ROBUST HAND MOTION CAPTURE AND PHYSICS-BASED CONTROL FOR GRASPING IN REAL TIME

A Dissertation

by

JIANJIE ZHANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jinxiang Chai
Committee Members,	John Keyser
	Scott Schaefer
	Ergun Akleman
Head of Department,	Dilma Da Silva

August 2017

Major Subject: Computer Science

ABSTRACT

Hand motion capture technologies are being explored due to high demands in the fields such as video game, virtual reality, sign language recognition, human-computer interaction, and robotics. However, existing systems suffer a few limitations, *e.g.* they are high-cost (expensive capture devices), intrusive (additional wear-on sensors or complex configurations) , and restrictive (limited motion varieties and restricted capture space). This dissertation mainly focus on exploring algorithms and applications for the hand motion capture system that is low-cost, non-intrusive, low-restriction, high-accuracy, and robust.

More specifically, we develop a realtime and fully-automatic hand tracking system using a low-cost depth camera. We first introduce an efficient shape-indexed cascaded pose regressor that directly estimates 3D hand poses from depth images. A unique property of our hand pose regressor is to utilize a low-dimensional parametric hand geometric model to learn 3D shape-indexed features robust to variations in hand shapes, viewpoints and hand poses. We further introduce a hybrid tracking scheme that effectively complements our hand pose regressor with model-based hand tracking. In addition, we develop a rapid 3D hand shape modeling method that uses a small number of depth images to accurately construct a subject-specific skinned mesh model for hand tracking. This step not only automates the whole tracking system but also improves the robustness and accuracy of model-based tracking and hand pose regression.

Additionally, we also propose a physically realistic human grasping synthesis method that is capable to grasp a wide variety of objects. Given an object to be grasped, our method is capable to compute required controls (*e.g.* forces and torques) that advance the simulation to achieve realistic grasping. Our method combines the power of data-driven synthesis and physics-based grasping control. We first introduce a data-driven method to

synthesize a realistic grasping motion from large sets of prerecorded grasping motion data. And then we transform the synthesized kinematic motion to a physically realistic one by utilizing our online physics-based motion control method. In addition, we also provide a performance interface which allows the user to act out before a depth camera to control a virtual object.

ACKNOWLEDGMENTS

First, I would like to express my deepest appreciation to my advisor, Professor Jinxiang Chai. During six years working with him, I obtain great opportunities to participate into fantastic research projects, and improve myself not only on research skills but also on the attitude and dedication towards the research. Without his guidance and persistent support, the dissertation would not have been possible. In addition, I would also like to thank my committee members, Professor John Keyser, Professor Scott Schaefer and Professor Ergun Akleman, for their valuable advice and guidance on my research projects.

Second, I would like to show my gratitude to my colleagues in the motion capture lab, Wenping Zhao, Fuhao Shi, Peizhao Zhang, Peihong Guo, Xiaolin Wei, Jianyuan Min, and Yen-lin Chen. It is an amazing experience working with them during the last six years.

Finally, I would like to thank my family's support during my Ph.D. study. My wife, Xiaoping Liu, tries her best to complete all housework and make me focus on my research. My parents always give their unconditional love, support and encouragement to me at all times.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professor Jinxiang Chai, Professor John Keyser and Professor Scott Schaefer from the Department of Computer Science and Engineering and Professor Ergun Akleman from the Department of Visualization.

The automatic hand shape modeling part (Section II.3) was conducted by Peizhao Zhang.

The kinematic synthesis of human grasping part (Section III.3, Section III.4, and part of Section III.6), and relevant experiments in Chapter III were conducted by Wenping Zhao.

All other work conducted for the dissertation was completed by the student, under the advisement of Professor Jinxiang Chai from the Department of Computer Science and Engineering.

Funding Sources

This work was supported in part by the National Science Foundation under Grants No. IIS-1065384 and IIS-1055046.

Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the National Science Foundation.

TABLE OF CONTENTS

Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Hand motion capture, especially real-time 3D hand tracking, plays an important role in our life, because it can facilitate interactive animation and control of human gestures in movies, video game, virtual reality, head mounted displays (HMD) and human-robot interaction. However, to track a 3D hand gesture is extremely difficult, because it requires to reconstruct both large-range global movement and delicate articulations of the hand.

Decades of research have explored a number of approaches, such as marker-based systems (*e.g. Vicon*[1]), glove-based systems (*e.g. Cyberglove*[2]), inertial systems (*e.g. Noitom*[3]) and image-based systems *e.g. 3Gear*[4]. However, to build an accurate and robust hand tracking system is extremely hard. Marker-based systems, even though with multiple cameras, still often produce ambiguous results due to significant self-occlusion during the hand movement. Glove-based systems and inertial systems are occlusion free but often fail to capture delicate hand articulations accurately. In addition, these three systems are often cumbersome and intrusive, because they require the hand to wear special markers, sensors or suites. Image-based systems offer an appealing solution for the tracking problem, because they require no additional wear-on sensors for the user. Also with the advancement of the depth capture technology, depth cameras are affordable to daily users and play an important role in the hand track problem. Our research mainly focuses on how to track the hand with a single depth camera.

One appealing direction is model-based hand tracking [5, 6, 7], which requires an initial pose for the first frame and then tracks the following poses sequentially by minimizing the inconsistency between the hypothesized poses and the observed image data. However, two main limitations exist for current methods. First, they are not fully automatic, because

they require manually constructing a subject-specific 3D hand model and specifying the initial pose for the first frame. Second, they cannot recover from the tracking failure once the systems get stuck in the local minimum, because they cannot initialize by themselves.

Another appealing direction is appearance-based detection method [8, 9, 10, 11], which retrieves a most probable hand pose from a prebuilt dataset consisting of the image-pose pairs. However, current methods still need manual intervention for the subject-specific modeling. Also, the tracking accuracy is usually worse than that of model-based tracking, and cannot be used in high-accuracy applications, *e.g.* motion synthesis, virtual remote control.

In this dissertation, we propose a hybrid tracking scheme that combines the model-based tracking with a per-frame pose regressor (which belongs to the appearance-based detection). The per-frame pose regressor does not require specific initialization at the first frame and directly estimates poses from depth images in a per-frame framework. However, the result of the regressor is usually less accurate than that of the model-based tracking. Model-based tracking can produce more accurate result but may stuck in the tracking failure and require manual initialization. The appropriate combination of these two methods makes them complementary to each other. Our pose regressor is built upon cascaded pose regression [12, 13, 10], where the object pose is estimated progressively by sequential weak regressors and each weak regressor uses the pose-indexed features that depends on the estimated pose from the previous stage. However, the pose-indexed features do not consider the variations of hand shapes, and then do not perform well when applying to subjects with different hand shapes. With the help of a low-dimensional 3D parametric hand model, we significantly advance the pose-indexed features to the shape-indexed features, which provide more robustness to variations in hand shapes, viewpoints and hand poses. Additionally, to automate the whole tracking process, we also propose a rapid 3D hand modeling method that can reconstruct a subject-specific hand model from

a small number of depth images. The accurate subject-specific hand model will also help improve the robustness of the model-based tracking and the pose regressor.

We demonstrate the power of our approach by testing on a wide range of subjects and tracking a variety of hand gestures in real time. Our method achieves state-of-the-art accuracy in comparison against alternative systems. We also validate our method by evaluating the importance of key components of our 3D hand tracking process.

Besides, we also explore one application of hand tracking on physically realistic grasping synthesis by utilizing prerecorded grasping motion capture data. In real life, the human is capable to grasp a large amount of objects with different properties, *e.g.*shapes, weights, frictions, and spatial orientations. However, to synthesize a physically realistic grasping motion is still a challenge task. An ideal grasping action should include a good planning of the hand reaching to the object, careful selection of contacts between fingers and the object, and adaption to objects with different geometry and dynamic characteristics. Otherwise, the grasping action may looks unnatural, *e.g.* motion jerkiness, unreasonable contacts, finger-object penetration, or unreal hand motion while grasping. Recent efforts [14, 15, 16, 17, 18, 19] have made some significant progress, but an automated and realtime system for grasping a large amount of objects with different geometry and physics properties is still challenging and unresolved. In this dissertation, we propose a novel grasping synthesis algorithm to solve the problem above. Our algorithm takes into account the following aspects:

- Physical realism. A synthesized grasping motion without physics may be judged as unacceptable, because it may include motion jerkiness, robotic movement and object-hand penetration. In addition, without considering the object physical properties, the interaction between hand and different objects will be unreal.

- Scalability. A subject is usually able to grasp a large amount of objects with different

grasp modes (*e.g.* power and pinch grip). Even with the same grasp mode and the same object, the subject should be able to display different grasping motions as the physical property (*e.g.*weight, friction) changes. And therefore, it requires our algorithm should include a rich set of grasping motion, grasp modes and objects.

- Control. An ideal animation system should be intuitive and empower the user with easy-to-use interfaces. The system should allow novice users to generate a desired grasping action quickly and easily–with virtually no learning curve.

- Realtime. The system should be quickly enough to allow interaction with the user. The user experience is essential for the system.

More specifically, we propose a realtime and robust system that synthesizes physically realistic human grasping motions. Given an object to be grasped, our system is capable to compute desired controls (*e.g.* forces and torques) that advances the simulation to achieve the realistic grasping. Our solution includes two significant components, the kinematic grasping motion synthesis and the physics-based grasping control. The kinematic grasping motion synthesis process utilizes a data-driven method that synthesizes realistic grasping motion with a large set of prerecorded grasping motion data. The physics-based control process then transforms the synthesized kinematic motion into a physically realistic one. Additionally, we also provide a performance interface that novice users can act out in a single *Kinect* camera to achieve virtual grasping. We demonstrate the power of our approach by generating physically realistic grasping motions for objects with different properties *e.g.* shapes, weights, frictions, spatial positions and orientations. We also validate our physics-based control approach by showing robustness to external perturbations and changes in physical quantities.

I.1 Contributions.

For our hand tracking system, we have made the following technical contributions:

- First and foremost, an end-to-end realtime 3D hand tracking system that accurately and automatically tracks 3D hand poses using a single depth camera.
- A shape-indexed cascaded pose regression method that significantly extends the idea of cascaded pose regression to 3D hand pose regression.
- An automatic 3D hand shape modeling method that accurately constructs a subject-specific skinned mesh model from a small set of depth images.
- A hybrid tracking scheme that complements model-based tracking with per-frame pose regressor.

For our physically realistic grasping synthesis, we have made the following technical contributions:

- An efficient data-driven synthesis algorithm that utilizes a large set of prerecorded human grasping data to generate realistic, controllable animation for grasping objects substantially different from database objects.
- A robust physics-based motion control algorithm that transforms kinematic motions of the hand and object into physically realistic ones.
- A performance interface that allows the user to create a desired grasping action by acting out the motion in front of a single *Kinect* camera.
- A high-quality human grasping database for grasping a wide variety of objects with different grip styles, that will enable other researchers to apply their algorithms to this problem.

In the second chapter, we will describe how to build an automated and robust hand tracking system using a single depth camera, including the shape-invariant pose regressor, hybrid tracking algorithm and rapid geometric hand shape modeling. In the third chapter, we will describe how to synthesize physically realistic grasping motions for a large variety of objects, including the construction of the grasping motion dataset, kinematic grasping motion synthesis, and physics-based transformation. In the final chapter, we will make a summary of our work and propose the future work.

CHAPTER II

AUTOMATIC AND REALTIME HAND TRACKING USING A SINGLE DEPTH CAMERA

The ability to accurately track 3D hand poses in realtime would allow interactive animation and control of human gestures in movies, video games, virtual environments and teleconferences. Such a system would also facilitate touchless user interaction in smartphones, computers, head mounted displays (HMD), robots and machines. However, realtime tracking of 3D hand gesture is extremely hard because it requires reconstructing both global hand movements and delicate hand articulations.

Decades of research in computer graphics have explored a number of approaches to capture hand motion data, including marker-based systems, glove-based systems, inertial systems and image-based systems. Despite the efforts, acquiring accurate hand motion data in realtime remains a challenging task. Marker-based systems (*e.g. Vicon*) often produce ambiguous solutions because of significant self-occlusion. Glove-based systems (*e.g. CyberGlove*) and inertial systems (*e.g. Noitom*) are occlusion-free but captured motion data are often noisy and might not be able to track delicate hand articulations. In addition, glove-based systems and inertial systems are cumbersome and unwieldy, thereby impeding the subject's ability to perform the motion. Image-based systems offer an appealing alternative to hand motion capture because they require no markers, no gloves, or no wear-on sensors and thereby do not impede the subject's ability to perform the motion.

One appealing solution to image-based systems is model-based hand pose tracking [5], which initializes a 3D hand pose at the first frame and sequentially tracks 3D poses by minimizing the inconsistency between the hypothesized poses and the observed image data. Recent efforts in model-based tracking (*e.g.*, [6, 7]) have focused on 3D hand tracking us-

ing RGBD images obtained by a single depth sensor and have demonstrated impressive results for 3D hand tracking. Model-based tracking methods, however, are often not fully automatic because they require manual intervention on constructing a subject-specific 3D hand geometric model for tracking and initialization of the starting poses. Another limitation is that they cannot automatically recover from failures once the system gets stuck in the local minimum.

This chapter presents an automatic and robust method for accurately tracking 3D hand poses in real time via a single depth camera (Figure II.1). Our key idea is to introduce an efficient hybrid tracking scheme that complements model-based tracking with an efficient per-frame pose regressor. Our per-frame pose regressor directly estimates 3D hand poses from depth images. Model-based tracking and per-frame pose regressor are complementary to each other. At one end, model-based tracking can produce more accurate results but often requires manual initialization and recovery. At the other end, 3D pose detection can automatically infer 3D human poses from single depth images but often with less accurate results. An appropriate combination of both techniques provides benefits at both ends. Our pose regressor is built upon cascaded pose regression [12, 13, 10], where the object pose is estimated progressively via a sequence of weak regressors and each weak regressor uses features that depend on the estimated pose from the previous stage. We significantly extend the idea of cascaded pose regression to 3D hand pose regression by utilizing a low-dimensional 3D parametric hand model to learn 3D shape-indexed features that are more robust to variations in hand shapes, viewpoints and hand poses. In addition, we introduce an automatic hand shape modeling method that accurately constructs a subject-specific skinning hand mesh model from a small number of depth images. This idea not only automates the whole tracking system but also improves the robustness and accuracy of model-based tracking and hand pose regression.

Our final system is appealing for realtime hand motion tracking because it is low-cost,

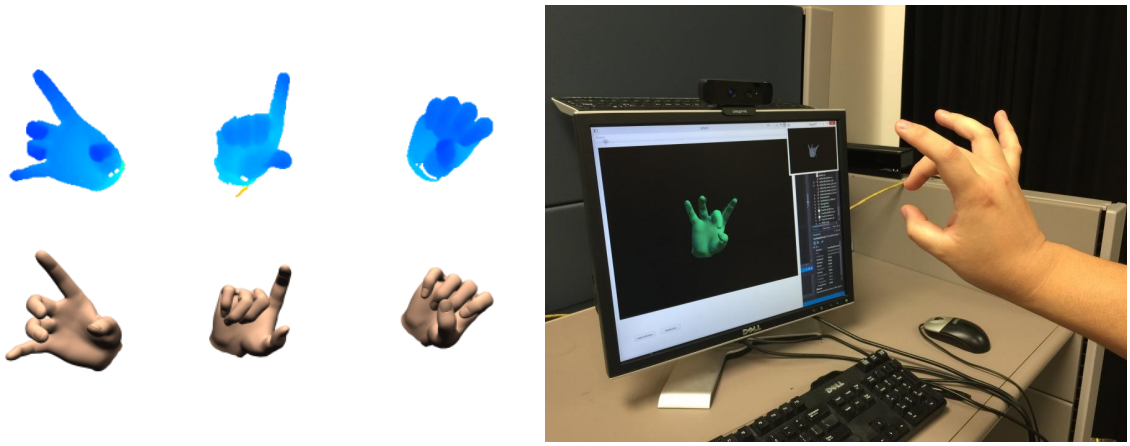


Figure II.1: Our automatic system accurately tracks 3D hand poses in real time via a single depth sensor. (Left) Top row: depth input data, Bottom row: reconstruction results; (Right) A live demo of our realtime hand tracking system.

non-intrusive, and robust to large variations in hand shapes, viewpoints and hand poses multiple and allows for accurate reconstruction of 3D hand poses even under significant occlusions. Our system is implemented on a GPU and achieves realtime performance (45 frames per second). We demonstrate the effectiveness of our system by testing on a wide range of complicated hand movements, including various combinations of global hand movements and delicate hand articulations. We achieve state-of-the-art accuracy in our comparison against alternative systems. We also validate our algorithm by evaluating the importance of key components of our 3D hand tracker.

Contributions. Our system has made the following technical contributions:

- First and foremost, an end-to-end realtime 3D hand tracking system that accurately and automatically tracks 3D hand poses using a single depth camera.
- A shape-indexed cascaded pose regression method that significantly extends the idea of cascaded pose regression to 3D hand pose regression.
- An automatic 3D hand shape modeling method that accurately constructs a subject-

specific skinned mesh model from a small set of depth images.

- A hybrid tracking scheme that complements model-based tracking with per-frame pose regressor.

II.1 Background

Various approaches have been proposed for tracking 3D hand poses from input RGB/RGBD data. Generally, those approaches can be divided into two categories: appearance-based detection and model-based tracking. Our approach combines the advantages of both approaches while avoiding their disadvantages. In the following, we discuss most relevant papers on hand tracking related to our approach in these two categories. In addition, we also introduce a low-dimensional parametric hand shape model for rapid construction of a subject-specific skinning hand mesh model required for 3D hand tracking. We, therefore, also discuss relevant work in 3D hand shape modeling and reconstruction.

Appearance-based detection. One appealing solution to hand motion tracking is to take a bottom-up approach to predict 3D positions of hand joints directly from RGB or depth images, which is called appearance-based detection approach. The approach is appealing because it does not assume any 3D hand mesh model, does not require 3D pose initialization, and does not get trapped to any local minima. Notably, Wang and his colleagues [20, 21] extracted image features from input RGB images and used them to predict corresponding 3D hand poses by searching the closest example in the preprocessed hand pose database. Recently, Tang and his colleagues [22, 23] presented an efficient regression forest technique to estimate 3D hand poses directly from depth images. More recently, Tompson and his colleagues [9] employed convolutional networks to learn a mapping that directly transforms input depth images to 3D hand joint locations. Oberweger and colleagues[24, 25] further extended Tompson's work by adding a feedback convolution network to refine the estimated 3D pose. Sun and his colleagues [10] extended

2D cascaded pose regression [12] to 3D hand pose regression by employing hierarchical regression to predict 3D joint positions directly from input depth images. Tang and his colleagues [11] predicted hand poses from input depth data using hierarchical pose regression and then refined the prediction poses by sampling around the candidate poses.

Our methods builds upon per-frame pose regressor that directly estimates 3D poses from single depth images. Among all the systems, our hand pose regressor is most similar to [10]. Both systems are based on per-frame cascaded pose regression. However, our work significant extends cascaded pose regression to hand pose prediction because we utilize a low-dimensional parametric hand model to learn 3D shape-indexed features that are more robust to variations in hand shapes, poses and viewpoint. The evaluation in Section II.6.3 clearly shows that our pose regression method achieves much more accurate results than theirs [10]. Another distinction is that we complement 3D pose regression with model-based tracking, thereby further improving the robustness and accuracy of our tracking system. Finally, our system is fully automatic because we introduce an automatic 3D hand modeling process that accurately reconstructs a subject-specific skinning hand model from a small number of depth images. In contrast, they rely on manual interaction to create a 3D hand model for each subject, which is required for cascaded 3D pose regression.

Model-based tracking. An alternative solution to track 3D hand poses from image data is model-based tracking (*e.g.*, [5, 26]), which initializes a 3D hand pose at the first frame and sequentially tracks 3D poses by minimizing the inconsistency between the hypothesized poses and the observed image data. One possibility is to sequentially search an optimal pose to best match observed depth data via Particle Swarm Optimization (PSO) [27, 28, 29, 30, 31]. PSO employs random sampling technique to search an optimal pose in the vicinity of the previous pose. Recently, Qian and his colleagues [30] improved the search efficiency of PSO by combining with Iterative Closest Points (ICP).

Particle filter is also introduced to improve the searching efficiency of PSO in [32, 33]. Unlike PSO, ICP performs local search of an optimal pose using gradient-based optimization, thereby increasing the tracking accuracy. ICP and its variants have been successfully applied in human body tracking[34, 35] and hand tracking [36, 7]. Both Qian et al. [30] and Taglisacchi et al. [7] employed fingertip detection as reinitializer to deal with tracking failure. However, it requires finger tips to be clearly visible. Sridhar et al. [37] represented the hand model and depth data as a mixture of Gaussian and estimated the poses by minimizing the difference between two Gaussian functions with parts detection as reinitializer.

Model-based pose tracking approaches can often produce more accurate results than detection-based approaches but they are not fully automatic and require manual initialization of the starting poses. They are also not robust because they cannot automatically recover from failures once the system gets stuck in the local minimum. We address these challenges by complementing model-based tracking with per-frame cascaded pose regression. In addition, we develop a rapid 3D hand shape modeling method that uses a small number of depth images to accurately construct a subject-specific skinned mesh model required for model-based tracking. This step not only automates the whole tracking system but also improves the robustness and accuracy of model-based tracking. It is worth mentioning that our hybrid tracking scheme is flexible and can be combined with any model-based tracking to enhance the accuracy and robustness of model-based tracking.

Rapid hand shape modeling. One critical component of our automatic system is to rapidly construct a subject-specific 3D hand shape model for model-based tracking. One possibility to model human hands is to use simple geometric primitives to approximate hand geometry (*e.g.*, [7, 33]). A more efficient way for 3D hand modeling is to use a skinned mesh model with Linear Blend Skinning (LBS) for pose deformation (*e.g.*, [28, 38, 9, 31]) as done in many hand tracking algorithms. More advanced algorithms such as [39] are also proposed to generate a more plausible or realistic hand model for pose

deformation. However, these methods only account for shape variances induced by pose deformation for a specific subject.

Our work is different because we focus on rapid construction of a subject-specific skinning hand model from depth images. To achieve this goal, we learn a low-dimensional parametric skinning hand model for pose deformation using a large database of high-quality scanning mesh models of human hands. Unlike previous work, our modeling process is also fully automatic because we estimate low-dimensional model parameters directly from a small set of input depth images. Our low-dimensional parametric hand model is relevant to a linear shape model for pose deformation [40]. Our model is different and more expressive because we decouple shape variations into skeleton scale variations and vertex offset variations, and model each of them using a low-dimensional parameters. In addition, we construct the parametric model from high-quality scanning mesh models annotated with a large set of point correspondences rather than low-resolution unlabeled depth images, thereby significantly improving the resolution and accuracy of parametric mesh models.

II.2 Overview

We aim to build an automatic realtime hand tracking system that robustly and accurately tracks 3D hand poses using a single RGBD camera. To address this challenge, we first introduce a rapid hand shape modeling method that automatically fits a low-dimensional parametric hand model to a small number of depth images. Then we propose a shape-indexed cascaded pose regression method robust to large variations in hand shapes, poses and camera viewpoints. Finally, we introduce a hybrid tracking scheme to combine the power of cascaded hand pose regression and model-based tracking. An overview of our system can be seen in Figure II.2.

Acquisition device. Our depth data capture device is *Intel Creative Gesture Camera*

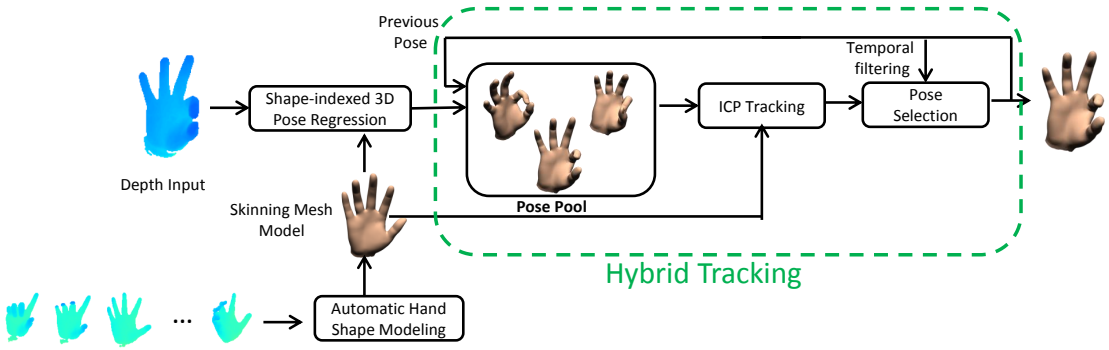


Figure II.2: System overview.

time-of-flight sensor as in Figure II.3. It can capture near-range (20cm to 50cm) depth data with 320*240 resolution up to 60fps. It also provides corresponding infrared data, which can be used to remove depth noise around the edges.

Automatic hand shape modeling. Accurate reconstruction of a subject-specific hand shape model is critical to our tracking system because both our shape-indexed hand pose regression and model-based tracking rely on the hand shape model. We represent the subject's hand skeleton and skinned mesh model using a low-dimensional parametric hand model $\mathbf{H}_i(\alpha, \beta, \mathbf{q})$, where the parameters α and β control the shape and skeleton scale variations, respectively, and the parameter $\mathbf{q}$ models the joint angle pose of 3D hand. We instruct the subject to perform a small set of predefined poses and automatically optimize the model's shape parameter α, β and hand poses $\mathbf{q}$ by minimizing the difference between the hypothesized hand model and the observed depth images. Each subject needs to perform hand shape modeling once before tracking.

Shape-indexed cascaded pose regression. We extend cascaded pose regression by incorporating the reconstructed subject-specific hand shape model into cascaded pose regression. Specifically, we develop a shape-indexed feature parameterization method that represents a 3D feature point as its associated skinned mesh vertex plus a 3D vertex offset.

Figure II.3: Capture device: Intel Creative Gesture Camera.

The association is computed by choosing the mesh vertex with the minimum distance to the 3D feature point. We train cascaded pose regressor based on a wide range of data sets generated by random sampling the low-dimensional parametric hand model. The pose regression works in hierarchical order that first regresses the palm and then five fingers respectively. In addition, for regression output, we do not simply use the mean or median of all leafs. Instead, we extract all the possible modes by clustering leafs and choose to keep all the modes for further decision.

Hybrid tracking. We introduce a hybrid tracking framework that combines advantages of both per-frame pose regression and model-based tracking. Our first idea is to apply model-based tracking to refine 3D poses obtained by cascaded pose regression. In addition, we utilize the temporal coherence to improve the accuracy of the cascaded pose regression. Such a combination not only automates the whole capturing process but also significantly improves the accuracy and robustness of the system.

We describe these components in detail in the next sections.

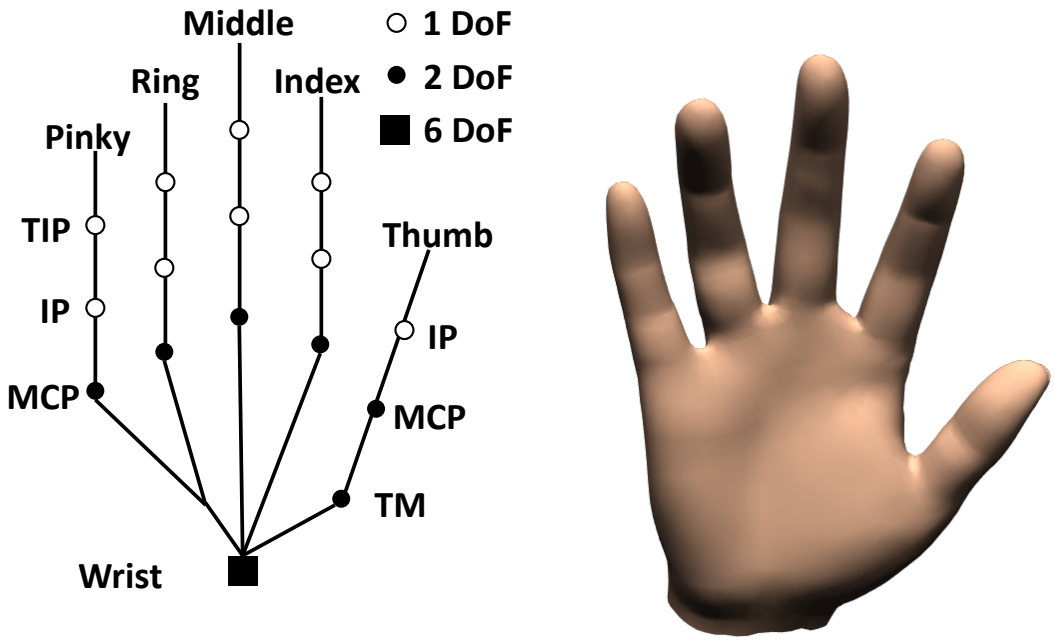
II.3 Automatic Hand Shape Modeling

In this session, we present a low-dimensional parametric model for 3D hand modeling. Our core idea is to construct a low-dimensional parametric model that compactly represent hand shape variations across individuals and enhance it by adding Linear Blend Skinning (LBS) for pose deformation. Mathematically, we model a 3D hand mesh model by $H(\alpha, \beta, \mathbf{q}; \mathbf{W})$, where the shape parameters α, β provides a low-dimensional representation of hand shape variances across individuals, the pose parameter $\mathbf{q}$ specifies the joint angle values of the 3D hand pose, and $\mathbf{W}$ represents the skinning weights required for skinning deformation. Our parametric model provides a continuous and compact representation for allowable shape variances across different human subjects, but is specific enough not to allow arbitrary variations that are not similar to those seen in the database. With this parametric model, we could randomly sample the parameters α, β and $\mathbf{q}$ to generate an infinite number of natural-looking hand models in different shapes and under different poses. Furthermore, we could choose the parameters so that the model would match various forms of user input.

In the following sessions, we describe how to represent a hand, how to model a hand geometry by a low-dimensional parametric model, and then how to acquire a hand model for a subject using depth data.

II.3.1 Hand Representation

We approximate the hand geometry using a skinned mesh model, which is driven by an articulated skeleton model using Linear Blend Skinning (LBS). The skinned mesh model consists of 4138 vertices and 8227 faces, and the skeleton model consists of 27 bone segments, as in Figure II.4 (a) and Figure II.4 (b). We describe a hand pose using a set of independent joint coordinates $\mathbf{q} \in R^{27}$, including absolute palm position and orientation



(a) Skeleton model (b) Mesh model

Figure II.4: Our hand pose consists of 27 degrees of freedom, including absolute palm position and orientation (6) and the relative joint angles of individual joints (21). (a) hand skeleton model; (b) skinned hand mesh model.

(6 Dof), as well as relative joint angles of individual joints. These bones includes thumb-tm (2 Dof), thumb-mcp (2 Dof), thumb-ip (1 Dof), index-mcp(2 Dof), index-pip (1 Dof), index-dip (1 Dof), middle-mcp (2 Dof), middle-pip (1 Dof), middle-dip (1 Dof), ring-mcp (2 Dof), ring-pip (1 Dof), ring-dip (1 Dof), pinky-mcp (2 Dof), pinky-pip (1 Dof) and pinky-dip (1 Dof). Here tm, mcp, ip, pip, dip represents trapeziometacarpal, metacarpophalangeal, interphalangeal, proximal interphalangeal and distal interphalangeal respectively.

II.3.2 3D Hand Shape Geometry Model

We model shape variances between subjects by two components: skeleton scales, which encodes coarse-level variation like the global scale and scale of the bones, and mesh vertex offsets, which encodes fine-level shape variation, such as thickness of a finger. We define $\hat{\mathbf{v}}_{ij}$ as the ith vertex in the jth bone's local coordinate, which can be directly computed as

$$\hat{\mathbf{v}}_{ij} = \hat{\mathbf{T}}_j^{-1} \hat{\mathbf{v}}_i, \tag{II.1}$$

where $\hat{\mathbf{v}}_i$ is the ith vertex of the template mesh, $\hat{\mathbf{T}}_j^{-1}$ is the inverse transformation for the jth bone of the template skeleton under rest pose. Note that both scale and offset transformation are operated on $\hat{\mathbf{v}}_{ij}$.

Skeleton scales. It is obvious to see that skeleton size varies among subjects, and the shape of a subject could largely be determined by its skeleton size (bone length), as shown in Figure II.5 (a). For a given template skeleton with bone sizes $\hat{\mathbf{B}} = [\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_{n-1}]$, we represent the skeleton size of a new subject's skeleton $\mathbf{B} = [b_0, b_1, \ldots, b_{n-1}]$ using an overall scale and scale of bones $\mathbf{S} = [s_g, s_0, s_1, \ldots, s_{n-1}]$,

$$b_i = s_g \cdot s_i \cdot \hat{b}_i, \;\; i = 0, 1, 2, ..., n-1, \tag{II.2}$$

where s_g is the overall scale and s_i is the scale for each bone i.

To apply the scale to the template, we use a method similar to Linear Blend Skinning (LBS) except that we scale the vertices in the local coordinate before transform it back to the world coordinate. That is, we first transform each vertex to the local coordinates of corresponding bones according to the skinning weights, scale them by the overall scale s_g uniformly, and then scale them in the direction of the bone based on the bone scales, transform the scaled vertices back to world coordinate, and then compute the final vertex

position by the linear combination of scaled vertices based on the skinning weights. Note that we treat the global scale s_g separately since we apply s_g on all axes uniformly but apply the bone scales only in the direction of the bones. To summarize, the scaled vertex $\hat{\mathbf{v}}'_{ij}$ in local coordinate of bone j can be expressed as

$$\hat{\mathbf{v}}'_{ij} = \mathbf{S} \otimes \hat{\mathbf{v}}_{ij},$$

$$\mathbf{S} \otimes \hat{\mathbf{v}}_{ij} = s_g s_j (\hat{\mathbf{v}}_{ij} \cdot \mathbf{d}_j)\mathbf{d}_j + s_g(\hat{\mathbf{v}}_{ij} - (\hat{\mathbf{v}}_{ij} \cdot \mathbf{d}_j)\mathbf{d}_j), \tag{II.3}$$

where $\hat{\mathbf{v}}_{ij}$ is the ith vertex in the jth bone's local coordinate, $\mathbf{d}_j$ is the jth bone's direction in its local coordinate, $(\hat{\mathbf{v}}_{ij} \cdot \mathbf{d}_j)\mathbf{d}_j$ and $\hat{\mathbf{v}}_{ij} - (\hat{\mathbf{v}}_{ij} \cdot \mathbf{d}_j)\mathbf{d}_j$ represent the component of $\hat{\mathbf{v}}_{ij}$ parallel to and perpendicular to the jth bone's direction respectively, $\mathbf{q}_{rest}$ is the rest pose, $T_j(\mathbf{q}_{rest})$ is the transformation of the jth bone for pose $\mathbf{q}_{rest}$, and $\mathbf{W} = [w_{ij}]$ is skinning weights for deformation, and n is the number of bones.

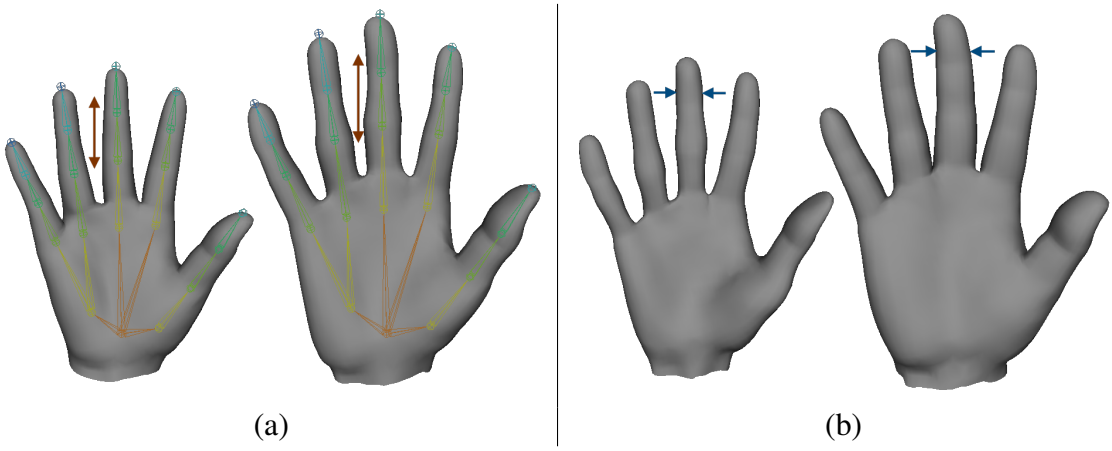


Figure II.5: Hand shape variations: (a) skeleton size variation; (b) thickness variation.

Using the skeleton scales, we could model the structure of the shape accurately. How-

ever, this representation is not compact, since the scales between bones are highly correlated. For example, if the proximal phalange of a finger is longer than the template, then it is likely that the intermediate and distal phalanges of the same finger will be longer too. To reduce the dimension, we apply Principal Component Analysis (PCA) to the scales $\mathbf{S}$ to build a compact model by

$$\mathbf{S}(\alpha) = \mathbf{S}_0 + \mathbf{C}_S\alpha, \tag{II.4}$$

where $\mathbf{S}_0$ is the mean skeleton scales of all subjects, $\mathbf{C}_S$ is the coefficient matrix of principal components. α is the low dimensional representation of $\mathbf{S}$.

Vertex offsets. Skeleton scales $\mathbf{S}$ only models the coarse shape of the subjects. To model subtle details of the subject, such as palm and finger thicknesses, we use vertex offsets $\mathbf{Q} = [\delta\mathbf{v}_0, \delta\mathbf{v}_1, \ldots, \delta\mathbf{v}_m]$, i.e., displacement vector between the subject and the template mesh for each template vertex for this purpose, as shown in Figure II.5 (b). We model the offsets in the coordinate of the template under rest pose, which eliminates the skeleton size and pose variances for the subject and makes the vertex offsets invariant to scale and pose. That is, we apply the offsets to the template before scaling the vertices using skeleton scales. To summarize, the offsetted vertex $\hat{\mathbf{v}}'_{ij}$ can be expressed as

$$\hat{\mathbf{v}}'_{ij} = \mathbf{Q}_i \oplus \hat{\mathbf{v}}_{ij} = \hat{\mathbf{v}}_{ij} + T_j^{-1}(\mathbf{q}_{rest})\mathbf{Q}_i, \tag{II.5}$$

where $T_j^{-1}(\mathbf{q}_{rest})$ represents the inverse transformation of the jth bone under the rest pose $\mathbf{q}_{rest}$, and $\mathbf{Q}_i = \delta\mathbf{v}_i$.

Although vertex offset is powerful to capture the shape details, it is a high dimensional and redundant representation due to the smoothness of neighboring vertices. For example, if the palm of a subject is thicker than template, then neighboring vertices of the palm will

probably have similar displacements. Similar to the skeleton scale model, we apply PCA to the vertex offsets to get a compact representation

$$\mathbf{Q}(\beta) = \mathbf{Q}_0 + \mathbf{C}_\mathbf{Q}\beta, \tag{II.6}$$

where β is the low dimensional representation, $\mathbf{Q}_0$ is the mean offsets and $\mathbf{C}_Q$ is the eigen basis.

Shape model. We can define a composite transformation by combining both scale (Equation (II.3)) and offset (Equation (II.5))transformations to get the local coordinates of the shape vertex $\hat{\mathbf{v}}'_{ij}$

$$\hat{\mathbf{v}}'_{ij} = \mathbf{S}(\alpha) \otimes (\mathbf{Q}(\beta) \oplus \hat{\mathbf{v}}_{ij}), \tag{II.7}$$

II.3.3 Hand Pose Model

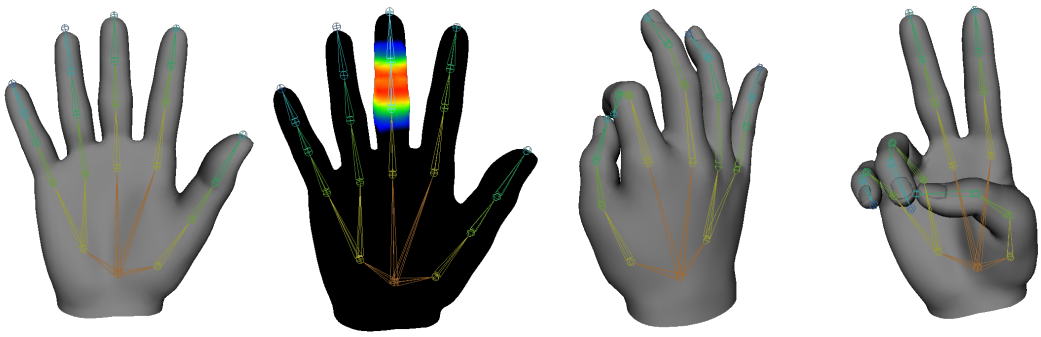
For a specific hand mesh model with a skeleton attached, we use Linear Blend Skinning (LBS) to deform the pose of the mesh model, as shown in Figure II.6.

LBS defines how a geometry deforms according to the underlying bones, and consists of three components: a template mesh M_s and a skeleton K_s in rest pose for subject s, and corresponding skinning weights $\mathbf{W}$. M_s and K_s define the geometry and underlying skeleton structure for subject s in rest pose $\mathbf{q}_{rest}$, which could be obtained from our shape model. The weight map describes how the mesh vertices are influenced by the bones. We deform M_s to generate mesh in new poses by changing $\mathbf{q}$, as shown in Figure II.6(c) and (d).

The vertex $\mathbf{v}_i$ after deformation is described as

$$\mathbf{v}_i(\mathbf{q}) = \sum_{j=0}^{n-1} w_{ij} T_j(\mathbf{q}) \hat{\mathbf{v}}_{ij}, \tag{II.8}$$

where $\hat{\mathbf{v}}_{ij}$ is the ith vertex coordinate expressed in the jth bone's local coordinate, $\mathbf{q}$ is the pose for deformation, $T(\mathbf{q})_j$ is the transformation of the jth bone for pose $\mathbf{q}$, and $\mathbf{W} = [w_{ij}]$ is the sparse skinning weights for deformation, and n is the number of bones.



(a) (b) (c) (d)

Figure II.6: Pose modeling: (a) Mesh and skeleton in rest pose; (b) weight map for one of the bones (hotter colors for larger weights); (c)-(d) new meshes generated by changing the joint angle pose $\mathbf{q}$.

II.3.4 Parametric Hand Model

By combining the shape and pose model in Equation (II.7) and (II.8), we have a low-dimensional parametric hand model representing by skeleton scales α, vertex offsets β, and pose $\mathbf{q}$

$$\begin{aligned} \mathbf{H}_i(\alpha,\beta,\mathbf{q};\mathbf{W},U) &= \sum_{j=0}^{n-1} w_{ij} T_j(\mathbf{q}) \hat{\mathbf{v}}'_{ij}, \\ \hat{\mathbf{v}}'_{ij} &= \mathbf{S}(\alpha) \otimes (\mathbf{Q}(\beta) \oplus \hat{\mathbf{v}}_{ij}), \end{aligned} \tag{II.9}$$

where $\mathbf{H}_i(\cdot)$ is the coordinate of the ith vertex of the mesh model, $\hat{\mathbf{v}}'_{ij}$ is the result of the scale and offset transformation on $\hat{\mathbf{v}}_{ij}$, $\mathbf{q}$ is the pose for deformation, $T_j(\mathbf{q})$ is the

transformation of the jth bone for pose $\mathbf{q}$, $U = \{\mathbf{S}_0, \mathbf{C}_S, \mathbf{Q}_0, \mathbf{C}_Q\}$ is the shape basis defined in Equation (II.4) and (II.6), and $\mathbf{W} = [w_{ij}]$ is a sparse weight map for deformation, n is the number of bones.

Using this model, we first generate the subject-specific template mesh M_s and skeleton K_s for each subject according to the shape model under rest pose, and then generate meshes under different poses using the skin weights $\mathbf{W}$ and pose $\mathbf{q}$. By combining the shape and pose model, we can describe arbitrary hand geometries. The model parameters, including the statistical representations of skeleton size and vertex offsets, and the skin weights, are all learned from a database.

II.3.5 Model Learning

To build the parametric hand model, we will need to learn the skinning weights $\mathbf{W}$ and the shape basis $U = \{\mathbf{S}_0, \mathbf{C}_S, \mathbf{Q}_0, \mathbf{C}_Q\}$ based on a template mesh $\hat{M}$ and its skeleton $\hat{K}$. We propose a data-driven method to learn these parameters from a scan database.

We build a scan database by scanning human right hands using an Artec Eva 3D scanner. The scanner provides dense and detailed 3D triangular meshes, which contain about 70k vertices and 140k faces. We captured data from 166 subjects (103 males and 63 females). Each subject is asked to perform $1 \sim 5$ poses from a set of predefined poses, which gives us 466 models in total.

For a scanned database $Y = \{Y^{st}\}$ representing the set of vertices for the scan st, where s is the subject index, t is the pose index for the subject, we would like to learn the shape basis U, the skin weights $\mathbf{W}$, as well as per scan parameters $SP = \{\alpha^s, \beta^s, \{\mathbf{q}^{st}\}, C^{st}\}$ so that the synthesized hand model $\mathbf{H}(\alpha^s, \beta^s, \mathbf{q}^{st}; \mathbf{W}, U)$ defined in Equation (II.9) fits the scanned meshes best

$$U^*, \mathbf{W}^* = \underset{U, \mathbf{W}, SP}{\arg\min} \sum_s \sum_t \left\| \mathbf{H}(\alpha^s, \beta^s, \mathbf{q}^{st}; \mathbf{W}, U) - Y_C^{st} \right\|^2, \tag{II.10}$$

where α^s, β^s are the shape parameters for subject s, $\mathbf{q}^{st}$ is the pose for the scan, $\mathbf{H}(\cdot)$ gives the set of vertices of the synthesized hand model, and Y_C^{st} is the corresponding set of vertices in the scan st. The correspondence building process is similar as in Section II.5.2.1.

To learn the model, we first initialize skin weights $\mathbf{W}$ according to [41], and then fix $\mathbf{W}$ to learn the skeleton scales $\mathbf{S}$, vertex offsets $\mathbf{Q}$ and poses $\{\mathbf{q}\}$ for each subject using non-linear optimization. Then, we optimize $\mathbf{W}$ on all subjects by fixing skeleton scales $\mathbf{S}$, vertex offsets $\mathbf{Q}$ and poses $\{\mathbf{q}\}$. Finally, we learn PCA models for skeleton scales $\mathbf{S}$, vertex offsets $\mathbf{Q}$ on all subjects.

II.3.6 Automatic Construction of Subject-specific Hand Model

The subject-specific hand model can be represented as $\mathbf{H}_i(\alpha, \beta, \mathbf{q}; \mathbf{W}, U)$ as described in Equation (II.9). $\mathbf{W}$ and U are already learnt in Section II.3.5. To acquire the subject's hand model, we should solve (α, β). Here is our model construction process:

(1) We instruct the subject to perform a small set of pre-defined hand poses $\{\mathbf{q}_t^d\}$ and record the corresponding depth images.

(2) To reconstruct both the shape and poses of the subject, we formulate an optimization problem and seek to find the optimal shape parameter α, β and poses $\{\mathbf{q}_t\}$ for each depth image I_t.

$$\arg\min_{\alpha,\beta,\mathbf{q}_t} \sum_t \|\mathbf{H}(\alpha, \beta, \mathbf{q}_t) - \mathbf{P}_C\|^2, \tag{II.11}$$

where C is the set of correspondence, and $\mathbf{H}(\cdot)$ the set of vertices of the synthesized hand model, $\mathbf{P}_C$ is the corresponding point cloud generated from the depth image I_t. The correspondence building process is similar to Section II.5.2.1.

(3) We optimize the shape and poses as follows. We initialize the pose $\{\mathbf{q}_t\}$ to $\{\mathbf{q}_t^d\}$. We first optimize the shape parameters α, β by fixing $\{\mathbf{q}_t\}$ and then update the poses $\{\mathbf{q}_t\}$ by keeping the shape parameters α, β constant. We iterate the process until convergence.

This step allows us to obtain the subject-specific hand model (α, β).

II.4 Cascaded Hand Pose Regression via Shape-Indexed Features

Cascaded pose regression has already been introduced in hand tracking [10]. However, their pose regression is not fully automatic and need manual intervention to estimate the skeleton size of the subject. In addition, they do not consider the variation of hand shape among subjects, because their feature parameterization is only related to a manual estimate of hand scale. Then, for regression output in leafs, they directly use the mean or the median, and therefore lose some significant modes.

Our method can automatically acquire the hand model of a subject by employing a parametric hand model as described in Section II.3. And we propose a shape-indexed feature parameterization that encodes hand shape information into features. It will significantly increase the feature invariance to hand shape. In addition, for regression output, we do not simply use the mean or median, but extract several modes by clustering and keep all modes for further decision.

In order to estimate the hand pose, we begin with a depth image I and an initial pose $\mathbf{q}_0$. Then we iteratively update the hand pose by prebuilt cascaded regressors $\mathbf{R} = \{\mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_C\}$:

$$\mathbf{q}_s = \mathbf{q}_{s-1} + \mathbf{R}_s(I, \mathbf{M}(\mathbf{q}_{s-1})), s = 1, 2, ..., C, \tag{II.12}$$

where $\mathbf{q}_s$ and $\mathbf{q}_{s-1}$ represents hand pose in sth and $(s-1)$th stage, C is the total number of cascades, $\mathbf{M}(\mathbf{q}_{s-1})$ represents the skinned mesh model under hand pose $\mathbf{q}_{s-1}$, and $\mathbf{R}_s(I, \mathbf{M}(\mathbf{q}_{s-1}))$ represents the regressor in sth stage with shape-indexed features on depth image I.

Each regressor $\mathbf{R}_s$ is represented by a standard random regression forest [42, 43], which is learnt to approximate the difference between the ground truth pose and previous pose

$\mathbf{q}_{s-1}$ on all training samples. Starting from the root node of each regression tree, it compares the depth difference between two parameterized pixels with a threshold, and then either goes to the left branch or the right branch determined by whether the difference is larger than the threshold. The process is terminated until it reaches the leaf. Each leaf stores a $\delta\mathbf{q}$ to update the input pose. Pixel selection, threshold and pose update stored in leafs are all learnt from training process. The proposed shape-index feature is to parameterize pixel selection in order to make each selected pixel as invariant as possible for all training samples. It relates pixel parameterization to both the hand shape model and previous pose $\mathbf{q}_{s-1}$. Such shape-indexed features not only provide good geometric invariance as pose-indexed features, but also provide better hand shape invariance than pose-indexed features. The shape-indexed features make our system much more robust when tracking the hand from different subjects.

In the following subsections, we will first describe how to perform shape-indexed feature parameterization, how to perform hierarchical regression, and how to build training datasets.

II.4.1 3D Shape-Indexed Features

We will first describe how to extract pose-indexed features for 3D hand pose regression, and then how to improve the shape variance by extending pose-indexed features to shape-indexed features.

Similar to previous cascaded pose regressions on object tracking [12] and facial tracking [13], we use the pixel-pair difference features, $I(u_1) - I(u_2)$, where u_1 and u_2 are two random pixels from the depth image I. The key to achieve pose invariance and shape invariance is how to parameterize u_1 and u_2.

Pose Invariance. To achieve pose invariance, u_1 and u_2 are defined as the reference pixel $u_{1,ref}$ and $u_{2,ref}$ plus 2D offsets δu_1 and δu_2 expressed in the pose-indexed local

coordinate:

$$\begin{aligned} u_1 &= u_{1,ref} + T_{1,ref}\delta u_1, \\ u_2 &= u_{2,ref} + T_{2,ref}\delta u_2, \end{aligned} \tag{II.13}$$

where $T_{1,ref}$ and $T_{2,ref}$ represent the transformation of the pose-indexed local coordinates defined in the image coordinate.

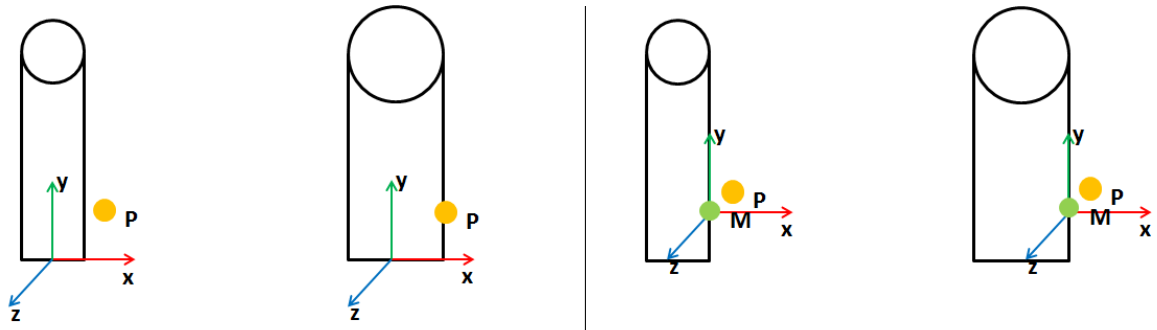
Now we extend it from 2D to 3D as in Equation (II.14) , as we can obtain 3D data from depth camera.

$$\begin{aligned} u_1 &= CamProj(u_{1,ref} + T_{1,ref}\delta u_1), \\ u_2 &= CamProj(u_{2,ref} + T_{2,ref}\delta u_2), \end{aligned} \tag{II.14}$$

where $u_{1,ref}$ and $u_{2,ref}$ are 3D reference points (the origin of the pose-indexed reference coordinate), δu_1 and δu_2 are 3D offsets expressed in the pose-indexed reference coordinate. $CamProj$ represents the camera project matrix that maps 3D point to 2D pixel in the image plane.

In hand pose regression, the reference point can be chosen as the palm position and the pose-indexed reference coordinate can be chosen as the local coordinate of the palm for palm regression. And the reference point can be chosen as the finger root (the position of mcp joint) position and the pose-indexed reference coordinate can be chosen as the local coordinate of the finger root (mcp joint) for finger regression.

Shape Invariance. Pose-indexed features provide good geometric invariance, because all features are selected in the pose-indexed reference coordinate. However, it does not consider the shape variance of the object. For example, one subject's hand is thicker than another's. Then the same 3D offsets **P** may correspond to a hand pixel for one subject and a background pixel for another, as in Figure II.7 (a). It will result in more ambiguity when training and testing the regression model in different subjects. And therefore, we introduce



(a) Pose-indexed features (b) Shape-indexed features

Figure II.7: Pose-indexed features (a) v.s. Shape-indexed features (b). Pose-indexed features do not consider the shape of the hand, and therefore are not robust to hands with different shapes. Shape-indexed features are parameterized by shape information, and therefore robust to hands with different shapes.

shape-indexed features based on pose-indexed features. The shape-indexed features still choose the same reference coordinate, but choose a mesh vertex as the reference point instead of the coordinate origin. As in Figure II.7 (b), **M** is chosen as the reference point, and the 3D offset of the point **P** is computed based on **M**. The parameterization can be represented as follows:

$$\begin{aligned} u_1 &= CamProj(u_{M_1} + T_{1,ref}\delta u_1), \\ u_2 &= CamProj(u_{M_2} + T_{2,ref}\delta u_2), \end{aligned} \tag{II.15}$$

where u_{M_1} and u_{M_2} represent two chosen vertices of skinned mesh model, $T_{1,ref}$ and $T_{2,ref}$ represent the transformation of the reference coordinate for M_1 and M_2. Since hand models can be acquired as described in Section II.3, all mesh models have the same structures and all mesh vertex have been well aligned. For example, we can choose the 100th and the 200th vertex as the reference points, δu_1 and δu_2 can be chosen 3D offsets respect to the 100th and the 200th vertex expressed in their reference coordinates. Similar to [13], we first sample a large amount of 3D points, and then associate each point to its closest

hand mesh vertex. Shape-indexed features are more robust than pose-indexed features, because they consider the shape variance of hands when selecting features.

II.4.2 Cascaded Hierarchical Regression

Generally, there are two methods to do hand pose regression, holistic regression and hierarchical regression.

The holistic regression simply regress the whole hand's pose $\mathbf{q}$. It chooses the palm's local coordinate as the reference coordinate for feature parameterization and pose update, as in

$$\begin{aligned} &\text{for s = 1 : C} \\ &\quad \mathbf{q}_s = \mathbf{q}_{s-1} + T_{palm}\, R_s(I, M_{s-1}), \\ &\text{end} \end{aligned} \tag{II.16}$$

where $\mathbf{q}_s$ and $\mathbf{q}_{s-1}$ are joint poses on the sth and $(s-1)$th stage, T_{palm} is the transformation of the palm's coordinate, $R_s(I, M_{s-1})$ is the regressor in the sth stage with the depth image I and the skinned mesh M_{s-1}

The hierarchical regression regresses the hand pose along the kinematic chain. First regress the palm, and then regress each part along the kinematic chain. In our system, our hierarchy is divided into two parts, palm and five fingers. We first perform palm regression. For palm regression, we choose the palm's local coordinate as the reference coordinate for feature parameterization and pose update. And then we perform finger regression for each finger separately. For each finger, we choose its finger-root(mcp joint)'s local coordinate as the reference coordinate for feature parameterization and pose update. Also we find that joint positions are more robust than joint angles when doing regression. Therefore, we do regression on joint positions and then transfer back to joint angles by inverse kinematics.

The hierarchical regression process can be described as follows:

(1) Regress the palm.

$$\begin{aligned} &\text{for s = 1 : C} \\ &\quad \mathbf{q}_s^{palm} = \mathbf{q}_{s-1}^{palm} + T_{palm}\, R_s(I, M_{s-1}), \\ &\text{end} \end{aligned} \tag{II.17}$$

(2) Update T_f according to T_{palm}.

(3) Regress each finger separately.

$$\begin{aligned} &\text{for s = 1 : C} \\ &\quad \mathbf{q}_s^f = \mathbf{q}_{s-1}^f + T_f\, R_s(I, M_{s-1}), \\ &\quad f \in \{index, middle, ring, pinky, thumb\} \\ &\text{end} \end{aligned} \tag{II.18}$$

where $\mathbf{q}_s^{palm}$ and $\mathbf{q}_{s-1}^{palm}$ are palm poses in the sth stage and $(s-1)$th stage, $\mathbf{q}_t^f$ and $\mathbf{q}_{s-1}^f$ are finger poses in the sth stage and $(s-1)$th stage, T_{palm} and T_f are transformation of the palm's and the finger's reference coordinate.

According to [10], hierarchical regression is more robust than holistic regression. So we choose hierarchical regression here.

Initialization. Both training and testing data require an initial pose $\mathbf{q}_0$ to start the regression. The initial pose for palm regression is computed as follows. First, we compute the mean of point clouds from depth camera as the position of joint middle-mcp. And then we do PCA on point clouds and obtain three axes. We choose the palm's Y axis as the direction with the largest variation, and the Z axis is the direction with the least variation, and then X axis by the cross product of Y axis and Z axis. Then we can compute the palm position according to the skeleton model and middle-mcp joint position. The finger regression initializes fingers at their default positions, *e.g.* joint angles are all 0. The initial

reference coordinate for finger regression, is chosen to be the local coordinate of the mcp joint when joint angles are 0.

II.4.3 Multiple Modes in Regression

Since our data includes a large amount of viewpoint variations and significant occlusions, the regression output may not converge very well in leafs. Simply choosing the median or the mean of all leafs, will not provide a good estimate of the pose, but lose some important modes. And therefore, we cluster all leafs, and keep cluster centers as candidates. All candidates will be kept for further decision.

In our data, we find that leafs of palm regression always have several different modes and must be all kept. 3 or 4 modes are enough for tracking according to our experiment. However, for finger regression, simply using the median or the mean will be good for our tracking. And therefore, for palm regression, we cluster all leafs, and keep cluster centers as palm candidates. For every palm candidate, we perform finger regression, and then obtain a hand pose candidate. All pose candidates will be kept for further decision.

II.4.4 Training Dataset

The regression model should include kinds of variations, including subject variation, pose variation and viewpoint variations. When designing training dataset, all variations above are considered.

Subject variation. In order to include training samples from different subjects, we synthesize 40 hand skeletons and skinned mesh models by randomly sampling (α, β) as described in Section II.3.

Pose variation. We choose poses either from American Sign Languages or from our motion capture data using three Kinects. Every pose is expanded into several poses by adding random perturbations in joint angles.

Viewpoint variation. In order to robust to viewpoint variation, we randomly sam-

ple a large amount of palm global transformations (translations and orientations). Those transformations include almost all valid movement range of palm.

We use synthesized data as our training dataset. We synthesize 40 skeletons and skinned mesh models. For each subject, we have 200 joint poses and 40 global transformations per joint pose. In total, we have around $200 \times 40 \times 40 = 3200000$ training samples in our dataset.

We render a depth image per sample via OpenGL render framework. Since the synthesized data is much cleaner than the real data, we add random noise, silhouette noise into synthesized data to simulate the real data. And therefore we could get 320000 training samples which includes depth image I and ground truth pose ($\mathbf{q}_g$).

II.5 Hybrid Tracking

The regression output may be not compatible with depth image very well due to the noise and large viewpoint variations. The low-quality and noisy output cannot be used for motion capture. In addition, without accurate reconstruction, we cannot acquire accurate temporal information from previous frames to help select the best candidate from a group of candidates. And therefore, we introduce hybrid tracking framework to solve these two issues. The hybrid tracking framework employs model-based tracking techniques to refine the regression output, and then help select the candidates from regression output with previous accurate reconstruction result. Our hybrid tracking framework can work with any model-based tracking methods. In the system, we choose ICP due to the reconstruction accuracy and computational time requirement.

In the section, we will first introduce our hybrid tracking framework, and then introduce our model-based tracking techniques.

II.5.1 Hybrid Tracking Framework

The hybrid tracking framework has two main components: pose refinement and candidate selection.

Pose refinement. Appearance-based approach cannot provide accurate pose reconstruction due to noise, significant occlusion and large viewpoint change. Here we employ model-based tracking *e.g.* Iterative Closest Point (ICP) and Particle Swarm Optimization (PSO), to help further refine the result. Since the regression is hierarchical and the palm regression has significant effects on finger regression, and therefore we use model-based tracking to refine palm regression result of each candidate before finger regression. And then we perform finger regression, and finally refine the whole pose of each candidate.

Candidate selection. To select the correct candidate from the pose pool is very important for tracking accuracy. Simply using depth difference metric will fail for some frames due to the ambiguity of pose-depth correspondence. Therefore, we introduce temporal information to filter candidates. When the previous frame is tracked successfully, the previous pose can be used as temporal filtering for current candidate selection. Here, we employ the average depth error between synthesized depth image of hand model and observed depth image from camera to determine whether a frame is tracked successfully or not. It can be defined as follows:

$$\begin{aligned} \delta &= (E_d < \tau), \\ E_d &= \sum \|I_r - I\| / card((I_r \cup I) > 0), \end{aligned} \tag{II.19}$$

where I_r and I are rendered depth image and observed depth image, $card((I_r \cup I) > 0)$ are the number of non-zero pixel for the union of I_r and I, and τ is a threshold that determines whether tracking is successful or not.

When the previous frame is tracked successfully, we can add its temporal information

into a candidate's cost. And therefore we define the cost of each candidate as:

$$\begin{aligned} E_c &= \lambda_1 E_d + \lambda_2 E_s, \\ E_d &= \sum \|I_r - I\| / card((I_r \cup I) > 0), \\ E_s &= \delta \left\| \mathbf{q}_i - \mathbf{q}_{i-1} \right\|^2, \end{aligned} \tag{II.20}$$

where I_r and I are rendered depth image and observed depth image of a candidate, $card((I_r \cup I) > 0)$ are the number of non-zero pixel for the union of I_r and I, δ is set to 1 if previous frame is tracked successfully and 0 if not, and $\mathbf{q}_i$ and $\mathbf{q}_{i-1}$ is the pose of *i*th frame and *i-1*th frame.

It works well to use regression output as initialization for model-based tracking for most frames. However, only using regression output as initialization will fail for some frames. One reason is that our training data cannot include all situations. Another reason is that both noise and self-occlusion will lower the accuracy of regression. And therefore, we add previous pose with one-iteration refinement into the pose pool.

In addition, since the result of finger regression highly depends on the result of palm regression according to the kinematic chain, we employ model-based tracking to refine the result of palm regression first, and then do finger regression on the refined palm result.

Here is our hybrid tracking framework.

Step 1: Given a depth frame, first run palm regressor to obtain several palm estimates. Then do one-iteration palm refinement for each palm estimate.

Step 2: Run finger regressor on each palm estimate. Put all candidates into the pose pool.

Step 3: Do one-iteration palm refinement on previous frame, and then put it into the pose pool.

Step 4: Perform pose refinement on each candidate in the pose pool. It usually takes 3 to 4 iterations.

Step 5: Choose the candidate with the lowest cost as in Equation (II.20).

Step 6: Update δ for the selected candidate according to Equation (II.19). The computed δ will be used for candidate selection in next frame.

II.5.2 Model-based Tracking

Iterative Closest Point (ICP) and Particle Swarm Optimization (PSO) are classical model-based tracking approaches in human body and hand. In the system, we choose ICP method because of its reconstruction accuracy and search efficiency over PSO. ICP has already been used in human body tracking [34, 35] and hand tracking [7, 36]. Similar to their methods, we formulate the reconstruction problem in a nonlinear optimization framework that iteratively registers our 3D hand model with observed 3d point cloud.

Given the previous hand pose $\mathbf{q}_{i-1}$ and the regression output pose $\mathbf{q}_r$, the observed point cloud data $\mathbf{P}_i$, we aim to refine the hand pose $\mathbf{q}^*$ that best matches the observed point cloud data $\mathbf{P}_i$.

We estimate the hand pose $\mathbf{q}^*$ by minimizing the following objective function:

$$\arg\min_{\mathbf{q}} \ \lambda_1 E_{data} + \lambda_2 E_{smoothness} + \lambda_3 E_{joint}, \tag{II.21}$$

where E_{data} represents 3d registration term that penalizes the registration error between our 3d hand model and the observed point cloud, $E_{smoothness}$ represents the pose smoothness term that penalizes the jerkiness and spurs in the motion, E_{joint} represents the joint limit term that penalizes invalid joint angles. Joint limits come from motion capture data and human experience. The weights $\lambda_1, \lambda_2, \lambda_3$ measure the importance of each term, and are set to 1.0, 0.2, and 1000.0 respectively according to our experiments. We will describe the details of each term in the following subsection.

II.5.2.1 Data Term

The data term evaluates how well the current hand pose **q** matches the observed point cloud data **P** via analysis-by-synthesis technique. Given the hand pose **q**, we first apply Skeleton Subspace Deformation to update the position and normal for each mesh vertex to synthesize the 3D hand model. Since the observed point cloud data is partial and incomplete, we should determine which part of synthesized 3D hand model will be used for registration. The next step is to compute the registration error between the synthesized point cloud and the observed point cloud. It requires to identify the correspondence between them.

Synthesized point cloud selection. An easy option is to use all synthesized point cloud. However, this option will result in tracking failure when the hand is moving away to the camera, because the observed point cloud data may incorrectly matches to the invisible synthesized point cloud. Then another option comes into our mind that using all visible synthesized point cloud. This idea is mainly used in human body tracking [34, 35], but cannot be applied into hand tracking due to significant occlusion. The occluded synthesized point cloud has no chance to register with the observed point cloud. It will cause tracking failure when an occluded finger reappears in the following frame. In this system, we choose the similar option in [7] that using all facing-camera point cloud including the visible point cloud and the occluded but facing-camera point cloud. See Figure II.8 for more details.

Correspondence building. For every observed depth point P_i , we find the mesh vertex M_j with minimum distance to P_i as a pair of correspondence (i, j), as in Equation

$$M_j = \underset{M_k \in S_{fc}}{\arg\min} \|P_i - M_k\|^2, \tag{II.22}$$

where S_{fc} is the set of facing-camera mesh vertices. However, searching in such a large

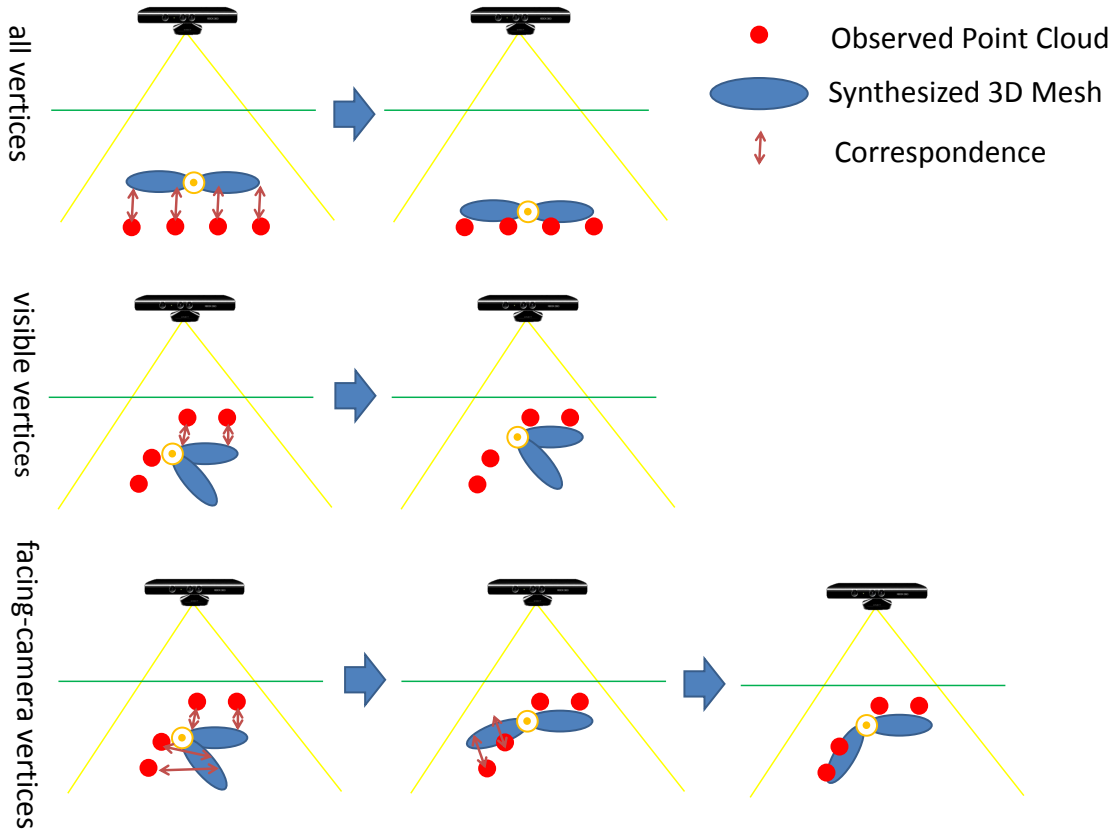


Figure II.8: Synthesized point cloud selection. The first row shows that using all vertices will cause observed data incorrectly matches to backwards vertices when the hand is moving away to the camera. The second row shows that using all visible vertices will cause tracking failure for occluded fingers. The third row shows how our method solves the issue.

set is usually time-consuming, as most vertices in the set are probably not the one that we are looking for. And therefore we build a small candidate set for an observed depth point P_i, which includes mesh vertices whose projected 2D image coordinates are close to P_i's projected 2D image coordinate. That is, we first build a P_i-centered window in 2D image coordinate, and then we build the set by including all facing-camera mesh vertices whose projected 2D image coordinate are within the window. The correspondence building process is updated once for each tracking iteration.

Implementation. Given the hand pose $\mathbf{q}$, we first apply Skeleton Subspace Deformation to update the position and normal for each mesh vertex to synthesize the 3D hand model. And then given intrinsic and extrinsic parameters of the depth camera, we render the 3D hand model into the 2D image plane via OpenGL framework. Using the shading language, we can easily store its 3D coordinate and color information for every pixel in the rendered image. By assigning a unique color to every mesh vertex, we can quickly associate a 2D pixel with a 3D mesh vertex in the rendered image according to the 2D pixel's color. In order to identify occluded but facing-camera vertices, we first compare every vertex's normal $\mathbf{n}$ with the direction of depth camera $\mathbf{n}_k$ (simply assumed to be (0,0,-1)), and then compare its depth d with the depth of its projected 2d pixel d'. Those vertices who satisfy both $\mathbf{n}^T\mathbf{n}_k > 0$ and $d > d'$ will be chosen as occluded but facing-camera vertices. For correspondence building process, for a depth point P_i, we first compute its project 2D image coordinate (x_{P_i}, y_{P_i}), and then building a window centered at (x_{P_i}, y_{P_i}). For every rendered pixel inside the window, we find its associated mesh vertex by checking its color and add it to a candidate set. Occluded but facing-camera vertices are also added to the candidate set if their projected 2D coordinates are within the window. And then we choose the mesh vertex with minimum distance to P_i from the candidate set as P_i's corresponding mesh vertex.

Generally, our data term can be represented as

$$E_{data} = \sum_{(P_i, M_j) \in S} \left\| P_i - M_j(\mathbf{q}) \right\|^2, \tag{II.23}$$

where S defines the correspondence set described as above, $M_j(\mathbf{q})$ represents the jth vertex for 3D hand model determined by the hand pose $\mathbf{q}$, P_i represents the ith observed depth point.

II.5.2.2 Smoothness Term

We add the smoothness term to penalize the jerkiness and spurs between two consecutive poses. If the previous pose is tracked successfully, δ is set to 1 and the smoothness term is to penalize the jerkiness between the current pose $\mathbf{q}_i$ and the previous pose $\mathbf{q}_{i-1}$. If the previous pose is tracked unsuccessfully, δ is set to 0 and smoothness term is to penalize the jerkiness between the current pose $\mathbf{q}_i$ and the regression output pose $\mathbf{q}_r$.

The smoothness term is defined as:

$$E_{smoothness} = \delta \left\| \mathbf{q}_i - \mathbf{q}_{i-1} \right\|^2 + (1-\delta) \left\| \mathbf{q}_i - \mathbf{q}_r \right\|^2, \tag{II.24}$$

II.5.2.3 Joint Limits Term

We add the joint limits term to penalize invalid joint poses that exceeds the range of joint movement. Each joint angle $\mathbf{q}^d, d = 7, 8, ...27$ should stay within $[\mathbf{q}_l^d, \mathbf{q}_u^d]$. The joint limits term can be represented as:

$$E_{joint} = \sum_{d=7}^{27} (\phi(\mathbf{q}^d < \mathbf{q}_l^d) \left\| \mathbf{q}^d - \mathbf{q}_l^d \right\|^2 + \phi(\mathbf{q}^d > \mathbf{q}_u^d) \left\| \mathbf{q}^d - \mathbf{q}_u^d \right\|^2), \tag{II.25}$$

where $\phi(x)$ is a binary function that

$$\begin{aligned} \phi(x) &= 1, if \ \ x \ \ is \ \ true, \\ \phi(x) &= 0, if \ \ x \ \ is \ \ false, \end{aligned} \tag{II.26}$$

Since the global translation and orientation does not have movement range, we do not add constraints on first 6 variables in $\mathbf{q}$

II.5.2.4 Optimization

The Equation (II.21) now can be represented a sum of squares, and therefore we can solve it by Gauss-Newton method. Since every term is differentiable, we can first compute the Jacobian matrix $J(\mathbf{q})$, and then follow the standard Gauss-Newton step to solve $\delta\mathbf{q}$ and update current $\mathbf{q}$:

$$\begin{aligned} J(\mathbf{q})^T J(\mathbf{q})\delta\mathbf{q} &= J(\mathbf{q})^T r(\mathbf{q}), \\ \mathbf{q} &= \mathbf{q} + \delta\mathbf{q}, \end{aligned} \tag{II.27}$$

where $r(\mathbf{q})$ the residual vector that concatenated from each term.

II.6 Results

In the section, we demonstrate the power of our approach by testing on a wide variety of hand motions using our system (Section II.6.1). Our comparison against alternative methods shows our system achieves state-of-the-art accuracy (Section II.6.2). We valid our method by evaluating the importance of shape-indexed features, and the importance of hybrid tracking (Section II.6.3). Our results are best seen in the accompanying video.

Computational timing. Our final system runs at 45fps on an Intel i7-2600K, 16GB RAM, Nvidia GeForce GTX 680 desktop with GPU acceleration. Regression tree traversal, correspondence building and Jacobian matrix evaluation are all highly parallel and can be accelerated on GPU efficiently. with 45fps, our method can perform model-based tracking to 4.2 poses per frame on average with 4 iterations per pose. Detailed timing information is summarized in Table II.1. Each regression forest includes 10 regression trees per cascade and 3 cascades in total. The tree depth is chosen to 15 experimentally. The training process takes about 48 hours without acceleration.

Component		Time(ms)
Upload depth to GPU and preprocess		1
Pose regression		4
ICP per iteration (4 in total)	Model rendering	0.6
	Correspondence and Jacobian	2.2
	Gaussian-Newton	1.5

Table II.1: Computational timing for our system

II.6.1 Test on Real Data

We have tested our system on a wide variety of hand movements, including significant global hand rotation/translation and complex finger articulation. Figure II.9 and Figure II.10 show sample frames of tracking results. We have also evaluated our system on a large number of subjects with different ages and genders. We show four of them in Figure II.11.

Figure II.9: Result: tracking 3D hand movements with significant hand translations and rotations.

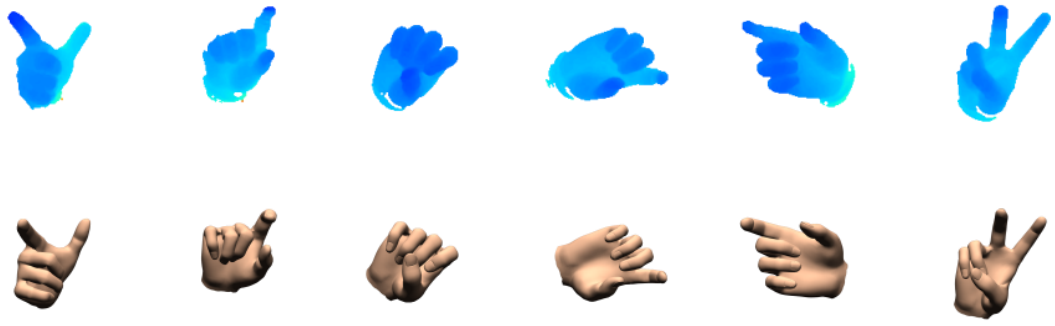


Figure II.10: Result: tracking 3D hand pose with delicate hand articulations.

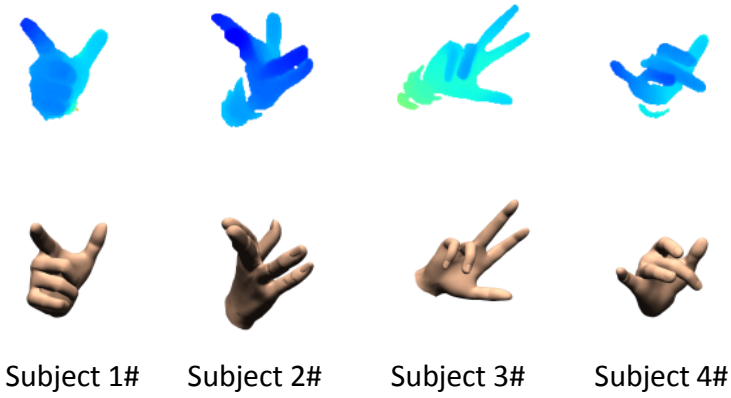


Figure II.11: Result: test on different subjects. Subject 1: male adult, 30 yrs old. Subject 2: female adult, 29 yrs old. Subject 3: young child, 12 yrs old. Subject 4: female adult, 35 yrs old.

II.6.2 Comparisons against Alternative Methods

We have evaluated our tracking system by comparing against state-of-the-art methods, including appearance-based detection methods [9, 10, 24] and model-based tracking methods [6, 7, 37].

We evaluate our method on three datasets, including our own dataset, NYU dataset [9] and Dexter dataset [44]. We define the evaluation metric based on the average success rate of all the tracking frames, which is defined as the percentage of frames that all joint errors are lower than a chosen threshold. Table II.2 summarizes the properties of three datasets.

Dataset	Capture device	Data quality	Motion description
Our data	Intel Creative Gesture Camera	Good	Significant palm rotation normal speed
NYU data	PrimeSense	High noise	Moderate palm rotation slow speed
Dexter data	Intel Creative Gesture Camera	Good	Small palm rotation normal speed

Table II.2: Description of three datasets for evaluation on different algorithms.

- Our own dataset consists of six motion sequences from a single subject captured by *Intel Creative Gesture Camera*. Our dataset is extremely challenging for hand pose tracking because the recorded data has significant palm rotations. Ground truth pose data is initialized by our tracking system and then refined by manual intervention.

- NYU dataset [9] is captured by *PrimeSense* and contains training and testing datasets from two different subjects. Ground truth data were obtained by PSO using three Kinects surrounding the subject. NYU dataset generally has slow hand movement and moderate palm rotations but contains high noise because it is captured by relatively old *Primesense* sensors. NYU dataset is challenging for tracking because of noisy input data.

- Dexter dataset [44] includes seven motion sequences from a single subject captured by *Intel Creative Gesture Camera*. This dataset has small palm rotations (most are

front-view) and normal speeds for hand movement. Ground truth data was manually labeled. Among all the three datasets, Dexter dataset is probably the easiest for tracking.

Throughout the whole comparison, our per-frame regression model is trained by our own dataset described in Section II.4.4.

II.6.2.1 Evaluation on Our Dataset

Our first evaluation compares our method against [7, 6, 10] on our own dataset.

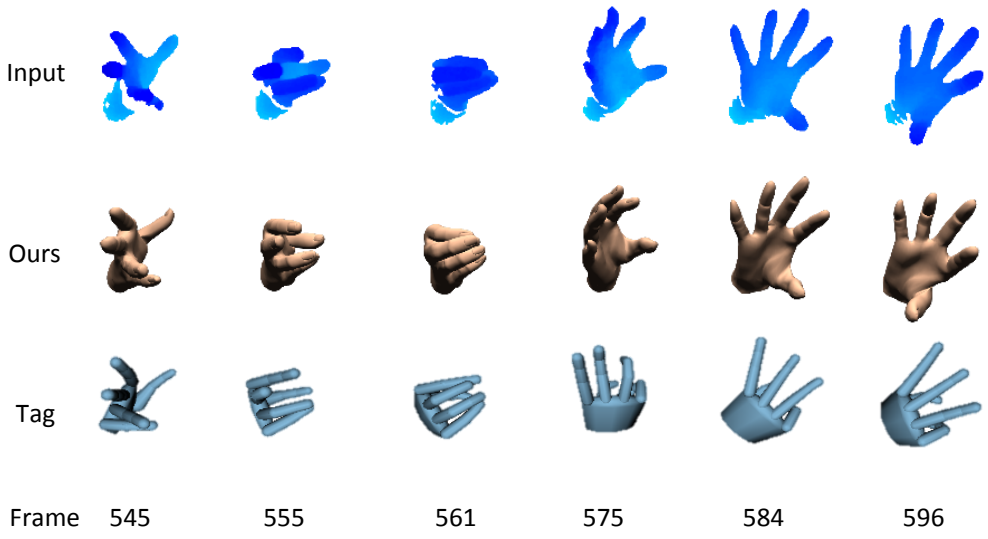


Figure II.12: Comparison against [Tagliasacchi et al. 2015]. The first row shows the depth input. The second row shows tracking results of our method. The third row shows tracking results of [Tagliasacchi et al. 2015].

Comparison against [7]. We compare our system against state-of-the-art model-based tracking [7]. We downloaded their source code and manually adjusted the hand scale to match the subject's hand size. The accompanying video shows a side-by-side comparison

between the two systems. Figure II.12 clearly shows the advantage of our system. Our system is more robust while their initializer seems to only work in front view. Note that, unlike their method, our system can automatically model a subject's hand without manual intervention.

Comparison against PSO [6]. We compare our system against model-based tracking via PSO. We implemented PSO-based tracking with 64 particles and 25 iterations for each frame. Since PSO tracking cannot initialize the pose automatically, we used the first frame of our method to initialize PSO tracking. A side-by-side comparison between the two systems is shown in the accompanying video. Figure II.13 shows sample frames of comparison results. Compared with their work, our system is more accurate and fully automatic, including modeling the subject-specific hand model and initializing the tracking pose.

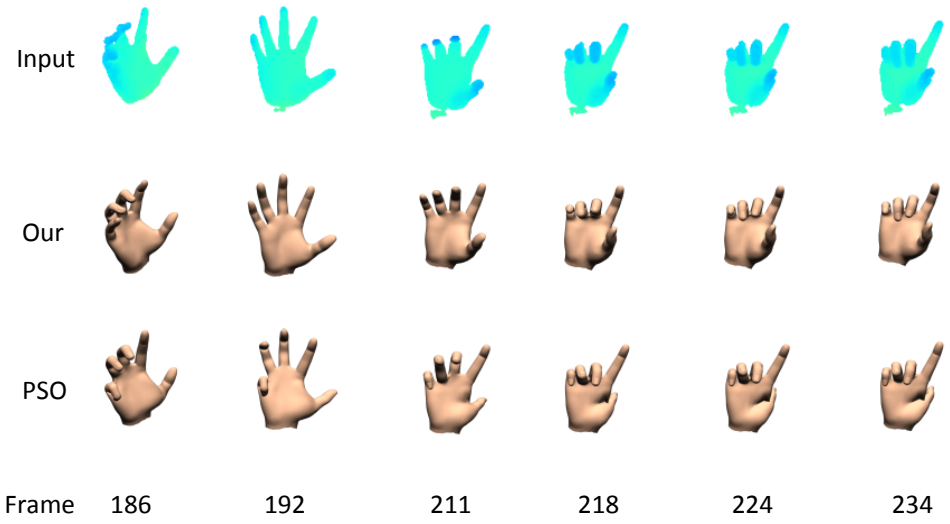


Figure II.13: Comparison against PSO [Oikonomidis et al. 2011a]. The first row shows the depth input. The second row shows tracking results of our method. The third row shows tracking results of PSO method.

Comparisons against Cascaded Pose Regression [10]. We implemented cascaded pose regression method in [10]. For our implementation, we directly used our subject-specific hand model for their regressor because the paper does not provide details about how to estimate the skeleton model for regression. We trained their pose regressor using our own training dataset. Figure II.14 shows the advantage of our system. Compared with our work, their method needs manual intervention to estimate a roughly hand scale. And their method does not consider hand shape into regression and therefore is not invariant to hand proportions and sizes of different subjects. The accompanying video shows a side-by-side comparison between the two systems. Our results are much more accurate because we complement per-frame pose regression with model-based tracking.

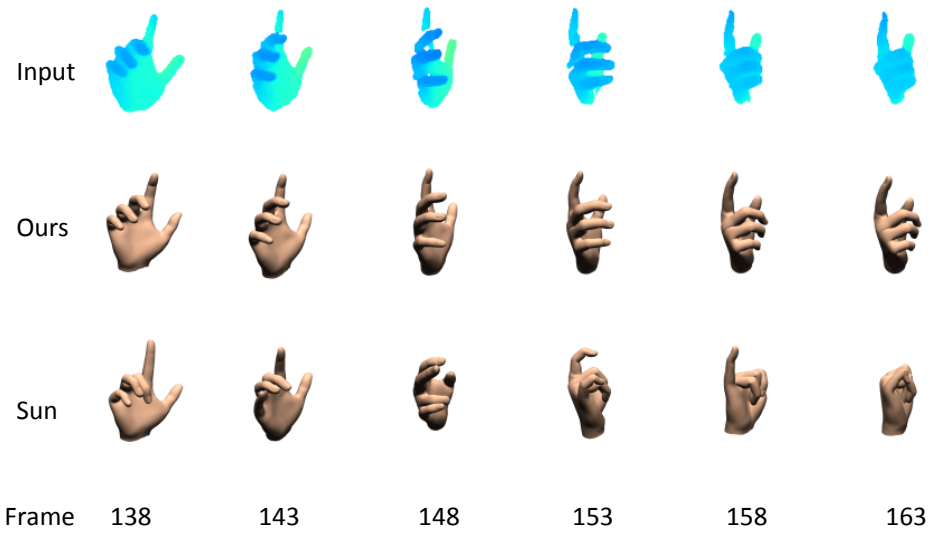


Figure II.14: Comparison against [Sun et al. 2015]. The first row shows the depth input. The second row shows tracking results of our method. The third row shows tracking results of Sun's method.

II.6.2.2 Evaluation on NYU Dataset

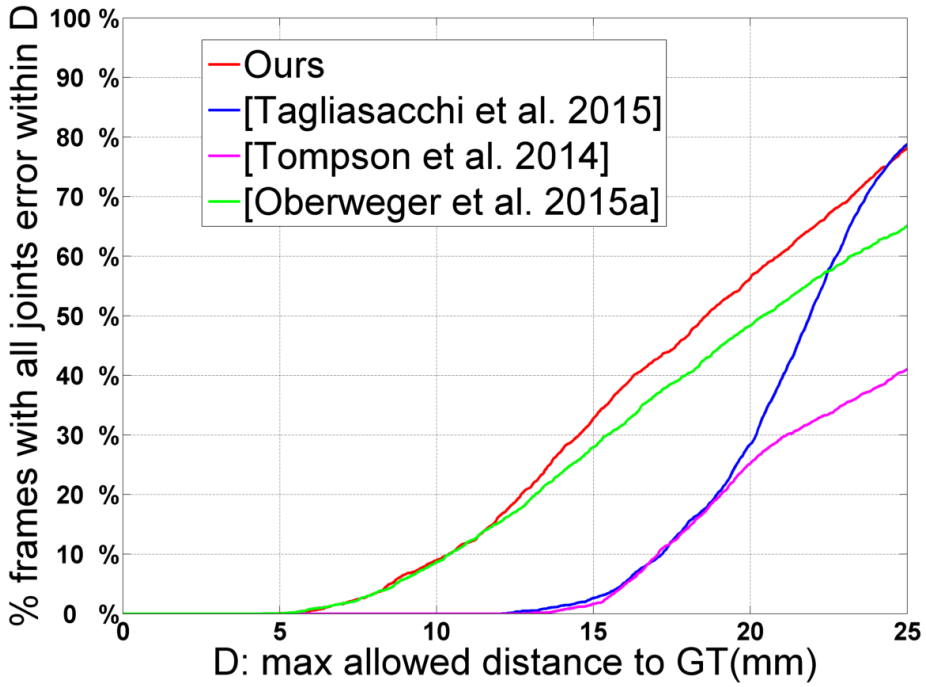


Figure II.15: Comparison against alternative methods in NYU dataset.

All authors of [9, 7, 24] have posted their result on the first 2440 frames of NYU dataset and therefore we evaluate the performance of our method on the first 2440 frames. Tompson et al.'s original result includes three joint positions on the palm that several methods find it difficult to localize. To avoid this issue, we used a common subset (10 joints) including two joints from each finger to compute the joint error in each frame. Tompson et al. only provided 2D predictions in the image coordinate and we transform their 2D predictions into 3D positions by acquiring the corresponding depth information from original depth images. If the depth data is missing, we used ground truth depth data instead, which could inevitably increase the performance of their method. The comparison result

is shown in Figure II.15. In NYU dataset, our method performs better than other three methods. In addition, compared with [24], our method is more robust because the training dataset of our method has no relationship with the testing dataset, while their method uses similar dataset for training and testing (both from NYU dataset).

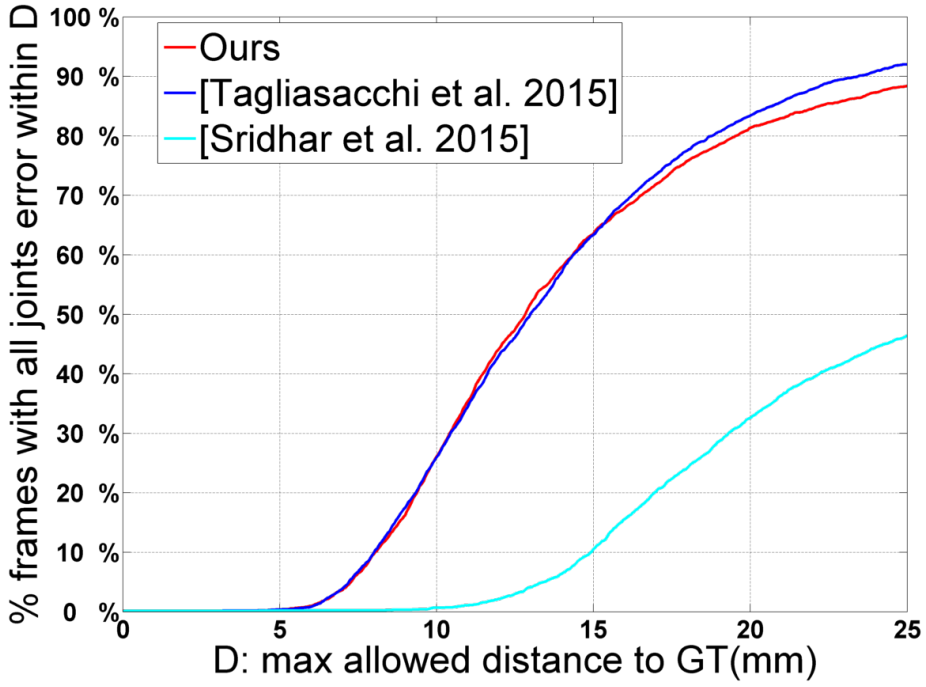


Figure II.16: Comparison against alternative methods in Dexter dataset.

II.6.2.3 Evaluation on Dexter Dataset

Both authors of [7, 37] have posted their results on Dexter dataset (3155 frames in total). In the evaluation, we use five finger tips (one per each finger) to compute joint errors in each frame, as the palm is difficult to localize in Dexter dataset. The comparison result is shown in Figure II.16. Our result is more accurate than [37] and is comparable to [7]. Note that there may exist labeling errors due to labeling inconsistency since ground

truth data was manually labeled.

II.6.3 Evaluation of Key Components

We now evaluate two key components of our system, including importance of the shape-indexed feature parameterization and the importance of hybrid tracking scheme.

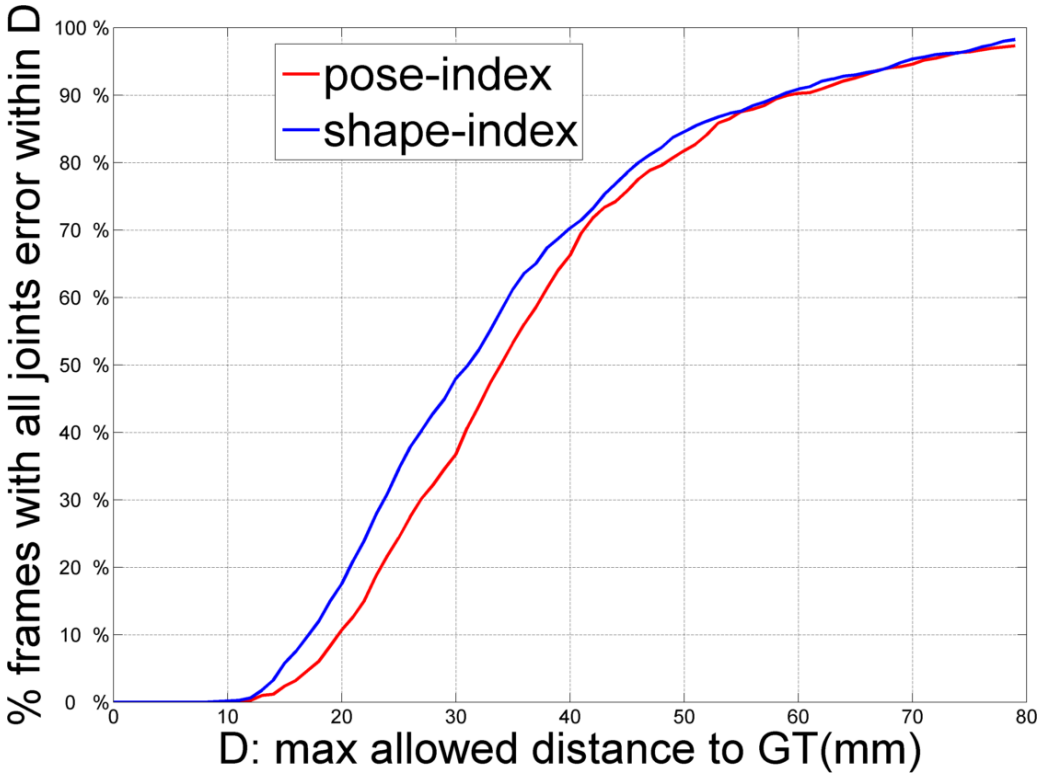


Figure II.17: Evaluation of shape-indexed features against pose-indexed features. The blue curve shows the successful rate of shape-indexed features, while the red curve shows that of pose-indexed features.

Evaluation on shape-indexed features against pose-indexed features. We first evaluate the importance of shape-indexed features against pose-indexed features by comparing results obtained by both regressors (without model-based tracking) against ground truth data. Our evaluation is based on cross validation on synthetic depth data generated by

our parametric hand model. We employ the success rate as the metric, which is the percentage of frames that all joint errors are lower than a chosen threshold. Here the joint error is defined as the distance between reconstructed joint position and the ground truth joint position. Figure II.17 clearly shows that our shape-indexed feature parameterization performs better than pose-indexed regression.

Importance of hybrid tracking. We compare our hybrid tracker against shape indexed regressor and model-based tracking (ICP). For ICP implementation, we used the first frame of hybrid tracking to initialize the beginning pose. The accompanying video shows a side-by-side comparison between the three methods. Figure II.18 clearly shows the advantage of combining model-based tracking and per-frame pose regression.

II.7 Conclusion and Limitations

In this chapter, we have presented an end-to-end realtime system that automatically and accurately tracks 3D hand poses via a single depth camera. Our system is appealing for hand motion tracking because it is low-cost, non-intrusive, and robust to large variations in hand shapes, hand poses and viewpoints and allows for accurate reconstruction of 3D hand poses even in the case of significant occlusions. We have tested our system on both live streams and recorded depth data, demonstrating its accuracy and robustness under a wide range of hand gestures and overcoming significant differences of shapes and poses across individuals. Our system achieves state-of-the-art accuracy by comparing against alternative methods.

Complementing tracking with detection not only automates the tracking process but also improves the accuracy and robustness of the whole system. Our framework for combining model-based tracking and per-frame pose detection is very flexible. We believe that any efficient model-based tracking technique such as [27, 7] can be plugged into our framework to allow for automatic and robust tracking of 3D hand poses from single-camera

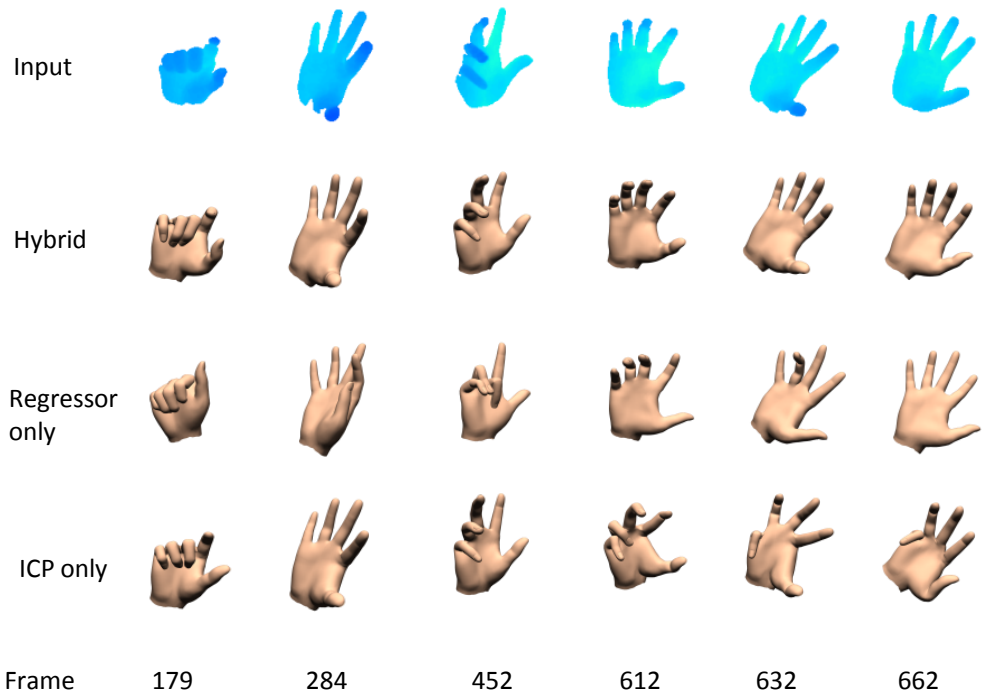


Figure II.18: Evaluation of importance of hybrid tracking. The first row shows the depth input. The second row shows reconstruction result of hybrid tracking. The third row shows reconstruction result with regressor only. The fourth row shows reconstruction result with ICP only.

depth data.

The current system has a few limitations. First, the whole system fails to produce good results when neither model-based tracking process nor per-frame detection process is able to produce good results. This often occurs when input depth data contains ambiguities caused by significant occlusions, motion blur (fast motion), or a lack of discernible features on a hand such as rotating the fist (*e.g.*, Figure II.19). Figure II.20 illustrates another concern. When previous frames fail to produce good results and no temporal information can be used to predict or filter the current frame, multiple 3D hand poses might be consistent with ambiguous observed depth data. Even though pose pool obtained by per-

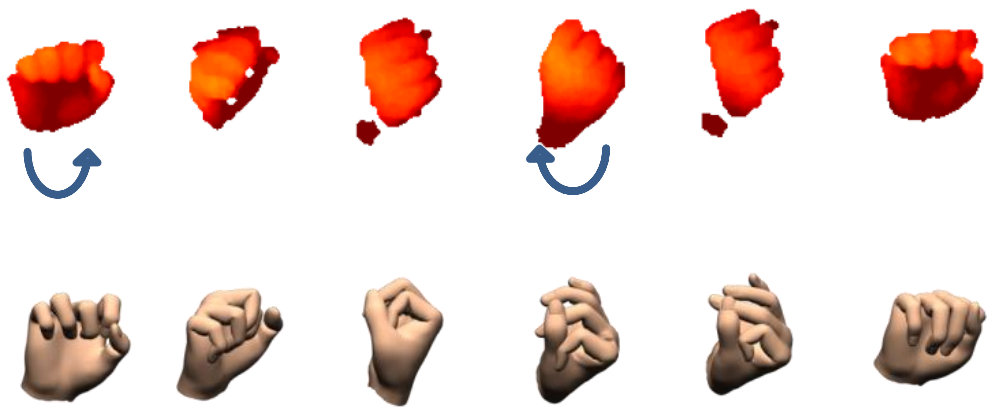


Figure II.19: Error case I: Failure when geometric features disappear.

Figure II.20: Error case II: (Left) Input depth image (Middle) Selected pose (Right) Correct candidate. Our method may select the wrong pose when no temporal information available.

frame regression includes the correct pose but noisy and ambiguous data is often not sufficient to select the correct pose as the output pose. Note that our hybrid tracking scheme can automatically recover from failure. In the future, we plan to continue improving our

model-based tracking process. As noted by prior research [45], kinematic priors learned from prerecorded motion data could be used to constrain the pose in the solution space of natural appearance to further improve the quality of reconstruction pose. In addition, we plan to construct the subject-specific penetration model for each subject according to the acquired skinned mesh to avoid finger penetration in output poses. The penetration model can be represented by a group of spheres. Then the problem is transformed to simplify a skinned mesh by a group of spheres. It will help us remove the finger penetration in output poses by adding penetration term in model-based tracking as in [7].

CHAPTER III

ROBUST REALTIME PHYSICS-BASED MOTION CONTROL FOR HUMAN GRASPING *

Human hands are capable of grasping an astounding variety of objects of different shapes, weights, frictions, and spatial orientations with little effort. However, creating physically realistic animation for human grasping is a nontrivial task. An ideal grasping action must take into account the geometry and dynamic characteristics of the object to be grasped and the selection of contact between the object and the fingers, thumb, and palm of the hand. Recent years have seen some significant advances in this area [15, 16, 17, 18, 19]. Still, the ultimate goal of building an automated realtime system that is capable of grasping a wide variety of objects with different geometry and physical quantities remains unsolved. With this goal in mind, we propose an algorithm motivated by the following principles:

Physical realism. Natural appearance is a must because people are extremely adept at judging whether an animated character appears realistic or not. A synthesized grasping motion that accomplishes an intended task might be judged as unacceptable if it appears jerky, moves like a robotic hand, or contains any unpleasant visual artifacts such as hand-object penetration. In addition, we require output animation to be physically plausible, which ensures human grasping takes into account dynamic aspects of objects crucial to human-object interaction.

Scalability. A lifelike human character must possess a rich repertoire of grasping actions and display a wide range of variation within the same action. This inevitably requires grasping objects of different shapes, weights, frictions, and spatial orientations

*Reprinted with permission from "Robust Realtime Physics-based Motion Control for Human Grasping" by Wenping Zhao, Jianjie Zhang, Jianyuan Min, and Jinxiang Chai, 2013. ACM Trans. Graph, 32, 207:1–207:12, DOI 10.1145/2508363.2508412, Copyright 2013 by ACM, Inc.

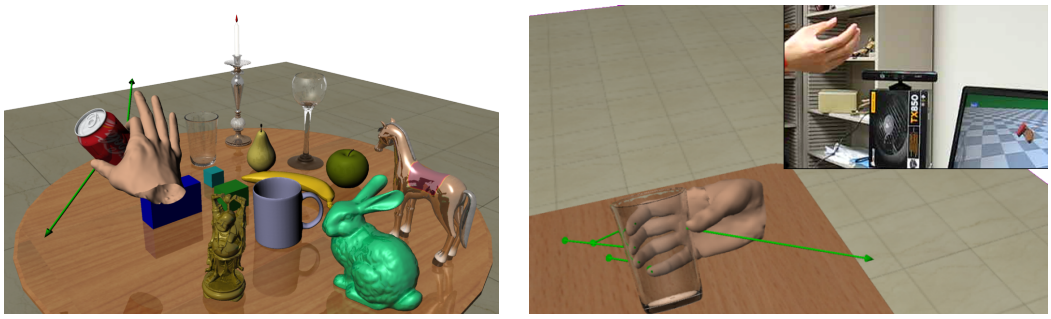


Figure III.1: Realtime generation of physics-based motion control for human grasping: (left) automatic grasping of objects with different shapes, weights, frictions, and spatial orientations; (right) performance interfaces: acting out the desired grasping motion in front of a single Kinect.

using a rich set of grip modes (*e.g.*, the power and pinch grip).

Control. An ideal animation system should empower the user with an easy-to-use interface. This is challenging since we aim to develop a system that allows a random user to generate a desired grasping action quickly and easily–with virtually no learning curve.

Realtime. Realtime animation is essential to many applications such as virtual reality, video games, and interactive animation design. The algorithm must be fast enough so that the interface appears responsive and the user remains engaged in the animation task.

The key contribution of this work is realtime generation of physics-based motion control for robust human grasping. Given an object to be grasped, our system automatically constructs physics-based motion control to achieve realistic manipulation with the object. Our solution leverages both prerecorded motion data and physics-based simulation. We first introduce a data-driven algorithm that utilizes large sets of prerecorded human grasping data to synthesize realistic animation for human grasping. The kinematic motion synthesis process runs in real time and allows the user to generate realistic, controllable animation for grasping a wide variety of objects with different shapes and sizes. Kinematic motion synthesis, however, does not consider the dynamics that cause the motion

and therefore often fails to model dynamic aspects of objects. This motivates us to develop physics-based motion control for transforming the synthesized kinematic motion into a physically realistic one in real time. In addition, we develop a performance interface for human grasping, which allows the user to act out the desired grasping motion in front of a single *Kinect* camera.

We demonstrate the power of our approach by generating physics-based motion control for grasping a wide variety of objects with different properties such as shapes, weights, spatial orientations, and frictions (Figure III.1(left)). Our interface for human grasping is intuitive and easy to use. The interface allows the user to sketch out the grasping action in greater or lesser details. The user can issue a simple kinematic control input such as "reaching the object and grasping it to move to the target location and orientation," or act out the desired grasping action in front of a single *Kinect* camera (Figure III.1(right)). In addition, we show our physics-based motion control for human grasping is robust to external perturbations and changes in physical quantities such as masses and frictions.

Contributions. Our work is made possible by a number of technical contributions:

- An efficient data-driven synthesis algorithm that utilizes a large set of prerecorded human grasping data to generate realistic, controllable animation for grasping objects substantially different from database objects.

- A robust physics-based motion control algorithm that transforms kinematic motions of the hand and object into physically realistic ones.

- A performance interface that allows the user to create a desired grasping action by acting out the motion in front of a single *Kinect* camera.

- A high-quality human grasping database for grasping a wide variety of objects with different grip styles, that will enable other researchers to apply their algorithms to

this problem.

III.1 Background

Our system combines the power of data-driven synthesis and physics-based simulation for human grasping and manipulation. We therefore focus discussion on hand motion synthesis and simulation, with an emphasis on human grasping and manipulation.

Rule-based approaches [46, 47, 48] have been traditionally used for grasp pose synthesis. This approach treats all or part of an object as a primitive shape (*e.g.*, box and cylinder) for which a grasp synthesis strategy is available. However, this approach requires solving a challenging classification problem: determining which primitive should be used as an approximation of an arbitrary shaped object. Moreover, there is no guarantee that synthesized grasp poses are natural-looking and consistent with real world observations.

An appealing alternative to grasp synthesis is to use prerecorded grasp data [14, 17, 49, 19]. This approach is appealing because synthesized grasp poses are often natural-looking and consistent with real world data. Elkoura and Singh [14] utilized a database of human grasps to process kinematically synthesized hand poses for guitar playing, so that a natural coupling between joint angles is preserved. Li and her colleagues [17] explored a data-driven approach to grasp synthesis by searching closest examples in a prerecorded grasp database to match the object shape. Amor and colleagues [49] constructed a probabilistic model for human grasp poses from prerecorded data and used it to constrain the solution space of grasp synthesis. Kyota and Saito [19] combined prerecorded grasp poses and grasp taxonomy for interactive grasp synthesis. In addition, researchers have also explored how to use prerecorded motion data to synthesize finger motions for gesturing characters. For example, Jörg and her colleagues [50] used a prerecorded database of body and finger motions to automatically add plausible finger motions to body motions.

Our approach utilizes prerecorded motion data for grasp synthesis. Our goal, however,

is different because we aim to synthesize an entire sequence of human grasping action, including reaching, closing, and manipulation, while previous approaches are focused on synthesis of either single grasp poses [14, 17, 49, 19] or freehand gestures [50]. This motivates us to develop a new data-driven model that compactly represents spatial-temporal behavior of human grasping. In addition, we combine data-driven synthesis with physics-based motion control to model dynamic aspects of human grasping synthesis. This ensures human grasping takes into account not only object geometry but also dynamic characteristics of objects such as weights, frictions, and external perturbations.

Our idea of combining prerecorded motion data and physics-based simulation for grasp synthesis is inspired by recent success on generating physically realistic human grasping from prerecorded motion data [15, 16, 51]. In particular, Pollard and Zoran [15] constructed physics-based motion control for grasping by fitting control set points from prerecorded grasping data. Kry and Pai [16] estimated joint compliances to best match recorded motion data and measured contact forces in hand-object interactions. Recently, Ye and Liu [51] combined physics-based optimization with contact-based sampling to transform a sequence of full-body poses with accurate wrist movements and a simultaneously acquired sequence of object poses into a physically plausible manipulation motion.

Our method is most similar to [15] because we both focus on physics-based motion control for human grasping. Physics-based motion control is appealing to human grasping because it can be generalized to objects with new geometric and physical quantities. However, unlike [15], our motion control is not restricted to whole-hand or enveloping grasps. Another distinction is that we complement physics-based motion control with data-driven motion synthesis. This allows us not only to grasp a wide variety of objects with different shapes, weights, frictions, and spatial orientations, but also to synthesize a desired grasping motion with intuitive and easy-to-use interfaces.

Our work is related to recent efforts on using physics-based optimization to human

grasp and manipulation [52, 18, 53]. Notably, Liu [52, 18] explored a physics-based optimization approach that synthesizes physically plausible human gasping from an initial hand grasping pose and a prescribed object motion. Our approach is different because we build our system on physics-based simulation rather than physics-based optimization. More importantly, our method does not assume any initial hand grasp poses or any prescribed object motions required by [52, 18]. Instead, we utilize prior knowledge embedded in prerecorded motion data for human grasping. Combining human motion priors with physics-based simulation not only improves the realism of synthesized motion but also reduces the time and effort required to create a desired animation.

III.2 Overview

Our goal herein is to generate robust realtime physics-based motion control for grasping a wide variety of objects with different shapes, weights, frictions, and spatial orientations. The problem is challenging because human grasping requires considering hand articulations, geometry and dynamic characteristics of the object to be grasped and the selection of contact between the hand and the object. We address the challenge by leveraging both prerecorded motion data and physics-based simulation for human grasping. Our system consists of the following major components:

Grasping data acquisition. Our system is data-driven, which requires large sets of prerecorded human grasping data for motion synthesis. We acquired a high-quality motion database for human grasping, including hand articulations, object movements, and contact information between the hand and the object, using a combination of a twelve-camera optical motion capture system [1] and two *Kinect* cameras.

Data-driven grasping modeling and synthesis. We decompose each grasping sequence into three phases, including reaching, closing, and manipulation, and develop an efficient data-driven algorithm for synthesizing each phase of human grasping. Our data-

driven motion synthesis process runs in real time and allows the user to synthesize realistic, controllable hand motion for grasping objects substantially different from database objects.

Physics-based motion control. Data-driven grasping synthesis, however, does not consider the dynamics that cause the motion and therefore often fails to model dynamic aspects of human grasping. We address this challenge by developing a robust physics-based motion control algorithm to transform synthesized kinematic motions into physically realistic ones.

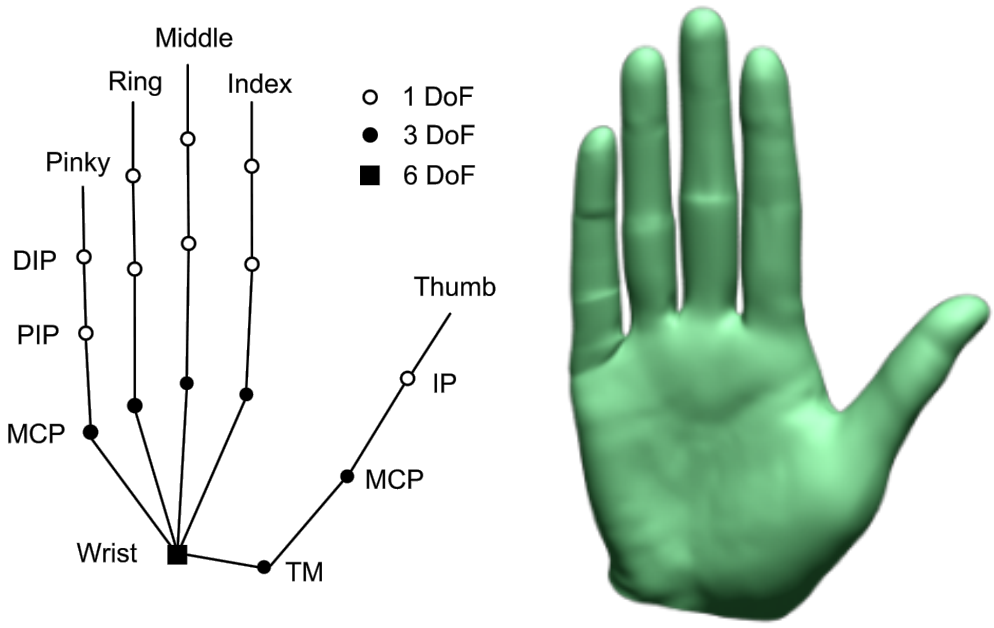
Performance interfaces for object manipulation. We develop an intuitive and easy-to-use performance interface for object manipulation. Given a virtual object to be grasped, the user acts out a desired motion in front of a single *Kinect*. The system automatically transforms the user's performance into a physically realistic motion for manipulating the virtual object.

We describe these components in more detail in next sections.

III.3 Building a Human Grasping Database

We build a human motion database for grasping a number of primitive geometries with different grip modes. Acquiring high-quality hand grasping motion, however, is challenging because it requires capturing hand articulations, object movements, and contact phenomena between the hand and the object.

Configuration space. We describe a hand pose using a set of independent joint coordinates $\mathbf{q} \in R^{33}$, including absolute root position and orientation as well as the relative joint angles of individual joints. Figure III.2 (a) shows the number of degrees of freedom for each joint. For Distal Interphalangeal (DIP) and Proximal Interphalangeal (PIP) joint, we use 1 Degree of Freedom (DoF) to describe their movement. We choose to model Metacarpophalangeal (MCP) joints using a ball and socket joint. Thus each finger has 5



(a) Skeleton model (b) Mesh model

Figure III.2: Our hand pose consists of 33 Degrees of Freedom (Dof), including absolute root position and orientation (6) and the relative joint angles of individual joints (27). (a) skeletal hand model; (b) skinned hand mesh model.

degrees of freedom (DoF) except the thumb. For the thumb, we use 1 DoF for Interphalangeal (IP) joint, 3 DoF for MCP, and 3 DoF for Trapeziometacarpal (TM) joint. So a 3D hand pose consists of 33 DoF in total. The poses of rigid body objects are defined by rigid transformations $\mathbf{o} \in R^6$. Figure III.2 (b) shows a skinned hand mesh model for hand animation.

III.3.1 Human Grasping Capture

Our motion capture process builds on the recent success of acquiring high-fidelity hand articulation data using a combination of an optical motion capture system and a single *Kinect* camera [54]. We extended their system to human grasping since their sys-

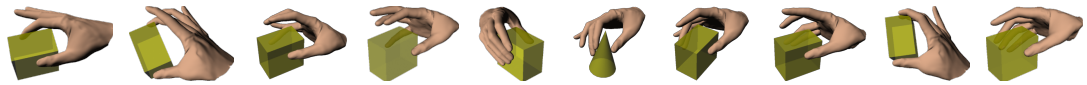


Figure III.3: Our grasping database included high-quality motion data for grasping primitive geometries with ten grip modes.

tem was focused on capturing hand articulation data alone. Briefly, we synchronized a twelve-camera *Vicon* motion capture system with two *Kinect* cameras and used them to simultaneously capture hand articulations, object movements, and contact information between the hand and object. We initialized the pose of the hand and object at the first frame and sequentially tracked 3D poses by minimizing the inconsistency between the hypothesized poses and the observed data, including both 3D marker positions obtained from the *Vicon* system and RGBD data from the two *Kinect* cameras. We employed Particle Swarm Optimization (PSO) [55] to search an optimal pose of both the hand and object over time.

Complementing marker-based mocap with RGBD data from multiple *Kinect* cameras significantly improves the reconstruction accuracy of human grasping. However, the method still suffers from significant occlusions and often fails to identify correct contact information between the hand and object. To address this issue, we instructed the human subject to perform a particular grip mode for grasping (*e.g.*, pinching grip) during motion capture sessions, manually labeled contact information throughout the whole motion sequence, and incorporated explicit contact information into the pose optimization process.

The final database included motion capture data for grasping ten different objects using ten different grip modes. The motion was reconstructed at 120 frames per second. Figure III.3 shows all the grip modes used for human grasping and Figure III.4 shows all the objects in the database.

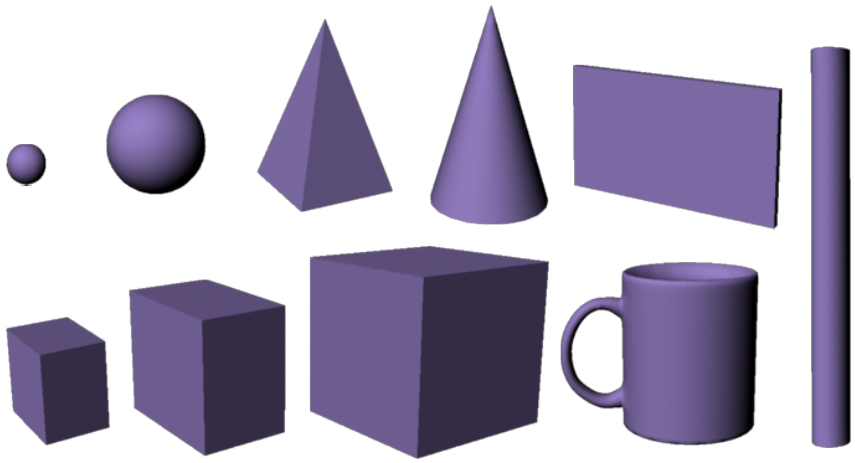


Figure III.4: Our grasping database recorded high-quality motion data for grasping ten database objects.

III.3.2 Grasping Data Processing

We decompose each human grasping sequence into three phases: "reaching" phase, "closing" phase, and "manipulation" phase. During the "reaching" phase, the hand moves to the object and opens the grip to enclose the object. For the "closing" phase, the grip closes until the object is grasped. For the "manipulation" phase, the object in grasp is moved to a target location. We segment each grasping sequence into three phases manually.

Besides, as the hand and the object may locate in different places for different capture sessions, the motion needs to be aligned together. We transform each grasping motion sequence to the object's local coordinate system, which leads to a consistent representation of human grasping data. Specifically, we build the local coordinate system in which the origin is defined by the center of mass of the object, the z axis is defined as the "up" direction perpendicular to the surface plane the object is sitting on, and the x axis is defined as the direction from the center of the hand to the center of the object of the grasping pose

(*i.e.*, the last frame of the "closing" phase). For grasping motion synthesis, we adopt the same reference system for the object to be grasped. We synthesize a motion in the local reference system of the object and then transform it back to the global coordinate system.

III.4 Kinematic Synthesis of Human Grasping

Given an object to be grasped, our kinematic motion synthesis process synthesizes motion of the "closing" phase first, then the "reaching" phase, and finally the "manipulation" phase. We prioritize "closing" motion synthesis because the last frame of "closing" motion determines the "grasping" pose which is the most important pose for human grasping. In the rest of this section, we discuss how to synthesize the motion for each phase in details.

III.4.1 Closing-phase Motion Synthesis

The goal of "closing" phase synthesis is to generate a motion that closes the grip and grasps the object. Ideally, synthesized motion should be natural-looking, collision free, dynamically feasible, and consistent with the goals specified by the user. In the following, we first apply statistical motion modeling techniques to prerecorded motion data to obtain a probabilistic motion model for closing motion synthesis. We then formulate the kinematic motion synthesis process as an optimization problem. We discuss how to solve the optimization in realtime and how to incorporate dynamic feasibility testing into the optimization process to ensure synthesized motion is dynamically feasible.

III.4.1.1 Closing-phase Motion Modeling

This section focuses on constructing a compact probabilistic motion model for "closing" phase. Our analysis algorithm builds upon deformable motion models for full-body motion synthesis [56]. We extend deformable motion models to human grasping by constructing grip-dependent deformable motion models for "closing" phase. This representation is motivated by our observation on human grasping data. While "closing" motion

displays a wide range of variations with respect to objects with different geometry and physical quantities, high-level structures of closing phase associated with a particular grip mode are always fixed. For example, closing motion using a pinch grip of the thumb and index has a consistent motion pattern and similar contact phenomena between the hand and the object. It involves simultaneously moving the thumb and index to enclose and contact the object while keeping other fingers approximately static.

For each grip mode $g = 1, ..., G$, we register all the motion segments against each other and decompose them into two functional data sets suitable for generative statistical modeling. Briefly, we pick one segment as a reference motion and use it to register the rest of segments with appropriate time warping functions. We employ dynamic time warping techniques to register all motion segments. Next, we warp each motion segment to a new motion segment in a canonical timeline defined by the reference motion using their corresponding time warping functions. This step allows us to decompose all the motion segments into two functional data sets: *warped motion segments* and *time warping functions*. Both data sets are defined in the canonical timeline and therefore are suitable for functional statistical analysis.

We apply functional PCA to all the warped motion segments associated with each grip mode. As a result, we can model a motion segment defined in a canonical timeline by the reference motion using a mean motion segment $\mathbf{p}_0$ and a weighted combination of eigen motion segments $\mathbf{p}_m, m = 1..., M$:

$$P(\vec{\alpha}) \quad = \quad \mathbf{p}_0 + [\mathbf{p}_1 ... \mathbf{p}_M]\vec{\alpha}, \tag{III.1}$$

where the vector $\vec{\alpha}$ stacks the eigen weights and the vectors $\mathbf{p}_m, m = 1, ..., M$ are a set of orthogonal modes to model geometric variations across the entire motion sequence.

Similarly, we apply functional PCA to all the time warping functions associated with

each grip mode to build a low-dimensional model for time warping functions. To preserve the property of time warping functions, which should be positive and strictly monotonic everywhere, we transform the precomputed time warping functions $w(t)$ into a new space $z(t)$: $z(t) = \ln(w(t) - w(t-1)), t = 1, ...T$ and apply the functional PCA to the transformed time warping functions in the new space. Finally, we transform the function back to the original space and obtain the following low-dimensional representation for time warping functions:

$$H(t;\vec{\gamma}) \quad = \quad \sum_{i=1}^{t} \exp(b_0(i) + [b_1(i)...b_K(i)]\vec{\gamma}), \tag{III.2}$$

where the vector $\vec{\gamma}$ is the combination weights to represent a time warping function in a low-dimensional space and the scalar $b_k(i)$ represents the i-th component of the k-th eigen vector $\mathbf{b}_k$.

After combining Equation (III.1) and (III.2), we obtain the following parametric function to model spatial-temporal variation of the "closing" motion $\mathbf{x}_g$ associated with a particular grip g:

$$\mathbf{x}_g = M_g(\vec{\alpha}, \vec{\gamma}) \ = P_g(\vec{\alpha}) \otimes H_g(\vec{\gamma}) \tag{III.3}$$

where the operator $\otimes$ warps a motion segment $P_g(\vec{\alpha})$ in the canonical time line with a time warping function $H_g(\vec{\gamma})$.

One appealing property of a grip-dependent deformable motion model is to automatically annotate contact information for generated motions. Note that the last frame of "closing" motion defines the "grasping" pose crucial to hand-object interaction. This ensures that synthesized motions are always consistent with a particular contact grip. Furthermore, we learn a joint probability density function $pr_g(\vec{\alpha}, \vec{\gamma})$ to model the correlation between the geometric and timing variations. We model the prior distribution with a Gaussian mixture

model (GMM). The parameters of the Gaussian mixture model are automatically estimated using an Expectation-Maximization algorithm [57].

The deformable model and the joint probability distribution define a generative model for closing-phase motion synthesis. We can generate an infinite number of natural-looking grasping motions for a particular grip mode by sampling the joint probability distribution $pr_g(\vec{\alpha}, \vec{\gamma})$ and warping the sampled motion $P_g(\vec{\alpha})$ in the canonical timeline with the sampled time warping function $H_g(\vec{\gamma})$. In the next section, we discuss how to use the generative motion model of "closing" phase to synthesize realistic "closing" motions for grasping objects with different shapes, sizes, and spatial orientations.

III.4.1.2 Objective Function

We formulate the motion synthesis process as an optimization problem:

$$\min_{\vec{\alpha},\vec{\gamma},g} \quad w_p E_p(\vec{\alpha},\vec{\gamma}) + w_c E_c(\mathbf{x}_g) + w_k E_k(\mathbf{x}_g) + w_d E_d(\mathbf{x}_g) + w_q E_q(\mathbf{x}_g) \tag{III.4}$$

where $\vec{\alpha}, \vec{\gamma}$ are parameters of the deformable motion model and $\mathbf{x}_g = M_g(\vec{\alpha}, \vec{\gamma})$ is a motion generated from the deformable motion model corresponding to a particular grip mode g. The first term E_p is the *motion prior* term which evaluates how well the synthesized motion is consistent with database examples. The second term E_c is the *motion control* term which ensures the synthesized motion achieves the goal specified by the user. The third term E_k is the *collision constraint* term which avoids the collision between the hand and the object. The fourth term E_d is the *contact distance* term which measures how well the contact fingers contact the object. And the last term E_q is the *grasp quality* term which measures the quality of the synthesized grasping pose. The weights w_p, w_c, w_k, w_d and w_q control the importance of each term and are experimentally set to 0.06, 0.9, 1.0, 0.4 and 2.0, respectively.

Motion prior term. The prior term, E_p, ensures the synthesized grasping motion is

consistent with prerecorded examples in the database. We minimize the negative log of the prior distribution function $pr(\vec{\alpha}, \vec{\gamma})$, yielding the following energy formulation:

$$E_p = -\ln pr(\vec{\alpha}, \vec{\gamma}). \tag{III.5}$$

Motion control term. The motion control term, E_c, ensures generated motion achieves the goal specified by the user. Specifically, it measures the inconsistency between the synthesized motion $\mathbf{x}_g$ and the user constraints $\mathbf{c}$. The system allows the user to sketch out the grasping motion in greater or lesser details. One way to control the grasping action is to specify grip modes g and/or contact point constraints for grasping fingers. Alternatively, the user could act out the desired grasping action in front of a single *Kinect* camera (Please refer to Section III.6 for details of performance interfaces).

Collision constraint term. This term ensures the synthesized hand does not penetrate the object during the entire sequence of "closing" motion. In our implementation, we penalize the penetration between the hand and the object, resulting in the following objective function term:

$$E_k(\mathbf{x}_g) = \frac{1}{N}\sum_{i=1}^{N} pene_i(\mathbf{x}_g, object) \tag{III.6}$$

$$pene_i(\mathbf{x}_g, object) = \begin{cases} 0 & \text{if no penetration} \\ d & \text{otherwise} \end{cases} \tag{III.7}$$

where N is the number of frames in closing phase and the function $pene_i$ outputs the penalty value based on the penetration between the hand and the object at frame i. In our experiment, we set the parameter d to 10.

Contact distance term. This term measures how well contact fingers contact the ob-

ject. Given a hypothesized motion $\mathbf{x}_g$, we first use forward kinematic function $\mathbf{f}$ to map the contact point on each contact finger from its local coordinates $\mathbf{p}$ to the global coordinates $\mathbf{f}(\mathbf{x}_g;\mathbf{p})$. We define the following contact term to minimize the distance between each contact point and its closest point O on the surface of the object:

$$E_d = \sum_{i=1}^{N_c} ||\mathbf{f}(\mathbf{x}_g;\mathbf{p}_i) - O_i||, \tag{III.8}$$

where N_c is the number of contact points for a particular grip model g. Note that for a particular grip mode we know the total number of contact points (N_c) and their local coordinates ($\mathbf{p}_i$) because we embed contact information into grip-dependent deformable motion models during the modeling process.

Grasp quality term. To enable physics-based motion control, unstable grasp poses (*e.g.*, grasping two adjacent surfaces of a cube) is dynamically unstable and should be avoided in kinematic motion synthesis. We evaluate the grasp quality term $E_q(\mathbf{x})$ based on a weighted combination of two grasp metrics (for more details on grasp quality, please refer to [58, 59, 60, 49]):

$$w_1 E_g^1 + w_2 E_g^2 \tag{III.9}$$

where the weights w_1 and w_2 control the importance of each term and are set to 0.06 and 4.0, respectively. Note that we evaluate the grasp quality term based on the "grasping" pose (*i.e.*, the last frame of the "closing" phase).

The first term, $E_g^1 = ||CM - C||$, measures the distance between the center of mass (CM) of the object and the centroid (C) of the contact polygon (2D) or polyhedron (3D). The contact polygon/polyhedron is the polygon/polyhedron that connects each contact point. Intuitively, the effect of inertial and gravitational forces on the grasp is minimized when the distance is minimized. A small distance often results in a more stable grasp. This

term is adopted from robotic literature [60].

The second term, $E_g^2 = \sum_{i=1}^{SP}(a_{i,1} + a_{i,2})$ is the friction cone constraint term. The cone of friction is a geometric interpretation of the maximally allowed angle ϕ between the surface normal and the applied force vector. Specifically, for each pair of antagonist fingers, we compute their nearest positions on the surface of the object and use them to define a connecting line. If this line lies within both cones of friction at the intersection points with the object surface, we regard the grasp as stable. In other words, for achieving a stable grasp with two fingers we need to minimize the angles θ_1 and θ_2 between the connecting line and the contact normals, until both are smaller than ϕ. Note that *SP* is the number of antagonist finger pairs, which depends on the grasp mode. An antagonist pair is two fingers that can exert forces in opposing directions so as to create stable grasps. For example, *SP* is "1" for pinching using the thumb and index, and "4" for power grip using all fingers. In the latter case the thumb is antagonist with all other four fingers. For more details of this term, please refer to [49].

III.4.1.3 Dynamic Feasibility Test

We further test whether a solution (*i.e.*, a grasping pose) can produce sufficient contact forces to counteract the gravitational force of the object in static equilibrium. This test is necessary for synthesizing dynamically feasible grasping motion because there is no guarantee that the optimization solution from Equation (III.4) is physically feasible even with the grasp quality term.

In our application, we model each contact patch using multiple contact points (4 in our experiment) to approximately model contact torques caused by soft bodies. Each contact force $\vec{f}_i$ is represented as a linear function of nonnegative basis coefficients $\vec{\lambda}_i$ [61]: $\vec{f}_i = \mathbf{B}_i\vec{\lambda}_i$, where the matrix $\mathbf{B}_i$ is a 3×4 matrix consisting of 4 basis vectors that approximately span the friction cone for the i-th contact force.

We conduct a dynamic feasibility test by solving the following optimization problem:

$$\arg\min_{\vec{\lambda}_i, f_i^{\perp}} \sum_i f_i^{\perp} \tag{III.10}$$

$$\text{subject to} \quad \sum_i \mathbf{B}_i \vec{\lambda}_i - \vec{G} = \mathbf{0} \tag{III.11}$$

$$\sum_i (\vec{r}_i \times \mathbf{B}_i \vec{\lambda}_i) = \mathbf{0} \tag{III.12}$$

$$\vec{n}_i{}^T \vec{f}_i^{/\!/} = 0, \forall i \tag{III.13}$$

$$\vec{f}_i^{/\!/} - f_i^{\perp} \vec{n}_i = \mathbf{B}_i \vec{\lambda}_i, \forall i \tag{III.14}$$

$$\vec{\lambda}_i \geqslant \mathbf{0}, \forall i \tag{III.15}$$

$$f_i^{\perp} \geqslant 0, \forall i \tag{III.16}$$

where $f_i^{/\!/}$ is the friction force of the i-th contact force and $f_i^{\perp}$ represents the magnitude of the i-th normal contact force. $\vec{r}_i$ is a vector that points from the center of mass of the object to the i-th contact point. Equation (III.11) and (III.12) ensure that the resultant force and torque of the object are both zeros, so that the object can be grasped in equilibrium. To ensure $\vec{f}_i = \mathbf{B}_i \vec{\lambda}_i$ is a valid contact force, we constrain the friction direction to be perpendicular to the contact normal (Equation (III.13)) and within the Coulomb friction cone (Equation (III.14)). Equation (III.16) ensures that the hand can only push, not pull on the object. In addition, we minimize the sum of normal contact forces $f_i^{\perp}$ to remove the ambiguous solutions.

The linear objective function defined in Equation (III.10), together with linear constraints Equation (III.11)–(III.16), forms a linear programming problem and can be solved efficiently by simplex methods. If a feasible solution cannot be found, we consider that the solution fails the dynamic feasibility test.

III.4.1.4 Motion optimization

We now discuss how to solve the optimization problem defined in Equation (III.4) in real time and how to incorporate the dynamic feasibility test into the optimization process.

We employ Particle Swarm Optimization (PSO) [55] to minimize the objective function defined in Equation (III.4). PSO is a population-based stochastic approach for solving continuous and discrete optimization problems. In particular, PSO optimizes a problem by having a population of candidate solutions, termed "particles", and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position and is also guided toward the global best known position in the search-space, which are updated when better positions are found by other particles. This is expected to move the swarm toward the best solutions.

We choose PSO for motion optimization because of three reasons. First, PSO is a sampling-based approach and does not demand derivative computation, which is almost impossible to evaluate for our objective function. Second, PSO is easy to parallelize and allows for a significant speed-up via GPU implementation. Third, unlike gradient-based optimization, PSO can output multiple solutions which are particularly suitable for our dynamic feasibility test.

We generate an initial population by sampling the learned prior distribution (GMMs). However, even with good initialization, the optimization might still run very slow, as each time the objective function is evaluated, collision detection needs to be done for the entire sequence of each particle (*i.e.*, the hypothesized "closing" motion). For complex objects with a large number of polygons, each optimization often takes more than ten minutes to run. We speed up the collision detection evaluation by precomputing signed distance transformation of the object [62]. We can precompute the signed distance transformation

of the object because the object is always static for the entire "closing" phase. The signed distance transformation computes the minimum distance between each 3D voxel and the surface of the object. We set the distance values to be positive for voxels outside the surface of object and negative for every inside voxel. With the precomputed signed distance field, the runtime collision detection evaluation is simplified as lookup table operations, which allow us to synthesize kinematic grasp motion in real time.

As mentioned in Section III.4.1.3, there is no guarantee that the solution obtained from the optimization is dynamically feasible. Although the dynamic feasibility test can be solved via simplex methods efficiently, testing it on each particle at each iteration will result in a heavy computational cost. To address this issue, we focus dynamic feasibility test only on the particles of the last generation. We pick the "best" particle that passes the dynamic feasibility test as the final synthesis solution. If there are no such particles, we run the optimization again, which rarely occurs in our experiment as it often indicates that the object might be too big and/or heavy to be grasped. Humans might not be able to grasp a huge object because of limited hand size and they cannot grasp a very heavy object because of joint torque limits.

Although we include the contact distance term in Equation (8), the contact fingers might still not be close enough to contact the object because of the nature of stochastic optimization. We refine the synthesized grasping motion by applying inverse kinematics (IK) to pull the contact point on the finger to the closest point on the object.

III.4.2 Reaching-phase Motion Synthesis

The goal of reaching-phase motion synthesis is to generate a realistic "reaching" motion that transitions from an initial hand configuration to the starting frame of synthesized "closing" motion. A simple way to achieve the goal is to linearly interpolate in-between motions. However, linear interpolations often fail to produce realistic movement as "reach-

ing" motion often exhibits certain characteristics. We address the issue by searching the closest example in the database and editing it to satisfy constraints specified at the starting and ending frames.

Specifically, we search the database using the initial hand configuration and the starting pose of the "closing" phase, and then apply Laplacian editing [63] to modify the closest example $\tilde{\mathbf{x}}_r$ to meet the new constraints on the boundary. Laplacian editing ensures the edited motion $\mathbf{x}_r$ satisfies the boundary conditions while preserving fine details of the original motion $\tilde{\mathbf{x}}_r$. This requires solving the following quadratic programming problem:

$$\begin{aligned} \arg\min_{\mathbf{q}} & \sum_{i=2}^{N-1} ||(\mathbf{q}_{i-1} + \mathbf{q}_{i+1} - 2\mathbf{q}_i) - (\tilde{\mathbf{q}}_{i-1} + \tilde{\mathbf{q}}_{i+1} - 2\tilde{\mathbf{q}}_i)||^2 \\ & + w_1(||\mathbf{q}_1 - C_1||^2 + ||\mathbf{q}_N - C_N||^2) \\ & + w_2||\mathbf{q}_{N+1} + \mathbf{q}_{N-1} - 2\mathbf{q}_N||^2 \end{aligned} \tag{III.17}$$

where $\tilde{\mathbf{q}}_i$ and $\mathbf{q}_i$ are the i-th pose of the closest motion $\tilde{\mathbf{x}}_r$ and the edited motion $\mathbf{x}_r$, respectively. The first term preserves fine details of the original motion (*e.g.*, 1D Laplacian coordinates). The second term ensures that the edited motion is consistent with boundary constraints, starting at the initial pose configuration C_1 and ending at the first pose C_N of the synthesized "closing" motion. The third term, where $\mathbf{q}_{N+1}$ is the second frame of the synthesized "closing" motion, ensures a smooth transition on the boundary by minimizing the velocity changes at the transition frame. The weights w_1 and w_2 are set to 10 and 0.01, respectively.

III.4.3 Manipulation-phase Motion Synthesis

During the "manipulation" phase, our goal is to move the object in grasp from an initial pose to a target pose. This requires synthesizing motions for both the hand and the object. Similar to "reaching"-phase motion synthesis, we search the prerecorded motion database

using the initial and target poses of the object and deform the closest example to interpolate the initial and target poses. This requires solving a quadratic optimization problem similar to Equation (III.17). After the object's trajectory is synthesized, the hand's trajectory can be synthesized accordingly, as we assume the hand is static with the object and the local joint angle values of the hand are unchanged during the "manipulation" phase.

III.5 Physics-based Grasping Control

This section introduces a robust motion control algorithm that supplies joint torques to drive the hand to track the reference trajectories of the hand and the object obtained from kinematic motion synthesis during each time step of the simulation. This process runs in real time and is fully automatic.

III.5.1 Hand Grasping Dynamics

The Newtonian dynamics equations for hand grasping can be described as follows:

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) + h(\mathbf{q}) = \tau + \mathbf{J}^T\mathbf{f} \tag{III.18}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ represent the joint angle poses, joint velocities, and joint accelerations of the hand, respectively. The quantities $M(\mathbf{q})$, $C(\mathbf{q}, \dot{\mathbf{q}})$ and $h(\mathbf{q})$ are the joint space inertia matrix, centrifugal/Coriolis, and gravitational forces, respectively. The vector τ represents joint torques supplied to the hand and the vector $\mathbf{f}$ represents contact forces/torques between the hand and the object. The Jacobian matrix $\mathbf{J}$ maps contact forces/torques to generalized forces.

III.5.2 Motion Control Representation

We model the total joint torque (τ) supplied to drive the hand as a combination of three terms:

$$\tau = \tau_{active} + \tau_{gravity} + \tau_{grasp} \tag{III.19}$$

where τ_{active} is the active joint torque which is used to track the reference trajectory of the hand. More specifically, this term is used to cancel components of $M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})$. The second term $\tau_{gravity}$ is the gravity compensation term which is set to exactly cancel the effects of gravitational forces $h(\mathbf{q})$. The third term τ_{grasp} is the grasp joint torque which is set to balance contact forces/torques $\mathbf{J}^T\mathbf{f}$. Note that we drop the grasp joint torque term for "closing" phase.

Active joint torque. We model active joint torque using PD-servos (Proportional Derivative). All joints in the hand have proportional derivative (PD) controllers which are active at all times. At any time instance, active joint torques (τ_{active}) of the hand are calculated as

$$\tau_{active} = k_p(\bar{\mathbf{q}} - \mathbf{q}) - k_d\dot{\mathbf{q}} \tag{III.20}$$

where $\bar{\mathbf{q}}$ is the target pose of the hand, which is directly obtained from the reference trajectory of the hand. $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the current joint pose and velocity of the hand. And k_p and k_d are gain and damping coefficients for PD controllers.

Grasping joint torque. Grasping joint torque τ_{grasp} is applied to balance contact forces from the object. In our application, we model contact forces applied at each contact patch as a combination of a contact force and a torsional torque applied at the center of pressure (COP). Torsional torques are commonly used in humanoid robotics and biomechanics community (*e.g.*, [64]) to model the resultant torque of multiple contact forces from the same contact patch. Mathematically, the resultant of multiple contact forces applied at a planar contact patch can be represented as a combination of a contact force and torsional torque applied at the center of pressure (COP)*. As a result, the net torque caused by multiple contact forces at the COP is just a torsional torque caused by multiple friction forces. One benefit of torsional torques is to enable us to approximately model the contact

*Note that COP is chosen to ensure the net torque caused by vertical contact forces at multiple points is zero.

torques caused by soft bodies, thereby producing more subtle contact phenomena between the hand and the object.

Let $\vec{f}_k$ and $\vec{\tau}_k$ be the contact force and torsional torque at a particular contact patch $k, k = 1, ..., K$. We can define the grasping joint torque to balance the contact forces and torsional torques as follows:

$$\tau_{grasp} = \sum_k \mathbf{J}_k^T [\vec{f}_k{}^T; \vec{\tau}_k{}^T]^T \tag{III.21}$$

where $\mathbf{J}_k$ are Jacobian matrices to map contact forces $\vec{f}_k$ and torsional torques $\vec{\tau}_k$ to generalized forces. This, however, requires explicitly modeling contact forces and torsional torques between the hand and the object. In the next section, we discuss how to compute appropriate contact forces and torsional torques required to track the reference trajectory of the object.

III.5.3 Contact Force Modeling

We introduce virtual forces and torques [65] to drive the simulation of the object to match the reference trajectory of the object, including the reference position data $\bar{\mathbf{o}}^p$ and reference orientation data $\bar{\mathbf{o}}^r$, both of which are obtained from kinematic synthesis process. We model virtual force $\vec{f}_v$ and virtual torque $\vec{\tau}_v$ using the following PD-servos:

$$\begin{aligned} \vec{f}_v &= k_p(\bar{\mathbf{o}}^p - \mathbf{o}^p) - k_d \dot{\mathbf{o}}^p \\ \vec{\tau}_v &= k_p(\bar{\mathbf{o}}^r - \mathbf{o}^r) - k_d \dot{\mathbf{o}}^r \end{aligned} \tag{III.22}$$

where $\mathbf{o}^p$ and $\dot{\mathbf{o}}^p$ are the current position and linear velocity of the object. $\mathbf{o}^r$ and $\dot{\mathbf{o}}^r$ are the current orientation and angular velocity of the object. Again, k_p and k_d are gain and damping coefficients for PD controllers. Unlike the hand, we cannot directly apply the above PD control to advance the simulation for the object. This is because the object is

not active and its movement is completely determined by contact forces. We, therefore, must compute appropriate contact forces/torsional torques to achieve the same effects of virtual forces/torques applied to the object.

We compute contact forces and torsional torques by solving the following optimization problem:

$$\begin{aligned} \min_{\{\vec{f}_k, l_k\}} \; & \alpha_1 \| \textstyle\sum_k \vec{f}_k + \vec{G} - \vec{f}_v \|^2 + \alpha_2 \| \textstyle\sum_k (\vec{r}_k \times \vec{f}_k + l_k \vec{n}_k) - \vec{\tau}_v \|^2 \\ & + \alpha_3 \textstyle\sum_k {f_k}^T f_k + \alpha_4 \textstyle\sum_k {l_k}^2 \\ \textit{s.t.} \quad & \vec{f}_k \text{ stays within the friction cone}, \forall k \end{aligned} \tag{III.23}$$

where $\vec{f}_k$ and $l_k\vec{n}_k$ are the contact force and torsional torque at the kth contact patch, where l_k and $\vec{n}_k$ are the magnitude and normal direction of the kth contact patch. $\vec{r}_k$ is a vector that points from the object's center of mass to the location of the kth contact point.

The first and second terms measure how well contact forces and torsional torques from all the contact patches realize the virtual force and torque required to track the reference trajectory of the object. The third and fourth terms are the regulation terms which minimize the magnitude of forces and torques. The weights α_1, α_2 , α_3 and α_4 are set to 1.0, 1.0, 0.1 and 0.1, respectively.

Again, we linearize the friction cone constraint for each contact force by using polyhedral approximation of the friction cone. Therefore, we express each contact force as a linear combination of base vectors, $\vec{f}_k = \mathbf{B}_k \vec{\lambda}_k$. This allows us to efficiently solve the optimization problem in Equation (III.23) via quadratic programming.

III.6 Performance Interfaces for Physically-based Grasping

Our performance-based interface for human grasping seeks to enforce realistic dynamic interaction with objects in the virtual world, while faithfully preserving the nuances of the actor's performance. Given a virtual object to be grasped, the user acts out a desired

grasping action in front of a single *Kinect* camera. The system automatically transforms the actor's performance into physically realistic grasping consistent with geometry and dynamic properties of the object on the fly. Figure III.5 shows several snapshots of live screen capture of performance interfaces in use. The computer screen displays the reconstructed hand motion from the performance interface to allow realtime feedback from the performer. In the following, we discuss how to reconstruct the performance of the hand using a single *Kinect* camera and how to transform them into physically realistic motion for manipulating the virtual objects.

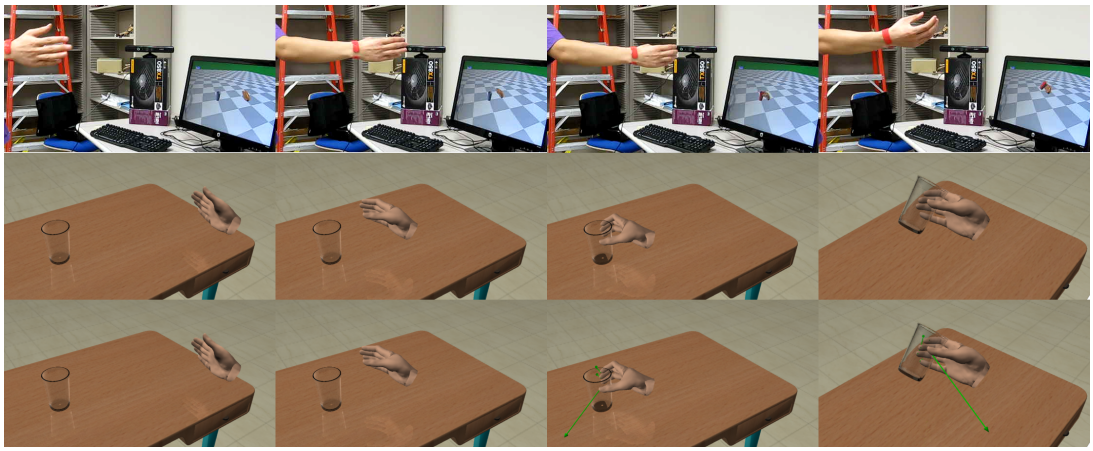


Figure III.5: Performance interfaces for human grasping: (top) live screen capture of performance interfaces in use–the computer screen displays the reconstructed motion from a single *Kinect* to allow realtime feedback from the performer; (middle) kinematic synthesis results; (bottom) physically simulated motion. Note that green arrows visualize the contact forces for each contact patch.

Realtime tracking of hand grasping. We employ a single *Kinect* camera to track the performance of human grasping. We choose the *Kinect* camera because of its low cost and simplified setup. The *Kinect* camera provides a 320×240 depth image at 30 frames per second (fps) with depth resolution of a few centimeters. Similar to [66], we formulate

the kinematic hand tracking as an optimization problem by minimizing the discrepancies between the hypothesized data and the observed data, including depth data and color skin of the hand, and employ Particle Swarm Optimization (PSO) [55] to search an optimal pose of the hand over time. To improve the robustness and speed of the tracking process, we constrain the solution space of the hand to a low-dimensional subspace automatically constructed from prerecorded grasping data via Principal Component Analysis (PCA). The tracking system runs in realtime (20 frames per second). Figure III.5(top) shows live screen captures of our tracking process.

On-the-fly motion transformation. However, hand tracking data obtained from a single *Kinect* is often noisy due to noisy depth measurement and self-occlusion caused by a single *Kinect*. Even with high-quality hand motion data obtained by a "perfect" hand tracker, the performance of the actor still cannot be directly mapped to grasp the object in an appropriate manner. Keep in mind that the actor performs without objects in the hand and therefore the performance of the hand is often inconsistent with geometry and physical quantities of the object. In practice, we have observed that contact fingers often appears not to touch the object at all or penetrate into the object during the tracking process (Figure III.6(c)). Our solution is to utilize deformable motion model of human grasping and physics-based motion control to transform noisy hand motion data from hand tracking process into physically realistic manipulation data. Figure III.6(d) and Figure III.6(e) show the synthesized kinematic motion and the simulated manipulation motion.

We generate physically realistic manipulation via performance interfaces in the following order: reaching, closing, and manipulation. For "reaching" and "manipulation" phases, we directly transform the tracking motion into physically realistic one using physics-based motion control described in Section III.5. We choose to model kinematic motion of "reaching" and "manipulation" phases directly based on tracking data rather than database examples due to two reasons. First, hand motion for "reaching" and "manipulation" phases

(a) (b) (c) (d) (e)

Figure III.6: Performance interfaces for human grasping: (a) input RGB image; (b) input depth image; (c) tracking result; (d) synthesized kinematic motion; (e) simulated motion for human grasping.

are mainly about changes of global poses and therefore are easy to track. Second for both phases, there is little interaction between the hand and object because "reaching" does not contact the object and we assume the object stays relatively static with the hand during the "manipulation" phase. For "closing" phase, before we do physics-based motion transformation, we utilize deformable motion models of human grasping to transform noisy hand tracking data of closing phase into high-quality closing motion consistent with shapes and spatial orientations of objects. This is achieved by optimizing the same objective function described in Section III.4.1. Note that for performance interfaces, the motion control term E_c in Equation (III.4) evaluates the difference between the synthesized closing motion and the tracking motion obtained from a single *Kinect*.

Our kinematic motion transformation process requires on-the-fly classification of actor's performances. We provide a simple yet very effective way to automatically recognize different phases of human grasping: "reaching", "closing", "manipulation", and "releasing". The user starts with the "reaching" phase. If the distance between the virtual hand and the object is smaller than a threshold, the object is highlighted to indicate a switch to the "closing" phase. During the "closing" phase, the user closes the grip to grasp the object. Once the grip size is found to keep unchanged for a short period of time, the "closing" phase ends and the object is recognized to be grasped and is ready to be manipulated.

During the "manipulation" phase, the user can translate and rotate the virtual object in 3D space and move it to any target location and orientation using performance interfaces, because we assume the object stays relatively static with the hand during the "manipulation" phase. Releasing is automatically detected by checking the grip size. Once the grip size is found to increase for a short period of time, the object is released from the hand.

III.7 Results

We have demonstrated the power of our method by synthesizing physically realistic human grasping for a wide variety of objects of different shapes, sizes, masses, frictions, and external perturbations. We also show our system can generate different grip modes to grasp the same object. Our results are best seen in video form.

Joint	K_p	K_d	Joint	K_p	K_d
$root_{translate}$	400.0	50.0	$root_{rotate}$	600.0	50.0
$thumb_1$	10.0	0.3	$thumb_2$	8.0	0.2
$thumb_3$	4.0	0.4	$index_1$	4.0	0.2
$index_2$	3.0	0.1	$index_3$	2.0	0.1
$middle_1$	4.0	0.4	$middle_2$	4.0	0.1
$middle_3$	2.0	0.1	$ring_1$	4.0	0.4
$ring_2$	3.0	0.1	$ring_3$	2.0	0.1
$pinky_1$	4.0	0.4	$pinky_2$	3.0	0.1
$pinky_3$	2.0	0.1			
$object_{translate}$	5.0	0.5	$object_{rotate}$	5.0	0.5

Table III.1: Parameters for physical simulation. Finger id is named in a way from the root to the finger tip. For example: $index_1$ is the index joint closest to the root. And $index_3$ is the index joint closest to the tip. All the examples have the same K_p and K_d.

Computational timing. We implement our physically-based motion control on the Open Dynamics Engine (ODE). In our experiment, we set the simulation time step to *1ms*. Table III.1 shows all the simulation parameters used in our motion control. All

of our examples run on a machine with Inter Core(TM) i7-3930K and NVIDIA Geforce GTX 680 with realtime fps. A typical timing of kinematic motion synthesis is shown in Table III.2. Note that for each example, kinematic motion synthesis of closing, reaching, and manipulation is done only once. The computational time of contact force estimation for a single simulation is about *0.4ms*.

	timing	frame number	frame rate
closing	219.41ms	8	36.5
reaching	0.80ms	49	61250
manipulation	0.15ms	35	233333
total	220.36ms	92	417.5

Table III.2: Computational timings of kinematic motion synthesis for each phase of grasping motions.

Different shapes. Figure III.7(top) shows that the system can grasp a wide variety of objects substantially different from database ones.

Different sizes. Our system allows the user to grasp objects with a wide range of sizes. Figure III.7(middle) shows the results of grasping a cube of four different sizes, including *50mm*, *70mm*, *90mm*, and *110mm*. Note that none of the cubes is in the database.

Different grip modes. The system can generate a desired grasping action with different grip styles. For example, Figure III.7(bottom) shows the cup can be grasped using both precision grasps ("pinch") and power grips.

Kitchen table. We demonstrate the power of our synthesis system by grasping objects on the kitchen table. There are five objects on the table, including an "apple", a "pear", a "'banana", a "goblet", and a "can".

Stacking blocks. We demonstrate the control accuracy of our physics-based motion

Figure III.7: Object grasping: (top) grasping objects of different shapes; (middle) grasping a cube of different sizes; (bottom) grasping a cup with different grip modes; from left to right: thumb and index finger, thumb and middle finger, all fingers except index fingers, and all fingers.

synthesis algorithm for human grasping by stacking blocks to form a particular pattern. The final pattern from our synthesis algorithm is consistent with the desired one (Figure III.8). During the synthesis process, we generate a "releasing" motion to chain the grasping motion of each block together. The "releasing" motion is synthesized in a similar way as "reaching" phase, where the nearest neighbor is searched in the database and then modified to satisfy new constraints at the starting frame. The last frame of "releasing" phase for the current block is used as the starting frame of "reaching" phase for the next

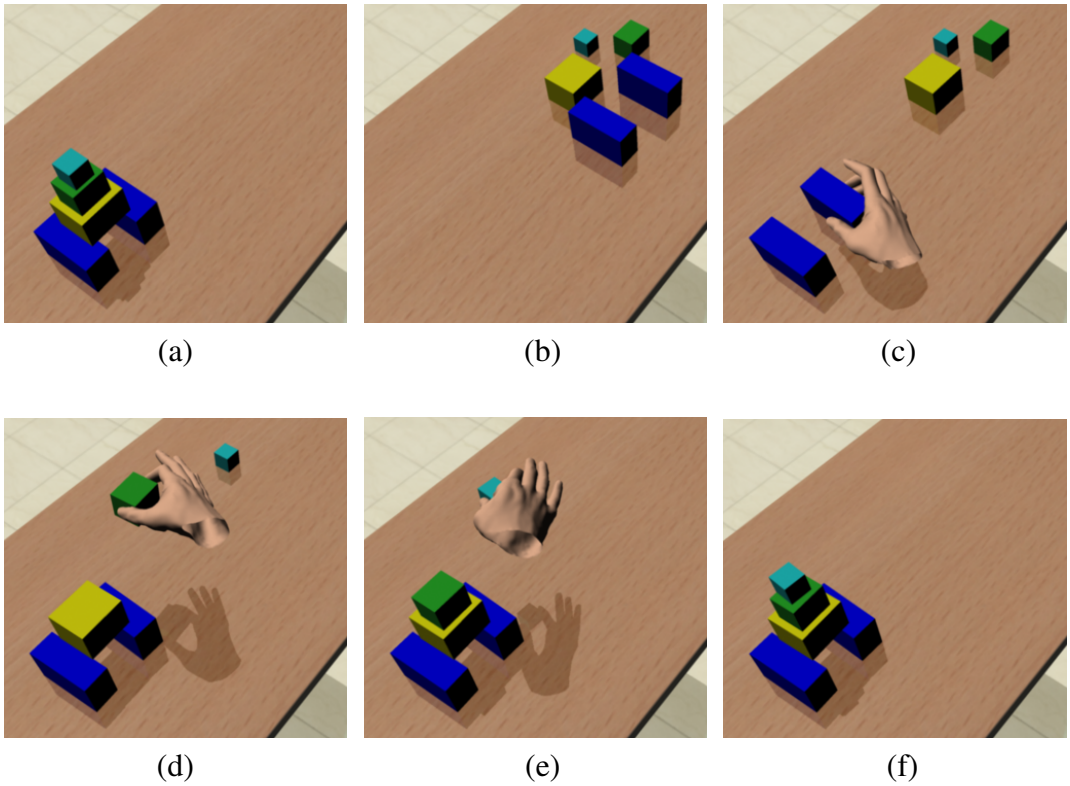


Figure III.8: Stacking blocks to form a particular pattern: (a) target blocks; (b) initial blocks; (c)–(e) grasping the objects to achieve the target pattern; (f) final blocks.

block.

III.8 Evaluation

In this section, we evaluate the performances of two key components of our system: kinematic motion synthesis and physics-based motion control. In addition, we evaluate the robustness of our system in terms of changes in physical quantities and external perturbations.

III.8.1 Evaluation on Kinematic Motion Synthesis

Our grip-dependent deformable motion model $\mathbf{x}_g = M_g(\vec{\alpha}, \vec{\gamma})$ and prior distributions $pr_g(\vec{\alpha}, \vec{\gamma})$ ensure synthesized kinematic motion of the hand is natural-looking. However,

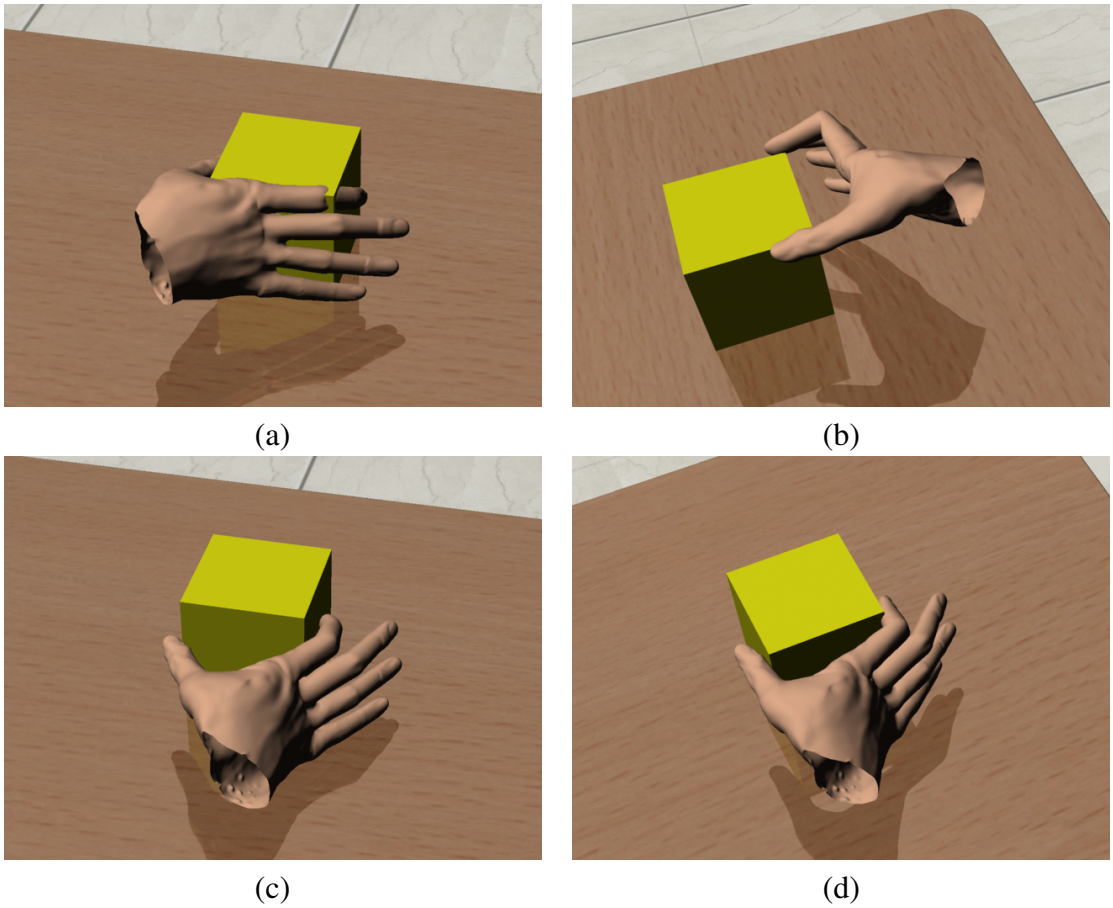


Figure III.9: Kinematic synthesis evaluation: (a) without the collision constraint term; (b) without the contact polygon term (the first grasp quality term); (c) without the friction cone term (the second grasp quality term); (d) with all terms.

in order to generate realistic human grasping consistent with the object, they need to be combined with the *collision constraint* term and *grasp quality* term. We have evaluated the importance of the *collision constraint* and *grasp quality* terms by dropping off each term in the objective function (Equation III.4). If the *collision constraint* term E_k is dropped off, the hand may collide with the object before it is grasped (Figure III.9(a)). If the contact polygon term E_g^1 (*i.e.*, the first part of the grasp quality term) is dropped off, the contact points may be too near to the object edge, which often leads to unstable grasping poses

(Figure III.9(b)). If the friction cone term E_g^2 (*i.e.*, the second part of the grasp quality term) is dropped off, the grasp becomes unstable, as can be seen in Figure III.9(c). With all terms included, the system outputs a realistic grasping motion consistent with the object (Figure III.9(d)).

As mentioned in Section III.6, hand tracking data obtained from a single *Kinect* is often noisy and inconsistent with the object to be grasped, and thereby cannot be directly used for grasping objects. Figure III.6 shows the input data from a single *Kinect*, tracked hand data, synthesized hand data, and simulated motion for human grasping. Our kinematic motion synthesis process automatically transforms noisy and inconsistent hand tracking data to realistic interaction with objects. Our evaluation video also shows that directly constructing physics-based motion control from noisy hand tracking data fails to achieve appropriate dynamic interaction with objects.

III.8.2 Evaluation on Physics-based Motion Control

To evaluate the effectiveness of our motion control and show the necessity of the To evaluate the effectiveness of our motion control and show the necessity of the grasping joint torque term in grasping motion control, we have compared against alternative methods, including trajectory tracking with PD control and physics-based motion control proposed by [15]

Comparison against trajectory tracking. We first compare our method with trajectory tracking using PD control. We implemented the trajectory tracking using open dynamics engine (ODE) by dropping off grasp joint torque (τ_{grasp}) in Equation (III.19). For trajectory tracking, the contact forces used for advancing the simulation are directly obtained from ODE. The accompanying video shows a side-by-side comparison between our method and trajectory tracking. Our motion control can achieve a stable grasp while the trajectory tracking method fails to grasp the object firmly. This is because trajectory

tracking ignores the movement of the object.

Comparison against [15]. We have compared our method against physics-based motion control developed by [15]. Our method is based on per-frame trajectory tracking, while their algorithm is based on finite state machine representation. To do a fair comparison between the two, we also extend their method to per-frame trajectory tracking. Per-frame trajectory tracking means that the target pose in their motion control is directly obtained from each frame of the reference kinematic motion, instead of interpolated by key poses from finite state machine representation in their original paper. The accompanying video shows a side-by-side comparison between our method and two alternatives. Both alternatives fail to grasp the object firmly. In contrast, our control method successfully grasps the object because we explicitly model contact forces required for tracking the reference trajectory of the object.

III.8.3 Generalization and Robustness of Motion Control

One appealing property of our physics-based motion control is its strong generalization ability. The accompanying video shows our motion control is robust to changes in geometry and physical quantities of the objects and external perturbations.

Robust to changes in geometry and physical quantities. We first show a motion control generated for grasping a "bunny" can be directly applied to grasp the same object with different frictions and weights. In the accompanying video, we also show a motion control generated for grasping a "cube" can be directly applied to grasp objects with completely different shapes such as "cylinder" and "pyramid".

Robust to external perturbations. Our motion control is robust to external perturbations. In our experiment, external perturbations are generated by impulses caused by falling balls. The video shows how our motion control reacts to different weights of the falling ball. A "light" ball only generates very subtle impacts while a "heavy" ball pro-

duces vibrations between the hand and the object.

III.9 Discussion

We have introduced a robust physics-based motion control algorithm for realtime synthesis of human grasping. We demonstrate the power of our approach by generating physically realistic motion for grasping and manipulating a wide variety of objects of different shapes, sizes, weights, spatial orientations, and frictions. Our performance interface allows the user to create a desired grasping action quickly and easily. In addition, we have shown our physics-based motion control is robust to changes in geometry and physical quantities of objects and external perturbations.

Our approach combines the power of prerecorded motion data and physics-based simulation. The use of prerecorded grasp motion data significantly reduces the solution space of our synthesis by constraining the human grasping to lie in natural appearance space. Meanwhile, incorporating physics-based simulation into the synthesis process ensures synthesized motions are physically plausible and take into account dynamic aspects of the hand and object crucial to human grasping and manipulation.

We have tested our system on grasping a wide variety of objects of different quantities. We have not yet attempted to rigorously assess when the system will fail. In the future, we would like to evaluate our system on grasping and manipulating more objects. The current system is focused on synthesizing hand motion with a single hand. We are interested in extending the system to two-hands grasping as well as synthesis of the movement of the arm and body. We would also like to extend our approach to dexterous manipulation and this inevitably requires modeling more subtle and complex contact phenomena between the hand and object. Another direction of future work is to modify and reuse the captured grasping data to achieve new tasks such as video-based hand tracking and classification.

CHAPTER IV

CONCLUSIONS AND FUTURE WORK

Realtime hand motion capture using a low-cost depth camera is a challenging but significant task that may have huge impacts on next-generation interaction techniques (gesture interaction). Compared with human body tracking, hand tracking is more difficult because hand movement has larger viewpoint changes and more self-occlusions. To deal with such challenges, we propose a novel and robust hand tracking algorithm that leverages both model-based tracking and a per-frame pose regressor. In addition, we also explore the area of the human grasping synthesis We propose a robust and realtime system that is capable to synthesize physically realistic grasping motions by utilizing large sets of prerecorded grasping motion data.

More specifically, in Chapter II, we propose an end-to-end automated hand tracking system that is capable to track 3D hand poses robustly in realtime via a single depth camera. Our system is appealing, because it is realtime, low-cost, non-intrusive (no special markers and suites) and fully automatic. To achieve this, we propose a hybrid tracking scheme that complements the model-based tracking with a shape-indexed pose regressor. In addition, to automate the full tracking process, we also propose a rapid 3D hand modeling algorithm that utilizes a low-dimensional parametric hand model. We demonstrate the power of our system by tracking a large amount of hand movements for multiple subjects. Our system also achieves state-of-the-art accuracy when compared against alternative methods. In the future, we plan to continue improving our model-based tracking process. As noted by prior research [45], kinematic priors learned from prerecorded motion data could be used to constrain the pose in the solution space of natural appearance to further improve the quality of reconstruction pose. In addition, we plan to construct

the subject-specific penetration model for each subject according to the acquired skinned mesh to avoid finger penetration in output poses. The penetration model can be represented by a group of spheres. Then the problem is transformed to simplify a skinned mesh by a group of spheres. It will help us remove the finger penetration in output poses by adding penetration term in model-based tracking as in [7].

In Chapter III, we propose a grasping synthesis method that is capable to synthesize physically realistic grasping animation in realtime. Our method leverages the power of precaptured grasping motion dataset and physics-based grasping control. We demonstrate the power of our method by synthesizing a large amount of grasping motion with different objects and different grasp modes. In addition, we also provide an easy-to-use performance interface that allows daily users to grasp a virtual object in front of a depth camera. There are two directions to extend our work. First, we would like to include two hands into grasping tasks, and there will be more contact selections and grasping dynamics to deal with. Also, with the synthesis of the body and arm movement, the whole movement will be complete. Second, the grasping synthesis is actually a simple version of manipulation, because the grasping assumes that the hand is relatively static with the object while grasping. We would like to further our research by focusing on dexterous manipulation, which inevitably requires more subtle contact modeling and planning between the hand and object.

Besides grasping synthesis, with our advanced realtime hand tracking system, we are capable to explore more areas.

- **Human Body Tracking.** The human body tracking is very similar to the hand tracking, and to some extent it is even easier than the hand tracking. Therefore, our realtime hand tracking algorithm can easily migrate into the human body tracking. In those applications that do not require high accuracy, we may also directly migrate

our shape-indexed pose regressor into the human body tracking.

- **Sign Language Recognition.** A sign language consists of a sequence of hand poses. By using HMM (Hidden Markov Model) or RNN (Recurrent Neural Network), we can build a model for every sign and recognize it in realtime. The technology will benefit to disabled people and help them communicate with normal people easily.

- **Virtual Reality.** Virtual reality devices (*e.g. Oculus rift, Sony VR headset)* is becoming popular nowadays. However, current VR devices are all headsets, which only allow using head rotation to control the viewpoint. Gesture interaction is still not added into VR devices yet. Oculus rift gives a temporary solution by using a controller (*Oculus touch*) to substitute the gesture interaction. However, it is quite intrusive for the user. With the advancement of realtime hand tracking technology, adding gesture interaction into VR devices is a necessary step to improve the user experience.

- **Smart Control.** Intelligent electrical home appliances (*e.g.*television, stream player, refrigerator) are replacing traditional ones. Voice and gesture will be main control methods for intelligent home appliances. With the realtime hand tracking, gesture control will be practical for intelligent appliances.

REFERENCES

[1] Vicon Systems, "http://www.vicon.com," 2012.

[2] CyberGlove, "http://www.cyberglovesystems.com/," 2012.

[3] Noitom Systems, "http://www.noitom.com," 2015.

[4] 3Gear Systems, "http://threegearsystems.blogspot.com/," 2015.

[5] J. Rehg and T. Kanade, "Model-based tracking of self-occluding articulated objects," in *Proceedings of the Fifth IEEE International Conference on Computer Vision*, ICCV '95, pp. 612–617, 1995.

[6] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *Proceedings of the British Machine Vision Conference*, pp. 101.1–101.11, BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.101.

[7] A. Tagliasacchi, M. Schroeder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, "Robust articulated-icp for real-time hand tracking," *Computer Graphics Forum (Symposium on Geometry Processing)*, vol. 34, no. 5, 2015.

[8] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 63:1–63:8, 2009.

[9] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Trans. Graph.*, vol. 33, pp. 169:1–169:10, Sept. 2014.

[10] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pp. 824–832, June 2015.

[11] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, and J. Shotton, "Opening the black box: Hierarchical sampling optimization for estimating human hand pose," in *Proc. ICCV*, Institute of Electrical and Electronics Engineers, December 2015.

[12] P. Dollar, P. Welinder, and P. Perona, "Cascaded pose regression," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 1078–1085, June 2010.

[13] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *Int. J. Comput. Vision*, vol. 107, pp. 177–190, Apr. 2014.

[14] G. Elkoura and K. Singh, "Handrix: animating the human hand," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 110–119, Eurographics Association, 2003.

[15] N. S. Pollard and V. B. Zordan, "Physically based grasping control from example," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pp. 311–318, 2005.

[16] P. G. Kry and D. K. Pai, "Interaction capture and synthesis," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 872–880, 2006.

[17] Y. Li, J. L. Fu, and N. S. Pollard, "Data-driven grasp synthesis using shape matching and task-based pruning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 732–747, July 2007.

[18] C. K. Liu, "Dextrous manipulation from a grasping pose," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 59:1–59:6, 2009.

[19] F. Kyota and S. Saito, "Fast grasp synthesis for various shaped objects," *Comp. Graph. Forum*, vol. 31, pp. 765–774, May 2012.

[20] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," in *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, (New York, NY, USA), pp. 63:1–63:8, ACM, 2009.

[21] R. Wang, S. Paris, and J. Popović, "6d hands: Markerless hand-tracking for computer aided design," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, (New York, NY, USA), pp. 549–558, ACM, 2011.

[22] D. Tang, T.-H. Yu, and T.-K. Kim, "Real-time articulated hand pose estimation using semi-supervised transductive regression forests," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, (Washington, DC, USA), pp. 3224–3231, IEEE Computer Society, 2013.

[23] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim, "Latent regression forest: Structured estimation of 3d articulated hand posture," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 3786–3793, June 2014.

[24] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a Feedback Loop for Hand Pose Estimation," in *Proceedings of the International Conference on Computer Vision*, 2015.

[25] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands Deep in Deep Learning for Hand Pose Estimation," in *Proc. Computer Vision Winter Workshop (CVWW)*, 2015.

[26] M. de La Gorce, D. J. Fleet, and N. Paragios, "Model-based 3d hand pose estimation from monocular video," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1793–1805, 2011.

[27] 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 *Proceedings*

of the 2011 International Conference on Computer Vision, ICCV '11, (Washington, DC, USA), pp. 2088–2095, IEEE Computer Society, 2011.

[28] W. Zhao, J. Chai, and Y.-Q. Xu, "Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, (Aire-la-Ville, Switzerland, Switzerland), pp. 33–42, Eurographics Association, 2012.

[29] I. Oikonomidis, M. I. A. Lourakis, and A. A. Argyros, "Evolutionary quasi-random search for hand articulations tracking," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, (Washington, DC, USA), pp. 3422–3429, IEEE Computer Society, 2014.

[30] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, (Washington, DC, USA), pp. 1106–1113, IEEE Computer Society, 2014.

[31] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi, "Accurate, robust, and flexible real-time hand tracking," CHI, April 2015.

[32] A. Makris, N. Kyriazis, and A. A. Argyros, "Hierarchical particle filtering for 3d hand tracking," in *IEEE Computer Vision and Pattern Recognition Workshops (HANDS 2015 - CVPRW 2015)*, (Boston, USA), pp. 8–17, IEEE, June 2015.

[33] A. Makris and A. A. Argyros, "Model-based 3d hand tracking with on-line shape adaptation," in *British Machine Vision Conference (BMVC 2015)*, (Swansea, UK), pp. 77–1, BMVA, September 2015.

[34] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Trans. Graph.*, vol. 31, pp. 188:1–188:12, Nov. 2012.

[35] P. Zhang, K. Siu, J. Zhang, C. K. Liu, and J. Chai, "Leveraging depth cameras and wearable pressure sensors for full-body kinematics and dynamics capture," *ACM Trans. Graph.*, vol. 33, pp. 221:1–221:14, Nov. 2014.

[36] S. Fleishman, M. Kliger, A. Lerner, and G. Kutliroff, "Icpik: Inverse kinematics based articulated-icp," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 28–35, 2015.

[37] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt, "Fast and robust hand tracking using detection-guided optimization," in *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[38] S. Melax, L. Keselman, and S. Orsten, "Dynamics based 3d skeletal hand tracking," in *Proceedings of Graphics Interface 2013*, GI '13, (Toronto, Ont., Canada, Canada), pp. 63–70, Canadian Information Processing Society, 2013.

[39] H. Huang, L. Zhao, K. Yin, Y. Qi, Y. Yu, and X. Tong, "Controllable hand deformation from sparse examples with rich details," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, (New York, NY, USA), pp. 73–82, ACM, 2011.

[40] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon, "Learning an efficient model of hand shape variation from depth images," in *CVPR*, Institute of Electrical and Electronics Engineers, June 2015.

[41] O. Dionne and M. de Lasa, "Geodesic voxel binding for production character meshes," in *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, (New York, NY, USA), pp. 173–180, ACM, 2013.

[42] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

[43] A. Criminisi and J. Shotton, *Decision Forests for Computer Vision and Medical Image Analysis*. Springer Publishing Company, Incorporated, 2013.

[44] S. Sridhar, A. Oulasvirta, and C. Theobalt, "Interactive markerless articulated hand motion tracking using rgb and depth data," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013.

[45] J. Chai and J. Hodgins, "Performance animation from low-dimensional control signals," in *ACM Transactions on Graphics*, 2005. 24(3):686–696.

[46] H. Rijpkema and M. Girard, "Computer animation of knowledge-based human grasping," *SIGGRAPH Comput. Graph.*, vol. 25, pp. 339–348, July 1991.

[47] R. M. Sanso and D. Thalmann, "A hand control and automatic grasping system for synthetic actors," *Computer Graphics Forum*, vol. 13, no. 3, pp. 167–177, 1994.

[48] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1824–1829, 2003.

[49] H. B. Amor, G. Heumer, B. Jung, and A. Vitzthum, "Grasp synthesis from low-dimensional probabilistic grasp models," *Comput. Animat. Virtual Worlds*, vol. 19, pp. 445–454, Sept. 2008.

[50] S. Jörg, J. Hodgins, and A. Safonova, "Data-driven finger motion synthesis for gesturing characters," *ACM Trans. Graph.*, vol. 31, pp. 189:1–189:7, Nov. 2012.

[51] Y. Ye and C. K. Liu, "Synthesis of detailed hand manipulations using contact sampling," *ACM Trans. Graph.*, vol. 31, pp. 41:1–41:10, July 2012.

[52] C. K. Liu, "Synthesis of interactive hand manipulation," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pp. 163–171, 2008.

[53] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pp. 137–144, 2012.

[54] W. Zhao, J. Chai, and Y.-Q. Xu, "Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pp. 33–42, 2012.

[55] M. Clerc and J. Kennedy, "The particle swarm–explosion, stability, and convergence in a multidimensional complex space," in *IEEE Transactions on Evolutionary Computation*, 2002. 6(1):58–73.

[56] J. Min, Y.-L. Chen, and J. Chai, "Interactive Generation of Human Animation with Deformable Motion Models," *ACM Transactions on Graphics*, 2009. 29(1): article No. 9.

[57] C. Bishop, *Neural Network for Pattern Recognition*. Cambridge University Press, 1996.

[58] A. T. Miller and P. K. Allen, "Examples of 3d grasp quality computations," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1240–1246, 1999.

[59] N. S. Pollard, "Closure and quality equivalence for efficient synthesis of grasps from examples," *International Journal of Robotics Research*, vol. 23, no. 6, pp. 595–614, 2004.

[60] R. Suárez, J. Cornellà, and M. R. Garzón, *Grasp quality measures*. Institut d'Organització i Control de Sistemes Industrials, 2006.

[61] N. Pollard and P. Reitsma, "Animation of Humanlike Characters: Dynamic Motion Filtering with A Physically Plausible Contact Model," in *In Yale Workshop on Adaptive and Learning Systems*, 2001.

[62] J. Baerentzen and H. Aanaes, "Signed distance computation using the angle weighted pseudonormal," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 3, pp. 243–253, 2005.

[63] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H. Seidel, "Laplacian surface editing," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 175–184, ACM, 2004.

[64] S.-H. Lee and A. Goswami, "Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3157–3162, IEEE, 2010.

[65] Y. Wang, J. Min, J. Zhang, Y. Liu, F. Xu, Q. Dai, and J. Chai, "Video-based hand manipulation capture through composite motion control," *ACM Trans. Graph.*, vol. 32, pp. 43:1–43:14, July 2013.

[66] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *British Machine Vision Conference*, 2011.