3D Hand Pose Estimation Using Randomized Decision Forest with Segmentation Index Points

Peiyi Li [1,2] Haibin Ling [2,*] Xi Li [3] Chunyuan Liao [1]
[1]Meitu HiScene Lab, HiScene Information Technologies, Shanghai, China
[2]Computer and Information Sciences Department, Temple University, Philadelphia, PA USA
[3]College of Computer Science, Zhejiang University, Hangzhou, China
lipy@hiscene.com, xilizju@zju.edu.cn, liaocy@hiscene.com, hbling@temple.edu

Abstract

In this paper, we propose a real-time 3D hand pose estimation algorithm using the randomized decision forest framework. Our algorithm takes a depth image as input and generates a set of skeletal joints as output. Previous decision forest-based methods often give labels to all points in a point cloud at a very early stage and vote for the joint locations. By contrast, our algorithm only tracks a set of more flexible virtual landmark points, named segmentation index points (SIPs)*, before reaching the final decision at a leaf node. Roughly speaking, an SIP represents the centroid of a subset of skeletal joints, which are to be located at the leaves of the branch expanded from the SIP. Inspired by recent latent regression forest-based hand pose estimation framework (Tang et al. 2014), we integrate SIP into the framework with several important improvements: First, we devise a new forest growing strategy, whose decision is made using a randomized feature guided by SIPs. Second, we speed-up the training procedure since only SIPs, not the skeletal joints, are estimated at non-leaf nodes. Third, the experimental results on public benchmark datasets show clearly the advantage of the proposed algorithm over previous state-of-the-art methods, and our algorithm runs at 55.5 fps on a normal CPU without parallelism.*

1. Introduction

Skeleton detection and pose estimation on highly articulated object are always a challenging topic in computer vision. For example, accurate estimation for hand gesture or human body pose plays a very important role in human-computer interaction. Because of the practical value associated with this topic, it has been attracting efforts from both the industry and academia. In the past few years, by using high-speed depth sensor [1] at a low cost, the application of real-time body pose estimation has become a reality in domestic use. Since then, human body pose estimation has received an increasing amount of attention. With the provision of new low-cost source input, depth imagery, many new algorithms have outperformed traditional RGB-based human pose estimation algorithms. Similar observa-

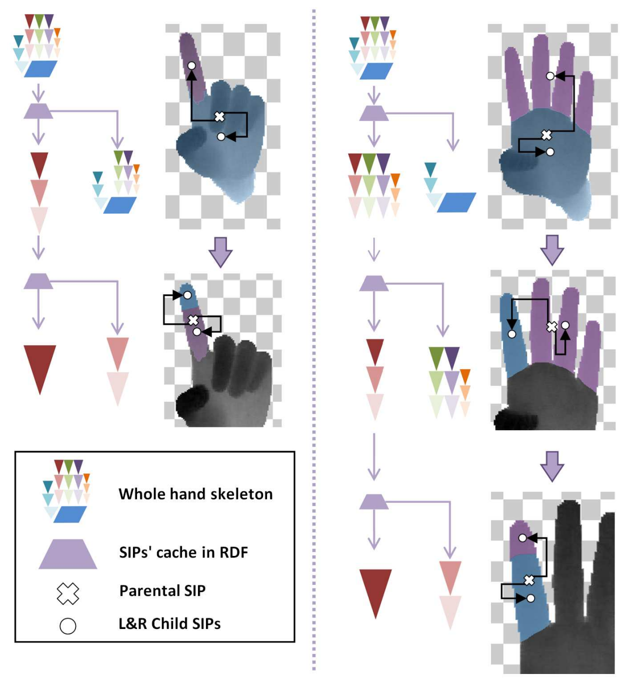


Figure 1. Our algorithm can be seen as a divide-and-conquer search process. We recursively cluster skeletal joints and divide the part under exploration into two finer sub-regions until reaches a leaf node, which represents the position of a skeletal joint. This figure shows two examples of locating the index finger tip. Different hand posture leads to different hand sub-region segmentation, therefore different SIPs and different tree structure. For simplicity, we only show the searching process for one joint. This figure is best viewed in color.

* Corresponding author.

tions have been reported for hand pose estimation as well (e.g, [21, 2, 22, 26, 31, 23]).

Compared with human pose [5, 6], hand pose usually has a higher degree of freedom and articulation. Hand pose estimation [12, 9, 11] is also subject to practical challenges due to, for example, frequent self-occlusion, viewpoint changes, low spatial input resolution, and data noises. Besides, real-time performance is often desired in many applications.

The use of *randomized decision forest* (RDF) and its variations is largely popularized by their applications in human pose estimation (e.g., [24, 4, 17, 14, 25, 30]). The idea is later adopted by researchers [13, 15, 27, 28] working on hand pose estimation, which is more challenging as discussed above. In particular, Tang *et al*. [27] proposed using *Latent Regression Forest* (LRF) framework, containing an ensemble of randomized binary decision trees, for hand pose estimation. A key innovation in their algorithm is the use of a *Latent Tree Model* (LTM) to guide the search process of skeletal joints during tree inference, and LTM is learnt in an unsupervised data-driven manner.

Inspired by but different from [27], the hand pose estimation algorithm proposed in this paper also uses RDF as the basic framework. Instead of using LTM, we introduce a novel idea, *Segmentation Index Points* (SIPs), to guide the searching process.

Roughly speaking, an SIP represents the centroid of a subset of skeletal joints, which are to be located at the leaves of the branch expanded, directly or recursively, from the SIP. We integrate SIP into the framework with two major improvements: First, we devise a new forest growing strategy, in which SIPs are used to adaptively guide the skeletal joints partition and randomized feature selection. Second, we speed-up the training procedure, since at most two SIPs, not the skeletal joints, are estimated in non-leaf nodes.

Compared with LTM that is pre-learnt and fixed for different hand poses, SIP is more flexible and automatically adjusts to pose variations. As a result, SIP effectively accommodates large pose distortion and articulation. The promising experiment results of our method on the public benchmark dataset [27] show clearly its advantage over recently proposed state-of-the-art methods. In addition, running at 55.5 fps, our method is friendly ready for many real-time applications.

Our framework has several advantages:

- The use of SIPs allows decision forest to defer the final skeleton localization to a later stage (i.e., terminating leaves), thus avoids more information to be accumulated for more precise decision.
- SIPs are trained in an unsupervised fashion based on the clustering results of 3D skeletal joints, and therefore is self-adaptive to different pose variations.
- The new forest growing procedure is guided by a randomized distinctive binary feature depending on SIPs, which do not rely on specific hand models.

In the rest of the paper, we first discuss related work in Sec. 2. Then, we show the training and testing of the proposed method in Sec. 3. After that, we report the experiment in Sec. 4. Finally, in Sec. 5, we draw our conclusions.

2. Related Studies

Hand pose estimation has caught computer vision's attention in recent few years due to the breakthrough in cheap fast RGBD sensor [1] and its corresponding randomized decision forest (RDF) based algorithm [24]. The algorithm of human body pose estimation is attributed to the work by Shotton et al. [24]. They used an RDF [4] to assign labels for all pixels of a human body, and furthermore to infer the location of body joints. Their work is further improved by Girshick *et al*. [8]. They used a same RDF methodology, but each labeled pixel will propose a shift vector for each joint, and the mean shift vector will derive the final joints location. This methodology greatly increased the accuracy of internal body joints even if they are occluded. Besides, model based methods are also mainstream, which a well-define articulated body model is used, and it is tracked by different filters [17, 14, 25].

Due to the similarity between human body pose and hand pose, researchers have applied related technologies to hand pose estimation. According to the review [7] we can categorize algorithms as model-based methods or non-model-based methods.

For single hand pose estimation, model-based top-down global approach uses a 3D hand model to fit the testing data [19, 15, 10]. These methods deal with self-occlusion, kinematic constraints and viewpoint changes by model fitting. Due to these properties, they are often used when a hand manipulates an object [19, 10] or interacts with multiple objects [15]. However, these methods, including Joint Tracker [19] and Set of Independent Trackers [20], require a very precise initialization, in both hand location and hand measurement property. Once this tracking procedure goes wrong, it is very hard to recover.

As a non-model-based method, Keskin *et al*. [13] improve RDF to multi-layer in order to address the pose variability issue. They assign hand pose to corresponding shape classes, and use the pose estimator trained specifically for that class. Tang *et al*. [27] use a Latent Regression Forest (LRF) to locate all skeletal joints. In contrast to most previous forest-based algorithms, their solution uses a unique LRF framework guided by an unsupervised latent tree model (LTM). The algorithm is very efficient, and, instead of assigning labels to pixels, it performs a dichotomous divide-and-conquer searching process. Under the guidance of LTM, the coarse-to-fine search procedure finally converges to the exact positions of all hand skeletal

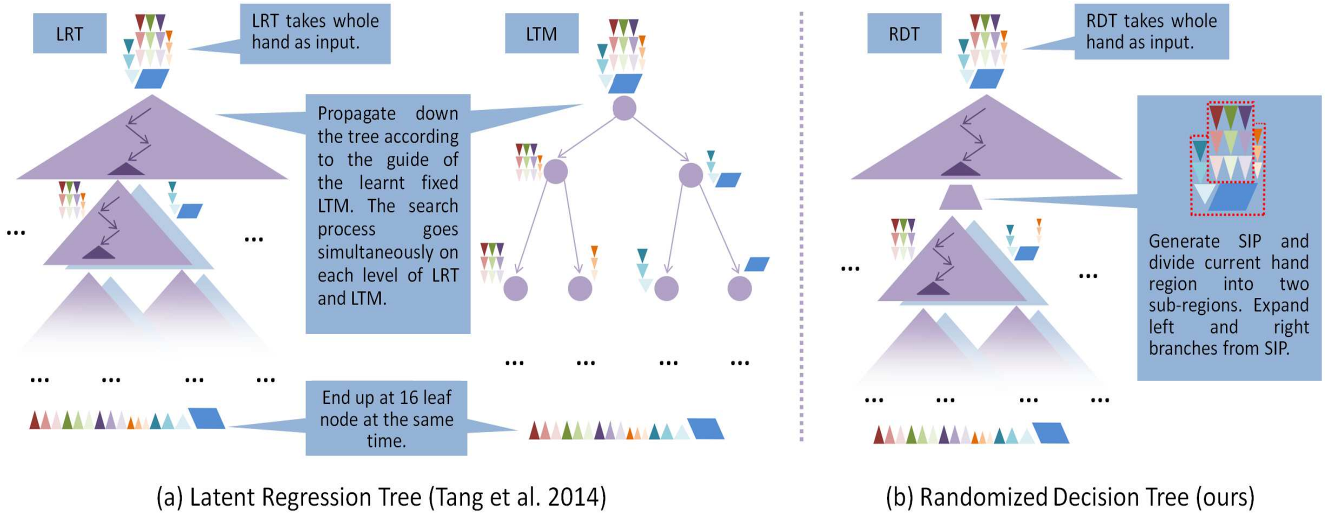


Figure 2. A comparison between LRT [27] (a) and our RDT (b). (a): In LRT, the search process is guided by the pre-learnt hand topology model LTM. During training, while LRT is growing on each level, LTM propagates down the tree simultaneously and provides reference pixels' locations to the growing process. (b): Our RDT uses the result of skeletal point clustering at each level to adaptively generate SIPs, which further guide the SIPs for next level.

joints.

Our work is inspired by [27] and [8], and can be viewed as a coarse-to-fine skeletal joints searching algorithm guided by the proposed segmentation index points (SIPs). Lower level SIPs always maintain offset vectors to higher level SIPs in the decision forest, and these SIPs converge to true hand skeletal joints position in leaves (see Fig. 1).

3. Framework and Methodology

We follow previous studies such as [27, 8] to use the random forest framework for hand skeleton joints estimation. In particular, our framework is similar to that in [27], though we do not use LTM for joints division. Instead, we propose using *segmentation index points* (SIPs) (will be described later) that are more flexible (see Fig. 2).

Given a 3D point cloud sampled on a hand surface, the hand pose estimation problem is to locate all hand skeletal joints. This is formulated as a dichotomous coarse-to-fine divide-and-conquer search problem. The main framework is designed as a binary *randomized decision forest* (RDF), which is composed of a set of *randomized decision trees* (RDT). For each RDT, the input hand 3D point cloud is recursively divided into two cohesive sub-regions until each sub-region contains only one skeletal joint. Then a weighted voting over all RDTs is used to report the locations of all hand skeletal joints.

In order to carry out a better division of hand topology so as to increase detection accuracy, we introduce SIPs as an instructor of these sub-regions. SIPs are associated with the segmented hand components.

In Section 3.1 we discuss the training procedure of an RDT in forest along with the formulation of SIPs and *randomized binary feature* (RBF). Then in Section 3.2 we discuss the testing algorithms.

3.1. Randomized Decision Tree (RDT)

The purpose of growing a *randomized decision tree* (RDT) is to build up a tree structure, which can be used to search an input depth image and retrieve all the skeletal joints of the hand. Searching is performed in a dichotomous divide-and-conquer fashion where each division is guided by a corresponding SIP.

In an RDT, we put a special cache between trees to record SIPs and its corresponding information (see Fig. 3). Despite of this special cache, RDT has three types of nodes: *split-node*, *division-node* and *leaf-node*. Split-nodes use RBF on input data and decide to route them to either left or right. Division-nodes divide the current search hand subregion into two disjoint smaller sub-regions and propagate input data down both paths in parallel. Then when we reach a leaf-node in RDT, the search ends and locations of single skeletal joints are reported.

More specifically, suppose we are training an RDT T at a certain level, the input image set is $\boldsymbol{I} = \{I_1, I_2, \ldots, I_n\}$, and the level deals with a hand sub-region containing skeletal joints $\boldsymbol{C} = \{C_1, C_2, \ldots, C_m\}$. During training, each split-node splits $\boldsymbol{I}$ into two subsets, this process is repeated until information gain falls below a threshold, and at this point a division-node in T takes subset $\boldsymbol{I}_c \subset \boldsymbol{I}$ to divide current hand sub-region. In other words, this division-node di-

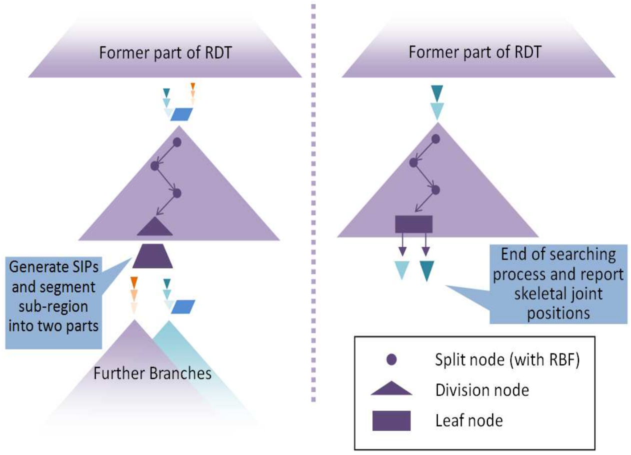


Figure 3. Illustration of three types of nodes in RDF, see Section 3.1 for details.

vides $\boldsymbol{C}$ into $\boldsymbol{C}_l$ and $\boldsymbol{C}_r$ and maintain an SIP. Then $\{\boldsymbol{I}_c, \boldsymbol{C}_l\}$ and $\{\boldsymbol{I}_c, \boldsymbol{C}_r\}$ are propagated down in parallel to create a left child branch and a right child branch.

Error accumulation in RDT is unavoidable. Therefore, in order to minimize the risk, we propose a training data allocation strategy during forest growing, which also speeds up the training process exponentially.

In Section 3.1.1 we discuss the details of growing an RDT, then in Section 3.1.2 we discuss the testing procedure. In addition to these, we also discuss the differences between SIPs and LTM in Section 3.2. After that, in Section 3.3, an essential data allocation trick during training is introduced.

3.1.1 Growing RDT with SIPs

Given a training sample set $\boldsymbol{I}$ with its corresponding labels of all skeletal joints (16 skeletal joints in total in our experiments), we introduce a tree growing algorithm (see Alg. 1).

Suppose we are growing RDT T at a node v. Given training sample set $\boldsymbol{I}$ we define a node v in T as

$$v = (\boldsymbol{C}(v), l(v), r(v), \rho_c(v), \psi(v), \boldsymbol{\rho}), \tag{1}$$

where $\boldsymbol{C}(v)$ is the set of skeletal joints handled by v; $l(v)$ and $r(v)$ point to the left and right children of v; $\rho_c(v)$ is the SIP of v, which roughly locates the centroid of the skeletal joints in $\boldsymbol{C}(v)$; $\psi(v)$ is the RBF stored in this node v, if v is a division-node then $\psi(v) = \emptyset$; and $\boldsymbol{\rho} = \{\overrightarrow{\rho_l}, \overrightarrow{\rho_r}\}$ are shift vectors for left and right child SIPs, if v is a split-node then $\boldsymbol{\rho} = \emptyset$.

At the root node v_0 of an RDT, we initialize the first SIP $\rho_c(v_0) = \rho_0$ as the mass center of the input point cloud. Then we set the current hand sub-region and its components index to full hand skeletal joints. After initialization, we repeat the following two growing steps.

First, we grow several levels of split-nodes in T. The purpose of each of these split-nodes is to split the training sample set $\boldsymbol{I}$ into $\boldsymbol{I}_l$ and $\boldsymbol{I}_r$. Then, both $\boldsymbol{I}_l$ and $\boldsymbol{I}_r$ continue propagating down on tree T, and new split-nodes are generated and split $\boldsymbol{I}_l$ and $\boldsymbol{I}_r$ respectively. This splitting process is repeated until information gain is sufficiently low.

Suppose node v with current SIP ρ_c in tree T handles m skeletal joints components, i.e., $\boldsymbol{C}(v) = \{C_1, C_2, \ldots, C_m\}$. Then a randomized binary feature is a tuple with two parts: a pair of perturbing vectors $\boldsymbol{V}_1, \boldsymbol{V}_2$ and a division threshold τ. These two parts work together with current SIP ρ_c.

Let $\boldsymbol{I} = \{I_1, I_2, \ldots, I_n\}$ be the training images at node v, the division of $\boldsymbol{I}$ to subsets $\boldsymbol{I}_l = \{I_j \in \boldsymbol{I} \mid f(\boldsymbol{V_1}, \boldsymbol{V_2}, \rho_c, I_j) < \tau\}$ and $\boldsymbol{I}_r = \boldsymbol{I} \setminus \boldsymbol{I}_l$ is guided by $f(\cdot)$ defined as:

$$f(\cdot) = D_I(\rho_c + \frac{\boldsymbol{V}_1}{D_I(\rho_0)}) - D_I(\rho_c + \frac{\boldsymbol{V}_2}{D_I(\rho_0)}), \tag{2}$$

where $D_I(\cdot)$ is the depth in image I at a specific pixel location, and ρ_c is the SIP with skeletal index set $\boldsymbol{C}$, formulated by $\rho_c = \mathrm{mean}(\mathrm{p}_{ij} \mid i \in \boldsymbol{C}, j \in 1, 2, \ldots, n)$ where p_{ij} is the center position of component C_i in image I_j. ρ_0 is the first SIP, i.e., the centroid of the hand point cloud. As in LRF, $\frac{1}{D_I(\rho_0)}$ is used to avoid depth shift accumulation.

Therefore, a single split-node learnt in the tree can be represented as a tuple $\psi = (\{\boldsymbol{V}_1, \boldsymbol{V}_2\}, \tau, \rho_c)$.

To learn an optimal ψ^*, first a set of randomly generated tuples $\psi_i = (\{\boldsymbol{V}_{i_1}, \boldsymbol{V}_{i_2}\}, \sim, \rho_c)$ are proposed, where $\sim$ stands for the parameter τ that will be determined later. Let I_j be one of the depth image in $\boldsymbol{I}$. For all $\{\boldsymbol{V}_{i_1}, \boldsymbol{V}_{i_2}\}$ and ρ_c, a set of depth differences are calculated using Eq. 2, and they formed a feature value space. Then this space is divided into o parts evenly, this division corresponds to a set $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_o\}$. The tuples set is completed as $\psi_{io} = (\{\boldsymbol{V}_{i_1}, \boldsymbol{V}_{i_2}\}, \tau_o, \rho_c) \in \boldsymbol{\Psi}$, and they are called RBF candidates. For all these candidates, the tuple ψ^* with the highest information gain is selected for node v. The information gain function is defined as

$$\mathrm{IG}_{\psi_i}(\boldsymbol{I}) = \sum_{s}^{\boldsymbol{C}_m^{\{l,r\}}} \mathrm{tr}(\Sigma^{\boldsymbol{I}}_{\boldsymbol{C}_m s}) - \sum_{t}^{\{l,r\}} \frac{|\,\boldsymbol{I}_t\,|}{|\,\boldsymbol{I}\,|} (\sum_{s}^{\boldsymbol{C}_m^{\{l,r\}}} \mathrm{tr}(\Sigma^{\boldsymbol{I_t}}_{\boldsymbol{C}_m s})), \tag{3}$$

where $\Sigma^{\boldsymbol{I}}_{(\cdot)}$ is the sample covariance matrix of the set of vectors $\{\rho_{\{l,r\}} - \rho_c \mid I_j \in \boldsymbol{I}\}$, and $\mathrm{tr}(\cdot)$ is the trace function, and $\rho_{\{l,r\}} = \mathrm{mean}\{\mathrm{p}_{ij} \mid i \in 1, 2, \ldots, m, I_j \in \boldsymbol{I}_{\{l,r\}}(\psi_i)\}$.

Then the $\psi^* \in \boldsymbol{\Psi}$ with highest information gain is recorded. Consequently, $\boldsymbol{I}$ is divided into $\boldsymbol{I}_l(\psi^*)$ and $\boldsymbol{I}_r(\psi^*)$, and are used for further training of RBF split-node in tree T. This process is repeated until information gain falls below a pre-set threshold.

Second, after growing T to the extent that information gain is low, then a Division node is trained. New SIPs are calculated and their location shift vectors are recorded for higher level tree growing use.

For a division-node v, with SIP $\rho_c(v)$, and component set $\boldsymbol{C}(v) = \{C_1, C_2, \ldots, C_m\}$, and training set $\boldsymbol{I_c} = \{I_1, I_2, \ldots, I_{n_c}\}$, let p_{ij} represent the position of skeletal joint C_i in depth image I_j. The position of all skeletal joints in all images are calculated, $\mathbf{P} = \{\mathrm{p}_{ij} \mid i \in 1, 2, \ldots, m, j \in 1, 2, \ldots, n_c\}$.

Then a bipartite clustering algorithm is to divide $\boldsymbol{C}$ to $\boldsymbol{C}^l$ and $\boldsymbol{C}^r$. Because we are using a binary randomized feature along with a dichotomous RDT, the bipartite clustering helps us to maintain a consistency in tree structure. The clustering algorithm takes a distance matrix D as input, whose entries are defined as

$$D_{i_1 i_2} = \frac{\sum_{I_j \in \boldsymbol{I}} \delta(i_1, i_2; I_j)}{\mid \boldsymbol{I} \mid}, \tag{4}$$

where $i_1, i_1 \in 1, 2, \ldots, m$, and $\delta(i_1, i_2; I_j)$ is the geodesic distance between skeletal joints C_{i_1} and C_{i_2} in image I_j, such distance is known to be robust against object articulation [16] and therefore applied well for hand poses.

The distortion for the clustering [3] is defined as:

$$\boldsymbol{\mathcal{J}}_{(D)} = \sum_{p=1}^{m} \sum_{q=q_1,q_2} r_{pq} D_{i_1,i_2}, \text{ w.r.t. } 1 \leq q_1, q_2 \leq m \tag{5}$$

and our goal is to find $r_{pq} \in \{0, 1\}$ and $\{q_1, q_2 \mid 1 \leq q_1, q_2 \leq m\}$ to minimize $\boldsymbol{\mathcal{J}}$. If i_1 is assigned to cluster q_1, then $r_{pq_1} = 1$ and other $r_{pq} = 0$ for $q \neq q_1$. We can perform an iterative procedure to find $\{r_{pq}\}$ and $\{q_1, q_2\}$ correspondingly. In a two-stage optimization algorithm, $\{r_{pq}\}$ is fixed to find the optimal $\{q_1, q_2\}$ and then $\{q_1, q_2\}$ is fixed to find the optimal $\{r_{pq}\}$. This procedure repeats until convergence or the iteration limitation is reached. Then $\{r_{pq}\}$ is used for cluster $\boldsymbol{C}$.

After $\boldsymbol{C}$ is divided into $\boldsymbol{C}^l$ and $\boldsymbol{C}^r$, they are used to calculate two new SIPs as

$$\begin{aligned} \rho_l &= \mathrm{mean}\{\mathrm{p}_{ij} \mid C_i \in \boldsymbol{C}^l, j \in 1, 2, \ldots, n_c\}, \\ \rho_r &= \mathrm{mean}\{\mathrm{p}_{ij} \mid C_i \in \boldsymbol{C}^r, j \in 1, 2, \ldots, n_c\}, \end{aligned} \tag{6}$$

then $\{\boldsymbol{C}^l, \rho_l - \rho_c\}$ and $\{\boldsymbol{C}^r, \rho_r - \rho_c\}$ are recorded in v.

The above training procedure is recursively carried out until a leaf-node v is reached, meaning that $\boldsymbol{C}(v)$ only contains a single hand skeletal joint. The only difference of training a leaf-node, compared with division-node, is that instead of calculating $\{\rho_{\{l,r\}} - \rho_c\}$, we directly record the offset vector of hand skeletal joints location according to their labels.

3.1.2 Testing

Given a trained RDF F and a testing image I_t, we first feed I_t to each tree T in F. After this coarse-to-fine search process, locations of all skeletal joints of testing image I_t are reported.

Algorithm 1 Growing an RDT T

Input: A set of training samples $\boldsymbol{I}$; the number of labeled hand skeletal joints k
Output: a learnt RDT T

Initialize root in T as $v_0 = Initialize(v_0)$
Divide $\boldsymbol{I}$ into $\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_n$ according to principle described in section 3.3
$v_0 = GrowForest\ (v_0, \boldsymbol{I}_1, \boldsymbol{C}(v_0))$

function $v_{\text{output}} = GrowForest(v, \boldsymbol{I}, \boldsymbol{C})$
if $Size(\boldsymbol{C}(v)) < 2$ **then**
  return
else
  Randomly generate a set of feature vectors $\boldsymbol{\Psi}$
  for all $\psi \in \boldsymbol{\Psi}$ **do**
    Use the information gain Eq. 3 to find the optimal $\{\psi^*\}$ and its corresponding $\{\boldsymbol{I}_l^*, \boldsymbol{I}_r^*\}$
  end for
  if $\mathrm{IG}(\{\boldsymbol{I}_l^*, \boldsymbol{I}_r^*\})$ is higher than the threshold **then**
    Store $\{\psi^*\}$ as a split-node in T
    $l(v) = GrowForest(v, \boldsymbol{I}_l^*, \boldsymbol{C})$
    $r(v) = GrowForest(v, \boldsymbol{I}_r^*, \boldsymbol{C})$
  else
    Divide $\boldsymbol{C}$ into $\boldsymbol{C}_l$ and $\boldsymbol{C}_r$ using bipartite clustering with Eq. 5
    Calculate SIPs and its shift vectors $\{\rho_{\{l,r\}} - \rho_p\}$ with $\{\boldsymbol{C}_l, \boldsymbol{C}_r\}$ and Eq. 6
    Store $\{\rho_{\{l,r\}} - \rho_p\}$ and $\{\boldsymbol{C}_l, \boldsymbol{C}_r\}$ in a division-node in T
    $l(v) = GrowForest(v, \boldsymbol{I}, \boldsymbol{C}_l)$
    $r(v) = GrowForest(v, \boldsymbol{I}, \boldsymbol{C}_r)$
  end if
end if
return

At the beginning, we initialize the first SIP as the mass center of the testing image I_t. Then according to the recorded RBF tuple $\psi = (\{\boldsymbol{V}_{i_1}, \boldsymbol{V}_{i_2}\}, \tau)$ at each split-node, we use Eq. 2 to decide whether the testing image route to the left branch or the right branch in T. If $f(\boldsymbol{V}_1, \boldsymbol{V}_2, \rho_c, I_t) < \tau$ then image I_t goes to the left, otherwise it goes to the right.

When I_t is propagated down to a division-node, the SIPs are updated by the record corresponding SIPs location offset vectors $\{\rho_{\{l,r\}} - \rho_c\}$, where ρ_c is the current SIP. Then both SIPs' left child ρ_l and right child ρ_r are propagated down simultaneously.

This process is repeated until it ends up at 16 leaf nodes in T with its corresponding skeletal joint index set $\boldsymbol{C}$. This index set $\boldsymbol{C}$ is maintained throughout the whole forest, and $\boldsymbol{C}$ provides the information that on which part of hand we are retrieving at a curtain stage in T. At leaf-node the set $\boldsymbol{C}$ contains a single hand skeletal joint. After a weighted voting in RDF F, the location of all 16 skeletal joints are reported.

3.2. SIPs versus LTM

The main difference between our work and that in [27] lies in that LTM is used to guide the search process in an LRT, and we use SIPs to replace LTM in the search process.

LTM tells LRT, at each division-node, how to divide hand posture components into two parts. LTM then decides this binary division by current searching stage in LRT. Specifically, when training a division-node v in LRT, suppose we are dealing with m (hand) components $C_1, C_2, \ldots, C_m$, and using n training images $I_1, I_2, \ldots, I_n$. We define p_{ij} as the center position of component C_i in image I_j. Then, the average location for C_i is defined as $\overline{\mathrm{p}_i} = \mathrm{mean}(\{\mathrm{p}_{ij}, j = 1, \ldots, n\})$. After that, $C_1, C_2, \ldots, C_m$ are clustered into two groups according to LTM. For each group, the mean of $\overline{\mathrm{p}_i}$'s is kept as the reference point provided by LTM to guide LRT.

LTM is the learnt prior of all procedures according to hand geometric properties, and it is fixed regardless of the individual hand poses. More specifically, $C_1, C_2, \ldots, C_m$ are always clustered the same way, regardless of what kind of hand gesture is under processing. This may not be the best solution especially when dealing poses with large variation. Practically, because of the limitation of raw 3D data, there are sometimes noise in the label of training images, and the geometric structure of hand varies from case to case (see Fig. 4). In these cases, it is natural to expect better performance if we have a more flexible division strategy.

Motivated this way, we introduce SIPs into our system. The trick lies in the clustering stage. After average locations for C_i are all calculated by $\overline{\mathrm{p}_i} = \mathrm{mean}(\{\mathrm{p}_{ij}, j = 1, \ldots, n\})$, a bipartite clustering is performed on $\overline{\mathrm{p}_i}$ to divide current hand skeletal joint components into two groups. For each group, an SIP is calculated according to Eq. 6. These two groups are then assigned to RDT to expand its left and right branches.

We summarize a few differences when we add SIPs to this complex RDT tree structure.

First, the LRF framework in [27] is an RDF guided by LTM. Since the division of hand joint components are fixed by LTM, so they do not need to maintain a record of the clustering at division-nodes. However, if we want to use SIPs for a more flexible clustering, this must be recorded after each division-node. As a result, the growing process of RDT needs to be modified. Also, the tree structure of RDT needs to be redesigned. Between a division-node and a split-node in the forest, one more special cache needs to be added for recording the clustering results (see Fig. 3).

Second, during the training process, when growing the forest, SIPs are different from case to case, so we are not able to pre-compute all the locations of hand joint components groups. As a consequence, our model needs longer time for training than [27]. However, this new RDF structure does not affect much on the testing phase, as observed in our experiments. Our approach runs at 55.5 fps on a normal CPU without any parallelism.

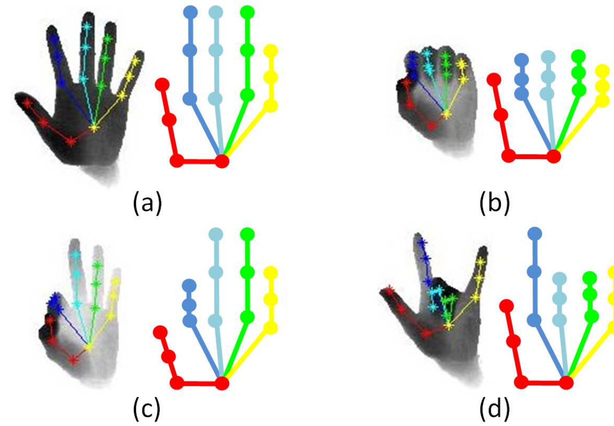


Figure 4. Ideally, all of these four types of hand pose should have same topology structure like (a). However, due to viewpoint and the labeling of raw depth image limitation, (a)(b)(c) and (d) have different hand models. This figure is best viewed in color.

Compared with [27], SIPs have advantages in dealing viewpoint change and errors in 3D annotation (see Fig. 4(b,c,d)). SIPs are less sensitive to both problems, which are non-trivial to address by previous approaches. Since the limitation of 3D raw data and viewpoint, from Fig. 4 we can see, there are usually some distorted-labeled training data. Take Fig. 4(b) as an example, four fingers are bent. However, when labeling the raw 3D data, the location of the bended finger joints are often marked on the surface of the 3D point cloud, which is incorrect. As a result, the geometric structure of the hand seems to have four shorter fingers. Even if we have the correct labeling, viewpoint change still affects the geometric structure of hand to some extents. SIP is a better solution than LTM on this issue, and can better restrain the influence in an acceptable range. This is also proved in our experiments in Section 4.

3.3. Training data allocation during tree growing

Training an RDF is very time consuming. The cost increases dramatically with respect to the number of split-nodes at lower-level layers of a tree. The exponentially increased training time is intolerable and unnecessary.

Taking a picture of the testing process, we can tell that, at an early stage of retrieving procedure, both LTM [27] and our proposed SIPs are only reference node to RBF. Their locations guide RBF to look into the local distinctive features. Though it is essential to constrain them in the correct sub-region of a hand (segmented hand parts), their accuracy with the sub-region is not a critical issue. This is because parental SIPs on early stage will finally be replaced by child SIPs at later stage, then child SIPs will continue to guide the search process. Of course, the more training data we use on each stage, the more accurate the output hand pose estimation will be. However, this is a trade-off between training time and accuracy. We want to restrict our framework training time tolerable, and plug in as many data samples as possible in the meantime on each tree growing process.

To achieve this, we introduce our training data allocation strategy for growing a single RDT T (see Alg. 2). At root of T, instead of using the whole dataset $\boldsymbol{I}$, we first equally divide it to several non-intersect subset $\boldsymbol{I}_i$ where $\boldsymbol{I} = \bigcup_{i=1}^{n} \boldsymbol{I}_i$. In our case, we set n to 10,000. At the first stage, only $\boldsymbol{I}_1$ is used for training T. At level 2 stage, $\bigcup_{i=1}^{2} \boldsymbol{I}_i$ is used for training. At level k stage, $\bigcup_{i=1}^{k} \boldsymbol{I}_i$ is used for training. For a leaf-node, we want the most accurate estimation of hand pose, so we feed the whole set $\boldsymbol{I}$ to train the last division-node before reaching leaf-node.

Algorithm 2 Allocating Training Set

Input: A set of divided training samples $\boldsymbol{I} = \bigcup_{i=1}^{n} \boldsymbol{I}_i$
Output: $\boldsymbol{I}_l$,$\boldsymbol{I}_r$

function $UpdateSet(\boldsymbol{I}, \boldsymbol{C}_l, \boldsymbol{C}_r, level)$
if $Size(\boldsymbol{C}_l) > 2$ **then**
  $\boldsymbol{I}_l \leftarrow \boldsymbol{I} \bigcup \boldsymbol{I}_{level+1}$
else
  $\boldsymbol{I}_l \leftarrow \boldsymbol{I} \bigcup (\bigcup_{level+1}^{n} \boldsymbol{I}_i)$
end if
if $Size(\boldsymbol{C}_r) > 2$ **then**
  $\boldsymbol{I}_r \leftarrow \boldsymbol{I} \bigcup \boldsymbol{I}_{level+1}$
else
  $\boldsymbol{I}_r \leftarrow \boldsymbol{I} \bigcup (\bigcup_{level+1}^{n} \boldsymbol{I}_i)$
end if
return

4. Experiments

In this paper, we use the dataset provided by Tang *et al.* [27] to test our algorithm. This dataset is captured using Intel®'s Creative Interactive Gesture Camera [18] as the depth sensor.

The training set, according to [27], was collected by asking 10 subjects to each perform 26 different hand postures. Each sequence was sampled at 3fps to produce a total of 20K images. The ground truth was manually labeled. An in-plane rotation was used to produce hand-pose training images with different rotation angles, then the training dataset contains 180K ground truth annotated images in total.

For testing, we follow rigorously [27] and use the same two testing sequences A and B, which do not overlap with training data. Sequences are produced by another subject, each containing 1000 frames capturing many different hand poses with severe scale and viewpoint change. Both sequences start with a clear frontal view of an open hand. This is to give a better initialization to other state-of-the-art hand pose tracking algorithm.

In Section 4.1 we evaluate the proposed algorithm against other state-of-the-art methods; then in Section 4.2 we show some of the results form our hand pose estimator.

4.1. Comparison with other methods

Our framework is a further improvement of the prior work from Tang *et al.* [27] and we use the same experiment setting in [27] for comparison. The whole training dataset is used to train an RDF F. Our dataset allocation strategy restricts the number of training images during forest growing process. For trees in F, we do not limit the depth of their structure. In other words, for a tree in our RDF, there may exists many predecessor RBF split-nodes before a division-node in the tree.

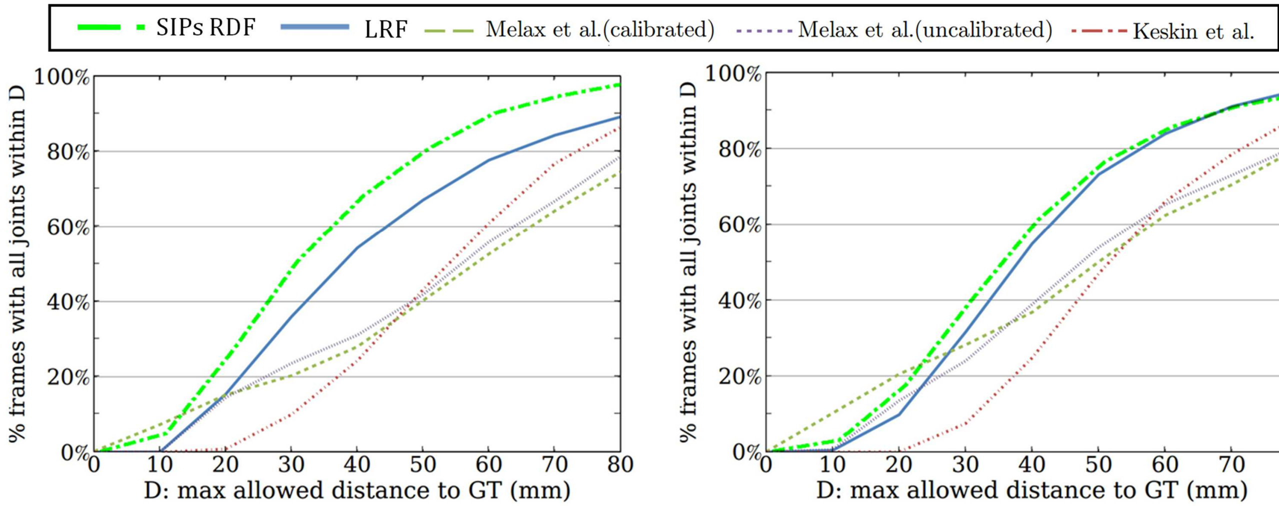
Under these settings, finally, the results are evaluated by a recently proposed challenging metric from Taylor *et al.* [29]. During testing, this metric evaluates the proportion of the testing images that have all estimated hand skeletal joint locations within a certain maximum distance from the ground truth.

We compare our methods with three other state-of-the-art methods proposed in [13, 18, 27]. The benchmark of these three state-of-art algorithms on the testing dataset is reported in [27]. We added our experiment results along with theirs for comparison.

As we can see from Fig. 5 (a) and (b), our framework outperforms the latest state-of-the-art algorithms. At the same time, our results outperform LRF most of the time.

As noted in [27], consider the two testing sequences, B is more challenge than A, since B has much more severe scale and viewpoint changes. As a consequence, our algorithm performs better on A than on B. That said, on both A and B, our algorithm outperforms previous solutions. An interesting observation noticed is that, the relative improvement over LRF is smaller on B than on A. In particular, our algorithm outperforms LRF by a large margin on sequence A, above 8% most of time (Fig. 5); by contrast, on sequence B, the margin is around 2.5% in average. A possible explanation is that, the LRF algorithm has certain robustness against scale or viewpoint change. By checking the curves in Fig. 5, it is interesting to note that the performance of LRF on A is similar to (or even worse than) that on B, despite the fact that B is more challenging than A.

Furthermore, our framework executes in real-time at 55.5 fps, compared with LRF 62.5 fps, the sacrifice of the



(a) Metric (Taylor et al. 2012) of test result on seq. A

(b) Metric (Taylor et al. 2012) of test result on seq. B

Figure 5. Experiments results (modified from Fig. 8(c) and (f) in [27]) with our algorithm (SIPs RDF) shown in a dash-dot green line.

testing running speed is acceptable, which can still provide promising real-time running speed.

4.2. Some sample results

In Fig. 6 we have showed some sample results from our learnt RDF. On left in (a) we show some success estimation, on right in (b) we show some failure estimation. The failure estimation may not maintain a normal hand topology structure since no learnt fixed model guides the limitation. However, our SIPs based framework is more accuracy on general.

5. Conclusion and future work

In this paper, we further improve the work in [27] and propose a new SIPs based randomized decision forest (RDF) in order to do a real-time 3D hand pose estimation using a single depth image. The hand pose estimate framework is not guided by a certain learnt fixed hand topology model, thus, a more flexible clustering of hand skeletal joints can be achieved at each level in a tree of RDF. Under the circumstances that the input training raw data image labels are restricted by the viewpoint, this strategy better handles articulations. With sacrifice of very minor running time speed, still at a real-time level, our implementation outperforms state-of-the-art algorithms.

As for future work, we will investigate the accuracy of this framework on other highly articulated objects, or other datasets. At the meantime, we are also interested in parallelize the algorithm to speed up the training process.

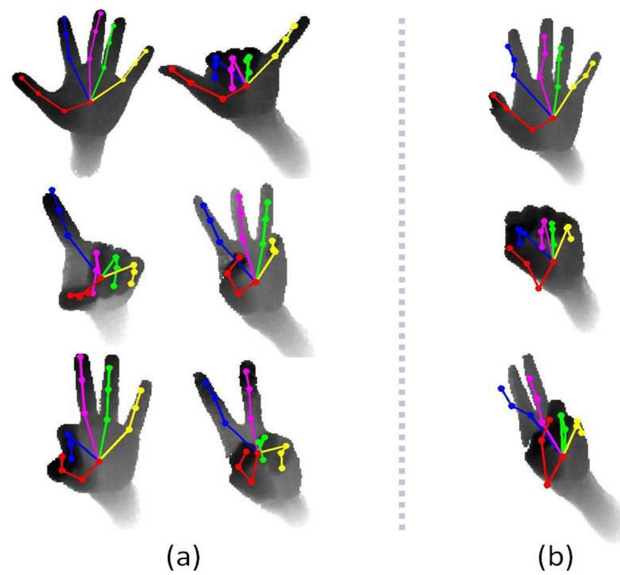


Figure 6. Examples of our results. (a) Success cases. (b) Failure cases - since our algorithm is not guided by a fixed hand topology model, some skeletal joints beyond a hand topology structure, such as thumb roots in (b), may cause problems. This illustration is best viewed in color.

References

[1] Kinect for xbox 360. Microsoft Corp. Redmond WA.

[2] L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *ECCV*, pages 640–653. Springer, 2012.

[3] C. Bishop. Pattern recognition and machine learning. 2006.

[4] L. Breiman. Random forests. *Machine Learning*, 45:5–32,

2001.

[5] B. Carlos and K. Ioannis, A. Estimating anthropometry and pose from a single image. *CVPR*, 1:669–676, 2000.

[6] B. Carlos and K. Ioannis, A. Estimating anthropometry and pose from a single uncalibrated image. *Computer Vision and Image Understanding*, 81(3):269–284, 2001.

[7] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1):52–73, 2007.

[8] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 415–422. IEEE, 2011.

[9] H. Guan, F. Rogerio, Schmidt, and T. Matthew. The isometric self-organizing map for 3d hand pose estimation. *Automatic Face and Gesture Recognition*, pages 263–268, 2006.

[10] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *Computer Vision (ICCV), 2009 IEEE 12th International Conference On*, pages 1475–1482. IEEE, 2009.

[11] F. Hardy, R.-d.-S. Javier, and V. Rodrigo. Real-time hand gesture detection and recognition using boosted classifiers and active learning. *Advances in Image and Video Technology*, pages 533–547, 2007.

[12] K. Ioannis, A, M. Dimitri, and B. Ruzena. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. *CVPR*, pages 980–984, 1994.

[13] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *ECCV*, pages 852–863. Springer, 2012.

[14] S. Knoop, S. Vacek, and R. Dillmann. Sensor fusion for 3d human body tracking with an articulated 3d body model. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1686–1691. IEEE, 2006.

[15] N. Kyriazis and A. Argyros. Scalable 3d tracking of multiple interacting objects. In *CVPR*, pages 3430–3437. IEEE, 2014.

[16] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, 29(2):286–299, 2007.

[17] A. López-Mendez, M. Alcoverro, M. Pardas, and J. R. Casas. Real-time upper body tracking with online initialization using a range sensor. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 391–398. IEEE, 2011.

[18] S. Melax, L. Keselman, and S. Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of the 2013 Graphics Interface Conference*, pages 63–70. Canadian Information Processing Society, 2013.

[19] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2088–2095. IEEE, 2011.

[20] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *CVPR*, pages 1862–1869. IEEE, 2012.

[21] I. Oikonomidis, M. I. Lourakis, and A. A. Argyros. Evolutionary quasi-random search for hand articulations tracking. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3422–3429. IEEE, 2014.

[22] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. *Computer Vision and Pattern Recognition (CVPR)*, 2014.

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

[24] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[25] M. Siddiqui and G. Medioni. Human pose estimation from a single view point, real-time range sensor. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 1–8. IEEE, 2010.

[26] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2456–2463. IEEE, 2013.

[27] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. *CVPR*, 2014.

[28] D. Tang, T.-H. Yu, and T.-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3224–3231. IEEE, 2013.

[29] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 103–110. IEEE, 2012.

[30] H. Trinh, Q. Fan, P. Gabbur, and S. Pankanti. Hand tracking by binary quadratic programming and its application to retail activity recognition. In *CVPR*, pages 1902–1909. IEEE, 2012.

[31] C. Xu and L. Cheng. Efficient hand pose estimation from a single depth image. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3456–3462. IEEE, 2013.