb and forefinger is used when no control is wished. The pinch gesture is one of the principal hand motions and is easy to perform for users.

To control *latitude* and *longitude* (panning), users can choose between two gestures – one that involves clutching and one that does not. The clutch-based gesture uses the flexion angle of the middle finger as a *delimiter* that enables panning relative to the current position of the hand. We observed that this gesture is a good delimiter since it can be moved without affecting the fore finger and the thumb and is seldom performed accidentally by users. For a

[1]http://worldwind.arc.nasa.gov/

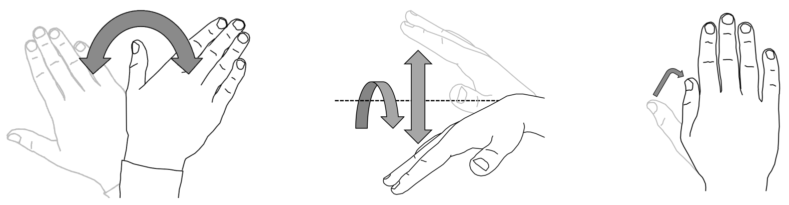


Figure 7.8 Interaction techniques for virtual globe in *terrain viewpoint*.

comfortable flexion angle, a pilot study shows that 45 degrees is a good compromise between robustness and comfort. For the clutchless gesture, the position of the hand on the table relative to a predefined center indicates both the direction of the pan and the speed as shown Figure 7.7. Furthermore, we introduced a circular dead zone of 200 mm diameter which worked well for many users. In designing this interaction technique for the space viewpoint, we followed several of the guidelines introduced earlier in designing this interaction (F1, L1, L3 and E1).

Terrain Viewpoint: This viewpoint is automatically activated below 4 km and has 7 camera parameters that are controllable (pitch, roll, yaw, latitude, longitude, heading and altitude). Figure 7.8 shows the gestures for controlling the camera parameters. The pitch, roll and yaw are controlled by the same metaphor as a *flying vehicle* which is familiar to many users. However, we also allow users to fly forwards and backwards by means of a delimiter which is the flexion of the thumb. This interaction choice was a direct result of a pilot study that we conducted that showed that the *flying vehicle* metaphor was the most natural for users.

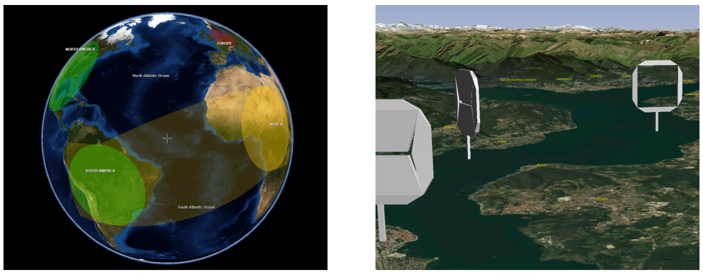
We provide users with a visual cue by means of a smooth camera transition when the 4 km mark is reached. The user can then seamlessly switch from one technique to another. We refer the reader to section 7.7 for a user study conducted using the interaction techniques described here.

7.6.5 Multiple Controls: First-person Shooter

In order to demonstrate that we are able to track more complex tasks that involves navigation and selection in a time-critical environment, we created free-hand interactions for a first-person shooter game. General movement of the character was performed by isometric hand motions similar to the Virtual Globe's space viewpoint. For instance, once the middle finger is clutched moving the hand to the left would cause the character to sidestep to the left. Aiming was performed akin to the Virtual Globe's terrain viewpoint, for shooting a pinching gesture analog to Space Invaders was used. Figure 7.6 shows screenshots from many of the above examples.

7.7 Study of Virtual Globe Navigation

To evaluate the capability of the tracking approach for finger articulations in interaction, we conducted a user study with the virtual globe application. We compared performance in four navigation tasks against the default mouse-based interaction option in WorldWind. The mouse controlled virtual globe navigation through the left, right and middle buttons along with motion. Free-hand interactions are pinching for zooming, hand motion with clutching for panning and palm orientation for orientation as in Figures 7.7 and 7.8.



(a) Continents Task (b) Terrain Task

Figure 7.9 Tasks used in the Virtual Globe study.

We chose the mouse as the baseline, because it provides a hard benchmark. Most computer users have thousands of hours of experience in mouse pointing, including uses for navigation tasks and 3D environments. To our knowledge, this is the first comparative user study using a *markerless* approach for articulated hand tracking.

7.7.1 Method

The participants were six postgraduate student volunteers, all male and right handed, with a mean age of 29.5 years (SD = 4.93 years). All participants confirmed that they use the mouse on a daily basis. The four navigation tasks, illustrated in Figure 7.9, were:

1. Cities: Flying between cities in different continents with city-sized target circles of size 1 km. The route length was in the order of 20000 km. This task was repeated 5 times.
2. Continents: Moving between continents in the space viewpoint where the entire globe is visible. The circle target size was of the order of 1000 km. The route length was of the order of 15000 km. This task was repeated 10 times.
3. Villages: Moving between regional towns. The average route length was 50 km. This task was repeated 10 times.
4. Terrain: Moving along valleys and rivers at the terrain level. The average route length was 150 km. This task was repeated 3 times.

In tasks 1-3, the user had to move the camera viewpoint through a predefined sequence of areas that were highlighted as circles on the globe's surface. Task 4 involved moving the camera through ring-shaped posts at a terrain level where natural formations like mountains and rivers serve as visual assists. The sizes of the areas ranged from continent-sized to about one kilometer radii. A waypoint was considered selected when a crosshair in the center of the display was brought on top of it. Since the users had no previous experience with hand tracking, each task was repeated multiple times with both interfaces. To eliminate order effects, half of the participants performed the tasks with the mouse first, while the other half started with the tracker. The order of Tasks 1-4 was randomized.

7.7.2 Results

The analyzed dataset has altogether 327 trials. For statistical testing, we performed a 4 (Task) × 24 (Interface) repeated measures ANOVA. Figure 7.10 provides an overview of the trends with 95% confidence intervals. Not surprisingly, the effect of task was significant, $F(1, 319)=302.5$, $p<0.001$. We also obtained a significant effect of Interface, $F(1, 319)=11.7$, $p=0.001$. Alas, performance with the mouse was better. However, a closer analysis of the tasks showed that this difference is attributable to Task 1. The interaction effect Task × Interface was significant, $F(3, 319)=7.5$, $p=<0.01$. Figure 7.10 suggests that user performance in Tasks 2-4 was equal with the mouse in the latter half of the repetitions. In contrast, in Task 1, performance with the mouse was always better. A Post Hoc comparison (Bonferroni) against the two showed a statistically significant difference between the mouse and free-hand interactions only for Task 1 ($p<0.001$).

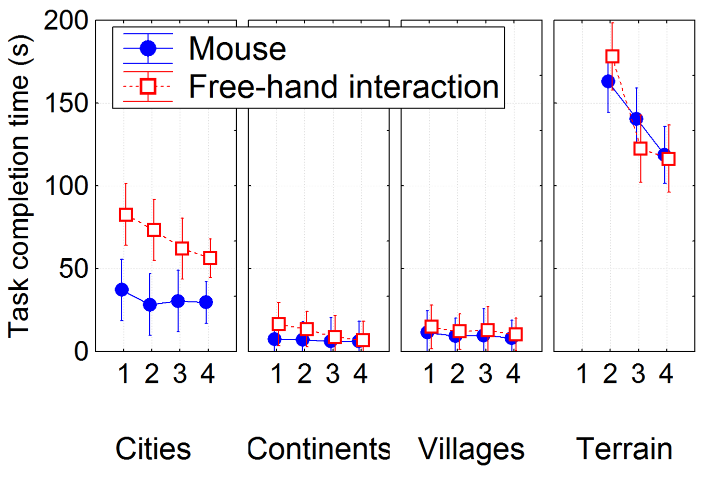


Figure 7.10 Development of task performance for mouse vs. free-hand interactions in four navigation tasks with the Virtual Globe. Vertical bars denote 95% confidence intervals.

To sum up, parallel performance was achieved for 3 out of 4 tasks. Given the small number of trials and the lack of previous experience with hand tracking, we consider this result promising. Furthermore, we learned that the poor performance with the tracker in Task 1 is due to hand tremor caused by the absense of an arm rest.

7.8 Discussion

FullHand extends the method presented in Chapter 3 to track hand articulation, and especially finger articulation, for interactive applications. It follows a hybrid approach and uses a multi-camera setup to track the skeletal motion of 26 degrees of freedom with a low latency. Whereas previous trackers have shown point designs without critical evaluation, we subjected the method to both technical and empirical assessments. Results from a motion elicitation study suggest that combining finger articulation with global hand motion is natural to users. The hand tracking algorithm had an error of <15 mm in 87% of the datasets that we collected. A broad range of interactive techniques were designed to further explore this capability. Our examples range from menu selection that uses multiple finger motion of two hands to first-person shooter where 3 fingers and global hand motion are simultaneously used for playing.

We developed one of the interaction techniques further to be used in a real application, a 3D virtual globe. Results from a controlled user study show that although interaction was difficult at first, users' performance in three out of four tasks rapidly developed to a level comparable with the mouse. Although the study has a limited sample size, it demonstrates that the capability of the tracker can be actually used for free-hand interactions. To our knowledge, it is the first controlled study of interactive applications of markerless hand articulation tracking that report objective measures of user performance.

Previous markerless free-hand interaction technologies imposed constraints on designers regarding the type of interactions that they could create due to technical limitations. Since we track a kinematic skeleton new interaction techniques can quickly and efficiently be detected and used for interaction. We regard these results favorable to the idea of using the hybrid tracking approach presented in Chapter 3 for HCI.

7.9 Conclusion

Presently, our hand model creation process is semi-automatic, and we plan to improve this by adopting automatic methods for hand shape estimation. The discriminative component of our method would also fail when multiple hands are present since this would affect fingertip detection. Finally, we also require users to wear a black sock for image segmentation purposes. The approach presented in Chapter 5 could be used to overcome these limitations. The problem of *gorilla arm*, i.e., arm fatigue due to extended gesture use, also needs to be investigated further.

In this chapter, we have showed continuous gesture input with a hand tracker driven by elicitation studies. Eliciation studies, however, can suffer from small sample sizes. The designer will need to perform numerous iterations before finding gestures that are most suitable. In the next chapter, we show how a computational, optimization-driven approach can be used for this purpose.

Chapter 8

Computational Gesture Design

In the previous chapter, we have seen how elicitation studies (i.e., eliciting gestures from users) can enable us to create continuous gesture-input for 3D navigation applications. Elication studies, however, can be hard to generalize, can be affected by sample size limitations, be hard to implement in practice, and are time consuming. In order to structure this problem and overcome the limitations of elicitation studies we discuss **computational gesture design** in this chapter. Computational gesture design refers to the process of automatically designing gestures for an interaction task to suit designer-specified criteria. This has the potential to find optimal gestures from the huge gesture space. In our approach, we build a model of hand movement that allows formulating gesture design as an optimization task. We base our model on the investigation of hand dexterity: i.e., how fast and accurate fingers can move, how individuated can they move, and what are their comfortable movement ranges. While we show how to use our model to design gestures for a discrete input task (text entry) our approach can also be used for continuous gesture design. Parts of the work presented in this chapter appeared previously in [127].

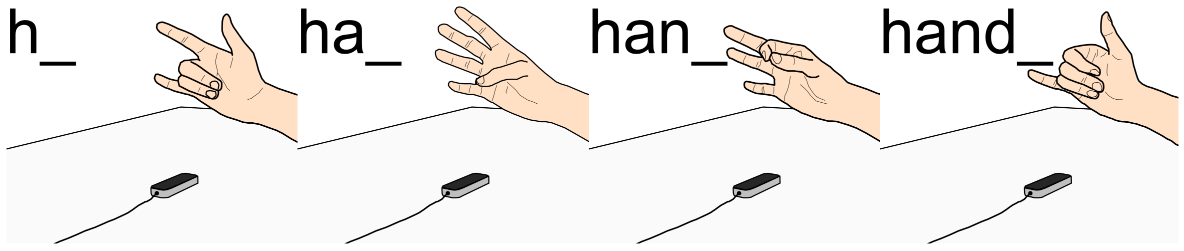


Figure 8.1 We investigate the dexterity of using multiple fingers for mid-air input. This chapter reports performance and individuation characteristics of fingers and deploys them to the design of a mid-air text entry method using multi-objective optimization. Here we show an example of the word ‘hand’ being typed using one of our automatically obtained designs.

8.1 Introduction

This chapter investigates an emerging category of input enabled by progress in computer vision-based hand tracking: *input by free motion of the hand involving any and all fingers*. Until recently, computer vision-based input was limited to gross movements of the arm and a few basic hand poses like pinching [10, 161]. However, methods to track full hand articulation using a single depth camera are now available such as the one presented in Chapter 5 (see also [86, 108]). Leveraging the hand's capacity "directly" without intermediary devices like joysticks or buttons has always appealed to HCI researchers. With its many degrees of freedom, and fast and precise movements, the hand is the most dexterous of the extremities [59, 83]. Furthermore, freehand motion could provide an always-on input method, as only a camera is required. The method could alleviate the known input limitations of wearable or mobile devices.

Our goal is to inform the design of *high performance input* using multiple fingers in mid-air. High performance is decisive in activities like text entry, virtual reality, command selection, and gaming. However, previous work, such as that presented in Chapter 7, has focused on eliciting intuitive multi-finger gestures from users (see also [89, 104]). This leaves out many issues, including performance characteristics of gestures involving single and multiple fingers simultaneously. To push the field forward, designers need to know some key factors affecting performance: How fast can users move their fingers? Can all fingers be moved independently and accurately? What are their movement ranges? How to combine fingers with different properties in one gesture?

Our work focuses on chord-like motions in mid-air as shown in Figure 8.1. These are easy-to-perform and familiar gestures, and among the few gesture categories that current computer vision sensors can reliably track. In this input gesture, there is no external target like a button (cf. most previous work on mid-air text entry [4, 84, 93, 120]). The involved fingers are extended or flexed at a single joint to a discriminable end posture. Although this input method *can* be used with visual feedback, it allows for eyes-free input after memorization.

We extensively study the dexterity of single fingers in a target selection task. Users were asked to move a finger quickly and accurately between two *angular targets* (e.g., from a neutral resting position to the maximum position "down"). We assess each finger separately to report on three critical factors:

- **Speed and accuracy** of angular motions of fingers measured by Fitts' law models [81].

- **Individuation** of fingers, as measured by the so-called Schieber index [117]. It captures the extent to which non-instructed fingers remain still when a finger is moved.
- **Comfortable motion ranges** of fingers reported by users.

The results afford several insights. First, we report performance characteristics of each finger. The data show differences of up to 50% in movement times. Second, we asked users to move fingers *comfortably* and report on their motion ranges when using computer vision tracking. Third, to our knowledge, this is the first work to report individuation indices for joints in HCI. For the middle and ring finger, coactivation can be so high that input may be compromised by false activations. In contrast, coactivation of other fingers while moving the thumb is virtually non-existent. We argue that individuation is a critical consideration in multi-finger input in mid-air which lacks physical resistance.

Our second contribution is to propose how to use this data in the design of high-throughput gesture sets. While our study considered only single joints, we attempt to apply our findings in the design of *multi-finger* input. The approach builds on literature in motor learning and assumes that multi-finger performance is limited by the *slowest* joint [60, 116]. Moreover, we exploit the fact that individuation constraints do not apply if co-dependent fingers participate together in a gesture. The benefit of these two assumptions is that the derivation of models to inform hand gestures is significantly less expensive than a study that tried to look at *all* combinations of fingers. Even with only three discretization levels per joint such an approach would have to cover roughly 10^{10} gestures. Finally, we use our findings to construct a proof-of-concept objective function called PALM to optimize text entry in mid-air. PALM considers performance (P), anatomical comfort (A: i.e., individuation), learnability (L), and mnemonics (M) to optimize multi-finger gestures. First investigations of a text entry method optimized for one-handed input show entry rates of 22 WPM. However, we note that users' performance was limited by brief training times, individuation constraints, and relatively limited performance of the tracker.

To summarize, this chapter informs the computational design of high-performance input methods in mid-air by

1. providing ready-to-use models and look-up tables on performance, individuation and movement ranges of fingers, and
2. showing the applicability of the results by proposing an extension to multi-joint gestures and exploring its use in the multi-objective optimization of mid-air text entry methods.

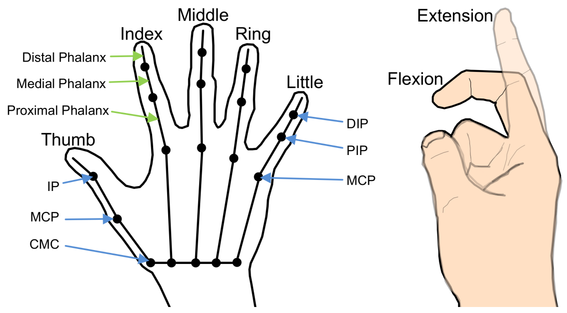
8.2 Background: Characteristics of Finger Motion

Our investigation of multi-finger input is informed by hand anatomy, the degrees of freedom of its joints, the performance of finger motion, and the limitations posed by dependencies on finger movement.

8.2.1 The Kinematic Skeleton

The skeleton of the human hand has 27 bones, the interfaces of which form the wrist and finger joints [59, 116] (see Chapter 1 and Figure 8.2a).

Together, this results in more than 25 degrees of freedom (DOFs) for the hand. In this chapter, we focus on a subset of these DOFs. As humans we can describe hand gestures with terms like *thumbs up* or *v sign*. However, a formal representation is needed for study and use in computer vision-based input. We use a *kinematic skeleton* [90] to parametrize gestures. The hand skeleton configuration $\boldsymbol{\Theta}$ can be specified by angles of the joints connecting the bones, i.e., $\boldsymbol{\Theta} = [\theta_1, \theta_2, \ldots, \theta_i]^{\mathrm{T}}$, $\theta \in \mathcal{R}$.



(a) **Left**: Aspects of human hand anatomy with bones (green) and joints (blue). **Right**: We focus on *flexion-extension* of the five fingers.

8.2.2 Movement Performance

Finger movement performance can be quantified by movement time MT which is the time it takes for an end-effector to reach a target from a given distance. Fitts' law has been highly successful for predicting MT with traditional input devices [81]. It estimates the upper bound of pointing performance achievable after practice. Given a target of width W and distance D, Fitts' law states that the MT to reach the target is given by $MT = a + b\log_2(D/W+1)$, where the free variables a and b need to be estimated experimentally. Fitts' law has also been used previously to quantify performance differences in fingers, wrist, and forearm [12, 31, 74, 82, 109]. However, in this work, we use *angular* motions at joints instead of translation [67]. Considering the angular target width α_W and distance β_D, we get:

$$mt_\theta = a_\theta + b_\theta \log_2\left(\frac{\alpha_D}{\beta_W} + 1\right). \tag{8.1}$$

To acquire the a and b parameters, we conduct an experiment that employs a unidimensional pointing task. We address speed–accuracy trade-off in this task by using *effective* width $\bar{W}$ and distance $\bar{D}$ (see [159] for details).

8.2.3 Inter-Finger Dependencies

Movements of the hand act over multiple joints which makes coactivation of non-contributing joints common [59]. For example, many people cannot move their ring finger without coactivated movement of the little finger. More generally, coactivation is known to be larger among the metacarpophalangeal and the proximal interphalangeal joints [59, 117]. Hand gestures should minimize the extent of *unintended coactivation* of non-instructed fingers. Coactivations can be hard to inhibit and can cause recognition errors.

Schieber [117] proposed an *index of individuation* that indicates how independently an instructed finger can be moved from all others. The index was modeled for monkeys and humans [45]. A fully independent finger does not involve coactivation of other fingers during its activation, or vice versa. The individuation index is widely known in neuroscience, but largely disregarded in HCI. In order to compute it for every finger, the position of the non-instructed digit is plotted as a function of the instructed digit's position. The resulting trajectories are typically linear and the slope of a line fitted to these data points serves as a measure for the *relative coactivation*: the extent to which a non-instructed finger moves relative to the instructed finger. Given the coactivation C_{ij} of finger i during the movement of finger j, the individuation index of j is

$$I_j = 1 - [(\sum_{i=1}^{n} \mid C_{ij} \mid -1)/(n-1)], \tag{8.2}$$

where $n = 5$ is the number of fingers. $I_j = 1$ indicates perfectly individuated movement, and $I_j = 0$ if all non-instructed fingers move simultaneously with j. The original study of individuation was reported for fingers, but it can be extended to multiple *joints* used in multi-finger input.

8.3 Experiment: Finger Dexterity

The goal of this experiment is to quantify the components of finger dexterity i.e., speed and accuracy of finger movements, finger individuation, and comfortable movement ranges. To achieve this goal we setup an experiment to gather data for all three components simultaneously. Our experimental method is based on the reciprocal selection task used in Fitts' law studies [81]. As shown in Figure 8.3, users move a finger between two targets. Instead of extrinsic targets (e.g., buttons), the target here is a joint angle. Visual feedback is provided on a monitor with high refresh rate. In contrast to most Fitts' law studies, we track not only the endpoints of movements but the full motion of the hand. This allows us to quantify three aspects of the dexterity of finger motion: performance (speed and accuracy), individuation (unwanted motion of non-instructed fingers), and comfortable motion ranges. In addition, the data allow us to look at the range of individual differences.

We chose to focus on six joints spanning seven degrees of freedom (see Figure 8.3). This selection is motivated by the capabilities of present-day trackers and our pursuit of studying joints that could be a "class" of input motions. We conducted a pilot study of the Leap Motion sensor[1] and learned that individuated motions of interphalangeal joints are not well tracked, except for the thumb. (The work in this chapter preceded the development of the tracker described in Chapter 5, and thus we were restricted to using the Leap Motion.) Therefore, we decided to focus on the flexion/extension of the MCP joints of the fingers and the CMC joint of the thumb, which intuitively correspond to "up" and "down" movements when the hand is in a neutral pose. Moreover, we included the IP joint of the thumb which was the only interphalangeal joint that could be moved *and* tracked well. Figure 8.3 also shows our naming convention and color coding used in the rest of the chapter. For the thumb we use Thumb-Down and Thumb-Right to denote "up-down" and "left-right" movement of the CMC joint.

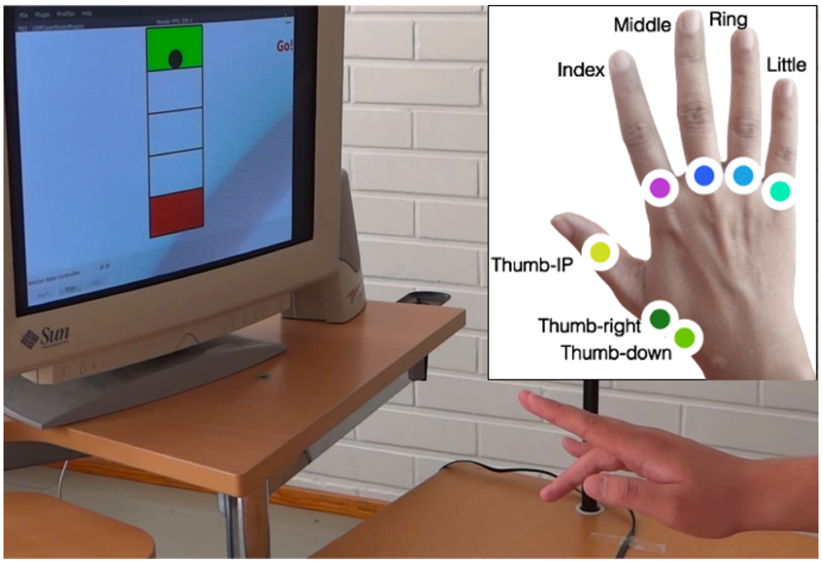


Figure 8.3 The experiment investigates the dexterity of six joints that can be reliably tracked with the Leap Motion sensor. The user is asked to move a finger between two target angles indicated on a display. Full hand motion was tracked. The color coding for joints is used in the Results section. Note that the CMC joint of the thumb is a special case, as it can be independently moved in two directions.

[1]https://www.leapmotion.com/

8.3.1 Participants

The study was conducted with 13 participants (8 male and 5 female) at two different locations. All participants were right-handed and had an age ranging from 22 to 32 (mean 27). Due to technical issues, one of the participants completed only 4 of the 7 joint conditions. The experiment took 1.5–2 hours per participant. Participants from one location were compensated with cinema vouchers. The trials were carried out under controlled lighting conditions with no distractions.

8.3.2 Experimental Design

The experiment followed a 7$\times$4 within-subjects design with 7 DOFs and 4 index of difficulty (ID) conditions. To minimize order effects, the DOFs and ID conditions were randomized for each participant. Pre-trial practice was employed and breaks were provided after the trial for each joint.

8.3.3 Task, Materials, and Procedure

The task was a unidimensional target selection task. Participants had to move a pointer up and down between two targets on a screen and were instructed to move as fast and accurately as possible without moving non-instructed fingers too much. Control occurred by angular motions of joints that were linearly mapped to a pointer on the display. A trial would start from a comfortable neutral pose. The target region turned green when the pointer reached it and the user had to change direction to select the previous target again. In each condition, users had to perform 50 repetitions. Auditory feedback was given in the form of a low-frequency click. Throughout, participants placed their hand in a horizontal position over the sensor with their arm resting on a support.

Because of anatomical differences, we determined the movement range of each user experimentally, and used it to determine concrete target widths and distances for each user. Therefore, we first recorded the user-specific angular limits of each joint at the beginning of each task. We asked the participants to flex and extend the joint without moving the other fingers too much. The corresponding movement range was then uniformly divided into 2, 3, 4, and 5 bins. This gave us the same four unique IDs for every user: 1, 1.6, 2 and 2.3. Over all discretization levels there were 10 different target pairs for each joint, resulting in 7 $\times$ 10 = 70 conditions.

8.3.4 Apparatus

The joint angles were tracked using the Leap Motion by transforming its output to a kinematic skeleton. The software for tracking and display of the task ran on a fast desktop computer (3.1 GHz Intel i7 at one place, 3.1 GHz Intel i5 at the other). We showed visual feedback on high refresh rate monitors (112 Hz CRT and 120 Hz LCD respectively) and the Leap Motion was capable of tracking at up to 100 Hz.

8.3.5 Analysis

Performance: The design and evaluation of the Fitts' law task was done according to [125]. Movements with a movement time or distance beyond 3 SD of the median were excluded. Accuracy was adjusted to allow an error rate of 5%, a rate common in high-performance tasks such as text entry. Based on the remaining movements, we determined the *effective* target width $\bar{W}_{5\%}$ and distance $\bar{D}_{5\%}$ which was used to compute the effective index of difficulty (ID_e) of each task: $ID_e = log_2(\frac{\bar{W}_{5\%}}{\bar{D}_{5\%}} + 1)$. This indicates the actual difficulty of the performed task and captures the speed–accuracy trade-off. To account for individual differences, we cluster the effective IDs into 5 equally sized bins and compute the average movement time within each bin. For this purpose, we excluded data points with an effective ID of 3 SD beyond the median. Least-squares linear regression was then used to determine the slope and intercept of the Fitts' law model.

Individuation: We followed the protocol described in [117] to determine individuation indices. We first plotted, separately for each user, the normalized angle of every *non-instructed* joint as a function of the normalized angle of an instructed joint. The resulting 500 trajectories were then averaged by taking the median. Outliers beyond 3 SD of the median were excluded. The slopes of the resulting data were determined by least-squares linear regression. While linear movement trajectories were the norm, there were a few outliers where a linear relationship could not be determined. We observed two reasons: (1) Problems in tracking the joint angle (Figure 8.5 (b)) and (2) drifting of fingers, a phenomenon in which the non-instructed joint gradually changes its angle due to fatigue, inattention, or corrective behavior (Figure 8.5 (c)). To account for this, we excluded models with a fit (coefficient of determination) of $R^2 < 0.5$. As suggested by Schieber, we averaged the *absolute* value for each slope, to generalize the relative individuation over all participants. These values were then used to compute the individuation index. In the next section, we report findings for performance, individuation, and movement ranges.

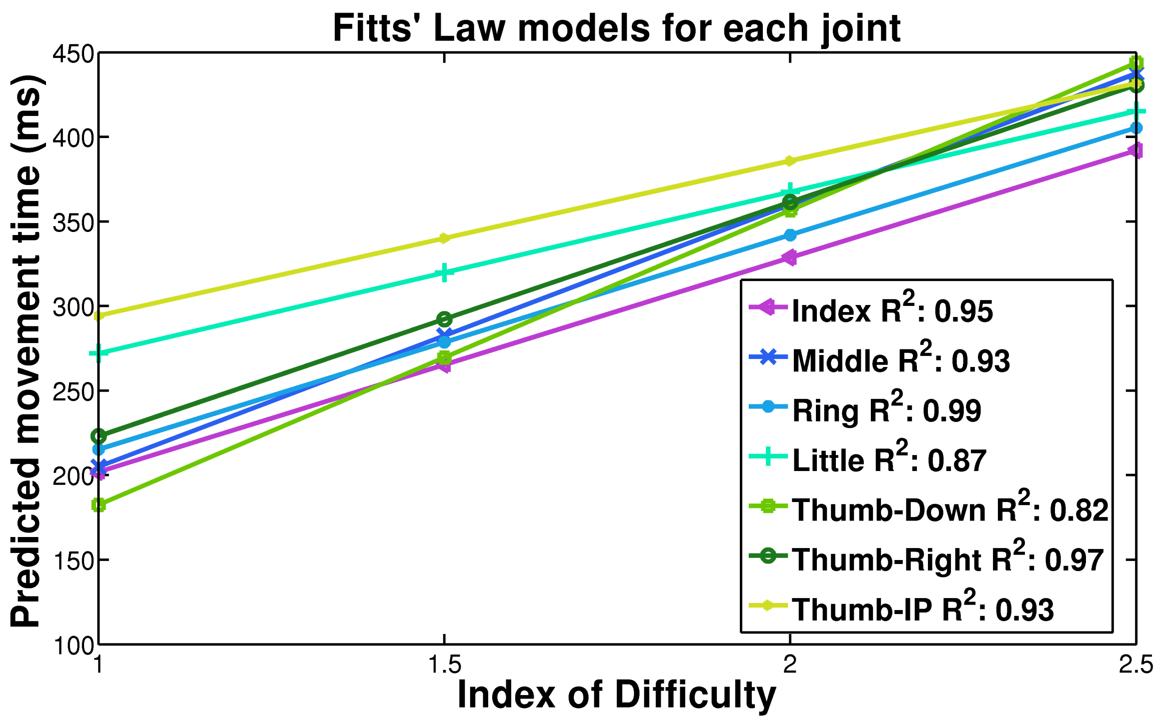


Figure 8.4 Performance models for each joint as given by Fitts' law. Overall, Index is the fastest, while Thumb and Little finger are the slowest.

8.4 Results

8.4.1 Performance: Fitts' Law Models

Fitts' law models and fitness scores for the joints are given in Table 8.1. The R^2 values range from high (0.82) to excellent (0.99). One-way repeated measures ANOVA showed statistically significant differences among the joints for MT: $F(6, 60) = 3.3$, $p < 0.05$. Overall, Index had the highest performance, while Thumb-IP was the worst.

Joint	Intercept *a*	Slope *b*	R^2
Index	75.140	126.77	0.95
Middle	49.940	155.03	0.93
Ring	88.450	126.79	0.99
Little	176.52	95.510	0.87
Thumb-Down	8.1900	174.26	0.82
Thumb-Right	84.590	138.44	0.97
Thumb-IP	202.73	91.590	0.93

Table 8.1 Fitts' Law models for each joint, given by intercept and slope.

More subtle differences can be observed by looking at the cross-over points of the slopes in Figure 8.4. The Index finger was the fastest for most part of the ID range. However, for small IDs, corresponding to large neighboring targets, Thumb-Down outperformed Index.

We also observe that for small IDs, MTs are spread for the different fingers

Instructed Joint	Index of Individuation	Relative Coactivation						
		Index	Middle	Ring	Little	Thumb-Down	Thumb-Right	Thumb-IP
Index	**0.819**	1	0.24	0.20	0.19	0.29	0.11	0.06
Middle	**0.817**	0.16	1	0.41	0.14	0.20	0.11	0.07
Ring	**0.808**	0.16	0.20	1	0.36	0.15	0.22	0.06
Little	**0.806**	0.18	0.35	0.29	1	0.14	0.12	0.08
Thumb-Down	**0.792**	0.12	0.12	0.10	0.08	1	0.69	0.14
Thumb-Right	**0.853**	0.07	0.09	0.10	0.09	0.27	1	0.26
Thumb-IP	**0.889**	0.11	0.13	0.11	0.09	0.12	0.12	1

Table 8.2 Individuation index and relative coactivation describe the involuntary motion of joints. The individuation index is an aggregate that describes the independence of a finger when averaged over all other fingers (1 = perfect individuation). Relative coactivation denotes the movement of a non-instructed joint when the instructed joint (each row) is moving. A value of 1 denotes that the two joints always move together.

Joint	Min° (SD)	Max° (SD)	Range (SD)
Index	48.39 (12.25)	−21.19 (8.70)	69.58 (11.81)
Middle	37.58 (11.95)	−18.69 (8.02)	56.27 (12.54)
Ring	44.66 (8.320)	−12.24 (7.70)	58.90 (11.46)
Little	39.47 (15.78)	−20.81 (8.64)	60.28 (14.89)
Thumb-Down	27.31 (1.680)	−6.280 (6.54)	33.58 (7.130)
Thumb-Right	22.18 (10.53)	−11.99 (8.43)	31.32 (12.59)
Thumb-IP	62.97 (12.94)	−27.41 (4.37)	90.38 (13.93)

Table 8.3 Angular limits and movement range of each joint. The table shows values averaged over all users together with standard deviations.

(difference of 112 ms, $ID = 1$) while they become more condensed for larger IDs (51 ms, $ID = 2.5$). In other words, there is more variation for "easy" movements. Significant individual differences could be observed. Differences in MT for the same joint were as large as 418 ms. The top performance was 91 ms for $ID = 1$, while the worst user performed at a speed of 509 ms per movement $ID = 1$.

8.4.2 Individuation: Schieber Indices

Table 8.2 provides an overview of the findings. We report aggregate indices per finger and by finger-pair coactivation.

Individuation Index: The individuation index for each finger can be found in the second column of Table 8.2. The values range from 1 for perfect individuation to 0 for perfect coactivation. Thumb-IP was found to be the most individuated joint, while Thumb-Down seemed to be the one with the highest coactivation. The individuation indices of the MCP joints showed only marginal differences.

Relative Coactivation: While the individuation index provides an elegant way to summarize the independence of each finger, greater insight is provided by the *relative coactivation* of joints, which denotes the movement of an non-instructed finger when the instructed finger is moved. In Table 8.2, we present the relative coactivation averaged over all users. It ranges from 0 to 1, where 1 is perfect coactivation, i.e., the non-instructed finger moves exactly along with the instructed finger. Note that the value range is the opposite to the individuation index, where 1 is better. We observe that Thumb-Down is closely correlated with Thumb-Right, explaining why it has the lowest individuation index. This indicates that the two DOFs of the thumb's CMC joint cannot be reliably distinguished and should be combined when implementing thumb movements for gestural input. Particularly high values were also observed for the movement of Ring during instructed movement of Middle, and the other way around (Figure 8.6). Thumb-IP shows low values throughout all joints which explains the good individuation index.

8.4.3 Comfortable Movement Ranges

The average angular limits and movement range for each joint are given in Table 8.3. The values represent joint limits that are comfortable for the user in this setting and reachable without moving the other joints too much. One-way repeated measures ANOVA (subjects with missing data excluded) showed statistically significant differences between movement ranges: $F(6,60) = 39.19$, $p < 0.0001$. We observe that the CMC joint of the thumb has the smallest movement range in both movement directions (34°and 31°). The range of the MCP joints is twice that, and Index has the largest range (70°). Thumb-IP has overall the largest movement range with an average of 90°.

8.4.4 Observations on Individual Differences

Large differences among users were observed. Some users were able to keep their non-instructed finger nearly static (slope close to 0), while others moved them to a large extent along with the instructed joint (slope $= 0.4$). Figure 8.7 shows the coactivation of Index relative to Middle. Movement strategies vary too, resulting in a positive slope (moving along with the instructed joint) or even a negative slope (moving opposite to the instructed joint). If a joint could not be kept static, users either moved it along with the instructed joint or opposite to it. Attempts at "counteracting" movement like this were also observed in the original work by Schieber [117]. It may represent a strategy for preventing non-instructed fingers from moving along instructed digits. This suggests that these strategies are applied unconsciously.

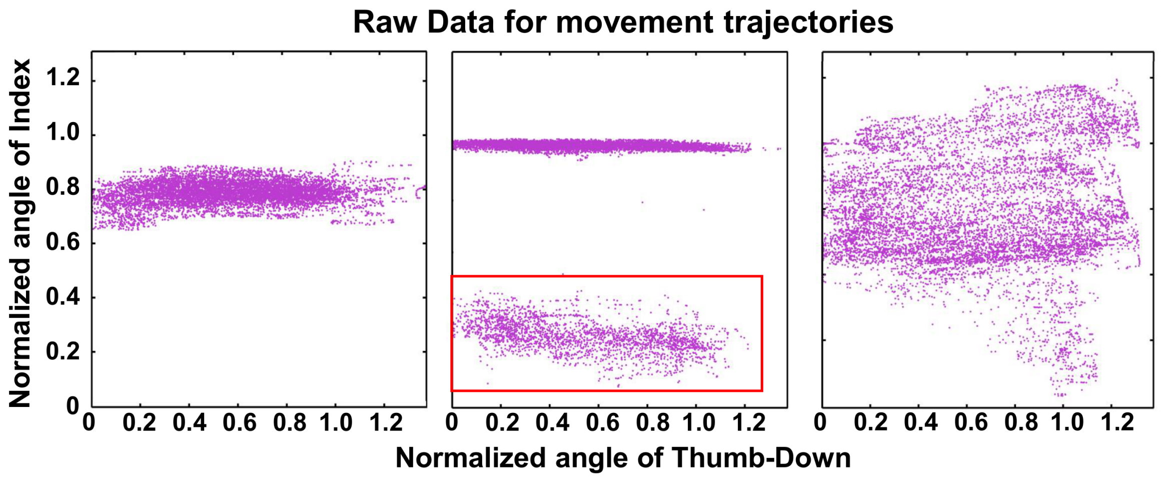


Figure 8.5 Raw data for movement of Index relative to instructed movement of Thumb-Down. Left (a): Example of high individuation, Middle (b): Tracking errors (red box), and Right (c): "drifting finger".

We also observed what we denote as the *drifting finger effect*: the position of non-instructed fingers may change gradually over time for some users, as they "forget" to keep the finger still. For some users, this poses no problem, they are able to produce the exact same movement over and over (Figure 8.5 (a)). We show raw data of this "drifting finger" problem in Figure 8.5 (c). Due to user-specific differences like this, the linear model of Schieber does not always fit to a user's motion. On average, an R^2 of 0.77 (SD 0.14) was found, ranging from 0.5 to excellent fits of 0.99. As discussed above, we excluded the data where no sufficient linear relationship could be found. On average, this amounted to excluding data from 4 users per joint-joint condition.

Finally, despite our efforts to ensure the ergonomics of the posture and to provide enough breaks, some users complained about fatigue, especially with their wrist or arm getting tired. This suggests that these motions are tiring even if they do not require the use of large forces.

8.5 Application to Text Entry

The results of the study offer a nuanced picture of the two characteristics of finger motions. The performance and independence of fingers differ and are inter-connected in subtle ways. In this section, we present a proof-of-concept that shows how to use the results to design multi-finger gestures for a high-performance input task. We chose to focus on text entry by mapping *static mid-air hand postures* to letters. We use the terms 'gesture' and 'posture' interchangeably in this section to denote static postures. Mid-air input is a promising

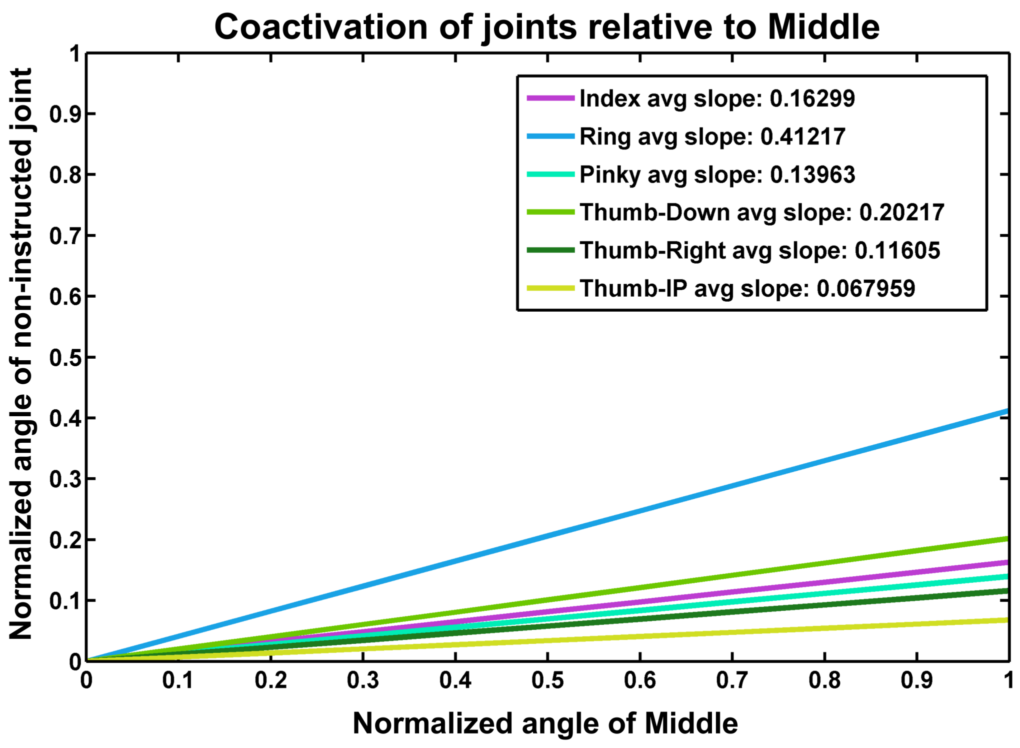


Figure 8.6 Average coactivation of all joints relative to the instructed movement of the middle finger. The slopes are the average of the absolute values over all users.

input modality for emerging devices like smartwatches and heads-up displays [84]. In contrast to previous mid-air text entry methods which used *extrinsic* key targets or handwriting gestures [4, 84, 93, 120], we focus on chord-like gestures controlled by angular motions. Although more complex than single finger input, it has been shown that a large number of chords can be memorized [118] and used for text entry (e.g., [36, 79]), as well as on multi-touch displays [9].

Since the space of possible posture-letter mappings is (exponentially) large, we follow an optimization approach (e.g., [34, 170]). We outline a novel objective function called PALM that can be used to optimize mappings for four objectives. In addition to performance and individuation constraints, it considers learnability and mnemonics. The outcomes can be used to enter text with any hand tracker and gesture recognizer. Our approach has four main steps, which serve as a roadmap for designing tasks other than text entry: (1) Discretizing Joint Angles, (2) Generalizing to Multi-Joint Gestures, (3) Formulating an Objective Function, and (4) Optimization.

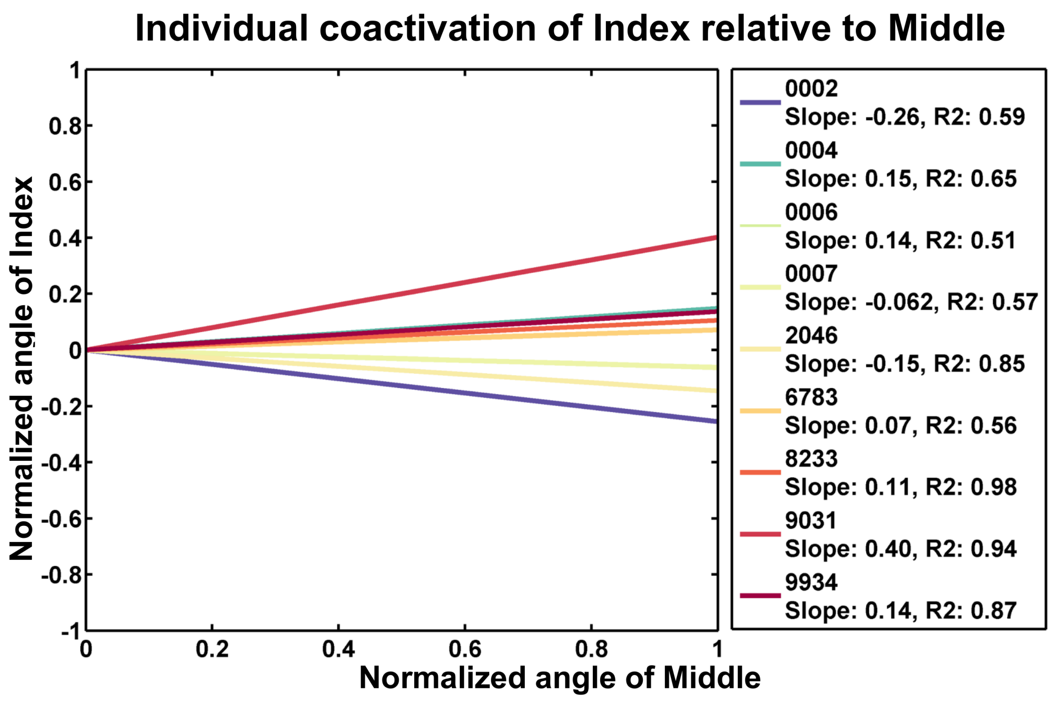


Figure 8.7 Differences among users (denoted by four digit user ID) in the movement of the index finger relative to the middle finger. A positive slope indicates that it follows the instructed joint, negative slope that it moves in the opposite direction.

8.5.1 Step 1: Discretizing Joint Angles

We first need to select the number of discretization levels of angular motion that each joint can afford. This is determined by the robustness of the hand tracker and by performance data we obtained. Our estimate for angular discretization when using the Leap Motion is between 2 and 5 levels per joint angle. For each joint, an integer from 0–k is used to represent the current joint angle, where k is the highest level. Thus, the posture of the hand can be compactly represented using a string of numbers which we call a *bin address*. For instance, the posture corresponding to the letter ‘h’ in Figure 8.1 can be denoted by the string [0,0,1,1,0] (using 5 joints). We also define a neutral pose for the hand, which is a comfortable position, and calibrate such that it corresponds to the bin address [0,0,0,0,0].

8.5.2 Step 2: Generalizing to Multi-Joint Gestures

Since the findings from our study are for single joints, we make two assumptions to generalize to multi-joint gestures. First, to estimate movement time (MT) for gestures involving multiple joints, we assume that it is bounded by the performance of the slowest contributing joint. We base this on evidence that movement of arm joints are timed so that all joints reach their final positions simultaneously [60, 116]. Thus, we estimate the time for a multi-joint gesture as the maximum over each of the MTs of all joints involved. Formally, we define time for moving from one posture to another as,

$$MT = \max\{mt_{\theta_i}\}, \theta_i \in \Theta, \tag{8.3}$$

where mt_θ corresponds to the movement time of one joint as given in Equation 8.1.

Second, to estimate individuation constraints of a multi-finger gesture, we extend the individuation index of Schieber to take into account the fact that coactivation between fingers is not an issue when those fingers are used in the same gesture. The middle finger, for example, has a poor individuation index, which is mainly dominated by the relative coactivation of the ring finger. A gesture involving both fingers can therefore be performed with higher individuation than a gesture involving only one of the fingers. To this end, we define the coactivation C_{iG} of a joint i relative to a gesture (or posture) G as the maximal coactivation of i relative to any joint j involved in the gesture: $C_{iG} = \max_{j \in G} C_{ij}$. Then, following the original Equation 8.2, we compute the individuation index for any multi-joint gesture as

$$I_G = 1 - [(\sum_{i=1}^{n} |C_{iG}| - |G|)/n - |G|], \tag{8.4}$$

where $|G|$ denotes the number of actively involved joints, and n is the total number of joints.

8.5.3 Step 3: Objective Function Formulation

Our design task is to maximize the *usability* U of a letter assignment, i.e., the mapping of each character in a character set to a unique posture (gesture) of the hand. To characterize U, we formulate a multi-term objective function for mid-air text entry called PALM which addresses four factors affecting mid-air text entry with multiple fingers: Performance, Anatomical comfort (individuation), Learnability, and Mnemonics. In addition to performance and individuation, we formalize learnability and mnemonics based on existing literature.

Usability U is thus defined as a weighted sum of four normalized (i.e., $\in [0, 1]$) terms[2]. Formally, we write our usability objective as

$$U = w_p \hat{P} + w_a \hat{A} + w_l \hat{L} + w_m \hat{M}, \tag{8.5}$$

[2]Normalized variables are marked with a hat.

where the positive weights w_p, w_a, w_l, and w_m, which are set by the interaction designer based on their criteria, sum up to 1. The remaining terms in the objective function are described below in turn.

Performance Term (P)

Our performance score P is measured in words per minute (WPM). Following previous work on keyboard optimization [34, 170], we use Fitts' law models to predict the time $mt_{k\ell}$ to articulate a joint from letter k to letter ℓ by computing the movement time as described in Equation 8.3.

We then compute WPM with 5 % error rate as:

$$P = 60/(\sum_k \sum_\ell f_{k\ell} mt_{k\ell}) \times 5, \tag{8.6}$$

where $f_{k\ell}$ is the frequency of bigram $k\ell$, where k and ℓ are over the letters of the alphabet.

Anatomical Comfort Term (A)

For each gesture, we use Equation 8.4 to estimate how well it individuates. An index of 1 corresponds to perfect individuation where none of the non-instructed joints moves along with the joints involved in the gesture, a value of 0 would mean that all fingers move to the same extent, even if they are not part of the gesture. Thus, $\hat{A}$ takes the value of the individuation index.

Learnability Term (L)

Learnability is an important factor to consider for any activity involving rapid and careful articulation of multiple joints. To develop a score for learnability of a gesture, we build on some prevalent theories of motor learning that view learning as a *hierarchical combination of primitives* [87]. According to this view, the brain simplifies multi-dimensional motor control by collapsing it into a few dimensions. Practicing a complex gesture gradually increases hierarchical organization and decreases reliance on feedback. This has two consequences. First, the fewer DOFs a gesture involves, the easier it will be to learn. For instance, gesturing with one finger is easier to learn than a gesture using three fingers. We name the number of involved DOFs u_{dofs}. Second, if the involved digits involve the same *end posture*, it will be easier to learn because the articulations can be represented with a single learning primitive. For example, it is easier to extend all digits by 40° than to extend some by 20° and others by 40°. We denote the number of DOFs for which a target angle is defined in a gesture by

u_{targets}. Our learnability score combines these two aspects:

$$L = 1 - \sum_{k} (0.5\,\hat{u}_{\text{targets, k}} + 0.5\,\hat{u}_{\text{dofs, k}}). \tag{8.7}$$

_M_nemonics Term (M)

Studies of human memory suggest that categorization, chunking, and mnenomics help forming more durable long-term memory traces among otherwise unrelated materials [149]. Our mnemonics score M considers the memorability of a letter assignment *as a whole*. We call a *mnemonic set* a set of similar gestures, such as gestures that all have a neighboring finger. To identify finger mnemonics, we build on a recent study of multi-finger chord gestures that showed a positive effect on learning [149]. We take the mnemonic principles presented there and extend them from three fingers to five. In particular, we include the following mnemonics rules: neighboring fingers (e.g., thumb and little finger together), base (e.g., thumb or index with other fingers), and single finger.

The M-score considers two aspects: (1) the proportion of gestures belonging to a mnemonic set m_{coverage} and (2) how *few* mnemonic sets are required m_{sets}, which is the inverse of the proportion of all mnemonic sets being in use. We define $M = 0.5\,(m_{\text{coverage}} + m_{\text{sets}})$. M thus rewards designs where a large proportion of gestures belong to a few mnemonics sets. While our learnability score L looks at motor learning "from scratch", this score focuses on the benefit of the set consisting of easily recognizable gestures.

8.5.4 Step 4: Optimization

To optimize the multi-term objective function we use techniques from multi-dimensional Pareto optimization [110]. Instead of searching for a global optimum in a single run, we use a multi-start local search method. Local search starts from a random position in the search space and randomly samples its neighborhood. When search converges, we store the current best solution to a file and restart search. A similar approach was used in a previous paper addressing a multi-objective task [34]. Our implementation reaches reasonable designs in minutes while good ones take about one day on a cluster computer.

8.6 Design Cases

This section presents mappings optimized for fast performance, learnability, as well as for different character sets. Apart from this, we present solutions with multiple discretization levels for the joint angles. This demonstrates how the approach can be used across varying design interests. Finally, we present a preliminary evaluation of one of our designs.

Before discussing the designs, we report our experiences regarding the value of optimizing for all four objectives of PALM. To learn if performance and individuation are compatible design goals, we optimized for P, A, and P+A goals separately. The results showed that the benefit of optimizing for only one of the goals is negligible. In other words, performance and individuation may not always be competitive goals for design. The P-only design has fewer multi-joint gestures, whereas both A-only and P+A have more gestures involving neighboring fingers. This encouraged further exploration of the multi-objective design space.

Bin Address	Character	Bin Address	Character
0,1,0,0,0	_	1,1,0,0,0	n
1,0,0,0,0	a	1,0,0,1,0	o
0,0,1,0,1	b	0,0,0,1,1	p
1,1,0,1,0	c	0,1,1,1,1	q
0,1,1,1,0	d	0,1,0,1,0	r
0,0,0,1,0	e	0,1,1,0,0	s
1,1,1,1,0	f	0,0,1,0,0	t
0,1,0,0,1	g	0,0,0,0,1	u
0,0,1,1,0	h	1,0,0,1,1	v
1,0,1,0,0	i	1,0,0,0,1	w
0,1,1,0,1	j	0,0,1,1,1	x
1,1,0,0,1	k	0,1,0,1,1	y
1,1,1,0,0	l	1,0,1,0,1	z
1,0,1,1,0	m		

Table 8.4 FastType was optimized favoring Performance. The bin addresses describe each gesture, see text for explanation. Observe how commonly occurring letters like ‘a’ are assigned to easy postures such as flexing the thumb.

Table 8.5 lists all outcomes along with two alternative text entry methods: Engelbart’s chording keyboard [36] and a fingerspelling method (American Sign Language). Words per minute is predicted considering expert motor performance only, using Equation 8.6. Due to space limitations we report the full mapping only for FastType in Table 8.4.

Standard Character Sets: NumPad is a solution that maps the numbers from 0–9 to postures formed by the 5 joints, one per finger. Each joint angle is discretized into 2 levels. The predicted performance for this mapping is the highest at 113.0 WPM due to the small character set. FastType is a solution with the letters *a–z* (including space), and 5 joints each with 2 discretization levels. This mapping was optimized for typing speed and uses chord-like movements with a predicted performance of 54.7 WPM. We show this mapping in Table 8.4. In the table, we use the concept of bin address as explained earlier. The joints are ordered from Thumb to Little. For example, [0,0,0,0,1] would mean flexing Little but keeping the rest in a neutral pose. BalanceType, a variant with balanced weights for the four objective function weights had a predicted performance of 50.1 WPM.

Extended Character Sets: FullType is optimized to map all letters of the alphabet, numbers, and special characters for a total of 48 characters. The predicted performance was 50.7 WPM with 5 joints and 5 discretization levels per joint. While this mapping has a good predicted performance, we hypothesize that it is hard to perform because of 5 discretization levels for joint angles. Finally, ThreeType optimizes a full keyboard to the three fingers with the highest individuations: Thumb, Index, Middle. It, too, assumes 5 discretization levels which is presently impossible with the Leap Motion and would require a long time to learn.

We also represented fingerspelling in American Sign Language using our bin address notation. For the represented mapping, our objective function predicts an entry rate of 43.9 WPM which is surprisingly close to the empirically observed rate of 40–45 WPM for experienced practitioners [112].

Mapping	**Character Set**	**Joint Discretization**	**Weights** (PALM)	**Objective values** (PALM)	**Predicted WPM**
NumPad	0–9	2, 2, 2, 2, 2	0.30, 0.30, 0.05, 0.05	0.27, 0.03, 0.22, 0.22	113.0
FastType	*a*–*z*	2, 2, 2, 2, 2	0.50, 0.10, 0.10, 0.30	0.53, 0.03, 0.18, 0.50	54.7
BalanceType	*a*–*z*	5, 5, 5, 4, 4	0.25, 0.25, 0.25, 0.25	0.42, 0.02, 0.19, 0.17	50.1
FullType	0–9, *a*–*z*	5, 5, 5, 5, 5	0.20, 0.20, 0.20, 0.20	0.41, 0.14, 0.19, 0.33	50.7
ThreeType	*a*–*z*	5, 5, 4	0.40, 0.40, 0.20, 0.00	0.38, 0.01, 0.28, 0.00	65.1
Fingerspelling	*a*–*z*	4, 3, 3, 3, 3	0.25, 0.25, 0.25, 0.25	0.51, 0.02, 0.28, 0.80	43.9
Engelbart's Chord Kbd	*a*–*z*	2, 2, 2, 2, 2	0.25, 0.25, 0.25, 0.25	0.58, 0.03, 0.17, 0.69	49.0

Table 8.5 An overview of optimized mappings and predicted WPM. The bottom part shows predictions for two existing methods.

8.6.1 First Observations on User Performance: FastType

In order to estimate if the predicted performance is indeed achievable with mid-air text entry, we conducted a preliminary evaluation of FastType with 10 users. We followed a word-level paradigm previously used by Zhai et al. [17]. Here, a randomly sampled word is practiced until performance peaks. The benefit of this is that the upper boundary of entry performance can be estimated even without having to learn the full gesture set.

Method: 10 right-handed participants took part in the experiment (9 male, 1 female; ages from 21 to 39, mean 26). The experiment took 1.5–2 hours and all participants were compensated. We randomly sampled 4–8 character strings from the Enron Email Dataset [148] for the stimulus. Each contained 1–2 frequently entered words and also included the space character. A task consisted of repeatedly entering a word. At the beginning, participants were allowed to practice the word by going through the gestures for all letters and exploring the fastest transitions between each gesture. As soon as they could memorize the mapping of the corresponding letters, the task started. The task was terminated by the experimenter when a performance plateau could be observed.

Prototype: We built a prototype that allowed users to enter text, and recorded performance of typed words. Our gesture recognizer used joint angle data from the Leap Motion, and used a combination of dwell times and signal peak detection to detect when users made a particular posture which was converted to text. A custom-built application displayed information to the user as well as recorded data for analysis. The hardware used was identical to the first experiment.

Result: Overall, 10 users entered 53 words at an average peak performance of 22.25 WPM (SD 8.9). For analyzing the peak performance of each word, we extracted the top 3 repetitions with an error rate less than 15% (measured by Damerau-Levenshtein distance). Three words had to be excluded due to this restriction. The remaining words were typed with an average error rate of 2.3% (SD 0.04). A one-way ANOVA on WPMs showed a statistically significant difference among users: $F(9, 49) = 7.68$, $p < 0.001$. Average peak performances ranged from 13 WPM to 38.1 WPM. This large performance range clearly shows the influence of individual differences in performance, individuation and anatomical limitations found in our first experiment. While these results serve as a first exploration of PALM, further detailed studies are needed to validate the effectiveness of our model.

8.7 Discussion

The results presented in this chapter deepen the understanding of multi-finger input in mid-air. The findings show that multi-finger input has potential for high throughput. While it was known previously that differences existed in performance and individuation between fingers, they were not quantified in a setting that is representative of modern computer vision-based input. Our results were obtained by adapting the familiar methodology of Fitts' law studies along with a measurement of individuation adopted from motor control research. This is in contrast to existing work in gesture design that has considered elicitation methods to learn about user preferences, intuitiveness, and social acceptability [89, 104, 113].

In a proof-of-concept, we demonstrated the applicability of our results by computationally optimizing a mid-air text entry method. Based on prior work on motor performance [60, 116], we extended our findings from single fingers to multi-joint gestures. The P and A terms of PALM are based on the empirical results, whereas the L and M terms are derived from prior work on human memory and motor learning [87, 149]. While further evaluation is needed to prove the validity of these assumptions, we show how our findings can serve in the search for good solutions among millions of designs.

To analyze the outcomes, we built a prototype and explored the performance for one of the optimized mappings which showed an entry rate of 22 WPM. While the performance predicted by Equation 8.6 was surprisingly close to the observed performance in fingerspelling, FastType falls short of the predicted rate of 54.7 WPM. As Equation 8.6 only predicts expert motor performance, this can be partially attributed to the lack of training and limited performance of the Leap Motion. We think that using the approach presented in Chapter 5 would already lead to better text entry performance. However, further evaluation is needed to investigate learning over time and cognitive effort involved in mid-air input.

8.8 Conclusion and Future Work

In this chapter, we presented, to our knowledge, the first investigation of the dexterity of human fingers for mid-air input. The results provide insights into the performance of individual fingers and their coactivation. The findings suggest that mid-air input is a promising input modality, but there are limitations to the capacity of the human hand.

The physiology and cognitive skills of humans pose two critical constraints that future work should consider. First, the learnability of gestures is a pragmatic obstacle for multi-finger input. If a gesture set for text entry is prohibitively time consuming to learn it will affect large-scale adoption. With PALM, we propose the first method to computationally design gestures and optimize for objectives such as learnability. However, further evaluation is needed to investigate the influence of the L and M term on performance and learnability, and evaluate the involved models. Second, the effect of fatigue in multi-finger input is not fully understood yet. Users in both our studies reported discomfort in their arm and wrist.

The technological challenges of hand tracking without markers pose additional constraints to mid-air input. Since the work in this chapter preceded the tracker described in Chapter 5, we were limited to using the Leap Motion sensor which has shortcomings when tracking hand pose. We restricted our study to 6 joints since the Leap Motion could not reliably track certain finger joints. We are confident that we can achieve better accuracy and text entry speed if we use our method due to its faithful reconstruction of difficult poses (see Figure 5.7 for a comparison).

Our evaluation showed that users were limited in their speed by errors in tracking all joint angles under fast motion. We assume that some of these issues arise from assumptions about finger individuation used by the tracker.

While no visual feedback is needed for our text entry method, it is unknown if proprioception alone suffices to perform fast and accurate mid-air gestures. As an alternative, tactile feedback was shown to improve performance on touch screens [54] and new technologies such as UltraHaptics [24] provide a way to bring non-contact haptic feedback to mid-air input.

This chapter has contributed the first known empirically derived models of performance factors involved in mid-air input and a proof-of-concept approach to design. Our optimizer allows finding designs that strike desirable trade-offs in this demanding design landscape. We believe that when the outstanding human and technological issues are solved, this category of input can achieve performance that is currently seen only for physical keyboards. In the next chapter, we present an approach for continuous and discrete gesture input for small factor devices like smartphones. We show how limiting hand tracking to only two fingertips can still enable expressive forms of input for wearable devices.

Chapter 9

On- and Above-Skin Sensing for Continuous and Discrete Input

In the previous two chapters, we discussed two different paradigms to gesture-based input design: elicitation studies and computational design. We showed applications of elicitation studies to continuous 3D navigation tasks and computational gesture design to discrete text entry. Even though we did not show examples, computational design can also inform continuous gesture design.

In this chapter, we investigate the combined use of both discrete and continuous gestures for input to small form factor devices (e.g., smartwatches). We also assume simpler tracking conditions by relying only on fingertip positions instead of full hand pose. We show that even under these simpler conditions, we can enable a rich set of expressive input for emerging devices. To our knowledge, ours is the first approach to support mid-air and multitouch interactions on- and above-skin. Parts of this chapter previously appeared in [128].

9.1 Introduction

This chapter discusses novel input capabilities enabled by computer vision sensing on small wearable devices such as smartwatches. Every new generation of these devices features better displays, processors, cameras, and other sensors. However, their small form factor imposes severe limitations on the efficiency and expressiveness of input. Touch input is restricted to a tiny surface, and gesture input may require moving a whole body part [15]. We address these challenges by investigating a class of emerging sensing techniques that support extending the input space to the space next to a wearable device. This could solve the problems caused by small surface area. Additionally, it may enable a new possibility

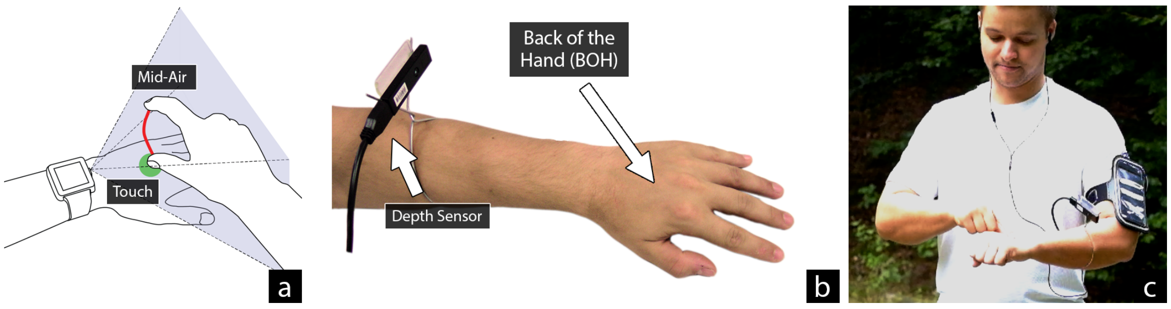


Figure 9.1 (a) *WatchSense* enables on- and above-skin input on the back of the hand (BOH) through a wrist-worn depth sensor. (b) Our prototype mimics a smartwatch setup by attaching a small depth camera to the forearm. (c) WatchSense tracks the 3D position of fingertips as well as touch on the BOH in real-time on consumer mobile devices. This enables a combination of mid-air and multitouch input for interactive applications on the move.

for multi-device interaction: controlling not only the wearable device itself but also relaying sensed input to allow interaction with nearby devices, such as TVs, smartphones, and virtual/augmented reality (VR/AR) glasses [55, 85].

We contribute to an emerging line of research exploring richer use of finger input sensed through a wearable device. In particular, we look at smartwatches, which have previously been supplemented by mid-air finger input [49, 66, 68, 76]. Recent work propose using the palm or forearm for gestures or touch input (e.g., [91, 150, 151, 158]). This enlarges the size of input space in which gestures can be comfortably performed. However, previous work focused on either touch *or* mid-air interactions. We address the *combination* of these two modalities, with the aim of increasing the efficiency and expressiveness of input. Recent advances in depth sensor miniaturization have led to the exploration of using both touch and mid-air interactions above smartphones [28]. To our knowledge, there is no work that explores the use of both touch and mid-air input in smaller, wearable form factor devices such as smartwatches.

Our second contribution is to address the technical challenges that arise from sensing of fingers that touch the skin and/or hover above the skin near a smartwatch with an embedded depth sensor. Recent improvements to real-time finger tracking in mid-air [65, 119, 129, 161] cannot directly be employed due the oblique camera view and resulting occlusions which are common in body-worn cameras. To address these challenges we propose a novel algorithm that combines machine learning, image processing, and robust estimators. Our method accurately detects and estimates 3D positions of interacting fingertips and robustly detects fingertips touching the back of the hand (BOH). Our prototype (Figure 9.1 (b, c)), which mimics the viewpoint of future embedded depth sensors, can detect fingertips and touch events in real-time (> 250 Hz on a laptop and 40 Hz on a smartphone). Additionally,

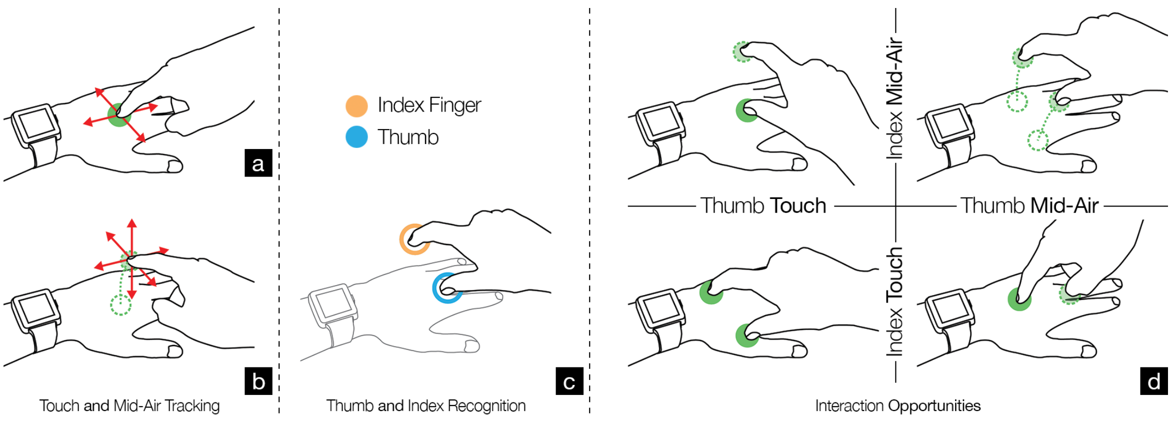

a
Touch and Mid-Air Tracking
b
Index Finger
Thumb
Thumb and Index Recognition
c
Index Mid-Air
Thumb Touch
Thumb Mid-Air
Index Touch
Interaction Opportunities
d

Figure 9.2 (a, b) *WatchSense* tracks fingertips in mid-air, touch, and position of touch on the back of the hand (BOH). (c) It also distinguishes between different fingers. In our prototype we can recognize the index finger and thumb. (d) The technical capabilities of *WatchSense* enable more expressive interactions such as purely mid-air (top right), purely touch (bottom left), and combinations of them.

technical evaluations show that our approach is accurate and robust for users with varying hand dimensions.

The capability enabled by our approach allows for *simultaneous* touch and mid-air input using multiple fingers on and above the BOH. Supporting both modalities with the same sensing approach is not only beneficial for users but provides more options to design and opens up new application possibilities. We show through several applications that this novel input space (or volume) can be used for interaction on the move (e.g., to the smartwatch itself or to other nearby devices), complementing solutions with touch or mid-air alone. In summary, this chapter contributes by:

- Exploring the interaction space of on- and above-skin input near wearable devices, particularly smartwatches.
- Addressing the technical challenges that make camera-based sensing of finger positions and touch a hard problem.
- Demonstrating the feasibility of our approach using a prototype, technical evaluations, and interactive applications.

9.2 WatchSense

Figure 9.1 (a) illustrates the vision of *WatchSense*. We assume that smartwatches will embed a depth sensor on their side, overseeing the back of the hand (BOH) and the space above it. In this section, we first outline the vision of embedded depth sensors and how we prototype this vision. Then, we outline the new interaction opportunities afforded by *WatchSense*, and present the arising tracking challenges.

9.2.1 Embedded Depth Sensors

Advances in time of flight (TOF) imaging technology has led to rapid miniaturization of depth cameras. A few years ago, the smallest TOF sensor (Swissranger SR4000[1]) had a size of $65 \times 65 \times 68$ mm. Today, the PMD CamBoard PicoFlexx[2] measures only $68 \times 17 \times 7.25$ mm. While these sensors do not yet fit into a smartwatch, the trend indicates that smaller sensors will be integrated into smartwatches in the near future.

To study the utility of such embedded sensors already, we created a prototype with viewing angles close to a hypothesized integrated depth sensor. Figure 9.1 (b) shows our prototype setup: a small depth sensor is attached to the user's forearm facing the wrist. Due to near range sensing limitations of these sensors (usually designed for sensing up to 2 m) we had to place them at a distance of 20 cm from the wrist. However, we envision specially designed future TOF sensors will allow better near range sensing capabilities.

9.2.2 Input Capabilities

WatchSense is capable of sensing fingertip positions (of the interacting hand) on and above the BOH. This opens up new interaction opportunities for multi-finger interactions – both while touching the BOH as well as in mid-air. The resulting input space provides higher expressiveness and degrees of freedom than skin-based touch. While this is interesting for input directly to smartwatches, we envision the watch to be the *input sensing device* for a large variety of other interactive devices (see Application section for examples). Figure 9.2 highlights the possible combinations with *WatchSense*.

Touch and Mid-Air Tracking: With *WatchSense*, the BOH can be used as a touchpad with the same operations: sensing when a touch operation began, when the finger moved (reporting its x, y coordinates in the plane, where z is 0), and when it is lifted (see Figure 9.2 (a)). Additionally, sensing the space above the BOH allows for using mid-air gestures (see Figure 9.2 (b)). Here, however, the sensor reports 3D x, y, z coordinates. Thus, *WatchSense* offers 3 degrees of freedom (DoF) per finger. Transitioning between touch and mid-air input allows for similar interactions as shown in *Air+Touch* [28].

Finger Identification: *WatchSense* supports the identification of fingers (see Figure 9.2 (c)). For instance, this allows for assigning different interactions to different fingers (i.e.,

[1]Swissranger SR4000: http://hptg.com/industrial/
[2]CamBoard PicoFlexx: http://pmdtec.com/picoflexx/

touching or gesturing with the thumb has a different meaning than when doing so with the index finger). While we envision to identify all five fingers, we here focus on showcasing the opportunities using the thumb and index finger.

Multi-Finger Touch & Mid-Air: Combining finger identification with touch and mid-air sensing (and the resulting 3 DoF per finger) enables compound interactions. The matrix in Figure 9.2 (d) showcases the possible combinations, and the examples presented later in this chapter highlight their use. Essentially, when the interacting hand is present, each finger is either *touching* the BOH, or positioned in *mid-air*. We use the following terminology throughout the chapter: the overall interaction state is described by a tuple containing the thumb's state and the index state (i.e., if the thumb is touching, and the index is not, the overall state is *Touch* + *Mid-Air*).

These combinations can be used with large variation. For example, in *Touch* + *Mid-air*, the hand can be utilized as joystick, where the thumb acts as base, while the index finger rotates around that base. In *Touch* + *Touch*, the BOH is utilized as multi-touch surface. *Mid-air* + *Touch* is often utilized when using the BOH as a touchpad in single-touch interactions. However, the thumb's mid-air position (and distance to the index finger may be used for value-changing operations (e.g., adjusting the volume of a music player). Lastly, in *Mid-air* + *Mid-air*, both fingers can gesture freely in 3D. We, however use this last state as a delimiter for entry/exit to other states.

9.2.3 Resulting Challenges

We assume that a camera obtains an oblique depth map of the BOH and the space directly above it. This differs greatly from previous approaches that use depth sensing for multitouch input. *Imaginary Phone* [44] and *OmniTouch* [48] assumed a near-perpendicular view of the surface, easing separation of the interaction surface from the interacting hand. These systems showed limited accuracy when distinguishing touch and hover states (e.g., *OmniTouch* reports 20 mm accuracy). Other systems, such as *Air+Touch* [28] rely on a perfectly planar, touch-sensitive surface on a smartphone in addition to the depth sensor.

Realizing our scenario without additional sensors on the hand poses new challenges: (1) the oblique view of the BOH causes perspective distortion and additional occlusions, (2) the BOH (as well as the forearm) is not a flat surface but curved, which complicates touch detection. (3) multi-finger interaction requires the discrimination and *identification* of fingertips, both when touching and hovering, and (4) compute limitations on mobile devices require the sensing technique to be fast with low latency. *WatchSense* supports *simultaneous* and *continuous* touch and mid-air interactions from an oblique view of the BOH in *real-time*—even in the presence of these challenges.

9.3 Related Work

The work presented in this chapter builds on recent approaches for interacting on smartwatches and on associated limbs, mid-air interaction techniques around wearable devices, as well as hand and finger tracking.

Touch Interaction On and Around Smartwatches: Interaction with consumer smartwatches is generally limited to touch and speech. Two main strategies have been explored to extend the capabilities of such devices: (1) on-device interaction, and (2) on-body interaction.

On-device interactions beyond the touchscreen employ other parts of the smartwatch. Pasquero et al. [101] extended input to the device's bezel. Xiao et al. [164] use the entire watch face for additional input, e.g., through tilting, twisting or panning it. *WatchIt* uses the wristband as alternative input canvas for simple gestures [102]. *WatchMI* [169] uses existing sensors to support pressure touch, twisting, and panning gestures. While shown to be beneficial, they all consider input only directly on the device.

Smartwatches have mostly planar body parts in close proximity (e.g., the hand and forearm). Thus, there is a large body of research on skin-based input to free the interaction from the watch itself. *iSkin* uses a thin skin overlay to detect touch and strokes [157]. *Skinput*'s bio-acoustic sensing array allows for detecting a touch directly on the skin. SkinTrack [171] uses the body as an electrical waveguide to support touch near smartwatches. Laser-based range scanners [151, 150] as well as infrared sensors placed at the device's borders [21, 91, 136] are vision-based approaches to detect on-skin touch and gesture interaction around a device.

Most related, however, is the use of depth cameras to detect skin-based input. *Imaginary Phone* used a depth camera to detect interaction on the palm [44] to operate a mobile phone which is not in sight. *OmniTouch* used a very similar setup to turn arbitrary (planar) surfaces (also the user's palm or wrist) into projected, interactive surfaces [48]. *WatchSense* is inspired by these systems but we go beyond by recognizing fingertip positions, identities, and touch on- and above-skin.

Gestural Interaction Around Wearable Devices: Mid-air space around wearable devices has also been investigated for input. Initially, researchers used that space for simple gestural input. *Gesture Watch* [66], *AirTouch* [76], and *HoverFlow* [68] used an array of infrared sensors to execute simple commands through eyes-free gestures. More recently, researchers began exploring techniques that rely on more accurate mid-air tracking. Here, they relied on magnetic tracking (e.g., *FingerPad* [26], *Abracadabra* [49], and *uTrack* [27]), or small infrared cameras (e.g., *Imaginary Interfaces* [43]). To test a set of interaction techniques, researchers often relied on sophisticated external tracking systems (e.g., [58, 51]).

Finally, there is research on using the fingers for gestural input, either using a vision-based approach [65, 155], or through strain sensors on the back of the hand [77].

The aforementioned systems solely used gestural input without considering touch, which is a key feature of *WatchSense*. One of the few systems considering both touch and mid-air during an interaction is *Air+Touch* [28]. Their focus is on sequential interactions near smartwatches, where mid-air interaction occurs before, after or in between touches. In contrast, *WatchSense* allows for simultaneous use of touch and mid-air.

Vision-based Tracking of Hands and Fingers: With the advent of commodity depth sensors, research on articulated hand tracking (e.g., *Digits* [65]) has gained more attention [64, 119]. We presented an approach in Chapter 5. These approaches aim at reconstructing hand pose from depth data, and would be, at first glance, an ideal solution for our scenario. Unfortunately, these methods fail under oblique views, occlusions, or additional objects in the scene. In addition, they are not well-suited for detecting (multi-)touch events. To bypass these issues, existing systems (that make use of finger input) often simplify the problem: first, systems avoid fully articulated hand tracking and only require detecting discrete touch points (e.g., [161, 21, 10, 28]). Second, several systems build on heuristic assumptions of the depth camera's location in relation to the interaction surface which is hard to realize in practice. For example, both *OmniTouch* [48] and *Imaginary Phone* [44] assume a perpendicular view of the interaction surface, easing separation of the interaction surface from the interacting hand. In addition, these systems have limited accuracy when distinguishing touch and hover states (e.g., *OmniTouch* reports 20 mm accuracy [48]). Other systems, such as *Air+Touch* rely on a perfectly plain, touch-sensitive interaction surface (a smartphone) [28].

In comparison, our work builds on less heuristic assumptions while accurately detecting fingertips on and above the interacting surface. Taking inspiration from [124, 48, 76] we use a combination of machine learning, image processing, and robust estimators to solve the challenging vision problem. Our approach is flexible and can be retrained to fit a wide range of depth sensor positions (e.g., in the device itself), surfaces (e.g., upper arm). Additionally, we obtain information about finger identity that increases the expressiveness of interactions possible with our approach.

9.4 Implementation

We now describe our depth camera-based method for supporting the expressive mid-air and multitouch interactions. Our focus is on fingers interacting on and above the BOH from an arm-worn camera. Our approach is fast and accurate —we can track the position of fingertips to within 15 mm, and touch points to within 10 mm. Our approach is also flexible—it can be reused with only a few changes to suit other wearable cameras, and viewpoints.

Previous methods [48, 76] for near-surface finger interaction support estimation of the following: (1) 3D hover/touch positions of fingertips, and (2) exact detection of finger touch events. Our approach supports these and additionally also (3) automatically and robustly identifies fingertips (currently index finger and thumb). This allows us to support a richer set of mid-air and multitouch interactions. Our approach also delivers better touch detection tolerances than previous work.

9.4.1 Prototype System

Our prototype can run on desktops, laptops, tablets, and smarphones and relays sensed fingertips positions, labels, and touch events through a WebSocket connection. Clients such as smartwatches, smartphones, public displays, or smartglasses can obtain this information wirelessly.

In our prototype, we use the PMD CamBoard PicoFlexx camera (see Figure 9.1), which is currently the smallest commercially available depth sensor. We found its size, resolution, and noise characteristics suitable for the BOH scenario. However, we also support other close range sensors like the Intel Senz3D depth, and the Intel RealSense F200. We position the sensor on the forearm (20 cm above the wrist) facing the BOH (see Figure 9.1). Placing the sensor closer to the wrist was not possible because commercial TOF cameras have limited near range sensing capability. Their infrared illumination source—designed for ranges >50 cm—saturates pixels with depth less than 20 cm thus making depth estimation unreliable. Specially designed cameras with less intense illumination sources will allow nearer sensing ranges.

9.4.2 Algorithm Description

Estimating fingertip positions, and touch events from an oblique view of the BOH is a hard problem. Even state-of-the-art articulated RGB-D hand trackers would fail under these conditions [129]. We use a detection rather than tracking strategy to help recover in case of failure. Our approach features a novel combination of random forests, advanced image processing, and robust estimators to achieve stable and accurate fingertip and touch detection. Figure 9.3 provides an overview of our approach. While the algorithms we use are known in the computer vision community, their novel combination and specific application to this problem has many unique contributions as we describe below.

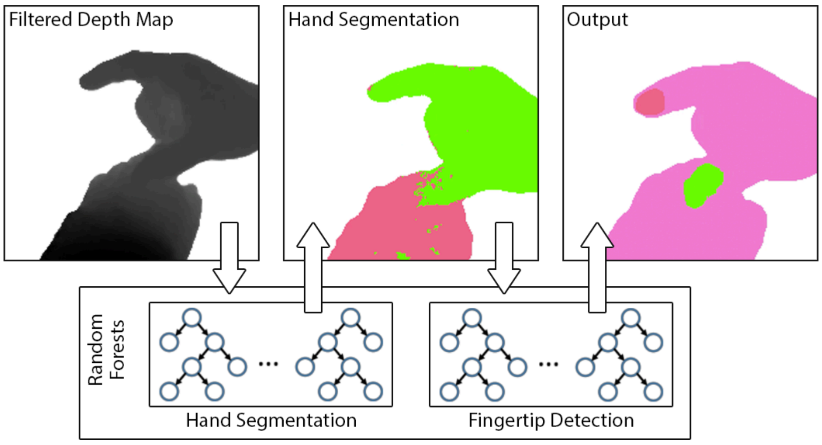


Figure 9.3 Overview of *WatchSense* implementation. After pre-processing the input depth image we use random forests to segment interacting hand from the BOH, and detect fingertips. The segmentation masks are used together with robust estimators and flood filling to obtain fingertip positions, and touch points.

Random Forests for Classification: We use random forests for per-pixel classification which have been shown to produce state-of-the-art results in human pose estimation and other segmentation problems [122, 64, 129]. We provide a brief overview and refer the reader to Chapter 2 and [30] for more details. Our contribution is to show that random forests in combination with other techniques enable new interaction opportunities for on- and above-skin wearable interaction.

Given an image, a classification forest is trained to label each pixel into a class label (e.g., part of a human body). At test time, for each input pixel, a tree in the forest makes a prediction about which part it likely belongs to. The output from all trees in the forest is aggregated to provide a final prediction about the pixel's class as $p(c\,|\,x, \boldsymbol{\tau}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c\,|\,x, \boldsymbol{\tau}_t)$, where p is the predicted class distribution for the pixel x given forest hyperparameters τ, T is the number of random trees that makes a prediction p_t. We use depth-based feature response functions similar to the one described in [122].

Input Preprocessing and Segmentation: The input depth map encodes real-world depth at each pixel. Noise in the depth map is removed using morphological erosion and a median filter to produce a filtered depth map [39]. To make subsequent steps in our method more robust, we first use a binary classification forest that segments the two interacting hands into BOH and interacting hand (see Figure 9.3). This segmentation generates two depth maps—one contains only the BOH and the other contains only the interacting hand.

Fingertip Detection and Recognition: The goal of this part is to detect and estimate the 3D position of interacting fingertips. In our prototype, we assume that only two fingers interact (i.e., index finger and thumb)—however our approach is flexible and can support more than two fingertips. Additionally, we trained our method to be robust to false positives on unsupported fingers. The key improvement over previous work is our ability to detect fingertips and also their unique identity even after periods of occlusion. In contrast, [76]

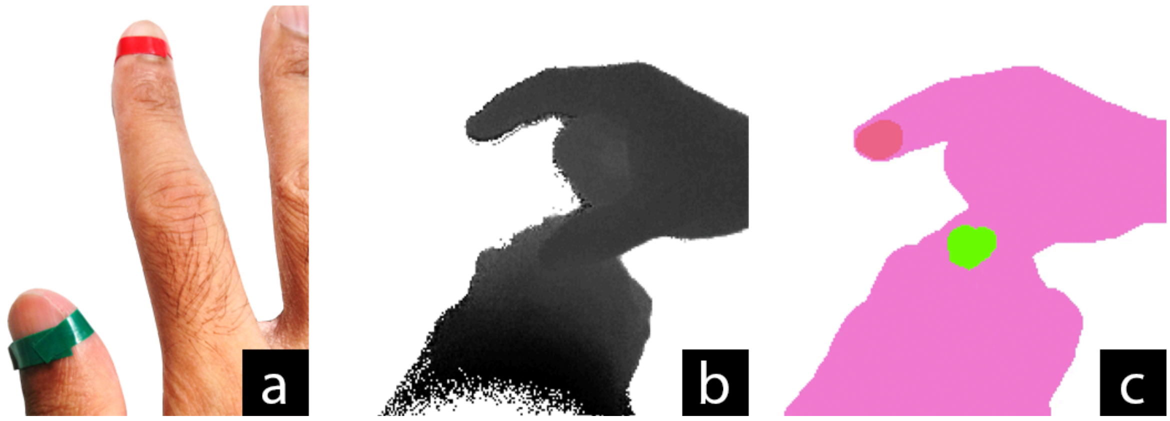


Figure 9.4 Fingertip detection. (a) Training time: Different users wear colored fingertip caps to provide pixel training data for fingertip locations. (b, c) Testing time: Fingertips and their respective labels are accurately detected from only depth images in at real-time.

uses only one finger while [48] uses heuristics to assign unique IDs without knowing finger identity. As we show in the applications section, fingertip identity allows us to create more expressive interactions previous not possible.

We rely on a random forest that classifies pixels into one of three classes: *IndexFinger, Thumb, Background*. More classes can be added if needed. At training time, we collected color and depth image pairs from multiple users interacting with the BOH wearing colored markers (see Figure 9.4). These markers were automatically detected in the color image and mapped onto the depth image. This provides labels for the forest to be trained on—we collected 20000 image pairs from different users to maximize forest generalization.

At test time, given an input depth image, the forest classifies pixels into one of the three classes. The result, shown in Figure 9.4, produces a group of pixels that are labelled into one of the fingertips. We remove noise in the resulting pixels by a median filter and morphological erosion. We then obtain a robust estimate for the 2D fingertip position on the image by applying the MeanShift algorithm [29] which is robust to outliers. The final 2D position is then backprojected using the depth map to obtain the 3D fingertip position along with its identity (Figure 9.4).

Our approach is resilient to temporary tracking failures since the fingertips are detected frame-by-frame. For added stability, we filter the final positions with the 1€filter [25]. Because we identify fingertips uniquely we can support more expressive interactions previously not possible, as we show in our interactive applications.

Touch Detection: The second goal is to robustly detect touching of the fingertips on the BOH. This is a hard because depth sensors have limited precision and much noise. The oblique camera view, general BOH shape, and camera motion make it even harder. We experimented with various techniques including distance computation from a plane fitted to the BOH. However, we found that flood filling, similar to the approach used by OmniTouch [48], worked best.

Figure 9.5 illustrates touch detection with flood filling. For each detected fingertip, we seed the flood filling process at the 2D fingertip position. We then fill a fixed mask around the fingertip such that pixels of certain depth in front and behind the fingertip (i.e., towards or away on the camera z-axis) are filled. We empirically chose the near and far thresholds to be 50 mm and 20 mm from the 3D fingertip position, respectively, which we found to cover a wide range of motion of the BOH, users, and finger orientations. Whenever more than 40% of the mask is filled, we activate a touch event. For robustness, we activate a touch event only when more than 10 frames (at the device runtime framerate) in sequence were detected as touching. As we show later, this method's touch detection tolerance varied from 1 mm to about 10 mm for different users which is better than the 20 mm reported by [48].

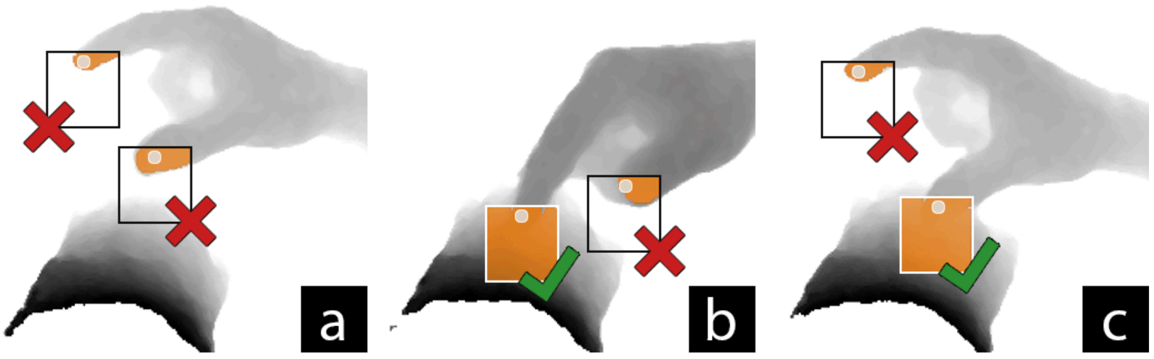
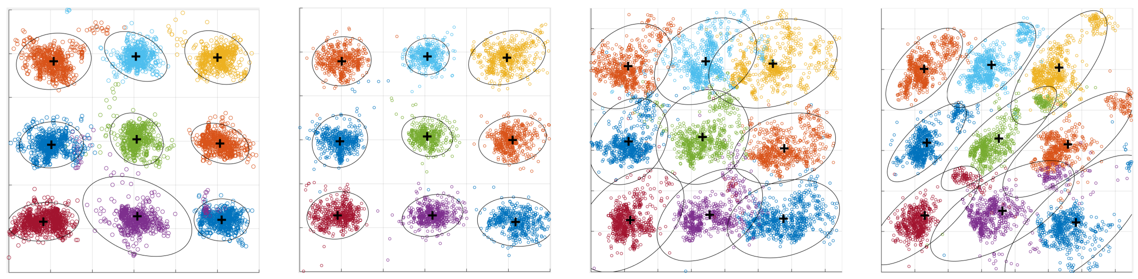


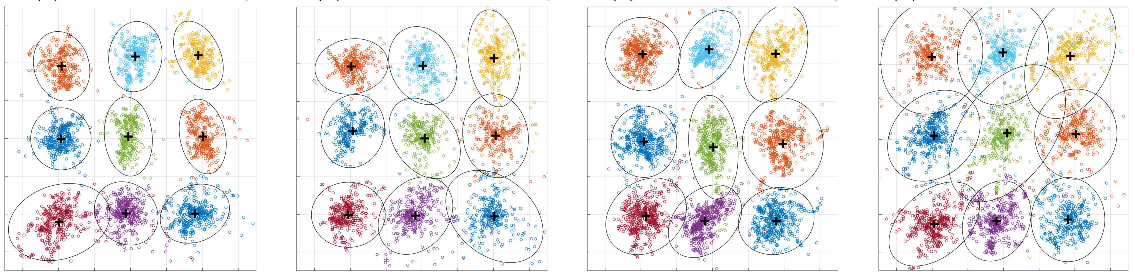
Figure 9.5 Touch detection. (a) When there is no touch, flood fill is restricted to filling only in parts of the finger. (b, c) When the finger touches flood fill grows into the BOH filling a larger area (White: seed point, Brown: flood filled pixels).

9.5 Technical Evaluation

In addition to assessing tracking speed, we evaluated several key performance aspects: (1) accuracy of fingertip tracking while touching the BOH and hovering above it; (2) reliable minimum distances (tolerance) between finger and the BOH to separate touch and hover; and (3) classification accuracy of the random forest. We first report our method's runtime performance.



(a) Senz3D Index Sitting (b) Senz3D Index Standing (c) Senz3D Thumb Sitting (d) Senz3D Thumb Standing



(e) PicoFlexx Index Sitting (f) PicoFlexx Index Standing (g) PicoFlexx Thumb Sitting (h) PicoFlexx Thumb Standing

Figure 9.6 Evaluation of touch accuracy on the BOH. Each image represents the 2D touch position distribution for a particular finger, condition, and camera [Senz3D is (a)–(d), PicoFlexx is (e)–(h)]. The plots contain all touch points recorded by the tracker during each trial. Ground truth positions are marked with a black plus symbol, and ellipses denote 95% confidence intervals. The index finger performed best for both sitting and standing conditions for all cameras. We attribute the relatively worse performance of the thumb to the lack of sufficient training data for the fingertip classification forest.

9.5.1 Runtime Performance

Our approach runs in real-time on ab Intel Core i7 laptop at >250 Hz, at >40 Hz on a recent smartphone (OnePlus 3), and at 35 Hz on a tablet (Nexus 9). However, we cap the output to 50 Hz to prevent flooding clients. All components of our method run completely on the CPU. Given the simplicity of our method and the increasing compute of smartphones, we expect to be able to run our method directly on smartwatches in the future.

9.5.2 Touch Accuracy

The goal of this evaluation is to assess the accuracy of fingertip position and touch detection. We model our evaluation on *OmniTouch* [48] and *SkinTrack* [171].

Method: We recruited 13 right-handed volunteers (2 female) from our institution, ranging in age from 23 to 36 years (mean 28.1 years). Their backs of the hand widths varied from 70 mm to 90 mm, and lengths varied from 60 to 80 mm (mean dimension was 82×70 mm).

The length of index fingers ranged from 69 to 86 mm (mean 79 mm), and the thumb length was between 55 mm and 70 mm (mean 63.5 mm). Since skin color affects depth and noise at each pixel, we recruited participants with diverse skin colors. An evaluation session took approximately 15 minutes. Data from one participant had to be excluded because of a software bug that affected the camera.

Design and Task: The *touch accuracy* task measures how accurately we can detect touch points on the BOH. We had two conditions in this task: (a) in the seated condition, participants were seated and their forearm was supported by the desk, (b) in the standing condition, participants stood without any arm-support. Participants then had to repeatedly touch dots on the back of their hand using either the thumb or their index finger. The computer next to the participants showed the dot they had to touch. The experiment began when participants pressed the spacebar, which would cause the first dot to be highlighted. Then participants had to touch that dot on the back of their hand, and subsequently press the spacebar to switch to the next trial. If there was no touch recorded prior to pressing the space-bar, participants could not advance to the next trial, and an error was recorded. We recorded x, y, z-coordinates for both fingers, and whether or not a finger (and which) was touching.

Apparatus: In the seated condition, participants rested their arm on a desk. The desk and chair used in our experiment were height-adjustable. The setup was replicated at two locations. Both seated and standing conditions took place in the front of a 55” 4K display or a 25” full HD display. The display and tracker was run on an Intel Xeon Dual Core (2.5 GHz) or on an Intel Xeon E3-1246 (3.5 GHz) machine. Half the participants were assigned to use the Creative Senz3D depth sensor while the other half used the PMD CamBoard PicoFlexx.

Procedure: In each of the two stages, participants either began with the index finger or the thumb, and performed all trials with that finger, before changing to the other finger. Half of our participants started with the index finger (the other half started with the thumb). The presentation of order in which the nine dots had to be touched was randomized for all tasks. In both touch accuracy tasks, each dot was touched 6 times per finger, resulting in *2* (*Tasks*) $\times$ *2* (*Fingers*) $\times$ *9* (*Dots*) $\times$ *6* (*Repetitions*) = *216* data points.

Before the experiment began, participants filled in a questionnaire containing demographic information. We then measured the size of their hands as well as the length of their thumbs and index fingers. Afterwards, we fitted the prototype on the forearm, and added 3$\times$3 dots on a participant’s back of the hand using a stencil to ensure equal separation of those dots (dots were separated by 20 mm).

Results: Figure 9.6 plots the distribution of touch points on the BOH, separately for standing and sitting, and for the two cameras used. Black crosses represent ground truth positions. The plots show that accuracy for index finger touch positions is high in sitting and standing conditions as well as for the two cameras. For the Senz3D, the mean standard deviation for the index finger was 4.1 mm for sitting and 3.7 mm for standing. For the PicoFlexx sensor, the mean standard deviation for the index finger was 5.2 mm for sitting and 3.7 mm for standing. The thumb performed slightly worse for both cameras. For the Senz3D, the mean standard deviation for the thumb was 7.7 mm for sitting and 8.4 mm for standing. For the PicoFlexx sensor, the mean standard deviation for thumb finger was 6.0 mm for sitting and 7.6 mm for standing. We attribute this difference to the lack of sufficient samples for the thumb during random forest training. However, we observe that the PicoFlexx camera performed better for the thumb than Senz3D. We would also like to highlight that our standard deviations improve over previous work [48] in spite of a smaller inter-dot distance of 20 mm instead of 30 mm.

9.5.3 Touch Tolerance

The purpose was to assess the hover interval, in which touch and hover detection can be ambiguous. Since we had no automated way of obtaining ground truth information for hover states, the evaluation was conducted through a series of manual measurements.

Participants: We recruited two right-handed volunteers (62 and 66 years). An evaluation session took 30 minutes.

Design, Task, and Procedure:: In order to provide as reliable measurements as possible, two tables were used to support the participant's arms during the evaluation. Participants were seated, resting their arm on one table, the other arm was resting on an adjacent elevation table with the hand hanging over the edge of the table. Before starting the evaluation, the participant's hand was annotated with 9 dots in the same way as in the *touch accuracy* evaluation.

The measurements were recorded through a five step procedure: (1) The elevation table was lowered until the finger touched the BOH; (2) The BOH and finger were aligned to touch a particular dot; (3) The table was elevated to a non-ambiguous hover state; (4) The finger was then lowered in small steps ($<$1 mm) through the area of ambiguity and stopped when a touch state was obtained for more than 2 seconds; and (5) The finger was then elevated in similar steps until a hover state was obtained for more than 2 seconds. Measurements were recorded at the end of step (4) and (5). The procedure was repeated for all of the nine dots for both fingers. A total of 72 dots were recorded.

Results: All measurements of non-ambiguous touch and hover states fell withing an interval between 1 mm and 10 mm. This indicates that our algorithm is capable of reliably detecting a touch state at 1 mm distance from the BOH. Further, it reliably detects hovering when the finger is 10 mm away from the surface. Compared to previous state of the art [48], which reported their interval to be between 10 mm and 20 mm, this is a notable improvement.

9.5.4 Random Forest Classification Accuracy

Additionally, we also report accuracy of using random forests for classification. When training our classification forests, we adopted a rigorous cross-validation procedure to tune the parameters. For the best parameters chosen the per-pixel classification accuracy was 77% for fingertip detection, and 98.8% for hand segmentation. We note that these numbers are comparable to those reported in [122].

9.6 WatchSense-Enabled Applications

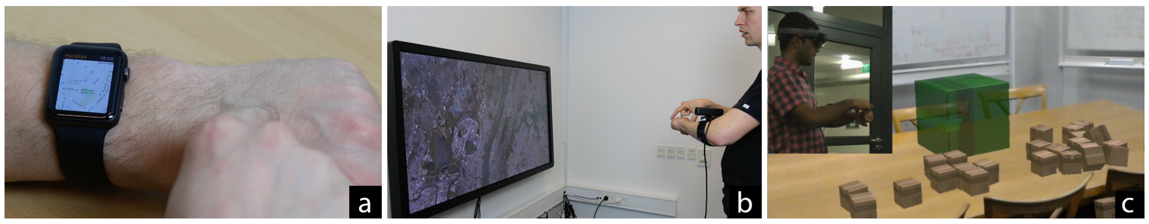


Figure 9.7 Some interactive applications enabled by *WatchSense*. (a) Single-touch input allows for panning a map on a smartwatch. (b) Multitouch allows for zooming/resizing a lens on a large display. (c) Compound touch and mid-air interactions allow for more expressive input for emerging mixed reality devices. Here we show a user interacting with virtual boxes through a HoloLens.

To illustrate the novel input capabilities enabled by *WatchSense*, we built several demonstrator applications. We thus explore how *WatchSense* can support existing multitouch interactions such as pinch-to-zoom, as well as open up completely new opportunities, thanks to finger identification and compound interactions.

To further showcase flexibility, we show our prototype running on different hardware platforms and depth sensors. We also show cross-platform interaction, i.e., *WatchSense* can run on a mobile device but be used for interaction with another surrounding device (e.g., HoloLens), or even several surrounding devices simultaneously.

Music Controller: When the *WatchSense* app runs on any mobile device (running Android), it provides a music player controller feature by default. We use the PicoFlexx camera connected to a mobile device. It enables users to control three functions: (1) adjust volume, (2) change sound track, and (3) toggle music playing. Because of the unique capability of recognizing fingertips, we support the above functions with simple interaction techniques. To adjust the volume, users can touch the BOH with their index finger (*Mid-air* + *Touch*)

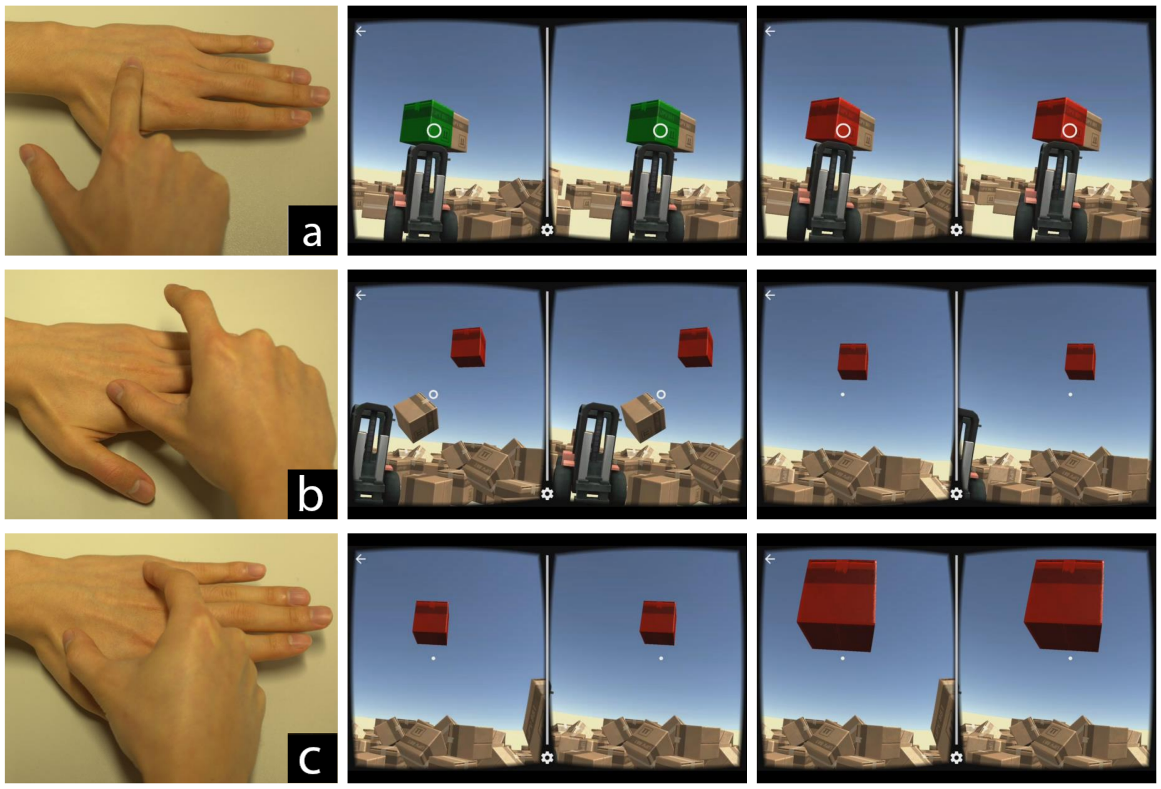


Figure 9.8 *CardboardBoxes* game for VR/AR. (a) User gazes at a box and uses index finger to select it (*Mid-air* + *Touch*). (b) Thumb touching while index finger in mid-air allows translating the selected box freely (*Touch* + *Mid-air*). (c) Both fingers touching while pinching scales the box (*Touch* + *Touch*).

to increase the volume or with their thumb (*Touch* + *Mid-air*) to decrease it. Touching the BOH with both fingers (*Touch* + *Touch*) toggles music playback or pausing. Finally, users can swich to the previous or next tracks by swiping with the index finger either towards or away from the fingers (*Mid-air* + *Touch*).

Virtual/Augmented Reality (VR/AR) Input: We built a game for virtual or augmented reality glasses called *CardboardBoxes*. Users can play with tens of cardboard boxes strewn across the a virtual or real environment. This game showcases the 3D interaction capabilities of *WatchSense*. Users can select a box from the scene by gazing at an object and touching the BOH with their index finger (*Mid-air* + *Touch*, see Figure 9.8). Once selected, boxes can be moved around the scene or scaled. Moving is achieved by a *Touch* + *Mid-air* gesture with the index finger's 3D position relative to the thumb being used for mapping the box's 3D position relative to the observer. Scaling the box can be achieved by pinching on the BOH with both fingers (*Touch* + *Touch*).

In VR mode, tracking runs on a smartphone as a background app. We use the Google Cardboard API to render the game on the same device. In AR mode, tracking can run on any device which relays sensed input to a heads-up display (we run it on a smartphone for mobility). The game is rendered on a HoloLens[3] which allows us to naturally interact with the game as well as the environment around the user. The default HoloLens interaction modality of using free hand gestures to move objects can be fatiguing for users. In contrast, our approach allows users to move objects with only finger movements. Additionally, with a pinch gesture on the BOH users can scale objects—this task is not easily doable with current HoloLens freehand gestures.

Map on a Watch: By combining on-BOH and mid-air input, we created more expressive interactions for a map application than a smartwatch allows. Our solution uses three interactions: (1) touching the BOH using the index finger (*Mid-air* + *Touch*), allowing for single-touch interactions much like on the screen (i.e., dragging the map); (2) likewise, when both fingers touch the back of the hand (*Touch* + *Touch*), users zoom in or out; (3) when only the thumb is touching (*Touch* + *Mid-air*), a pop-up menu is shown allowing for switching display modes (*map*, *transit*, and *satellite*). Selection is performed by changing the distance between the two fingers through moving the index finger.

Image Exploration on a Large Display: This application maps inputs to a large display showing a satellite image for an exploration task (see Figure 9.7 (b)). There are four modes of interaction: (1) using only the index finger (*Mid-air* + *Touch*) on the BOH allows for dragging the entire image across the display; (2) when touching the BOH with the thumb only (*Touch* + *Mid-air*), a fisheye lens is shown, which can be moved by moving the thumb; (3) touching the BOH with both thumb and index finger (*Touch* + *Touch*) allows for resizing the lens (unlike zooming the map in the watch application); and (4) having the thumb touch the BOH with the index finger in mid-air allows for changing the zoom level within the lens. Again, this solution is more expressive than what would be possible with touch or mid-air alone.

Controlling a Game: *WatchSense* enables also joystick-like input for a wearable device. This is achieved by touching the BOH with thumb and controlling *pitch* (forward/backward tilt of the hand) and *roll* (left/right tilt of the hand). Figure 9.9 shows this *Touch* + *Mid-air* gesture: the interacting hand forms a *joystick* with the thumb as base and the index finger acting as the top. Our game is a space game involving space navigation and shooting other spaceships and asteroids. Three interactions were implemented: (1) *pitch* controls the forward and backward motion of the spacecraft (up/down on the display); (2) *roll* controls the left and right movement of the spacecraft; and (3) the index finger in air controls firing

[3]https://www.microsoft.com/microsoft-hololens

by quickly moving it down and up again in a trigger-like fashion. To our knowledge, this is a novel way of using the hand as a joystick to control a game with a wearable device.

9.7 Discussion

WatchSense is a solution for fast and expressive multitouch and mid-air input on and above the BOH. In particular, it supports new combinations of the two types of input. To make interaction even more fluid and expressive future work will need to address some limitations of *WatchSense*.

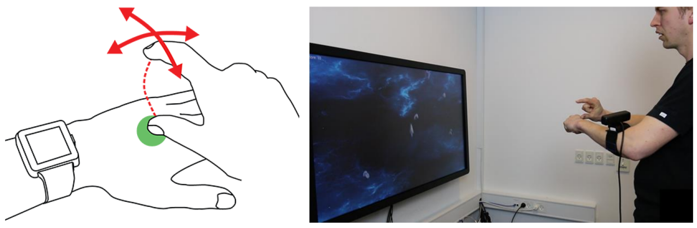


Figure 9.9 *WatchSense* allows for joystick-like directional control for gaming. Here a 2D spacecraft shooting game is controlled by compound *Touch* + *Mid-air* gesture. The spaceship fires when the index finger is quickly moved towards the BOH and up again.

First, our prototype has to be worn about 20 cm from the wrist due to depth sensor limitations. Future work should explore depth sensing technology for near-surface sensing. Second, although our algorithm achieves better accuracy for fingertip position than previous work, there is room for improvement. Third, it might be possible to extend our algorithm to support input on and above arbitrary surfaces to broaden potential applications. Future work should also look at hand-object interactions to further explore new interaction avenues.

Finally, the interactions that we propose represent only a subset of the possible interaction space. More research is needed to explore how detection of fingertips, their identities, and touch can enable richer input. We restricted out interactions to linear gestures. Future work should also look at more complex gesture patterns that can be tracked by *WatchSense*. This introduces new challenges such as gesture segmentation [28]. Moreover, further work is needed to see if the contributions of this chapter can be combined with full hand pose estimation techniques presented in Chapters 5–6. This would lead to a much broader and expressive set of input techniques for gesture-based interaction.

9.8 Conclusion

This chapter has contributed to methods for extending the interaction capabilities of small form factor wearable computers. *WatchSense* allows extending the input space on wearable devices to the back of the hand and the space above it. The BOH offers a natural and always-available surface for input, which now can be utilized in different postures and even if the hand is carrying an object. Because *WatchSense* estimates fingertip locations, identities, and touch positions, it can support interactions that were previously not possible. In particular, continuous interaction and familiar multitouch gestures like pinch can be carried out on the back of the hand and combined with mid-air gestures thus increasing the expresiveness of input. Finger identification adds the possibility to trigger events and map controls to index and thumb separately.

Tracking fingertips, their identities, and touch on the BOH in real-time from the viewpoint of a depth camera in a smartwatch is a hard computer vision problem. Our algorithm has shown advances by using a combination of machine learning, image processing, and robust estimators. It tackles the issues posed by oblique viewpoint, occlusions, fast motions, and fingertip ambiguity.

Finally, we demonstrated that the back of the hand may not only serve as an input device for wearable computers, but could show potential as an input device for other displays and devices in the users' surroundings. We have shown that contact with skin can be detected reasonably well with a close-range depth sensing approach. This, combined with finger identification and real-time tracking, may enable using areas of skin and rigid surfaces flexibly to control near-by devices.

Chapter 10

Conclusions and Future Work

This thesis has presented solutions to several important and unsolved problems in computer vision-based hand tracking and gesture-based computer input. Markerless tracking of hands is a hard problem due to fast motions, occlusions, uniform skin color, and high dimensionality of the optimization task. We presented some of the first techniques that allow tracking of the full 3D articulated pose of the hands, even together with objects, under different sensor setups such as multiple RGB cameras or a single depth sensor. Designing gesture-based input driven by markerless hand tracking is an equally challenging problem because of the large design space and human factors. We have showed through qualitative and quantitative results, new optimization frameworks, user studies, evaluations, and interactive applications (e.g., 3D navigation) that gesture-based input is a viable means of input.

In Part I, we started by solving a relatively less difficult (due to fewer occlusions) version of the hand tracking problem with multiple cameras and interactive runtime requirements (Chapter 3). Results from this chapter indicated that real-time multi-camera tracking with higher accuracy was attainable with further work. We demonstrated this in Chapter 7, which showed real-time tracking at 50 FPS. We further improved accuracy in the multi-view setting, as we describe in Chapter 4, by using a new Sum of Anisotropic Gaussians representation while still running at over 25 FPS. In Chapter 5 we addressed the harder problem of tracking from a single depth camera within a Gaussian mixture framework and yet attained real-time performance without using additional hardware. This allowed us to track hands, for the first time, from a single moving depth sensor such as those in head-mounted cameras. Our approach could also support fine grained motions and poses with self-occlusions. Finally, in Chapter 6 we addressed the much harder problem of joint hand and object tracking in real-time. Our approach is, to our knowledge, the first to simultaneously track both a scene object and a hand interacting with it from a single depth camera.

In Part II, we presented methods for gesture-based input enabled by markerless hand tracking. In Chapter 7, we investigated the use of elicitation studies for continuous gesture input. We showed many examples of 3D navigation tasks where users could interact with their fingers and yet achieve similar performance to traditional input devices like the mouse. Interaction techniques driven by elicitation studies have their limits, however. In Chapter 8, we showed one of the first methods for computational gesture design. To this end, we collected data from users on finger dexterity. We showed that this data can be used in a novel combinatorial optimization framework to automatically design appropriate gestures mid-air text entry. Finally, we showed that restricting tracking to only fingertips can still enable new expressive forms of input for small form factor devices such as smartwatches (Chapter 9). *WatchSense* enabled interactions which were previously not possible by simultaneously supporting both mid-air and multitouch gestures on the back of the hand.

10.1 Future Directions and Outlook

This thesis has advanced the state of the art in both computer vision-based hand tracking and gesture-based input. There are several directions for future work that are natural progressions of the work presented in this thesis. We summarize some of these directions below.

10.1.1 Single RGB Camera

In Chapter 5, we showed how a single *depth* camera can be used for real-time hand tracking. Depth cameras, however, may be harder to integrate into small devices, consume a lot of power, and may not work in general outdoor conditions. Color cameras, on the other hand, are ubiquitous on devices like smartphones and tablets. Tracking 3D hand pose with only color cameras has potentially large benefits but is a hard problem due to varying lighting conditions. Recently, convolutional neural networks (CNNs) have achieved surprisingly good results for full body 2D pose estimation in RGB images. We believe that extending this to 3D pose estimation is an interesting direction for future work.

10.1.2 Tracking in Cluttered Scenes from Different Viewpoints

We showed in Chapter 5 that tracking a single hand in scenes with relatively little background clutter works well. When there is scene clutter, however, our approach would fail. The problem of segmenting hand from background or clutter is difficult due to lighting changes, and similar local shape. In order for hand tracking to be used for everyday interaction, it is essential to track hands in cluttered scenes. Furthermore, the negative effect of clutter is compounded when different viewpoints are used e.g., egocentric viewpoints from head- or body-mounted cameras. We believe that machine learning-based approaches in conjunction with model-based generative tracking can partially solve the segmentation and viewpoint problems along with strong priors about hand pose.

10.1.3 Tracking Hands in Conjunction with Full Body

In this thesis, we showed hand tracking together with and without different objects they are manipulating. Tracking hands in conjunction with full body pose could have numerous medical, and virtual reality (VR) applications. The challenges in achieving this include the small size of the hand in relation to the body, occlusions, and limited data for the hands. We think that advancements in depth sensing technology and improvements to existing algorithms will be required for this to work.

10.1.4 Tracking Multiple Strongly Interacting Hands

In everyday life, humans greet each other with gestures such as a handshake, a clap, or a *high five*. These gestures involve the interaction of two or more hands belonging to the same or different persons. The ability to track this interaction opens a range of applications in human activity and gesture recognition. As can be expected, occlusions can cause major challenges to this problem. While there has been some work in this direction [97], tracking interacting hands under real-time constraints remains an unsolved problem. We believe that strong priors for how hands interact will be critical to solving this problem.

10.1.5 Computational Gesture Design for Continuous Input

In Chapter 8, we showed how computational gesture design can be used for a discrete input task (i.e., text entry). Our model of hand dexterity, however, can also be used for continuous gesture input design. It would be interesting to create new interaction techniques based on our model of hand movement performance and evaluate them. There are many challenges to doing this including making sure that gestures are ergonomic, memorable, and easy to perform.

10.1.6 Application Specific Tracking for Interaction

As we showed in Chapter 9 limiting tracking to only a subset of the full hand pose can have many benefits. Firstly, it reduces the computational complexity of the tracking algorithm enabling tracking to run on small form factor devices. Secondly, it enables a richer set of novel input for specific applications such as input to smartwatches. We think that many such new expressive interactions can be enabled by limiting the space of tracking parameters. Other examples include *near surface* finger tracking which could turn any surface into a touchscreen or on- and above-skin input from different parts of the body.

10.1.7 Tangible Computer Input

The concept of coupling the interaction bits with everyday physical objects and architectural surfaces was proposed by *TangibleBits* [56]. When algorithms for tracking hands together with objects reach a sufficient level of accuracy it might be possible to truly realize the vision of *TangibleBits*. It could be possible to use everyday objects as input devices. Computers would automatically be able to digitize what we write on paper. A smartphone could have no touch sensitive sensors and instead rely completely on finger and surface tracking.

In this thesis, we have made contributions to solving some important problems in enabling gesture-based input for human–computer interaction. We believe that this will constitute an important foundation for future work including the directions mentioned above. The success of next generation computers including smartwatches, and smartglasses depend on efficient means of input. We believe that hands and their interaction with their surroundings will play an important role in the future of ubiquitous computers.

Bibliography

[1] Leap Motion. https://www.leapmotion.com/.

[2] NimbleVR. http://nimblevr.com/.

[3] Peter Ahrendt and Peter Ahrendt. The multivariate gaussian probability distribution. Technical report, 2005.

[4] C. Amma, M. Georgi, and T. Schultz. Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3D-space handwriting with inertial sensors. In *Proc. ISWC*, pages 52–59, 2012.

[5] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. In *Proc. CVPR*, volume 2, pages II – 432–9 vol.2, June 2003.

[6] Ayoub B. Ayoub. The central conic sections revisited. *Mathematics Magazine*, 66(5):322–325, December 1993.

[7] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel, and C. Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Proc. ICCV*, pages 1092 –1099, November 2011.

[8] Ishrat Badami, Jörg Stückler, and Sven Behnke. Depth-Enhanced Hough Forests for Object-Class Detection and Continuous Pose Estimation. In *Workshop on Semantic Perception, Mapping and Exploration (SPME)*, 2013.

[9] Gilles Bailly, Jörg Müller, and Eric Lecolinet. Design and evaluation of finger-count interaction: Combining multitouch gestures and menus. *Int. J. Hum.-Comput. Stud.*, pages 673–689, 2012.

[10] Gilles Bailly, Jörg Müller, Michael Rohs, Daniel Wigdor, and Sven Kratz. ShoeSense: a new perspective on gestural interaction and wearable applications. In *Proc. CHI*, pages 1239–1248, 2012.

[11] Gilles Bailly, Robert Walter, Jörg Müller, Tongyan Ning, and Eric Lecolinet. Comparing free hand menu techniques for distant displays using linear, marking and finger-count menus. In *Proc. of INTERACT 2011*, number 6947 in Lecture Notes in Computer Science, pages 248–262. January 2011.

[12] Ravin Balakrishnan and I Scott MacKenzie. Performance differences in the fingers, wrist, and forearm in computer input control. In *Proc. CHI*, pages 303–310, 1997.

[13] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. Motion capture of hands in action using discriminative salient points. In *Proc. ECCV*, volume 7577, pages 640–653. Springer Berlin / Heidelberg, 2012.

[14] Thomas Baudel and Michel Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Commun. ACM*, 36(7):28–35, July 1993.

[15] Yannick Bernaerts, Matthias Druwé, Sebastiaan Steensels, Jo Vermeulen, and Johannes Schöning. The office smartwatch: Development and design of a smartwatch app to digitally augment interactions in an office environment. In *Proceedings of the 2014 Companion Publication on Designing Interactive Systems*, DIS Companion '14, pages 41–44, New York, NY, USA, 2014. ACM.

[16] A. Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhya: The Indian Journal of Statistics (1933-1960)*, 7(4):401–406, July 1946.

[17] Xiaojun Bi, Barton A Smith, and Shumin Zhai. Multilingual touchscreen keyboard design and optimization. *Human–Computer Interaction*, 27(4):352–382, 2012.

[18] Richard A. Bolt. "put-that-there": Voice and gesture at the graphics interface. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '80, pages 262–270, New York, NY, USA, 1980. ACM.

[19] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola, and Ivan Poupyrev. An introduction to 3-d user interface design. *Presence: Teleoperators and Virtual Environments*, 10(1):96–108, February 2001.

[20] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for 3d hand tracking. In *Proc. Intl. Conf. Automatic Face and Gesture Recognition*, pages 675–680, 2004.

[21] Alex Butler, Shahram Izadi, and Steve Hodges. Sidesight: multi-touch interaction around small devices. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 201–204. ACM, 2008.

[22] Dylan Campbell and Lars Petersson. Gogma: Globally-optimal gaussian mixture alignment. *arXiv preprint arXiv:1603.00150*, 2016.

[23] J. B. Carter and E. W. Banister. Musculoskeletal problems in VDT work: a review. *Ergonomics*, 37(10):1623–1648, 1994. PMID: 7957019.

[24] Tom Carter, Sue Ann Seah, Benjamin Long, Bruce Drinkwater, and Sriram Subramanian. UltraHaptics: multi-point mid-air haptic feedback for touch surfaces. In *Proc. UIST*, pages 505–514, 2013.

[25] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2527–2530, New York, NY, USA, 2012. ACM.

[26] Liwei Chan, Rong-Hao Liang, Ming-Chang Tsai, Kai-Yin Cheng, Chao-Huai Su, Mike Y. Chen, Wen-Huang Cheng, and Bing-Yu Chen. Fingerpad: Private and subtle interaction using fingertips. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 255–260, New York, NY, USA, 2013. ACM.

[27] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. utrack: 3d input using two magnetic sensors. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 237–244, New York, NY, USA, 2013. ACM.

[28] Xiang 'Anthony' Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott E. Hudson. Air+touch: Interweaving touch &#38; in-air gestures. In *Proc. of UIST '14*, pages 519–525, New York, NY, USA. ACM.

[29] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[30] Antonio Criminisi and Jamie Shotton. *Decision forests for computer vision and medical image analysis*. Springer, 2013.

[31] Andrew Crossan, John Williamson, Stephen Brewster, and Rod Murray-Smith. Wrist rotation for interaction in mobile contexts. In *Proc. MobileHCI*, pages 435–438, 2008.

[32] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, volume 1, pages 886–893. IEEE, 2005.

[33] M. de La Gorce, D.J. Fleet, and N. Paragios. Model-Based 3D Hand Pose Estimation from Monocular Video. *IEEE TPAMI*, 33(9):1793–1805, 2011.

[34] Mark Dunlop and John Levine. Multidimensional Pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking. In *Proc. CHI*, pages 2669–2678, 2012.

[35] David Eberly. Perspective projection of an ellipsoid. http://www.geometrictools.com/, 1999.

[36] Douglas C Engelbart and William K English. A research center for augmenting human intellect. In *Proc. of Fall Joint Computer Conference*, pages 395–410, 1968.

[37] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *CVIU*, 108(1-2):52–73, October 2007.

[38] Sean Ryan Fanello, Cem Keskin, Shahram Izadi, Pushmeet Kohli, David Kim, David Sweeney, Antonio Criminisi, Jamie Shotton, Sing Bing Kang, and Tim Paek. Learning to be a depth camera for close-range human capture and interaction. *ACM TOG*, 33(4):86:1–86:11.

[39] David A Forsyth and Jean Ponce. *Computer vision: a modern approach.* Prentice Hall Professional Technical Reference, 2002.

[40] Kentaro Fukuchi, Toshiki Sato, Haruko Mamiya, and Hideki Koike. Pac-pac: pinching gesture recognition for tabletop entertainment system. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI 2010, pages 267–273, 2010.

[41] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real-time human pose tracking from range data. In *Proc. ECCV*, volume 7577, pages 738–751. Berlin, Heidelberg, 2012.

[42] R. Girshick, J. Shotton, P. Kohli, Antonio Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV 2011*, pages 415–422, 2011.

[43] Sean Gustafson, Daniel Bierwirth, and Patrick Baudisch. Imaginary interfaces: Spatial interaction with empty hands and without visual feedback. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 3–12, New York, NY, USA, 2010. ACM.

[44] Sean Gustafson, Christian Holz, and Patrick Baudisch. Imaginary Phone: Learning Imaginary Interfaces by Transferring Spatial Memory from a Familiar Device. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 283–292, New York, NY, USA, 2011. ACM.

[45] Charlotte Häger-Ross and Marc H Schieber. Quantifying the independence of human finger movements: comparisons of digits, hands, and movement frequencies. *J. Neuroscience*, 20(22):8542–8550, 2000.

[46] H. Hamer, J. Gall, T. Weise, and L. Van Gool. An object-dependent hand pose prior from sparse training data. In *Proc. CVPR*, pages 671–678, 2010.

[47] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *Proc. ICCV*, pages 1475–1482, 2009.

[48] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. Omnitouch: Wearable multitouch interaction everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 441–450, New York, NY, USA, 2011. ACM.

[49] Chris Harrison and Scott E. Hudson. Abracadabra: Wireless, high-precision, and unpowered finger input for very small mobile devices. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 121–124, New York, NY, USA, 2009. ACM.

[50] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge Univ Press, 2000.

[51] Khalad Hasan, David Ahlström, and Pourang Irani. Ad-binning: Leveraging around device space for storing, browsing and retrieving mobile device content. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 899–908, New York, NY, USA, 2013. ACM.

[52] T. Heap and David Hogg. Towards 3d hand tracking using a deformable model. In *Proc. Intl. Conf. Automatic Face and Gesture Recognition*, pages 140–145, Oct 1996.

[53] Otmar Hilliges, Shahram Izadi, Andrew D. Wilson, Steve Hodges, Armando Garcia-Mendoza, and Andreas Butz. Interactions in the air: adding further depth to interactive tabletops. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 139–148, 2009.

[54] Eve Hoggan, Stephen A Brewster, and Jody Johnston. Investigating the effectiveness of tactile feedback for mobile touchscreens. In *Proc. CHI*, pages 1573–1582, 2008.

[55] Steven Houben and Nicolai Marquardt. Watchconnect: A toolkit for prototyping smartwatch-centric cross-device applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1247–1256, New York, NY, USA, 2015. ACM.

[56] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, pages 234–241, New York, NY, USA, 1997. ACM.

[57] Bing Jian and Baba C Vemuri. Robust point set registration using gaussian mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1633–1645, 2011.

[58] Brett Jones, Rajinder Sodhi, David Forsyth, Brian Bailey, and Giuliano Maciocci. Around device interaction for multiscale navigation. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '12, pages 83–92, New York, NY, USA, 2012. ACM.

[59] Lynette A. Jones and Susan J. Lederman. *Human Hand Function*. Oxford University Press, 1 edition, 2006.

[60] T Kaminski and AM Gentile. Joint control strategies and hand trajectories in multi-joint pointing movements. *Journal of Motor Behavior*, 18(3):261–278, 1986.

[61] Maria Karam. *PhD Thesis: A framework for research and design of gesture-based human-computer interactions*. phd, University of Southampton, October 2006.

[62] C. Keskin, F. Kirac, Y.E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *Proc. of ICCV Workshops 2011*, pages 1228–1234.

[63] Cem Keskin, Furkan Kirac, Yunus Emre Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proc. ECCV*, pages 852–863. Springer Berlin Heidelberg, 2012.

[64] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, pages 119–137. Springer, 2013.

[65] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. Digits: Freehand 3d interactions anywhere using a wrist-worn gloveless sensor. In *UIST '12*, pages 167–176, New York, NY, USA, 2012. ACM.

[66] Jungsoo Kim, Jiasheng He, Kent Lyons, and Thad Starner. The gesture watch: A wireless contact-free gesture based wrist interface. In *Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers*, ISWC '07, pages 1–8, Washington, DC, USA, 2007. IEEE Computer Society.

[67] G.V. Kondraske. An angular motion Fitt's law for human performance modeling and prediction. In *Proc. IEEE EMBS*, pages 307–308 vol.1, November 1994.

[68] Sven Kratz and Michael Rohs. Hoverflow: Expanding the design space of around-device interaction. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 4:1–4:8, New York, NY, USA, 2009. ACM.

[69] Jaka Krivic and Franc Solina. Contour based superquadric tracking. Lecture Notes in Computer Science, pages 1180–1186. 2003.

[70] Daniyar Kurmankhojayev, Nils Hasler, and Christian Theobalt. Monocular pose capture with a depth camera using a sums-of-gaussians body model. In *Pattern Recognition*, number 8142 in LNCS, pages 415–424. January 2013.

[71] Daniyar Kurmankhojayev, Nils Hasler, and Christian Theobalt. Monocular pose capture with a depth camera using a sums-of-gaussians body model. In *Pattern Recognition*, number 8142 in LNCS, pages 415–424, January 2013.

[72] N. Kyriazis and A. Argyros. Physically Plausible 3d Scene Tracking: The Single Actor Hypothesis. In *Proc. IEEE CVPR*, pages 9–16, 2013.

[73] N. Kyriazis and A. Argyros. Scalable 3d Tracking of Multiple Interacting Objects. In *Proc. IEEE CVPR*, pages 3430–3437, June 2014.

[74] Gary D Langolf, Don B Chaffin, and James A Foulke. An investigation of Fitts' law using a wide range of movement amplitudes. *Journal of Motor Behavior*, 8(2):113–128, 1976.

[75] Jinha Lee, Alex Olwal, Hiroshi Ishii, and Cati Boulanger. SpaceTop: integrating 2d and spatial 3d interactions in a see-through desktop environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI 2013, pages 189–192, 2013.

[76] Seungyon Claire Lee, Bohao Li, and Thad Starner. Airtouch: Synchronizing in-air hand gesture and on-body tactile feedback to augment mobile gesture interaction. In *Proceedings of the 2011 15th Annual International Symposium on Wearable Computers*, ISWC '11, pages 3–10, Washington, DC, USA, 2011. IEEE Computer Society.

[77] Jhe-Wei Lin, Chiuan Wang, Yi Yao Huang, Kuan-Ting Chou, Hsuan-Yu Chen, Wei-Luan Tseng, and Mike Y. Chen. Backhand: Sensing hand gestures via back of the hand. In *UIST '15*, pages 557–564, New York, NY, USA, 2015. ACM.

[78] John Lin, Ying Wu, and T.S. Huang. Modeling the constraints of human hand motion. In *Proc. HUMO*, pages 121 –126, 2000.

[79] Kent Lyons, Thad Starner, and Brian Gane. Experimental evaluations of the twiddler one-handed chording mobile keyboard. *Human-Computer Interaction*, 2006.

[80] John MacCormick and Michael Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proc. ECCV*, number 1843, pages 3–19. January 2000.

[81] I Scott MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7(1):91–139, 1992.

[82] Shahzad Malik, Abhishek Ranjan, and Ravin Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *Proc. UIST*, pages 43–52, 2005.

[83] Zhi-Hong Mao, Heung-No Lee, R.J. Sclabassi, and Mingui Sun. Information capacity of the thumb and the index finger in communication. *IEEE Trans. Biomedical Engineering*, 56(5):1535 –1545, May 2009.

[84] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbaek. Vulture: A mid-air word-gesture keyboard. In *Proc. CHI*, pages 1073–1082, 2014.

[85] Simon Mayer and Gábor Sörös. User interface beaming - seamless interaction with smart things using personal wearable computers. In *Proceedings of the 11th International Conference on Wearable and Implantable Body Sensor Networks (BSN 2014)*, pages 46–49, Zurich, Switzerland, June 2014.

[86] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3D skeletal hand tracking. In *Proc. i3D*, pages 184–184, 2013.

[87] Suvobrata Mitra, Polemnia G Amazeen, and Michael T Turvey. Intermediate motor learning as decreasing active (dynamical) degrees of freedom. *Human Movement Science*, 17(1):17–65, 1998.

[88] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *CVIU*, 104(2–3):90–126, November 2006.

[89] Meredith Ringel Morris, Andreea Danielescu, Steven Drucker, Danyel Fisher, Bongshin Lee, m. c. schraefel, and Jacob O Wobbrock. Reducing legacy bias in gesture elicitation studies. *interactions*, 21(3):40–45, 2014.

[90] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., 1st edition, 1994.

[91] Kei Nakatsuma, Hiroyuki Shinoda, Yasutoshi Makino, Katsunari Sato, and Takashi Maeno. Touch Interface on Back of the Hand. In *ACM SIGGRAPH 2011 Emerging Technologies*, SIGGRAPH '11, pages 19:1–19:1, New York, NY, USA, 2011. ACM.

[92] Tao Ni. *A Framework of Freehand Gesture Interaction: Techniques, Guidelines, and Applications*. PhD thesis, VirginiaTech, September 2011.

[93] Tao Ni, Doug Bowman, and Chris North. AirStroke: Bringing unistroke text entry to freehand gesture interfaces. In *Proc. CHI*, pages 2473–2476, 2011.

[94] Onno A. van Nierop, Aadjan van der Helm, Kees J. Overbeeke, and Tom J. P. Djajadiningrat. A natural human hand model. *Visual Comput*, 24(1):31–44, January 2008.

[95] Ian Oakley, John Sunwoo, and Il-Yeon Cho. Pointing with fingers, hands and arms for wearable computing. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 3255–3260, 2008.

[96] 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 *Proc. ICCV*, pages 2088–2095, 2011.

[97] I. Oikonomidis, N. Kyriazis, and A.A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *Proc. of CVPR 2012*, pages 1862–1869, June 2012.

[98] I. Oikonomidis, M.I.A. Lourakis, and A.A. Argyros. Evolutionary quasi-random search for hand articulations tracking. In *Proc. of CVPR 2014*, pages 3422–3429.

[99] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proc. BMVC*, pages 101.1–101.11, 2011.

[100] Paschalis Panteleris, Nikolaos Kyriazis, and Antonis A. Argyros. 3d tracking of human hands in interaction with unknown objects. In *Proc. BMVC*, 2015.

[101] Jerome Pasquero, Scott J. Stobbe, and Noel Stonehouse. A haptic wristwatch for eyes-free interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3257–3266, New York, NY, USA, 2011. ACM.

[102] Simon T. Perrault, Eric Luecolinet, James Eagan, and Yves Guiard. Watchit: Simple Gestures and Eyes-free Interaction for Wristwatches and Bracelets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1451–1460, New York, NY, USA, 2013. ACM.

[103] Tu-Hoa Pham, Abderrahmane Kheddar, Ammar Qammaz, and Antonis A. Argyros. Towards Force Sensing From Vision: Observing Hand-Object Interactions to Infer Manipulation Forces. In *Proc. IEEE CVPR*, 2015.

[104] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. User-defined gestures for augmented reality. In *INTERACT 2013*, number 8118, pages 282–299. January 2013.

[105] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-time identification and localization of body parts from depth images. In *Proc. ICRA*, pages 3108–3113, May.

[106] R. Plankers and P. Fua. Articulated soft objects for multiview shape and motion capture. *IEEE TPAMI*, 25(9):1182—1187, 2003.

[107] Ronald Poppe. Vision-based human motion analysis: An overview. *CVIU*, 108(1–2):4–18, October 2007.

[108] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Proc. CVPR*, 2014.

[109] Mahfuz Rahman, Sean Gustafson, Pourang Irani, and Sriram Subramanian. Tilt techniques: investigating the dexterity of wrist-based input. In *Proc. CHI*, pages 1943–1952, 2009.

[110] Singiresu S Rao and SS Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

[111] James Rehg and Takeo Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. In *Proc. ECCV*, volume 801, pages 35–46. Springer Berlin / Heidelberg, 1994.

[112] Susanna Ricco and Carlo Tomasi. Fingerspelling recognition through classification of letter-to-letter transitions. In *Proc. ACCV*, pages 214–225. 2010.

[113] Julie Rico and Stephen Brewster. Usable gestures for mobile interfaces: evaluating social acceptability. In *Proc. CHI*, pages 887–896, 2010.

[114] J. Romero, H. Kjellstrom, and D. Kragic. Hands in action: real-time 3D reconstruction of hands in interaction with objects. In *Proc. ICRA*, pages 458–463, 2010.

[115] Romer Rosales and Stan Sclaroff. Combining generative and discriminative models in a framework for articulated pose estimation. *International Journal of Computer Vision*, pages 251–276 v.67, 2006.

[116] David A Rosenbaum. *Human motor control*. Academic Press, 2009.

[117] Marc H Schieber. Individuated finger movements of rhesus monkeys: a means of quantifying the independence of the digits. *J Neurophysiology*, 65(6):1381–91, 1991.

[118] Robert Seibel. Data entry through chord, parallel entry devices. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 6(2):189–192, 1964.

[119] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim Christoph Rhemann Ido Leichter, Alon Vinnikov Yichen Wei, Daniel Freedman Pushmeet Kohli Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proc. CHI*, volume 8, 2015.

[120] Garth Shoemaker, Leah Findlater, Jessica Q. Dawson, and Kellogg S. Booth. Mid-air text input techniques for very large wall displays. In *Proc. GI*, pages 231–238. Canadian Information Processing Society, 2009.

[121] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proc. CVPR*, pages 1297 –1304, June 2011.

[122] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[123] Edgar Simo Serra. *Kinematic Model of the Hand using Computer Vision*. PhD thesis, Institut de Robotica i Informatica Industrial, 2011.

[124] Jie Song, Fabrizio Pece, Gábor Sörös, Marion Koelle, and Otmar Hilliges. Joint estimation of 3d hand position and gestures from monocular video for mobile interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3657–3660. ACM, 2015.

[125] R. William Soukoreff and I. Scott MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *Intl. J. of Human-Computer Studies*, 61(6):751–789, December 2004.

[126] Srinath Sridhar, Gilles Bailly, Elias Heydrich, Antti Oulasvirta, and Christian Theobalt. Technical report, Max-Planck-Institut für Informatik, Saarbrücken.

[127] Srinath Sridhar, Anna Maria Feit, Christian Theobalt, and Antti Oulasvirta. Investigating the dexterity of multi-finger input for mid-air text entry. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3643–3652. ACM, 2015.

[128] Srinath Sridhar, Anders Markussen, Antti Oulasvirta, Christian Theobalt, and Sebastian Boring. On- and above-skin input sensing through a wearable depth sensor. Research Report MPI-I-2016-4-003, Max-Planck-Institut für Informatik, Saarbrücken, October 2016.

[129] Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

[130] Srinath Sridhar, Franziska Mueller, Michael Zollhoefer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. In *Proceedings of European Conference on Computer Vision (ECCV)*.

[131] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. Interactive markerless articulated hand motion tracking using RGB and depth data. In *Proc. of ICCV 2013*, pages 2456–2463.

[132] Srinath Sridhar, Helge Rhodin, Hans-Peter Seidel, Antti Oulasvirta, and Christian Theobalt. Real-time hand tracking using a sum of anisotropic Gaussians model. In *3DV 2014, International Conference on 3D Vision*, Tokyo, Japan, December 2014. IEEE Computer Society.

[133] B. Stenger, A. Thayananthan, P. H S Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. 28(9):1372–1384, 2006.

[134] Bjoern Stenger, Paulo RS Mendonça, and Roberto Cipolla. Model-based 3d tracking of an articulated hand. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–310. IEEE, 2001.

[135] C. Stoll, N. Hasler, J. Gall, H. Seidel, and C. Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In *Proc. ICCV*, pages 951 –958, November 2011.

[136] Paul Strohmeier. Diy ir sensors for augmenting objects and human skin. In *Proc. of Augmented Human International Conference '15*, pages 181–182, New York, NY, USA. ACM.

[137] D.J. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, 1994.

[138] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *Proc. IEEE CVPR*, 2015.

[139] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust Articulated-ICP for Real-Time Hand Tracking. *Computer Graphics Forum (Proc. of SGP)*, 34(5), 2015.

[140] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proc. of CVPR 2014*.

[141] Danhang Tang, Jonathan Taylor, and Tae-kyun Kim. Opening the Black Box : Hierarchical Sampling Optimization for Estimating Human Hand Pose. In *Proc. IEEE ICCV*, 2015.

[142] Danhang Tang, Tsz-Ho Yu, and Tae-Kyun Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Proc. of ICCV 2013*.

[143] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-Class Hough Forests for 3d Object Detection and Pose Estimation. In *Proc. ECCV*, pages 462–477. 2014.

[144] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM TOG*, 33(5):169:1–169:10, September 2014.

[145] Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. Capturing hands in action using discriminative salient points and physics simulation. *IJCV*, 2016.

[146] Dimitrios Tzionas and Juergen Gall. 3D Object Reconstruction from Hand-Object Interactions. In *Proc. IEEE ICCV*, 2015.

[147] Dimitrios Tzionas, Abhilash Srikantha, Pablo Aponte, and Juergen Gall. Capturing Hand Motion with an RGB-D Sensor, Fusing a Generative Model with Salient Points. In *Proc. GCPR*, 2014.

[148] K. Vertanen and P. O. Kristensson. A versatile dataset for text entry evaluations based on genuine mobile emails. In *Proc. of MobileHCI*, pages 295–298, 2011.

[149] Julie Wagner, Eric Lecolinet, and Ted Selker. Multi-finger chords for hand-held tablets: recognizable and memorable. In *Proc. CHI*, pages 2883–2892, 2014.

[150] Cheng-Yao Wang, Wei-Chen Chu, Po-Tsung Chiu, Min-Chieh Hsiu, Yih-Harn Chiang, and Mike Y. Chen. PalmType: Using Palms As Keyboards for Smart Glasses. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 153–160, New York, NY, USA, 2015. ACM.

[151] Cheng-Yao Wang, Min-Chieh Hsiu, Po-Tsung Chiu, Chiao-Hui Chang, Liwei Chan, Bing-Yu Chen, and Mike Y. Chen. PalmGesture: Using Palms As Gesture Interfaces for Eyes-free Input. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 217–226, New York, NY, USA, 2015. ACM.

[152] Robert Wang, Sylvain Paris, and Jovan Popović. 6d hands: Markerless hand-tracking for computer aided design. In *Proc. ACM UIST*, pages 549–558, 2011.

[153] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM TOG*, 28(3):63:1–63:8, July 2009.

[154] Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. Video-based hand manipulation capture through composite motion control. *ACM TOG*, 32(4):43:1–43:14, July 2013.

[155] David Way and Joseph Paradiso. A usability user study concerning free-hand microgesture and wrist-worn sensors. In *Proceedings of the 2014 11th International Conference on Wearable and Implantable Body Sensor Networks*, BSN '14, pages 138–142, Washington, DC, USA, 2014. IEEE Computer Society.

[156] Xiaolin Wei, Peizhao Zhang, and Jinxiang Chai. Accurate realtime full-body motion capture using a single depth camera. *ACM TOG (Proc. SIGGRAPH Asia)*, 31(6), November 2012.

[157] Martin Weigel, Tong Lu, Gilles Bailly, Antti Oulasvirta, Carmel Majidi, and Jürgen Steimle. iskin: Flexible, stretchable and visually customizable on-body touch sensors for mobile computing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 2991–3000, New York, NY, USA, 2015. ACM.

[158] Martin Weigel, Vikram Metha, and Jürgen Steimle. More than touch: Understanding how people use skin as an input surface for mobile computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, 2014. ACM.

[159] Alan Traviss Welford. Fundamentals of skill. 1968.

[160] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv Comput Math*, 4(1):389–396, December 1995.

[161] Andrew D. Wilson. Robust computer vision-based detection of pinching for one and two-handed gesture input. In *Proc. UIST*, pages 255–258, 2006.

[162] Ying Wu and T.S. Huang. View-independent recognition of hand postures. In *Proc. IEEE CVPR*, pages 88–94, 2000.

[163] Ying Wu, J.Y. Lin, and T.S. Huang. Capturing natural hand articulation. In *Proc. ICCV*, pages 426 –432 vol.2, 2001.

[164] Robert Xiao, Gierad Laput, and Chris Harrison. Expanding the input expressivity of smartwatches with mechanical pan, twist, tilt and click. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 193–196, New York, NY, USA, 2014. ACM.

[165] Chi Xu and Li Cheng. Efficient hand pose estimation from a single depth image. In *Proc. of ICCV 2013*.

[166] Chi Xu and Li Cheng. Efficient hand pose estimation from a single depth image. In *Proc. ICCV*, 2013.

[167] Mao Ye, Xianwang Wang, Ruigang Yang, Liu Ren, and M. Pollefeys. Accurate 3D pose estimation from a single depth image. In *Proc. ICCV*, pages 731–738, 2011.

[168] Mao Ye and Ruigang Yang. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2353–2360, June 2014.

[169] Hui-Shyong Yeo, Juyoung Lee, Andrea Bianchi, and Aaron Quigley. Watchmi: Pressure touch, twist and pan gesture input on unmodified smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '16, pages 394–399, New York, NY, USA, 2016. ACM.

[170] Shumin Zhai, Michael Hunter, and Barton A. Smith. The Metropolis Keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Proc. UIST*, pages 119–128, 2000.

[171] Yang Zhang, Junhan Zhou, Gierad Laput, and Chris Harrison. Skintrack: Using the body as an electrical waveguide for continuous finger tracking on the skin. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1491–1503, New York, NY, USA, 2016. ACM.

[172] Thomas G. Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A hand gesture interface device. *SIGCHI Bull.*, 17(SI):189–192, May 1986.

[173] Michael Zollhöfer, Matthias Niessner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. Real-time non-rigid reconstruction using an RGB-D camera. *ACM TOG*, 33(4):156:1–156:12, July 2014.