VISION-BASED HAND POSE ESTIMATION AND GESTURE RECOGNITION

LIANG HUI

School of Electrical & Electronic Engineering

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

2015

Statement of Originality

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

.....	.....
Date	LIANG HUI

To my family for their encourangment and support.

Acknowledgements

First and foremost, I would like to express my sincere gratitude and appreciation to my supervisor, Professor Junsong Yuan and co-supervisor, Professor Daniel Thalmann for the opportunity and guidance they provided for the whole period of my PhD study. During the past four years, they gave me a lot of constructive advices for my research, life and career plan. Their insights and suggestions help me to become a capable researcher. Their encouragement helps me to overcome the difficulties encountered in my research. Besides, I would like to thank Professor Nadia Magnenat Thalmann for her expert suggestions for my research and the financial support during the entire PhD study. The time working with them is really a great pleasure and precious memory.

I want to thank my fellow PhD students and research staffs in Professor Junsong Yuan's computer vision group, in particular Dr. Gang Yu, Dr. Gangqiang Zhao, Dr. Yang Xiao, Dr. Hongxing Wang, Dr. Ye Luo, Zhou Ren, Chaoqun Weng, Kang Dang, Jiong Yang, Chunluan Zhou, Xinchen Yan, Liuhao Ge, Bo Hu. It is really joyful to have you share your insights on research and life. I will always remember the time we work and play together.

I also appreciate all the help and friendship from my colleagues in Institute for Media Innovation and BeingThere Center. I would like to thank my fellow PhD students Junhui Hou, Jianfeng Ren, Zhenpeng Bian, Yong Chen and Juzheng Zhang here. We have a lot of fruitful discussions and a joyful period working with each other. I would like to thank Dr. Jun Lee, Dr. Aryel Beck, Dr. Flavien Picon, Dr. Zhijun Zhang, Dr. Jan Perhac, Teh Li Jun and Rubha Shri

Narayanan for all their kind support in research in BeingThere Center. I would like to thank Dr. Cao Qi, Poh Yian Lim, Tham Yiep Soon, Marilyn Zeng and Yaminn Khin for all their generous administrative support in Institute for Media Innovation. I will never forget all these great years in IMI & BTC.

I would like to thank my wife, Jia Liu. Without her company, support, love and encouragement, it is not possible for me to focus on study and research dedicatedly. My success to complete my PhD study is largely attributed to her.

Finally, I want to thank my parents Fusheng Liang and Xiumei Tian and all other family members in China, for their dedication, love and support.

Best wishes to them all!

Abstract

Real-time hand pose estimation and gesture recognition from the visual inputs are important for human-computer interaction. Compared to the specialized hardware to fulfill this task, *e.g.* the data-gloves, the vision-based methods are much cheaper and capable of providing more natural and non-intrusive interaction experiences. Despite the previous work in this field, this problem remains challenging due to the high flexibility and shape variations of the articulated hand.

This thesis focuses on inference of the full degree-of-freedom hand pose and semantic hand gestures from the visual image inputs. Particularly, the RGB-D camera is selected as the input device to record the hand images for the proposed methods in this thesis, since it can capture more detailed 3D structure of the hand and is thus less ambiguous for hand pose and gesture inference. To develop a practical solution, the various related aspects have been investigated, such as hand part recognition, fingertip tracking and hand skeleton extraction. Overall, this thesis has made the following contributions:

1. A hand parsing scheme to extract the hand parts from the depth images. It consists of a novel Superpixel-Markov Random Field to enforce both the spatial smoothness and the co-occurrence priors of the hand part labels to improve per-pixel classification results, which proves more superior to pixel-level filtering. In addition, the method generalizes well to human body parsing. In the follow-up work on hand pose estimation, the parsed hand parts prove to be effective for discriminative pose prediction by enforcing the correlations among the pose parameters. A hand gesture recognition method is proposed to

take advantage of the parsed hand parts, which reports high accuracy and robustness to hand rotation.

2. A model-based framework for hand pose estimation with continuous depth image sequence input. It adopts a divide-and-conquer scheme and combines fingertip tracking and articulated iterative closest point approach to recover the hand pose. To track the fingertip robustly, we propose several novel depth features to differentiate the fingertip and non-fingertip points and utilize the particle filter to track the fingertips through successive frames. Compared to previous methods, our proposed method can locate the fingertip position for each of the five fingertips accurately for relatively complex hand configurations. For hand pose inference, the tracked fingertip positions provide an initial estimate and an articulated ICP algorithm is utilized for further refinement.

3. A discriminative framework to predict the 3D hand joint positions from a single depth image, which addresses the self-occlusion issue encountered in our model-based framework. It enforces the hand part correlations to improve the regression forest based methods from two different aspects. First, the hand parts are utilized as the additional cue for regression. Second, a Multi-modal Prediction Fusion algorithm is proposed to fuse the ambiguous per-pixel predictions within the low dimensional hand pose manifold. This method improves the prediction accuracy considerably compared to the competing methods and is especially effective in handling the discrepancies between the synthesized training data and real-world inputs.

These proposed approaches are all capable of running in real-time or near real-time. To further exploit their potential in human-computer interaction, various applications are developed, such as hand-based communication with the virtual avatar and virtual object manipulation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer vision can serve as a powerful tool to understand the various aspects of human behaviors, such as head movement, facial expression, hand/body pose and actions. This is vital in human-computer interaction (HCI) as it provides a way for the computer to parse the raw visual inputs into a set of semantic information, making it possible for computers to react to human beings in an anthropopathic way. Among these topics, vision-based analysis of the hand movement has been an important research focus over the years due to the capability of the hands to convey rich information [34]. In the HCI scenarios, the hand motion can serve for both communicative and manipulative purposes, *e.g.*, sign language recognition [1, 60] and virtual object manipulation [2], as shown in Fig.1.1. This not only provides natural and touchless interaction experiences, but also makes it more convenient for the disabled to use and control the electronic devices. In addition, it also has various applications in film industry to capture the hand motion via computer vision techniques, based on which the animators can synthesize more realistic motions for the cartoon characters [3].

Since the hand is highly flexible, it is generally quite a challenging task to capture the precise hand motion. There have been many sensor-based methods to fulfill this task, in which the specialized hardware is used to measure the hand motion, *e.g.*, data-gloves [4] and optical sensors [5]. Although they provide quite accurate measurements, such systems are cumbersome

and expensive to use. By contrast, vision-based approaches are much cheaper, and can provide non-intrusive and natural interaction experiences. However, the vision-based approaches have their own challenges. First, the degree of freedom (DOF) of hand motion is high, which results in a large space of feasible hand pose parameters. Therefore, tracking and estimating hand postures in such a large space of hand pose parameters are hard. In a generative hand pose estimation framework, such difficulty mainly comes from the high-dimensional non-linear optimization problem involved to find the optimal pose. In a discriminative framework, the difficulty is mainly due to the ambiguity and complexity during high-dimensional regression. Since high-dimensional optimization and regression are often time consuming, the high computational complexity is also a big concern for vision-based hand pose estimation. In addition, in contrast to other articulated objects such as the human body, the hand can rotate freely in 3D space, and thus suffers from severe self-occlusion in many viewpoints in monocular inputs, *e.g.*, the fingers occlude each other or they are occluded by the palm. As a result, the estimated poses are often ambiguous due to the loss of the occluded parts. Moreover, the environment for the HCI applications is usually uncontrolled, *e.g.*, illumination variation, cluttered background, *etc.* The incurred imperfectness in hand detection and feature extraction also degrades the pose estimation accuracy. Finally, the hand is homogeneous in color and the fine details of the hand, *e.g.*, the nails or skin texture, are difficult to extract in real-world interaction scenarios. Lacking such discriminative features, mapping between the image features and the hand configuration is highly nonlinear. Due to these issues, robust hand gesture and pose estimation for unconstrained hand motion from the visual inputs remains unresolved.

This thesis investigates the various aspects of full DOF hand pose estimation & tracking and hand gestures recognition with visual image inputs, and presents several solutions that can partially handle the above challenges in terms of prediction accuracy, robustness and running speed, which include hand part recognition, fingertip tracking and hand skeleton extraction. In most of these solutions the RGB-D camera is used as the input device to record the hand

Figure 1.1: Vision-based hand pose analysis and its various applications.

images and video, since it is capable of capturing more detailed 3D structure of the hand and thus introduces less ambiguity into hand pose estimation and gesture recognition. As the main application of the investigated research topic is in HCI, this thesis also presents various related applications, such as virtual object manipulation and human-virtual avatar communication using hand.

1.1 Motivation and Objectives

Technically, one of the main focuses of this thesis is the problem of markerless hand pose estimation using the input of visual images, in which the task is to recover the kinematic parameters of the hand skeleton, including the parameters of global hand translation and rotation and local finger articulations. Since the common color images are not discriminative for hand pose estimation, we mainly use the commercial depth cameras, *e.g.*, Kinect [102] and SoftKinetics [90] to capture the hand motion, which provide fine details of the 3D hand surface. In the methodological aspect, our proposed research is largely inspired by the discriminative methods for body and hand pose estimation [41, 65, 66] developed in recent years, which shows promising results especially with the input of depth cameras. These methods have demonstrated to be fast in running speed and are quite robust to various hand and body postures. However, they still have shortcomings, *e.g.*, the results are noisy and the motion constraints of the hand are not well enforced during pose estimation.

Fig.1.2 illustrates the overview of the proposed research in this thesis, in which we address the two important problems in vision-based hand pose estimation and gesture recognition. The first problem is high-level hand feature extraction. Since certain hand parts, *e.g.*, fingertips or knuckles, are highly correlated with the hand pose, the task to infer the hand pose can be considerably simplified if they can be detected robustly from the input images. To this end, we propose several solutions to extract the high-level hand features, including 3D fingertip positions [2] and the parsed palm and finger knuckles [73]. In [2] we improve the geodesic extrema extraction algorithm with two additional depth features to differentiate the fingertip and non-fingertip points, and utilize the particle filter to track the fingertips in continuous input images. The method works for complex hand configurations such as bending fingers or multiple side-by-side fingers, but its assumptions are still restrictive. In [73] we take a fully data-driven way to parse the hand image into a set of semantic parts and exploit the spatial-temporal constraints with a novel superpixel-Markov random field, which can effectively suppress the misclassified hand parts compared to previous methods. The second problem is full-DOF hand pose estimation, in which the high-level features prove useful. In [48, 112] the tracked 3D fingertips are integrated into model-based frameworks to help estimating the finger articulation parameters via inverse kinematics, and the pose estimations are further refined by fitting a 3D hand model to the input data. The fingertips demonstrate good performance to speed up the iterative model-fitting procedure and fast recovery from tracking failure. However, since the cues for fingertip detection is empirically determined and ad hoc, the method is not always reliable, especially when the hand is going through large viewpoint changes and suffers from long-term multiple finger interaction and self-occlusion. To address this issue, in [42] we present a discriminative framework for hand pose estimation. With a large annotated dataset of hand postures, a regression forest is trained for robust prediction of the 3D joint positions against self-occlusion, and its discriminative power is largely enhanced by utilizing the parsed hand parts as additional features. Compared to the empirically determined cues in [48, 112], such a fully data-driven

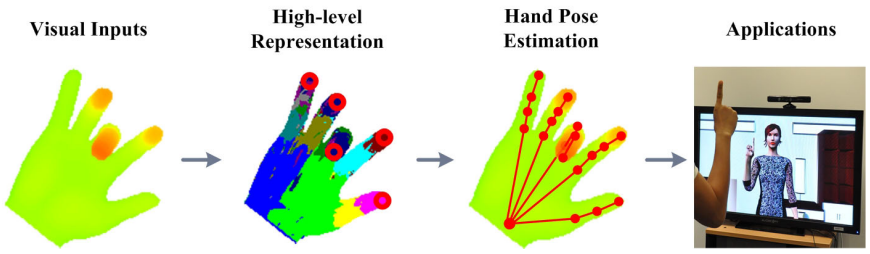


Figure 1.2: Overview of the proposed research on vision-based hand pose estimation and gesture recognition.

method can better learn the mapping from image features to the hand poses and can thus handle much more complex hand configurations. However, the hand motion constraints are not fully exploited in previous regression forest based methods, we further propose a Multi-modal Prediction Fusion algorithm so that a low dimensional constraint is enforced on the hand pose parameters when fusing per-pixel votes, which proves especially effective when making up the gap between synthesized training data and real-world testing images. Based on the algorithms developed above, we have built various real-time systems to enable human-computer interaction with bare hands, such as virtual object manipulation and hand gesture recognition.

1.2 Thesis Contribution

The major contributions of this thesis are summarized as follow:

1. A Superpixel-Markov Random Field for efficient hand parsing, which enforces depth discontinuity, spatial smoothness and co-occurrence priors of hand part labels for inference.

2. A model-based framework for hand pose estimation, which exploits the fingertip positions for fast initialization and recovery for hand pose tracking. A robust fingertip tracking scheme is presented to improve the geodesic extrema extraction method by more discriminative fingertip patterns.

3. A discriminative framework for hand pose estimation. A Multi-modal Prediction Fusion

algorithm is presented to aggregate the votes from spatially-distributed voting pixels, in which the hand motion constraints are efficiently enforced among the per-pixel predictions during inference.

4. Various real-time human-computer interaction applications built upon the developed hand pose tracking and gesture recognition algorithms.

1.3 Organization of the Thesis

This chapter has introduced the background, motivations and contributions of the thesis. The remaining contents of this thesis are organized as follows:

Chapter 2. This chapter presents a detailed literature review of existing works in vision-based hand pose estimation, which mainly includes visual hand feature extraction and full-DOF hand pose estimation methods. The various low-level and high-level hand features are reviewed and analyzed in terms of their application in hand pose estimation. The general pipelines for discriminative and generative hand pose estimation methods are presented and their key techniques are reviewed.

Chapter 3. This chapter presents a robust hand parsing method to extract the high-level hand parts such as fingers and palm from the depth images. It utilizes a novel Superpixel-Markov Random Field (SMRF) to enforce the spatial smoothness and the label co-occurrence prior to remove the misclassified regions. Compared to pixel-level filtering, the SMRF scheme is more suitable to model the misclassified regions. A depth-context feature based on a distance-adaptive sampling scheme is also proposed, which proves more effective for hand parsing compared to the previous binary depth comparison feature. In our follow-up work on discriminative hand pose estimation, the parsed hand parts prove to be effective for discriminative pose prediction by enforcing the correlations among the pose parameters. In addition, a rotation-invariant hand gesture recognition algorithm is proposed to take advantage of the hand parsing method,

which achieves high accuracy in real-time interaction.

Chapter 4. This chapter presents a model-based framework for hand pose estimation. It relies on a novel 3D fingertip tracking algorithm for fast pose initialization and recovery, which extends the geodesic extrema with two extra depth-based features to classify the fingertip points and combine them with the particle filter for robust tracking. The method works robustly for challenging cases such as bending and side-by-side fingers. In addition, a divide-and-conquer scheme is proposed to estimate the global and local hand poses successively to handle the high dimensionality of the pose parameters. Two virtual reality applications are built upon the hand pose estimation algorithm to allow the users to interact with the virtual environment, including virtual object manipulation and simulated water-oscillator instrument play.

Chapter 5. This chapter presents a discriminative framework for hand pose estimation, which can predict the 3D hand joint positions from the depth images. Compared to our model-based framework that heavily relies on fingertip detection, this framework is more robust to hand viewpoint variations and self-occlusion. In addition, the hand parsing algorithm is exploited to improve the pose prediction accuracy. The research focus of this framework is to address the inconsistency between the real-world inputs and the synthesized training datasets by enforcing the hand part correlations. Two improvements are made to the existing regression forest. First, the semantic hand parts are used as additional cue to improve the discriminative power of the regression forest. Second, a Multi-modal Prediction Fusion algorithm is proposed to fuse the per-pixel predictions on a pre-learned low dimensional hand pose space. The method proves to generalize well to different users with varying hand sizes.

Chapter 6. This chapter summarizes the previous chapters and presents the concluding remarks of the thesis. The limitations of the proposed schemes and possible future work are discussed.

Chapter 2

Related Work

This chapter focuses on the related techniques involved in vision-based hand pose estimation. Following the conventions in literature, the term hand pose can refer to either the joint positions [9, 10] or the joint angles [6–8] of the hand skeleton, and Fig. 2.1 shows an example for each of the two conventions.

Typically, a vision-based hand pose estimation system consists of three parts: hand localization and segmentation, feature extraction, and hand pose inference, as illustrated in Fig. 2.2. The localization and segmentation step tries to locate and determine the boundary of the hand in the frame accurately. The feature extraction part finds discriminative information in the image, which will then be used for hand pose inference. The pose inference step determines the optimal hand pose that best matches the input hand features, which can be solved by either treating each of the input frames independently or tracking the pose over successive frames. The former class of methods tries to solve the problem by assuming equal prior for each feasible hand configuration. The latter methods formulate the pose estimation problem as continuous estimation of pose parameters from a sequence of images, which helps to reduce the search space based on the pose estimation results from the previous frame. That is, pose estimation for the current frame is based on predictions made by applying a dynamic model to the previous pose estimation.

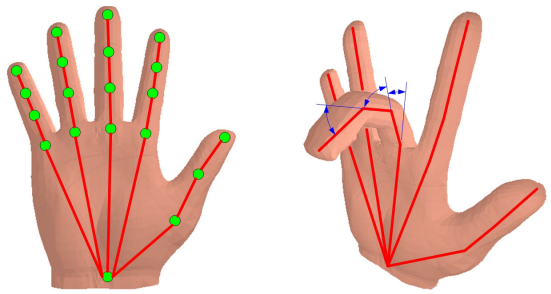


Figure 2.1: The definitions of hand pose. Left: the joint positions of the hand skeleton, denoted as green circles. Right: the joint angles of the skeleton. Here we only show the joint angles of the middle finger for clarity.

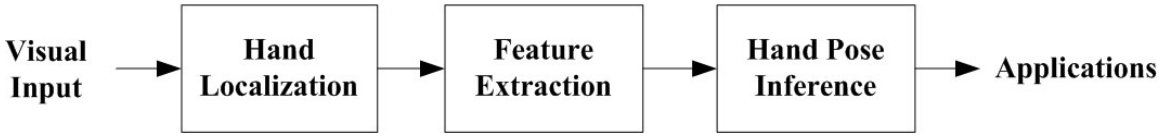


Figure 2.2: Vision-based hand pose estimation pipeline.

The following part of this chapter mainly focuses on low-level and high-level hand feature extraction and hand pose estimation and tracking techniques, which are closely related to the topic of this thesis. Since the temporal tracking techniques are mostly general that they apply to many vision problems, this chapter lays stress on single-frame based pose estimation techniques and does not cover the details in pose tracking.

2.1 Feature Extraction

The quality of the features extracted from the visual inputs plays a key role in analyzing the hand movement. We categorize the commonly used hand features into low-level features and high-level features. The low-level features, *e.g.*, edges, silhouette and skin color, can be extracted efficiently and robustly, and have been widely used for hand pose tracking and gesture recognition. However, their discriminative ability is quite limited. The high-level features, *e.g.*, the fingertips and labels of five fingers and palm parts, provide more compact representations of the raw visual inputs. It has been shown that these features are more efficient for hand movement

analysis [11, 12]. However, it is quite challenging to extract the high-level features robustly. Usually, we need to consider the specific system requirement and deployment environment to select the features for hand pose inference. For instance, for animation synthesis, the hand pose parameters need to be recovered as accurately as possible, and the capturing environment can be highly controlled. In such condition, the fine details of the hand can be used as the features, *e.g.*, the hand skin texture and fingertip. In contrast, for real-time interaction with an ordinary camera in unconstrained environment, such fine hand details is difficult to capture and hand pose needs to be recovered at low time costs. In this case, the simple binary silhouette may be more favorable. One can select one of the features or a combination of multiple features according to the system settings.

2.1.1 Low-level Features

The low-level image features, such as edges, silhouette, skin color and texture and raw depth, are widely adopted in many hand pose estimation systems. These features can be either used directly to infer the hand pose or be further processed to obtain the high-level hand features. Fig. 2.3 illustrates some examples of them. In contrast to the high-level features, they do not require much prior knowledge of the hand shape to extract and can be obtained relatively robustly and efficiently with basic image processing techniques. However, their correlation with pose parameters is generally weak and mapping from the low-level features to hand pose is not straightforward.

Edges and silhouettes are universal features for pose recovery in color images [22, 26, 115]. Edge contains rich information about the external hand shape and the internal finger boundaries, and thus can be useful to separate the hand region from the background as well as to determine the finger configurations. It can be obtained by applying the various edge operators to the images, such as the Sobel and Kirsch operators, and proves effective in a lot of previous systems for hand pose recovery [15, 26]. To recover the hand pose, the hand model edges can be

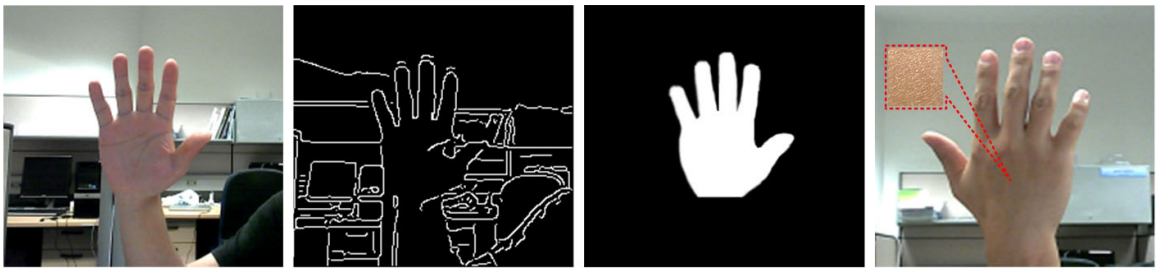


Figure 2.3: Low-level color features for hand pose estimation. First: input color image. Second: edge features. Third: hand silhouette. Fourth: skin texture.

matched to the extracted edges by gradually adjusting the model parameters in the model-based fitting frameworks [13, 22], and fitness between the image inputs and the hand pose hypothesis is usually evaluated based on the distance between the image edges and the projected hand model edges. However, the hand and finger edges are mostly confused with the non-informative edges for pose inference, *e.g.*, edges from the background clutter or the shading and texture on the hand, as shown in Fig. 2.3. Although additional cues, *e.g.*, the edge orientations [25], can partially improve robustness to the image clutters, it is still not reliable enough to infer unconstrained hand pose with edgee only. In many frameworks, edges are used in combination with other visual cues such as color and silhouette [13, 14, 27].

Hand silhouette describes the boundary of the hand region and gives coarse clues for the hand orientation and finger articulations *e.g.*, a protrusion of the silhouette probably indicates a stretching finger. Skin color detection is one common technique to extract the hand silhouette in color images [10, 21], which can be combined with edges and optical flow to differentiate from other objects of similar color [24]. These silhouette extraction methods work well in controlled environment, but are still sensitive in general HCI scenarios. In the depth images, extracting hand silhouette is relatively easy. For instance, it can be obtained by simply threshold of the depth value [2, 29] in certain depth ranges. Instead, the body part contexts can also be used to segment the hand region, *e.g.*, the current configuration of the body parts like arm and torso. For example, the positions of the body joints from Kinects skeleton tracking can be used

to locate the hand [30], in which a depth threshold range is adaptively determined for hand segmentation with the tracked 3D hand and wrist positions from the Kinect sensor. Using such prior information of the hand positions, the hand silhouette can be obtained much more robustly than using pre-determined depth threshold.

Compared to edges, silhouette is less discriminative since the inner shapes of the hand are ignored, *e.g.*, the edges of the bending fingers. Therefore, silhouette is mostly used as a supplementary cue to other image features such as edge or depth for pose estimation in the model-based fitting framework [13,14,22,28] by constraining the model projection of the hypothesized hand pose to stay within the extracted silhouette. The matching quality can be evaluated via the overlapping area between the model and image silhouettes. Silhouette can also be utilized to find the hand pose by template matching within a pre-defined dataset. However, due to the ambiguity in silhouette-based mapping, these methods only work well for small dataset [31] or are used as an initialization stage before the time-consuming model-fitting procedure [7, 32]. In some works [33, 35] multi-view silhouettes are combined to resolve the ambiguity in single views and thus can infer hand pose for more general hand motion.

Unlike other body parts, human hands are not covered with clothing at most times. Therefore, the skin color is another useful cue to locate the hand region and model the hand surface. The skin color models can be learned either parametrically or non-parametrically [17,18] using training data annotated with skin/non-skin classes. Previously the skin is modeled mostly by the color values of the pixels, *e.g.*, RGB, YCbCr and HSV. In [17] it is shown the non-parametric methods, *e.g.*, normalized lookup table [18] and self-organizing map [19], are relatively robust to the choices of different color spaces, while the parametric methods, *e.g.*, the Gaussian mixture model [20], are more sensitive. Since the color value is still not discriminative enough to describe the skin, researchers have also exploited the local skin texture to increase separability of skin/non-skin classes [21], which demonstrates improved robustness. Overall, in the uncontrolled environment with ordinary RGB camera inputs, the skin color is still sensitive to

lighting condition variations and can be easily confused with background clutters of similar color, and the detailed skin texture are difficult to capture. As a result, it is usually used to segment the hand silhouettes from the background by per-pixel skin/non-skin classification in such conditions, which is then used for pose recovery [10, 35]. When the hand motion capture environment is well controlled and high-quality images are available, skin can also be modeled in more detail. In [36] the texture and shading of the skin are captured from the input images and synthesized in the hand model, and the illumination sources are controlled in real scenario and simulated during hand modeling. In this way hand pose can be recovered quite accurately since matching ambiguity is largely reduced.

Depth cameras are powerful tools to recover the hand pose as the 3D data can often capture the subtle aspects of the hand surfaces and the detailed finger articulations, as shown in Fig. 2.4. While it is difficult to recover the accurate 3D hand pose in color images, *e.g.*, the bending angle of a finger, such depth ambiguity can be resolved with the depth camera. Moreover, with depth information, hand can be segmented more easily from the background since the depth image is invariant to color and illumination variations. The raw depth image itself has been shown effective to recover body and hand pose and gesture [39, 40, 122, 123], and the depth difference between input images and model hypothesis can be directly used as a good measure to seek the optimal pose. Another category of depth features are the local depth patches, *e.g.*, the depth comparison feature [9, 41] and the depth context [42], which prove very successful when combined with spatial pooling for both full-body and hand pose recovery. Their success is largely attributed to the fact that they are simple and efficient to calculate and effective to capture the local structure of the articulated body, based on which fast and robust pose inference is possible.

There are also various other low-level features to analyze hand pose and gesture, such as the shape context [23, 25], the Fourier descriptor [24], the orientation histogram [37] and histogram of 3D facets [38], which encode the high dimensional raw images into descriptors of much

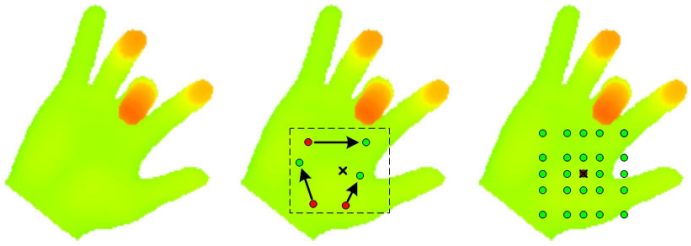


Figure 2.4: Left: raw depth image. Middle: depth comparison feature [41]. Right: depth context feature [42].

lower dimension for template matching. Although these features prove effective in gesture recognition, in which a relatively small alphabet is usually involved, they are not commonly used in full-DOF hand pose estimation. Therefore, we do not go into their details in this thesis.

2.1.2 High-level Features

Compared to the low-level features, the high-level features such as the fingertips, anchor points on palm and wrist and protrusions on the hand contour, can better encode hand part semantics, and thus are much more informative to analyze the hand motion. Given that they are extracted reliably, hand pose estimation accuracy can be considerably improved. However, it is generally not easy to extract them robustly for unconstrained hand motion, since they are susceptible to occlusion and the heuristics to find them usually only work well for limited number of hand postures. The color markers can aid high-level feature extraction. For example, in [43] a glove with color-coded rings is used to locate the finger joints by detecting the rings of different colors. However, a markerless solution should be more favorable for natural and comfortable hand-based HCI. These problems are partially alleviated with the advent of the depth cameras, as they capture the 3D hand shape in more detail. In the following part we focus on fingertip detection and hand part segmentation due to their importance in hand movement analysis.

2.1.2.1 Fingertip Detection

The positions of the fingertips are important high-level features of the human hand, which can be utilized in various applications for human-computer interaction. For instance, the trajectories of the fingertips can be used for gesture recognition [44, 45] or manipulative purpose in multi-touch systems [46], *e.g.*, in Fig. 2.5 (a). Especially, the fingertip positions are highly correlated with the joint angles of the fingers, and are thus useful for full-DOF hand pose estimation. Therefore, they can be used directly as the inputs to inverse kinematics solvers to infer the finger articulation parameters [5, 40, 118], *e.g.*, in Fig. 2.5 (b). This can be achieved by assuming that the bone lengths are fixed and building a kinematic chain for each finger. Let the hand pose parameters be denoted as $\mathbf{\Phi} = \{\phi_g, \phi_{f,1}, ..., \phi_{f,5}\}$, where ϕ_g is the 6 DoF global hand motion parameters, $\phi_{f,i}, i = 1, ..., 5$ are the finger articulation parameters for each finger. Let the fingertip positions be $\boldsymbol{v}_i, i = 1, ..., 5$. With forward kinematics, each of the fingertip positions can be calculated using the global hand pose parameters and local finger articulation parameters as:

$$\boldsymbol{v}_i = f(\phi_g, \phi_{f,i}), \tag{2.1}$$

where f is the forward kinematics function to calculate the fingertip position based on the bone lengths and the hand pose parameters. Also, the articulated hand motion is constrained, *e.g.*, infeasible hand postures and correlations of different finger motions, and such constraints can be used along with the fingertip positions for hand pose inference. Let the hand pose constraints be denoted as $\mathbf{\Phi} \in \Omega$, where Ω is the set of feasible hand pose parameters. Thus, given the fingertip positions, the hand pose parameters can be recovered via the inverse kinematics solver, in which the fingertip positions are used as end-effectors of the kinematic chain for hand pose recovery:

$$\begin{aligned} &\mathbf{\Phi}^* = \arg\min_{\mathbf{\Phi}} \sum_{i=1}^{5} \left\| \boldsymbol{v}_i - f(\phi_g, \phi_{f,i}) \right\|^2 \\ &s.t. \mathbf{\Phi} \in \Omega. \end{aligned} \tag{2.2}$$

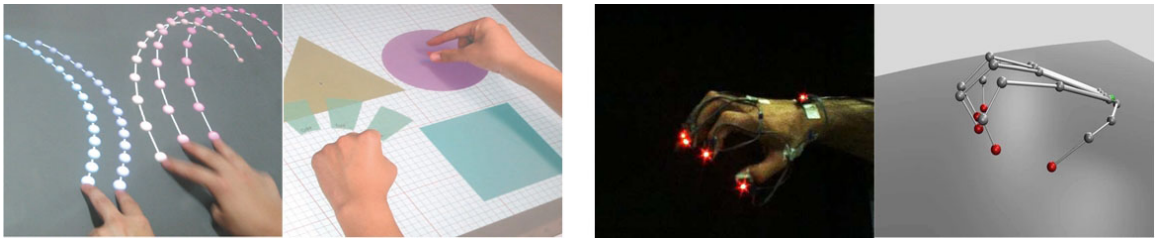


Figure 2.5: Left: Multi-touch systems based on fingertip tracking [44]. Right: Hand pose estimation with detected fingertips [5].

However, such a method that completely relies on the fingertip positions is usually underdetermined to recover all the hand pose parameters unless Ω is set to be highly constrained. Besides, it is also quite sensitive to fingertip localization results, which can be difficult to use for markerless hand pose estimation. To this end, fingertip positions are mostly used as subsidiary cues to enforce extra constraints to resolve ambiguous matching during model-fitting [3,12,48]. In these frameworks, not only the hand model is adjusted to match to the low-level image features such as edge and raw depth, but also the model fingertips are forced to move to the detected fingertips as close as possible, which helps to produce more accurate hand pose estimation.

Generally, fingertip detection relies on certain discriminative patterns to locate the fingertips directly, either with the markers attached to the fingertips or classification based on the learned visual patterns of the fingertip images. Since vision-based fingertip detection is difficult for bare hands, color or optical markers are sometimes utilized for robust extraction. In [49] color markers are put on the fingertips and palm. The palm is marked with three points, which are used to recover both the position and orientation of the palm. Each fingertip is marked with a unique color and their 3D positions are reconstructed using the ortho-perspective camera projection approximation based on the palm position and orientation. Occlusions handling is achieved by temporal analysis of the hand motion. A similar scheme is adopted in [50], in which a stereo vision system is used to get the 3D positions of marked fingertips and wrists and scale variability of human hands is also considered. In [51] the retro-reflective markers are used to track user's index finger and its tip, which allows the user to grab, rotate and release

objects naturally. In [5] the optical markers are placed on the fingertips, and their locations are detected and tracked with a ten camera PhaseSpace motion capture system. While they achieve high accuracy, the attached markers are inconvenient for the users.

A lot of visual patterns have been utilized to detect the fingertips for the bare hands. As the stretched fingers can be easily detected in the contour or foreground image of the hand, some methods rely on shape analysis of the hand contour or silhouette for fingertip detection. In [44] the fingertips are detected in infrared image sequences by template matching in the extracted foreground of the hand. The Kalman filter is adopted to track the multiple fingertips in successive frames. In [52] the 3D position of a single fingertip is detected and tracked for a pointing finger in stereoscopic image sequences, and the fingertip is taken to be the point that maximizes both the distance to the gravity center of the hand and the contour curvature on the hand contour. In [45] the fingertips are located by classification of the points on the hand contour based on the skin/non-skin region distributions in their neighborhood, and the particle filter is used to track their motions.

The contour and silhouette are not discriminative enough to detect the fingertips for complex hand configurations, *e.g.*, bending or multiple side-by-side fingers. To this end, researchers have proposed various features to encode more information for fingertip detection. In [53] a circular Hough transform feature is built from the edge map generated by multi-scale edge extraction, which is capable to tackle more complex cases such as overlapping fingers and palm. In [3] the fingertips are located by finger nail detection in high-definition image sequences due to their more distinct appearance than the skin, and the Hough forest classifier is adopted to fulfill this task. The results show the accuracy of hand pose estimation is considerably improved with the detected fingertips. In [12] the HoG descriptor is utilized in combination with the SVM classifier to find the fingertips in depth images. The fingertips are located by sliding-window detection and further refined with the assumption that the fingertips cannot lie too far away or close to the palm center.

The geodesic distance measures the shortest distance between two points along a given 3D surface. It has been utilized to detect the salient points on the body [54, 55], *e.g.*, the head and hands, which can be taken as the geodesic extrema on the geodesic distance map. It is also an important cue for fingertip detection as the fingertips are usually located at the points that locally maximize the geodesic distances from the palm center, particularly when the fingers are not fully extended. The idea has been adopted in [2, 56] to detect the fingertips in depth images. However, due to the self-occlusion of the hand, the hand surface is incomplete in the depth images. Thus, construction and evaluation of the geodesic distance map can hardly be robust. For instance, the fingertip points and the palm can be connected in the map for a closed hand, and thus the fingertips do not correspond to the geodesic extrema in such cases.

2.1.2.2 Hand Part Parsing

The purpose of hand part parsing is to refine the raw image inputs to get a semantic segmentation of the hand region, *e.g.*, the joints, the fingers and the palm. These hand parts are important features for hand pose estimation and gesture recognition, as they are more efficient and accurate for matching and classification in such tasks. Although there has been considerable work in the similar field of human body parsing [57, 58], hand parsing has its own challenges. Different from the human body, there are no clear visual boundaries between the hand parts in the color image, and the hand motion is much more flexible, *e.g.*, the hand is fully articulated and can rotate freely in 3D space. As a result, many part detectors adopted in body parsing, *e.g.*, the HOG templates [59], do not work well for the hand. In addition, in data-driven approaches, the large appearance and viewpoint variations also indicate the need of huge amount of training data, which makes it a difficult task for both training and inference for hand parsing. The following part reviews some of the recent methods for both hand and body parsing.

The contour and silhouette are the commonly used features for hand and body parsing in unmarked visual inputs, as they are relatively robust to extract in the controlled environments.

Especially, the stretching parts, *e.g.*, the fingers and the limbs, are quite distinct on the contour. Motivated by this observation, some shape analysis techniques have been applied to hand parsing. In [61] the individual fingers and palm are segmented using the convex shape decomposition algorithm, which decomposes the hand contour into separated near-convex polygons. However, this method can only handle the open hand, and the knowledge of the articulated objects is not utilized. Another popular technique for articulated object parsing is the deformable pictorial model (DPM) [62], which arranges the different parts of the object in a deformable configuration. A separate detector is trained for each part, and the correlations between the different parts are modeled as a tree structure to allow efficient inference. In the original work of [62], the part detector utilizes only the silhouette, and thus its capability to handle overlapping parts is quite limited. As the parsing accuracy of the DPM-based methods largely relies on the performance of the part detector, various improvements have been proposed. In [57] an iterative parsing scheme is proposed to parse the body parts in unconstrained color images by finding the optimal body pose. The initial pose is first determined by using only the edge feature, and the obtained pose is then used to compute the new appearance model to determine the new parse. This process iterates and the local features are gradually tuned to detect the individual parts.

The depth cameras are more powerful tools for body and hand parsing compared to the RGB cameras. The occluded parts with similar colors can be separated with their depths, and the pixels can be better described based on the depth value contrast. In [63] the hand contour is parsed into semantic segments for gesture recognition. The depth image is first classified into hand parts with the position feature, and the results are then used to label the contour points to provide extra clue for matching. In [64, 66] the feature of the pixel in the input depth image is represented by the depth difference between a set of neighboring point pairs, and the part label of the pixel is determined by classification based on its feature values via a learned randomized decision forest (RDF) classifier [67]. In [65] the authors further extend the method in [64] with a

multi-layered random forest framework. The input depth image of the hand is first classified into a predefined gesture set and then parsed into hand parts with expert classifiers specially trained for each gesture set. However, while they achieve good gesture recognition performance, their method is not suitable for hand parsing for full-DOF hand motion, given the large variation of hand shape and viewpoint change. By assuming small hand pose changes in successive frames, the iterative closest point (ICP) algorithm is used in [48] to build the temporal correspondence between the previous frame and the current frame for hand parsing. A relative position vector is proposed to encode the relative positions of a pixel with respect to the hand part edges into binary strings, which forces the parsing parts to be consistent with hand part boundaries. Since this method largely relies on the temporal reference for parsing, it is sensitive to tracking failure due to its assumption of small hand shape deformation between successive frames.

The parsing results obtained via the per-pixel classification based methods [64, 66] are still quite noisy, since the temporal and spatial constraints are not well utilized. This is important as the training dataset usually cannot cover all the possible hand motion due to its high articulation and large viewpoint changes. As a result, a large portion of the input images can be misclassified based on only per-pixel parsing schemes. To this end, the authors in [72] propose to refine per-pixel classification by applying graph cut optimization to a pixel-level graph built on the current and previous frames, which improves parsing accuracy for both human body and hands compared to [66]. However, due to the large quantities of pixels, graph cut optimization on the pixel-level graph can be quite time-consuming. In [73] a Superpixel Markov Random Field is proposed to build a superpixel-level graph to improve per-pixel classification. The superpixels segment the image so that the misclassified regions are isolated and depth discontinuity is well reserved, which largely improve the parsing results and reduce the computational cost compared to the pixel-level graph [72].

2.2 Pose Estimation

The problem of vision-based pose estimation for the human hand and the full body has been extensively studied in literature. Since the hand is highly flexible and the hand parts occlude each other, it is still difficult to restore the full-DOF hand motion from the color and depth inputs. This section reviews the relevant works of both marker-based and markerless methods for hand pose estimation, and lays stress on the latter due to their importance in the field.

2.2.1 Marker-based Methods

The bare hand is homogeneous in color and lacks the boundaries between its different parts. To this end, the color markers are often adopted to parse the hand in RGB inputs, as shown in Fig. 2.6. In [47], a set of markers with different colors are placed on the key points of the hand, such as the fingertips and wrists so that they can be easily detected and differentiated. A 3D hand model with twenty seven pose parameters is used to fit to their 2D positions to estimate the pose. In order to reduce the complexity, the hand motion constraints are analyzed to reduce the twenty seven pose parameter to twelve. The hand pose parameters are recovered using inverse kinematics solver with the positions of the wrist and fingertips under these hand motion constraints. In [103] eighteen markers with four different colors are placed on the hand joints to track the pitcher's hand motion when capturing the baseball. As the number of colors used to label the markers is limited for robust extraction, the markers are distributed to maximize the distance of the markers with the same colors. In this way the different joints can be better recognized. In [11] a color glove is used to segment each hand part. The glove consists of twenty patches and is labeled with ten fully saturated colors so that the different patches can be robustly segmented, which provides sufficient discriminative power to estimate the natural hand rotation and articulation via template matching in a large database. A Hausdorff-like distance metric is used for template matching and the database is indexed by similarity sensitive coding to

Figure 2.6: Marker-based methods. Left: the reflective markers used in [84]. Middle: the color markers with different colors used in [103]. Right: the color glove used in [11]

accelerate nearest neighbor search. The final hand pose is determined by blending the multiple nearest neighbors retrieved from the database. In [84] twenty reflective markers are placed on the hand joints, and their 3D positions are recovered by the Vicon camera system. Although the identities of the markers are not recovered during runtime, their positions provide strong constraints on the feasible hand pose and thus largely improve the estimation accuracy.

2.2.2 Markerless Methods

Model-based fitting and template-matching are two main categories of methods for markerless articulated hand pose estimation, in which the optimal pose is inferred in either a generative or discriminative manner respectively. They have been adopted in various applications, but each has their own shortcomings. On the one hand, model-based methods work well in controlled environments and produce continuous pose predictions by gradually fitting the model to the visual inputs, but they are slow to converge in the high dimensional pose space and sensitive to initialization. On the other hand, the template-matching methods are fast and robust to initialization, but require a large amount of training data and lack the power to differentiate among the retrieved ambiguous pose predictions. In particular, since the training data are usually generated by sampling the hand pose parameters and the sampling cannot be too dense due to the high dimensionality of the hand pose, template-matching methods thus can only produce discrete pose predictions. To this end, both methods can be combined to supplement each other. For instance, the template-matching methods can serve as an initialization stage for model-based

fitting, or can retrieve a set of possible pose candidates among which the optimal one can be selected by checking the compatibility between the model and the inputs.

2.2.2.1 Model-based Fitting Methods

The model-based fitting methods are usually built upon a generative deformable hand model and seek for the optimal pose by iterative adjustment of pose parameters of the model and compatibility check between model features and input images. To this end, the model should resemble the hand in terms of both the appearance and the feasible pose configurations. Therefore, selection of the hand model representation is important for a feasible model-fitting based hand pose tracking system. On the one hand, operations on the hand model should be as fast as possible to allow efficient matching to the input image observations, *e.g.*, model update, model projection calculation and model feature extraction. On the other hand, the hand model should keep sufficient details to resemble the realistic hand for arbitrary hand postures. These two requirements are contradictory to each other, *e.g.*, more hand details on the model involves more computational cost and vice versa. In some applications like animation synthesis, the real-time requirement is not a big concern as hand pose tracking can be performed offline. In such conditions, a realistic hand model with detailed skin mesh and full-articulated skeleton is usually adopted to allow accurate matching to the visual inputs [3, 36]. However, many hand-based applications in HCI need to run in real-time, which lay more emphasis on fast operation of hand model. In these cases a simplified hand model is usually adopted, which can be built with some basic volumetric elements such as quadratic surfaces [28, 40]. Recently the sum of Gaussians model has also been popular to represent the hand and body for pose estimation, in which a set of Gaussians are attached to the kinematic skeleton to approximate the hand and body surfaces [12, 116, 117]. Compared to the geometrical models, they have the intriguing property that the gradient of the objective function can be computed analytically, which enables efficient optimization. Fig. 2.7 illustrates the various representations of hand models.

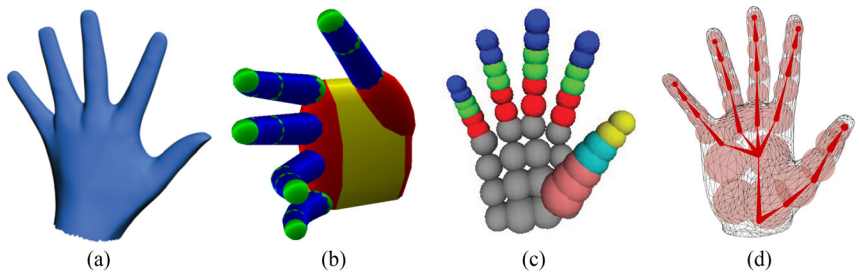


Figure 2.7: (a) Detailed mesh surface model [3]. (b) Quadratic surface model [28]. (c) Sphere approximated model [40]. (d) Sum-of-Gaussians model [12].

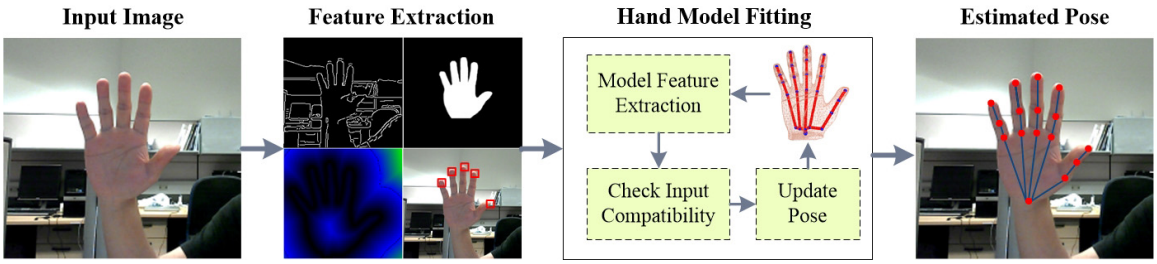


Figure 2.8: Model-based fitting based framework for hand pose estimation.

With a proper hand model, one can build the correspondences between the model and the visual inputs to estimate the matching error. Based on the error metric, the model-fitting task can usually be formulated as an objective function minimization problem. The procedure begins with some selected or randomly generated estimates of the model parameters. In the following iterations, it updates the pose estimation until certain termination criteria is met. Fig. 2.8 illustrates the general framework of model-fitting based hand pose estimation.

In [93], the hand pose parameters are decoupled into the global motion and local finger articulations and estimated separately. The global motion is estimated by assuming the finger poses are fixed. The local finger motion is estimated by inverse kinematics using the fingertips as the end-effectors. The method is not robust as extraction of fingertips is difficult and sensitive to self-occlusion. In [94] the quadric surfaces are used to model the hand to generate the model contours efficiently, which are used to match to the image contour. A frame rate of 3 Hz is reported on a seven DOF hand motion sequence. In [13] the feasible hand configuration space

is discretized and indexed with a KD-tree. The Nelder-Mead simplex algorithm is adopted to search for the hypothesized pose that best matches the input in terms of edge and silhouette similarities. However, no quantitative results are reported. In [62] the articulated body is modeled as the deformable pictorial structure, in which the pair-wise correlations between the body parts are approximated as spring-like connections. The body pose is estimated by jointly minimizing the matching error between the model and inputs and the pair-wise energy between the body parts, which is efficiently solved by the generalized distance-transform technique.

In [33] multiple hand silhouettes are extracted from the images captured with several cameras around the hand, where the background is set using blue boards for easy hand segmentation. A voxel model is generated with the multi-view data and matched to a 3D hand model, and the optimal pose is sought to make the hand model surface stay inside the voxels. In [36] the texture and shading of the skin are captured from input images and synthesized in the hand model, and the illumination sources are controlled in real scenario and simulated during hand modeling. A variational formulation is proposed to estimate the full DOF hand pose. In this way hand pose is recovered quite accurately since matching ambiguity is largely reduced. However, this method is difficult to use in real HCI scenarios. In [28] a Kinect depth camera is adopted to capture the hand image as it can better handle the background clutter and pose ambiguity in monocular color image, the particle swarm optimization algorithm is used to find the optimal pose that best fits the image projection of a 3D hand model to the input depth image and skin silhouette. With the point clouds generated by the depth camera, the iterative closest points algorithm and its extensions to articulated objects are also commonly used for hand pose estimation [91, 92], which iteratively build point-to-point correspondences between model and input point cloud and seek for the skeleton transform to minimize the distance between the point pairs.

2.2.2.2 Template-Matching Methods

Although model-based methods work well in controlled environments, they are slow to converge in the high dimensional pose space and sensitive to initialization. Another drawback is that they require estimating all the model pose parameters to find the optimal match to the image observation. In contrast, the template-matching methods are able to infer an arbitrary subset of the pose parameters by directly mapping of the image features to pre-indexed templates, *e.g.*, estimating only palm motion despite the articulated finger motion. Another advantage of them is that they do not need to compute the projection of the hand model for matching for a lot of iterations, which is quite time-consuming. Besides, hand motion constraints are automatically included in the pose parameters associated with the templates in the dataset. Generally, these methods need to build a large dataset to cover the possible hand postures, and each template in the dataset contains certain features for matching and the associated pose parameters. The dataset are usually indexed for fast search. During testing, the input hand pose is recovered by looking for the templates that share the similar features. Such methods have been brought up quite long ago, but have gained high popularity in recent years with the advent of low-cost depth cameras, which have demonstrated promising results. Therefore, we summarize the related work with color and depth camera inputs separately in the following parts. Fig. 2.9 illustrates the general framework of template-matching based hand pose estimation.

Color cameras: These methods map the color features to the pose parameter space. In [104] the set of possible body poses are defined by a few clusters obtained from the training data, and a function is learned to map the low-level descriptors, *e.g.*, the image moment, to each of the clusters. The pose of the input is inferred by fusing the multiple candidate poses based on the mapping confidence. This method can handle a very limited number of hand poses. In [97] the hand edge image is encoded into a score value vector by matching to a pre-defined set of shape templates, and a multivariate relevance vector machine uses it as the input to retrieve some pose

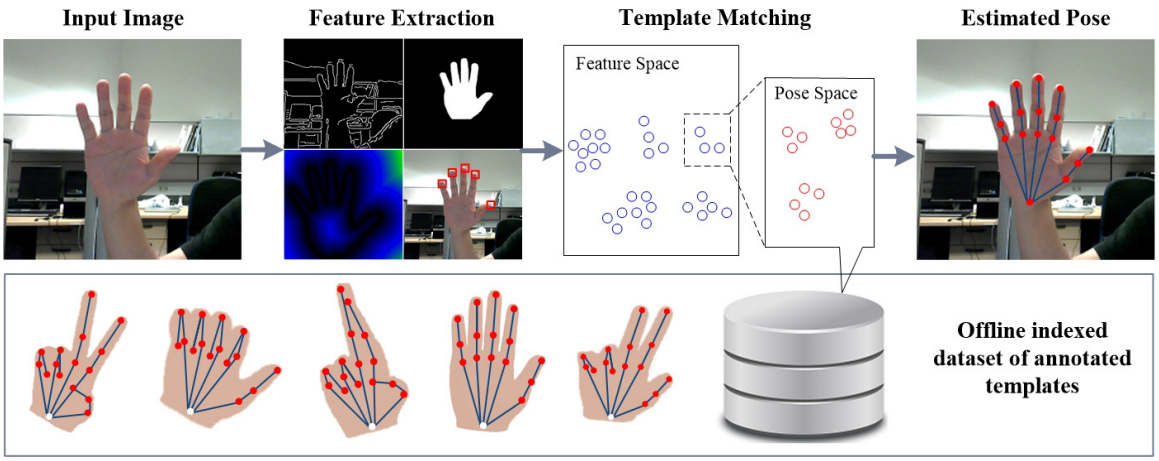


Figure 2.9: Template-matching based framework for hand pose estimation.

hypotheses. The optimal pose is obtained by a verification stage with the hand model projection. In [15] an isometric self-organizing map is used to learn a nonlinear mapping between image features and pose, which reduces the dataset redundancy by grouping templates with similar features and poses together. The hand edges are captured at only depth discontinuities with a multi-flash camera and encoded into shape context for matching. In [99] the simple hand grasping motion is captured with a single color camera. The locality-sensitive Hashing (LSH) is utilized to retrieve multiple candidates from the database based on the HoG feature of the input image. The hand pose is estimated by applying the temporal constraints on the retrieved candidates to resolve ambiguity. In [98] a two color camera system is presented to capture 6 DOF palm motion and simple gestures like pinching or pointing for both hands. A pair of hand silhouettes are extracted and coded into binary strings for fast query in the database to retrieve the hand pose. In [7], the whole parameter set of the hand pose is decomposed into many overlapping subsets. LSH-based nearest neighbor search is used to get the partial estimation for each subset, and the results are further integrated by a simulated annealing EM algorithm to estimate the global pose.

Since monocular color images lack discriminative power for hand pose recovery, template-matching based methods usually retrieve a set of ambiguous pose candidates. Sometimes, this

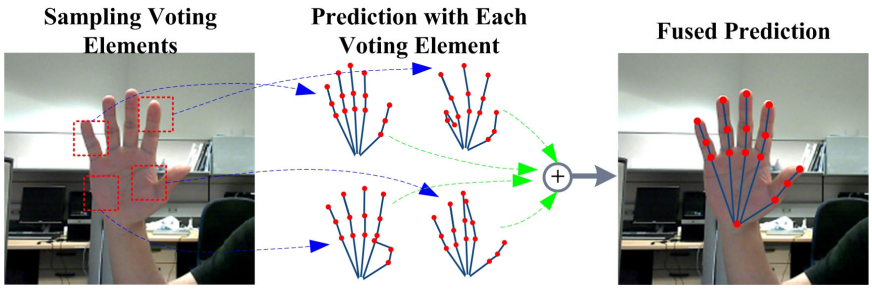


Figure 2.10: Spatial-voting for hand pose estimation.

issue is solved using multi-camera setting [16, 98] or temporal constraints [99]. Another common approach is to add a verification stage after the discriminative pose retrieval stage [97, 105]. For instance, given the set of ambiguous poses retrieved and their associated features, the optimal one can be determined by choosing the one whose feature is most compatible with the input hand feature.

Depth cameras: Compared to the color cameras, the depth cameras can largely resolve the prediction ambiguity as they can capture more detailed 3D structure of the hand surface. Among the discriminative methods, the random forest and its variants have proven very effective to capture the pose of articulated objects in depth images, such as the human body [41, 107] or hand [8, 9, 100]. Overall, the success of these methods can be attributed to two aspects. First, the random forest has intriguing characteristics to fulfill the task, which can be implemented to run very fast for both training and testing due to its suitability for parallel processing and is robust to labeling noise. Especially, it has reported good performances for high-dimensional inputs [108], which is suitable for the articulated hand pose. Based on simple depth features, the random forest has shown to be fast and robust for pose estimation. Second, these methods utilize the spatial voting technique to aggregate the pose predictions from spatially-distributed voting elements, so that the fused prediction can be less sensitive to imperfect inputs, *e.g.*, inaccurate hand segmentation. Fig. 2.10 illustrates the concept of spatial-voting for pose estimation.

At first the combination of random forest and spatial-voting for articulated pose estimation

shows its effectiveness for human body. In [66], the RDF classifier is trained for per-pixel classification of the depth images into semantic body parts and the joint locations are obtained by mean-shift mode seeking based on the parsed body parts. In [41], the authors propose to directly regress for the body joints from the depth images. Based on a learned random regression forest model, each pixel casts its votes for the joint positions individually, and the votes from all the pixels are fused by mean-shift mode-finding to give the final predictions. This method is fast and can infer the locations for even occluded joints. However, the distribution of the individual votes tends to be multi-modal for imperfect inputs, and the mean-shift search can be easily trapped to the wrong mode. In [107] the authors further improve the performance of the random regression forest by modeling certain body-specific parameters, *e.g.*, orientation or height, as a hidden variable for inference, which are inferred jointly with the body pose parameters. The random forest soon finds its applications in hand pose estimation. In [8], the authors propose to use the random forest to directly regress for the hand joint angles from depth images. With a pre-trained forest, each pixel casts its votes for the joint angles individually, and the votes from all the pixels are fused to a set of candidates. The optimal one is determined by a verification stage with a hand model. A similar regression forest base method is proposed in [9], with the new characteristic that transfer learning is utilized to handle the discrepancy between synthesized and real-world data. In [85] the authors propose to utilize the regression forest in [41] to predict the hand pose. To resolve the ambiguous predictions, their method first finds a set of candidate locations for each joint through mode-seeking, and then applies the bone length constraints to obtain the optimal combination of the different joint locations via Dynamic Programming. However, the bone lengths alone are still insufficient to describe the feasible hand pose space, *e.g.*, the motion constraints between multiple fingers. Besides, this method can only work for a fixed hand size, and does not generalize well to different users.

2.2.2.3 Hybrid Methods

The model-based fitting and template-matching methods both have their pros and cons. The model-based fitting methods are slow to converge in the high dimensional pose space and sensitive to initialization. The template-matching methods need large quantities of training data to cover the feasible hand motion parameter space, and suffer from more ambiguous pose predictions. Especially, it can only give discrete pose predictions due to discretization of the pose space for sampling. Therefore, they can be combined to supplement each other so that their advantages can both be exploited.

As discussed previously, model-based fitting can serve as a verification stage after the discriminative pose retrieval stage [8, 97]. On the contrary, pose retrieval can also serve as an initialization stage for model-based fitting. In [109] a human body pose tracking framework based on 3D model fitting is proposed. While the input body size can vary a lot, the random forest classifier provides rough body parsing for fitting the size of the 3D model to the real inputs as well as for initialization and recovering from tracking failure. Both methods can also be used for pose estimation independently and their predictions are finally fused up to certain criteria. In [55] the geodesic extrema are extracted from the depth images, which are used to retrieve the candidate body pose by searching in the database of geodesic extrema templates. Another candidate pose is obtained by fitting a mesh body model to the depth image, and the final prediction is taken to fit to both estimations.

Noting that the semantic hand parts, *e.g.*, fingertips or knuckles, can largely prune infeasible hand configurations, which helps to reduce the search space of the hand pose parameters for hand pose estimation, many methods utilize pre-trained classifiers to locate them to facilitate full-DOF pose recovery. In [3] a model-based framework is presented to capture the very subtle hand motion with multi-view inputs of eight HD cameras. The fingernails are detected in each view by Hough forest classifiers and used with image edges and optical flow to fit to an elaborate

hand model. In [12] the fingertips are detected in depth images by SVM classification with HoG descriptor, which are used for discriminative pose inference with offline synthesized database. Meanwhile, another pose candidate is obtained via generative model fitting. The optimal pose is chosen from the two candidates by selecting the one minimizing the matching error. A similar framework is presented in [40], in which the protrusive fingertips are detected by morphological analysis in the depth image. The partial hand pose is recovered from the possible incomplete 3D fingertip positions and used for initialization for the subsequent model-fitting stage, which can help to speed up convergence as well as to avoid local optima.

Chapter 3

Hand Part Parsing

This chapter presents a robust hand parsing scheme to extract the high-level hand parts from the depth images [73], which aims to improve the noisy per-pixel classification results [66] by utilizing both temporal and spatial constraints. To this end, we propose a Superpixel-Markov Random Field (SMRF) parsing scheme to enforce the spatial smoothness and the label co-occurrence prior to remove the misclassified regions. Compared to pixel-level filtering, the SMRF scheme is more suitable to model the misclassified regions. The temporal references are used in combination with the per-pixel classifier as an ensemble of classifiers, which further improves the parsing accuracy. In addition, we develop a depth-context feature and further improve its performance by a distance-adaptive sampling scheme, which proves more effective for hand parsing compared to the binary depth comparison feature in [66]. Overall, the proposed method is accurate and runs at linear computational complexity. The tests on synthesized dataset show it gives much higher accuracy for single-frame parsing and enhanced robustness for continuous sequence parsing compared to benchmarks. The tests on real-world depth images of the hand and human body show the robustness to complex hand configurations of our method and its generalization power to different kinds of articulated objects.

The pipeline of the proposed hand parsing algorithm is illustrated in Fig. 3.1. The input of the algorithm is a sequence of depth images, where we assume that only one hand is visible and

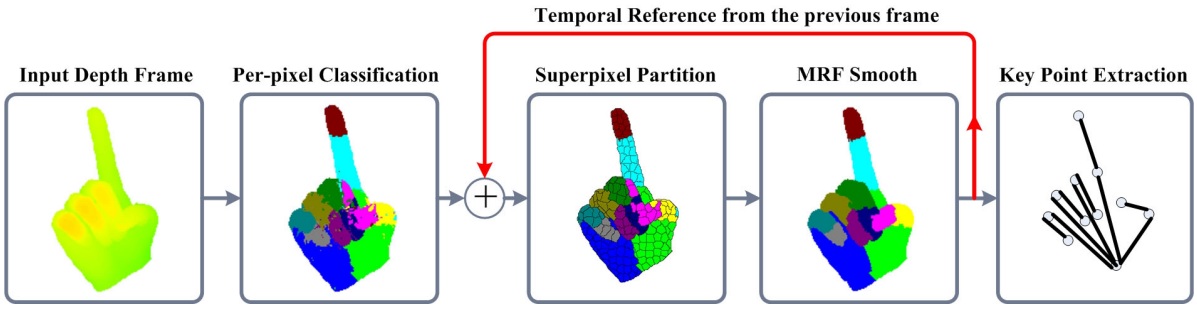


Figure 3.1: The pipeline of the proposed hand parsing scheme.

the hand is the nearest object to the camera. The output is the part label images and key point sets corresponding to the hand parts. The functions of the modules are listed as follows:

Per-pixel classification: to assign an initial part label to each pixel by classification with a trained random decision forest classifier and the temporal position classifier. A distance adaptive feature selection scheme is combined with a depth-context feature to describe each pixel for classification.

Superpixel partition: to partition the initial hand part segmentation into a set of superpixels to reduce the computational complexity involved in MRF smoothing.

MRF inference: to reassign the hand part labels via superpixel-level inference. The depth discontinuity, superpixel boundary shapes and hand label co-occurrence are all handled to construct the MRF graph for efficient inference.

Key point extraction: to infer the center of each labeled hand part based on their distributions in 3D space.

The main idea of the depth-context feature is to describe each pixel of the depth image using a sampling grid centered at the current pixel. In our distance-adaptive scheme, the grid is sampled with non-fixed steps, *e.g.*, more densely sampled near to the center and vice versa. The RDF is used for per-pixel classification based on the depth-context feature. The temporal constraints are enforced by learning the 3D position distribution of the hand parts from the previous

frame, and combining it with the RDF classifier to form an ensemble of classifiers. As to the spatial constraints, the neighborhood smoothness of the pixel labels is enforced in the SMRF framework to suppress the per-pixel classification error. By using the superpixel, the misclassified isolated regions can be represented as an atomic element and the computational complexity of Markov Random Field (MRF) based smoothing can be largely reduced. Finally, the parsed hand parts are further processed to get a set of key points as a high-level representation of the hand.

3.1 Hand Modeling

A 3D hand model is needed to generate the training samples for hand part classification as well as to measure the compatibility between the image features and a hypothesized hand pose ϕ. We build a fully deformable model with 3D closed mesh to simulate the real hand, as shown in Fig. 3.2(a). The model consists of a skeleton system and a skin surface mesh. The skeleton has 27 degrees of freedom, including 6 DOFs of global motion and 21 DOFs of local motion [74]. It is modeled as a kinematic chain of 20 joints connected by bones in a tree structure with the root at the wrist, as in Fig. 3.2(b). A set of static and dynamic constraints [74, 75] are adopted to limit the parameter space of the hand motion. The skeleton thus has an equivalent of 18 degrees of freedom. The skin mesh consists of about 5000 vertices, which form approximately 7000 triangles. Each vertex and triangle is assigned a label $l \in L$ to indicate which hand part they belong to. Give a hypothesized hand pose ϕ, the joint positions of the skeleton are first computed using forward kinematics, and the positions of the vertices on the skin mesh are then updated using the skeleton subspace deformation method [76].

In order to generate the depth image and label image to train the hand part classifier, we adopt the pin-hole camera model [77] to project the hand model onto the image plane. This is done by calculating the projection of all the triangles of the skin mesh, as shown in Fig. 3.3(a).

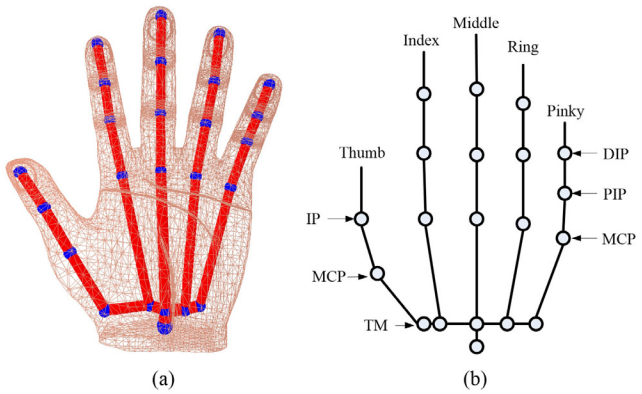


Figure 3.2: The kinematic chain (Left) and the 3D hand model (Right).

The pixel coordinate $\mathbf{p}_i$ of a 3D point $\mathbf{v}_i = [a_i, b_i, c_i]^T$ is projected by:

$$\mathbf{p}_i = \Psi_P(\mathbf{v}_i) = \frac{F}{c_i} \times \begin{bmatrix} D_x a_i \\ D_y b_i \end{bmatrix} + \begin{bmatrix} a_o \\ b_o \end{bmatrix}, \tag{3.1}$$

where F is the focal length; D_x and D_y are the coefficients to define the metric units to pixels; a_o and b_o are the principal point of the image plane. These parameters are intrinsic parameters of the camera and can be obtained by camera calibration techniques.

Following this projection model, the three vertices of each triangle are projected onto the image plane to get $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$. The pixels with the projected triangle in the label image are assigned the corresponding label of the triangle $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$. Besides, we use a linear interpolation method to fill the pixels within $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ in the depth image, as shown in Fig. 3.3(b). The 3D point $\mathbf{v}$ corresponding to $\mathbf{p}$ is a linear combination of the three vertices:

$$\mathbf{v} = \frac{(d_1 d_3 \mathbf{v}_2 + d_2 d_3 \mathbf{v}_1)}{(d_1 + d_2)(d_3 + d_4)} + \frac{d_4 \mathbf{v}_3}{d_3 + d_4}, \tag{3.2}$$

where (d_1, d_2, d_3) are the distances from $\mathbf{p}$ to the vertices, and d_4 is the length from $\mathbf{p}$ to $(\mathbf{p}_1, \mathbf{p}_2)$ along the extension of d_3. Besides, since the skin mesh is closed, multiple triangles on the mesh can be projected to the same pixel on the depth and label images. In this case, the pixel will take values from the triangle that gives the minimum depth value.

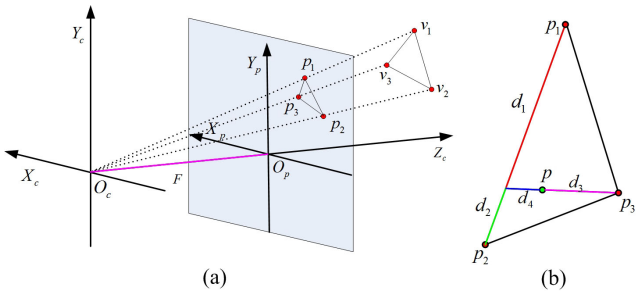


Figure 3.3: The pin-hole camera model (Left) and the depth image interpolation scheme (Right).

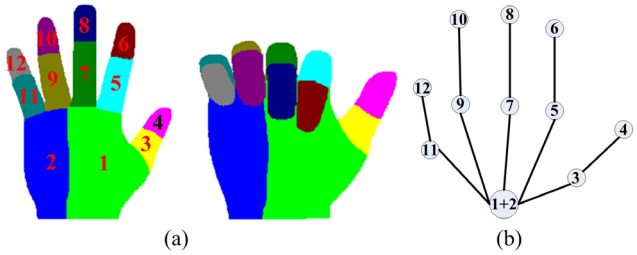


Figure 3.4: The hand partition scheme. (a) The label distributions for different hand parts. (b) The tree-structured hierarchy of the positions of the labeled hand parts.

3.2 Per-pixel Classification

The task of hand part classification is to assign a label $l \in L$ to each pixel in the depth image of the hand region. Fig. 3.4 (a) shows our hand label partition scheme, where the whole hand is classified into twelve non-overlapping parts. These hand parts are not independent from each other. We further assume that their positions follow a tree-structured hierarchy to assist hand gesture recognition in Section 3.5, as shown in Fig. 3.4 (b). To perform per-pixel classification, we propose a depth-context feature with a novel distance-adaptive sampling scheme to describe the 3D context for each pixel, based on which a trained RDF classifier is used to assign the part labels. In case that the temporal reference is available, the position distribution of each labeled part will also be learned and used in combination with the RDF classifier to form an ensemble of classifiers for per-pixel classification.

3.2.1 Depth-Context Feature

We propose a depth-context feature for classification of each pixel in the depth image and improve its performance with a distance-adaptive sampling scheme. The feature describes the 3D context of each pixel using the relative depths between the pixel and its neighboring pixels as shown in Fig. 3.5 (a). The red circle denotes the current point for classification, and the green circles denote the context points to calculate the feature values. The 3D relative position between each context point and the current pixel is defined as a 3D offset $\mathbf{v}_d = [a_d, b_d, 0]^T$. Given the pixel $\mathbf{p}$ and its corresponding 3D point $\mathbf{v}$, the pixel coordinate $\mathbf{p}_c$ of the context point can thus be obtained by projecting the 3D position of the context point to the image plane $\mathbf{p}_c = \Psi_P(\mathbf{v} + \mathbf{v}_d)$. This projection procedure ensures the relative pixel coordinate of the context point is adaptively adjusted by the depth value of the current pixel and their 3D relative positions are thus conserved. This is similar to the normalization of the feature offsets in [66] and the depth context feature is thus depth-invariant. The feature value is obtained by the depth difference between the current pixel and the projected pixel of the context point:

$$f_F(\mathbf{p}, \mathbf{v}_d) = d_z(\mathbf{p}) - d_z[\Psi_P(\mathbf{v} + \mathbf{v}_d)], \tag{3.3}$$

where d_z is the depth value at the given pixel in the depth image. In addition, the feature value f_F is restricted to the range $[-\varepsilon_d, \varepsilon_d]$, where ε_d is a constant value. This is because the background pixels in the synthesized training images are usually assigned a large constant depth value, which is generally different from the background depth values in the test images. Thus, given the same hand configuration, the feature values on the context points that lie on the background can be very different on the training and testing depth images. Such inconsistency can be eliminated by threshold with the constant ε_d. Meanwhile, for a point on the hand, ε_d should be big enough to capture the depth difference between the point and any other context point lying on the hand. In our implementation we set $\varepsilon_d = 0.3m$ to parse the hand.

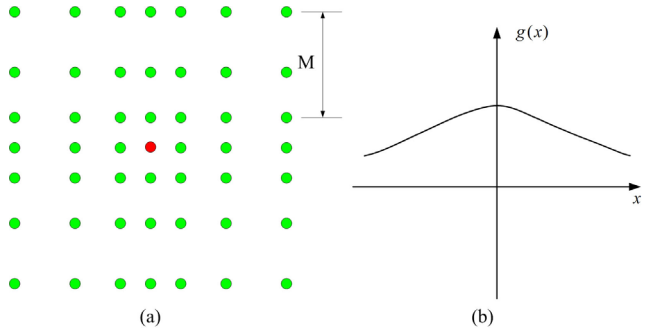


Figure 3.5: The distance-adaptive scheme to select depth feature indices, in which the red circle indicates the current pixel, and the green circles indicate the candidate offset features (Left). The sampling density function along each axis (Right).

Another issue is the selection of the context point positions. In previous work [64, 66], the context points are randomly generated within certain ranges. However, the nearer points should be more important to describe the context than the faraway points. Especially, most of the faraway context points of the finger parts lay either on the background or the palm part, both of which are quite homogeneous regions. To this end, we propose a distance-adaptive sampling scheme to generate the context points, as shown in Fig. 3.5(b). The points nearer to the current pixel are more densely sampled, and vice versa. For simplicity, we focus the discussion on one dimension as the sampling scheme is symmetric. The sampling density function $g(x)$ is adopted to determine the location of each context point. Let the feasible range of the context points be $[-h, h]$. $g(x)$ is a non-increasing function, and satisfies $\int_0^h g(x)dx = M$. M is a parameter to determine the grid size. In our implementation we choose a linear function for $g(x)$, in which we define $g(h) = g(0) \times 0.2$ to ensure that the context points are more densely sampled around the central pixel. The coordinate of the context point can be obtained by solving the following equation for x_d, $i = 1, ..., M$:

$$\int_0^{x_d} g(x)dx = i. \tag{3.4}$$

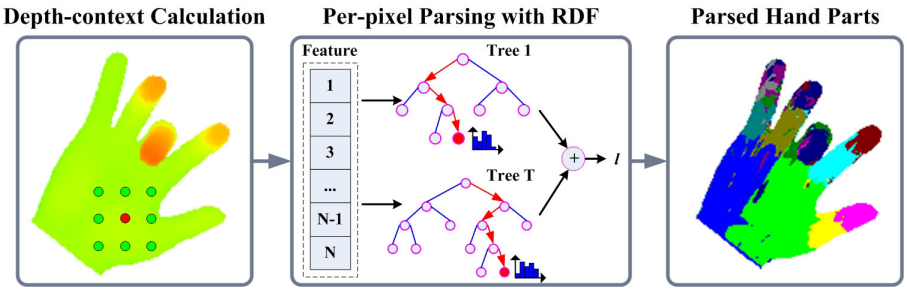


Figure 3.6: Per-pixel hand parsing with RDF.

3.2.2 RDF Classification

Based on the depth-context feature, we adopt the Random Decision Forest classifier [67] to label the hand parts by per-pixel classification. The training procedure of the random decision forest is essentially the same as that in [66]. To generate the training samples, we randomly select ten pixels for each hand part and the background from each training image, and each sample consists of the label of the pixel and the corresponding depth-context feature value.

During the test stage, an input pixel $(\mathbf{p}, \mathbf{v})$ is first processed by each tree in the random decision forest, as illustrated in Fig. 3.6. For each tree, the posterior probability $P_i(l|\mathbf{v})$ is obtained by starting at the root and recursively assigned to the left or the right child based on the tree node test result until it finally reaches a leaf node. The final posterior probability $P(l|\mathbf{v})$ is obtained by fusing the results of all the trees in the random forest:

$$P(l|\mathbf{v}) = \frac{1}{N}\sum_{i}^{N} P_i(l|\mathbf{v}), \tag{3.5}$$

where N is the number of trees in the random decision forest. The label of the pixel can be directly determined by MAP estimation:$l^* = \arg\max_l P(l|\mathbf{v})$. However, the final label decision is not made here, and the posterior $P(l|\mathbf{v})$ will be further processed by the Superpixel-MRF framework to give the refined results.

3.2.3 Key Point Extraction

The key points are compact representations of the hand parsing results. Instead of calculating the 3D centroid of each labeled hand part, we calculate the expected positions of each hand part based on the label distribution given by hand parsing. Let the 3D point set on the input hand be H. Let the label distribution given by the RDF classifier be $P(l|\mathbf{v})$, $\mathbf{v} \in H$. The 3D position distribution of a hand part l can be calculated by:

$$P(\mathbf{v}|l) = \frac{P(l|\mathbf{v})P(\mathbf{v})}{\sum_{\mathbf{v}_i \in H} P(l|\mathbf{v}_i)P(\mathbf{v}_i)} = \frac{P(l|\mathbf{v})}{\sum_{\mathbf{v}_i \in H} P(l|\mathbf{v}_i)}, \tag{3.6}$$

where we assume uniform prior of the 3D positions of the depth points. Thus, the expected position of the hand part l is given by:

$$\mathbf{v}_c^l = \sum\nolimits_{\mathbf{v}_i \in H} \mathbf{v}_i P(\mathbf{v}_i|l). \tag{3.7}$$

Note for the two hand parts 1 and 2 that belong to the palm in Fig. 3.4(a), the 3D points within them and the corresponding label distribution are merged to get the palm position. Let the resulting positions of each hand part be $U_I = \{\mathbf{v}_c^l | l = 1, ..., L\}$. These points are further arranged into a tree hierarchy based on their relationship in the hand skeleton, as illustrated in Fig. 3.4(b), where the root corresponds to the palm position.

3.2.4 Temporal Constraints

The temporal references are useful when parsing the hands in successive images. Given that the hand parsing result in the previous frame is available, we propose to use it as an auxiliary classifier for the RDF classifier for per-pixel classification in the current frame. This classifier forms an ensemble of classifiers with the RDF classifier. To this end, we use the 3D Gaussian distribution to approximate the point distribution within each hand part in the previous frame, *i.e.* $P_T(\mathbf{v}|l) \sim \mathcal{N}(\mu_l, C_l)$, and the parameters μ_l and C_l are learned from the points that belong

to each hand part in the previous frame. The temporal classifier is thus given by:

$$P_T(l|\mathbf{v}) \propto P_T(\mathbf{v}|l)P_T(l), \tag{3.8}$$

where $P_T(l)$ is proportional to the number of points within each part. For clarity we denote the RDF classifier as $P_R(l|\mathbf{v})$. The previous per-pixel classification scheme can thus be accomplished by the ensemble of P_T and P_R to incorporate the temporal reference, that is:

$$P(l|\mathbf{v}) = \eta P_R(l|\mathbf{v}) + (1-\eta)P_T(l|\mathbf{v}). \tag{3.9}$$

Here η is the coefficient to control the relative significance between the temporal classifier and the RDF classifier. As the variation of hand motion speed is large and the performance of P_T can degrade a lot when the hand moves very fast, we make the significance of P_R outweigh the temporal term P_T so that their combination is robust to tracking failure, *i.e.* $\eta > 0.5$. In practice we find with a value of 0.5 the method seldom fails when parsing successive depth image sequences.

3.3 Superpixel-MRF Hand Parsing

The hand parsing results produced by per-pixel classification are quite noisy since dependencies between neighboring pixels are not fully utilized. Simply applying the pixel-level filtering techniques, *e.g.*, the median filter, to the labeling results will not work well since many misclassified pixels form small isolated regions surrounded by other parts, as shown in Fig. 3.7 (a). The Markov Random Fields [78] can be used to refine the classification results, which can well model the constraints from the neighboring states and the image observations [79]. However, the traditional pixel-based MRF is time consuming for real-time HCI applications. Besides, the isolated misclassified regions are more suitable to be represented as a whole rather than a set of pixels for MRF inference. Therefore, we partition the hand region into superpixels, and combine them with the MRF inference to refine the parsed hand parts based on the per-pixel

parsing results.

The proposed superpixel-MRF framework is built based on both the posterior probability $P(l|\mathbf{v})$ given by per-pixel classification and the depth image. The superpixels are constructed with two criteria. First, the depth discontinuity must be conserved when determining the borders of neighboring superpixels. Second, pixels within one superpixel should have similar posterior probability. These principles are incorporated into the simple linear iterative clustering (SLIC) superpixel partition algorithm [80], and the resulting superpixel partition is compact in terms of 3D space distribution and posterior probability. The MRF network is then constructed for the superpixels with the similar way in [81].

Given the compact representation of the superpixels, they still behave differently from the pixels. Some superpixels have quite irregular shapes in order to conserve the depth discontinuity during superpixel partition. Also, the relative sizes of neighboring superpixels are sometimes uneven, as shown in Fig. 3.7(b). This suggests that some superpixels can have larger influence on their neighbors than others. Therefore, in the MRF framework, we model the interaction energies between superpixels based on their common borders, relative sizes and label co-occurrence distribution when determining the pairwise term. That is, if two neighboring superpixels have many common borders, they are quite likely to have the same label. Also, a small superpixel can be more easily smoothed by a big neighbor than the opposite. We also take the idea of label co-occurrence from [82] so that the unsmoothness between the neighbors with low co-occurrence rate is high.

3.3.1 Superpixel Partition

Though the per-pixel classification result can be noisy, the labeled parts are mostly locally homogeneous. Besides, a large portion of the wrongly classified pixels form isolated small regions. Therefore, the labeled image is suitable to be represented by a set of super pixels for further processing, which can well model the misclassified regions as well as reduce the

computational cost involved in the MRF energy minimization process. To this end, we modify the SLIC algorithm to get our superpixel partition in depth images. The original SLIC method is developed for color images, while we need to get the superpixel partition that conforms to both the hand part classification results and the depth discontinuity. To be specific, the criteria for our superpixel partition scheme are:

1. Depth continuity: the depth difference between neighboring pixels within the superpixel is smaller than a threshold d_T.

2. Similar posterior probability: the pixels within a superpixel should have similar $P(l|\mathbf{v})$.

According to these requirements, it is not reasonable to apply the SLIC method directly to the posterior distribution $P(l|\mathbf{v})$, as the Euclidean distance in $P(l|\mathbf{v})$ does not make much sense. Thus, we perform superpixel partition in the space of posterior distributions and adopt the Kullback-Leibler divergence to measure the difference between each pixel and the superpixel cluster center. Let the set of superpixel partition be $S = s_1 \cup s_2 \cup ... \cup s_K$. The posterior probability and depth of each superpixel are taken as the average of all the pixels within the partition, that is:

$$d_{sk} = \frac{1}{|s_k|} \sum_{\mathbf{v}_j \in s_k} c_j \tag{3.10}$$

$$P(l|s_k) = \frac{1}{|s_k|} \sum_{\mathbf{v}_j \in s_k} P(l|\mathbf{v}_j), \tag{3.11}$$

where $d_(S_k)$ and $P(l|s_k)$ are the depth and posterior probability of the superpixel s_k. To perform superpixel clustering, we first define the distance metric in the posterior probability to measure the difference between the pixels and the superpixel cluster to be:

$$D_{kl}(s_k, \mathbf{p}_j) = \sum_{l=1}^{L} P(l|s_k) \log \frac{P(l|s_k)}{P(l|\mathbf{p}_j)}. \tag{3.12}$$

Besides, in order to preserve the depth discontinuity in the superpixel partition, the pixel $\mathbf{p}_j$ can be assigned to s_k only if $|c_j - d_{s_k}| \leq d_T$. In the implementation we set $d_T = 6mm$. Also,

the superpixel should be compact in the 2D image coordinate space, as in the original SLIC method. Therefore, we define the distance metric for superpixel partition as:

$$D = \begin{cases} \sqrt{D_{kl} + \left(\frac{D_S}{M_S}\right)^2 \gamma^2} & if |c_j - d_{s_k}| \leq d_T \\ \infty & otherwise \end{cases} \tag{3.13}$$

where M_S is the regular grid step on the image plane to determine the superpixel size; D_S is the pixel distance between $\mathbf{p}_j$ and the superpixel center; γ controls the relative importance of the two terms. Based on the distance metric D, the superpixel partition is performed by the clustering scheme in the SLIC method. The posterior probability of these superpixels forms the observation data for the following MRF inference stage. Note that the sizes of the superpixels are mostly quite close, except that the wrongly labeled regions are generally small or have irregular shapes. This indicates such misclassified parts are more likely to be neutralized by their neighboring superpixels.

3.3.2 Superpixel-MRF Inference

Given the set of superpixels $S = \{s_k\}$, the goal for MRF inference is to assign each superpixel a new label $l \in L$. Let the associated labels for the superpixels be $Y = \{y_k\}$. The task for MRF inference of the labels is to get the MAP solution of $Y^* = \arg\max_Y P(Y|S)$, which is equivalent to the minimization of the following energy function:

$$E = E_d + E_S = \sum_{i \in S} \phi_i(y_i) + \sum_{i \in S, j \in N_i} \psi_{i,j}(y_i, y_j). \tag{3.14}$$

Here E_d is the unary term to measure the discrepancy between the inferred label and the per-pixel classification results, and we set $\phi_i(y_i) = -\log P(y_i|s_i)$.

The pairwise term E_s is used to measure the smoothness between neighboring superpixels, and we utilize the idea of label co-occurrence [82] to define $\psi_{i,j}$. The label co-occurrence represents the conditional probability $P(y_i|y_j)$, which indicates how likely a superpixel with

state y_j will have a neighbor with a state y_i. Since some hand parts are more likely to be adjacent than others, *e.g.*, the chances that the part 1 and 2 are adjacent are higher than the part 1 and 6, the label co-occurrence can be useful during inference by punishing the unlikely adjacent states. However, unlike [82] in which a superpixel in the color images is equally affected by all its neighbors, we take the depth discontinuity and the irregular shapes of the superpixels into consideration to model the pairwise interaction energy. First, the depth discontinuity between the superpixels is used to define the adjacency of a pair of nodes. The nodes (y_i, y_j) are adjacent only if the following two criteria are met:

$$\Omega_i \cap \Omega_j \neq \emptyset, \tag{3.15}$$

$$|d_{s_i} - d_{s_j}| \leq d_T \tag{3.16}$$

where Ω is the set of border pixels of the superpixel. With these criteria, there is a pairwise energy term between two nodes only if they are neighboring superpixels and have similar depths. In addition, some superpixels can have quite irregular shapes, which make them interact with their neighbors in an uneven way. Specifically, a superpixel is more likely to take the same label with the neighbors that share more borders than others, and small superpixels are more likely to take the same label with its big neighbors than the opposite way. This is especially significant for the wrongly labeled regions, which are usually isolated and small. For two adjacent nodes i and j, the uneven influences resulting from these factors should be reflected in the pairwise energy term, in addition to their state differences (y_i, y_j). Thus, we define a weight coefficient $\alpha_{i,j}$ for the pairwise term of adjacent nodes, which is given by:

$$\alpha_{i,j} = \frac{|\Omega_i \cap \Omega_j|}{|\Omega_i|} \times \frac{|s_j|}{|s_i|}. \tag{3.17}$$

A big value of $\alpha_{i,j}$ indicates the superpixel i is more likely to be affected by its neighbor j.

Fig. 3.7 shows an example to illustrate the effectiveness of modeling the pairwise energy based on the superpixel partition results. Fig. 3.7 (b) shows the superpixel partition of the

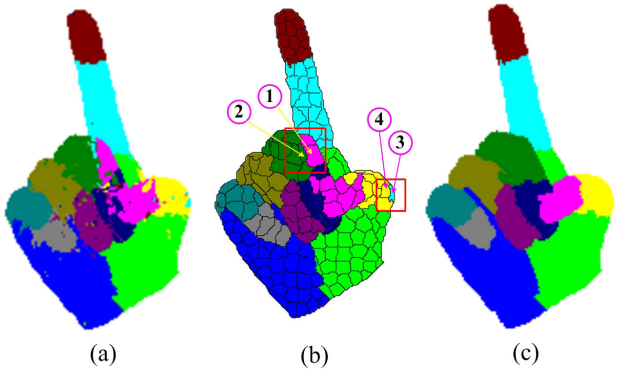


Figure 3.7: Illustration of the effectiveness of MRF inference based on superpixel partition and label co-occurrence. (a) Per-pixel classification result. (b) Superpixel partition result. (c) SMRF inference result.

per-pixel classification results in Fig. 3.7 (a). As the pixels within each superpixel are assigned the same state, the small and scattered misclassified points are largely suppressed even without MRF inference. However, the larger misclassified regions still cannot be removed, *e.g.*, the regions labeled with 1 and 3 in the red rectangles. Region 1 is misclassified to No. 4 hand part and region 3 is misclassified to No. 5 hand part. According to the co-occurrence probability of different hand parts in Fig. 3.8, both these misclassified regions result in a large pairwise energy. First consider region 3. It is small in relative size compared to its surrounding superpixels, *e.g.*, region 4, thus assigning it with the label of its neighbors will produce a big energy decrease, which is favored during inference. For region 1, note it is only adjacent to the superpixels on the middle finger since depth discontinuity is handled to build the MRF framework, thus it will be influenced by only one misclassified superpixel and multiple correctly classified regions in the middle finger. By comparison, its adjacent region 2, which is correctly classified, has much more correctly classified neighbors. Again, region 1 is more inclined to be neutralized by region 2. Fig. 3.7(c) shows the results given by MRF inference, in which the misclassified regions are successfully removed.

Combing the label co-occurrence, node adjacency and weight coefficient $\alpha_{i,j}$, the pairwise

potential function $\psi_{i,j}$ for two adjacent node i and j takes the following form:

$$\begin{aligned}\psi_{i,j}(y_i, y_j) = -\alpha_{i,j} \times [1 - \delta(y_i, y_j)] \\ \times \log\left[\frac{P(y_i|y_j)+P(y_j|y_i)}{2}\right]\end{aligned} \tag{3.18}$$

where δ is the Kronecker delta function to indicate a zero pairwise energy if the node i and j have the same state. $P(y_i|y_j)$ is the conditional probability learned from the training dataset by counting the co-occurrence of the labels of neighboring superpixels. The co-occurrence distribution of the labels is shown in Fig. 3.8. Note here all the $P(y_i|y_i)$ terms are set to zero as they have no effect on the inference results. By incorporating the co-occurrence probability distribution in SMRF inference, we encode the prior to eliminate the unlikely neighboring labels to smooth the results given by per-pixel classification. Based on the above formulation, we adopt the iterated conditional modes (ICM) algorithm [83] to minimize the energy function to get Y^*, and the initial label for each superpixel is taken to be $y_k = \arg\max_l P(l|s_k)$. The ICM algorithm is a greedy search algorithm, and is guaranteed to converge fast.

3.3.3 Computational Complexity Analysis

Overall, the computational cost involved in the proposed hand parsing scheme consists of four parts: per-pixel classification, superpixel partition, MRF network construction and MRF inference. Let the number of pixels in the hand region be n. The RDF classifier is known to have a computational cost of $O(nND_R)$, where D_R is the tree depth. The temporal classifier has a complexity of $O(n)$ as it only involves calculation of the twelve Gaussian terms for each pixel. For superpixel partition with the SLIC algorithm, the complexity is $O(nK_{SLIC})$. We explicitly write out the number of iterations K_{SLIC} for SLIC to converge rather than $O(n)$ [80], as K_{SLIC} is important for the final partition quality.

On average, the number of partitioned superpixels is n/M_S^2. To construct the MRF network, the weight coefficient $\alpha_{i,j}$ needs to be calculated for each pair of adjacent nodes, and the average

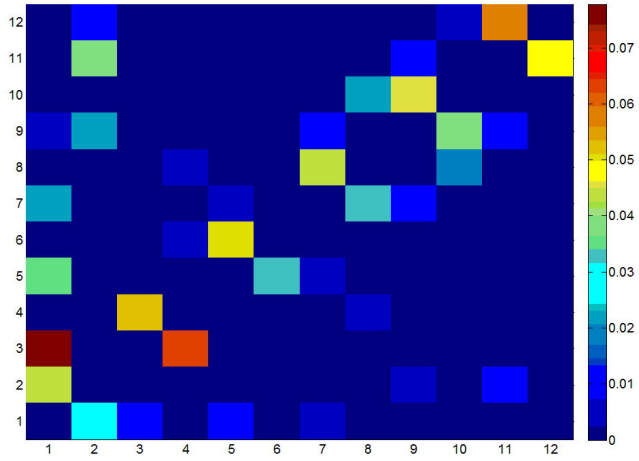


Figure 3.8: The co-occurrence probability distribution of the hand part labels of the neighboring superpixels for $M_S = 4$.

number of neighbors for each node is a constant approximately between four and eight. This results in a complexity of $O(n/M_S^2)$. Finally, the complexity of the ICM algorithm for MRF inference is known to be linear in the number of nodes, *i.e.* $O(nK_{ICM}/M_S^2)$, where K_{ICM} is the number of iterations needed to converge. Thus the overall complexity of the proposed parsing scheme is:

$$C_{Total} = O\left\{n(ND_R + K_{SLIC})\right\} + O(nK_{ICM}/M_S^2). \tag{3.19}$$

Note that all parameters except n can be predefined, thus the overall complexity is indeed linear in the number of the pixels.

3.4 Experimental Results

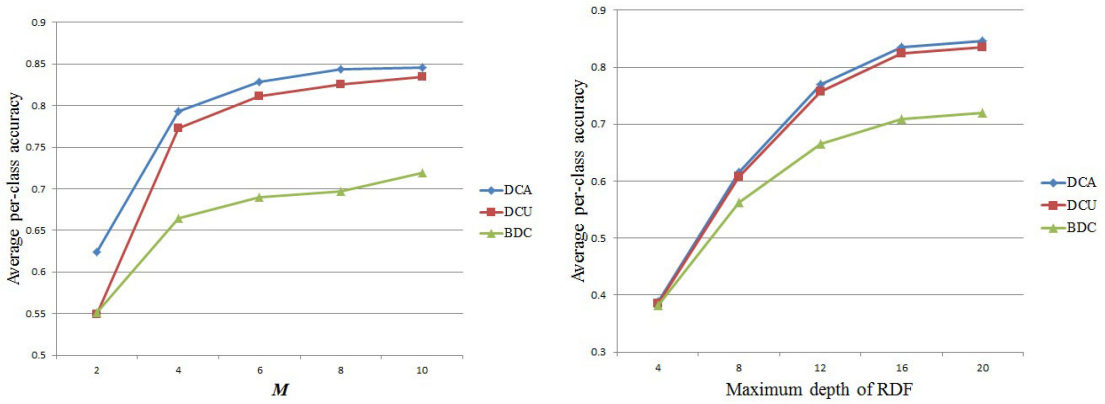
In this section we present the results to validate the effectiveness of the proposed depth context feature with distance-adaptive sampling scheme (DCA) and the SMRF framework by comparing them to the state-of-the-art methods [65, 66, 72]. For hand parsing, the experiments include the tests on single-frame synthesis datasets, synthesis continuous hand motion sequences and

real-world hand motion sequences. To further test the generalization power of the proposed methods to other articulated objects such as the human body, we also present the results to fulfill the human body segmentation task on a public body part annotation dataset [72]. The performances of the proposed methods are compared to the state-of-the-art methods, which are evaluated in terms of average hand part classification accuracy and running time. The whole program was coded in C++/OPENCV without parallelization, and tested on a PC with Intel i5 750 CPU and 4G RAM.

3.4.1 Quantitative Evaluation for Single Frames

We synthesized a dataset of 22.5k templates to quantitatively evaluate the performance of our method on the single frames. The resolution of the images is 320×240. Each template consists of a pair of depth image and the ground truth hand part labels. To generate the dataset, we capture a set of hand articulation parameters by the CyberGlove II [4]. The captured hand articulation parameters are combined with the 3D global rotation parameters in certain ranges to handle the viewpoint variation. Here we define the ranges to be $(-20°, 20°)$ for global rotation around the X and Y axes, i.e. the axes parallel to the image plane of the camera, and $(-35°, 35°)$ around the Z axis, i.e. the axis perpendicular to the image plane. These hand motion parameters are used to drive the 3D hand model in Section 3.1 to generate the templates. To evaluate the classification accuracy on the synthesized images, we use 80% of the templates in the dataset for training and the rest 20% for testing.

First, we illustrate that the proposed DCA feature achieves considerably higher accuracy than the benchmark binary depth comparison feature (BDC) [66]. The feature value in BDC is calculated by the binary depth comparison of two neighboring context points, *i.e.* $f_F = d_z(\mathbf{p} + \mathbf{u}/d_z(\mathbf{p})) - d_z(\mathbf{p} + \mathbf{v}/d_z(\mathbf{p}))$, where $\mathbf{u}$ and $\mathbf{v}$ are a pair of relative offsets of the context points. As in [66], the offset pairs of $\mathbf{u}$ and $\mathbf{v}$ are randomly sampled. In the experiments we set the number of offset pairs in BDC and the number of context points in DCA to be the same so



(a) (b)

Figure 3.9: Comparison of the average per-class accuracy of the three depth features (a) with different values of M and (b) with different values of the maximum depth of RDF.

that the resulting features have the identical dimension. Besides, we also test the performance of the depth-context feature with uniform sampling (DCU), *e.g.*, $g(x)$ is a constant function. Fig. 3.9(a) shows the per-class classification accuracy of the three methods with respect to different number of M, in which the RDF consists of three trees with a maximum depth of 20. The DCA feature outperforms the benchmark BDC feature by 12.2%, in which the distance-adaptive sampling scheme contributes 2.8% improvement. Especially, the DCA method performs quite well even with a very small value of M, *i.e.* $M = 2$. This proves the capability of the distance-adaptive sampling scheme to capture the 3D context of the pixels in depth image. Fig. 3.9(b) shows the per-class classification accuracy with different depth of the RDF at fixed value of $M = 10$. We can see that the DCA method again provide better classification results for all choices of the tree depths.

To validate the effectiveness of the SMRF framework compared to per-pixel classification, we have combined it with the RDF classifier with both the BDC feature and the DCA feature and test the performances on the same dataset. The results are presented in Table 3.1. The label "SMRF-#" represents our method running with different superpixel grid step M_S. The feature grid size is $M = 10$ for DCA, and equivalently the dimension of the BDC feature is 440. Table

3.1 also shows the per-frame time cost for per-pixel classification and the superpixel partition (SP) and MRF inference (MRF) modules of SMRF. Overall, the SMRF method provides the highest increase of 5.5% with $M_S = 4$ for the BDC feature at an extra time cost of $218.2ms$, and 4.8% with $M_S = 4$ for the DCA feature at an extra time cost of $200.8ms$. Compared to the benchmark per-pixel classification with RDF and the BDC feature [66], we achieved an overall 17.4% higher accuracy for the twelve hand part classification for $M_S = 4$.

The results for the SMRF method with different M_S also validates that superpixel is a more appropriate representation to remove the misclassified regions than the pixel. Note that "SMRF-1" is equivalent to per-pixel inference with MRF, and the result is not as good as the superpixel-level counterparts, even at a higher extra time cost of about $380ms$ per frame. This is because at the pixel-level, a pixel within the isolated misclassified region will be influenced equally by its surrounding pixels, among which there are also the misclassified pixels. By comparison, at the superpixel-level, a misclassified superpixel is mostly surrounded by the superpixels with the correct labels, and thus are more likely to be converted by its neighbors. Besides, according to the results in Table 3.1, the size of the superpixel cannot be too large, as it produces over-smoothing effects. Fig. 3.10 shows an example of applying SMRF with different values of M_S. Not the finger regions, we can see the isolated misclassified regions are not well suppressed with $M_S = 1$, while the thumb and index fingers are wrongly smoothed for $M_S = 12$. The size $M_S = 4$ produces the result that best conforms to the ground truth.

We also implement the Multi-layered Random Decision Forests [65] to parse the hand based on the hand part partition scheme in Fig. 3.4. In this approach, the hand configuration parameters of the training data are first clustered into K classes by spectral clustering. During training, a RDF classifier is learned for hand shape classification (SCF), which classifies the whole depth image of the hand into the K classes. An individual RDF classifier (GEN) is trained for hand parsing on each of the K subsets of the original dataset. During testing, the SCF classifier is first applied to the input hand depth image, and its result is used to pick up the GEN classifiers

Table 3.1: Comparison of the parsing results of per-pixel classification with RDF, SMRF inference with different superpixel sizes and the Multi-layered RDFs.

Method	Avg. Accuracy	Avg. Running Time (ms)			
		RDF	SP	MRF	Total
Per-pixel RDF with BDC [66]	71.94%	69.3	-	-	69.3
SMRF-1 with BDC	74.51%	69.1	-	386.4	483.4
SMRF-4 with BDC	**77.40%**	69.2	198.2	20.0	291.0
SMRF-8 with BDC	77.31%	68.9	179.4	6.8	257.1
SMRF-12 with BDC	75.62%	69.9	163.4	4.8	239.8
Per-pixel RDF with DCA	**84.53%**	72.9	-	-	72.9
SMRF-1 with DCA	87.30%	72.7	-	380.3	480.8
SMRF-4 with DCA	**89.29%**	72.8	181.2	19.6	277.1
SMRF-8 with DCA	88.82%	73.1	166.4	7.0	248.5
SMRF-12 with DCA	87.01%	73.0	151.2	4.6	230.6
Multi-layered RDF with BDC [65]	72.14%	235.2	-	-	235.2

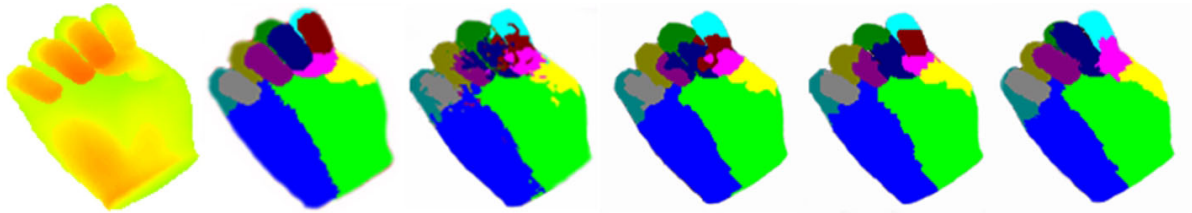


Figure 3.10: Illustration of SMRF performance with different superpixel sizes. First: the input depth image (Left). Second: the ground truth (GT) hand part labels. Third: per-pixel classification with DCA. Fourth: SMRF results with $M_S = 1$. Fifth: SMRF with $M_S = 4$. Sixth: SMRF with $M_S = 12$.

for hand parsing. The BDC feature is used for classification in both the SCF and GEN classifiers. In [65] they report that this method improves the classification accuracy to 91.2% on their own dataset, compared to 68.0% obtained by a single RDF classifier [66]. As in [65], we set $K = 25$, and the SCF classifier is used to determine the three most probable classes of the test image. The corresponding three GEN classifiers are picked up and their results are weighted to give the final parsing. All the RDF classifiers consist of three trees with a maximum depth of 20. The result obtained with the Multi-layered RDF classifier on our dataset is presented

in Table 3.1, which shows very little improvement compared to [66], *i.e.* 72.14% vs. 71.94%. This is not surprising as our dataset consists of a large number of natural hand configurations and the hand pose parameters do not form specific clusters. Therefore, the hand pose parameters within each individual pose cluster still contain large variations, and thus the SCF/GEN framework in [65] cannot work well. By contrast, our proposed SMRF framework shows better generalization capability and gives 5.5% improvement compared to [66].

3.4.2 Quantitative Evaluation for Continuous Sequences

To evaluate how the temporal reference can help to improve the hand parsing results, we synthesized two continuous hand motion sequences with the same procedure in Section 3.4.1, both of which are approximately 500 frames long and contain complex combinations of global hand motion and local finger articulations. The first sequence includes the continuous motion of the hand when it is posing digit gestures, including the transition motion to change from one gesture to another. The 3D hand rotation and transition are also included. The second sequence contains different single/multiple finger motions combined with 3D hand rotation. Especially, these two testing sequences contain some clips in which the hand motions are not covered by the training dataset in order to test the robustness of the proposed parsing scheme.

In this experiment we set the weighting parameter to fuse the RDF classification and the temporal reference to be $\eta = 0.5$, and the superpixel grid size is $M_S = 4$. The per-frame accuracy curves for the two sequences are shown in Fig. 3.11 and Fig. 3.12, in which we compared three methods: per-pixel classification with the proposed DCA feature, the proposed single-frame SMRF with the DCA feature and the temporal SMRF with the DCA feature. Note the average accuracy is shown in log-scale for better illustration. The average accuracies on the whole sequences are summarized in Table 3.2, in which the per-pixel classification results with the BDC feature [66] are also included. For the two sequences, the singer-frame SMRF improves the per-pixel classification results with DCA by 4.7% and 4.0% respectively. The tem-

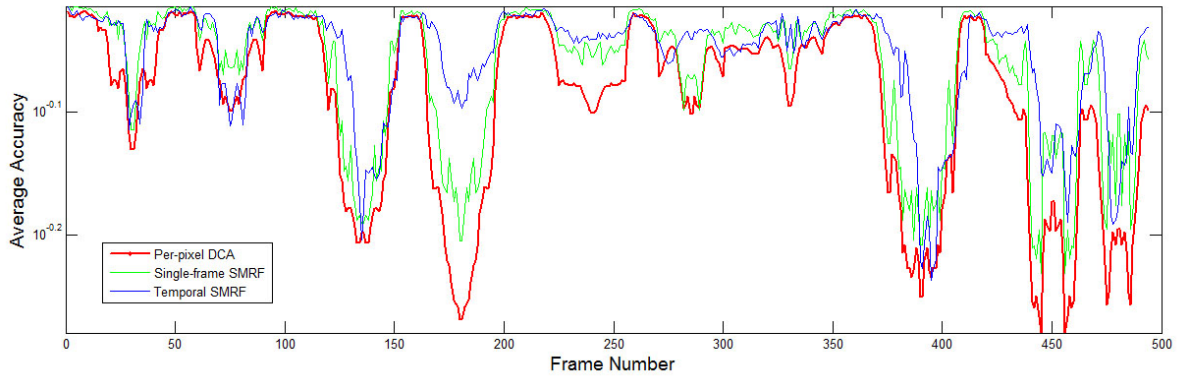


Figure 3.11: Comparison of the hand parsing results using per-pixel classification with DCA, single-frame SMRF and SMRF with temporal reference on the first synthesized continuous hand motion sequence.

poral SMRF achieves further 1.5% and 0.7% improvement respectively on the two sequences, compared to the single-frame SMRF.

Table 3.2: Comparison of the classification results on the two synthesized hand motion sequences.

	Per-pixel BDC [66]	Per-pixel DCA	Single-frame SMRF	Temporal SMRF
Seq. 1	77.2%	82.4%	87.1%	89.6%
Seq. 2	81.7%	86.8%	90.8%	91.5%

While the improvement obtained by the temporal reference seems not striking in terms of the average accuracy, it is important to see that it largely increases the overall robustness. Since some parts of the testing sequences are either not covered by the training dataset or are going through complex hand motion and self-occlusion, the performances by per-pixel classification with the pre-trained classifier and the SMRF method that built upon it can suffer from drastic degradation, *e.g.*, frames between 150 and 200 in Seq. 1 and frames between 130 and 160 in Seq. 2. In these parts we see the system performance is dramatically improved by fusing the temporal information, and thus the resulting overall classification accuracy remains much more stable than the single-frame based counterparts.

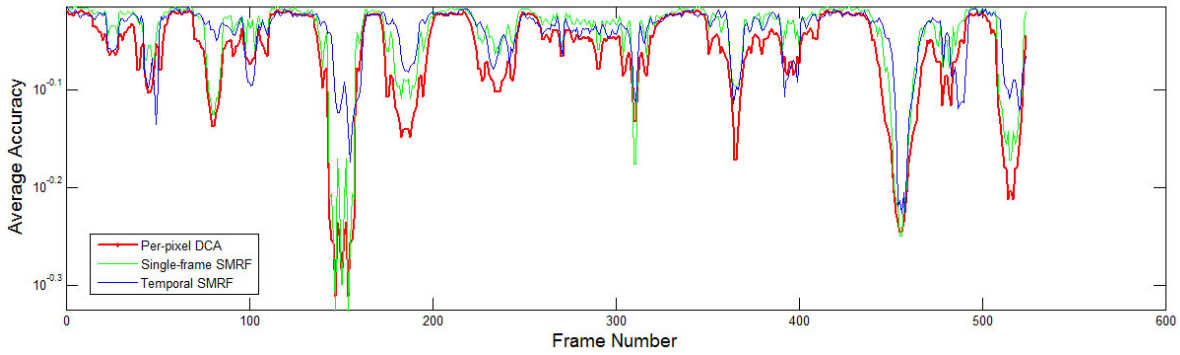


Figure 3.12: Comparison of the hand parsing results using per-pixel classification with DCA, single-frame SMRF and SMRF with temporal reference on the second synthesized continuous hand motion sequence.

3.4.3 Overall Qualitative Evaluation

We further test the parsing performance of the SMRF framework on real-world input sequences captured by a SoftKinetic DS325 depth camera,which is about 900 frames long, and the result is illustrated in Fig. 3.14. The parsed hand parts are shown with different colors, which is consistent with the labeling scheme in Fig. 3.4 (a). The results show the effectiveness of the SMRF method. From the figure we can see that the results of per-pixel parsing are very noisy. In addition, its performance for small finger parts get even worse for the challenging cases such as tightening fingers or when some fingers are occluded. By comparison, the SMRF method produces more meaningful parsing in such cases.

3.4.4 Human Body Part Segmentation

We adapt the proposed hand parsing scheme to human body part segmentation to demonstrate its generalization power to different kinds of articulated objects, and test it on the body part annotation dataset in [72]. The idea of the body part segmentation method in [72] is close to ours in that they also improve the RDF classification results by utilizing the spatial-temporal context in a graph cut optimization framework. The dataset consists of 500 frames of annotated

human body parts and the corresponding depth and color images, in which each of two subjects stands still in front of the camera and performing various body and arm motion. The resolution of the images is 640×480. The body partitions consist of seven parts: the torso, the left/right upper arms (LU/RU), the lower arms (LW/RW) and the hands (LH/RH). In this experiment we compare the per-pixel classification results with the RDF classifier using the BDC and DCA features separately, based on which the SMRF framework is also tested. We use the same setting as [72], *i.e.* we perform 5-fold cross-validation on the dataset, and select 1000 pixels per image for training. The RDF consists of 3 trees and each tree has a maximum depth of 20. Due to the larger resolution of the images, the superpixel size is set to be $M_S = 12$ for SMRF inference. For this experiment we do not utilize the temporal references.

The performances of the proposed parsing scheme are compared with [72] in Table 3.3, in which we also cite their best results with three different methods. The first is their implementation of RDF classification with the BDC feature. The second is frame-by-frame graph-cut optimization, which uses the single-frame information. The third is temporal-coherent graph-cut optimization, which incorporates multiple frames for inference and reports the highest accuracy of 87.91%. Note that our implementation of the RDF classification using the BDC feature achieves similar average accuracy to [72], *i.e.* 81.43% vs. 81.95%, while the classification accuracies are different for the individual classes. This can be a result of our different selection strategy of the training samples. In our implementation to train the RDF classifier, we select the 1000 sample pixels from each training image by choosing the same number of pixels from each class, and thus the training samples are balanced for all the classes. In [72], the 1000 samples are selected uniformly in the whole image. As the torso part occupies the largest area and the number of torso samples in the training data is thus more than the others, their trained RDF classifier gives much higher accuracy for the torso over the other parts.

Overall, the results in Table 3.3 show the effectiveness of the proposed SMRF method. With the BDC feature, the SMRF method improves the per-pixel results by 8.88% from 81.43% to

90.31%. By comparison, the frame-by-frame graph cut method in [72] achieves 5.06% increase compared to per-pixel classification. By inference in multiple frames, their temporal-coherent graph cut method achieves 5.96% increase. The results with the DCA feature are quite close to that of the BDC feature though. The reason may be due to the difference between the hand and the body, *e.g.*, the hand has more serious self-occlusion and can rotate freely around all three axes. Several body segmentation examples are given in Fig. 3.13. The first row is the ground truth annotations. The second and fourth rows are the per-pixel classification results with the BDC and DCA features respectively, and the third and fifth rows are the improved results with SMRF respectively. In the black rectangles we show some failure cases of the SMRF method. In these cases the per-pixel results are too noisy to recover the correct label.

Table 3.3: Comparison of body part segmentation results between our methods and [72].

Method	Torso	LUA	LWA	LH	RUA	RWA	RH	Avg.
Per-pixel RDF & BDC [72]	94.1	79.8	78.7	76.6	81.2	83.1	80.2	82.0
Frm-by-Frm Graph Cut [72]	98.9	75.0	83.4	92.4	77.5	87.7	94.2	87.0
T.C. Graph Cut [72]	98.4	78.9	84.4	88.3	82.6	88.9	93.9	87.9
Per-pixel RDF & BDC (Ours)	86.5	88.9	84.7	71.4	87.4	81.3	69.8	81.4
SMRF-12 & BDC (Ours)	94.1	93.6	90.4	87.3	93.1	87.8	85.8	**90.3**
Per-pixel RDF & DCA (Ours)	87.2	90.1	87.3	72.8	89.4	84.2	70.5	83.1
SMRF-12 & DCA (Ours)	94.4	93.3	91.7	85.5	93.3	88.9	82.4	89.9

3.5 Gesture Recognition Applications

We propose to utilize the extracted 3D key points to recognize the hand gestures by matching to pre-defined templates. There are totally ten gestures to be recognized. The corresponding hand shapes of the gestures are illustrated in Fig. 3.15, which map to the digit numbers from zero to nine. The gesture templates are defined in a tree hierarchy as that in Fig. 3.4 (b), with different configurations of key point positions corresponding to different gestures, *i.e.* $\{U_t|t = 1, ..., 10\}$, and each template consists of L 3D key points: $U_t = \{\mathbf{v}_t^l|l = 1, ..., L\}$. Given the key point set

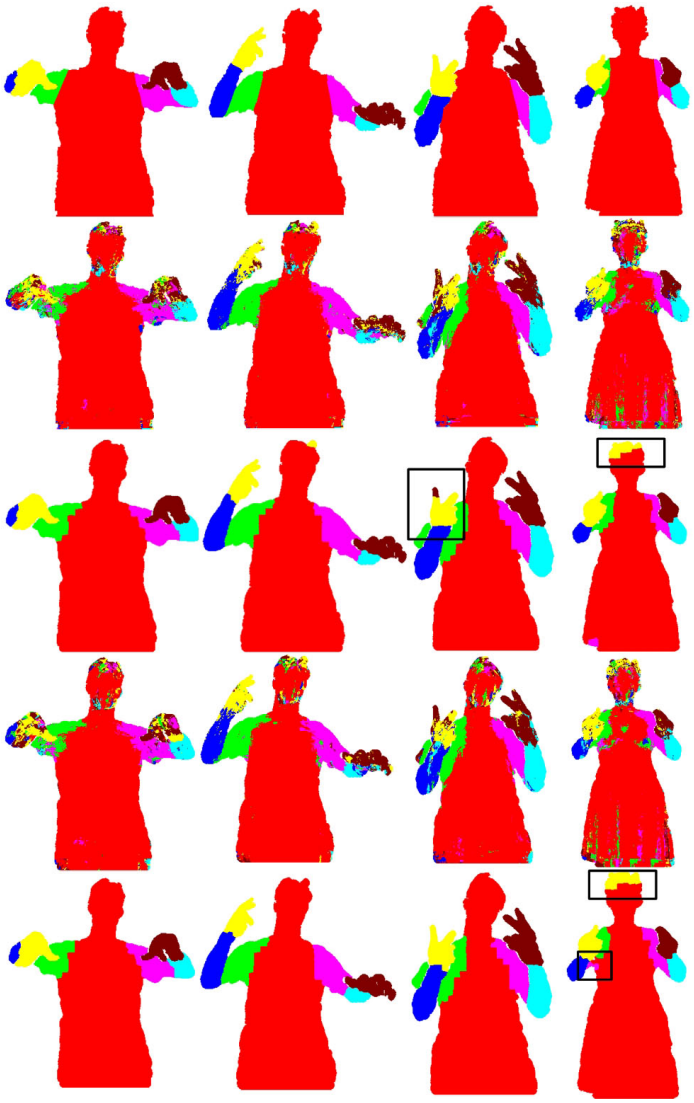


Figure 3.13: Comparison of the human body part segmentation results. Ground truth (first row), per-pixel classification with BDC and RDF (second row), SMRF with BDC (third row), per-pixel classification with DCA and RDF (fourth row), SMRF with DCA (fifth row). The black rectangles show the failure cases of SMRF.

$U_I = \{\mathbf{v}_c^l | l = 1, ..., L\}$ extracted from the input depth image following the method in Section 3.2.3, the input gesture can be recognized by finding the template that best fit U_I. The problem can thus be formulated as:

$$t^* = \arg\min_t D_U(U_I, U_t). \tag{3.20}$$

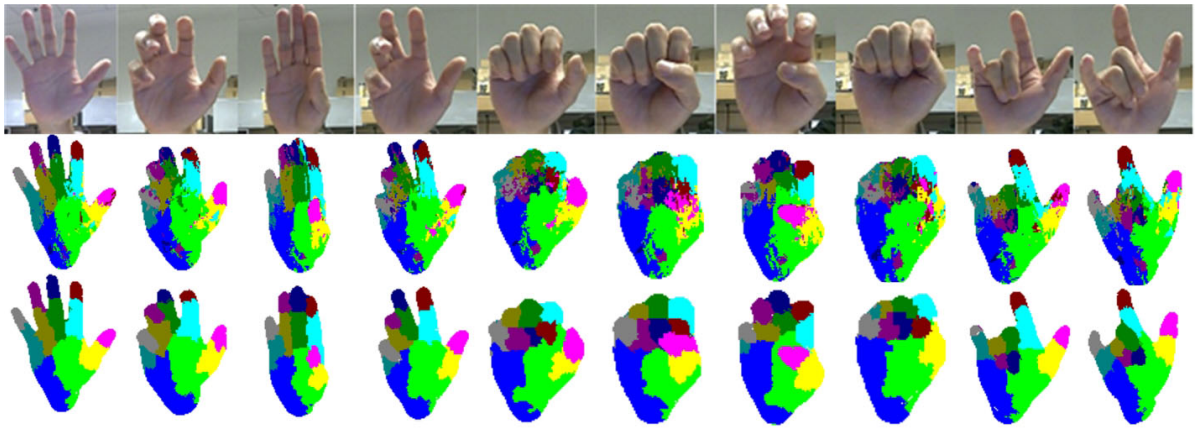


Figure 3.14: Comparison of the hand parsing results using per-pixel classification with DCA and the RDF classifier (middle row), and the proposed Temporal SMRF framework (lower row) on real-world hand motion sequences.

The key issue in this problem is how to measure the distance D_U against the viewpoint change of the input and the hand size variations. To this end, we propose to use 3D point set matching to evaluate D_U. These two sets are first aligned with the least-squares fitting method in [101], and the distance metric between key point sets is thus defined as:

$$\begin{aligned} D_U(U_I, U_t) &= \|U_t - R_U U_I - T_U\|^2 \\ &= \sum_{l=1}^{L} \left\|\mathbf{v}_t^l - (R_U \mathbf{v}_c^l + T_U)\right\|^2, \end{aligned} \tag{3.21}$$

where R_U is the rotation matrix and T_U is the translation vector to align the point sets U_t and U_I. As shown in [101], the above problem can be solved in two steps. First, the points in U_t and U_I are normalized to have zero mean. Let the means of U_t and U_I be μ_t and μ_I respectively. The optimal rotation matrix R_U^* can be found by solving the below problem:

$$R_U^* = \arg\min_{R_U} \sum_{l=1}^{L} \left\|(\mathbf{v}_t^l - \mu_t) - R_U(\mathbf{v}_c^l - \mu_I)\right\|^2. \tag{3.22}$$

Let $Cov(U_I, U_t)$ be the cross-covariance matrix between the point sets in U_t and U_I and its SVD decomposition to be $Cov(U_I, U_t) = U\Lambda V^T$, where U and V are two 3×3 orthogonal matrices and Λ is a 3×3 diagonal matrix with nonnegative elements. The optimal rotation matrix is

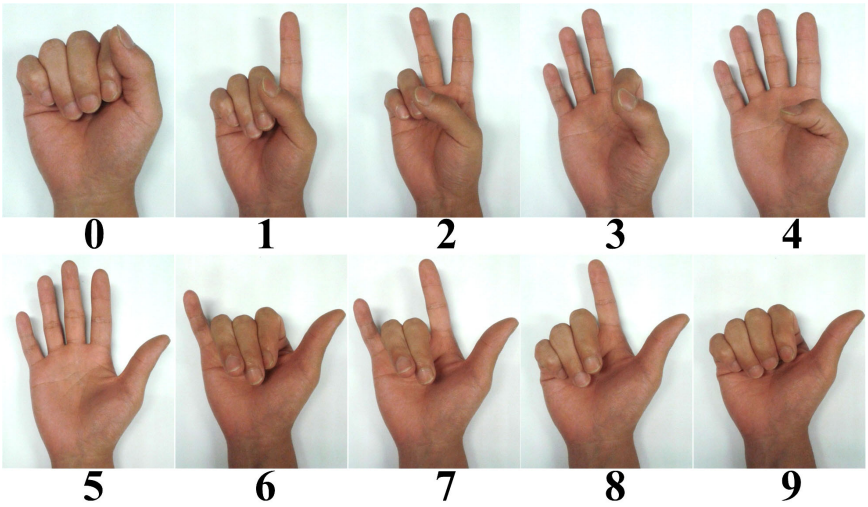


Figure 3.15: The hand shapes corresponding to the digit gestures.

Figure 3.16: Human-virtual avatar interaction via hand gesture recognition.

thus obtained via $R_U^* = VU^T$. In the second step, the optimal translation vector T_U^* is obtained by $T_U^* = \mu_t - R_U^* \mu_I$. Based on this gesture recognition algorithm, we have built a human-virtual avatar interaction system to enable the user to communicate with the virtual avatar. In this system, the avatar can recognize the hand gesture of the user and simulate it to provide the visual feedback. In Fig. 3.16 we show some sample figures to illustrate the interaction process between a user and the virtual avatar.

3.6 Discussion

This chapter presents a novel hand parsing scheme with the input of depth images. Our contributions mainly include a depth context feature with the distance-adaptive sampling scheme which

proves more accurate for hand parsing compared to the binary depth comparison feature, and a Superpixel-MRF framework to enforce both the spatial and temporal constraints to improve the per-pixel classification results. The results on both the synthesized depth images and real-world test sequences demonstrate the good performance of the proposed hand parsing scheme compared to the state-of-the-art methods. The tests on the human body annotation dataset further demonstrate the generalization power of our method to different kind of articulated objects. While we only utilize a basic random forest for per-pixel classification, the parsing accuracy can still be improved with better forest learning techniques [106, 111]. More importantly, in our follow-up work in Chapter 5 on discriminative hand pose estimation, we show that the parsed hand parts have encoded the correlations between the hand joints, and thus can be utilized to improve the pose prediction accuracy.

Chapter 4

Model-based Hand Pose Estimation

This chapter presents a model-based framework for hand pose estimation from depth image sequence input, which aims to restore both the global hand motion and local finger articulations parameters [2, 112]. To handle the high dimensionality of the hand pose parameters, the proposed framework adopts a divide-and-conquer scheme to estimate the global and local hand poses successively. Especially, a fingertip tracking algorithm is developed to track the 3D positions and identities of the five fingertips, which provide an initial estimation of the hand pose via Inverse Kinematics. Starting from a reference hand pose, an articulated iterative closest point approach is adopted to refine the hand pose by fitting the mesh surface of the hand model to the input point cloud. Experiments on both synthetic data and real-world sequences show the hand pose estimation scheme can accurately capture the natural hand motion. In addition, we developed two virtual reality applications to take advantage of the estimated hand pose to interact with the virtual environments.

Fig. 4.1 illustrates the main modules of the proposed system. The 3D hand model developed in Section 3.1 is used for model fitting to the input depth data, which can calculate the positions of hand joints and the 3D mesh vertices with a feasible hand pose hypothesis. Its skeleton is used for Inverse Kinematics calculation. Based on the availability of the temporal reference of hand pose estimation, the system can work in either the intra mode or in the joint mode. The intra

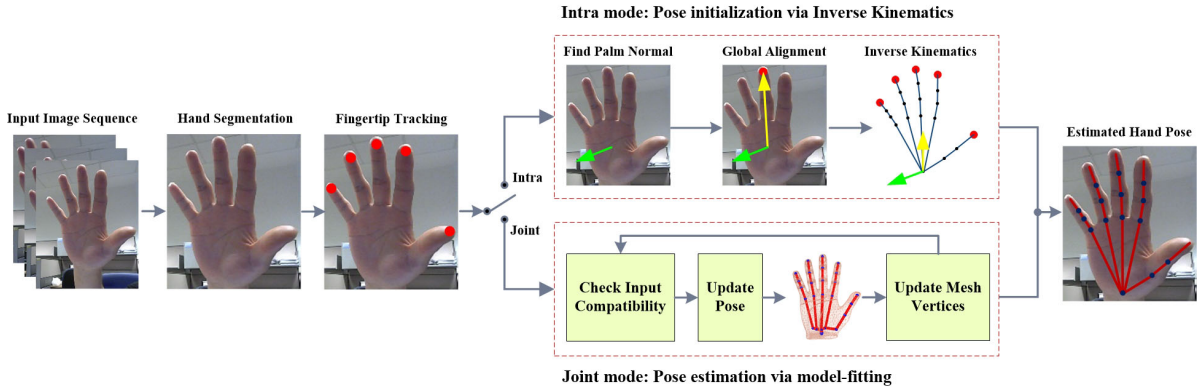


Figure 4.1: Overview of the proposed model-based hand pose estimation framework.

mode corresponds to the initialization or re-initialization stage for hand pose tracking, when the hand pose is recovered using only the information of the current frame. The global hand pose is first recovered by analyzing the spatial relationship between the palm and the middle fingertip. With the hand globally aligned, the local finger motion parameters are estimated by Inverse Kinematics using the detected fingertips as end-effectors. In the joint mode, the hand pose is recovered based on both the reference pose estimation result in the previous frame and the information of the current frame via model-based fitting. In this mode the pose parameters are optimized to align the 3D hand model to the input point cloud and the fingertip positions are used to obtain initial finger pose parameters for model-fitting. This mode is more robust than the intra mode, as the matching quality between the point clouds of the input depth image and the 3D hand model enforces extra constraints during model-fitting.

4.1 Hand Segmentation

Since robust hand segmentation itself is a difficult problem [70, 71], we simplify the problem by making several assumptions on the static hand configuration and utilizing the morphology of the hand to segment it in the depth image [2]. First, we assume hand is the nearest object to the camera and constrain global hand rotation within $(-15°, 15°)$ for global rotation around the X and Y axes, *i.e.* the axes parallel to the image plane of the camera, and $(-70°, 70°)$ around

the Z axis, *i.e.* the axis perpendicular to the image plane. Second, the depth value differences within the forearm and hand region are assumed to be less than a threshold, *e.g.*, $0.2m$. Third, based on morphology of hand, we assume that hand palm forms a globally largest blob in the hand and forearm region in the depth image when the hand rotates within the predefined range. With these constraints, the hand region is segmented by first locating the palm blob and then utilizing it to refine the foreground region in the depth image. The pipeline is illustrated in Fig. 4.3, which consists of three steps: foreground segmentation, palm localization and hand segmentation. Let the depth image be I. The foreground region is obtain by per-pixel threshold of the whole depth image at first, *i.e.* $F = \{p_i | z_i < z_0 + \sigma, p_i \in I\}$, where z_i is the depth value of the pixel p_i, z_0 is the minimum depth value in the depth image and σ is a positive threshold value. This ensures that both hand and forearm regions are extracted from the depth frame, as illustrated in Fig. 4.3 (b).

We then calculate a segmentation line based on the palm region and the orientation of the forearm to separate hand and arm from each other, which should go through the hand wrist and be perpendicular to the forearm. Based on the hand morphology within the specified viewpoint, *i.e.*, $(-15°, 15°)$ around X and Y axes and $(-70°, 70°)$ around Z axis, the palm region is approximated with a circle $\mathscr{C}$, which equals to the largest inscribed circle of the foreground region in Fig. 4.3 (b). Such a circle can be efficiently obtained via distance transform algorithm [119]. Specifically, we calculate a new image with the contour points of the foreground mask, in which the value of each pixel is set to the distance to the nearest contour points if the pixel is in the foreground F, and zero otherwise. The distance transform image is illustrated in gray in Fig. 4.3 (c). The inscribed circle $\mathscr{C}$ of the foreground contour thus has the property that its center is located at the pixel that has the maximum distance to contour and its radius equals to the distance value at this pixel, as shown in Fig. 4.3 (c). To improve the accuracy of palm localization, a 2D Kalman filter is used to track the center of $\mathscr{C}$.

To assist segmentation, we further define a 2D line to approximate the orientation of the

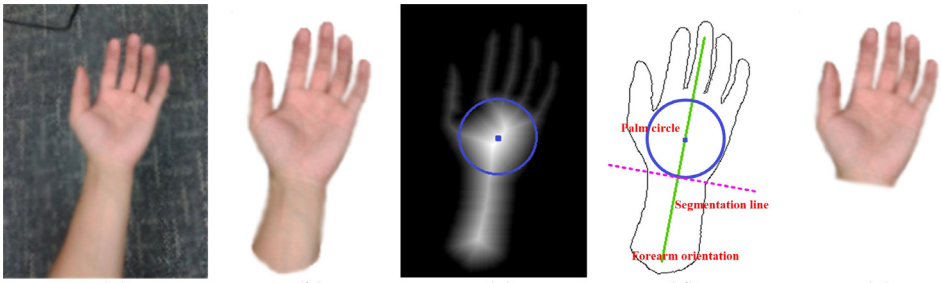


Figure 4.2: Hand segmentation pipeline. (a) Original input image. (b) Foreground segmentation. (c) Palm circle identified by distance transform. (d) Forearm orientation line and segmentation line. (e) Final hand segmentation.

forearm, *i.e.*, the line lies parallel to the forearm region. Note that the foreground region has a lathy shape, and we thus perform Principal Component Analysis (PCA) [120] on the 2D positions of the foreground pixels in F to find such a line. First, the positions of the pixels in F are normalized to have zero mean, *i.e.*, $F_n = \{p_i - \mu_F | pi \in F\}$, where μ_F is the average position of the foreground pixels. The 2D orientation of the forearm can be regarded as the direction on which the contour pixel coordinates in F_n have the largest variances, which can be obtained by:

$$\boldsymbol{w}_f = \underset{\|\boldsymbol{w}\|=1}{\arg\max} \sum_{p_i \in F_N} \left(p_i^T \boldsymbol{w}\right)^2. \tag{4.1}$$

We further enforce this line to go through the palm center, as illustrated by the green line in Fig. 4.2 (d). Based on this forearm orientation line and the palm circle, the segmentation line to separate hand and forearm can thus be defined as the line that is both perpendicular to $\boldsymbol{w}_f$ and tangent to $\mathscr{C}$. As can be seen in Fig. 4.2 (d), there are two lines that can meet these two requirements, and we select the lower one that goes through the hand wrist, *i.e.*, the dotted line in Fig. 4.2 (d). Finally, the segmented hand region is obtained by dropping the foreground part under the wrist line, as illustrated in Fig. 4.2 (e).

Figure 4.3: Difficult examples for 3D fingertip tracking. Left: multiple side-by-side fingers. Middle: bending fingers. Right: nearby fingertips: the thumb and index fingertips are too close to be labeled correctly.

4.2 Fingertip Detection & Tracking

This section presents an approach for robust 3D fingertip tracking in depth images [2]. Despite the large amount of related work, many previous methods only focus on extracting 2D fingertips and cannot detect or track fingertips robustly for a freely moving hand [44–46, 68]. Research in 3D fingertip localization and tracking is still limited and cannot meet the needs for real-life applications [52, 53]. Difficulty in robust 3D fingertip tracking lies in several aspects. First, multiple fingertips of several side-by-side fingers are hard to be distinguished due to lack of discriminative patterns in color or depth. Second, the traditional contour analysis based methods cannot locate the fingertips for bending fingers, which do not form a protrusion on the contour. Third, given the positions of the detected fingertips, it is still a challenging problem to label each detected fingertip correctly, *e.g.*, pinky, thumb, *etc*. Fig. 4.3 illustrates the problems.

In the proposed framework, the task of fingertip tracking is to find the 3D locations of the five fingertips and assign a finger label to each of them in successive depth image sequences. Let the fingertip locations and labels be $\{v_k, l_k\}_{k=1}^{5}$, where $l_k \in L$ consists of the pinky, ring, middle, index and thumb fingers. The proposed method is largely inspired by the concept of Accumulative Geodesic Extrema (AGEX) in the 3D point cloud data [54], which shows that geodesic distances on a 3D surface mesh are largely invariant to mesh deformations and rigid transformations. Here the geodesic distance is defined as the shortest distance between

two points along the 3D surface containing them. Such an invariance property is useful to describe a point on the 3D surface of an articulated object like the hand and body under various deformations. By assuming the endpoints of the body to be geodesic extrema, *e.g.*, head, hands and feet, the method has proven effective to locate these points in depth images. However, despite that the fingertips on the hand share similar characteristic to these endpoints on the body, AGEX alone is incapable of locating and classifying the fingertips robustly, due to the much higher flexibility of hand. For instance, the hand posture examples in Fig. 4.3 are very common in natural hand motion, while the AGEX extraction algorithm [54] cannot locate the multiple nearby fingertips as it assumes the endpoints are sparsely located in the depth image and its computational complexity is high since Dijkstra's algorithm needs to be performed every time when an AGEX point is to be extracted. Also, the AGEX extraction algorithm only locates the fingertips, but their finger identities remain unknown.

To this end, we calculate a geodesic distance map (GDM) by running the Dijkstra's algorithm only once to estimate the geodesic distance between each pixel in the hand region and the palm center, which is more computationally efficient. We then directly utilize this GDM as input features to infer the fingertip positions. As illustrated in Fig. 4.4, the geodesic distance between each fingertip and the palm center is relatively stable against hand deformations, and the fingertip point usually locally maximizes the geodesic distance from the palm center within each finger region. However, due to self-occlusion of the hand, the geodesic distances may not be correctly estimated for all points with the hand region. The proposed method address these issues by imposing more constraints on possible fingertip locations based on GDM and using two extra depth-based features to differentiate the fingertip and non-fingertip points. First, it assumes that fingertips can be only detected where the depth values of pixels are discontinuous. Second, the relative depth differences between one pixel and its neighboring pixels are useful to differentiate fingertip and non-fingertip points. An example is illustrated in the middle of Fig. 4.4, where fingertips are denoted as crosses and non-fingertips are denoted as asterisks. It

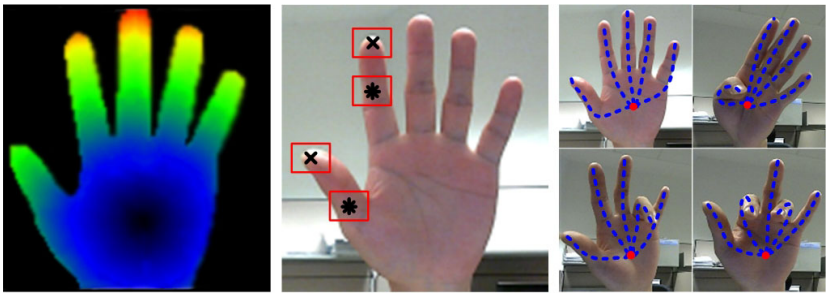


Figure 4.4: Depth features for fingertip detection. First: the geodesic distance map. Second: local rectangle feature. Third: geodesic shortest path.

can be easily seen that the pattern in the rectangles centered at the fingertip/non-fingertip pixels are quite different, *e.g.*, less than half of the pixels in the rectangle take similar depth values to the center pixel for the fingertip, while the non-fingertip pixels are just opposite. Therefore, we propose a local rectangle feature to take advantage of such patterns for fingertip classification. Third, since the positions of the fingertips are not reliable enough for finger label assignment, we propose to utilize the 3D geodesic shortest path (GSP) from the palm center to the candidate fingertip calculated on GDM to fulfill this task, as illustrated in the right of Fig. 4.4. The order of the pixels on different GSPs is less sensitive to hand deformation, which proves quite robust to classify the fingertips in practice. To smooth the predicted fingertip positions and track their identities, we use the particle filter [69] to track each fingertip over successive frames.

The overall pipeline consists of an initialization stage and a tracking stage. During initialization, the user is requested to pose the hand so that the fingers are not side-by-side, *e.g.*, the input hand in Fig. 4.5. The fingertip positions are detected using three depth-based features. Each fingertip is then given a label $l \in L$ with a novel GSP-based voting strategy. The second stage starts only when all five fingertips are detected in the first stage. In the second stage, each of the detected fingertips in the first stage is tracked with a single particle filter. Note that the first stage can be performed not only at the first frame, but also at intermediate frames for automatic re-initialization during the fingertip tracking process when all five fingertips are detected.

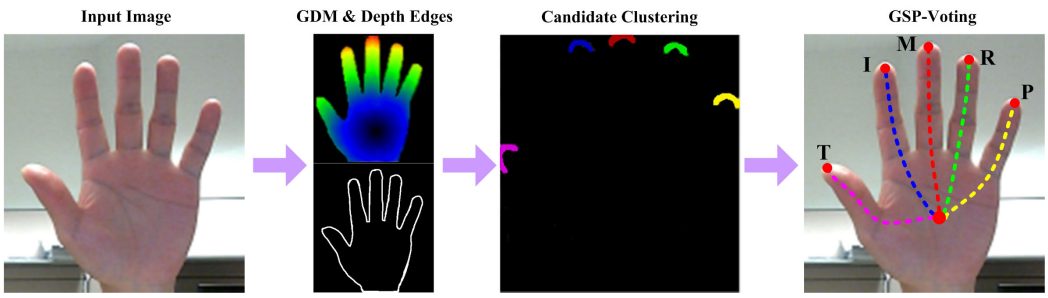


Figure 4.5: The initialization stage for fingertip tracking.

4.2.1 Initialization

During the initialization stage of fingertip tracking, the purpose is to detect and classify all five fingertips in a single depth image, as illustrated in Fig. 4.5. Let the extracted hand regions in the depth image be F_h. To construct GDM, the depth measurements within F_h are transformed into a 3D point cloud V_h using pre-calibrated camera projection parameters. As in [54], we first build a graph $G_h = (V_h, E_h)$, where the points in the 3D point cloud V_h constitute the graph vertices, and E_h is the set of edges among V_h. For each pair of vertices $(i, j) \in V_h$, there is an edge between i and j if and only if they are in the 8-neighbohood of each other in the depth image coordinates and their depth difference is within threshold τ. The weights of the edges in E_h are defined as the 3D Euclidean distances between the vertices connected by them. In addition, we define the source point in G_h as the point that corresponds to the center of the detected palm circle $\mathscr{C}$ in Section 4.1. Therefore, GDM is obtained by calculating the shortest distance from the palm center to each of the vertices in the graph G_h. The value of each point in GDM equals to the calculated shortest geodesic distance. In addition, we define the geodesic shortest path for each point on the graph G_h and the palm center to be the shortest path between them on G_h, and the GSP points for each point in GDM to be the set of points on such shortest path, which will be used to vote for the finger label l.

Since the quality of the hand depth image captured by the Kinect sensor is generally not good, G_h may not be well connected, *e.g.*, some pixels on a bending finger can stay isolated.

Therefore, we utilize the union-find algorithm [121] to search for a set of connected components in G_h, and the one containing the palm center is identified as the master component. Each remaining component is then connected to the master component by adding an edge between the pair of nearest vertices between them. With the refined graph G_h, GDM is generated by calculating the geodesic distance from the center of $\mathscr{C}$ for each vertex in V_h using Dijkstra graph search.

Besides the fingertips, some other pixels in GDM can also have large and locally maximum geodesic distances, *e.g.*, the pixels at lower palm corner. As analyzed previously, the local pattern can help to differentiate between fingertip/non-fingertip pixels, which is represented as a local rectangle feature $\mathcal{R}$. We define $\mathcal{R}$ to represent the pattern inside a binarized rectangle patch centered at the query pixel. Specifically, for a pixel i, its local rectangle feature $\mathcal{R}_i$ is obtained via comparing the depth value of i to every other pixel q within the rectangle:

$$\mathcal{R}_i(q) = \begin{cases} 1 & if \ |z_i - z_q| \leq \delta \\ 0 & otherwise \end{cases}, \tag{4.2}$$

where z_q is the depth value at the pixel q inside the rectangle, δ is a threshold value of approximately $1cm$ and $\mathcal{R}_i(q)$ is a binary value to indicate whether pixel q has similar depth value to the center pixel i or not. Fig. 4.6 illustrates some example of local rectangle features for the fingertip and non-fingertip pixels calculated on the depth images, which shows that the pattern $\mathcal{R}$ differs considerably for fingertip/non-fingertip pixels, especially when the fingertips are isolated from each other, *e.g.*, fingertip pixels have much less neighboring pixels with binary value 1 than non-fingertip pixels. In such condition, the local rectangle feature for the fingertip takes a small η while opposite for the non-fingertips. Therefore, we further define η as the ratio of the number of nonzero points in $\mathcal{R}$ to its total size, and the value of η indicate the occupancy pattern of 1's within the rectangle of $\mathcal{R}$. η will be used to classify the fingertips in the following sections.

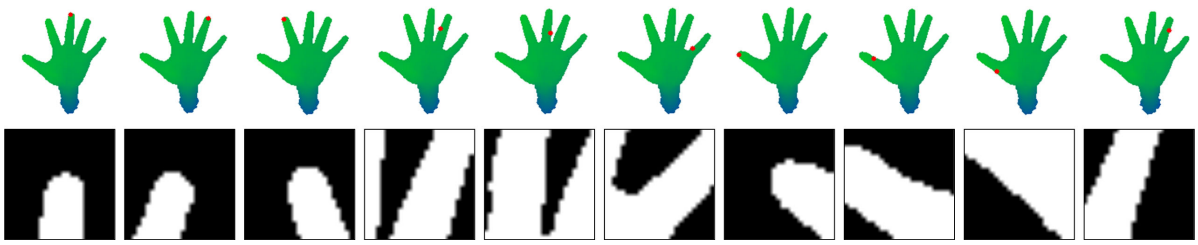


Figure 4.6: Examples of local rectangle feature for fingertip and non-fingertip points. First row: locations for feature calculation, which are denoted as red circles. Second row: local rectangle features, in which white regions denote binary value of 1 and black regions denote 0.

We now summarize the fingertip initialization pipeline. Let the geodesic distance of a vertex i on GDM be ω_i. For a stretched input hand similar to that in Fig. 4.5, a fingertip can only locate where the geodesic distance takes large value and is locally maximized, and the points around a fingertip take a small η, *i.e.*, $\omega > \omega_T$ and $\eta < \eta_T$, where ω_T and η_T are thresholds. In addition, the possible locations of fingertips are also confined within the set of border points of the depth image, *e.g.*, the pixels where depth value is discontinuous. In our implementation, the depth edge map is generated by comparing a pixel's depth value to its eight neighboring pixels. If the number of a pixel's neighboring pixels with depth differences larger than a threshold is more than three, we assume the pixel to be staying on the edge. Let the set of border points be A_b. Based on these observations, the border points that satisfy both the geodesic distance and the occupancy pattern constraints are taken as the candidate pixels where the fingertips may locate. These candidates are then dilated by a 3×3 window and clustered into M clusters based on their 2D coordinates, and the small clusters with sizes less than N are dropped, *e.g.*, isolated candidate pixels that may result from noisy data. The largest five clusters are kept if there are more than five remaining clusters. Finally, for each cluster, the pixel that maximizes the geodesic distance within the cluster are taken as a fingertip. Since the depth value of each pixel is known, the 3D fingertip positions can be obtained by back-projecting the pixel into 3D space. Algorithm 1 summarizes the steps to locate the fingertips.

If Algorithm 1 fails to detect all five fingertips, it simply waits for the next frame and per-

Algorithm 1: Fingertip localization.

1: Generate geodesic distance and depth edge maps;
2: Candidate pixel selection: $A_c = \{i | i \in A_b, \omega_i > \omega_T, \eta < \eta_T\}$;
3: Clustering in A_c: $A_c = A_1 \cup A_2 \cup ... \cup A_M$;
4: Drop small clusters: $A_c^* = \{A_i | A_i \in A_c, |A_i| \leq N\}$;
5: Sort clusters in A_c^* via their size and keep the largest five clusters if $|A_c^*| \geq 5$;
6: Fingertip detection: $v_k = \arg\max_{i \in A_k} \omega_i$, $A_k \in A_c^*$;

forms the above procedure again. Otherwise, *i.e.*, all five fingertips are detected, we need to classify them to get their finger labels $\{l_k\}_{k=1}^5$. However, the relative positions of the fingertip are not reliable for labeling as they show large spatial uncertainty. By contrast, in the right of Fig. 4.4 the order of the geodesic shortest paths from the palm center to the detected fingertips demonstrates certain invariance to the various fingertip configurations, which can be robust for fingertip labeling. Therefore, we utilize the pixels on these paths to vote for the finger label l_k for each detected fingertip v_k, which is readily available after the Dijkstra's algorithm is run to generate GDM. Let the set of pixels on the geodesic shortest path for v_k be U_k. This voting strategy is based on the fact that the order of points on different GSPs remains stable against finger bending and global hand transformation, especially for the points near the palm center. For a fingertip v_k, a five element counter array $\Gamma_{k,j}$, $j = 1, 2, ..., 5$ is maintained to estimate the probability that v_k takes the finger label l_j, and the array values are determined by accumulating the relative order of the pixels on the five GSPs. Let the palm center be p_p, which is obtained in Section 4.1. The fingertips are thus labeled using the Algorithm 2. Fig. 4.7 illustrates the pipeline for GSP-voting, where white cells indicate high probability and vice versa. In Fig. 4.8 we further present the examples of GSP-voting on real-world testing sequences used in [100], which illustrates the indices of unclassified fingertips, the classified fingertips and accumulated confusion matrices for voting. Note the indices of the fingertips before classification are not in certain order. Rather are mostly random according to Algorithm 1. The thumb, index, middle, ring and pinky fingertips are denoted by red, green, blue, yellow and pink circles respectively

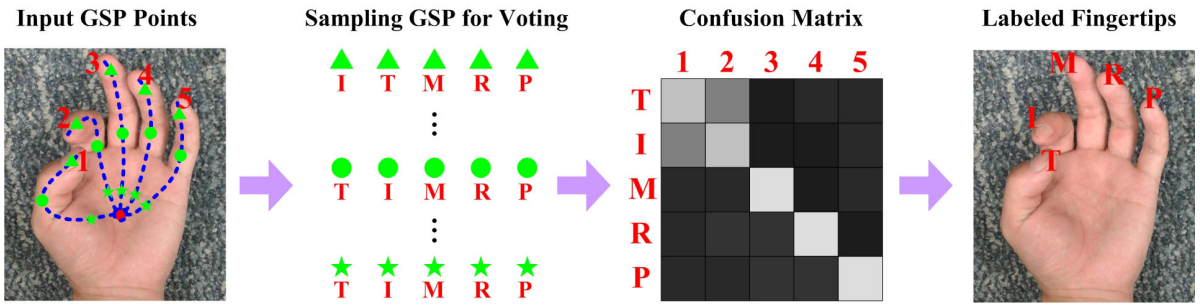


Figure 4.7: Fingertip classification via GSP-voting. First: the detected fingertips with their GSP trajectories. Second: GSP-voting with each sampling set of five points on the different fingers. Third: the accumulated confusion matrix Γ. Fourth: classified fingertip labels

after classification. We can see this labeling scheme is quite robust to finger articulation.

Algorithm 2: Fingertip classification via GSP-voting.

$\Gamma_{k,j} = 0$, $N^* = \max |U_k|$, $i = 0$;
while $i \leq N^*$ **do**
  Extract five points $p_{k,j}$, where $j = i \times |U_k|/N^*$;
  Sort the five points by arranging the five vectors $\Delta_k = p_{k,j} - p_p$ clockwisely;
  Let the order number of $p_{k,j}$ be $\bar{j}$, $\Gamma_{k,\bar{j}} = \Gamma_{i,\bar{j}} + 1$;
  $i = i + 1$;
end while
$l_k = \arg\max_j \Gamma_{k,j}$;

4.2.2 Fingertip Tracking

After the initial positions of the five fingertips are detected, we build a particle filter for each fingertip to track their positions through successive frames. Compared to the other hand parts, the positions of the fingertips are much more flexible and move very fast. In addition, they can be easily confused with other hand parts in challenging postures like closed hand and twisted fingers. In such postures, they are difficult to differentiate from the contacting hand parts in depth images with the heuristics used in the proposed method. Therefore, we assume that the frame rate of the sequences for fingertip tracking is high, *e.g.*, 30fps, and that the fingertip will not have long-term contact with other hand parts. Let t denote the frame number. Let (x, w) denote

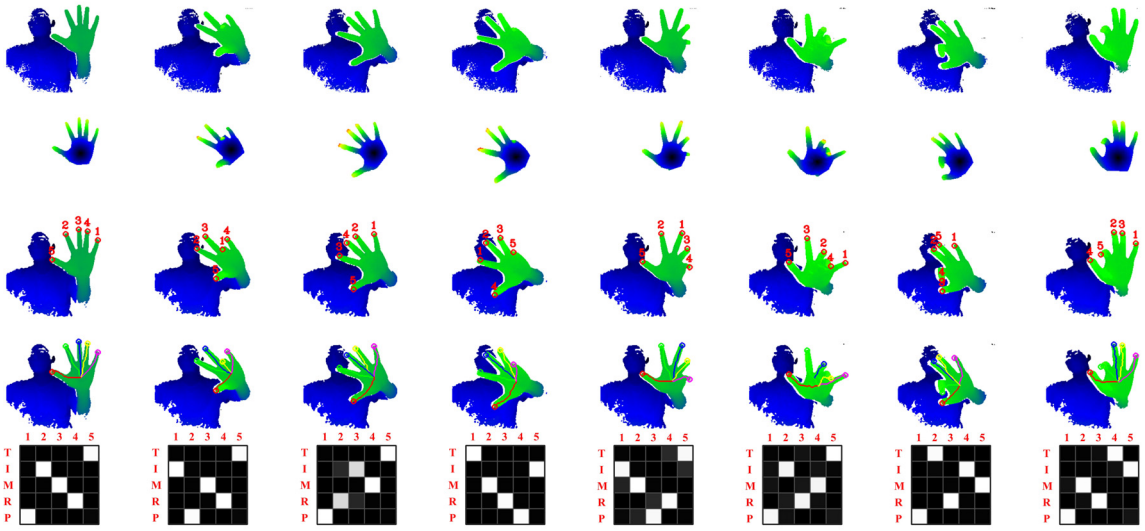


Figure 4.8: GSP-voting on real-world testing sequences. First row: input depth images. Second row: geodesic distance map of the hand region. Third row: the detected five fingertips. Fourth row: classified fingertips with GSP points. Fifth row: confusion matrices for GSP-voting, where the finger indices are consistent to that in the third row.

a particle, where the state hypothesis x is the 2D position of a possible fingertip and w is the particle weight. The basic idea is to constrain the positions of each particle to the border points of the depth image to reduce the search space, instead of choosing arbitrary positions within the 2D space. Therefore, the state transition function $P(x^t|x^{t-1})$ is actually deterministic shifting from the value x^{t-1} in the previous frame to the nearest border point x^t in the current frame. The set of border points A_b can be obtained via the same procedures in Section 4.2.1. Besides, let $P(y^t|x^t)$ be the likelihood function, where y^t represents the depth image observation and GDM calculated upon it. For a specific value of x^t, its likelihood $P(y^t|x^t)$ is evaluated based on its geodesic distance ω, geodesic shortest path U to the palm center and the local rectangle feature $\mathcal{R}$:

$$\begin{aligned} P(y^t|x^t) &= P(\omega, U, \mathcal{R}|x^t) \\ &= P(\omega|x^t)P(U|x^t)P(\mathcal{R}|x^t) \end{aligned}, \tag{4.3}$$

where we assume the observations ω, U and $\mathcal{R}$ are conditionally independent. In our implementation, the three likelihood terms in $P(y^t|x^t)$ all take the form of an exponential function of

certain distance metric. In $P(\omega|x^t)$, the distance metric is defined as the absolute difference between a pre-defined geodesic distance Ω and ω, where $\Omega \in \{\Omega_k\}_{k=1}^5$ are empirically determined values for each finger. Besides, we simply use the temporal references to estimate $P(U|x^t)$ and $P(\mathcal{R}|x^t)$. Let the geodesic shortest path and local rectangle feature associated with reference fingertip position be U_r and $\mathcal{R}_r$ respectively. $P(U|x^t)$ is thus defined based on the Hausdorff distance between U and U_r and $P(\mathcal{R}|x^t)$ is defined based on the feature distance between $\mathcal{R}$ and $\mathcal{R}_r$. Since the local rectangle feature $\mathcal{R}$ is a binarized pattern, we define its feature distance as the ratio of the number of points with the same values to the size of the rectangle. Overall, $P(y^t|x^t)$ takes the following form:

$$P(y^t|x^t) = \exp(-\lambda_g d_g - \lambda_u d_u - \lambda_r d_r) \tag{4.4}$$

where $d_g = |\Omega - \omega|$, d_u is the Hausdorff distance between U and U_r and d_r is the feature distance of $\mathcal{R}$. The constants λ_g, λ_u and λ_r represent the weights of the three terms to evaluate the likelihood and can be empirically determined. With the above formulations of the state transition function and likelihood function, the standard particle filter are used to track each of the five fingertips.

4.3 Hand Pose Estimation

Articulated hand pose consists of the global translation and rotation, and local finger motion. Let the parameter space of hand pose be described by a vector $\phi = (R_g, T_g, \phi_l)$, where $R_g = (\theta_x, \theta_y, \theta_z)$ and $T_g = (x_g, y_g, z_g)$ are the global rotation and translation and ϕ_l is the local finger articulation. We further decompose $\phi_l = \{\phi_{l,k}|k = 1, 2, ..., 5\}$, and each subset $\phi_{l,k}$ corresponds to the pose parameters of a single finger. In the proposed framework, we follow a divide-and-conquer approach for hand pose estimation in both the intra and joint modes. That is, R_g, T_g and ϕ_l are estimated separately, rather than being estimated by greedy search in the whole parameter space. This is because the dimensionality of the parameter space is large, and direct search in

this space is very time consuming and can easily be trapped into local optima. In addition, we take advantage of the motion constraints between the finger pose parameters to reduce the pose dimensionality, which include the static constraints to model the feasible range of the finger movement and the dynamic constraints to model the interdependency between the different finger parts. The details of these constraints can be referred to [75].

At each frame, we apply the algorithms in Section 4.1 and Section 4.2 to find the hand region and track the positions of the fingertips. With the extracted hand regions F_h in the depth image and the fingertip positions $\{v_k\}_{k=1}^5$, the goal of hand pose estimation is to find a pose ϕ^* that best explain F_h and $\{v_k\}_{k=1}^5$, which is decomposed into two sub-tasks:

1. Global pose estimation: find the optimal global pose R_g^* and T_g^* to fit the hand model to F_h or $\{v_k\}_{k=1}^5$, with the finger pose parameters fixed.

2. Local pose estimation: find $\phi_{l,k}$ for each finger to minimize the matching error between the visible vertices on the model and the corresponding points in F_h;

In the proposed framework, these two sub-tasks are fulfilled in different modes, *i.e.* intra or joint mode, depending on the availability of the temporal reference for hand pose estimation. The intra mode works in the initialization stage for hand pose tracking and the hand pose is inferred with single-frame observations. Based on the positions of the palm center and the middle fingertip and the orientation of the palm, the global hand pose is obtained by estimating the rigid transformation parameters, which is then used to align the model hand skeleton with the fingertips. The detected fingertips are then used as end-effectors to align the model fingertips via Inverse Kinematics. In the joint mode, the hand pose is first initialized as the reference pose from the previous frame and further refined by model-fitting with the current frame observations. In addition to the fingertip observations, the hand model is also fitted to the point cloud via articulated iterative closest point algorithm, which can handle more challenging cases such as occluded fingertips and clenched hand.

4.3.1 Intra Mode

The intra mode is used for pose initialization before the pose tracking stage begins, and the temporal reference pose is not available. For simplicity, we assume the hand faces the camera, and is in a stretched pose. While no reference local hand pose is available, we use an alternative approach to fulfill sub-task 1. The global translation T_g is obtained using palm tracking in Section 4.1. Besides, when the hand is stretched, the global rotation R_g can be approximately determined by two vectors $\{\mathbf{v}_n, \mathbf{v}_t\}$, where $\mathbf{v}_n$ is the normal vector of the hand palm and $\mathbf{v}_t$ is the vector from the palm center to the middle fingertip. We define $\mathbf{v}_n^0 = (0, 0, 1)$ and $\mathbf{v}_t^0 = (0, 1, 0)$ when $R_g = (0, 0, 0)$. To estimate $\mathbf{v}_n$, we perform PCA analysis to the 3D points within the palm circle. Based on the 3D shape of hand palm, $\mathbf{v}_n$ is approximated by the Eigenvector that corresponds to the smallest Eigenvalue of the covariance matrix of these 3D point coordinates. Let the rotation matrix corresponding to R_g be M_g. M_g can then be obtained by solving the following equation:

$$\begin{cases} M_g * \mathbf{v}_t^0 = \mathbf{v}_t \\ M_g * \mathbf{v}_n^0 = \mathbf{v}_n \end{cases} \tag{4.5}$$

The hand model is then aligned to F_h using the global pose estimation. To fulfill subtask 2, a kinematic chain is built for each finger based on the skeleton hierarchy in the hand model. By using the Type 2 constraints in Table4.1 the number of DOF for each kinematic chain is reduced to three. For each kinematic chain we minimize the difference between the model fingertip locations and detected fingertip locations:

$$\boldsymbol{\phi}_{l,k}^* = \underset{\boldsymbol{\phi}_{l,k}}{\operatorname{argmin}} \left\| \bar{\boldsymbol{v}}_k(R_g, T_g, \boldsymbol{\phi}_{l,k}) - \boldsymbol{v}_k \right\|^2, \tag{4.6}$$

where $\boldsymbol{\phi}_{l,k}$ is the set of joint angle parameters for the k_{th} finger, $\bar{\boldsymbol{v}}_k(R_g, T_g, \boldsymbol{\phi}_{l,k})$ is the k_{th} model fingertip location as a function of $\boldsymbol{\phi}_{l,k}$ and (R_g, T_g) and $\boldsymbol{v}_k$ the detected fingertip location. This minimization problem can be efficiently solved with the cyclic coordinate descent algorithm

[113]in our system.

4.3.2 Joint Mode

In the joint mode, a reference hand pose $\phi_0 = \{R_g^0, T_g^0, \phi_l^0\}$ is available from the previous frame. It can be combined with the information in the current frame for pose estimation. Besides, we observe that most parts of the hand do not go through severe deformation over successive frames when hand motion is captured at ordinary frame rate. Therefore, the global hand pose can be obtained by aligning a reference hand model to the input depth data. To this end, we use the iterative closest point approach [114] to fulfill the subtask 1. The mesh surface of the 3D hand model is used to fit to the point cloud observation Let $V_m(\phi)$ be the set of visible model vertices generated by driving the hand model with a hypothesis pose ϕ. In this task, ϕ_0 is first used to generate model vertices $V_m(\phi_0)$, which is then aligned to the input 3D point cloud V_h by iteratively adjusting the global pose parameters to minimize the following cost function:

$$f_{ICP}(R_g, T_g) = \frac{1}{N} \sum_{i=1}^{N} \left\| v_h^i - (M_g * v_m^i + T_g) \right\|^2, \tag{4.7}$$

where $v_h^i \in V_h$ and $v_m^i \in v_m$ are a pair of corresponding points in the sets of vertices V_h and V_m, M_g is the global rotation matrix corresponding to R_g and T_g is the global translation vector. Minimization of the equation gives the global motion estimation.

After the ICP rigid alignment, the finger movement parameters ϕ_l of the hand model still needs to be adjusted to fit the input point cloud V_h. To this end, we first use the extracted fingertips to give an initial estimation of the finger pose $\phi_{l,k}$ using inverse kinematics. Another articulated ICP algorithm is then applied to fit the visible mesh vertices of the hand model to the input point cloud to estimate the finger pose. The algorithm is given below:

Algorithm 3: Finger Pose Estimation.

1: Update the hand model to get visible vertex set $V_m = \{v_m^j(R_g, T_g, \phi_l)\}$;
2: Find the visible vertex set for each model finger k, U_k;
3: Build the correspondence: for each vertex in U_k, find the nearest neighbor in V_h;
4: Local optimization:
$\phi_{l,k} = \arg\min \sum_{j \in U_k} \left\| v_h^j - v_m^j(R_g, T_g, \phi_{l,k}) \right\|^2$;

4.4 Experiments

This section presents the experimental results of the proposed method on several real-world sequences and a synthesized dataset. For quantitative evaluations on real-world data we utilize the two testing sequences used in [100], the lengths of which are 702 and 894 respectively. In these sequences the whole upper body is visible, and thus we also test the performance of the proposed hand segmentation algorithm on them. However, the frame rate of these two sequences are relatively low and the fingertips constantly have long-term contact with other hand parts, *e.g.*, fist posture and twisted fingers. The proposed fingertip tracking algorithm is not designed to handle such sequences. Therefore, we only report the single-frame fingertip localization accuracy with Algorithm 1 proposed in 4.2.1 on these two real-world sequences. In addition, we also illustrate the qualitative results of hand pose estimation on a sequence captured by a Kinect sensor. To get an overall evaluation of both the fingertip tracking and hand pose estimation algorithms, we synthesize five sequences with the hand model in 3.1, which contain the ground truth data of both the fingertip locations and the hand pose parameters. We report both the fingertip localization accuracy and hand pose estimation results on this synthesis dataset.

4.4.1 Evaluation on Real-world Dataset

The two real-world testing sequences in [100] are performed by a user sitting in front of the camera performing various hand motions, and thus the whole upper body is visible in the depth

images. We first report the hand segmentation results with the algorithm presented in Section 4.1, in which we define a segmentation is successful if and only if the hand is segmented from the other body parts, *e.g.*, forearm and main body, and none of the hand parts such as palm and fingers are misclassified as the background. Overall, our method successfully segments the hand regions in all the 702 frames for the first sequence, as this sequence well meets the assumptions of the hand configuration for hand segmentation in Section 4.1, *e.g.*, both the hand and forearm are fully visible and the hand does not have large out-of-plane rotation. However, in the second sequence, the hand and forearm are often pointing towards the camera and thus the forearm becomes invisible in the depth images. In these frames, the orientation line of the forearm cannot be determined robustly and thus the hand fails to be accurately segmented from the forearm. Due to these issues, the hand segmentation fails in 49 out of the 894 frames in the second sequence, and thus the overall hand segmentation accuracy is 96.9%.

Fig. 4.9 and Fig. 4.10 illustrate the examples of successfully segmented hand regions in the first and second real-world sequences respectively. To better illustrate the effectiveness of the algorithm, we also overlay the detected palm circle and the orientation line of the forearm on the depth images, based on which the hand can be segmented from the forearm. The results show that the algorithm can detect the palm circle and orientation line quite accurately, even for the challenging frames in the second sequence in which the hand goes through large viewpoint and posture changes. In Fig. 4.11 we also show some examples in which the algorithm fail to crop the whole hand region. For most of these frames some fingers are misclassified as the background due to the wrong estimation of the forearm orientation. Note that in these frames the forearm is almost pointing perpendicularly to the image plane, and only very small portion of the forearm is visible in the depth images.

Since the sequences have low frame rate and include long-term twisting-finger and close-hand postures, we only report the single-frame fingertip localization accuracy with the algorithm in Section 4.2.1 on these two sequences. As the Algorithm 2 can classify the fingertips

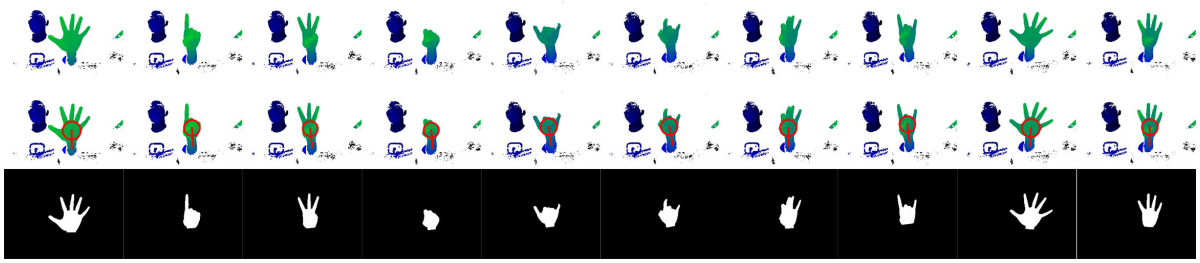


Figure 4.9: Hand segmentation results for the first real-world testing sequence in [100]. First row: input depth images. Second row: detected palm circle and forearm orientation line. Third row: cropped hand mask.

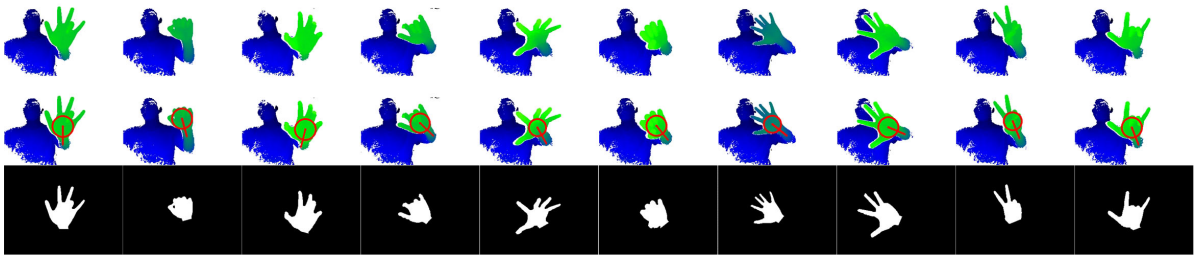


Figure 4.10: Hand segmentation results for the second real-world testing sequence in [100]. First row: input depth images. Second row: detected palm circle and forearm orientation line. Third row: cropped hand mask.

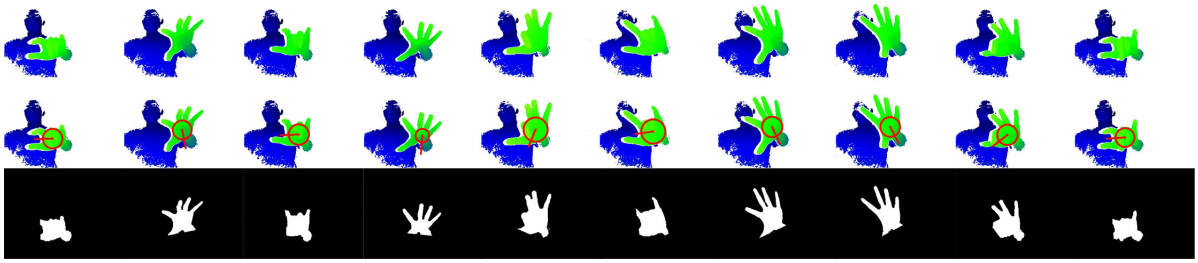


Figure 4.11: Inaccurately cropped hand masks. Note that the forearm orientation estimation is incorrect due to the large out-of-plane rotation of hand and forearm.

only when all five fingertips are detected using Algorithm 1, we define the evaluation metric as follows: if all five fingertips are detected via Algorithm 1, we then use Algorithm 2 to determine the fingertip label for each of them and the fingertip prediction error is defined as the 3D Euclidean distance between the predicted fingertips and the corresponding ground truth positions. If less than five fingertips are detected using Algorithm 1, we cannot determine the fingertip

labels from only one frame with Algorithm 2, and the fingertip prediction error is thus defined as the 3D Euclidean distance between the predicted fingertips and its nearest neighbors in the ground truth fingertips. Based on such definitions of error metric, we report the average fingertip localization errors for different numbers of fingertip detections separately to better illustrate the performance of the proposed method. In our implementation we use a fixed size of $(25, 25)$ for the local rectangle feature $\mathcal{R}$ as the distance of the user hand does not change drastically in the sequences, while it can also be set to be distance-adaptive in order that the method gets more robust to scale change of the hand. Besides, the parameters in Algorithm 1 are empirically set to be $N = 15$, $\omega_T = 6cm$ and $\eta_T = 0.5$.

Table 4.1: Single-frame fingertip localization accuracy on real-world sequences.

	Sequence 1			Sequence 2		
Finger Num.	Frames Num.	Error	Std. Dev.	Frames Num.	Error	Std. Dev.
0	40	N.A.	N.A.	2	N.A.	N.A.
1	40	2.17	1.13	8	2.77	1.34
2	149	1.67	0.63	50	3.24	2.52
3	132	1.61	0.51	100	2.09	1.20
4	71	1.67	0.56	224	1.89	0.85
5	270	1.86	0.48	510	2.07	1.22

The single-frame fingertip localization results are presented in Table 4.1 for the two real-world sequences separately, in which we report the number of frames for different number of fingertip detections, the average value and standard deviation of the localization errors in centimeter. Overall, all five fingertips are detected in 780 out of the total 1596 frames, and their average fingertip localization error is $1.91cm$. In 42 frames no fingertips are detected mainly due to that the hand is in a completely closed posture. In the rest 774 frames not all five fingertips are detected, and the average localization error to the nearest ground truth fingertips is $1.92cm$.

Fig. 4.12 and Fig. 4.13 illustrate the results of fingertip localization on the two real-world

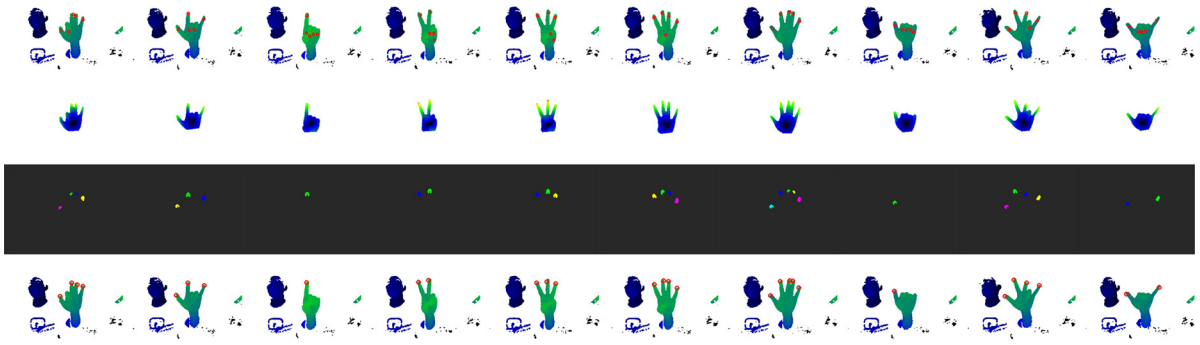


Figure 4.12: Fingertip localization results for the first real-world sequence in [100]. First row: ground truth fingertip annotations. Second row: geodesic distance maps calculated within the hand region. Third row: clusters of the candidate fingertip positions in Step 3 of Algorithm 1. Fourth row: detected fingertips.

sequences. It is worth noting that the above fingertip localization error is also partially due to the different definitions of fingertip locations in our method and the ground truth annotation in [100]. That is, in [100] the fingertips are annotated as the distal points of the finger skeleton, while they are defined as the distal points of the fingers on the skin surface in our method, as can be seen in the ground truth annotation in Fig. 4.12 and Fig. 4.13. As further evidence, most of the standard deviations of the fingertip localization errors in Table 4.1 are quite small. Besides, the average localization errors and standard deviations of the results in Sequence 2 are much higher than that in Sequence 1. The reason is that the hand is segmented inaccurately in quite a few frames of Sequence 2 because of the more complex hand and forearm configuration compared to Sequence 1, as discussed previously. In Fig. 4.14 we illustrate several examples in which either the fingertips are not completely located for all the stretched fingers or some fingertips are located at wrong positions, which shows the most of such failures result from inaccurate hand segmentation results.

The values of ω_T and η_T are important for the performance of Algorithm 1. A large value of ω_T and a small value of η_T indicate a high confidence of fingertip for a pixel, and thus will maintain fewer candidate pixels for fingertip detection. For a stretched hand posture, this can help the candidate pixels to concentrate on the fingertips. However, if very fewer candidate

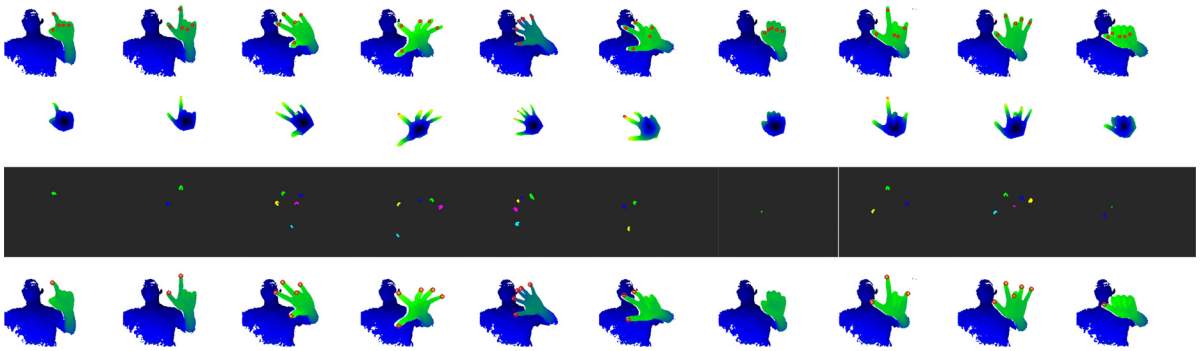


Figure 4.13: Fingertip localization results for the second real-world sequence in [100]. First row: ground truth fingertip annotations. Second row: geodesic distance maps calculated within the hand region. Third row: clusters of the candidate fingertip positions in Step 3 of Algorithm 1. Fourth row: detected fingertips.

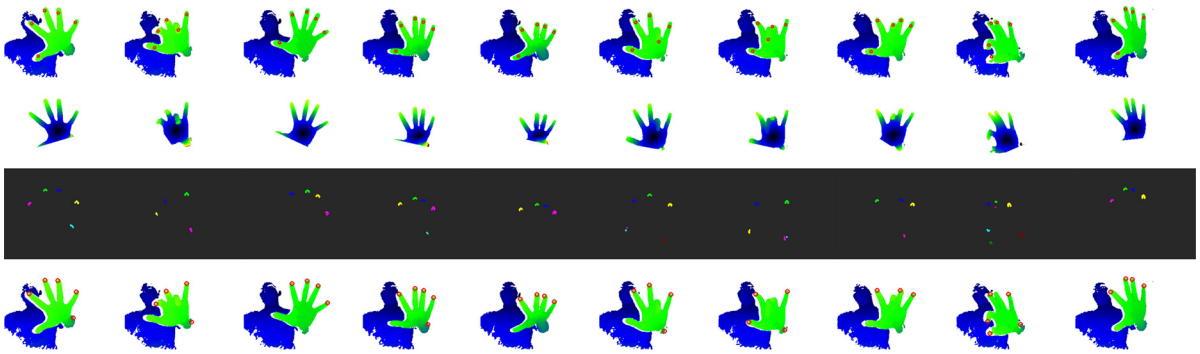
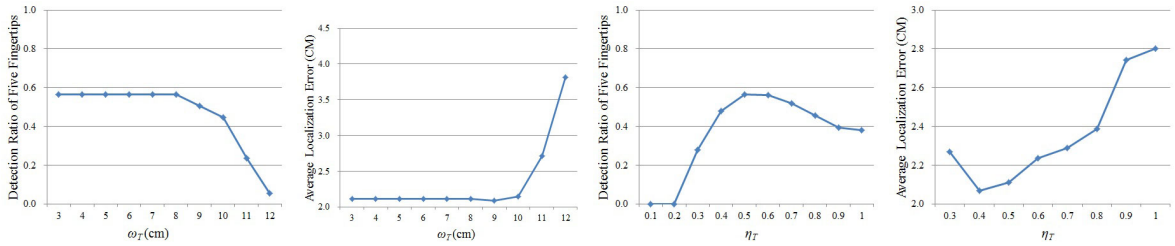


Figure 4.14: Inaccurately detected fingertips. Most of these detection failures result from inaccurate hand segmentation.

pixels are selected, they may also contain certain isolated outlier pixels resulting from noisy input or inaccurate hand segmentation. In this case, the outliers and the candidate pixels on the real fingertips can be difficult to differentiate. On the contrary, a small value of ω_T and a large value of η_T will retain many candidate pixels for fingertip detection, but many of them can lie far away from the fingertips. This can be detrimental for the subsequent clustering step in Algorithm 1, *e.g.*, the candidate pixels for different fingertips can be connected to each other. As the main purpose of Algorithm 1 is to detect the fingertips of stretched fingers when the hand is in a stretched posture, we investigate their impact on the fingertip localization accuracy by checking the ratio of frames in which all five fingertips are detected and the average localization



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

Figure 4.15: Parameter tests for fingertip localization accuracy. (a-b) The detection ratios of five fingertips and localization errors for different values of ω_T. (c-d) The detection ratios of five fingertips and localization errors for different values of η_T.

error when all five fingertips are detected. For this test we change the values of ω_T and η_T separately and leave the uninvestigated parameters unchanged. The results are presented in Fig. 4.15, which shows the optimal value of ω_T is less than $8cm$ and the optimal value of η_T is between 0.4 to 0.5. The reason that the performance becomes stable for $\omega_T \leq 8cm$ is because the local rectangle feature helps to filter out non-fingertip pixels. In contrast, when ω_T is set fixed to be $6cm$, the performance still changes drastically with different values of η_T, which suggests the local rectangle feature is more capable to differentiate fingertip/non-fingertip pixels.

We have also captured a depth image sequence using a Kinect sensor and test the complete fingertip tracking and hand pose estimation scheme presented in this Chapter on it for qualitative test. To track the fingertips, we generate a set of 50 particles for each fingertip during particle filtering. The sequence contains a variety of natural hand motion. Lacking ground truth data, we present comparisons between the color image and the projection of the 3D reconstructed hand model for subjective test, as shown in Fig. 4.16. The hand model is shown in a perspective view with the three arrows indicating the rotated coordinates to better demonstrate the pose estimation results.

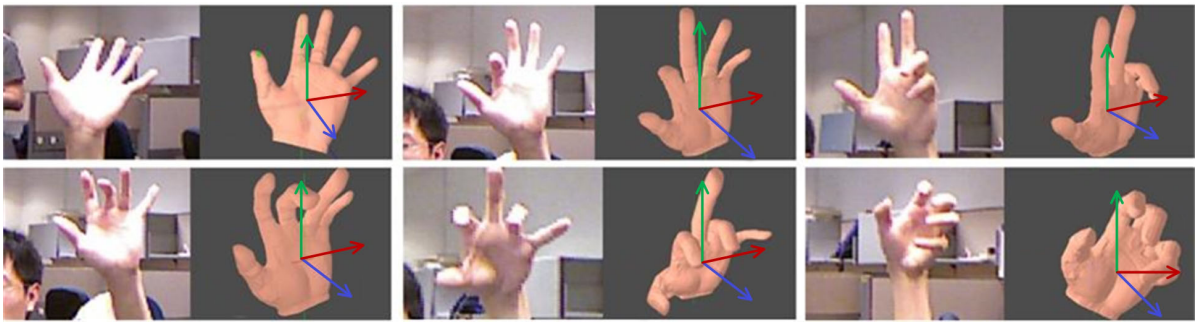


Figure 4.16: Pose estimation results on real-world input sequence.

4.4.2 Evaluation on Synthesis Dataset

The synthesis dataset consists of five sequences of continuous hand motion, and each frame is annotated with the ground truth data of both the fingertip locations and the hand pose parameters. They are generated with the 3D hand model in 3.1. In these sequences only the hand visible, and thus hand segmentation is not needed to crop the hand region. The five sequences include different types of hand motions, with Seq. 1 for grasping motion, Seq. 2 for adduction/abduction motion, Seq. 3 for flexion motion of index and middle fingers, Seq. 4 for global rotation and Seq. 5 for combination of grasping and global rotation motion. The first three sequences contain no global hand rotation and the last two sequences contain global hand rotation ranging from $(0°, 70°)$ around the X axis and $(0°, 70°)$ around the Z axis. These sequences all start from an initial pose in which the hand is full stretched and facing the camera in the first frame, and in the following frames the hand performs the specified motion continuously. The lengths of the sequences range from 25 to 60. For fingertip localization and tracking, the error metric is defined as the 3D Euclidean distance between the tracked fingertips and the corresponding ground truth positions. The ground truth fingertip positions are obtained by calculating the phalanx end point of each finger using forward kinematics on the hand model. The parameters are set to be the same as Section 4.4.1. Table 4.2 shows the average localization errors in centimeter on all five sequences. Note the localization error also partly results from the fact that the fingertips are detected on the skin surface rather than the hand skeleton.

Table 4.2: Average localization error of fingertip tracking on synthesis dataset in centimeters.

	Thumb	Index	Middle	Ring	Pinky
Seq. 1	2.51	1.53	1.50	1.27	0.77
Seq. 2	1.63	0.93	0.78	0.74	0.69
Seq. 3	2.11	1.15	1.16	0.84	0.81
Seq. 4	1.20	0.82	0.75	0.52	0.59
Seq. 5	1.44	0.93	0.89	0.77	0.86

Table 4.3: Quantitative test results on synthetic sequences.

	Local	Global-X	Global-Y	Global-Z
Seq.1	$7.75°$	$2.36°$	$1.48°$	$2.10°$
Seq.2	$3.61°$	$1.79°$	$3.31°$	$0.77°$
Seq.3	$5.59°$	$0.85°$	$1.46°$	$2.03°$
Seq.4	$4.65°$	$3.79°$	$6.44°$	$5.76°$
Seq.5	$7.69°$	$2.45°$	$1.85°$	$1.38°$

For hand pose estimation, we define the error metric of the finger articulation parameters as the mean absolute errors between the recovered local joint angles and ground truth data. For global rotation, we define the error metric as the absolute errors between ground truth data and recovered global rotation for all three dimensions. The errors are averaged over the entire sequence and the results are given in Table4.3. The per-frame prediction errors for local finger pose for the five sequences are presented in Fig. 4.17 and we can see the estimation errors are quite small and stable. The results validate the effectiveness of the proposed joint mode hand pose estimation scheme. However, in some sequences, the hand pose prediction error becomes a little larger as hand motion continues. In Seq. 3 the index and middle fingers are soon connected with each other in the depth images due to the finger motion, in which the fingertips may not be tracked robustly. Also, in Seq. 4 and Seq. 5 the fingertips become difficult to track as the out-of-plane hand rotation becomes large. Such problem mostly results from the fact that the proposed hand pose estimation algorithm relies largely on fingertip tracking. The algorithm takes around 200ms to process one frame for hand pose estimation.

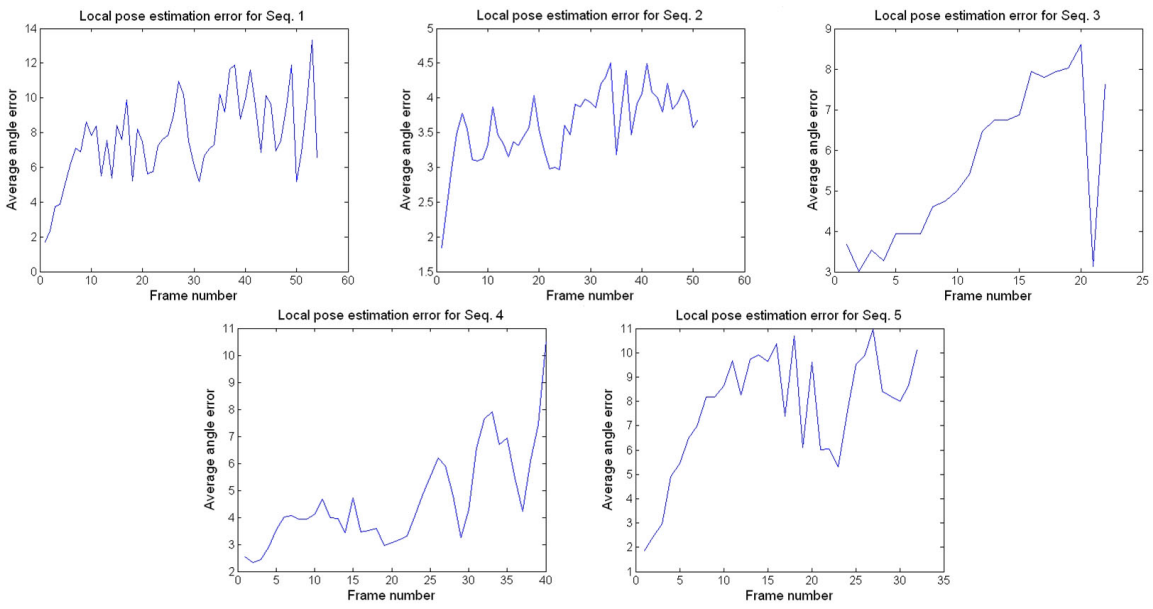


Figure 4.17: Local pose estimation error for the synthetic sequences.

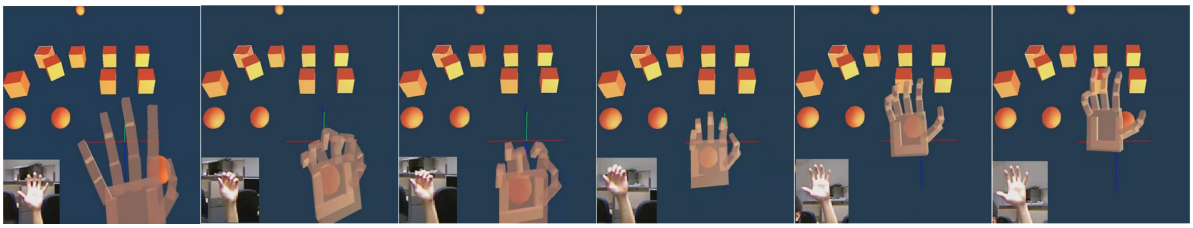


Figure 4.18: Virtual object manipulation via model-based hand pose estimation.

4.5 Applications

This section illustrates two virtual reality applications to demonstrate the potential of the proposed hand pose estimation scheme in real-world scenarios. The first demo is virtual object manipulation with the bare hand. We build a virtual environment using the Nvidia Physx SDK, which contains a 3D hand model and some virtual objects like boxes and spheres. Users can use their bare hands to perform some manipulative tasks such as moving, pushing and grasping. A sequence of snapshots for virtual object manipulation is shown in Fig. 4.18.

In the second application we directly utilize the tracked 3D fingertips to play a simulated water-oscillator instrument. We define five dynamic gestures to interact with the instrument

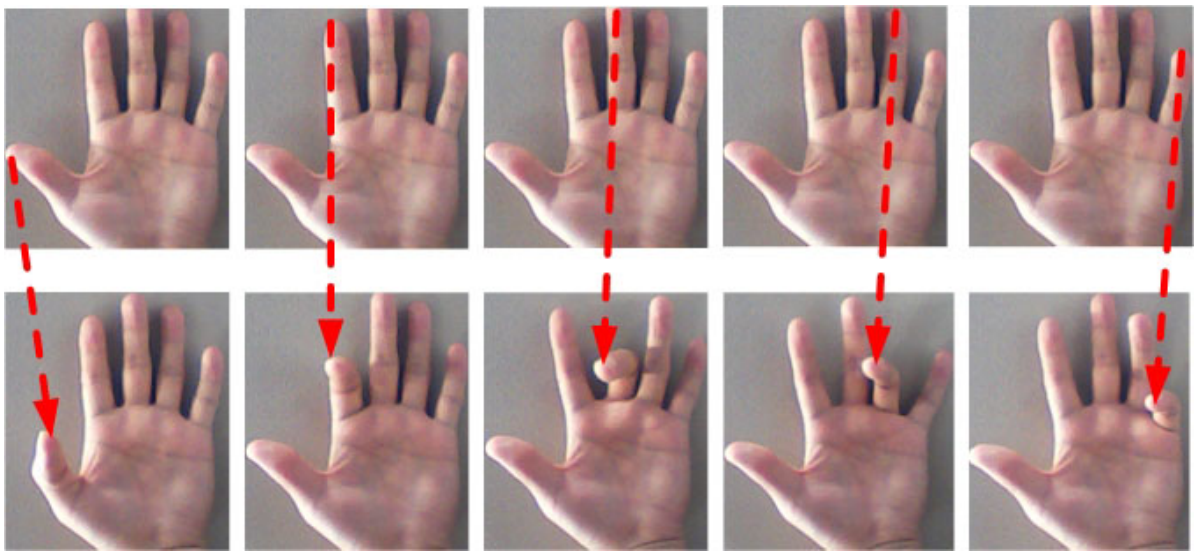


Figure 4.19: Dynamic gesture set to play a simulated water-oscillator instrument. Each dynamic gesture is defined by the 3D fingertip motion.

and each gesture corresponds to the motion of a single finger. To recognize the gesture, we constantly check whether the corresponding finger is performing a sudden flexion movement based on the fingertip position changes over certain time interval. The gesture set is shown in Fig. 4.19. When a dynamic gesture is recognized, the corresponding sound is played and simulated water wave is generated around the oscillator. Fig. 4.20 illustrates the user interface for this application.

4.6 Discussion

This chapter presents a divide-and-conquer scheme for model-based hand pose estimation with the input of depth image sequences. The tracked fingertip positions are used to provide initial pose estimate and the capability to recover from failure during pose tracking. The articulated iterative closest point method is adopted to obtain refined estimation via model-fitting. Experiments on both the synthetic data and real-world input sequence validate the effectiveness of the proposed method. However, one main drawback of this framework is that it heavily relies on

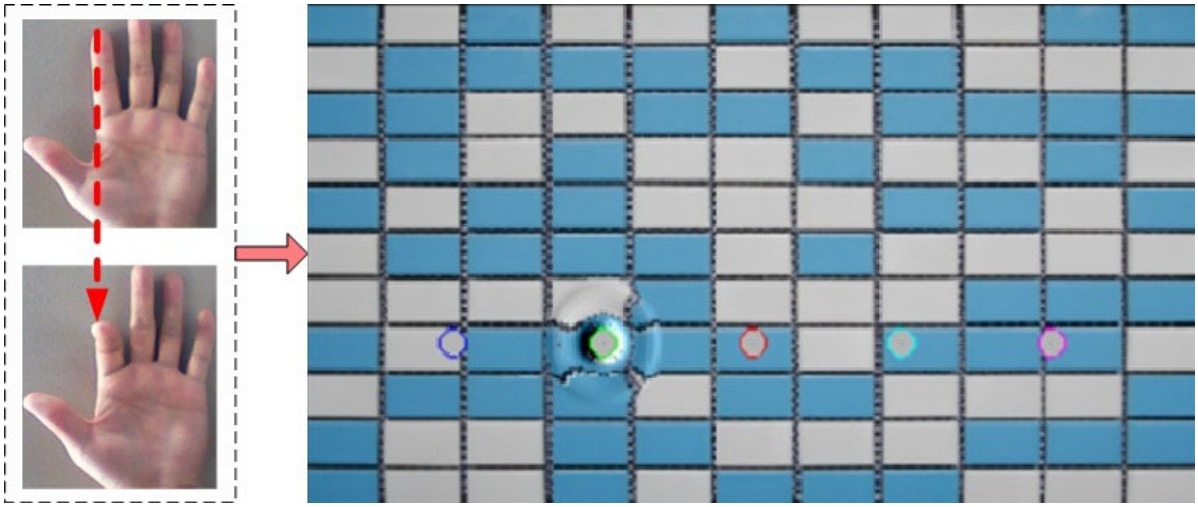


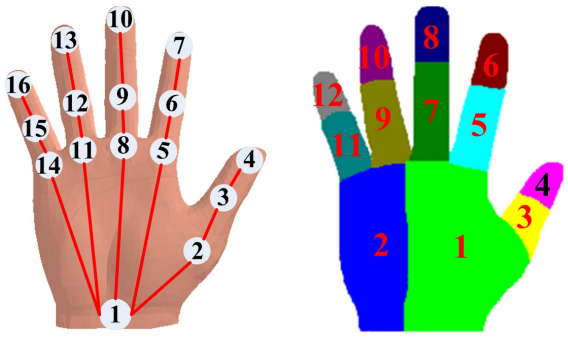
Figure 4.20: Playing a simulated water-oscillator instrument via fingertip tracking.

fingertip detection and tracking to estimate the pose, which can be vulnerable to complex hand configurations and self-occlusion. In case that the fingertips have long-term contact with other fingers or hand parts, the assumptions for fingertip detection and tracking may be no longer valid, *e.g.*, fingertips locate at pixels where the depth is discontinuous and have large geodesic distances. Moreover, the heuristics used to determine the fingertip positions are still relatively simple, *e.g.*, threshold with the occupancy ratio of the local rectangle feature and geodesic distance. To this end, we further develop a discriminative framework in Chapter 5 for hand pose estimation, in which the regression forest is adopted to learn the mapping from depth image features to the articulated hand pose in a full data-driven manner. This method can work much more robustly for arbitrary hand motion.

Chapter 5

Discriminative Hand Pose Estimation

In Chapter 4 we present algorithms for 3D fingertip detection and tracking and utilize them to build a model-based hand pose estimation framework. However, the heuristics used in the algorithms to locate the fingertip positions are mostly empirically determined and are still not robust enough to handle complex hand configurations like fist posture. Also, in Chapter 3 we present a hand parsing scheme to classify the hand region into a set of finger and palm parts, but it requires further efforts to take advantage of them to get the full-DOF hand pose. Therefore, in this chapter we present a discriminative framework to predict the 3D hand joint positions from the depth images, and the focus is to address the inconsistency between the real-world inputs and the synthesized training datasets by enforcing the hand part correlations [42]. The sixteen joint positions are shown in Fig. 5.1(a), which consist of the wrist center (no. 1), the five fingertips (no. 4, 7, 10, 13 and 16), the inter-phalangeal and metacapophalangeal joints of the thumb (no. 2 and 3), and the proximal inter-phalangeal (no. 6, 9, 12 and 15) and metacapophalangeal (no. 5, 8, 11 and 14) joints of all the other four fingers. The basic idea is to fuse a set of weak pose estimators for robust prediction, which is motivated by the human pose regression framework in [41] in which the regression forest is used for per-pixel joint prediction. Despite its high accuracy and capability to handle partial occlusion, it relies on the independent per-pixel votes and the correlations among the hand parts are not exploited. Given the inconsistency between



(a) (b)

Figure 5.1: (a) The positions of the sixteen hand joints, denoted as the circles. (b) The hand partition scheme for hand parsing.

the real-world inputs and the synthesized training data, many pixels will be misclassified and the vote distribution can be multi-modal. Besides, the different joint positions are estimated independently of each other by aggregating the votes from all the pixels via the Mean-Shift algorithm in [41]. However, such independent estimation scheme can easily lead to infeasible poses without exploiting the joint correlations, considering the severe self-occlusion of the hand and large viewpoint variations compared to the body.

To this end, the proposed method aims to enforce the hand part correlations to resolve the ambiguous per-pixel predictions. Similar to [41], the regression forest is utilized for per-pixel voting for the individual joints separately. However, we enforce the hand part correlations to improve the per-pixel regression results by (1) improving the discriminative power of the regression forest and (2) fusing the per-pixel votes for all the joints simultaneously. First, the semantic hand parts, *e.g.*, the hand partition in Fig. 5.1(b), are highly correlated with the joint positions, and the co-occurrence pattern between the hand parts provides discriminative information to refine the per-pixel votes. By utilizing the semantic hand parts as the additional cue to the raw depth image, we design a more effective regression forest to predict the joint locations than that in [41], without increasing the computational cost significantly. Second, as the hand motion is highly constrained, the parameters of the sixteen joints are essentially embedded in a much lower dimensional space. To handle the multi-modal per-pixel predictions obtained by the

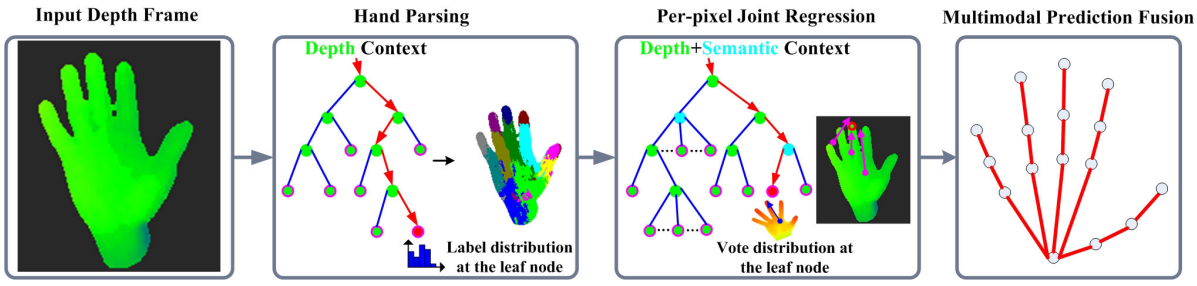


Figure 5.2: The pipeline of the proposed hand pose estimation scheme.

regression forest, the pose can be refined by maximizing the joint posterior of the pose parameters in the low dimensional space instead of the original space. Here the posterior distribution is modeled as a weighted Products of Experts [86] based on the per-pixel joint predictions. The low dimensional space of the joint parameters is learned by PCA analysis of the training samples. Based on this formulation we show that the posterior distribution can be efficiently maximized by the Expectation-Maximization (EM) algorithm [87].

The pipeline to process one frame during the testing stage is shown in Fig. 5.2. The depth image is first parsed by per-pixel classification using a pre-trained RDF classifier to obtain the hand parts. The parsing results are then combined with the depth image to be encoded into the depth and semantic context descriptors. Each pixel casts its votes for the joint locations with the Regression Forest, which is trained using both descriptors. To illustrate this we show some exemplary per-pixel votes for the middle fingertip in Fig. 5.2. The final joint locations are obtained by fusing the multimodal per-pixel predictions. By utilizing the hand part correlations in this way, the prediction accuracy is substantially improved compared to [41] and [85] on both a synthesized dataset and a real-world dataset. Especially, on the challenging real-world dataset which consists of four different subjects, the proposed method improves the prediction accuracy by 14.44% compared to the baseline method [41], and many of the results are visually very close to the ground truth joint positions.

5.1 The Overall Framework

Let the sixteen joint positions be $\mathbf{\Phi} = \{\phi_k\}_{k=1}^{K}$ and $K = 16$. The regression forest determines a set of maximum J relative votes $\{\mathbf{\Delta}_{ijk}, w_{ijk}\}_{j=1}^{J}$ for each ϕ_k for the pixel i during testing, where $\mathbf{\Delta}_{ijk}$ is the 3D relative offset between the 3D position of the pixel and the joint; w_{ijk} is the weight of the vote. Given the 3D position $\boldsymbol{v}_i$ of the pixel i, the relative votes can be converted to the absolute votes $\{\boldsymbol{v}_{ijk}, w_{ijk}\}_{j=1}^{J}$, where $\boldsymbol{v}_{ijk} = \mathbf{\Delta}_{ijk} + \boldsymbol{v}_i$. Let the depth image and label image be I_D and I_L respectively, and the depth context and semantic context descriptors for the pixel be $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{S}}$ respectively. Let the low dimension joint space be Ω. We present the proposed pose estimation method in Algorithm 4, with the details provided in the following sections. Overall, the goal to predict the joint locations can be achieved by two sub-tasks, each of which models different aspects of the hand part correlations:

Per-pixel Joint Regression: for each pixel i, retrieve the absolute votes $\{\boldsymbol{v}_{ijk}, w_{ijk}\}_{j=1}^{J}$ reached by $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{S}}$ in the regression forest. This sub-task corresponds to Step 1-7 in Algorithm 4.

Multimodal Prediction Fusion: aggregate the individual votes from all the candidate pixels to produce the posterior distribution $P(\mathbf{\Phi}|I_D)$. Maximize $P(\mathbf{\Phi}|I_D)$ with the constraints $\mathbf{\Phi} \in \Omega$ to find the optimal joint locations. This sub-task corresponds to Step 8-9 in Algorithm 4.

5.2 Per-Pixel Joint Regression

We utilize the regression forest [41] to obtain the joint location predictions for each pixel in the depth images. The regression forest is an ensemble of multiple random regression trees, each of which consists of a number of split nodes and leaf nodes. Each split node contains one split function learned from the training data to branch to the child node based on the feature values of the descriptor of an input pixel i. Each leaf node contains the distributions over the 3D relative

Algorithm 4: Hand Joint Position Prediction.

1: Parsing the depth image I_D to get the label image I_L;
2: **for all** pixel i in the input depth image I_D **do**
3: Retrieve the 3D position $\boldsymbol{v}_i$ of the pixel;
4: Retrieve the relative votes $\{\boldsymbol{\Delta}_{ijk}, w_{ijk}\}_{j=1}^{J}$ from the regression forest based on both $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{S}}$;
5: Calculate the absolute votes $\{\boldsymbol{v}_{ijk}, w_{ijk}\}_{j=1}^{J}$ by setting $\boldsymbol{v}_{ijk} = \boldsymbol{\Delta}_{ijk} + \boldsymbol{v}_i$;
6: **end for**
7: Eliminated unreliable long range predictions by setting the vote weights to zero for those votes which satisfy $\|\boldsymbol{\Delta}_{ijk}\| > \lambda_k$;
8: Down-sample the pixels and fuse their votes to obtain the joint posterior distribution $P(\boldsymbol{\Phi}|I_D)$;
9: Estimate the optimal joint positions $\boldsymbol{\Phi}^*$ by maximizing $P(\boldsymbol{\Phi}|I_D)$ subject to the hand motion constraints Ω;

offsets to the joint positions, which are collected from the training samples.

Different from [41] which directly regresses for the relative votes from the depth image, we propose to use the hand parsing results as the supplemental cue to the depth image for regression. Fig. 5.1(b) shows our hand label partition scheme for hand parsing, and the whole hand is divided into twelve non-overlapping parts. Such hand part partition has encoded a rough hand pose configuration and thus is possible to assist hand pose estimation. Similar idea proves effective in [88], in which the input image is first parsed to get the individual body part potentials at each pixel and the potentials are then used as extra multi-channel feature for pose regression. That is, a pre-trained classifier for body part classification is used to get a potential image for each body part via per-pixel classification of the image, in which the value of each pixel indicate the confidence that the pixel belongs to this body part. In our problem of hand pose estimation, this will produce a set of 12 potential images for the hand part partition in Fig. 5.1 (b). However, this is quite memory-inefficient for training compared to only 1-channel depth image in [41], especially considering the large amounts of training samples involved.

We therefore use a 1-channel discrete hand part labels I_L instead, which is obtained by assigning a label $l \in L$ to each pixel in the depth image of the hand region. Note the hand

partition scheme in Fig. 5.1(b) is the same to that in Fig. 3.4 in Section 3.2 for hand part parsing. However, since the SMRF framework for hand parsing in Chapter 3 is a little time-consuming for real-time hand pose estimation, we take advantage of only the per-pixel hand parsing scheme in Section 3.2 in this chapter, which if fulfilled via the RDF classifier [67] and the depth-context feature [73]. Such a choice of hand parsing won't incur too much extra time costs, as demonstrated in the experiments.

We illustrate this idea in Fig. 5.3. During training, the depth images are used to train a hand parser to classify the testing depth image into non-overlapping hand parts. By running the trained hand parser on the training depth images, their label images can be obtained. The depth and parsed label images are then combined to train the regressor. By training the regression forest with both the depth and semantic contexts, the training samples that reach each leaf node will be similar in terms of both the depth values and hand part co-occurrence pattern in the neighborhood, which is helpful for the leaf nodes to generate more consistent predictions of the joint locations. In the testing phase, the raw input depth image is first processed to generate I_L. Then both the depth image I_D and label image I_L are concatenated as the input of the regression forest to predict the joint positions.

5.2.1 Prediction with Regression Forest

The regression forest is used to predict a set of up to J votes $\{\boldsymbol{v}_{ijk}, w_{ijk}\}_{j=1}^{J}$ for each input pixel i and joint ϕ_k given the depth context and semantic context descriptors $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{S}}$. During the training phase, the depth images in the training dataset are first parsed to get the label images by a learned RDF. We then concatenate the depth context and semantic context to generate the new samples to train the regression forest for prediction. As in Section 3.2.1, the depth context descriptor $\boldsymbol{\mathcal{D}}$ for a pixel i is defined as the depth differences between i and a set of its

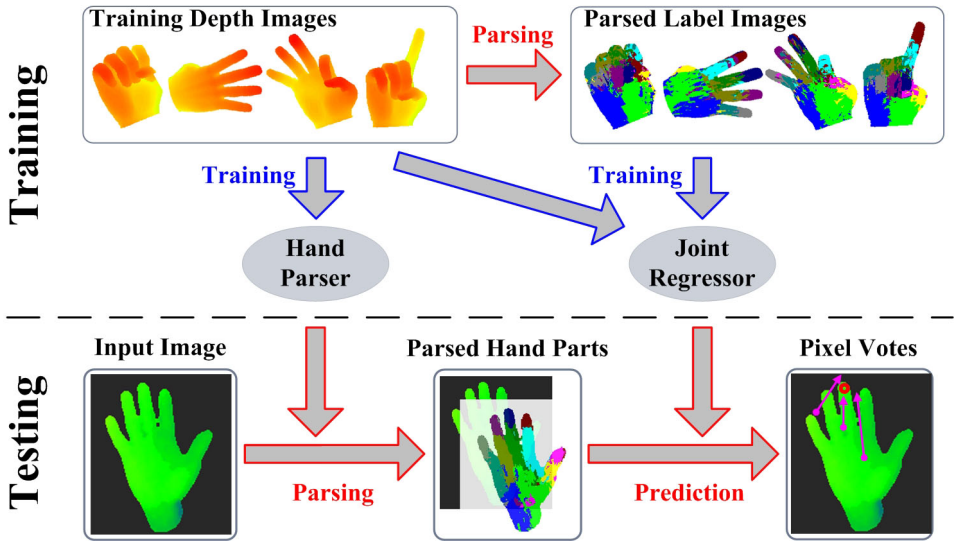


Figure 5.3: The training and testing phases for per-pixel joint regression with the depth and semantic contexts. The exemplary votes for the middle fingertip are illustrated.

neighboring points, which takes the following equivalent form:

$$\mathcal{D} = I_D\left(\boldsymbol{p} + \frac{\boldsymbol{u}}{I_D(\boldsymbol{p})}\right) - I_D(\boldsymbol{p}), \tag{5.1}$$

where $\mathcal{D}$ is one dimension of $\boldsymbol{\mathcal{D}}$; $\boldsymbol{p}$ is the pixel coordinate of the pixel i; $\boldsymbol{u}$ is the offset of a neighboring point. Usually the depth context $\boldsymbol{\mathcal{D}}$ consists of hundreds of dimensions, each of which uses a different offset $\boldsymbol{u}$ to estimate $\mathcal{D}$. At the same time, the semantic context $\boldsymbol{\mathcal{S}}$ is defined with the hand part labels of a set of offsets $\boldsymbol{u}_l$ which are similar to those used in calculating $\boldsymbol{\mathcal{D}}$. The descriptor $\boldsymbol{\mathcal{S}}$ is obtained by:

$$\mathcal{S} = I_L\left(\boldsymbol{p} + \frac{\boldsymbol{u}_l}{I_D(\boldsymbol{p})}\right), \tag{5.2}$$

where $\mathcal{S}$ is one dimension of $\boldsymbol{\mathcal{S}}$. To train the regression forest, each sample pixel i is also associated with the ground truth of the hand part label l_i and the offsets between its 3D position and the sixteen joints, *i.e.* $\boldsymbol{\Delta}_{ik}$, $k = 1, ..., K$.

Let the regression forest consist of T_r random regression trees, each of which contains a set of split and leaf nodes. As the feature values may be either continuous or discrete, *i.e.* $\mathcal{D}$ and $\mathcal{S}$,

and the split functions are tested on a single dimension of the feature, the number of the child nodes and the split criteria are thus different for the split nodes. Below we explicitly write out the indices of the feature dimensions for clarity. For the split nodes that contain the continuous dimension $\mathcal{D}^m$, they have two child nodes and the split function takes the following form:

$$\mathcal{D}^m \leq \tau, \tag{5.3}$$

where $\mathcal{D}^m$ is the m^{th} dimension of $\boldsymbol{\mathcal{D}}$, and τ is a threshold to determine the branch to one of the child nodes, *i.e.* the left child for $\mathcal{D}^m \leq \tau$ and right child otherwise. For the split nodes that contain the discrete dimension $\mathcal{S}^n$, they have a maximum of twelve child nodes, each of which corresponds to one hand part label. The split function of the node selects the branch to the child node by checking the label value of $\mathcal{S}^n$.

To learn the tree structures of the regression forest, a set of candidate split functions $\{\psi\} = \{\psi_D, \psi_S\}$ are first generated as the proposals. $\{\psi_D\}$ are associated with the continuous dimensions and generated by sampling m and τ. $\{\psi_S\}$ are associated with the discrete dimensions and generated simply by sampling n. Similar to [41], we choose the Shannon Entropy Gain for the hand part labels to select the split functions, which proves to be more effective than the other criteria such as the variances of the offsets. The Shannon Entropy Gain for ψ_D is calculated by:

$$G(\psi_D) = H(A) - \sum_{b\in\{l,r\}} \frac{|A_b(\psi_D)|}{|A|} H(A_b(\psi_D)), \tag{5.4}$$

where H is the entropy of the hand part label distributions in the sample set A that reaches the split node, and A_l and A_r are the two subsets of A split by the function ψ_D. The Shannon Entropy Gain for the split function ψ_S is calculated by:

$$G(\psi_S) = H(A) - \sum_{b=1}^{12} \frac{|A_b(\psi_S)|}{|A|} H(A_b(\psi_S)), \tag{5.5}$$

where A_b, $b = 1, ..., 12$ are the twelve subsets of A, each of which contains the samples that have the hand part label b. Thus, the optimal split function is selected so that $\psi^* = \arg\max_\psi G(\psi)$

at each split node during training. With this criterion, the tree structure of the forest is learned with the procedure similar to that in [66].

The regression models for the relative votes at the leaf nodes are learned from the set of relative offsets $\{\boldsymbol{\Delta}_{ik}\}$ associated with the training samples reaching them. At each leaf node, we define the regression model as a single relative vote $(\boldsymbol{\Delta}_k, w_k)$ for each joint, where $\boldsymbol{\Delta}_k$ represents the possible prediction of the relative offset based on the relative offsets $\{\boldsymbol{\Delta}_{ik}\}$ from the training samples, and w_k represents the confidence of the prediction. As in [41], we adopt the mean-shift algorithm [89] to obtain the modes of the relative offset using the following density estimator:

$$g_k(\boldsymbol{\Delta}) = \sum_{i=1}^{n_L} \exp\left(-\left\|\frac{\boldsymbol{\Delta} - \boldsymbol{\Delta}_{ik}}{b_k}\right\|^2\right), \tag{5.6}$$

where n_L is the number of the training samples reaching the leaf node; b_k is the bandwidth. Besides, the weight w_k is estimated for each mode at the leaf node to reflect their significance in prediction. Following [41], it is defined as the sum of the depth-adjusted weights of the samples that reach each mode.

In the testing phase, an input pixel i will recursively branch down the tree and reach one leaf node in each regression tree in the forest based on the descriptors $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{S}}$. In total the pixel reaches T_r leaf nodes in the regression forest and thus retrieves at most $J = T_r$ votes from the forest for each joint, *i.e.* $\{\boldsymbol{\Delta}_{ijk}, w_{ijk}\}_{j=1}^{J}$. In addition, as shown in [41], the long range predictions are usually unreliable and could be eliminated to improve the prediction accuracy by threshold of $\boldsymbol{\Delta}_{ijk}$ with a constant λ_k. Therefore, the votes that do not satisfy $\|\boldsymbol{\Delta}_{ijk}\| \leq \lambda_k$ are taken away from the vote set of each pixel by setting their corresponding weights to zero. The threshold λ_k takes different value for each joint and can be learned during training. With the 3D position $\boldsymbol{v}_i$ of the pixel i, we can finally obtain the absolute votes $\{\boldsymbol{v}_{ijk}, w_{ijk}\}_{j=1}^{J}$ with the regression forest, where $\boldsymbol{v}_{ijk} = \boldsymbol{\Delta}_{ijk} + \boldsymbol{v}_i$ is the absolute vote for the k^{th} joint.

Since we also learn classification when building the regression forest during the training

stage, the regression forest can thus be used for hand parsing again with the input of the depth image and the parsing results from the RDF classifier. Therefore, our proposed two-layered forest can be further extended to N_L layers. During testing, the parsing results from the n layer are reutilized with the depth image as the input for the $n+1$ layer, and finally we regress for the joint positions at the N_L layer with the depth and parsing results from $N_L - 1$ layer. We have tested the performance of such extended multi-layered forest in the experiments.

5.2.2 Discussion of the Impact of the Semantic Context

To make the regression forest accurate in estimating the joint locations, we expect the samples reaching the same leaf node to give consistent predictions. However, as only a subset of the dimensions of $\boldsymbol{\mathcal{D}}$ is tested for a testing sample going from the root to the leaf, the chances that the sample pixels at completely different positions of the hand can reach the same leaf node is still high during the testing stages. In addition, as the multiple regression trees of the forest are trained independently and the subsets of the dimensions of $\boldsymbol{\mathcal{D}}$ tested can thus be quite different for the same input sample going through different trees. As a result, the misclassified pixels give considerable false responses during testing, and the votes retrieved for the same pixel from the multiple trees are also multimodal. The problem becomes more prominent especially for the real-world test inputs that are inconsistent with the synthesized training data.

In contrast, the samples reaching one leaf node in the proposed regression forest will be similar in terms of the depth context as $\boldsymbol{\mathcal{D}}$ is utilized. Moreover, $\boldsymbol{\mathcal{S}}$ has encoded the semantic context of the pixel, *i.e.*, the co-occurrence pattern of the different hand parts. That is, some hand parts are more likely to be neighbors than others [73], *e.g.*, the hand part 8 and 7 stay together more often than 8 and 3 in Fig. 5.1(b). By utilizing $\boldsymbol{\mathcal{S}}$ to build the regression forest, we enforce that the samples reaching the same leaf node will share similar semantic contexts in their neighborhood, and thus the predictions could be more consistent. In Fig. 5.4 we show several examples to compare the distributions of the prediction confidence for the middle fingertip

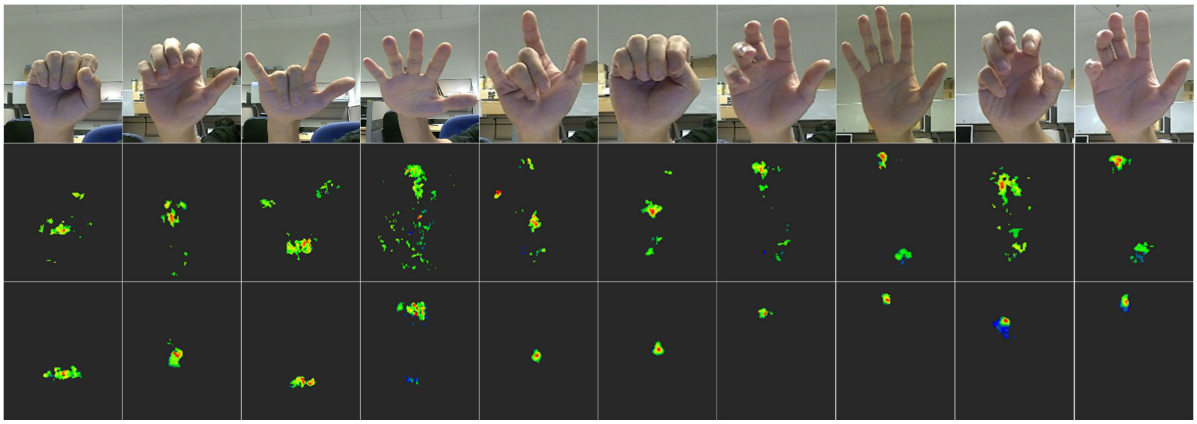


Figure 5.4: Comparison of the vote distributions for the middle fingertip, in which the long range votes have been eliminated. **Middle Row:** The remaining votes obtained with [41]; **Lower Row:** the votes obtained with the proposed regression forest.

obtained with [41] and the proposed regression forest, which are obtained by projecting the 3D votes of each pixel to the 2D image plane and accumulating the votes for all the pixels. Note here the long range predictions have been eliminated for both methods as in Section 5.2.1, and we can see the proposed regression forest produces much more compact prediction votes. In Section 5.4 the experimental results also demonstrate that the prediction accuracy is improved considerably by incorporating the semantic context.

5.3 Multimodal Prediction Fusion

By per-pixel prediction with the regression forest in Section 5.2, each pixel i casts maximum J votes for the individual joints independently, *i.e.* $\{\boldsymbol{v}_{ijk}, w_{ijk}\}_{j=1}^{J}$. As we have seen in Fig. 5.4, the per-pixel predictions can form multi-modal votes for each joint even when the unreliable long range predictions have been eliminated. It is difficult to determine which mode corresponds to the real joint position if we perform mode-seeking for each joint separately as in [41]. However, as the hand motion is constrained, the 3D positions of the multiple joints are highly correlated. Therefore, a large portion of the false mode combinations of different joints can be easily eliminated if we seek the modes of the per-pixel predictions for multiple joints simul-

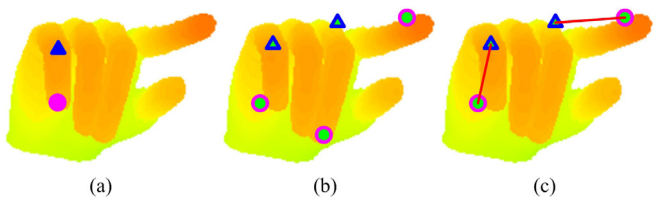


Figure 5.5: Illustration of multimodal prediction fusion using the joint position constraints. (a) The two joints to be predicted. (b) The multiple modes obtained by per-pixel regression. (c) The valid mode combination subject to the constraints.

taneously subject to specific learned hand configuration constraints. In Fig. 5.5 we illustrate this idea with a simple example in which we only constrain the relative positions between the proximal inter-phalangeal joint and the fingertip of the pinky. Assume two modes are found for the fingertip and three modes are found for the proximal inter-phalangeal joint according to per-pixel regression, which are denoted by the triangles and circles separately. In total there are six combinations of the modes for the two joints. Since the maximum distance between the two joints are limited, the number of candidate valid combinations is reduced from six to only two, as shown in Fig. 5.5 (c).

Based on the above observations, we propose to fuse the multimodal per-pixel votes through the maximization of the joint posterior distribution of all the joints in a learned low dimensional joint parameter space, which can be efficiently solved via an Expectation-Maximization (EM) framework. Due to the vast number of the pixels, we first randomly sample N candidate pixels from the input depth image for further prediction fusion to strike a balance between computational cost and prediction accuracy. For each candidate pixel i, its retrieved votes $\{\boldsymbol{v}_{ijk}, w_{ijk}\}_{j=1}^{J}$ for the k^{th} joint can be taken as a multimodal distribution $P(\phi_k|\boldsymbol{p}_i)$. We approximate this distribution with the Gaussian Mixture Model:

$$P(\phi_k|\boldsymbol{p}_i) = \sum_{j=1}^{J} \rho_{ijk} \exp\left(-\frac{\|\phi_k - \boldsymbol{v}_{ijk}\|^2}{\delta^2}\right), \tag{5.7}$$

where $\rho_{ijk} = w_{ijk} / \sum_j w_{ijk}$ is the weight of each mode. For simplicity we assume the same bandwith δ for the J modes. Following the weighted Products of Experts model [86], the joint

posterior distribution of the entire joint set given the depth image observation can be formulated as the weighted product of the individual predictions from all the candidate pixels:

$$\begin{aligned} P(\mathbf{\Phi}|I_D) &\propto \prod_i P(\mathbf{\Phi}|\boldsymbol{p}_i) = \prod_i \prod_k P(\phi_k|\boldsymbol{p}_i)^{w_{ik}} \\ &= \prod_i \prod_k \left[\sum_j \rho_{ijk} \exp\left(-\frac{\left\| \phi_k - \boldsymbol{v}_{ijk} \right\|^2}{\delta^2} \right) \right]^{w_{ik}}, \end{aligned} \tag{5.8}$$

where $w_{ik} = \sum_j w_{ijk} / \sum_{i,j} w_{ijk}$ is the normalized weight to indicate the total contribution of pixel i to ϕ_k, and $\sum_i w_{ik} = 1$.

The optimal joint locations can be obtained by maximizing $\log P(\mathbf{\Phi}|I_D)$ with respect to $\mathbf{\Phi}$, which is difficult to solve directly as the $\log \sum$ term in $\log P(\mathbf{\Phi}|I_D)$ cannot be further simplified. To this end, we further assume that the real joint location ϕ_k could be consistent with at most one mode among the maximum J votes from each pixel i. Therefore, ρ_{ijk} should be adjusted so that the inconsistent modes could be filtered out before estimating ϕ_k. To this end, we use an EM algorithm to maximize $P(\mathbf{\Phi}|I_D)$ with respect to both the joint locations ϕ_k and the mode weights ρ_{ijk} alternately. In addition, as the locations of the joints are highly correlated, such inter-dependence can be further utilized to resolve the ambiguous predictions. To be specific, we perform PCA analysis to the joint locations in the training data to learn a low dimensional representation Ω of the hand configuration. During maximization of the posterior, $\mathbf{\Phi}$ is constrained to take the linear form $\mathbf{\Phi} = \sum_m^M \alpha_m \boldsymbol{e}_m + \boldsymbol{\mu}$, $M \ll 3 \times K$, where $\{\boldsymbol{e}_m\}$ is the set of the principal components. The problem to find the optimal $\mathbf{\Phi}^*$ is thus formulated as follows:

$$\begin{aligned} \mathbf{\Phi}^*, \rho^* &= \arg\max_{\mathbf{\Phi},\rho} \log P(\mathbf{\Phi}|I_D) \\ &= \arg\max_{\mathbf{\Phi},\rho} \sum_{i,k} w_{ik} \log \left[\sum_j \rho_{ijk} \exp\left(-\frac{\left\| \phi_k - \boldsymbol{v}_{ijk} \right\|^2}{\delta^2} \right) \right] \\ s.t.\ \mathbf{\Phi} &= \textstyle\sum_m \alpha_m \boldsymbol{e}_m + \boldsymbol{\mu}, \sum_j \rho_{ijk} = 1 \end{aligned} \tag{5.9}$$

Note that the constraint on the mode weights is enforced only within the J votes, and ρ_{ijk} can thus be optimized separately in Formula 5.9 during the E step. By maximizing $P(\mathbf{\Phi}|I_D)$ with

respect to ρ_{ijk} we can get:

$$\rho^*_{ijk} = \begin{cases} 1 \quad if\ j = \arg\max_j \exp\left(-\frac{\left\|\phi_k - \boldsymbol{v}_{ijk}\right\|^2}{\delta^2}\right) \\ 0 \qquad\qquad otherwise \end{cases} \tag{5.10}$$

This result conforms to our goal to filter out the inconsistent modes, *i.e.*, the optimized weights ρ^*_{ijk} only keep the mode $\boldsymbol{v}^*_{ijk}$ that is most consistent with the current estimation ϕ_k among the maximum J modes of the pixel i, and discard all other modes. Given ρ^*_{ijk}, $P(\phi_k|\boldsymbol{p}_i)$ is now simplified to a uni-modal distribution, and Formula 5.9 is thus easy to optimize in the M step. Also, the partitioned representation for each joints can be written as: $\phi_k = \sum_m \alpha_m \boldsymbol{e}_{m,k} + \mu_k$, where $\{\boldsymbol{e}_m\} = [\boldsymbol{e}^T_{m,1},,\boldsymbol{e}^T_{m,K}]^T$ is a partition of the principal components for each joint k. The M step is thus equivalent to finding the optimal coefficients $\{\alpha^*_m\}$ to maximize the posterior distribution:

$$\begin{aligned} \boldsymbol{\Phi}^* &= \arg\max_{\boldsymbol{\Phi}} \sum_{i,k} w_{ik} \log\left[\exp\left(-\frac{\left\|\phi_k - \boldsymbol{v}^*_{ijk}\right\|^2}{\delta^2}\right)\right] \\ &= \arg\min_{\boldsymbol{\Phi}} \sum_{i,k} w_{ik} \frac{\left\|\phi_k - \boldsymbol{v}^*_{ijk}\right\|^2}{\delta^2} \\ &= \arg\min_{\alpha} \sum_{i,k} w_{ik} \frac{\left\|\sum_m \alpha_m \boldsymbol{e}_{m,k} + \mu_k - \boldsymbol{v}^*_{ijk}\right\|^2}{\delta^2} \end{aligned} \tag{5.11}$$

Denote $\boldsymbol{\chi} = [\chi^T_1, ..., \chi^T_K]^T$, where $\chi_k = \sum_i w_{ik} \boldsymbol{v}^*_{ijk}$ and $\boldsymbol{\chi}$ is thus the weighted sum of the filtered votes $\boldsymbol{v}^*_{ijk}$ from all the candidate pixels. Without the low-dimensional assumption of the joint parameter space, the solution ϕ^*_k of Formula 5.11 actually equals to χ_k. Given the constraints $\boldsymbol{\Phi} = \sum_m \alpha_m \boldsymbol{e}_m + \boldsymbol{\mu}$, and note that Formula 5.11 is in the quadric form of the variable α, we can thus find the optimal solution of α by setting its derivative with respect to α to zero. The derivative of the function to be minimized in Formula 5.11 with respect to the

coefficient α_x takes the following form:

$$\begin{aligned}
\frac{\partial \log P}{\partial \alpha_x} &= 2 \sum_k \sum_i w_{ik} \frac{\boldsymbol{e}_{x,k}^T \left(\sum_m \alpha_m \boldsymbol{e}_{m,k} + \mu_k - \boldsymbol{v}_{ijk}^* \right)}{\delta^2} \\
&\propto \sum_k \sum_i w_{ik} \frac{\sum_m \alpha_m \boldsymbol{e}_{x,k}^T \boldsymbol{e}_{m,k}}{\delta^2} - \sum_k \sum_i \frac{w_{ik} \boldsymbol{e}_{x,k}^T \left(\boldsymbol{v}_{ijk}^* - \mu_k \right)}{\delta^2} \\
&\propto \sum_k \sum_m \alpha_m \boldsymbol{e}_{x,k}^T \boldsymbol{e}_{m,k} - \sum_k \sum_i w_{ik} \boldsymbol{e}_{x,k}^T \left(\boldsymbol{v}_{ijk}^* - \mu_k \right) \\
&= \sum_m \alpha_m \sum_k \boldsymbol{e}_{x,k}^T \boldsymbol{e}_{m,k} - \sum_k \sum_i w_{ik} \boldsymbol{e}_{x,k}^T \left(\boldsymbol{v}_{ijk}^* - \mu_k \right) \\
&= \alpha_x - \sum_k \sum_i w_{ik} \boldsymbol{e}_{x,k}^T \left(\boldsymbol{v}_{ijk}^* - \mu_k \right)
\end{aligned} \tag{5.12}$$

In the above derivation we use the property of the principal components, *i.e.* $\sum_k e_{x,k}^T e_{m,k}$ equals to 1 for $m = x$ and 0 otherwise. By setting $\partial f / \partial \alpha_x = 0$, we can get the solution $\{\alpha_x^*\}$ for the optimization problem:

$$\begin{aligned}
\alpha_x^* &= \sum_k \boldsymbol{e}_{x,k}^T \sum_i w_{ik} \left(\boldsymbol{v}_{ijk}^* - \mu_k \right) \\
&= \sum_k \boldsymbol{e}_{x,k}^T \left[\sum_i w_{ik} \boldsymbol{v}_{ijk}^* - \mu_k \right]
\end{aligned} \tag{5.13}$$

We can see the optimal solution of $\{\alpha_m^*\}$ with the constraints is the projection coefficients of $\boldsymbol{\chi}$ on the principal component subspace, *i.e.* $\alpha_m^* = \boldsymbol{e}_m^T (\boldsymbol{\chi} - \boldsymbol{\mu})$. The optimal joint locations $\boldsymbol{\Phi}^*$ are then reconstructed by back projecting the coefficients $\{\alpha_m^*\}$ to the original space with the principal components.

To sum up, the proposed multimodal prediction fusion (MPF) algorithm consists of a series of iterations. During each iteration, the pixel modes that are inconsistent with the current estimation are first filtered out. The optimal joint locations are then obtained by maximization of the posterior with the remaining modes subject to the hand motion constraints Ω, and the solution proves to be the reconstructed vector of the weighted sum of the filtered votes $\boldsymbol{\chi}$ on the space Ω. To start the iterative procedure, we first calculate the weighted sum of the unfiltered votes from all the candidate pixels, *i.e.* $\phi_k(0) = \sum_{i,j} w_{ijk} \boldsymbol{v}_{ijk}$, and then choose $\boldsymbol{\Phi}^*(0)$ to be

the reconstructed vector of $\{\phi_k(0)\}_{k=1}^{K}$ on the subspace spanned by $\mathbf{\Phi} = \sum_m \alpha_m \boldsymbol{e}_m + \boldsymbol{\mu}$ to initialize the EM steps. The E and M steps then iterate until a minimum increase of $P(\mathbf{\Phi}|I_D)$ or a maximum number of iterations are met. Also note that the value of $P(\mathbf{\Phi}|I_D)$ is monotonically increasing during both the E and M steps, and the optimization algorithm is thus guaranteed to converge.

5.4 Experimental Results

In this section we present the experimental results on both synthesized dataset and real-world dataset. The synthesized dataset is used for both forest training and quantitative evaluation of the prediction accuracy. The real-world dataset is used to test the generalization ability of the proposed method when the regression forest is trained on synthesized datasets. In addition, we also investigate the impact of various parameters on the system performance.

5.4.1 Datasets and Evaluation Metrics

The synthesized dataset consists of $114.2k$ templates to quantitatively evaluate the performance of the methods for a large variety of hand configurations. Each template in the dataset includes the depth image, the ground truth of the hand part labels and the sixteen joint positions. Similar to [73], we use a CyberGlove II [4] to capture the hand articulation parameters for various hand motions, *e.g.*, grasping, pinching, single and multiple finger bending, performing ASL gestures, etc. The captured local articulation parameters are clustered to approximately 400 templates. The range of global hand rotation is defined to be to $(-60°, 20°)$ for global rotation around the X axis and $(-80°, 80°)$ around Y axis, *i.e.* the axes parallel to the image plane of the camera, and $(-35°, 35°)$ around the Z axis, *i.e.* the axis perpendicular to the image plane. The global rotation parameters are discretized uniformly into near 300 sets within this range, and combined with the local articulation parameters to drive the 3D hand model in 3.1 to generate

Figure 5.6: Examples of the hand configurations with large viewpoint variations and different finger articulations in the synthesized dataset.

the synthesized datasets. This dataset is quite challenging for joint location prediction as a large portion of the joints are invisible in the templates, as shown in Fig. 5.6. In the experiments 80% of the synthesized templates are used to train the regression forests and the rest 20% for testing, *i.e.* $91.4k$ vs $22.8k$.

To demonstrate that the proposed depth+semantic contexts based regression forest and the MPF algorithm can well handle the discrepancy between the synthesized datasets and real-world inputs, we annotate the NTU Hand Posture Dataset [42], which consists of in total 1354 real depth images of four subjects using a SoftKinetic DS325 camera. In this dataset the hands of the subjects go through large viewpoint changes and various postures, and the hand sizes also vary considerably for the different subjects. In addition, the depth points are relatively noisy compared to the synthesized image; the wrist of the real hand inputs is not well segmented and part of the lower arm is still visible, as shown in Fig. 5.7. The resulting per-pixel votes can thus be multimodal, as in Fig. 5.4. This is disadvantageous for independent mode-seeking for each joint alone, since it can easily get trapped in local optima.

For each image in NTU Hand Posture Dataset, the Iisu Middleware SDK provided by Soft-

Kinetic [90] is utilized to get relatively rough hand segmentation from the depth image. The ground truth joint positions in NTU Hand Posture Dataset are obtained by manual annotation of one annotator. To ensure the annotation quality, we require the 3D hand skeleton to match both the 2D depth image and the 3D point cloud, as shown in Fig. 5.7. Each pixel in the depth image view is associated with a point in the 3D point cloud view. Starting from the initial skeleton template in Fig. 5.7 (a), the depth image is first used for fast annotation of the non-occluded joints by moving the projected joints to the correct pixel, as shown in Fig. 5.7 (b). The 3D position associated with the pixel is assigned to the joint. Since this position lies on the hand surface, an offset of $0.75cm$ is then added to the depth of the joint to compensate the surface-to-interior distance. As shown in Fig. 5.7 (b), the four occluded joints in the yellow rectangle cannot be annotated in the depth image. Therefore, the occluded joints are labeled by moving each of them in the 3D space and matching to the point cloud as in Fig. 5.7 (c). Besides, the inaccurate annotations in the previous step, *e.g.*, the wrist, can also be corrected with the point cloud. Fig. 5.7 (d) illustrates some annotated examples in the dataset. This dataset and its ground-truth annotations can be found on our website [1].

For performance evaluation of the methods on the synthesis dataset and NTU Hand Posture Dataset, we define the prediction accuracy for each joint as the percentage of the predictions that are within a distance of D_T centimeters from the ground truth. The overall accuracy is obtained by the average of the prediction accuracies of the sixteen joint locations. In this chapter we define $D_T = 1.5cm$, while the overall accuracies with different values of D_T *i.e.* $D_T \in [1.0, 4.0]$, are also provided to better illustrate the distribution of correct joint predictions. Note that the high accuracy corresponding to small values of D_T should be more favorable, as the large values of D_T indicate imprecise estimates.

[1] https://sites.google.com/site/seraphlh/projects

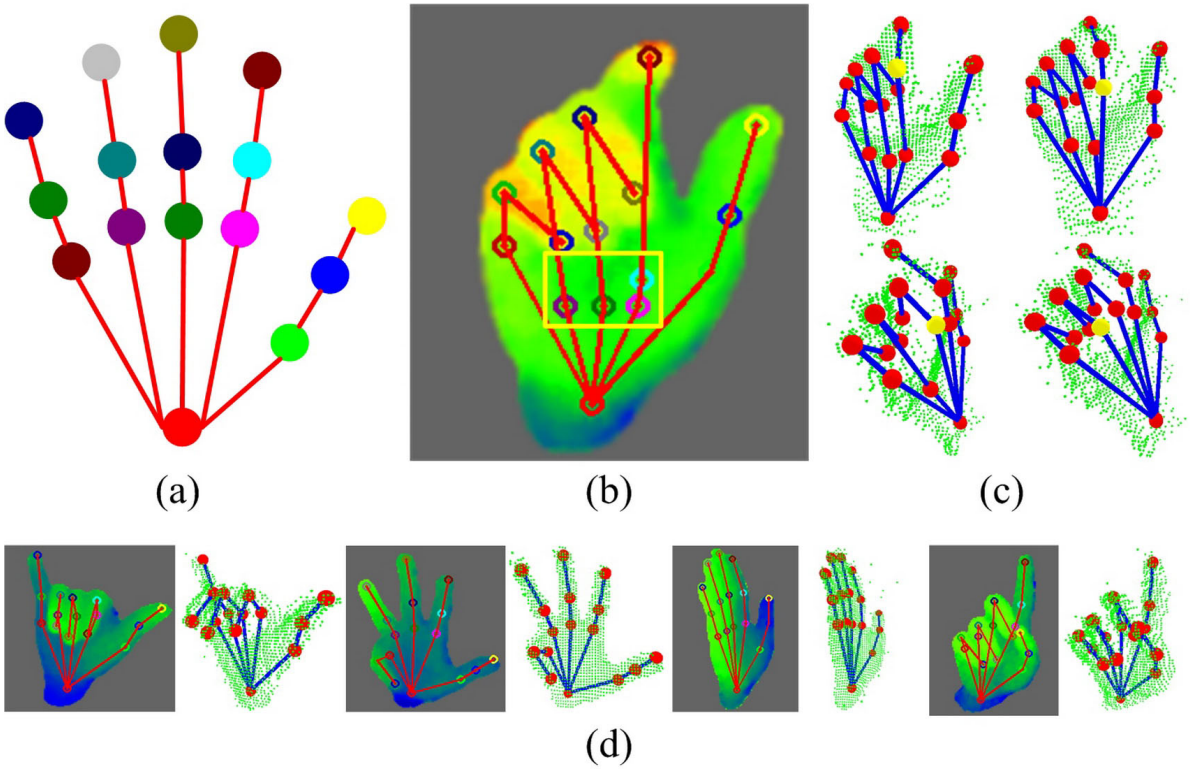


Figure 5.7: Joint position annotation on the real dataset by manually fitting the hand skeleton to the depth image and the 3D point cloud. (a) The initial skeleton template for annotation. (b) Fast annotation of the non-occluded joints by moving their projection to the correct pixel in the depth image view. (c) Occluded joints are labeled by moving and matching in the 3D point cloud view. (d) Some annotated examples in the dataset.

5.4.2 Implementation Details

We implemented [41] and [85] with the depth context descriptor in Formula 5.1 for comparison. For [41], the E^{cls} criterion is adopted to learn the tree structure, which minimizes the Shannon entropy of the hand part labels at the split nodes and is reported to have the best performances. The regression forest is then learned in the same way to Section 5.2. Following [41], the final joint locations are obtained by mode-seeking with the Mean-Shift algorithm based on the votes and weights produced by per-pixel regression. The density estimator for each joint location is given by:

$$g_k(\boldsymbol{v}) = \sum_{i=1}^{N} \sum_{j=1}^{J} w_{ijk} \exp\left(- \left\| \frac{\boldsymbol{v} - \boldsymbol{v}_{ijk}}{b_k} \right\|^2 \right) \tag{5.14}$$

where $\boldsymbol{v}$ is the 3D point, $(\boldsymbol{v}_{ijk}, w_{ijk})$ is the pixel votes of the i^{th} pixel, and b_k is the bandwidth.

To implement [85], the same E^{cls} criterion and density estimator is used, and we perform Mean-Shift mode-seeking to find maximum five modes for each joint. The Dynamic Programming (DP) algorithm is utilized to find their best combination given the bone length constraints, as in [85]. In addition to (A) $\boldsymbol{\mathcal{D}}$ and MS [41] and (B) $\boldsymbol{\mathcal{D}}$ and DP [85], we further tested other three methods: (C) Regression with $\boldsymbol{\mathcal{D}}$ and MPF fusion; (D) Regression with $\boldsymbol{\mathcal{D}} + \boldsymbol{\mathcal{S}}$ and Mean-Shift (MS) mode-seeking; (E) Regression with $\boldsymbol{\mathcal{D}} + \boldsymbol{\mathcal{S}}$ and MPF fusion. In the experiments, the regression forests consist of 4 trees with a maximum depth of 20. To learn the tree structure of the regression forest, 10000 candidate split functions are selected to train each tree, and each leaf node contains at most three relative votes for each joint. During testing, 1000 pixels are randomly selected for per-pixel regression, and the EM iteration lasts for at most three times in the MPF algorithm. These parameters are used in the following tests unless explicitly specified. All the evaluated methods were coded in C++/OpenCV, and tested on a server with two Intel Xeon X5675 CPUs and 16G RAM. The resolution of the images in all the datasets is 320×240.

5.4.3 Quantitative Evaluations on Synthesis Datasets

The results obtained with [41], [85] and our three methods on the synthesized dataset are summarized in Table 5.1, which consist of the average prediction accuracies ζ_{pred} for $D_T = 1.5cm$, the time costs for hand parsing, per-pixel regression and joint prediction with MS, DP and MPF, denoted as t_{parse}, t_{reg}, t_{MS}, t_{DP} and t_{MPF}, and the overall time cost t_{total}. Besides, note the regression forest can also classify the hand part labels with the label distribution in the leaf nodes, we also report the part classification accuracy ζ_{cls} over all the pixels in Table 5.1 for reference, while it is worth mentioning that only a small number of the pixels are needed for regression to predict the joint positions. Fig. 5.8 shows the overall accuracies for $D_T \in [1.0, 4.0]$.

In this experiment the training and testing data are relatively consistent as they are synthesized by the same hand model, and therefore the per-pixel votes obtained with the regression

forest are compact and the problem of multimodal joint predictions is not quite serious. Overall, the five tested methods produce good prediction accuracy, as shown in Table 5.1, and all our three methods outperform the baseline methods [41] and [85]. Compared to [41], the prediction accuracy is improved by 4.33% via incorporating the semantic context in discriminative regression alone, and is improved by 3.20% via the MPF algorithm. Finally, the combination of the depth and semantic contexts and the MPF algorithm obtains the highest prediction accuracy, which is 6.42% enhancement compared to the baseline. In contrast, [85] obtains very little improvement upon [41] by utilizing the bone length constraints alone, as the motion constraints between the joints are still not modeled. Moreover, the results in Fig. 5.8 show that the predictions obtained with the proposed methods are more compact, as the distributions with different values of D_T indicate that the joint predictions obtained by our methods are nearer to the ground truth joint locations on average.

Table 5.1: Overall comparison of [41], [85] and our three methods on the synthesized dataset. The time costs are evaluated in milliseconds. SAA denotes Same As Above.

Method	ζ_{pred}	ζ_{cls}	t_{parse}	t_{reg}	t_{MS}	t_{DP}	t_{MPF}	t_{total}
(A) $\mathcal{D}$ and MS [41]	83.26%	81.96%	-	38.81	1.35	-	-	42.70
(B) $\mathcal{D}$ and DP [85]	83.51%	SAA	-	38.70	-	6.64	-	47.90
(C) $\mathcal{D}$ and MPF	86.46%	SAA	-	37.63	-	-	2.66	43.59
(D) $\mathcal{D} + \mathcal{S}$ and MS	87.59%	**86.18%**	28.16	41.15	1.21	-	-	74.04
(E) $\mathcal{D} + \mathcal{S}$ and MPF	**89.68%**	SAA	28.23	40.88	-	-	2.28	75.83

5.4.4 Quantitative Evaluations on NTU Hand Posture Dataset

For this experiment we tested [41], [85] and our three methods with the regression forests trained on the synthesis datasets in Section 5.4.1. Due to the noisy depth image, inaccurate wrist

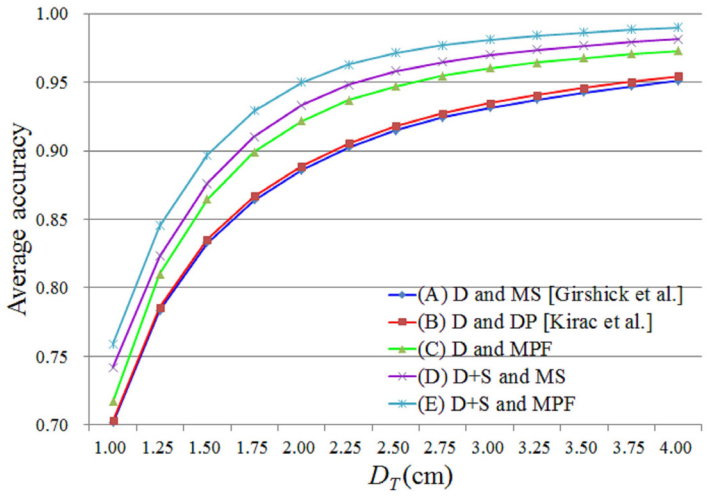


Figure 5.8: Comparison of the average prediction accuracies on the synthesized dataset for different D_T.

segmentation and the various hand sizes of the different subjects, the performance of all the five methods degrades substantially on this dataset when the regression forests and the PCA space in the MPF algorithm are learned with the synthesized data. However, we show that the MPF is capable of compensating the gap between the training data and the real-world input to some extent by learning the PCA space from the real data.

We randomly pick up a small portion of the real dataset, *i.e.* 270 images, to learn the PCA space for the MPF algorithm and use the rest 1084 images for testing. Besides, the bone length constraints of the hand for [85] are obtained by calculating the average bone lengths in the 270 images. The performance of the MPF algorithm is also evaluated with the PCA space learned from the synthesized training data for comparison, which is denoted as (C*) and (E*). Fig. 5.9 shows the comparison of the overall accuracies for $D_T \in [1.0, 4.0]$. As illustrated, the improvement of [85] over [41] is still trivial. Besides, MPF alone does not perform quite well with the PCA space learned from the synthesized data, while it still produces 9.09% enhancement when combined with the semantic context compared to [41]. In contrast, with the constraints learned from only a small portion of the dataset, MPF achieves 64.18% prediction accuracy with the depth context and 72.40% with the depth and semantic contexts,

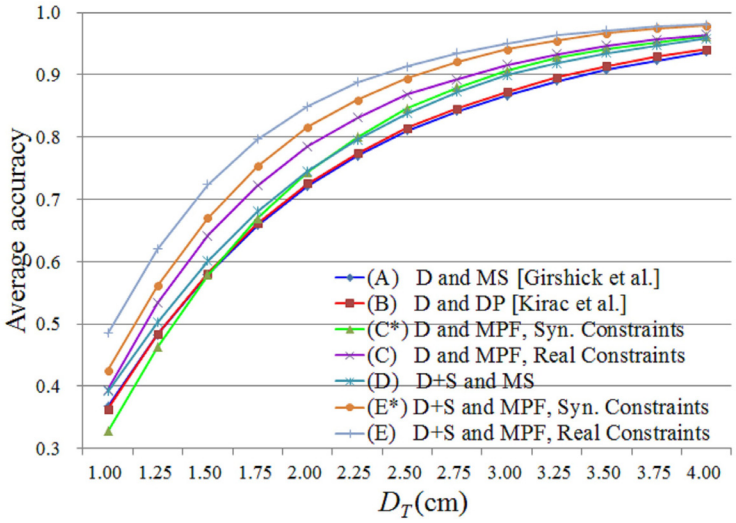


Figure 5.9: Comparison of the average prediction accuracies on the real-world dataset for different D_T.

which improves [41] by 6.22% and 14.44% respectively. As it is generally difficult to model both the pose and size variations of the hand to train the regression forest due to the huge amount of training data, these results give us an insight into another way to generalize well to different users. Fig. 5.10 and Fig. 5.11 show some exemplary results obtained with [41], (C) and (E). The 3D joint positions are projected onto the image plane to get their 2D positions and overlaid on the depth images for better illustration.

It is interesting to note that the semantic context and MPF appear to utilize the hand part correlations from different aspects and complement each other. That is, both methods can improve the prediction accuracy separately, while their combination produces further improvement. By utilizing the parsed hand parts, (D) improves [41] by 2.13%. On the other hand, with MPF, (C) improves [41] by 6.22%, while the best performance, *i.e.* 14.44% improvement, is obtained by the combination of the two. A similar conclusion can also be drawn from the results on the synthesized dataset.

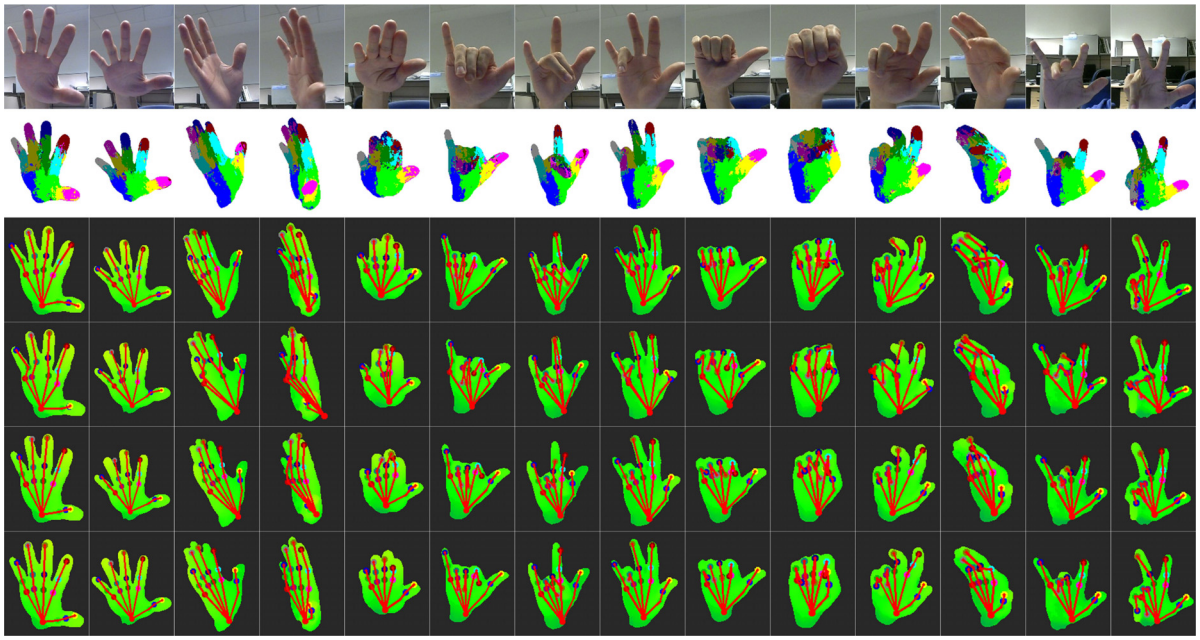


Figure 5.10: Comparison of joint position predictions with [41], (C) $\mathcal{D}$ and MPF and (E) $\mathcal{D}+\mathcal{S}$ and MPF on the real dataset. **Second row**: parsed hand parts with the RDF. **Third row**: ground truth joint annotations. **Fourth row**: predictions obtained with [41]. **Fifth row**: predictions obtained with (C). **Sixth row**: predictions obtained with (E).

5.4.5 Extension to Multi-Layered Forest

The results on the synthesis dataset and NTU Hand Posture Dataset demonstrate that the parsed hand parts can effectively improve the joint prediction accuracy at the extra cost to classify all the pixels first. As discussed in Section 5.2.1, our two-layered forest can be easily extended to N_L layers, where $N_L = 1$ means no parsed label inputs from the previous layer, *i.e.* the regression forest in [41], and $N_L = 2$ corresponds exactly to our regression forest in Section 5.2.1, *etc.*. To implement an N_L-layered forest, N_L separate forests must be trained sequentially. The first layer forest is trained with the training depth images in the synthesized dataset. Given that the n-layer forest is trained, it is then used to parse the training depth images to get the label images, and the depth and newly parsed label images are used to train the $n+1$-layer forest. This procedure continues until all N_L forests are trained.

We test the performance of the Mean-Shift algorithm and the MPF algorithm with the per-

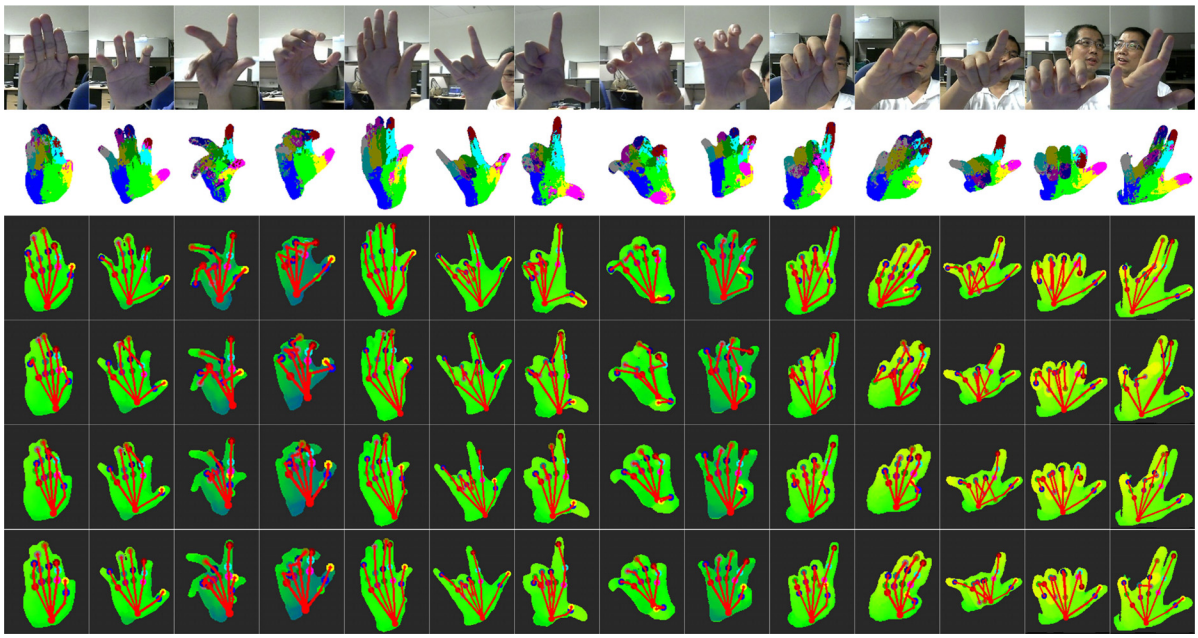
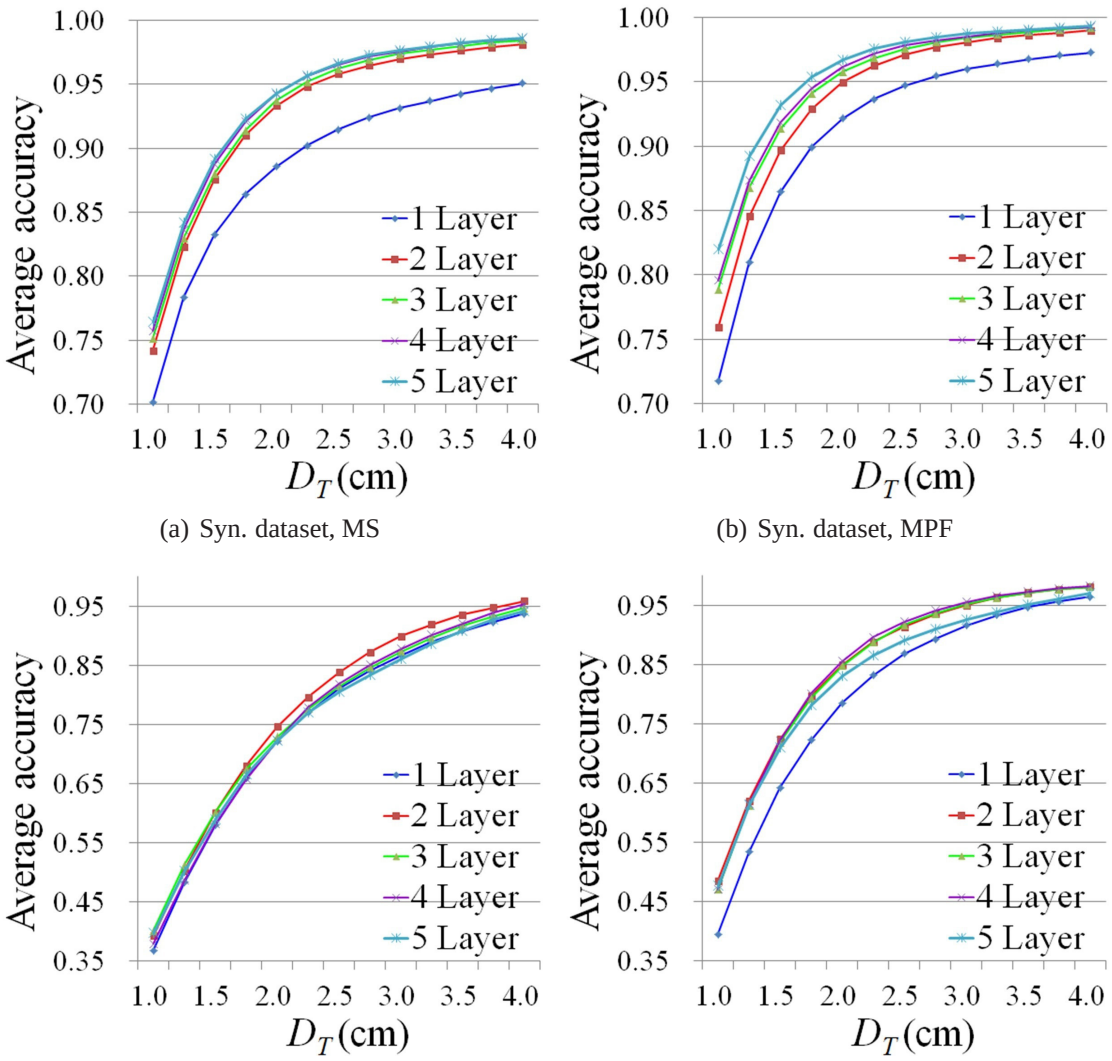


Figure 5.11: More results on the real dataset. **Second row**: parsed hand parts with the RDF. **Third row**: ground truth joint annotations. **Fourth row**: predictions obtained with [41]. **Fifth row**: predictions obtained with (C). **Sixth row**: predictions obtained with (E).

pixel votes from the N_L-layered forest for different N_L. Fig. 5.12 illustrates the results on the synthesized and real datasets for $N_L = 1, ..., 5$. Overall, $N_L = 2$ improves the performance by the largest margin on all the tests. On the synthesized dataset we observe that the prediction accuracy keeps improving when N_L increases, and the hand part classification accuracy shows the same trend, as in Table 5.2. However, on the real dataset the joint prediction accuracy increases very little for $N_L > 2$ or even begins to drop for large N_L. This may indicate that using too many layers of the forest tends to overfit the synthesized training data, and does not necessarily improve the performance for real inputs.

Table 5.2: The average hand parsing accuracies on the synthesized dataset for different N_L.

Layer Num	1 Layer	2 Layer	3 Layer	4 Layer	5 Layer
Accuracy	81.96%	86.18%	88.31%	89.42%	90.14%



(a) Syn. dataset, MS (b) Syn. dataset, MPF

(c) Real dataset, MS (d) Real dataset, MPF

Figure 5.12: Comparison of the average prediction accuracies on the synthesized and real datasets for different N_L.

5.4.6 Evaluations on the Parameters

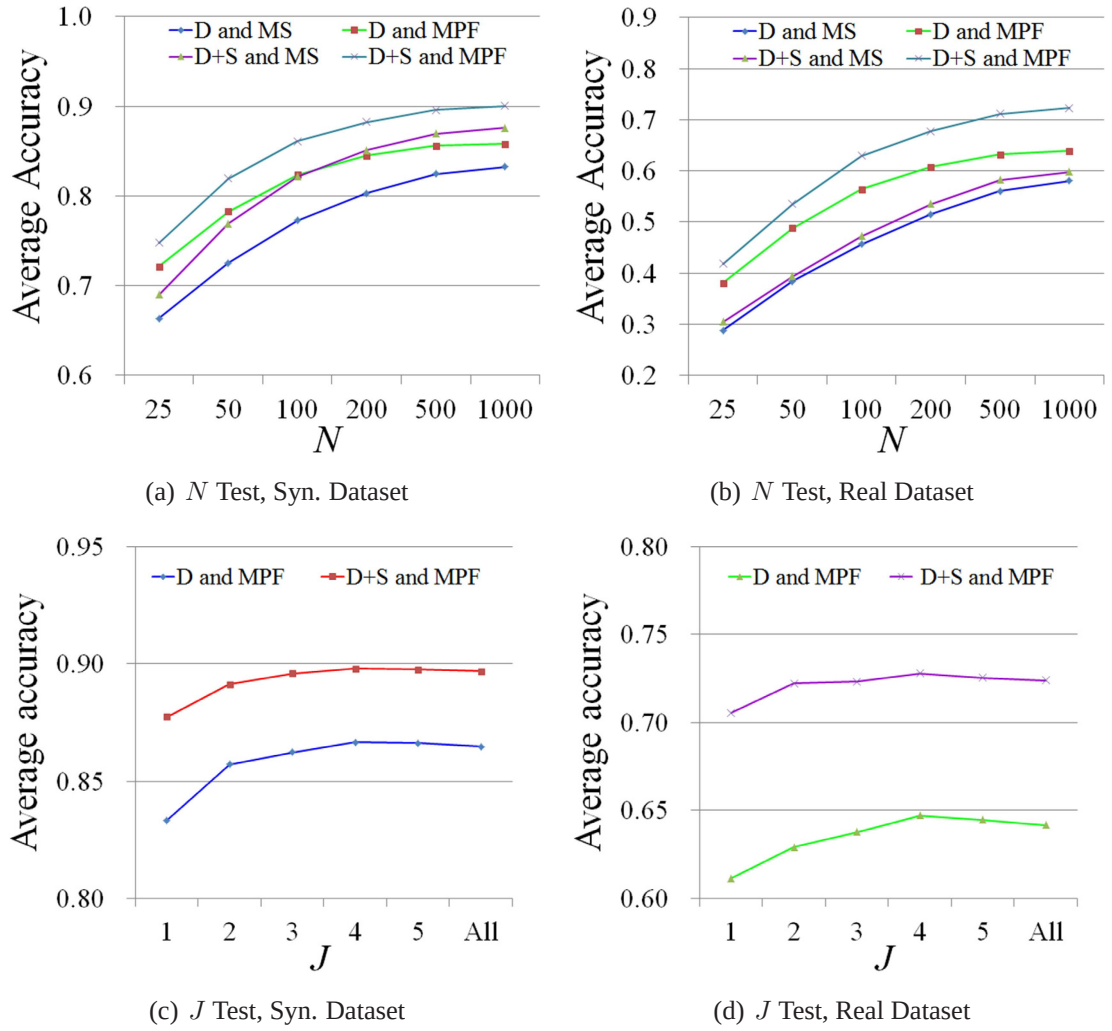
In this part we investigate the impact of the following parameters: the number of pixels used for voting N, the maximum number of per-pixel votes J and the maximum number of EM iterations. Besides, we also investigate the impact of the amount of the synthesized training data on the performance on the real dataset. For each test the uninvestigated parameters take the same values as in Section 5.4.2.

Number of voting pixels N. Fig. 5.13 (a-b) illustrates the prediction accuracy changes with respect to different N on the synthesized and real datasets, where we use $D_T = 1.5cm$. All the methods show the same trend on both datasets, *i.e.*, the more pixels used for voting, the better the prediction accuracy. While the computational cost for per-pixel regression is approximately linear with N, it should be noted that all the pixels need to be classified in advance if the semantic context is to be used, which leads to a relatively constant time cost regardless of N for hand parsing. This makes the methods "$\boldsymbol{\mathcal{D}} + \boldsymbol{\mathcal{S}}$ and MS"and "$\boldsymbol{\mathcal{D}} + \boldsymbol{\mathcal{S}}$ and MPF"inefficient especially when only a small number of pixels are used for voting.

Maximum number of per-pixel votes J. As defined in Section 5.4.2, at most three votes are stored for each joint in the leaf nodes by mode seeking in the samples. In the experiment we actually find most leaf nodes contain only one vote for each joint and the average number is about 1.2. Since the forest consists of four trees, the average number of J should be between 4 and 5. Therefore, we investigate the performance of the MPF algorithm for $J = 1, ..., 5$, where $J = 1$ means there is no mode selection procedure as in Formula 5.10. Fig. 5.13 (c-d) presents the prediction accuracies for $D_T = 1.5cm$ on the synthesized and real datasets, and "All" means all the retrieved votes are used for the pixels. While $J = 1$ has the poorest performance, the accuracy does not improve much for $J \geq 3$ and $J = 3$ produces an average of 2.3% improvement over $J = 1$.

Maximum number of EM iterations. The results in Section 5.4.4 demonstrate the good performance of the MPF algorithm, and we thus investigate the drop of the average joint prediction error with respect to the number of EM iterations in MPF. Fig. 5.14 (a) illustrates the results on both datasets, which indicates the prediction error decreases quite fast within the first several iterations, especially on the real dataset. Since the prediction error changes very little after about three iterations, we choose the maximum number of EM iterations as three in all the other experiments.

Synthesized training data size. In Fig. 5.14 (b) we compare the prediction accuracy of



(a) N Test, Syn. Dataset

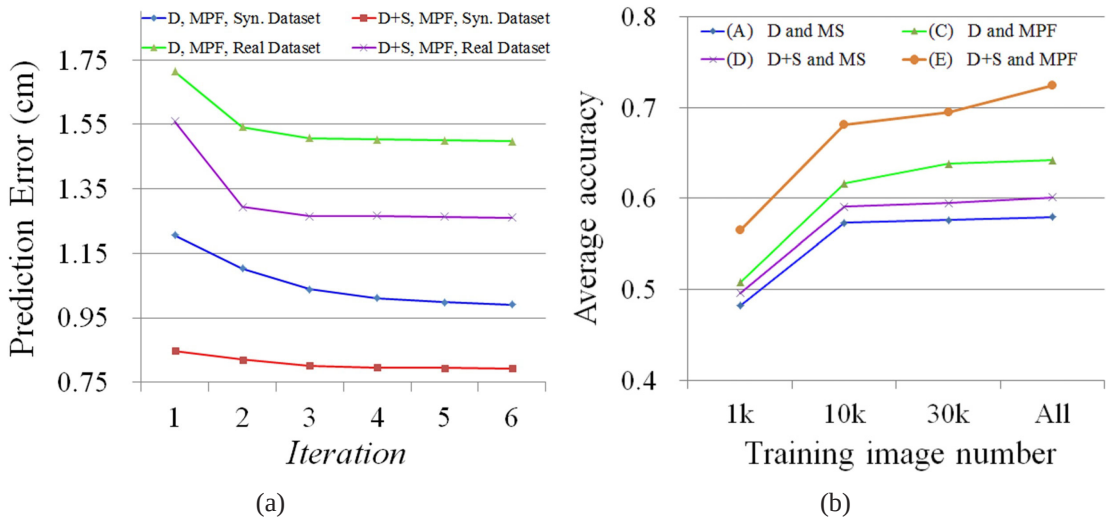
(b) N Test, Real Dataset

(c) J Test, Syn. Dataset

(d) J Test, Real Dataset

Figure 5.13: System parameter tests for voting pixel number N and per-pixel vote number J. (a-b) Accuracy change of [41] and our three methods for different N on both datasets. (c-d) Accuracy changes of the MPF algorithm for different J on both datasets.

the four methods on the real-world dataset when the regression forests are trained with different sizes of synthesized data, where $D_T = 1.5cm$. "All" means the $91.4k$ training images are used, as in the other experiments. The results are consistent with that in Section 5.4.4, and our methods still outperform [41].



(a) (b)

Figure 5.14: (a) Drop of the prediction error with more EM iterations on both the synthesized and real datasets. (b) Prediction accuracy on the real dataset with different numbers of synthesized training data.

5.4.7 Comparison on MSRA Hand Tracking Database [40]

To further compare the proposed methods with other state-of-arts, we utilize another recent real-world hand pose dataset [40] for performance evaluation. This dataset consists of 2400 frames of various rapid hand motion captured for six different subjects, and a 400-frame video sequence is recorded for each subject. This dataset include substantial hand posture variation and scale variation of different subjects. Following [40], the pose estimation accuracy is evaluated in terms of the average error of six hand joints including five finger tips and the wrist on this dataset. Since this dataset lacks the hand part annotation, we only test the methods (A) $\mathcal{D}$ and MS [41] and (C) Regression with $\mathcal{D}$ and MPF fusion, which do not utilize the semantic context for hand pose regression. We use a leave-one-subject-out manner to test these two methods, in which the data of one subject are used for testing and the data of the rest five subjects are used to train the regression forest. Besides, a global hand model scale is specified for each subject during testing in [40] to adjust for different hand sizes. Therefore, we also use the specified scales in [40] to adjust the hand region sizes to compensate for the hand size variations.

Table 5.3: Comparison of different methods on MSRA hand tracking database [40].

Methods	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 6	Average
FORTH [28]	35.40	19.80	27.30	26.30	16.60	46.20	28.60
PSO [40]	29.30	14.80	40.20	17.30	16.20	24.30	23.68
ICP [91]	29.90	20.70	30.80	23.90	18.50	32.80	26.10
ICP-PSO [40]	10.10	24.10	13.00	12.80	11.90	20.00	15.32
ICP-PSO* [40]	8.60	7.40	9.80	10.40	7.80	11.70	9.28
$\mathcal{D}$ and MS [41]	33.00	25.30	25.20	25.60	38.10	30.70	29.65
Proposed $\mathcal{D}$ and MPF	29.90	22.30	23.60	23.80	35.10	28.80	27.25

In Table 5.3 we compare the performance of different methods for hand pose estimation on MSRA hand tracking database [40], which reports the average errors of the predicted hand pose and the ground truth for each individual subject and the average prediction error of all the subjects. In addition to the methods (A) [41] and (C) tested, Table 5.3 also includes the results from other state-of-arts, including FORTH [28], ICP [91], PSO [40], ICP-PSO and ICP-PSO* [40]. Note that the results of the proposed method (c) is comparable to the methods FORTH [28], ICP [91] and PSO [40]. Particularly, with the proposed MPF algorithm, the prediction error is reduced by 8.1% compared to [41].

It is worth noting that in Table 5.3, the results for FORTH [28], ICP [91], PSO [40], ICP-PSO and ICP-PSO* [40] are all based on hand pose tracking with continuous image sequences, *i.e.* the results are obtained by utilizing the information from the current and previous several frames altogether and the ground truth hand pose are used to initialize the first frame during tracking. Thus their predictions are much smoother. In contrast, our results are obtained by using only single-frame information. Moreover, the images used for training is still too few, *i.e.* 2000 images for each subject for leave-one-subject-out testing. In contrast, the training images in our experiments in Section 5.4.3 and 5.4.4 are more than 90k. Such small amount of training data makes it difficult to capture a reasonable range of hand pose and shape variations. Finally, there is not hand part annotation in this dataset, which seriously degrades the performance of

the proposed methods, as it has been shown that the semantic context can largely improve the hand pose estimation results in our previous experiments.

5.5 Discussion

This chapter presents a discriminative framework for hand pose estimation, which utilizes the hand part correlations to improve the joint prediction accuracy. Compared to the model-based framework in Chapter 4 that heavily relies on fingertip detection, this framework is much more robust to arbitrary hand motion. Especially, the good results are obtained by single-image estimation, and thus there is still large room for improvement by taking advantage of the temporal constraints, such as the linear dynamic system models [95, 96]. One drawback of this framework is the inherent problem with the template-matching methods. That is, it does not produce continuous pose estimation due to the discrete sampling of the pose parameters when generating the training dataset. In this chapter this issue is partially handled by densely sampling to generate a large dataset for training. However, it would be better to combine our model-fitting framework in Chapter 4 and this discriminative framework to utilize the former's capability of continuous prediction and the latter's robustness. We regard this point as our future research direction.

Chapter 6

Conclusion

This thesis has focused on the problem of vision-based hand pose estimation and gesture recognition, which has various applications in human-computer interaction. Particularly, the problem is studied mainly with the input of depth cameras, since they can capture more detailed 3D structure of the hand surface and thus reduce the ambiguity in pose prediction. The main contributions of this thesis are a detailed review of related works in the field and a set of real-time solutions that utilizes depth cameras to recover the 3D hand pose and gesture. In these solutions we lay stress on two related topics: high-level hand feature extraction and 3D hand pose estimation. The high-level features such as the fingertips and labels of fingers and palm provide compact representations of the raw visual inputs, which could facilitate hand motion analysis. To this end, we have developed an algorithm for 3D fingertip detection and tracking and an algorithm for hand parsing to classify the hand region into a set of finger and palm parts. These algorithms are also further utilized in our hand pose estimation framework and prove to improve the pose prediction accuracy considerably. For hand pose estimation, we have developed both a model-based fitting framework and a discriminative framework. The fingertip tracking algorithm is utilized in the model-based fitting framework for fast pose initialization and recovery from failure, and an articulated iterative closest point method is adopted to obtain refined hand pose estimation based on the initial pose. The discriminative hand pose estimation framework

addresses one important problem in the spatial-voting based methods: what is the good way to fuse the independent pose votes from the spatially-distributed voting elements? In our work this is solved by a novel multi-modal prediction fusion method to naturally fuse the votes with low-dimensional pose parameter constraints. Based on these algorithms, this thesis also presents a set of applications for human-computer interaction. Our work in this thesis is summarized as follows.

First, we present a literature review to discuss and analyze the existing work in vision-based hand pose estimation and tracking and the related key techniques, which mainly include hand feature extraction from visual images and the various discriminative and generative model-fitting techniques for hand pose estimation. The low-level features such as edges and silhouette and the high-level features such as fingertips and classified finger and palm parts and their effectiveness and limitations in hand pose estimation are analyzed. The general pipelines of generative model-fitting methods and discriminative template-matching methods in hand pose estimation are presented, and their strengths and limitations are discussed.

Second, we present a hand parsing scheme with the input of depth images. A depth context feature with the distance-adaptive sampling scheme is proposed to provide accurate hand part classification, and a Superpixel-MRF framework is developed to enforce both the spatial and temporal constraints to improve the per-pixel classification results. The results on both the synthesized depth images and real-world test sequences demonstrate the good performance of the proposed hand parsing scheme compared to other state-of-the-art methods. Moreover, the tests on the human body annotation dataset further demonstrate the generalization power of our method to different kind of articulated objects. In our following work on discriminative hand pose estimation, the hand parsing results have further demonstrated to be quite effective to improve the hand pose estimation accuracy.

Third, we present a model-based framework for hand pose estimation, which lay stresses on the problem of robust fingertip tracking in depth images for initial hand pose estimation and

recovery of tracking failure. While many existing fingertip detection and tracking approaches cannot track the 3D positions of fingertips robustly due to the flexible finger motion, we address these issues by using multiple depth-based features for accurate fingertip localization and adopting a particle filter to track the fingertips over successive frames. Compared to existing contour-analysis methods and geodesic extrema based methods, this fingertip tracking algorithm can handle challenging cases such as bending and side-by-side fingers. In addition, we develop a divide-and-conquer scheme for model-based hand pose estimation, which handles the high dimensionality of the hand pose parameters. However, the method still has some limitations. The heuristics used to locate the fingertips are mostly empirically determined, and thus they are inadequate to robustly track the fingertips under various complex hand configurations, such as fist hand posture and long-term twisted fingers. Such limitations can degrade the performance of the model-based hand pose estimation framework built upon fingertip tracking. In Chapter 5 we thus present a fully data-driven framework for hand joint prediction including the fingertips, which is much more robust even for very challenging hand postures.

Fourth, we present a discriminative framework for hand pose estimation in depth images, in which the random regression forest is adopted to predict fully articulated hand pose with the depth inputs. Compared to the fingertip localization algorithm in Chapter 4, the random regression forest in this framework is constructed in a full data-driven manner by training on a large annotated hand posture dataset. Therefore, it can predict more hand joints including the fingertips and handle arbitrary hand postures that are covered by the training data. As the final hand pose prediction is obtained by voting with a number of pixels with 3D offsets, the method can also predict the positions of occluded hand joints. However, in the existing random regression forest, the hand part correlations are not fully enforced. Therefore, we propose to exploit the hand part correlations to improve the hand joint prediction accuracy from two different aspects. First, it uses the parsed hand parts to extract the semantic context to construct more discriminative regression forest, which produces more compact per-pixel votes compared

to using the depth image alone. Second, a multi-modal prediction algorithm is proposed to fuse the multimodal per-pixel pose votes subject to the learned hand motion constraints. It is especially effective in handling the discrepancies between the synthesized training data and real-world inputs, and can be efficiently solved via Expectation-Maximization. The methods have shown state-of-art performances on both a synthesized dataset and two real-world datasets.

Last but not least, we have applied our work in hand pose estimation for both static and dynamic gesture recognition. In addition, to exploit the real-time capability of the proposed methods, we have built various real-time human-computer interaction applications so that a user can interact with the virtual environments with his bare hands, which include virtual object manipulation, simulated water-oscillator instrument play and human-virtual avatar interaction.

6.1 Future work

Despite the recent progress in the field of articulated hand pose estimation, the problem is still far from solved. For example, most state-of-the-art methods assume that the hand is not touching other real objects in the problem of hand pose estimation. However, the hand is always interacting with various objects in real life and interaction between hand and objects is also important to analyze human activities. Such assumption is also adopted in all the works presented in this thesis, in which we assume that the hand and forearm are the only visible objects when capture the hand images. Therefore, our methods also suffer from similar limitations. In addition, there are still other limitations in our works. First, our model-based framework and discriminative framework are still separate. This could be addressed by utilizing the generative model-fitting methods as a follow-up stage after discriminative regression. Second, our current pose inference method mainly works on single depth images and the hand motion dynamics are not fully exploited, *e.g.*, only simple linear transition models. As shown in previous work, the hand motion dynamics are also highly constrained and can help to constrain the search space

for intermediate frames and smooth the prediction over successive frames. With these concerns, we will continue our work and focus on the following several topics:

Robust hand detection. To recover the high DOF hand pose from the image inputs, the hand region needs to be accurately detected and segmented from the background and the other body parts such as the torso and forearm. However, the hand has large shape variations when performing different postures and its motion is highly flexible, which makes hand detection and segmentation in unconstrained conditions very challenging with either color or depth image inputs. Up to now the problem still has no reliable solution, and most methods need to make certain assumptions on the position or appearance of the hand to fulfill this task. We plan to address this problem with both the color and depth inputs.

Pose estimation with depth images. There are still limitations in our current work with the depth camera inputs, such as simple motion models. To this end, we plan to analyze the dynamic constraints of the joint positions and track the hand joint positions in the successive input images to reduce the jitter of the predictions. In addition, we will further enlarge the training dataset to allow more complex hand configurations. However, when the training dataset gets larger, the mapping logics between the feature and the pose parameters also become much more complex and thus require efficient learning. This will be our research focus on this topic.

Pose estimation with RGB images. Despite the progress of hand pose estimation with the depth camera, the problem is much more challenging with RGB inputs. The RGB image is more sensitive to illumination variations and difficult for hand segmentation from the background clutter. Especially, the discriminative power of the RGB image is much lower than the depth image, which introduces high ambiguity in inference. With the input of an ordinary RGB camera, we cannot expect the fine details of the hand to be captured, *e.g.*, the nails, the texture and shading of the skin, due to the usually uncontrolled environment and limited image quality. This topic is significant since the color cameras require much less power and are still far more popular than the depth cameras especially on mobile platforms.

Hand interaction with real-objects. The problem of hand pose estimation during hand-object interaction is useful for both hand motion capture and human activity understanding. However, the problem is much more difficult compared to the prevailing schemes without object interaction, due to the fact that there are both hand self-occlusion and hand-object occlusion. Despite that there are several related works [124, 125] on this topic, they either require multi-camera setting or the hand pose prediction accuracy is still too poor. As the hand and objects can be difficult to differentiate in either the depth image or RGB image alone, we plan to fuse the two channel RGB-D visual inputs to recover the hand pose and recognize their interaction in our future research.

Author’s Publications

Book Chapters

1. Hui Liang and Junsong Yuan, Hand Parsing and Gesture Recognition with a Commodity Depth Camera, in Computer Vision and Machine Learning with RGB-D Sensors, Part IV, pp. 239-265, Springer, 2014.

2. Yang Xiao, Hui Liang, Junsong Yuan and Daniel Thalmann, Body Movement Analysis and Recognition, in Context Aware Human-Robot and Human-Agent Interaction, pp. 31-53, Springer, 2016.

Journal Papers

1. Hui Liang, Junsong Yuan, Daniel Thalmann and Zhengyou Zhang, Model-based Hand Pose Estimation via Spatial-temporal Hand Parsing and 3D Fingertip Localization, in the Visual Computer Journal, vol. 29, no. 6-8, pp. 837-848, June 2013.

2. Hui Liang, Junsong Yuan and Daniel Thalmann, Parsing the Hand in Depth Images, in IEEE Trans. Multimedia, vo. 16, no. 5, pp. 1241-1253, Aug. 2014.

3. Hui Liang, Junsong Yuan and Daniel Thalmann, Resolving Ambiguous Hand Pose Predictions by Exploiting Part Correlations, in IEEE Trans. Circuits and Systems for Video Technology, vol. 25, no. 7, pp. 1125-1139, July, 2015.

Conference Papers

1. Hui Liang, Junsong Yuan and Daniel Thalmann, 3D Fingertip and Palm Tracking in Depth Image Sequences, in Proc. of the 20^{th} ACM Int'l Conf. on Multimedia, 2012.

2. Hui Liang, Junsong Yuan and Daniel Thalmann, Hand Pose Estimation by Combining Fingertip Tracking and Articulated ICP, in Proc. of the 11^{th} ACM SIGGRAPH Int'l Conf. on Virtual-Reality Continuum and its Applications in Industry, 2012.

3. Xinchen Yan, Junsong Yuan and Hui Liang, Efficient Online Spatio-Temporal Filtering for Video Event Detection, in ECCV Workshop on Video Event Categorization, Tagging and Retrieval towards big data, 2014.

4. Hui Liang, Junsong Yuan and Daniel Thalmann, Improved Hand Pose Estimation via Multimodal Prediction Fusion, in Proc. of the 31^{st} Computer Graphics International, 2014.

5. Hui Liang, Junsong Yuan and Daniel Thalmann, Egocentric Hand Pose Estimation and Distance Recovery in a Single RGB Image, in Proc. of the IEEE Int'l Conf. on Multimedia and Expo, 2015.

6. Hui Liang, Junsong Yuan, Daniel Thalmann and Nadia Magnenat-Thalmann, AR in Hand: Egocentric Palm Pose Tracking and Gesture Recognition for Augmented Reality Applications, in Proc. of the 23^{rd} ACM Int'l Conf. on Multimedia (MM'15), demo paper, 2015.

7. Hui Liang*, Jin Wang*, Qian Sun, Yong-Jin Liu, Junsong Yuan, Jun Luo and Ying He, Barehanded Music: Real-time Hand Interaction for Virtual Piano, in ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D'16), 2016.

Bibliography

[1] N. Pugeault and R. Bowden, Spelling It Out: Real-Time ASL Fingerspelling Recognition, in IEEE Workshop on Consumer Depth Cameras for Computer Vision, pp. 1114-1119 , 2011.

[2] H. Liang, J. Yuan and D. Thalmann, 3D Fingertip and Palm Tracking in Depth Image Sequences, in Proc. ACM Int'l Conf. on Multimedia, pp. 785-788, 2012.

[3] L. Ballan, A. Taneja, J. Gall, L. V. Gool and Marc Pollefeys, Motion Capture of Hands in Action using Discriminative Salient Points, in Proc. European Conf. on Computer Vision, pp. 640-653 , 2012.

[4] Cyberglove 2, http://www.cyberglovesystems.com

[5] A. Aristidou and J. Lasenby, Motion Capture with Constrained Inverse Kinematics for Real-Time Hand Tracking, in Proc. Int'l Symposium on Communications, Control, and Signal Processing, pp. 1-5, 2010.

[6] N. Shimada, Y. Shirai, Y. Kuno and J. Miura, Hand Gesture Estimation and Model Refinement using Monocular Camera-Ambiguity Limitation by Inequality Constraints, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 268-273, 1998.

[7] J. Xu, Y. Wu and A. Katsaggelos, Part-based Initialization for Hand Tracking, in Proc. IEEE Int'l Conf. on Image Processing, pp. 3257-3260, 2010.

[8] C. Xu and L. Cheng, Efficient Hand Pose Estimation from a Single Depth Image, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 3456-3462, 2013.

[9] D. Tang, T. H. Yu and T-K. Kim, Real-time Articulated Hand Pose Estimation using Semi-supervised Transductive Regression Forests, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 3224-3231, 2013.

[10] N. Stefanov, A. Galata, and R. Hubbold, A Real-Time Hand Tracker using Variable-Length Markov Models of Behavior, in Computer Vision and Image Understanding, vol. 108, no. 1-2, pp. 98-115, October 2007.

[11] R. Y. Wang and J. Popovic, Real-Time Hand Tracking with a Color Glove, in ACM Transactions on Graphics, vol. 28, no. 3, Article No. 63, August 2009

[12] S. Sridhar, A. Oulasvirta and C. Theobalt, Interactive Markerless Articulated Hand Motion Tracking Using RGB and Depth Data, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 2456-2463, 2013.

[13] J. Y. Lin, Y. Wu and T. S. Huang, 3D Model-Based Hand Tracking Using Stochastic Direct Search Method, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 693-698, 2004.

[14] B. Stenger, A. Thayananthan, P. H.S. Torr and R. Cipolla, Model-Based Hand Tracking using A Hierarchical Bayesian Filter, in IEEE Trans. Pattern Analysis and Machine Intelligence. vol. 28, no. 9, pp. 1372-1384, September 2006.

[15] H. Guan, R. S. Feris and M. Turk, The Isometric Self-Organizing Map for 3D Hand Pose Estimation, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 263-268, 2006.

[16] H. Guan, J. S. Chang, L. Chen, R. S. Feris and M. Turk, Multi-view Appearance-based 3D Hand Pose Estimation, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 154-154, 2006.

[17] V. Vezhnevets, V. Sazonov and A. Andreeva, A Survey on Pixel-Based Skin Color Detection Techniques, in Proc. GraphiCon, vol. 3, pp. 85-92, 2003.

[18] M. J. Jones and J. M. Rehg, Statistical Color Models with Application to Skin Detection, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 1, pp. 280-280, 1999.

[19] D. Brown, I. Craw and J. Lewthwaite, A SOM Based Approach to Skin Detection with Application in Real Time Systems, in Proc. British Machine Vision Conference, vol. 1, pp. 491-500 , 2001.

[20] J.-C. Terrillon, H. Fukamachi, S. Akamatsu and M. N. Shirazi, Comparative Performance of Different Skin Chrominance Models and Chrominance Spaces for the Automatic Detection of Human Faces in Color Images, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 54-61, 2000.

[21] C. Li and K. M. Kitani, Pixel-level Hand Detection in Ego-Centric Videos, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 3570-3577, 2013.

[22] Y. Wu, G. Hua and T. Yu, Tracking Articulated Body by Dynamic Markov Network, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 1094-1101, 2003.

[23] S. Belongie, J. Malik and J. Puzicha, Shape Matching and Object Recognition using Shape Contexts, in IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pp. 509-522, April 2002.

[24] F. S. Chen, C. M. Fu and C. L. Huang, Hand Gesture Recognition using a Real-Time Tracking Method and Hidden Markov Models, in Image Vision Computing, vol. 21, no. 8, pp.745-758, August 2003.

[25] A. Thayananthan, B. Stenger, P.H.S. Torr and R. Cipolla, Shape Context and Chamfer Matching in Cluttered Scenes, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 1, pp. 127-133, 2003.

[26] V. Athitsos and S. Sclaroff, Estimating 3d Hand Pose from a Cluttered Image, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, pp. II-432, 2003.

[27] I. Oikonomidis, N. Kyriazis and A. A. Argyros, Markerless and Efficient 26-DOF Hand Pose Recovery, in Proc. Asian Conf. on Computer Vision, pp. 744-757, 2010.

[28] I. Oikonomidis, N. Kyriazis and A. A. Argyros, Efficient Model-Based 3d Tracking of Hand Articulations using Kinect, in Proc. British Machine Vision Conference, vol. 1, no. 2, pp. 101.1-101.11, 2011.

[29] Z. Mo and U. Neumann, Real-time Hand Pose Recognition Using Low-Resolution Depth Images, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, pp. 1499-1505, 2006.

[30] C. Wang, Z. Liu and S. C. Chan, Superpixel-Based Hand Gesture Recognition with Kinect Depth Camera, in IEEE Trans. Multimedia, vol. 17, no. 1, pp. 29-39, January 2015.

[31] Z. Ren, J. Yuan, J. Meng and Z. Zhang, Robust Part-based Hand Gesture Recognition using Kinect Sensor, in IEEE Trans. on Multimedia, vol. 15, no. 5, pp. 1110-1120, August 2013.

[32] N. Shimada, K. Kimura and Y. Shirai, Real-time 3-D Hand Posture Estimation based on 2-D Appearance Retrieval Using Monocular Camera, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 23-30, 2001.

[33] E. Ueda, Y. Matsumoto, M. Imai and T. Ogasawara, Hand Pose Estimation using Multi-Viewpoint Silhouette Images, in IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 4, pp. 1989-1996, 2001.

[34] V. Pavlovic, R. Sharma and T. Huang, Visual Interpretation of Hand Gestures for Human-Computer Interaction: a Review, in IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 677-695, July 1997.

[35] D. Chik, Jochen Trumpf and N. N. Schraudolph, Using an Adaptive VAR Model for Motion Prediction in 3D Hand Tracking, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 1-8, 2008.

[36] M. de La Gorce, D. J. Fleet and N. Paragios, Model-Based 3D Hand Pose Estimation from Monocular Video, in IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 33, no. 9, pp. 1793-1805, September 2011.

[37] W.T. Freeman and M. Roth, Orientation Histograms for Hand Gesture Recognition, in International Workshop on Automatic Face and Gesture Recognition, vol. 12, pp. 296-301, 1995.

[38] C. Zhang, X. Yang and Y. Tian, Histogram of 3D Facets: A Characteristic Descriptor for Hand Gesture Recognition, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 1-8, 2013.

[39] V. Ganapathi, C. Plagemann, D. Koller and S. Thrun, Real-Time Human Pose Tracking from Range Data, in Proc. European Conf. on Computer Vision, vol. 4, pp. 738-751, 2012.

[40] C. Qian, X. Sun, Y. Wei, X. Tang and Jian Sun, Realtime and Robust Hand Tracking from Depth, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 1106-1113, 2014.

[41] R. Girshick, J. Shotton, P. Kohli, A. Criminisi and A. Fitzgibbon, Efficient Regression of General-Activity Human Poses from Depth Images, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 415-422, 2011.

[42] H. Liang, J. Yuan and D. Thalmann, Resolving Ambiguous Hand Pose Predictions by Exploiting Part Correlations, in IEEE Trans. Circuits and Systems for Video Technology, vol. 25, no. 7, pp. 1125-1139, July 2015.

[43] B. Dorner, Chasing the Color Glove: Visual Hand Tracking, Mater thesis, 1996.

[44] K. Oka, Y. Sato and H. Koike, Real-Time Fingertip Tracking and Gesture Recognition, in IEEE Computer Graphics and Applications, vol. 22, no. 6, pp. 64-71, 2002.

[45] K. Hsiao, T. Chen and S. Chien, Fast Fingertip Positioning by Combining Particle Filtering with Particle Random Diffusion, Proc. IEEE Int'l Conf. on Multimedia and Expo, pp. 977-980, 2008.

[46] I. Katz, K. Gabayan and H. Aghajan, A Multi-Touch Surface Using Multiple Cameras, in Proc. Int'l Conf. on Advanced concepts for intelligent vision systems, pp. 97-108, 2007.

[47] C.-S. Chua, H. Guan and Yeong-Khing Ho, Model-based 3d Hand Posture Estimation from a Single 2d Image, in Image and Vision Computing, vol. 20, no. 3, pp. 191-202, January 2002.

[48] H. Liang, J. Yuan, D. Thalmann and Z. Zhang, Model-based Hand Pose Estimation via Spatial-temporal Hand Parsing and 3D Fingertip Localization , in the Visual Computer Journal, vol. 29, no. 6-8, pp. 837-848, June 2013.

[49] F. Lathuiliere and J. Y. Herve, Visual Tracking of Hand Posture with Occlusion Handling, in Proc. Int'l Conf. on Pattern Recognition, vol. 3, pp. 1129-1133, 2000.

[50] C. Lien, A Scalable Model-based Hand Posture Analysis System, in Machine Vision and Applications, vol. 16, no. 3, pp. 157-169, May 2005.

[51] K. Dorfmuller-Ulhaas and D. Schmalstieg, Finger Tracking for Interaction in Augmented Environments, in Proc. Augmented Reality, pp. 55-64, 2001.

[52] S. Conseil, S. Bourennane and L. Martin, Three Dimensional Fingertip Tracking in Stereovision, in Proc. Int'l Conf. on Advanced Concepts for Intelligent Vision Systems, pp. 9-16, 2005.

[53] M. Do, T. Asfour and R. Dillmann, Particle Filter-based Fingertip Tracking with Circular Hough Transform Features, in Proc. IAPR Conf. on Machine Vision Applications, vol. 2, pp. 471-474, 2011.

[54] C. Plagemann, V. Ganapathi, D. Koller and S. Thrun, Real-time Identification and Localization of Body Parts from Depth Images, in Proc. IEEE Int'l Conf. on Robotics and Automation, pp. 3108-3113, 2010.

[55] A. Baak, M. Muller, G. Bharaj, H.P. Seidel and C. Theobalt, A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 71-98, 2011.

[56] P. Krejov and R. Bowden, Multi-touchless: Real-Time Fingertip Detection and Tracking Using Geodesic Maxima, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 1-7, 2013.

[57] D. Ramanan, Learning to Parse Images of Articulated Bodies, in Proc. Neural Information Processing Systems Conference, pp. 1129-1136, 2007.

[58] L. Pishchulin, M. Andriluka, P. Gehler and B. Schiele, Strong Appearance and Expressive Spatial Models for Human Pose Estimation, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 3487-3494, 2013.

[59] N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 1, pp. 886-893, 2005.

[60] T. S. Huang and V. Pavlovic, Hand Gesture Modeling, Analysis, and Synthesis, in Int'l Workshop on Automatic Face and Gesture Recognition, pp. 73-79, 1995.

[61] Z. Ren, J. Yuan, C. Li and W. Liu, Minimum Near-Convex Decomposition for Robust Shape Representation, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 303-310, 2011.

[62] P. F. Felzenszwalb and D. P. Huttenlocher, Pictorial Structures for Object Recognition, in International Journal of Computer Vision, vol. 61, no. 1, pp. 55-79, January 2005.

[63] Y. Yao and Y. Fu, Real-time Hand Pose Estimation from RGB-D Sensor, in Proc. IEEE Int'l Conf. on Multimedia and Expo, pp. 705-710, 2012.

[64] C. Keskin, F. Kirac, Y. E. Kara and L. Akarun, Real-time Hand Pose Estimation using Depth Sensors, in Proc. IEEE Int'l Conf. on Computer Vision Workshops, pp. 1228-1234, 2011.

[65] C. Keskin, F. Kirac, Y. E. Kara and L. Akarun, Hand Pose Estimation and Hand Shape Classification Using Multi-layered Randomized Decision Forests, in Proc. European Conf. on Computer Vision, pp. 852-863, 2012.

[66] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman and A. Blake, Real-Time Human Pose Recognition in Parts from Single Depth Images, in IEEE Conf. on Computer Vision and Pattern Recognition, pp. 1297-1304, 2011.

[67] L. Breiman, Random forests. Mach. Learning, vol. 45, no. 1, pp. 5-32, October 2001.

[68] J. L. Raheja, A. Chaudhary and K. Singal, Tracking of Fingertips and Centers of Palm using Kinect, in Proc. the 3rd Int'l Conf. on Computational Intelligence, Modelling and Simulation, pp. 248-252, 2011.

[69] M. Isard and A. Blake, CONDENSATION - Conditional Density Propagation for Visual Tracking, in International Journal of Computer Vision, vol. 29, no. 1, pp. 5-28, August 1998.

[70] G. Panin, S. Klose and A. Knoll, Real-time Articulated Hand Detection and Pose Estimation, in Proc. International Symposium on Advances in Visual Computing, pp. 1131-1140, 2009.

[71] M. Kolsch and M. Turk, Robust Hand Detection, in Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition, pp. 614-619, 2004.

[72] A. Hernandez-Vela, N. Zlateva, A. Marinov, M. Reyes, P. Radeva, D. Dimov and S. Escalera, Graph Cuts Optimization for Multi-Limb Human Segmentation in Depth Maps, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 726-732, 2012.

[73] H. Liang, J. Yuan and D. Thalmann, Parsing the Hand in Depth Images, in IEEE Trans. Multimedia, vo. 16, no. 5, pp. 1241-1253, August 2014.

[74] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle and X. Twombly, A Review on Vision-Based Full DOF Hand Motion Estimation, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 75-75, 2005.

[75] J. Lin, W. Ying and T. S. Huang, Modeling the Constraints of Human Hand Motion, in Proc. the Workshop on Human Motion, pp. 121-126, 2000.

[76] J. P. Lewis, M. Cordner and N. Fong, Pose Space Deformation: a Unified Approach to Shape Interpolation and Skeleton-Driven Deformation, in Proc. Annual Conf. on Computer Graphics and Interactive Techniques, pp. 165-172, 2000.

[77] J. Heikkila, A Four-step Camera Calibration Procedure with Implicit Image Correction, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 1106-1112, 1997.

[78] R. Kindermann and J. L. Snell, Markov Random Fields and Their Applications, the American Mathematical Society, 1980.

[79] M. Fauvel, J. Chanussot and J. A. Benediktsson, SVM- and MRF-Based Method for Accurate Classification of Hyperspectral Images, in IEEE Geoscience and Remote Sensing Letters, vol. 7, no. 4, pp. 736-740, October 2010.

[80] A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Susstrunk, SLIC Superpixels Compared to State-of-the-Art Superpixel Methods, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pp. 2274-2282, November 2012.

[81] X. Wang, A New Localized Superpixel Markov Random Field for Image Segmentation, in Proc. IEEE Int'l Conf. on Multimedia and Expo, pp. 642-645, 2009.

[82] J. Tighe and S. Lazebnik, SuperParsing: Scalable Nonparametric Image Parsing with Superpixels, in Proc. European Conf. on Computer Vision, pp. 352-365, 2010.

[83] J. Besag, On the Statistical Analysis of Dirty Pictures, Journal of the Royal Statistical Society, ser. B, vol. 48, no. 3, pp. 259-302, 1986.

[84] W. Zhao, J. Chai and Y. Xu, Combining Marker-based Mocap and RGB-D Camera for Acquiring High-fidelity Hand Motion Data, in ACM SIGGRAPH Symposium on Computer Animation, pp. 33-42, 2012.

[85] F. Kirac, Y. E. Kara and L. Akarun, Hierarchically Constrained 3d Hand Pose Estimation using Regression Forests from Single Frame Depth Data, in Pattern Recognition Letters, vol. 50, no. 1, pp. 91-100, December 2014.

[86] K. C. Sim, Discriminative Product-of-Expert Acoustic Mapping for Cross-lingual Phone Recognition, in IEEE Workshop on Automatic Speech Recognition & Understanding, pp. 546-551, 2009.

[87] A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum Likelihood from Incomplete Data Via The EM Algorithm, in Journal of the Royal Statistical Society, vol. 39, no. 1, pp. 1-38, 1977.

[88] M. Dantone, J. Gall, C. Leistner and L. V. Gool, Human Pose Estimation using Body Parts Dependent Joint Regressors, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 3041-3048, 2013.

[89] D. Comaniciu and P. Meer, Mean shift: A Robust Approach toward Feature Space Analysis, in IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 603-619, May 2002.

[90] Softkinetic Depthsense Camera and SDK, http://www.softkinetic.com

[91] S. Pellegrini, K. Schindler and D. Nardi, A Generalisation of the ICP Algorithm for Articulated Bodies, in Proc. British Machine Vision Conference, pp. 87.1-87.10, 2008.

[92] M. Schroder, J. Maycock, H. Ritter and M. Botsch, Real-Time Hand Tracking using Synergistic Inverse Kinematics, in Proc. Int'l Conf' on Robotics and Automation, pp. 5447-5454, 2014.

[93] Y. Wu and T. S. Huang, Capturing Articulated Human Hand Motion: A Divide-and-Conquer Approach, in Proc. Int'l Conf. on Computer Vision, vol. 1, pp. 606-611, 1999.

[94] B. Stenger, P. R. S. Mendonca and R. Cipolla, Model-Based 3D Tracking of an Articulated Hand, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol.2, pp. II-310-II-315, 2001.

[95] H. Zhou and T. S. Huang, Tracking Articulated Hand Motion with Eigen Dynamics Analysis, in Proc. Int'l Conf. on Computer Vision, pp. 1102-1109, 2003.

[96] V. Pavlovic, J. M. Rehg and J. MacCormick, Learning switching linear models of human motion, in Proc. Neural Information Processing Systems, pp. 981-987, 2000.

[97] A. Thayananthan, R. Navaratnam, B. Stenger, P.H.S. Torr and R. Cipolla, Pose Estimation and Tracking Using Multivariate Regression, in Pattern Recognition Letters, vol. 29, no. 9, pp. 1302-1310, July 2008.

[98] R. Y. Wang, S. Paris, J. Popovic, 6D Hands: Markerless Hand Tracking for Computer Aided Design, in Proc. ACM Symposium on User Interface Software and Technology, pp. 549-558, 2011.

[99] J. Romero, H. Kjellstrom and D. Kragic, Monocular Real-Time 3D Articulated Hand Pose Estimation, in Proc. Int'l Conf. on Humanoid Robots, pp. 87-92, 2009.

[100] D. Tang, H.J. Chang, A. Tejani, T.K. Kim, Latent Regression Forest: Structured Estimation of 3D Articulated Hand Posture, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 3786-3793, 2014.

[101] K. S. Arun, T. S. Huang and S. D. Blostein, Least-Squares Fitting of Two 3D Point Sets, in IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-9, no. 5, pp. 698-700, September 1987.

[102] Microsoft Corp. Redmond WA. Kinect for Xbox 360.

[103] C. Theobalt, I. Albrecht, J. Haber, M. Magnor and H. Seidel, Pitching a Baseball - Tracking High-Speed Motion with Multi-Exposure Images, in ACM Transactions on Graphics, vol. 23, no. 3, pp. 540-547, August 2004.

[104] R. Rosales and S. Sclaroff, Inferring Body Pose without Tracking Body Parts, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, pp. 721-727, 2000.

[105] M. Potamias and V. Athitsos, Nearest Neighbor Search Methods for Hand-Shape Recognition, in Proc. Int'l Conf. on Pervasive Technologies Related to Assistive Environments, Article No. 30, 2008.

[106] X. Lv, T. Han, Z. Liu and Z. He, Randomized Support Vector Forest, in Proc. British Machine Vision Conference, 2014.

[107] M. Sun, P. Kohli and J. Shotton, Conditional Regression Forests for Human Pose Estimation, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 3394-3401, 2012.

[108] R. Caruana, N. Karampatziakis and A. Yessenalina, An Empirical Evaluation of Supervised Learning in High Dimensions, in Proc. Int'l Conf. on Machine Learning, pp. 96-103, 2008.

[109] X. Wei, P. Zhang and J. Chai, Accurate Real-time Full-body Motion Capture Using a Single Depth Camera, in ACM Transactions on Graphics, vol. 31, no. 6, Article No. 188, November 2012.

[110] Y. Zhao, Z. Song and X. Wu, Hand Detection using Multi-Resolution HoG Features, in Proc. IEEE Int'l Conf. on Robotics and Biomimetics, pp. 1715-1720, 2012.

[111] X. Liu, M. Song, D. Tao, Z. Liu, L. Zhang, C. Chen and J. Bu, Semi-supervised Node Splitting for Random Forest Construction, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 492-499, 2013.

[112] H. Liang, J. Yuan and D. Thalmann, Hand Pose Estimation by Combining Fingertip Tracking and Articulated ICP, in ACM SIGGRAPH Int'l Conf. on Virtual-Reality Continuum and its Applications in Industry, pp. 87-90, 2012.

[113] L. C. T. Wang and C. C. Chen, A Combined Optimization Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators, in IEEE Trans. Robotics and Automation, vol. 7, no. 4, pp. 489-499, August 1991.

[114] P. J. Besl and N. D. McKay, A Method for Registration of 3-D Shapes, in IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256, February 1992.

[115] K. Moon and V. Pavlovic, Monocular 3d Human Motion Tracking using Dynamic Probabilistic Latent Semantic Analysis, in Canadian Conf. on Computer and Robot Vision, pp. 155-162, 2008.

[116] C. Stoll, N. Hasler, J. Gall, H. P. Seidel and C. Theobalt, Fast Articulated Motion Tracking using A Sums Of Gaussians Body Model, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 951-958, 2011.

[117] S. Sridhar, F. Mueller, A. Oulasvirta, C. Theobalt, Fast and Robust Hand Tracking Using Detection-Guided Optimization, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 3213-3221, 2015.

[118] G. Hillebrand, M. Bauer, K. Achatz, G. Klinker and A. Oferl, Inverse Kinematic Infrared Optical Finger Tracking, in International Conference on Humans and Computers, 2006.

[119] H. Breu, J. Gil, D. Kirkpatrick and M. Werman, Linear Time Euclidean Distance Transform Algorithms, in IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, no. 5, pp. 529-533, May, 1995.

[120] I.T. Jolliffe, Principal Component Analysis, Springer, 2002.

[121] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, MIT press, 2001.

[122] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, Real-time Continuous Pose Recovery of Human Hands using Convolutional Networks, in ACM Transactions on Graphics, vol. 33, no. 5, Article No. 169, 2014.

[123] A. Tang, K. Lu, Y. Wang, J. Huang and H. Li, A Real-Time Hand Posture Recognition System Using Deep Neural Networks, in ACM Transactions on Intelligent Systems and Technology, vo. 6, no. 2, Article No. 21, May 2015.

[124] I. Oikonomidis, N. Kyriazis and A. Argyros, Full DoF Tracking of a Hand Interacting with an Object by Modeling Occlusions and Physical Constraints, in Proc. IEEE Int'l Conf. on Computer Vision, pp. 2088-2095, 2011.

[125] J. Romero, K. Hedvig, C. H. Ek and D. Kragic, Non-Parametric Hand Pose Estimation with Object Context, in Image and Vision Computing, vol. 31, no. 8, pp. 555-564, August 2013.