Noname manuscript No.
(will be inserted by the editor)

Estimate Hand Poses Efficiently from Single Depth Images

Chi Xu · Ashwin Nanjappa · Xiaowei Zhang · Li Cheng

Received: date / Accepted: date

Abstract This paper aims to tackle the practically very challenging problem of *efficient* and *accurate* hand pose estimation from single depth images. A dedicated two-step regression forest pipeline is proposed: Given an input hand depth image, step one involves mainly estimation of 3D location and in-plane rotation of the hand using a pixel-wise regression forest. This is utilized in step two which delivers final hand estimation by a similar regression forest model based on the entire hand image patch. Moreover, our estimation is guided by internally executing a 3D hand kinematic chain model. For an unseen test image, the kinematic model parameters are estimated by a proposed dynamically weighted scheme. As a combined effect of these proposed building blocks, our approach is able to deliver more precise estimation of hand poses. In practice, our approach works at 15.6 frame-per-second (FPS) on an average laptop when implemented in CPU, which is further sped-up to 67.2 FPS when running on GPU. In addition, we introduce and make publicly available a data-glove annotated depth image dataset covering various hand shapes and gestures, which enables us conducting quantitative analyses on real-world hand images. The effectiveness of our approach is verified empirically on both synthetic and the annotated real-world datasets for hand pose estimation, as well as related applications including part-based labeling and gesture classification. In addition to empirical studies, the consistency property of our approach is also theoretically analyzed.

Keywords Hand Pose Estimation, Depth Images, GPU acceleration, Regression Forests, Consistency Analysis, Annotated Hand Image Dataset

Chi Xu, Ashwin Nanjappa, Xiaowei Zhang
BII, A*STAR, Singapore
E-mail: xuchi, ashwinn, zhangxw@bii.a-star.edu.sg

Li Cheng
Tel.: +65-64788358
Fax: +65-64789047
BII, A*STAR, Singapore & SoC, NUS, Singapore
E-mail: chengli@bii.a-star.edu.sg

1 Introduction

Vision-based hand interpretation plays important roles in diverse applications including humanoid animation [34, 39], robotic control [18], and human-computer interaction [20, 27], among others. In its core lies the nevertheless challenging problem of 3D hand pose estimation [14, 25], owing mostly to the complex and dexterous nature of hand articulations [18]. Facilitated by the emerging commodity-level depth cameras [2, 3], recent efforts such as [24, 35, 40, 41] have led to noticeable progress in the field. The problem is however still far from being satisfactorily solved: For example, not much quantitative analysis has been conducted on annotated real-world 3D datasets, partly due to the practical difficulty of setting up such testbeds. This however imposes significant restrictions on the evaluation of existing efforts, which are often either visually judged based on a number of real depth images, or quantitatively verified on synthetic images only as the ground-truths are naturally known. As each work utilizes its own set of images, their results are not entirely comparable. These inevitably raise the concerns of progress evaluation and reproducibility.

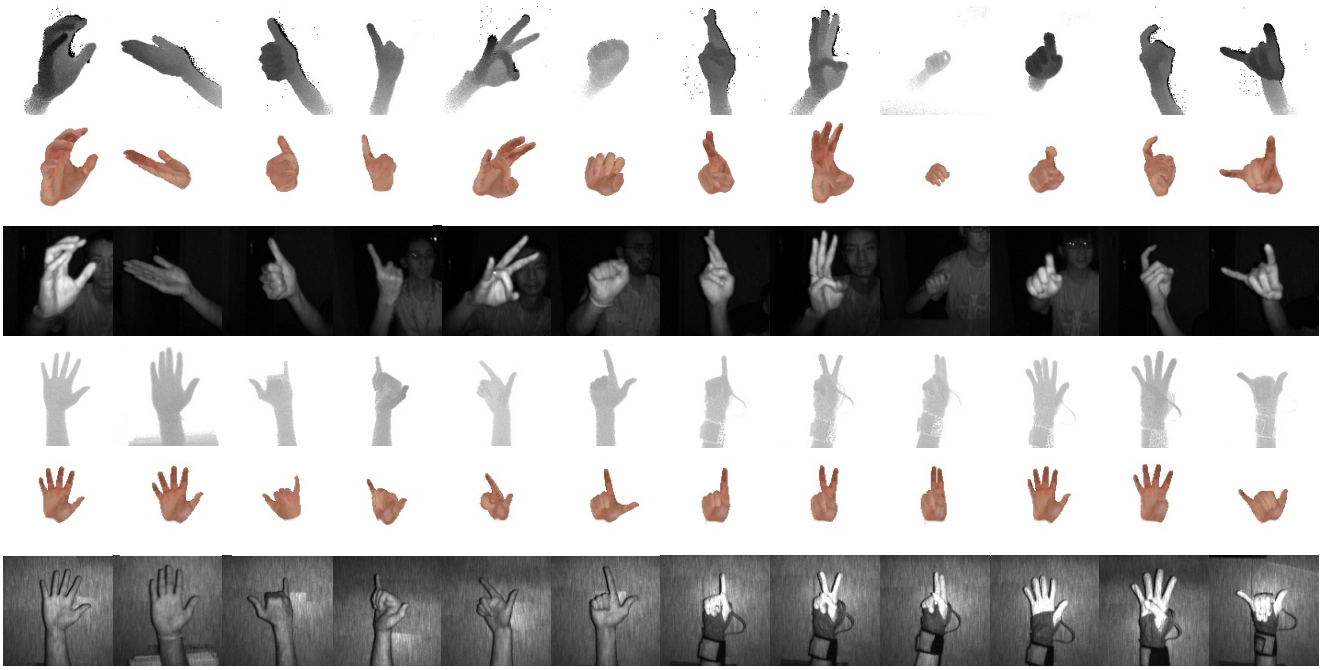

Fig. 1: Exemplar hand pose estimation results of our system under various scenarios including w/ and w/o gloves, and different camera set-ups (top-down view vs. frontal view). Each image involves three rows: The first row presents the input hand depth image, and the pose estimation result is displayed in the second row; To facilitate the interpretation of our results, the third row also provides its corresponding amplitude image. Note the amplitude image is used as normal gray-scale image here only for reference purpose. In the amplitude image, brighter pixels denote higher confidence for closer objects and vice versa.

In this paper[1], we tackle the problem of efficient hand pose estimation from single depth images. The main contributions of our work are four-fold:

- For an unseen test image, a dynamically weighted scheme is proposed to regress our hand model parameters based on a two-step pipeline, which is empirically shown to lead to significantly reduced errors comparing to the state-of-the-arts. As presented in Fig. 1 as well the supplementary video, our system estimates hand poses from single images and with 3D orientations. This also enables our system to work with a mobile depth camera.
- We provide an extensive, data-glove annotated benchmark of depth images for general hand pose estimation. The benchmark dataset, together with the ground-truths and the evaluation metric implementation, have been made publicly available. This is the first benchmark of such kind to our knowledge, and we wish it can provide an option for researchers in the field to compare performance on the same ground.
- To maintain efficiency and to offset the CPU footprint, the most time-consuming components of our approach have also been identified and accelerated by GPU implementations, which gains us a further 5-fold overall speed-up. These enable us to deliver a practical hand pose estimation system that works efficiently, at about 15.6 frame-per-second (FPS) on an average laptop, and 67.2 FPS when having access to a mid-range GPU.
- The reliance on synthetic training examples naturally brings up the consistency question when infinitely many examples are potentially available for training. Our paper makes first such attempt to propose a regression forest-based hand pose system that is theoretically motivated. To this end we are able to provide consistency analysis on a simple variant of our learning system. Although the complete analysis is still open, we believe this is a necessary and important step toward full comprehension of the random forests theory that has been working so well on a number of applications in practice.

Finally, the competitive performance is demonstrated during empirical experiments on both synthetic and real datasets. Several visual examples are demonstrated in Fig. 1, where each image involves three rows: The first row presents the input hand depth image and its pose estimate in the second row, while the corresponding gray-scale amplitude image (also referred to as the

[1] A project webpage can be found at `http://web.bii.a-star.edu.sg/~xuchi/handengine.htm`, which contains supplementary information of this paper such as the demo video.

confidence map in literature) is shown in the third row to facilitate the interpretation of our results. For a time-of-flight (TOF) depth camera such as Softkinetic [3], each pixel of the amplitude stores the returning Infrared (IR) intensity from the modulated IR light source, and can be regarded as the relative confidence in its depth measurement [22]. In our approach, it is only used for filtering away noisy depth pixel observations during preprocessing. Although our empirical results in this paper are primarily based on Softkinetic TOF camera, we would like to point out that our approach works with generic depth cameras including TOF cameras as well as the structured illumination depth cameras such as Kinect [2], where image denoising strategy of [40] is adopted during preprocessing. Fig. 2 presents a flowchart outlining our two-step approach.

1.1 Related work

An earlier version of our work appears in [40] that deals with the problem of depth image-based hand pose estimation. There are a number of differences of our work here comparing to that of [40]: First, a simple two-step pipeline is utilized in our approach, in contrast to a more complicated approach in [40] containing three steps. Second, in this work we attempt to consider random forest models that can be analyzed theoretically, while the random forest models in [40] are not able to be studied theoretically. Third, there are also many other differences: The kinematic model parameters are estimated by a dynamically weighted scheme that leads to a significant error reduction in empirical evaluations. The information gains and split criteria, the usage of whole hand image patch rather than individual pixels, as well as the DOT features to be detailed later are also quite different. Meanwhile, various related regression forest models have been investigated recently: In [15], the head pose has 6 degree-of-freedom (DoF), which is divided into 2 parts: 3D translation and 3D orientation. In each leaf node, the distribution is approximated by a 3D Gaussian. In [16], the Hough forest model is instead utilized to represent the underlining distributions as voting with a set of 3D vectors. In [32], a *fixed* weight is assigned to each of the 3D voting vectors during training, and the experimental results suggest that the weight plays a crucial role in body pose estimation. Our scheme of *dynamical* weights can be regarded as a further extension of this idea to allow adaptive weight estimation at test-run that is dedicated to the current test example. A binary latent tree model is used in [35] to guide the searching process of 3D locations of hand joints. For the related problem of video-based 3D hand tracking, a user-specific modeling method is proposed by [36], while [28] adopts an evolutionary optimization method to capture hand and object interactions.

Leap Motion [4] is a commercial system designed for close-range (within about 50 cm in depth) hand pose estimation. As a closed system based proprietary hardware, its inner working mechanism remains undisclosed. Our observation is that it is not well tolerant to self-occlusions of finger tips. In contrast, our system works beyond half a meter in depth, and works well when some of the finger-tips are occluded as it does not rely on detecting finger tips.

Additionally, instead of directly estimating 3D locations of finger joints from the depth image as e.g. that of [17], our model predicts the parameters of a predefined hand kinematic chain, which is further utilized to build the 3D hand. This is mainly due to the fact that compared to 3D location of joints, kinematic chain is a global representation and is more tolerant to self-occlusion, a scenario often encountered in our hand pose estimation context. Second, for human pose estimation, once the body location is known (*i.e.,* the torso is fixed), the limbs and the head can be roughly considered as independent sub-chains: *e.g.* a change from left hand will not affect the other parts significantly. In contrast, motions of the five fingers and the palm are tightly correlated.

For the related problem of optical motion capture, depth cameras have been utilized either on their own [6], or together with existing marker-based system [33, 42] for markerless optical motion capture. While being able to produce more precise results, they typically rely on more than one cameras, and operate in an off-line fashion. In term of annotated datasets of hand images, existing efforts are typically annotated manually with either part-based labels (e.g. [35]) or finger tips [33]. However these annotations do not explicitly offer 3D information of the skeletal joints. [37] instead engages a human annotator to annotate 2D locations of joints, and aggregates them to infer the 3D hand joint locations. This dataset unfortunately does not provide depth image input, also one possible concern is its annotation might not be fully objective.

In what follows, we start by giving an overall account of the regression forest models that are the core modules in our proposed learning system.

2 Our Theoretically Motivated Regression Forest Models

As will become clear in later sections, the training of our regression forests relies on large quantity of synthetic examples. It is thus of central interest to pro-

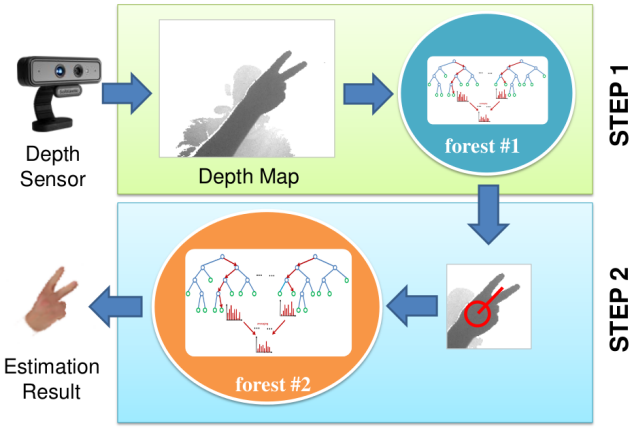

Fig. 2: Flowchart of our two-step approach: Given an input hand depth image, step one involves mainly estimation of 3D location and in-plane rotation of the hand using a pixel-wise regression forest. This is utilized in step two which delivers final hand estimation by a similar regression forest model based on the entire hand image patch. See text for details.

vide *consistency* analysis to characterize their asymptotic behaviors, which concerns the convergence of the estimate to an optimal estimate as the sample size goes to infinity. Most existing papers [7, 8, 10, 13] on the consistency of regression forests focus on stylized and simplified algorithms. The unpublished manuscript of Breiman [10] suggests a simplified version of random forests and provides a heuristic analysis of its consistency. This model is further analyzed in [7] where, besides consistency, the author also shows that the rate of convergence depends only on the number of strong features. An important work on the consistency of random forests for classification is [8] which provides consistency theorems for various versions of random forests and other randomized ensemble classifiers. Despite these efforts, there is still a noticeable gap between theory and practice of regression forests learning systems. This is particularly true for the pose estimation systems that have make tremendous progresses during the past few years in looking at human full-body, head, and hand, where random forests have been very successful. On the other hand, little theoretical analysis has been provided for the learning systems underpinning these empirical successes.

Different from most existing practical random forest models, the random forest model considered in our approach is theoretically motivated, which is inspired by existing theoretical works [7, 8, 10] and in particular [13]. The theoretical analysis of the resulting random forest model closely follows that of [13]. Meanwhile our proposed random forest model is sufficiently sophisticated to be practically capable of addressing real world problems. Our models and its variants are specifically applied to the real problem of hand pose estimation with competitive empirical performance. Meanwhile, it is worth noting that the proposed models are generic and can work with problems beyond hand pose estimation.

In what follows, we introduce our generic regression forest model in term of training data partition, split criteria during tree constructions, prediction, as well as its variants. Its asymptotic *consistency* analysis is also offered.

2.1 Training Data: A Partition into Structure Data and Estimation Data

Formally, let $\overline{X}$ denote a $[0,1]^d$-valued random variable and $\overline{Y}$ denote a $\mathbb{R}^q$-valued vector of random variables, where d is the dimension of normalized feature space, and q is the dimension of the label space. Denote $\bar{\mathbf{x}}$ (or $\bar{\mathbf{y}}$) a realization of the random variable $\overline{X}$ (or $\overline{Y}$). A training example can be defined as an (instance, label) pair, $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. Therefore, the set of n training examples is represented as $D_n = \left\{(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)\right\}_{i=1}^n$. Inspired by [13], during tree constructions, we partition D_n randomly into two parts: *structure data* U_n and *estimation data* E_n by randomly selecting $\lfloor \frac{n}{2} \rfloor$ examples as structure data and the rest as estimation data. Examples in structure data are used to determine the tests used in split nodes (i.e. internal nodes of the trees), and examples in estimation data are retained in each leaf node of the tree for making prediction at test phase. This way, once the partition of the training sample is provided, the randomness in the construction of the tree remains independent of the estimation data, which is necessary to ensure consistency in the follow-up theoretical analysis.

2.2 The Split Criteria

The performance of the regression forest models is known to be crucially determined by decision tree constructions and particularly the split criteria, which are the focuses here.

In the regression forests, each decision tree is independently constructed. The tree construction process can be equivalently interpreted as a successive partition of the feature space, $[0,1]^d$, with axis-aligned split functions. That is, starting from the root node of a decision tree which encloses the entire feature space, each tree node corresponds to a specific rectangular hypercube with monotone decreasing volumes as we visit node deeper into the tree. Finally, the union of the hy-

percubes associated with the set of leaf nodes forms a complete partition of the feature space.

Similar to existing regression forests in literature including [13, 15, 31], at a split node, we randomly select a relatively small set of s distinct features $\Phi := \{\phi_i\}_{i=1}^{s}$ from the d-dimensional space as candidate features (i.e. entries of the feature vector). s is obtained via $s \sim 1 + \text{Binomial}(d-1, p)$, where Binomial denotes the binomial distribution, and $p > 0$ a predefined probability. Denote $t \in \mathbb{R}$ a threshold. At every candidate feature dimension, we first randomly select M structure data in this node, where M is the smaller value between the number of structure data in this node and a user-specified integer m_0 (m_0 is independent of the training size), then project them onto the candidate feature dimension and uniformly select a set of candidate thresholds $\mathcal{T}$ over the projections of the M chosen examples. The best test (ϕ^*, t^*) is chosen from these s features and accompanying thresholds by maximizing the information gain that is to be defined next. This procedure is then repeated until there are $\lceil \log_2 L_n \rceil$ levels in the tree *or* if further splitting of a node would result in fewer than k_n estimation examples.

The above-mentioned split test is obtained by

$$(\phi^*, t^*) = \arg \max_{\phi \in \Phi, t \in \mathcal{T}} \mathcal{I}(\phi, t).$$

Here the information gain $\mathcal{I}(\phi, t)$ is defined as:

$$\mathcal{I}(\phi, t) = H(S) - \left(\frac{|S_l|}{|S|} H(S_l) + \frac{|S_r|}{|S|} H(S_r) \right), \tag{1}$$

where $|\cdot|$ counts the set size, S denotes the set of structure examples arriving at current node, which is further split into two subsets S_l and S_r according to the test (ϕ, t). Now, consider to model the parameter vector of current internal node as following a q-variate Gaussian distribution. The entropy of a q-variate Gaussian is defined as

$$H(S) = \frac{1}{2} \ln \Big(\det\big(\Sigma(S)\big) \Big) + c, \tag{2}$$

with the constant $c := \frac{q}{2}\big(1 + \ln(2\pi)\big)$, $\Sigma(\cdot)$ the associated $q \times q$ covariance matrix, and $\det(\cdot)$ the matrix determinant. For a point set S in the q-dimensional space, the determinant of its covariance matrix characterizes the volume of the Gaussian ellipsoid. So a smaller entropy $H(S)$ suggests a more compact cluster. The first term of (1) is fixed, so maximizing the information gain amounts to minimizing the sum of the entropies from its children branches – which can be interpreted as the pursuit of more compact clusters in the course of tree constructions.

2.3 Prediction

During the training stage, regression trees in the forests have been constructed following the above mentioned procedure. At the prediction stage, one concerns how to deliver a prediction for a new query instance $\bar{\mathbf{x}}$. We start with a few more notations. We use random variable $\Psi = (J, G)$ to denote the randomness presented in a regression forest, which consists of random variable J denoting the randomness in the partition of training data D_n, and random variable G denoting the randomness in the construction of trees. Let n_s and n_e denote the number of structure examples and estimation examples in each tree, respectively. From the construction of trees, we have $n_s = \lfloor \frac{n}{2} \rfloor \leq \frac{n}{2}$ and $n_e = n - n_s \geq \frac{n}{2}$. Besides, $A_n(\bar{\mathbf{x}}, \Psi)$ stands for the leaf node containing $\bar{\mathbf{x}}$ and $N_{n_s}(\bar{\mathbf{x}}, \Psi)$, $N_{n_e}(\bar{\mathbf{x}}, \Psi)$ represent the number of structure examples and estimation examples in $A_n(\bar{\mathbf{x}}, \Psi)$, respectively. Each tree thus makes prediction by

$$r_n(\bar{\mathbf{x}}, \Psi) = \frac{1}{N_{n_e}(\bar{\mathbf{x}}, \Psi)} \sum_{i=1}^{n} \bar{\mathbf{y}}_i \, \mathbb{1}_{\left\{\bar{\mathbf{y}}_i \in A_n(\bar{\mathbf{x}}, \Psi), \bar{\mathbf{y}}_i \in E_n\right\}} \tag{3}$$

for a query instance $\bar{\mathbf{x}}$, where $\mathbb{1}$ is the indicator function. Suppose the regression forest is a collection of Z trees $\left\{r_n(\bar{\mathbf{x}}, \Psi_j)\right\}_{j=1}^{Z}$, with $\{\Psi_j\}_{j=1}^{Z}$ being identically and independently distributed (i.i.d.) random variables of Ψ. The prediction of the forest is simply given by

$$r_n^{(Z)}(\bar{\mathbf{x}}) = \frac{1}{Z} \sum_{j=1}^{Z} r_n(\bar{\mathbf{x}}, \Psi_j). \tag{4}$$

Up to now we have introduced our random forest model using empirical mean, which is also referred to as *Baseline-M*. Before proceeding to its asymptotic analysis in Section 2.4, we would like to further discuss two more sophisticated variants that possess the same training stage as illustrated above and differ only at the prediction stage: One is the forest model using static weights, and is called *Baseline-S*; The second variant is a regression forest model using dynamical weights at the leaf nodes, and is termed *DHand*.

2.3.1 Baseline-S *vs.* DHand*: Static vs. Dynamical Weights at the Leaf Nodes*

Instead of making prediction with the empirical average as in *Baseline-M*, we consider to deliver final prediction by mode-seeking of the votes as in the typical Hough forests of [16]. More specifically, let l denote the current leaf node, and let $i \in \{1, \ldots, k_l\}$ indexes over the training examples of leaf node l. These examples are subsequently included as vote vectors in the voting space.

Now, consider a more general scenario where each of the training examples has its own weight. Let $\mathbf{z}_{li}$ represent the parameter vector of a particular training example i of leaf node l, together with $w_{li} > 0$ as its corresponding weight. The set of weighted training examples at leaf node l can thus be defined as $\mathcal{V}_l = \{(\mathbf{z}_{li}, w_{li})\}_{i=1}^{k_l}$. Note this empirical vote set defines a point set or equivalently, an empirical distribution. In existing literature such as [16], $w_{li} = 1$ for any training example i and any leaf node l. In other words, the empirical distribution $\mathcal{V}_l$ is determined during tree constructions in training stage, and remains unchanged during prediction stage. This is referred to as the statically weighted scheme or *Baseline-S*. Rather, we consider a dynamically weighted scheme (i.e. *DHand*) where each of the weights, w_{li}, can be decided at runtime. This is inspired by the observation that the typical distribution of $\mathcal{V}_l$ tends to be highly multi-modal. It is therefore crucial to assign each instance $\mathbf{z}_{li}$ a weight w_{li} that properly reflects its influence on the test instance.

More specifically, for a specific test hand image patch I_t, the distribution $\mathcal{V}_l$ is allowed to be adapted by weights to capture its similarity w.r.t. each training patch in the leaf node l, I_{li}, as $w_{li} = \mathcal{S}_l(I_t, I_{li})$, where $\mathcal{S}_l$ denotes a similarity function between the pair of test and training instances. Ideally, the similarity function $\mathcal{S}_l$ should be inversely proportional to the distance of the two kinematic models, $\|\mathbf{z}_t - \mathbf{z}_{li}\|$, which is unfortunately impractical to compute as $\mathbf{z}_t$ is exactly the quantity we would like to estimate and is thus unknown. On the other side, it can be approximated by measuring the similarity between the two corresponding hand patches, I_t and I_{li}. Here the DOT feature matching [23] is adopted to provide such a measure between the two patches known as $C(I_t, I_{li})$, as to be discussed next. $\mathcal{S}_l$ is thus computed as

$$\mathcal{S}_l(I_t, I_{li}) = \frac{c_s}{c_s + \Big(C^* - C(I_t, I_{li})\Big)},$$

where $c_s = 5$ is a constant, and C^* denotes the maximum similarity score over all leaf nodes. From the empirical distribution $\mathcal{V}_l$, the final output is obtained by applying the weighted mean-shift method [11] to find the local modes in the density in a manner similar to [32]. Briefly, the mean-shift iteration starts with an initial estimate $\mathbf{z}$. A kernel function $K(\|\mathbf{z}_{li} - \mathbf{z}\|)$ is used to calculate the weighted mean of the points around $\mathbf{z}$. Define

$$m(\mathbf{z}) = \frac{\sum_l \sum_i w_{li} K\big(\|\mathbf{z}_{li} - \mathbf{z}\|\big)\mathbf{z}_{li}}{\sum_l \sum_i w_{li} K\big(\|\mathbf{z}_{li} - \mathbf{z}\|\big)} \tag{5}$$

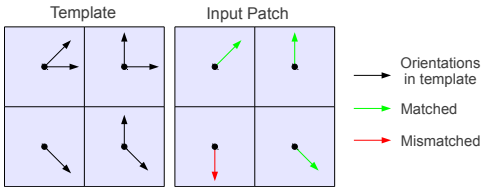

Fig. 3: An illustration of the DOT feature matching [23].

as the mean-shift function. The mean-shift algorithm works by setting $\mathbf{z} \leftarrow m(\mathbf{z})$ and repeat the estimation until $m(\mathbf{z})$ converges.

DOT [23]: As illustrated in Fig. 3, the DOT feature is used in our context to compute a similarity score $C(I, I_T)$ between an input (I) and a reference (I_T) hand patches. DOT works by dividing the image patch into a series of blocks of size 8×8 pixels, where each block is encoded using the pixel gradient information as follows: Denote as η the orientation of the gradient on a pixel, with its range $[0, 2\pi)$ quantized into n_η bins, $\{0, 1, \cdots, n_\eta - 1\}$. We empirically set $n_\eta = 8$, the span of each bin is thus 45°. This way, η can be encoded as a vector $\mathbf{o}$ of length n_η, by assigning 1 to the bin it resides and 0 otherwise. We set $\mathbf{o}$ to zero vector if there is no dominant orientation at the pixel. Now, consider each block of the input patch, its local dominant orientation η^* is simply defined as the maximum gradient within this block, which gives the corresponding vector $\mathbf{o}^*$. Meanwhile for each block in a template patch, to improve the robustness of DOT matching, we utilize a list of local dominant orientations $\{\eta_1^*, \eta_2^*, \cdots, \eta_r^*\}$, each corresponds to the template under a slight translation. Each entry of the list is mapped to the aforementioned orientation vector, and by applying bitwise OR operations successively to these vectors, they are merged into the vector $\mathbf{o}_T^*$.

The similarity $C(I, I_T)$ is then measured block-wise as the number of matched blocks between I and I_T: For each block, if the local dominant orientation $\mathbf{o}^*$ of I belongs to the orientation list of I_T, i.e. $\mathbf{o}^* \,\&\, \mathbf{o}_T^* \neq \mathbf{0}$ or $\mathbf{o}^* = \mathbf{o}_T^* = \mathbf{0}$, this block is deemed as a matched. Here & means bitwise AND operation. Note DOT features are computed from raw input image data and are used as sufficient statistics to fully represent the input hand image patch, and are thus stored in the leaf nodes. At test run, when a new hand image patch goes through the tree from root to certain leaf node, the similarity score is obtained by executing the very simple bitwise OR operation for DOT matching. This is computationally very efficient (bitwise OR operations) and also saves huge storage memory as there is no need to store raw images.

2.4 Theoretical Analysis

Here we present the theoretical analysis for our basic regression forest model (*Baseline-M*). Denote $(\overline{X},\overline{Y})$ a pair of random variables following certain joint distribution, and μ the marginal distribution of $\overline{X} \in [0,1]^d$. In regression analysis, one is interested in estimating the regression function $r(\overline{\mathbf{x}}) := \mathbb{E}\{\overline{Y}|\overline{X} = \overline{\mathbf{x}}\}$ for fixed $\overline{\mathbf{x}}$ based on the training sample. A sequence of regression estimates $r_n(\overline{\mathbf{x}})$ is called *weakly consistent* for a certain distribution of $(\overline{X}, \overline{Y})$ if

$$\lim_{n\to\infty} \mathbb{E}\Big\{\|r_n(\overline{X}) - r(\overline{X})\|^2\Big\} = 0, \tag{6}$$

where $\|\cdot\|$ is the standard Euclidean norm in $\mathbb{R}^q$. The following consistency analysis is obtained for our aforementioned regression forest model, *Baseline-M*.

Theorem 1 *Assume that $\overline{X}$ is uniformly distributed on $[0,1]^d$ and $\mathbb{E}\Big\{\big\|\overline{Y}\big\|^2\Big\} < \infty$, and suppose the regression function $r(\overline{\mathbf{x}})$ is bounded. Then the regression forest estimates $\big\{r_n^{(Z)}\big\}$ of our* Baseline-M *model in* (4) *is consistent whenever $\lceil \log_2 L_n \rceil \to \infty$, $\frac{L_n}{n} \to 0$ and $\frac{k_n}{n} \to 0$ as $n \to \infty$.*

Proof details of our consistency theorem is relegated to the appendix. Recall the optimal estimator is the regression function $r(\overline{\mathbf{x}})$ which is usually unknown. The theorem guarantees that as the amount of data increases, the probability that the estimate $r_n(\overline{\mathbf{x}})$ of our regression forests is within a small neighbourhood of the optimal estimator will approach arbitrarily close to one. In our context when infinitely many synthetic examples are potentially available for training, it suggests that our estimate, constructed by learning from a large amount of examples, is optimal with high probability.

We would like to point out in passing that our proposed random forest model and the theorem bear noticeable differences from existing ones and especially [13] that have been theoretically analyzed in literature. The work of [13] considers only univariate problems while our regression forests deal with more general multivariate problems. Besides, the split criteria of our regression forest model possess two major differences: The first is how we select the split dimension and split threshold. In our model, the number of candidate split dimensions follows a binomial distribution, while in [13] it follows a Poisson distribution. More importantly, when deciding the best test (ϕ^*, t^*), we consider multi-variate Gaussian entropies while the squared error is used in [13]; The second is the forest depth control: There is no depth control in [13] during tree constructions, as the split will continue as long as there are sufficient examples in current split node. Meanwhile, most practical random forests require a tree depth control mechanism, which is also considered in our approach. We will stop splitting if one of the following two situations happens:

(1) The maximum tree depth $\lceil \log_2 L_n \rceil$ is reached.
(2) The splitting of the node using the selected split point results in any child with fewer than k_n estimation points.

These criteria ensure that each tree has no more than $\lceil \log_2 L_n \rceil$ levels and each leaf node in the tree contains at least k_n estimation points. In the theoretical analysis, we require $L_n \to \infty$ and $\frac{L_n}{n} \to 0$ as $n \to \infty$, while [13] requires $k_n \to \infty$ and $\frac{k_n}{n} \to 0$ as $n \to \infty$. We would also like to point out that so far we are able to provide analysis of the basic model (*Baseline-M*), while the analysis of *Baseline-S* and *DHand* remains open for future study. Meanwhile empirically *DHand* is shown to outperform the rest two models by a large margin.

3 The Pipeline of Our Learning System

3.1 Preprocessing

Our approach relies on *synthetic* hand examples for training, where each training example contains a synthetic hand depth image and its corresponding 3D pose. The learned system is then applied to *real* depth images at test stage for pose estimation. Particularly, depth noises are commonly produced by existing commodity-level depth cameras, which renders noticeable differences from the synthetic images. For TOF cameras, this is overcome by applying median filter to clear away the outliers, which is followed by Gaussian filter to smooth out random noises. The amplitude image, also known as the confidence map, is used to filter out the so called "flying pixel" noise [22]. The pixel with low confidence value is treated as the background. For structured illumination cameras, the preprocessing strategy of [40] can be applied. To obtain a hand image patch, a simple background removal technique similar to that of [31] is adopted, followed by image cropping to obtain a hand-centered bounding box. Moreover, to accommodate hand size variations, a simple calibration process is applied to properly scale a new hand size to match with that of the training ones, by acquiring an initial image with hand fully stretched and flat, and all fingers spread wide. Empirically these preprocessing ingredients are shown to work sufficiently well.

3.2 Our Two-step Pipeline

After preprocessing, our approach consists of two major steps as in Fig. 2: Step one involves mainly estimation

of 3D location and in-plane rotation of the hand base (i.e. wrist) using a regression forest. This is utilized in step two which subsequently establishes its coordinate based on the estimated 3D location and in-plane rotation. In step two, a similar regression forest model estimates the rest parameters of our hand kinematic model by a dynamically weighted scheme, which produces the final pose estimation. Note that different from existing methods such as [24] where by introducing the conditional model, a lot of forest models (each catering one particular condition) have to be prepared and kept in memory, our pipeline design requires only one forest model after the translation and in-plane rotation of step one to establish the canonical coordinate.

In both steps of our pipeline, two almost identical regression forests are adopted. In what follows, separate descriptions are provided that underline their differences. This allows us to present three variants of our learning system with a slight abuse of notations: The *Baseline-M* system employs the basic *Baseline-M* regression model on both steps; Similarly, the *Baseline-S* system utilizes instead the *Baseline-S* models in both steps; Finally, the *DHand* system applies the *DHand* regression forest model only at step two, while the *Baseline-S* model is still engaged in step one. It is worth mentioning that for the *Baseline-M* system, our theoretical analysis applies to both regression forests models used in the two steps of our pipeline.

Before proceeding to our main steps, we would like to introduce the 3D hand poses, the related depth features and tests utilized in our approach, which are based on existing techniques as follows:

Our kinematic chain model: As displayed in Fig. 4, the representation of our 3D hand poses follows that of [40]: 4 DoF are used for each of the five fingers, and 1 DoF is explicitly for palm bending, as well as 6 DoF reserved for the global hand location (x_1, x_2, x_3) and orientation (α, β, γ), where α stands for the in-plane rotation. This amounts to a 27-dimensional vector $\Theta := \big(x_1, x_2, x_3, \alpha, \beta, \gamma, \cdots\big)$ as the hand kinematic chain model, used in our system to represent each of the 3D hand poses. Sometimes it is more convenient to denote as $\Theta = \big(x_1, x_2, x_3, \alpha, \mathbf{z}\big)$, with $\mathbf{z}$ being a 23-dimensional sub-vector.

Depth features and binary tests: Let I denote the hand image patch obtained from raw depth image. Without loss of generality, one depth image is assumed to contain only one right hand. The depth features as mentioned in [31] are adapted to our context here. That is, at a given pixel location $\mathbf{x} = (\hat{x}_1, \hat{x}_2)$ of a hand patch I, denote its depth value as a mapping $d_I(\mathbf{x})$, and construct a feature $\phi(\mathbf{x})$ by considering two 2D offsets positions u, v from $\mathbf{x}$:

$$\phi(\mathbf{x}) = d_I\Big(\mathbf{x} + \frac{\mathrm{u}}{d_I(\mathbf{x})}\Big) - d_I\Big(\mathbf{x} + \frac{\mathrm{v}}{d_I(\mathbf{x})}\Big). \tag{7}$$

Following [9], a binary test is defined as a pair of elements, (ϕ, t), with ϕ being the feature function, and t being a real-valued threshold. When an instance with pixel location $\mathbf{x}$ passes through a split node of our binary trees, it will be sent to the left branch if $\phi(\mathbf{x}) > t$, and to the right side otherwise.

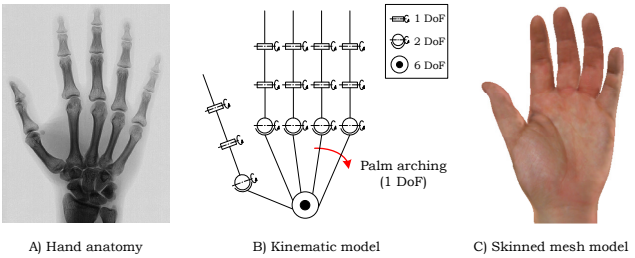

Fig. 4: Our 3D hand kinematic model Θ contains 21+6 degree-of-freedom (DoF), including the hand base (i.e. wrist) position and orientation (6 DoF), and the relative angles of individual joints (21 DoF). From (A) to (C): The hand anatomy, the underlying skeleton kinematic model, and the skinned mesh model.

3.2.1 Step One: Estimation of Coordinate Origin and In-plane Rotation

This step is to estimate the 3D location and in-plane rotation of the hand base, namely (x_1, x_2, x_3, α), which forms the origin of our to-be-used coordinate in step two. The (instance, label) pair of an example in step one is specified as follows: The instance (aka feature vector) $\bar{\mathbf{x}}$ is obtained from an image patch centered at current pixel location, $\mathbf{x} = (\hat{x}_1, \hat{x}_2)$. Each element of $\bar{\mathbf{x}}$ is realized by feeding particular u, v offset values in (7). Correspondingly the label of each example $\bar{\mathbf{y}}$ is the first four elements of the full pose label vector Θ, namely (x_1, x_2, x_3, α). A regression forest is used to predict these parameters as follows: Every pixel location in the hand image patch determines a training example, which is parsed by each of the T_1 trees, resulting in a path from the root to certain leaf node that stores a collection of training examples. Empirically we observe that this 3D origin location and in-plane rotation are usually estimated fairly accurately.

Split Criterion of the First Step For the regression forest of the first step, its input is an image patch centered at current pixel, from which it produces the 4-dimensional parameters (x_1, x_2, x_3, α). The entropy term of (2) is naturally computed in this 4-dimensional space (i.e. $q = 4$).

3.2.2 Step Two: Pose Estimation

The depth pixel values of a hand image patch naturally form a 3D point cloud. With the output of step one, the point cloud is translated to $(x_1,\ x_2,\ x_3)$ as coordinate origin, which is followed by a reverse-rotation to the canonical hand pose by the estimated in-plane rotation α. An almost identical regression forest is then constructed to deliver the hand pose estimation: With the location output of step one, $(x_1,\ x_2,\ x_3)$, as the coordinate origin, each *entire* hand patch from training is parsed by each of the T_2 trees, leading down the tree path to a certain leaf node. The regression forest model of this step then delivers a 23-dimensional parameter vector $\mathbf{z}$, by aggregating the votes of the training example of the leaf nodes. The final 27-dimensional parameter estimation Θ is then obtained by direct composition of results from both steps. Meanwhile for step two, $\bar{\mathbf{x}}$ stands for a feature vector of the entire hand image patch, while $\bar{\mathbf{y}} := \mathbf{z}$ represents the remaining 23 elements of Θ.

Split Criterion of the Second Step The second step focuses on estimating the remaining 23-dimensional parameters, which resides in a much larger space than what we have considered during the first step. As a result, by straightforwardly following the same procedure as in step one, we will inevitably work with a very sparsely distributed empirical point set in a relatively high dimensional space. As a result, it consumes considerably amount of time, while the results might be unstable. Instead we consider an alternative strategy.

To start with, empirically we observe that a precise estimation of the rotation around Y-axis (i.e. roll) is among the most challenging factors in hand pose estimation. Fig. 5 displays an exemplar scenario, where the appearances of the same gesture in depth maps will be visually very similar in spite of significant rotations around Y-axis. This inspires us to concentrate on rotations around Y-axis when measuring the differential entropy (2), which is only 1-dimensional. Moreover, to avoid the unbalance phenomenon during tree constructions, a balance term is also introduced and incorporated into an augmented information gain objective function:

$$\left(\phi^* t^*\right) = \arg \max_{\phi \in \Phi, t \in \mathcal{T}} \mathcal{I}_B(\phi, t), \tag{8}$$

with $\mathcal{I}_B(\phi, t) := B(\phi, t)\mathcal{I}(\phi, t)$ and $B(\phi, t)$ being the balance term:

$$B(\phi, t) = \frac{\min(|S_l|, |S_r|)}{\max(|S_l|, |S_r|)}.$$

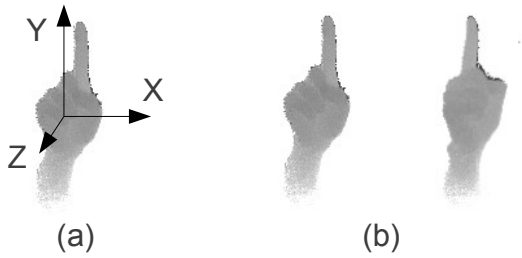

Fig. 5: (a) Three rotation parameters of a hand. Rotation around Z is termed the in-plane rotation, rotations around X and Y are called pitch and roll, respectively. (b) Two views of the same gesture (Chinese number counting "1") by rotating the hand around the Y-axis (i.e. roll). As suggested in (b), the appearances of the hand in depth maps are nevertheless very similar. In practice, precise estimation of the rotation around Y-axis turns out to be among the leading challenges in hand pose estimation.

To sum up, in term of split criteria, regression forests of the two steps follows almost the same scheme, except for the main differences below: 1) Step one is based on single pixel, while step 2 works with the entire hand patch. 2) Differences in computing entropy and information gain: As shown in (2), the entropy is computed in 4-dimensional space in step one, and in 1-dimensional space (i.e $q = 1$) for the second step. Moreover, the augmented information gain of (8) is used in step two. Note our theoretical consistency analysis in Theorem 1 also applies to this augmented information gain of (8).

4 GPU Acceleration

4.1 Motivation

Typically a leaf node is expected to contain similar poses. The vast set of feasible poses however implies a conflicting aim: On one hand, this can be achieved by making ready as many training examples as possible; On the other hand, practically we prefer a small memory print for our system, thus limiting the amount of data. A good compromise is obtained via imposing a set of small random perturbations including 2D translations, rotations, and hand size scaling for each of existing training instances, I_t. This way, a leaf node usually has a better chance to work with an enriched set of similar poses. For this purpose, small transformation such as in-plane translations, rotations and scaling are additionally applied on the training image patches. We remap I_t using m_t transformation maps. Every transformation map is generated using a set of small random

perturbations including 2D translations, rotations, and hand size scaling of the same hand gesture, and is of the same dimensions (i.e. $w \times h$) as of I_t. After remapping the values of I_t using a transformation map, its DOT features are generated and compared with each of the features of the k_l instances of I_{li} to obtain a similarity score. These DOT-related executions turns out to be *the* computation bottleneck in our CPU implementation, which can be substantially accelerated using GPU by exploiting the massive parallelism inherent in these steps. It is worth noting that the purpose here is to duplicate our CPU system with GPU-native implementation, in order to obtain the same performance with much reduced time.

4.2 Using texture units for remapping

The m_t random transformation maps used for remapping I_t have translational, rotational and scaling components. We generate these maps in advance to save on the cost of computing them for every depth image at runtime. By applying a map M_t, every pixel location $\mathbf{x} = (\hat{x}_1, \hat{x}_2)$ in I_t is mapped to a floating point coordinate $\mathbf{x}_f = \left(\hat{x}_1^{(f)}, \hat{x}_2^{(f)}\right)$ in I_t. The translation parameters $\left(\hat{x}_1^{(t)}, \hat{x}_2^{(t)}\right)$, the rotation parameter ξ and scaling parameters $\left(\hat{x}_1^{(s)}, \hat{x}_2^{(s)}\right)$ are sampled uniformly for every transformation map. The transformed coordinates $\mathbf{x}_f$ are computed as:

$$\hat{x}_1^{(f)} = \hat{x}_1^{(t)} + \frac{w}{2} + \hat{x}_1^{(s)} \left(\cos\xi \left(\hat{x}_1 - \frac{w}{2} \right) - \sin\xi \left(\hat{x}_2 - \frac{h}{2} \right) \right)$$

$$\hat{x}_2^{(f)} = \hat{x}_2^{(t)} + \frac{h}{2} + \hat{x}_2^{(s)} \left(\cos\xi \left(\hat{x}_1 - \frac{w}{2} \right) + \sin\xi \left(\hat{x}_2 - \frac{h}{2} \right) \right)$$

Each of these maps, M_t, has the same size as I_t and is stored as two maps in GPU global memory for efficient access, for the X and Y coordinate values respectively. Since $\mathbf{x}_f$ may not be exactly located at a pixel position, the pixels around $\mathbf{x}_f$ are interpolated to obtain the resulting depth values.

To perform this remapping on GPU, we first launch one thread for every $\mathbf{x}$ to read its $\mathbf{x}_f$ from M_t. Since all the threads in a thread block read from adjacent locations in GPU memory in a sequence, the memory reads are perfectly coalesced. To obtain a depth value at $\mathbf{x}_f$, we use the four pixels whose locations are the closest to it in I_t. The resulting depth value is computed by performing a bi-linear interpolation of the depth values at these four pixels [5] [21]. Reading the four pixels around $\mathbf{x}_f$ is inefficient since the image is stored in memory in row order and memory accesses by adjacent threads can span across multiple rows and columns of I_t and thus cannot be coalesced. This type of memory access is not a problem in CPU computation due to its deep hierarchy of caches with large cache memories at each level. However, the data caches in GPU architecture are tiny in comparison and are not very useful for this computation. The row order memory layout that is commonly used has poor locality of reference. Instead an isotropic memory layout is needed, with no preferred access direction. Instead this operation can be performed in GPU by utilizing its two-dimensional texture memory, which ensures that pixels that are local in image space are almost always local in the memory layout [30].

4.3 Computing DOT features

Computing the DOT features for each of the m_t remapped images takes two steps: computing the gradient at every pixel and then the dominant orientation of every block in the image. One thread is launched on GPU for every pixel to compute its X and Y gradient values. We apply a 3×3 Sobel filter to compute the gradient and the memory reads are coalesed across a warp of threads for efficiency. Using the gradient values, the magnitude and angle of the gradient vector is computed and stored in GPU memory. We use the fast intrinsic functions available on GPU to compute these quickly.

To pick the orientation of the pixel whose magnitude is largest in a block, the common strategy of launching one thread per pixel is not practical. The cost of synchronization between threads of a DOT block is not worthwhile since the dimensions of the block (8×8) are quite small in practice. Instead, we launch one thread for every DOT block to compare the magnitude values across its pixels and note the orientation of the largest magnitude vector.

4.4 Computing DOT feature matching

The DOT feature comparison is essentially composed of two steps: bitwise comparison and accumulation of the comparison result. The bitwise comparisons can be conveniently performed by using one thread per orientation in the DOT feature. A straightforward method to accumulate the comparison result is to use parallel segmented reduction. However, this can be wasteful because the size of DOT feature is typically small and the number of training examples is typically large. To accumulate efficiently, we use the special atomic addition operations that have been recently implemented in GPU hardware.

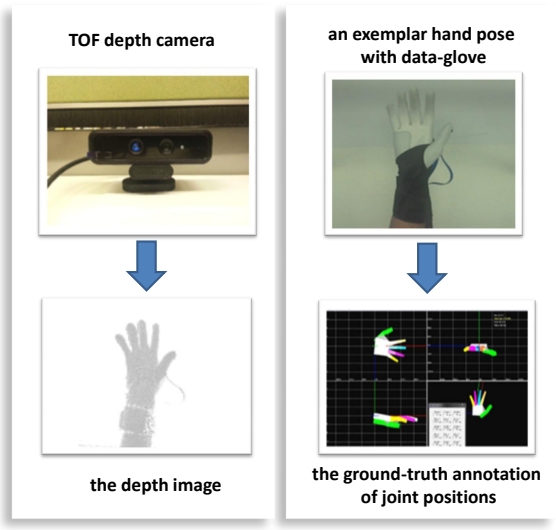

Fig. 6: An illustrative flowchart of our annotated dataset. The depth images are acquired through a time-of-flight (TOF) camera. The corresponding annotated finger joint locations of each depth image are obtained from the state-of-the-art data-glove, ShapeHand. See text for details.

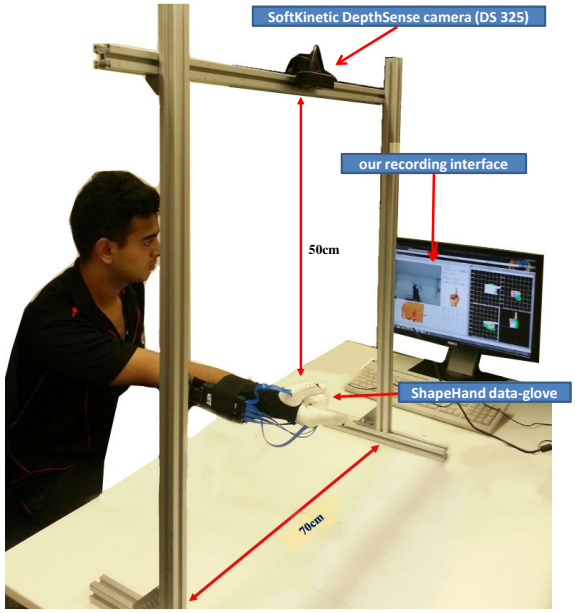

Fig. 7: A photo of our data capture set-up.

5 Our Annotated Real-world Dataset and Performance Evaluation

The annotated real-world dataset:

To facilitate the analysis of hand pose estimation systems, we also make available our data-glove annotated real-world dataset as well as online performance evaluation[2]. We wish this can provide an option for researchers in the field to compare performance on the same ground. As presented in Fig. 6, in our dataset the depth images are acquired through a TOF camera (SoftKinetic DS325 [3]). The corresponding annotated depth images is obtained from the state-of-the-art data-glove, ShapeHand [1]. Our data capture setup is depicted in Fig. 7, where a data-gloved hand is performing in a desktop setting with the depth camera positioned overhead. Each depth image contains only one hand, and without loss of generality we consider only the right hand, and fix the camera to hand distance to around 50 cm. Our dataset contains images collected from 30 volunteers varying in age (18 - 60 years), gender (15 male and 15 female), race and hand shape. 29 images are obtained for each volunteer during the capture sessions, where 8 of these images are from the Chinese Number Counting system (from 1 to 10, excluding 3 and 7), and the remaining ones are from the American Sign Language (ASL) alphabet (from A to Z, excluding J, R, T, W and Z), as illustrated in Fig. 9. Together, these amount to 870 annotated examples, with each example consisting of a hand depth image and its label (the data-glove annotation).

In addition to our kinematic chain model of Fig. 4, an alternative characterization [17] of a 3D hand pose consists of a sequence of joint locations $\mathbf{v} = \{v_i \in \mathbb{R}^3 : i = 1, \cdots, m\}$, where m refers to the number of joints, and v specifies the 3D location of a joint. In term of performance evaluation, this characterization by joint locations (as illustrated in Fig. 8) is usually easily interpreted when comes to comparing pose estimation results. As this hand pose characterization is obtained from the ShapeHand data-glove, there exists some slight differences in joints when comparing with the kinematic model: First, all five finger tips are additionally considered in Fig. 8; Second, there are three thumb joints in Fig. 4 while only two of them are retained in Fig. 8, as ShapeHand does not measure the thumb base joint directly. Nevertheless there exists a unique mapping between the two characterizations.

Finally, of all the subjects (15M&15F), half (i.e. 8M&7F) are used for training while the other half (7M&8F) are retained as test data for performance evaluation. For a training example, both the depth image and its label are presented. For a test example, only the depth image are present.

Performance evaluation metric and its computation:

Our performance evaluation metric is based on the *joint error*, which is defined as the averaged Euclidean

[2] Our annotated dataset of depth images and the online performance evaluation system for 3D hand pose estimation are publicly available at http://hpes.bii.a-star.edu.sg/.

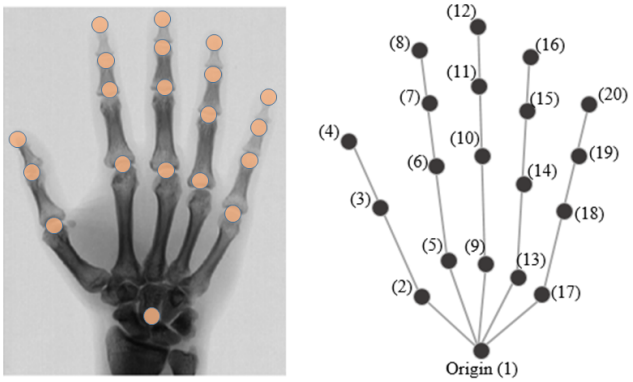

Fig. 8: An illustration of a ground-truth annotation used in our annotated dataset for performance evaluation. For an hand image, its annotation $\mathbf{v}$ contains 20 joints. It can also be considered as a vector of length $60 = 20 \times 3$, consisting of the 3D locations of the joints following a prescribed order. Note the joint locations here are exactly all the finger joints of our skeletal model as of Fig. 4 plus the tips of the five fingers and the hand base, except for one thumb joint that is not included here due to the specific data-glove apparatus used during empirical experiments. In practice, the three thumb-related joints are not considered, which gives $m = 20 - 3 = 17$.

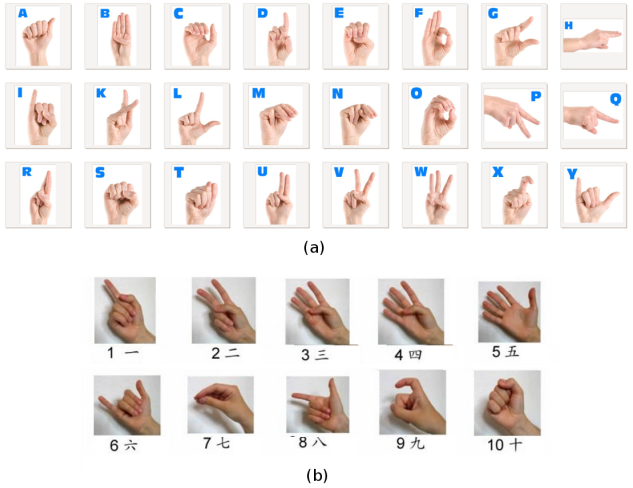

Fig. 9: An illustrative list of 29 gestures used in our real-world dataset, which are from (a) the American sign language (ASL) and (b) the Chinese number counting. In particular, images of the ASL letters used in this figure are credited to http://schoolbox.wordpress.com/2012/10/30/3315/, while images of Chinese number counting are to http://www.movingmandarin.com/wordpress/?p=151. We note that gestures of letters J and Z are not considered here as they involve motions thus require an image sequence to characterize one such letter. Moreover, gestures of numbers 3 and 7 as well as letters R, T, W, although displayed here, are also not used in our dataset. It is mainly due to the measurement limitation of ShapeHand data-glove, which restricts from consideration gestures that are substantially involved with either thumb finger articulations, or palm arching. See text for more details.

distance in 3D space over all the joints. Note the joints in this context refer to the 20 joints defined in Fig. 8, which are exactly all the joints of our skeletal model as of Fig. 4 plus the tips of the five fingers and the hand base, except for one thumb joint that is not included here due to the compatibility issue with ShapeHand data-glove used during empirical experiments. Formally, denote $\mathbf{v}_g$ and $\mathbf{v}_e$ as the ground truth and estimated joint locations. The *joint error* of the hand pose estimate $\mathbf{v}_e$ is defined as $e = \frac{1}{m}\sum_{i=1}^{m} \|v_{gi} - v_{ei}\|$, where $\|\cdot\|$ is the Euclidean norm in 3D space. Moreover, as we are dealing with a number of test hand images, let $j = 1, \ldots, n_t$ run over the test images, the corresponding joint errors are $\{e_1, \cdots, e_{n_t}\}$, then the *mean joint error* is defined as $\frac{1}{n_t}\sum_j e_j$, and the *median joint error* is simply the median of the set of errors.

When working with annotated real depth images, there are a number of practical issues to be addressed. Below we present the major ones: To avoid the interference of the tapes fixed at the back of the ShapeHand data-glove, our depth images focus around the frontal views. Empirically, we have evaluated the reliability of the data-glove annotations. This is achieved via a number of simple but informative tests where we have observed that the ShapeHand device produces reasonable and consistent measurements (i.e. within mm accuracy) on all the finger joints except for the thumb, where significant errors are observed. We believe that the source of this error lies in the design of the instrument. As a result, even though we have included the thumb-related joints in our dataset, they are presently ignored during performance evaluation. In other words, the three thumb-related joints are not considered while evaluating the hand-pose estimation algorithms. As displayed in Fig. 8, this gives $m = 20 - 3 = 17$ in practice. The data-glove also gives inaccurate measurements when the palm arches (bends) deeply. Therefore we have to withdraw from consideration several gestures including 3, 7, R, T and W. Note on synthetic data all finger joints are considered as discussed previously. The last which is nevertheless the most significant issue is an alignment problem as follows: Due to the physical principle of ShapeHand data acquisition, its coordinate frame is originated at the hand base, which is different from the coordinate used by the estimated hand pose from depth camera. They are related by a linear coordinate transformation. In other words, the estimated joint positions need to be transformed from the camera coordinate to the ShapeHand coordinate frame. More specifically, denote the 3D location of a joint by v_i^S and v_i^C, the 3D vectors corresponding to the ShapeHand (S) and the camera (C) coordinate frames, respectively, where i indexes over the m joints. The transformation matrix T_C^S

is the 3D transformation matrix from (C) to (S), which can be uniquely obtained following the least-square 3D alignment method of [38].

6 Experiments

All experiments are carried out on a laptop with an Intel Core-i7 CPU and 4 Gb memory. The Softkinetic DS325 TOF camera [3] is used as the primary apparatus to acquire real-world depth images, with image size 320 × 240, and field of view (H × V) is 74° × 58°. It can be derived that the resolution along X and Y direction is 1.73mm at 500mm distance. The resolution along Z direction is reported [3] to be within 14mm at 1m. TOF cameras are also known to contain noticeable noises including e.g. the so-called flying pixels [22].

Throughout experiments we set $T_1 = 7$ and $T_2 = 12$. The depth of the trees is 20. Altogether 460K *synthetic* training examples are used, as illustrated in Fig. 11. These training examples cover generic gestures from American sign language and Chinese number counting, together with their out-of-plane pitch and roll rotations, as well as in-plane rotational perturbations. The minimum number of estimation examples stored at leaf nodes is set to $k_n = 30$. m_0 is set to a large constant of $1e7$, that practically allows the consideration of all training examples when choosing a threshold t at a split node. The evaluation of depth features requires the access to local image window centered at current pixel during the first step, and of the whole hand patches during the second step, which are of size $(w, h) = (50, 50)$ and $(w, h) = (120, 160)$ respectively. The size of feature space d is fixed to 3000, and the related probability $p = 0.2$. Distance is defined as the Euclidean distance between hand and camera. By default we will focus on our *DHand* model during experiments.

In what follows, empirical simulations are carried out on the synthetic dataset to investigate myriad aspects of our system under controlled setting. This is followed by extensive experiments with real-world data. In addition to hand pose estimation, our system is also shown to work with related tasks such as part-based labeling and gesture classification.

6.1 Experiments on Synthetic Data

To conduct quantitatively analysis, we first work with an in-house dataset of 1.6K synthesized hand depth images that covers a range of distances (from 350mm to 700mm). Similar to real data, the resolution of the depth camera is set to 320 × 240. When the distance from the hand to the camera is dist = $350mm$, the bounding box of the hand in image plane is typically of size 70×100; when dist = $500mm$, the size is reduced to 49×70; When dist = $700mm$, the size further decreases to 35×50. White noise is added to the synthetic depth images with standard deviation $15mm$.

Fig. 10: The color code of our 3D hand.

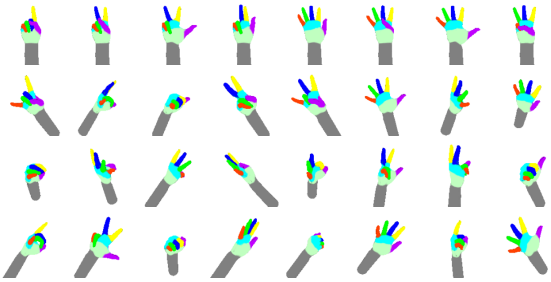

Fig. 11: Exemplar *synthetic* hand gestures used during training. The training examples cover generic gestures from American sign language and Chinese number counting, their out-of-plane pitch and roll rotations, as well as in-plane rotational perturbations. The first row illustrates various gestures in frontal view, while the rest rows display different gestures observed from diverse viewpoints.

Estimation Error of Step One

As being a two-step pipeline in our system, it is of interest to analyze the errors introduced by step one, namely the hand position and in-plane rotation errors. Fig. 12 displays in box-plots the empirical error distributions on synthetic dataset. On average, the errors are relatively small: 3D hand position error is around 10mm, while the in-plane rotation error is around 2 degrees. It is also observed that there are no significant error changes as the distances vary from 350mm to 700mm.

We further investigate the effects of perturbing the estimates of step one toward the final estimates of our system on the same synthetic dataset. A systematic study is presented in the three-dimensional bar plot of Fig. 13, where the hand position and in-plane rotation errors of step one form the two-dimensional input, which produces as output the mean joint error: Assume the inputs from step one are perfect (i.e. with zero errors in both dimensions), final error of our system is around 15mm. As both input errors increase, the

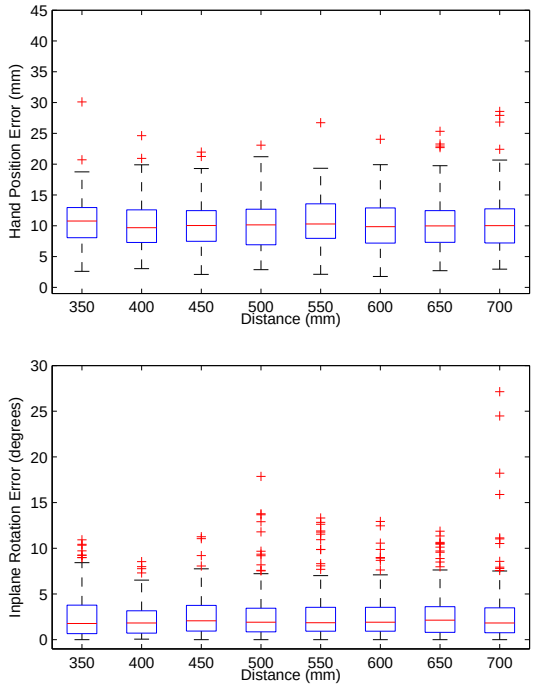

Fig. 12: Box-plot of the hand position errors and the in-plane rotation errors of the *first* regression forest in our system as a function of the distance from hand to camera, obtained on the synthetic dataset.

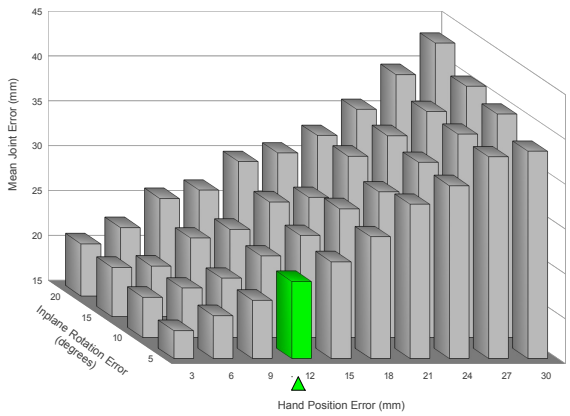

Fig. 13: Effect of perturbations in hand position and in-plane rotation errors of step one toward the our final system outputs.

final mean joint error will go up to over 40mm. So it is fair to say that our system is reasonably robust against perturbation of the results from step one. Interestingly, our pipeline seems particularly insensitive to the in-plane rotation error of step one, which changes only 5mm when the in-plane rotation error varies between 0 to 30 degrees. Finally, as shown in Fig. 13 where the errors of our first step (the green bar) is relatively small, our final estimation error is around 22mm (mean joint error).

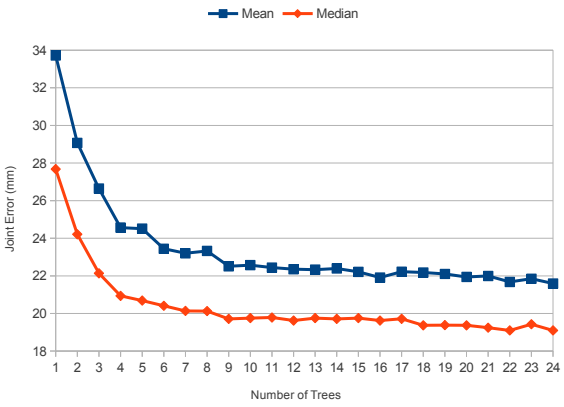

Fig. 14: Mean and median joint errors of our system when the number of trees in step two varies.

Number of Trees

Experiments are conducted to evaluate on how much the number of trees influences on the performance of our regression forest model. As the forests of both steps are fairly similar, we focus on step two and present in Fig. 14 the mean/median joint errors as a function of the number of trees. As expected, the errors decreases as the number of trees increases. The rate of decreases primes at 4 trees, and at around 12 trees or larger numbers, the decreases become negligible. This motivate us to set $T_2 = 12$ in our system. We note in the passing that empirically the median errors are slightly smaller than the mean errors, which is to be expected as median error metric is known to be less insensitive to outliers (i.e. the few test instances in the test set with largest errors).

Split Criteria

Two different split criteria are used for tree training in the second forest. When all 23 parameters are used to compute the entropy, the mean and median joint errors are 21.7mm and 19.1mm respectively. The hand rotation around Y-axis plays an important role in training the forest. In each node considering only the distribution of Y-axis rotation and the balance of the split (8), the mean and median joint errors are 21.5mm and 19.2mm respectively. The performance of considering only Y-axis rotation is as good as that of considering all 23 parameters.

Performance Evaluation of the Proposed vs. Existing Methods

Fig. 15 provides a performance evaluation (in term of mean/median joint error) among several competing methods. They include the proposed two baseline methods

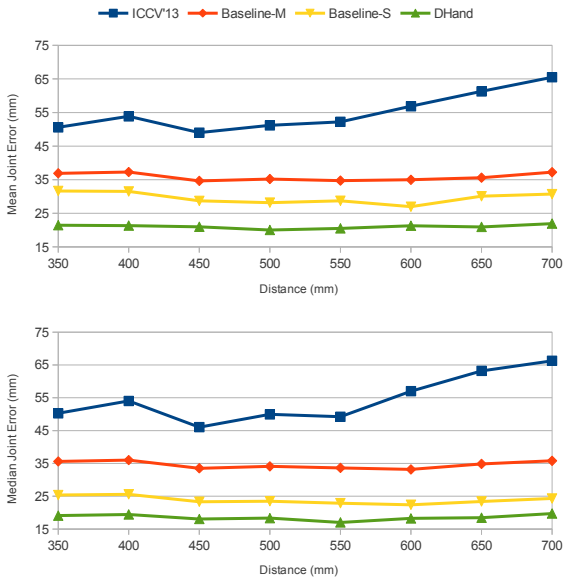

Fig. 15: Performance evaluation of the proposed and the comparison methods.

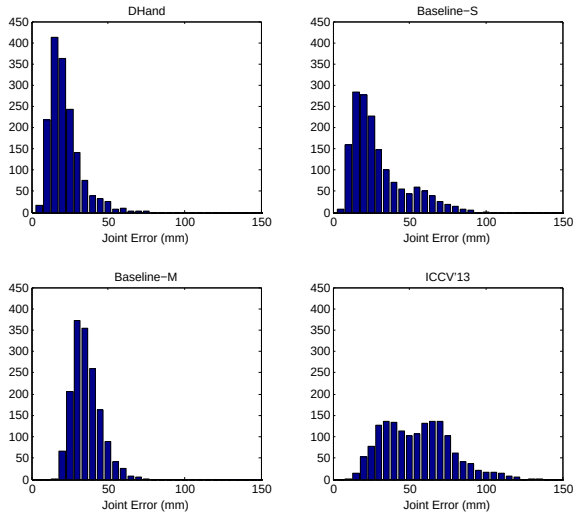

Fig. 16: Empirical error distributions of the comparison methods.

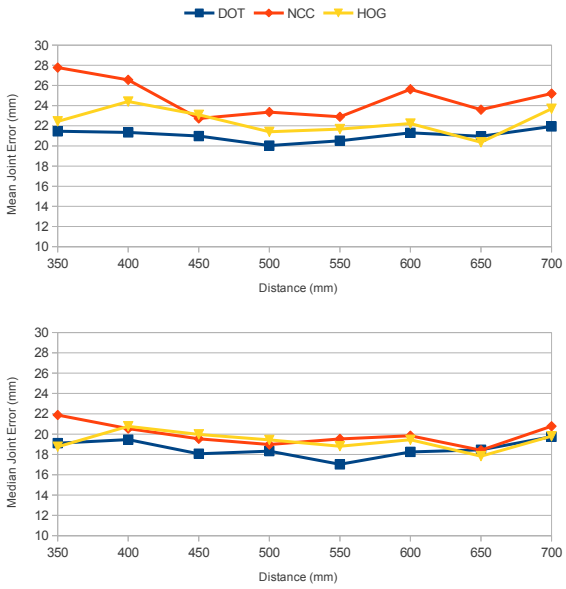

Fig. 17: Performance comparison over different matching methods: DOT [23], HOG [12], and NCC [26].

(*Baseline-M* and *Baseline-S*), the proposed main method (*DHand*), as well as a comparison method [40] denoted as *ICCV'13*. Overall our method *DHand* deliver the best results across all distances, which is followed by *Baseline-M* and *Baseline-S*. This matches well with our expectation. Meanwhile *ICCV'13* achieves the worst performance. In addition, our proposed methods are shown to be rather insensitive to distance changes (anywhere between 350-700mm), while *ICCV'13* performs the best around 450mm, then performance declines when working with larger distance.

We further analyze the empirical error distributions of the comparison methods, as plotted in Fig. 16. Here it becomes clear that the inferior behavior of *ICCV'13* can be attributed to its relatively flat error distributions, which suggests some joints deviate seriously from their true locations. This is in sharp contrast to the error distribution of *DHand* shown at the top-left corner, where majority of the errors reside at the small error zone. Finally, *Baseline-M* and *Baseline-S* lie somewhere in-between, with their peaks lie on the small error side.

Comparison over Different Matching Methods: There are a few state-of-the-art object template matching methods that are commonly used for related tasks, including DOT [23], HOG [12], and NCC [26]. Fig. 17 presents a performance comparison of our approach when adopting these matching methods. It is clear that DOT consistently performs the best, which is followed by HOG, while NCC always delivers the worst results. In term of memory usage, DOT consumes 100MB, HOG takes 4GB, while and NCC needs 2GB. Clearly DOT is the most cost-effective option. Note that in addition to the 100MB DOT consumption, the 288MB memory footprint of our system also includes other overheads such as third-party libraries.

6.2 Experiments on Real Data

Experiments of this section focus on our in-house real-world depth images that are introduced in Section 5. By default, the distance from the hand to the camera is fixed to 500mm. Throughout experiments three sets of depth images are used as presented in Fig. 1: (1) bare

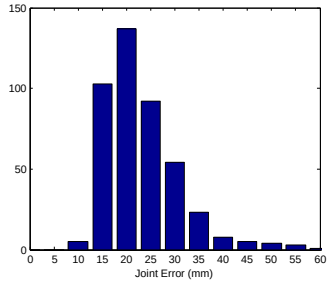

Fig. 18: Empirical error distribution of *DHand* on the aforementioned annotated real-world dataset.

hand imaged by top-mounted camera; (2) bare hand imaged by front-mounted camera; (3) hand with dataglove imaged by top-mounted camera.

Fig. 18 presents the empirical error distribution with our proposed *DHand* on this test set. Empirically only a small fraction of the errors occurs with very large errors (e.g. 40mm and above), while most resides in the relatively small error (e.g. 10-25mm) area. This suggests that the pose estimation of *DHand* usually does not incur large errors such as mistaking a palm by a hand dorsum (i.e. the back of hand) or vice versa. As a summary of this empirical error distribution, its median/mean joint errors are 21.1/22.7mm, which are comparable with what we have on the synthetic dataset where the median/mean joint errors are 19.2/21.5mm. We further look into its spread over different subjects and gestures, which are displayed in Fig. 19: In the top plot, we can see the errors over subjects are quite similar. The differences over subjects may due to the hand sizes, as smaller hand tends to incur smaller errors. In the bottom plot, it is clear that simple gestures such as "5" receive relatively small errors, while some gestures such as "6", "10", and "F" tend to have larger errors, as many finger joints are not directly observable.

Furthermore, Fig. 20 presents a list of hand pose estimation results of *DHand* on real-world depth images of various gestures, orientations, hand sizes, w/ vs. w/o gloves, as well as different camera set-ups. Throughout our system is shown to consistently deliver visually plausible results. Some failure cases are also shown in Fig. 21, which will be analyzed later.

Comparisons with State-of-the-art Methods

Experiments are also conducted to qualitatively evaluate *DHand* and the state-of-the-art methods [4, 29, 35] on pose estimation and tracking tasks, as manifested in Fig. 22 and Fig. 23. Note [35] is re-implemented by ourselves while original implementations of the rest two methods are employed.

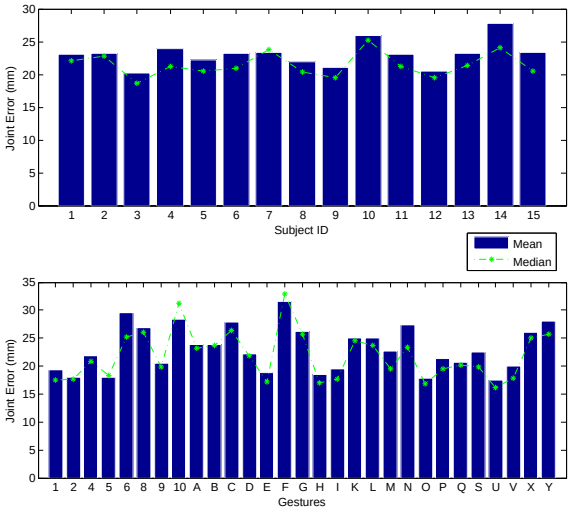

Fig. 19: Mean/Median joint errors of *DHand* over different subjects/gestures on the annotated real-world dataset.

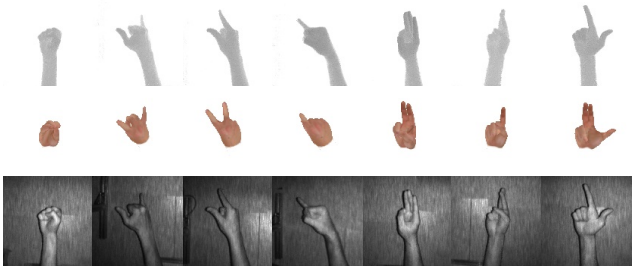

Fig. 21: Failure cases.

Recently the latent regression forest (LRF) method has been developed in [35] to estimate finger joints from single depth images. As presented in Fig. 22, for the eight distinct hand images from left to right, LRF gives relatively reasonable results on the first four and makes noticeable mistakes on the rest four scenarios, while our method consistently offers visually plausible estimates. Note in this experiment all hand images are acquired in frontal facing view only, as LRF has been observed to deteriorate significantly when the hands rotate around the Y-axis, as is also revealed in Fig. 5 in our paper, an issue we have considered as the leading challenges for hand pose estimation.

We further compare *DHand* with two state-of-the-art hand tracking methods, which are the well-known tracker of [29], and a commercial software, Leap Motion [4], where the stable version 1.2.2 is used. Unfortunately each of the trackers operates on a different hardware: [29] works with Kinect [2] to take as input a streaming pairs of color and depth images, while Leap Motion runs on proprietary camera hardware [4]. Also its results in Fig. 23 are screencopy images from its visualizer as being a closed system. To accommodate the

differences, we engage the cameras at the same time during data acquisition, where the Kinect and the Softkinetic cameras are closely placed to ensure their inputs are from similar side views, and the hands are hovered on top of Leap motion with about 17 cm distance. we also allow both trackers with sufficient lead time to facilitate both functioning well before exposing to each of the hand scenes as displayed at the first row of Fig. 23. Taking each of these ten images as input, *DHand* consistently delivers plausible results, while the performance of both tracking methods are rather mixed: [29] seems not entirely fits well with our hand size, and in particular, we have observed that its performance degrades when the palm arches as e.g. in the seventh case. Leap Motion produces reasonable results for the first five cases and performs less well on the rest five.

Part-based Labeling

Our proposed approach can also be used to label hand parts, where the objective is to assign each pixel to one of the list of prescribed hand parts. Here we adopt the color-coded part labels of Fig. 10. Moreover, a simple scheme is adopted to convert our hand pose estimation to part-based labels: From input depth image, the hand area is first segmented from background. Our predicted hand pose is then applied to a synthetic 3D hand and projected onto the input image. This is followed by assigning each overlapping pixel a proper color label. For pixels not covered by the synthetic hand model, we allocate each of them with a label from the nearest overlapped regions.

Fig. 24 presents exemplar labeling results on real-world depth images where the data-glove is put on. To illustrate the variety of images we present horizontally a series of unique gestures and vertically instances of the same gesture but from different subjects. Visually the results are quite satisfactory, where the color labels are mostly correct and consistent across different gestures and subjects. It is also observed that our annotation results are remarkably insensitive to background changes including the wires of the data-glove.

Gesture Classification

Instead of emphasizing on connecting and comparing with existing ASL-focused research efforts, the aim here is to showcase the capacity of applying our pose estimation system to address related task of gesture recognition. Therefore, we take the liberty of considering a combined set of gestures, which are exactly the 29 gestures discussed previously in the dataset and evaluation section — instead of pose estimation, here we consider the problem of assigning each test hand image to its corresponding gesture category. Notice that some gestures are very similar to each other: They include e.g. {“t”, “s”, “m”, “n”, “a”, “e”, “10”}, {“p”, “v”, “h”, “r”, “2”}, {“b”, “4”}, and {“x”, “9”}, as illustrated also in Fig. 9. Overall the average accuracy is 0.53. The overall low score is mainly due to the similarity of several gesture types under consideration. For example, X in ASL is very similar to number counting gesture 9, and is also very close to number 1. This explains why for letter X, it is correctly predicted with only 0.27, and with 0.27 it is wrongly classified as number 9 and with 0.13 as number 1, as displayed in the confusion matrix at Fig. 25.

Execution Time

Efficient CPU enables our system to run near realtime at 15.6 FPS, while our GPU implementation further boosts the speed to 67.2 FPS.

Limitations

Some failure cases of our hand pose estimation are presented in Fig. 21. A closed hand is usually difficult to deal with since no finger joints or tips are visible: As demonstrated in the first column, it might be confused with some similar gestures. Even when few fingers are in sight, different hand gestures might still be confused as they look very similar when projecting to the image plane from certain viewing angles, as presented in the 2rd to 4th columns. The last two columns display scenarios of overlapped fingers which might also be wrongly estimated.

As our approach is based on single depth images, the results may appear jittered when working with video streams. We also remark that our method is fundamentally different from tracking-based methods, where gradient based or stochastic optimization method would be used to exploit temporal information available. As a result, the accuracy of our method might slightly lag behind a tracking enabled approach with good initializations.

7 Conclusion and Outlook

This paper presents an efficient and effective two-step pipeline for hand pose estimation. GPU-acceleration of the computational bottleneck component is also presented that significantly speeds up the runtime execution. A data-glove annotated hand depth image dataset

is also described as an option for performance comparison of different approaches. Extensive empirical evaluations demonstrate the competitive performance of our approach. This is in addition to theoretical consistency analysis of its slightly simplified version. For future work, we consider to integrate into our consideration the temporal information, to eliminate the jittering effects of our system when working with live streams.

Acknowledgements

This research was partially supported by A*STAR JCO and IAF grants. We would like to thank Vaghul Aditya Balaji for helping with the data collection and website design processes during his intern attachment at BII.

Proof of the Main Theorem

Before the formal proof of Theorem 1, We start with several intermediate results. These results can also be found in [8, 13], and are included here for completeness.

Proposition 1 *Suppose $\{r_n\}$ is a sequence of consistent regression estimates for a certain distribution of $(\overline{X}, \overline{Y})$. Then the sequence of averaging estimates $\left\{r_n^{(Z)}\right\}$ (for any value of Z) is also consistent.*

Proof Since $\{r_n\}$ is consistent, we have

$$\lim_{n\to\infty} \mathbb{E}_{\overline{X},\Psi,D_n}\left\{\|r_n(\overline{X},\Psi) - r(\overline{X})\|^2\right\} = 0.$$

Moreover,

$$\begin{aligned}
&\mathbb{E}_{\overline{X},\{\Psi_j\},D_n}\left\{\left\|r_n^{(Z)}(\overline{X}) - r(\overline{X})\right\|^2\right\} \\
&= \mathbb{E}_{\overline{X},\{\Psi_j\},D_n}\left\{\left\|\frac{1}{Z}\sum_{j=1}^{Z} r_n(\overline{X},\Psi_j) - r(\overline{X})\right\|^2\right\} \\
&\le \frac{1}{Z}\mathbb{E}_{\overline{X},\{\Psi_j\},D_n}\left\{\sum_{j=1}^{Z}\left\|r_n(\overline{X},\Psi_j) - r(\overline{X})\right\|^2\right\} \\
&= \mathbb{E}_{\overline{X},\Psi,D_n}\left\{\left\|r_n(\overline{X},\Psi) - r(\overline{X})\right\|^2\right\},
\end{aligned}$$

where we used the fact that $\left(\sum_{j=1}^{Z} a_j\right)^2 \le Z\sum_{j=1}^{Z} a_j^2$ in the inequality and that $\{\Psi_j\}_{j=1}^{Z}$ are i.i.d. random variables of Ψ in the last equality. Therefore, we have

$$\lim_{n\to\infty} \mathbb{E}_{\overline{X},\{\Psi_j\},D_n}\left\{\left\|r_n^{(Z)}(\overline{X}) - r(\overline{X})\right\|^2\right\} = 0,$$

that is, $\left\{r_n^{(Z)}\right\}$ is consistent.

Proposition 2 *Suppose $\{r_n\}$ is a sequence of regression estimates which are consistent* conditioned *on the partition random variable J for a certain distribution of $(\overline{X}, \overline{Y})$. That is,*

$$\lim_{n\to\infty} \mathbb{E}_{\overline{X},G,D_n}\left\{\|r_n(\overline{X},(J,G)) - r(\overline{X})\|^2 \mid J\right\} = 0,$$

for all J. Moreover, suppose $\|r(\overline{\mathbf{x}})\|$ is bounded and $\|r_n(\overline{\mathbf{x}},\Psi)\|$ is bounded with probability 1*. Then the sequence $\{r_n\}$ is* unconditionally *consistent.*

Proof Since

$$\begin{aligned}
&\mathbb{E}_{\overline{X},\Psi,D_n}\left\{\left\|r_n(\overline{X},\Psi) - r(\overline{X})\right\|^2\right\} \\
&= \mathbb{E}_J\left\{\mathbb{E}_{\overline{X},G,D_n}\left\{\left\|r_n(\overline{X},(J,G)) - r(\overline{X})\right\|^2 \mid J\right\}\right\},
\end{aligned}$$

and for any given J

$$\begin{aligned}
&\mathbb{E}_{\overline{X},G,D_n}\left\{\left\|r_n(\overline{X},(J,G)) - r(\overline{X})\right\|^2\right\} \\
&\le 2\mathbb{E}_{\overline{X},G,D_n}\left\{\|r_n(\overline{X},(J,G)\|^2\right\} + 2\mathbb{E}_{\overline{X}}\left\{\|r(\overline{X})\|^2\right\} \\
&\le 2\sup_{\overline{\mathbf{x}}}\mathbb{E}_{G,D_n}\left\{\|r_n(\overline{\mathbf{x}},(J,G)\|^2\right\} + 2\sup_{\overline{\mathbf{x}}}\|r(\overline{\mathbf{x}})\|^2,
\end{aligned}$$

where the last inequality is derived from the boundedness assumption, it follows from the dominated convergence theorem that

$$\begin{aligned}
&\lim_{n\to\infty}\mathbb{E}_{\overline{X},\Psi,D_n}\left\{\left\|r_n(\overline{X},\Psi) - r(\overline{X})\right\|^2\right\} \\
&= \mathbb{E}_J\left\{\lim_{n\to\infty}\mathbb{E}_{\overline{X},G,D_n}\left\{\left\|r_n(\overline{X},(J,G)) - r(\overline{X})\right\|^2 \mid J\right\}\right\} \\
&= 0.
\end{aligned}$$

To summarize, Proposition 1 states that the consistency of the regression forests is implied by the consistency of the trees it contains. Proposition 2 states that proving the consistency conditioned on the partition random variable J is sufficient for the consistency of each tree. As a preparation, we also need the following lemma.

Lemma 1 *If $\{\overline{X}_i\}_{i=1}^m$ is a set of i.i.d. $\mathcal{U}(0,a)$ random variables, then*

$$\mathbb{E}\left\{\max\left(\max_{1\le i\le m}\overline{X}_i,\, a - \min_{1\le i\le m}\overline{X}_i\right)\right\} = \frac{2m+1}{2(m+1)}a$$

Proof Let $\overline{X} = \max\left(\max_{1\le i\le m}\overline{X}_i,\, a - \min_{1\le i\le m}\overline{X}_i\right)$, then the cumulative distribution function (cdf) of $\overline{X}$ is given by

$$\begin{aligned}
F_{\overline{X}}(x) =& \mathbb{P}(\overline{X} \le x) \\
=& \mathbb{P}\left(\max_{1\le i\le m}\overline{X}_i \le x,\, \min_{1\le i\le m}\overline{X}_i \ge a - x\right) \\
=& \prod_{1\le i\le m}\mathbb{P}\left(a - x \le \overline{X}_i \le x\right) \\
=& \left(\frac{2x}{a} - 1\right)^m
\end{aligned}$$

for all $\frac{a}{2} \le x \le a$. The probability density function (pdf) of $\overline{X}$ is then

$$f_{\overline{X}}(x) = F'_{\overline{X}}(x) = \frac{2m}{a}\left(\frac{2x}{a} - 1\right)^{m-1},$$

which implies

$$\mathbb{E}\left\{\overline{X}\right\} = \int_{a/2}^{a} x f_{\overline{X}}(x)dx = \frac{2m+1}{2(m+1)}a.$$

A few more notations need to be in place: Let $K_n(\overline{X},\Psi)$ denote the number of splits required to get the leaf node $A_n(\overline{X},\Psi)$, $V_{nj}(\overline{X},\Psi)$ be the size of the j-th dimension of $A_n(\overline{X},\Psi)$ and $A_n^{(t)}(\overline{X},\Psi)$ as the $(t+1)$-th node in the path from the root to $A_n(\overline{X},\Psi)$. Also denote as $V_{nj}^{(t)}(\overline{X},\Psi)$ the size of the j-th dimension of $A_n^{(t)}(\overline{X},\Psi)$, and $N_{ns}^{(t)}(\overline{X},\Psi)$ as well as $N_{ne}^{(t)}(\overline{X},\Psi)$ the number of structure and estimation data in $A_n^{(t)}(\overline{X},\Psi)$, respectively.

Moreover, we use $M_{n_s}^{(t)}(\overline{X},\Psi)$ to denote the number of structure data selected to choose candidate threshold set $\mathcal{T}$ in node $A_n^{(t)}(\overline{X},\Psi)$. Since $M_{n_s}^{(t)}(\overline{X},\Psi)$ is set as the smaller value between $N_{n_s}^{(t)}(\overline{X},\Psi)$ and a user-specified integer m_0, it follows that

$$M_{n_s}^{(t)}(\overline{X},\Psi) \le m_0$$

always holds. With the above results in hand, we are now ready to prove the main consistency theorem of our regression forests. Although our proof closely follows that of [13], it still bears many differences due to the usage of different stopping criteria.

Proof From Proposition 1 and Proposition 2, we know that to prove the consistency of $\{r_n^{(Z)}\}$ it is sufficient to prove the consistency of each tree conditioned on random variable J. To this end, we appeal to a general consistency theorem (the Stone's theorem) for partitioning estimate [19]. According to this theorem, it is sufficient to show that

$$N_{n_e}(\overline{X},\Psi) \to \infty \quad \text{and} \quad diam(A_n(\overline{X},\Psi)) \to 0$$

in probability[3].

Consider a tree defined by random variable Ψ conditioned on J (i.e., defined by the given structured data and random variable G). First we show that $N_{n_e}(\overline{X},\Psi) \to \infty$ in probability. Suppose there are Q_n leaf nodes in the tree denoted by $A_1,\cdots,A_{Q_n}$, then by the construction of the tree we have $Q_n \le 2^{\lceil \log_2 L_n \rceil} \le 2L_n$. Let $\mathcal{S} = \{\overline{X}\} \cup \{\overline{X}_i | \overline{X}_i \in E_n\}$ denote the n_e+1 points composed of the union of $\overline{X}$ and all estimation data, and $N_1,\cdots,N_{Q_n}$ denote the number of points of $\mathcal{S}$ falling in those leaf nodes. Since all data points are i.i.d., we have $\mathbb{P}\{\overline{X} \in A_i | \mathcal{S},\Psi\} = N_i/(n_e+1)$. That is, given the set $\mathcal{S}$ and Ψ, the conditional probability that $\overline{X}$ falls in the i-th leaf node is $N_i/(n_e+1)$. Therefore, for every fixed $c > 0$,

$$\begin{aligned}
\mathbb{P}\{N_{n_e}(\overline{X},\Psi) < c\} =& \mathbb{E}\{\mathbb{P}\{N_{n_e}(\overline{X},\Psi) < c \mid \mathcal{S},\Psi\}\} \\
=& \mathbb{E}\left\{ \sum_{i:N_i<c} \mathbb{P}\{\overline{X} \in A_i \mid \mathcal{S},\Psi\} \right\} \\
=& \mathbb{E}\left\{ \sum_{i:N_i<c} \frac{N_i}{n_e+1} \right\} \\
\le& \frac{cQ_n}{n_e+1} \le \frac{2cL_n}{n_e+1} \le \frac{4cL_n}{n+2},
\end{aligned}$$

which converges to zero by the assumption that $\frac{L_n}{n} \to 0$.

To show that $diam(A_n(\overline{X},\Psi)) \to 0$ in probability, it suffices to show that $V_{nj}(\bar{\mathbf{x}},\Psi) \to 0$ in probability for all $j \in \{1,2,\cdots,d\}$ and all $\bar{\mathbf{x}}$ in the support of $\overline{X}$, so it is enough to show that $\mathbb{E}\{V_{nj}(\bar{\mathbf{x}},\Psi)\} \to 0$ for all $\bar{\mathbf{x}}$. Consider the node $A_n^{(t)}(\bar{\mathbf{x}},\Psi)$ and let $\overline{X}_1,\cdots,\overline{X}_{N_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi)}$ denote all structure data in this node and $\overline{X}_i^{(j)}$ be the size of the j-th dimension of $\overline{X}_i$. Then

$$\overline{X}_i^{(j)} \sim \mathcal{U}(0, V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi)),\ i = 1,\cdots,N_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi),\ j = 1,2,\cdots,d.$$

Provided that the j-th dimension is split, and without loss of generality we assume $\left\{\overline{X}_i^{(j)}\right\}_{i=1}^{M_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi)}$ are selected to choose the threshold, then the size of the j-th dimension of the child nodes is bounded by

$$\begin{aligned}
&\bar{V}_{nj}^{(t+1)}(\bar{\mathbf{x}},\Psi) \\
&= \max\left\{ \max_{1\le i\le M_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi)} \overline{X}_i^{(j)},\ V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi) - \min_{1\le i\le M_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi)} \overline{X}_i^{(j)} \right\}.
\end{aligned}$$

[3] $diam(A) = \sup_{\bar{\mathbf{x}}_1,\bar{\mathbf{x}}_2\in A} \|\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2\|$ for a set $A \subset \mathbb{R}^d$.

By Lemma 1, we have

$$\mathbb{E}\left\{\bar{V}_{nj}^{(t+1)}(\bar{\mathbf{x}},\Psi)\right\} = \frac{2M_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi)+1}{2(M_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi)+1)} V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi).$$

Define the following two events E_1 ={There is only one candidate feature} and E_2 ={The j-th dimension is a candidate feature}. Since we randomly select s candidate features and $s \sim 1 + Binomial(d-1,p)$, it follows that $\mathbb{P}\{E_1 \cap E_2\} = \mathbb{P}\{E_2|E_1\}\mathbb{P}\{E_1\} = (1-p)^{d-1}/d$. Thus,

$$\begin{aligned}
&\mathbb{E}\left\{V_{nj}^{(t+1)}(\bar{\mathbf{x}},\Psi)\right\} \\
&\le \mathbb{E}\left\{\mathbb{1}_{\{(E_1\cap E_2)^c\}} V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi) + \mathbb{1}_{\{E_1\cap E_2\}} \bar{V}_{nj}^{(t+1)}(\bar{\mathbf{x}},\Psi)\right\} \\
&= (1-\mathbb{P}\{E_1\cap E_2\})\mathbb{E}\left\{V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi)\right\} + \mathbb{P}\{E_1\cap E_2\}\mathbb{E}\left\{\bar{V}_{nj}^{(t+1)}(\bar{\mathbf{x}},\Psi)\right\} \\
&= \left(1 - \frac{(1-p)^{d-1}}{2d(M_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi)+1)}\right)\mathbb{E}\left\{V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi)\right\},
\end{aligned}$$

which implies that

$$\begin{aligned}
&\mathbb{E}\{V_{nj}(\bar{\mathbf{x}},\Psi)\} \\
&= \mathbb{E}\left\{V_{nj}^{(K_n(\bar{\mathbf{x}},\Psi))}(\bar{\mathbf{x}},\Psi)\right\} \\
&\le \prod_{t=0}^{K_n(\bar{\mathbf{x}},\Psi)-1} \left(1 - \frac{(1-p)^{d-1}}{2d(M_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi)+1)}\right)\mathbb{E}\left\{V_{nj}^{(0)}(\bar{\mathbf{x}},\Psi)\right\} \\
&\le \left(1 - \frac{(1-p)^{d-1}}{2d(m_0+1)}\right)^{K_n(\bar{\mathbf{x}},\Psi)},
\end{aligned}$$

where in the last inequality we used $\mathbb{E}\left\{V_{nj}^{(0)}(\bar{\mathbf{x}},\Psi)\right\} = 1$ and $M_{n_s}^{(t)}(\bar{\mathbf{x}},\Psi) \le m_0$.

Since $\left(1 - \frac{(1-p)^{d-1}}{2d(m_0+1)}\right) < 1$ and is independent of n, it remains to show that $K_n(\bar{\mathbf{x}},\Psi) \to \infty$ in probability. For the leaf node $A_n(\overline{X},\Psi)$, the ending of splitting can be resulted from either the tree has reached the maximum level $\lceil \log_2 L_n \rceil$ or further splitting will results in less than k_n estimation data in its children. If the former is the case, then $K_n(\bar{\mathbf{x}},\Psi) = \lceil \log_2 L_n \rceil - 1 \to \infty$ as $n \to \infty$. Therefore, in the rest of the proof, we assume that further splitting of $A_n(\overline{X},\Psi)$ will results in less than k_n estimation data in its child nodes, and show that $K_n(\bar{\mathbf{x}},\Psi) \to \infty$ in probability. The idea is to prove that for any fixed $T \ge 1$ the probability $\mathbb{P}\{K_n(\bar{\mathbf{x}},\Psi) < T\}$ approaches zero as $n \to \infty$.

For any fixed $T \ge 1$ and $0 < \delta < (\frac{1}{2})^T$, we have by Lemma 1

$$\begin{aligned}
&\mathbb{P}\left\{\bar{V}_{nj}^{(t)}(\bar{\mathbf{x}},\Psi) \le (1-\delta^{\frac{1}{T}})V_{nj}^{(t-1)}(\bar{\mathbf{x}},\Psi) \mid V_{nj}^{(t-1)}(\bar{\mathbf{x}},\Psi)\right\} \\
&= \left(2(1-\delta^{\frac{1}{T}})-1\right)^{M_{n_s}^{(t-1)}(\bar{\mathbf{x}},\Psi)},\ \forall 1 \le t \le T,\ 1 \le j \le d.
\end{aligned}$$

In other words, the size of the j-th coordinate of the child nodes of $A_n^{(t-1)}(\bar{X},\Psi)$ is at least $\delta^{\frac{1}{T}} V_{nj}^{(t-1)}(\bar{\mathbf{x}},\Psi)$ with probability $(2(1-\delta^{\frac{1}{T}})-1)^{M_{n_s}^{(t-1)}(\bar{\mathbf{x}},\Psi)}$, that is,

$$\begin{aligned}
&\mathbb{P}\left\{V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi) \ge \delta^{\frac{1}{T}} V_{nj}^{(t-1)}(\bar{\mathbf{x}},\Psi) \mid V_{nj}^{(t-1)}(\bar{\mathbf{x}},\Psi)\right\} \\
&= \left(2(1-\delta^{\frac{1}{T}})-1\right)^{M_{n_s}^{(t-1)}(\bar{\mathbf{x}},\Psi)},\ \forall 1 \le t \le T,\ 1 \le j \le d,
\end{aligned}$$

which implies that

$$
\begin{aligned}
&\mathbb{P}\left\{V_{nj}^{(T)}(\bar{\mathbf{x}},\Psi)\geq\delta\right\}\\
\geq&\mathbb{P}\left\{V_{nj}^{(T)}(\bar{\mathbf{x}},\Psi)\geq\delta^{\frac{1}{T}}V_{nj}^{(T-1)}(\bar{\mathbf{x}},\Psi)\mid V_{nj}^{(T-1)}(\bar{\mathbf{x}},\Psi)\geq\delta^{\frac{T-1}{T}}\right\}\\
&\times\mathbb{P}\left\{V_{nj}^{(T-1)}(\bar{\mathbf{x}},\Psi)\geq\delta^{\frac{T-1}{T}}\right\}\\
&\vdots\text{(recursively apply the above inequality to } T-1,T-2,\ldots)\\
\geq&\prod_{t=0}^{T-1}\mathbb{P}\left\{V_{nj}^{(t+1)}(\bar{\mathbf{x}},\Psi)\geq\delta^{\frac{1}{T}}V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi)\mid V_{nj}^{(t)}(\bar{\mathbf{x}},\Psi)\geq\delta^{\frac{t}{T}}\right\}\\
&\times\mathbb{P}\left\{V_{nj}^{(0)}(\bar{\mathbf{x}},\Psi)\geq 1\right\}\\
\geq&\prod_{t=0}^{T-1}\left(2(1-\delta^{\frac{1}{T}})-1\right)^{M_{ns}^{(t)}(\bar{\mathbf{x}},\Psi)}\\
\geq&\left(2(1-\delta^{\frac{1}{T}})-1\right)^{Tm_0},
\end{aligned}
$$

where we used $\mathbb{P}\left\{V_{nj}^{(0)}(\bar{\mathbf{x}},\Psi)\geq 1\right\}=1$ in the last inequality and $M_{ns}^{(t)}(\bar{\mathbf{x}},\Psi)\leq m_0$ in the last equality.

Thus, for any $\epsilon_1>0$, we can select δ satisfying

$$
0<\delta^{\frac{1}{T}}\leq 1-\frac{1}{2}\left((1-\epsilon_1)^{\frac{1}{dTm_0}}+1\right)<\frac{1}{2}
$$

so that

$$
\mathbb{P}\left\{V_{nj}^{(T)}(\bar{\mathbf{x}},\Psi)\geq\delta\right\}\geq(1-\epsilon_1)^{1/d},\ \forall 1\leq j\leq d
$$

and know that $A_n^{(T)}(\bar{\mathbf{x}},\Psi)$ contains a hypercube with sides of length δ with probability at least $1-\epsilon_1$.

Let $\tilde{A}_n^{(T)}(\bar{\mathbf{x}},\Psi)$ denote the sibling node of $A_n^{(T)}(\bar{\mathbf{x}},\Psi)$ (i.e., $\tilde{A}_n^{(T)}(\bar{\mathbf{x}},\Psi)$ and $A_n^{(T)}(\bar{\mathbf{x}},\Psi)$ are two child nodes of $A_n^{(T-1)}(\bar{\mathbf{x}},\Psi)$) and $\tilde{N}_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)$ be the number of estimation data it contains. Then, following the same procedure as the above, we can show that $\tilde{A}_n^{(T)}(\bar{\mathbf{x}},\Psi)$ contains a hypercube with sides of length δ with probability at least $1-\epsilon_1$. Let

$$
\kappa:=\min\left\{\mu(A_n^{(T)}(\bar{\mathbf{x}},\Psi)),\mu(\tilde{A}_n^{(T)}(\bar{\mathbf{x}},\Psi))\right\},
$$

we know that $\mathbb{P}\{\kappa>0\}=1$ since for any $\epsilon_1>0$, we can find $\delta>0$ such that both $A_n^{(T)}(\bar{\mathbf{x}},\Psi)$ and $\tilde{A}_n^{(T)}(\bar{\mathbf{x}},\Psi)$ contains a hypercube with sides of length δ (i.e. $\kappa\geq\delta^d$) with probability at least $1-\epsilon_1$.

Next, we show that for any $\epsilon_2>0$ we can choose sufficiently large n such that

$$
\mathbb{P}\{N_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)<k_n\}\leq\frac{\epsilon_2}{2}.
$$

Notice that for any hypercube of volume $\kappa>0$ in $[0,1]^d$, the number of estimation data is contains denoted as N_e satisfies $N_e\sim Binomial(n_e,\kappa)$, by Hoeffding's tail inequality we have

$$
\mathbb{P}\{N_e<k_n\}\leq\exp\left(-\frac{2(n_e\kappa-k_n)^2}{n_e}\right)=\exp\left(-2n_e\left(\kappa-\frac{k_n}{n_e}\right)^2\right).
$$

Since $\frac{k_n}{n}\to 0$ and $n_e\to\infty$ as $n\to\infty$, it follows that for any $\epsilon_2>0$ we have

$$
\frac{k_n}{n_e}\leq\frac{2k_n}{n}\leq\kappa-\sqrt{\frac{1}{2n_e}\ln\left(\frac{2}{\epsilon_2}\right)},
$$

for sufficiently large n, which together with the inequality above implies that

$$
\mathbb{P}\{N_e<k_n\}\leq\frac{\epsilon_2}{2},
$$

which further implies that

$$
\begin{aligned}
\mathbb{P}\{N_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)<k_n\}=&\mathbb{P}\{N_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)<k_n\mid\kappa>0\}\\
\leq&\mathbb{P}\{N_e<k_n\}\leq\frac{\epsilon_2}{2}.
\end{aligned}
$$

Similarly, we have

$$
\mathbb{P}\{\tilde{N}_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)<k_n\}\leq\frac{\epsilon_2}{2}.
$$

Remember that we assume that the further splitting of $A_n(\bar{X},\Psi)$ would result in child node containing fewer than k_n estimation data. Therefore,

$$
\begin{aligned}
\mathbb{P}\{K_n(\bar{\mathbf{x}},\Psi)<T\}\leq&\mathbb{P}\{N_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)<k_n\ or\ \tilde{N}_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)<k_n\}\\
\leq&\mathbb{P}\{N_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)<k_n\}+\mathbb{P}\{\tilde{N}_{n_e}^{(T)}(\bar{\mathbf{x}},\Psi)<k_n\}\\
\leq&\epsilon_2,
\end{aligned}
$$

for sufficiently large n.

In summary, we have proven that for both stopping criteria in our tree construction $K_n(\bar{\mathbf{x}},\Psi)\to\infty$ in probability, which completes the proof.

References

1. ShapeHand. http://www.shapehand.com/shapehand.html (2009)
2. Kinect. http://www.xbox.com/en-US/kinect/ (2011)
3. Softkinetic. http://www.softkinetic.com (2012)
4. Leapmotion. http://www.leapmotion.com (2013)
5. Andrews, H.C., Patterson, C.L.: Digital interpolation of discrete images. Computers, IEEE Transactions on **C-25**(2), 196–202 (1976)
6. Ballan, L., Taneja, A., Gall, J., Gool, L., Pollefeys, M.: Motion capture of hands in action using discriminative salient points. In: ECCV (2012)
7. Biau, G.: Analysis of a random forests model. Journal on Machine Learning Research **13**, 1063–1095 (2012)
8. Biau, G., Devroye, L., Lugosi, G.: Consistency of random forests and other averaging classifiers. Journal on Machine Learning Research **9**, 2015–2033 (2008)
9. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
10. Breiman, L.: Consistency for a simple random forests. Tech. rep., UC Berkeley (2004)
11. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**, 603–619 (2002)
12. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893 (2005)
13. Denil, M., Matheson, D., de Freitas, N.: Narrowing the gap: Random forests in theory and practice. In: ICML (2014)
14. Erol, A., Bebis, G., Nicolescu, M., Boyle, R., Twombly, X.: Vision-based hand pose estimation: A review. Comput. Vis. Image Underst. **108**(1-2), 52–73 (2007)
15. Fanelli, G., Gall, J., Gool, L.V.: Real time head pose estimation with random regression forests. In: CVPR (2011)
16. Gall, J., Lempitsky, V.: Class-specific hough forests for object detection. In: Decision Forests for Computer Vision and Medical Image Analysis, pp. 143–157. Springer (2013)
17. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. In: ICCV (2011)

18. Gustus, A., Stillfried, G., Visser, J., Jorntell, H., van der Smagt, P.: Human hand modelling: kinematics, dynamics, applications. Biological Cybernetics **106**(11-12), 741–755 (2012)
19. Gyröfi, L., Kohler, M., Krzyzak, A., Walk, H.: A Distribution-Free Theory of Nonparametric Regression. Springer (2002)
20. Hackenberg, G., McCall, R., Broll, W.: Lightweight palm and finger tracking for real-time 3d gesture control. In: IEEE Virtual Reality Conference, pp. 19–26 (2011)
21. Hamming, R.: Digital Filters, 3 edn. Dover Publications (1997)
22. Hansard, M., Lee, S., Choi, O., Horaud, R.: Time-of-Flight Cameras: Principles, Methods and Applications. Springer (2013)
23. Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., Navab, N.: Dominant orientation templates for real-time detection of textureless objects. In: CVPR (2010)
24. Keskin, C., Kirac, F., Kara, Y., Akarun, L.: Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In: ECCV (2012)
25. de La Gorce, M., Fleet, D., Paragios, N.: Model-based 3d hand pose estimation from monocular video. IEEE Trans. Pattern Anal. Mach **33**(9), 1793–1805 (2011)
26. Lewis, J.: Fast normalized cross-correlation. In: Vision interface, vol. 10, pp. 120–123 (1995)
27. Melax, S., Keselman, L., Orsten, S.: Dynamics based 3d skeletal hand tracking. In: Graphics Interface (2013)
28. Oikonomidis, I., Lourakis, M., Argyros, A.: Evolutionary quasi-random search for hand articulations tracking. In: CVPR (2014)
29. Oikonomidis, N., Argyros, A.: Efficient model-based 3d tracking of hand articulations using kinect. In: BMVC (2011)
30. Peachey, D.: Texture on demand. Tech. rep. (1990)
31. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR (2011)
32. Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., Moore, R.: Real-time human pose recognition in parts from single depth images. Comm. of ACM **56**(1), 116–124 (2013)
33. Sridhar, S., Oulasvirta, A., Theobalt, C.: Interactive markerless articulated hand motion tracking using rgb and depth data. In: ICCV (2013)
34. Sueda, S., Kaufman, A., Pai, D.: Musculotendon simulation for hand animation. In: SIGGRAPH, pp. 83:1–83:8 (2008)
35. Tang, D., Tejani, A., Chang, H., Kim, T.: Latent regression forest: Structured estimation of 3d articulated hand posture. In: CVPR (2014)
36. Taylor, J., Stebbing, R., Ramakrishna, V., Keskin, C., Shotton, J., Izadi, S., Fitzgibbon, A., Hertzmann, A.: User-specific hand modeling from monocular depth sequences. In: CVPR (2014)
37. Tzionas, D., Gall, J.: A comparison of directional distances for hand pose estimation. In: German Conference on Pattern Recognition (2013)
38. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Trans. on Pattern Analysis and Machine Intelligence **13**, 376380 (1991)
39. Wang, R., Popović, J.: Real-time hand-tracking with a color glove. In: SIGGRAPH, pp. 63:1–63:8 (2009)
40. Xu, C., Cheng, L.: Efficient hand pose estimation from a single depth image. In: ICCV (2013)
41. Ye, M., Zhang, Q., Wang, L., Zhu, J., Yang, R., Gall, J.: Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications, chap. A Survey on Human Motion Analysis from Depth Data, pp. 149–187. Springer (2013)
42. Zhao, W., Chai, J., Xu, Y.: Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data. In: Eurographics Symposium on Computer Animation (2012)

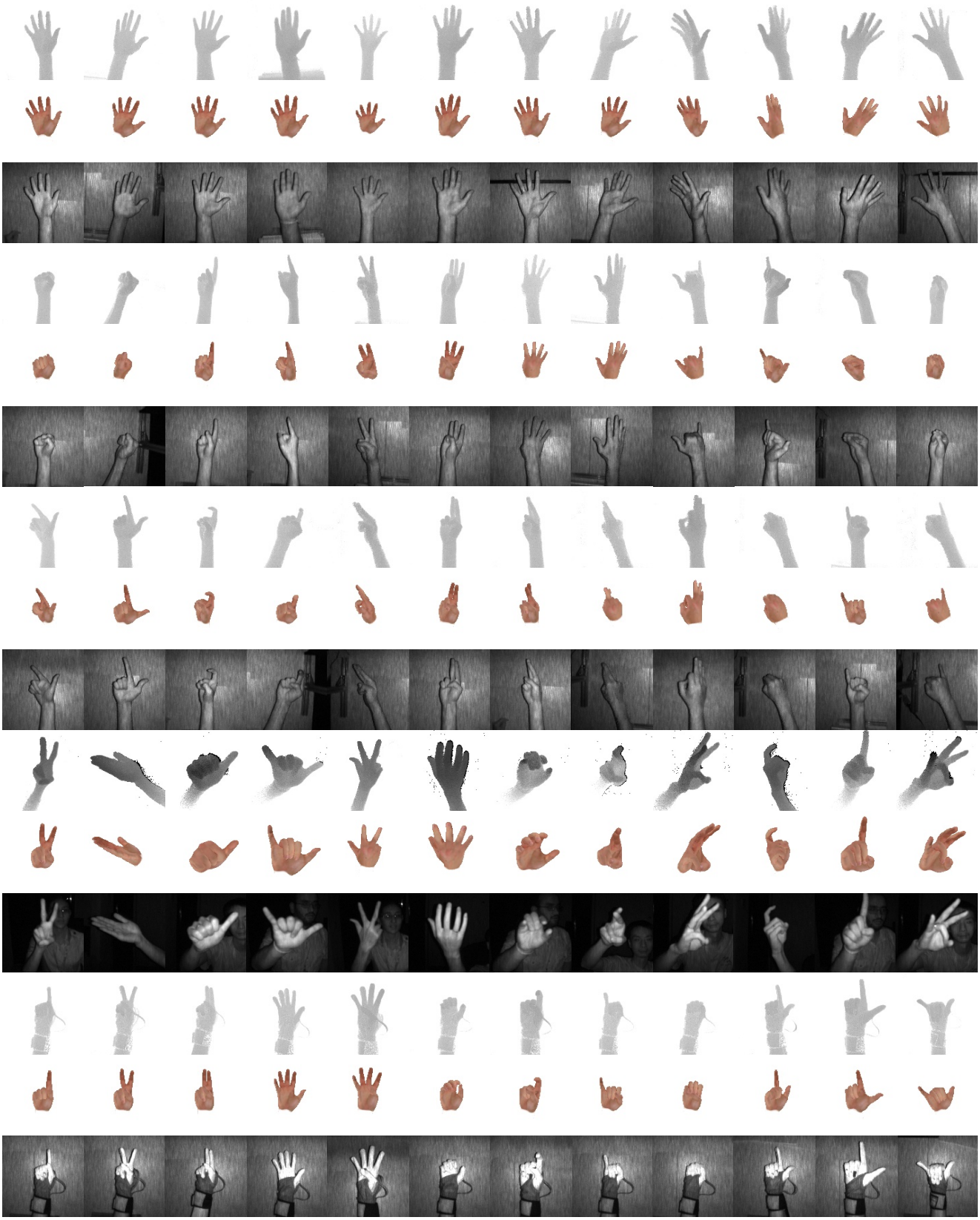

Fig. 20: Pose estimation results of *DHand* on real-world depth images of diverse gestures, orientations, and hand sizes. The first three rows present the results of hands from different subjects with roughly the same gesture. The next six rows showcase various gestures under varied orientations. The following three rows are captured instead from a frontal camera and with changing distance (instead of the default 500mm distance). The last three rows display our results on gloved hands.

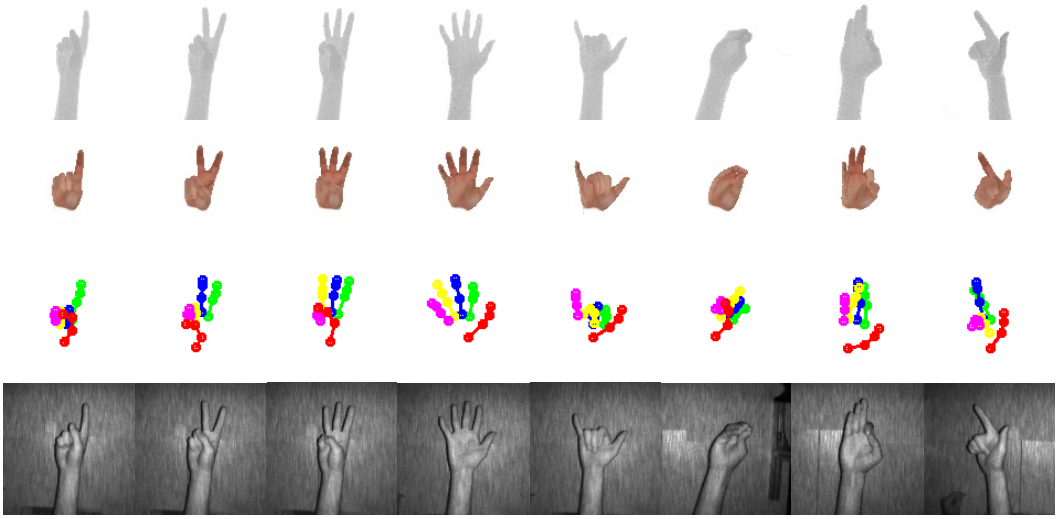

Fig. 22: Comparison of *DHand* with LRF of [35] for hand pose estimation. Presented as input depth images in the first row, the second row displays the results of *DHand*, while the third row shows the corresponding results of [35], visualized by the finger joints (red for thumb, green for index finger, blue for middle finger, yellow for ring finger, and purple for little finger). The corresponding amplitude images are also provided in the fourth row as a reference.

Fig. 23: Comparison of *DHand* with two state-of-the-art tracking methods: [29], and Leap Motion [4]. First and second rows present the input depth images and results of *DHand*, while the third and fourth rows displays the corresponding results of [29] and [4]. See text for details.

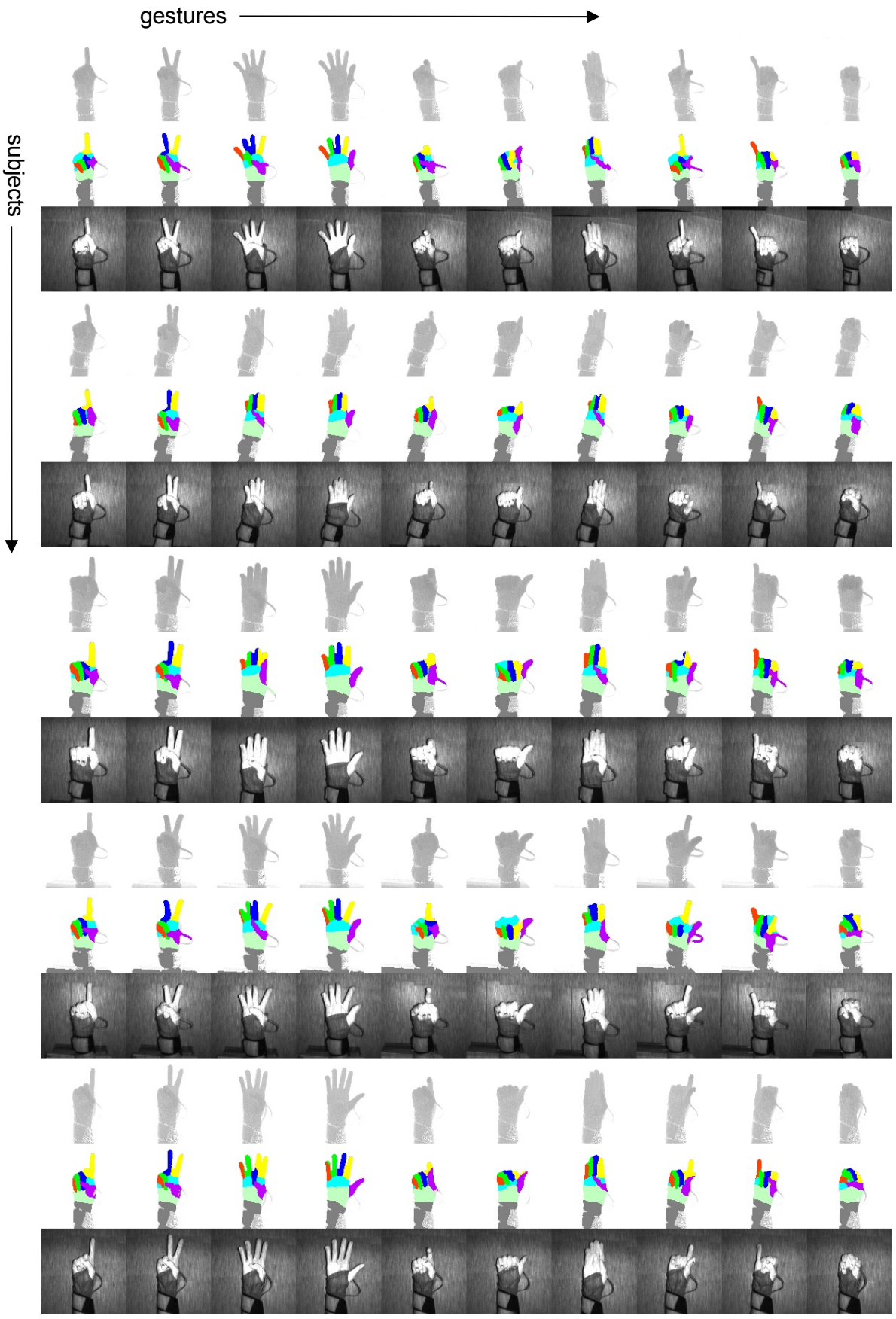

Fig. 24: Part-based labeling of hands. Each fingers and different parts of the palm are labeled by distinct colors, following the color code of Fig. 10. See text for details.

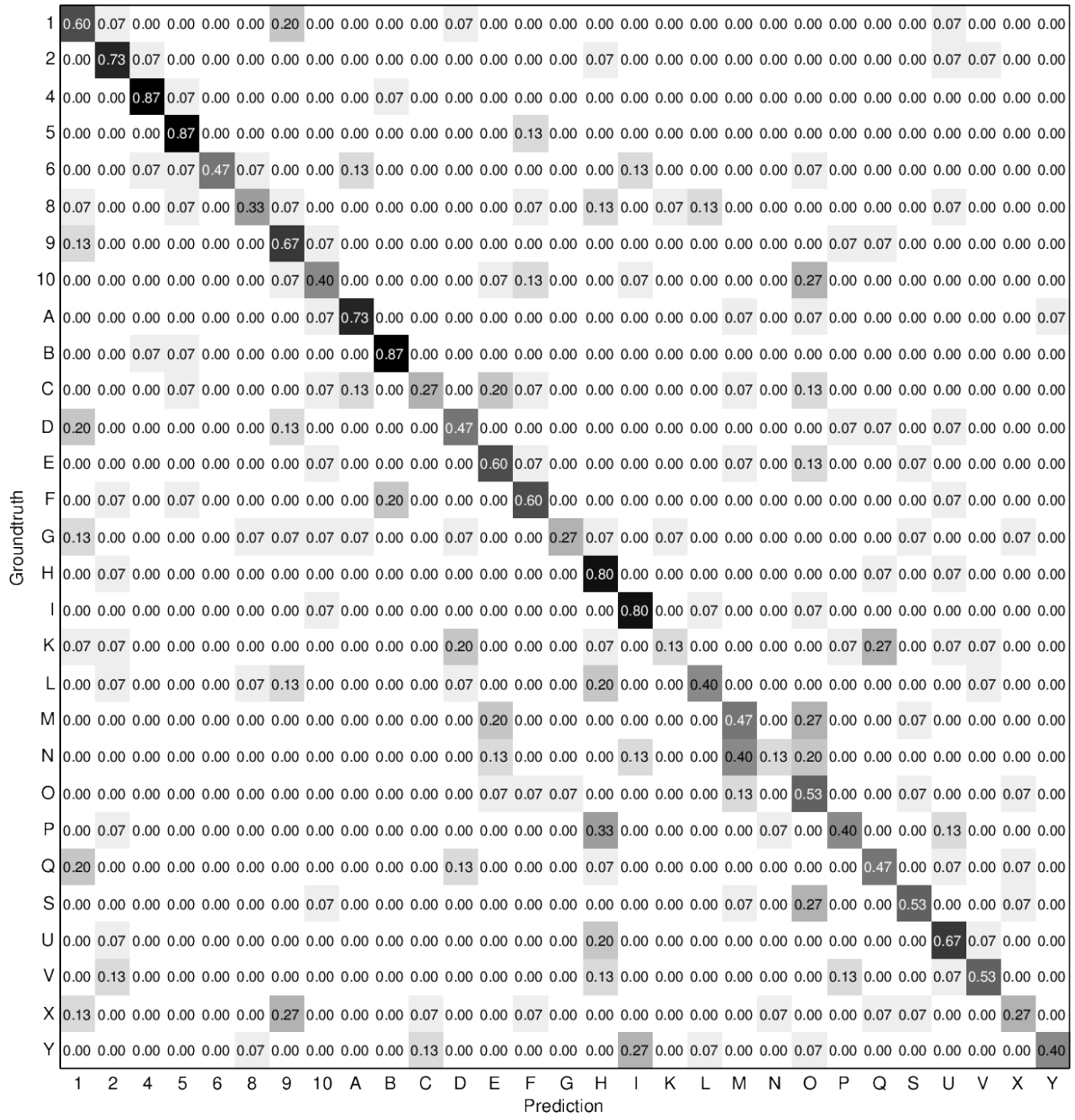

Fig. 25: Confusion matrix of the hand gesture recognition experiment using *DHand*.