Hierarchically Constrained 3D Hand Pose Estimation using Regression Forests from Single Frame Depth Data

Furkan Kirac*, Yunus Emre Kara, Lale Akarun

Department of Computer Engineering, Bogazici University, TR-34342 Bebek, Istanbul, Turkey

Abstract

The emergence of inexpensive 2.5D depth cameras has enabled the extraction of the articulated human body pose. However, human hand skeleton extraction still stays as a challenging problem since the hand contains as many joints as the human body model. The small size of the hand also makes the problem more challenging due to resolution limits of the depth cameras. Moreover, hand poses suffer from self-occlusion which is considerably less likely in a body pose. This paper describes a scheme for extracting the hand skeleton using random regression forests in real-time that is robust to self-occlusion and low resolution of the depth camera. In addition to that, the proposed algorithm can estimate the joint positions even if all of the pixels related to a joint are out of the camera frame. The performance of the new method is compared to the random classification forests based method in the literature. Moreover, the performance of the joint estimation is further improved using a novel hierarchical mode selection algorithm that makes use

*Corresponding author.

Email addresses: kiracmus@boun.edu.tr (Furkan Kirac), yunus.kara@boun.edu.tr (Yunus Emre Kara), akarun@boun.edu.tr (Lale Akarun)

of constraints imposed by the skeleton geometry. The performance of the proposed algorithm is also tested on a complex dataset where self-occlusion is frequently encountered. The new algorithm which runs in real time using a single depth image is shown to outperform previous methods.

Keywords:
hand gesture, articulated hand pose, depth image, kinect, decision tree

1. Introduction

Interaction of humans and computers has always been an important area of research. Enhancing the communication between humans and computers increases work efficiency and improves the quality of the work being produced. For that reason, new mice and keyboards are still being designed. In spite of the new designs, the use of classical input devices has become a limiting factor when compared to the capabilities of today's computers. As a result, natural interfaces which make use of speech, touch, and gestures are sought. As a natural interface, speech recognition is widely used and has proved its success in recognizing a multitude of languages. After the release of multi-touch enabled devices, touch interfaces have become also mainstream. However, the real breakthrough will be through the use of the human hand as an input device.

Gesture based communication is very intuitive for humans. A simple way to employ gesture recognition is tracking the position of hands and detecting a gesture. Detecting and tracking the centers of naked hands are relatively simpler when compared to detecting the exact configuration of the articulated pose. Unfortunately, most of the hand gestures require the exact detection of the hand pose and tracking the pose variations in a robust fashion. Even in the presence of powerful CPUs, articulated hand pose extraction is a very difficult problem. Moreover, the hand is a small limb that produces self-occluded poses during gesture performance.

With the introduction of inexpensive depth cameras, the human computer interaction field has been revolutionized; and it is now feasible to establish methods employing computer vision based interaction. Bypassing the illumination based problems encountered on the images captured by conventional cameras, the new depth cameras made it possible to establish very fast and less power hungry methods. Unfortunately, inexpensive depth cameras that are widely used still work on low resolution.

As discussed before, most of the gestures produce depth images where a large proportion of the hand parts are unseen. The challenge is to extract the articulated hand pose from self-occluded, noisy, and low resolution hand depth images, with algorithms that are suitable for real time implementation on current CPUs/GPUs.

There are several surveys on hand pose estimation and gesture recognition (Erol et al., 2007; Suarez and Murphy, 2012). Erol et al. (2007) review hand pose estimation methods. They investigate both partial and full pose estimation methods. They categorize the full pose estimation methods into single frame and model-based tracking methods. Most of the works in the literature focus on grayscale or color based methods. These works use either single or multiple cameras. Athitsos and Sclaroff (2003) create a large synthetic hand pose database using an articulated model and estimate 3D hand pose from a single frame cluttered image by finding the closest match.

de Campos and Murray (2006) recover hand pose from a single frame using an RVM-based learning method. In order to overcome the self-occlusion problem, multiple views are combined. Oikonomidis et al. (2011) use Particle Swarm Optimization for solving the 3D hand pose recovery problem. de La Gorce et al. (2011) work on monocular videos for 3D hand pose estimation. They track hand poses using a generative model-based method. Thippur et al. (2013) use visual shape descriptors for describing the hand shape.

Recent advances have been made on the depth camera front. Usually, time-of-flight depth cameras are used to acquire depth (range) images (Malassiotis and Strintzis, 2008; Doliotis and Athitsos, 2012; Gallo and Placitelli, 2012; Billiet et al., 2013). Some works make use of disparities to get depth information, and some works use color and depth data together for estimating hand poses (Yao and Fu, 2012; Liang et al., 2012). Liu and Fujimura (2004) use time-of-flight cameras to acquire depth images for hand gesture recognition. Their hand detection method is based on measuring the shape similarity by thresholding the depth data and using Chamfer distance. They recognize gestures using shape, location, trajectory, orientation and speed features of the hand. Mo and Neumann (2006) work on low-resolution depth images acquired from a laser-based camera. Their algorithm requires manual initialization and uses basic sets of finger poses for interpolating a hand pose. Malassiotis and Strintzis (2008) use depth images generated from synthetic 3D hand models. Suryanarayan et al. (2010) use depth for dynamically recognizing scale and rotation invariant poses. Using a volumetric shape descriptor formed by augmenting a 2D image, they classify six signature hand poses. Lee et al. (2012) model hand shape variations using manifolds. Shotton et al. (2011, 2013) use classification forests for human pose estimation using depth data. Keskin et al. (2011) adapted this method to hand pose estimation.

Recent advances have shown that using random classification or regression forests on depth images is a suitable choice for hand pose skeleton extraction, since the recognition phase is very fast and requires minimal algorithmic complexity.

In this work, we investigate the use of random decision forests for the hand pose estimation problem. We adapt the regression forests method to the hand pose estimation task. This method was initially proposed for the human pose estimation area by Girshick et al. (2011). We compare the results of the newly proposed method to the results of Keskin et al. (2011). In addition, we introduce a hierarchical mode selection method that makes use of constraints imposed by the hand skeleton geometry. Regression forests of Girshick et al. (2011) extract multi-modal joint position distributions per joint. They only consider the global mode. However, disregarded local modes of the joint distribution provide invaluable information in the case of self-occlusions and missing data. Knowing all joint configurations are sampled from the same skeletal constraints, provides a strong prior knowledge about the hierarchy of the modes over different joints. We investigate possible skeleton configurations that fit not only on the global modes but also on the local modes. The probable configurations are then filtered out using distance constraints based on a priori positions from the hand skeleton model. At the end, the best skeletal configuration is selected to be the hand pose using dynamic programming.

In Section 2, we give the details of random decision forests for classification and regression along with the features we used within these methods. Moreover, we introduce our novel hierarchical mode selection method which provides a significant improvement to the joint estimation step. Section 3 includes the details of our synthetic data generation method and the datasets along with our cross validated parameter selection and the results on the test sets. We conclude with the discussion of the results in Section 4.

2. Methodology

This section gives the details of the hand pose estimation methods. The outline of the methods discussed are shown in Figure 1. Section 2.1 includes description of the random decision forests approach. Sections 2.2 and 2.3 explain the use of random decision forests for hand pose estimation based on classification (RDF-C) and regression (RDF-R), respectively. In Section 2.4, we introduce a new hierarchical mode selection algorithm which we call RDF-R+.

2.1. Randomized Decision Trees

A decision tree is used for inferring a set of posterior probabilities for the input. It consists of internal nodes and leaf nodes; where the internal nodes propagate the data to one of its children. In the binary case, decisions to split the data are simply yes/no decisions. Leaf nodes do not make a decision but give statistical information about the nature of the data. The type of statistical information depends on the application.

In randomized decision trees, the decisions on internal nodes are made by selecting a random subset of the features. The aim is to reach leaf nodes

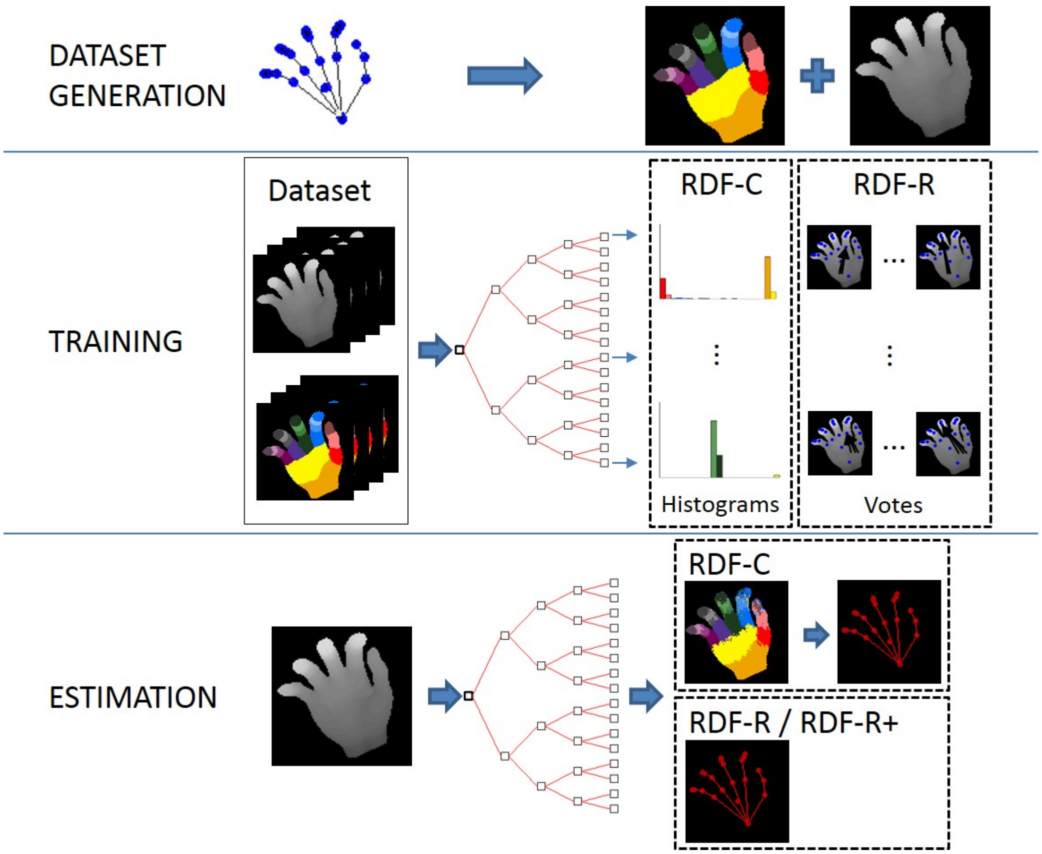


Figure 1: Overview of the methods. The only difference of RDF-C and RDF-R training phases is at the leaves of the trees. RDF-C stores histograms whereas RDF-R stores relative votes. RDF-C classifies each pixel and estimates the joint positions using the estimated pixel labels. However, RDF-R estimates joint positions directly by using the relative votes stored at the leaves.

that are as pure as possible. A pure node consists of samples of only one class. Thus, the features are selected to yield maximum information gain, in other words, minimum entropy. The decision rule is usually of the form:

$$f_n(v) < \tau_n \tag{1}$$

where $f_n(v)$ is a split function, v is the feature vector and τ_n is a threshold, at split node n. A split function is a real valued function on a subset of features.

During training, the split functions and thresholds of nodes are chosen to satisfy the minimum entropy condition. On the leaf nodes, statistics are collected using the data associated with that node. In the case of classification, this is usually a histogram of the class labels of the leaf node data.

Decision Forests are ensembles of decision trees. Each tree can be trained on the same or slightly different datasets. During testing, the given sample is processed in each tree separately. The statistics on the reached leaves are combined for a common response. In classification problems, this is usually done by accumulating normalized histograms on the leaves.

2.2. Hand Pose Estimation using Randomized Decision Forest for Classification (RDF-C)

Shotton et al. (2011) used RDF-C for human body pose estimation. Keskin et al. (2011) adapted that method to the hand pose estimation problem. The aim of the method is to find the pixels closest to each joint. Centroids of those pixels would give the joint position. However, since some classification errors are anticipated, it would be better to find the mode of the pixel positions instead of the mean. For this purpose, first the training data pixels are labeled to define the area around the joints. A decision forest is trained using this data. On the leaves, histograms are calculated using the classes of the pixels.

2.2.1. Pixel Classification using RDF-C

At each node of a randomized decision tree, a random subset of features must be selected and a decision must be made using this subset. Keskin et al. (2011) used the same feature family for hand pose estimation as Shotton et al. (2011). The training data consists of large number of pixels of different depth images. Given a depth image I, features are computed as

$$F_{\boldsymbol{u},\boldsymbol{v}}(I,\boldsymbol{x}) = I\left(\boldsymbol{x} + \frac{\boldsymbol{u}}{I(\boldsymbol{x})}\right) - I\left(\boldsymbol{x} + \frac{\boldsymbol{v}}{I(\boldsymbol{x})}\right) \tag{2}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are offsets relative to the pixel position $\boldsymbol{x}$, and they are normalized by the depth at $\boldsymbol{x}$, $I(\boldsymbol{x})$.

The node data consisting of $(I,\boldsymbol{x})$ pairs are split into two sets for each child as

$$C_L(\boldsymbol{u},\boldsymbol{v},\tau) = \{(I,\boldsymbol{x})|F_{\boldsymbol{u},\boldsymbol{v}}(I,\boldsymbol{x}) < \tau\} \tag{3}$$

$$C_R(\boldsymbol{u},\boldsymbol{v},\tau) = \{(I,\boldsymbol{x})|F_{\boldsymbol{u},\boldsymbol{v}}(I,\boldsymbol{x}) >= \tau\}. \tag{4}$$

Since it is desired to split the data into purer children nodes, the tuple $((\boldsymbol{u},\boldsymbol{v},\tau))$ that gives the maximum information gain is chosen among randomly created tuples. Maximum information gain is found using entropy. First, a candidate split is found and the total decrease in entropy that results from this split is calculated. The split score is

$$S(\boldsymbol{u}, \boldsymbol{v}, \tau) = H(C) - \sum_{s \in \{L,R\}} \frac{|C_s(\boldsymbol{u}, \boldsymbol{v}, \tau)|}{|C|} H(C_s(\boldsymbol{u}, \boldsymbol{v}, \tau)) \tag{5}$$

where $H(K)$ is the Shannon entropy estimated using the normalized histogram of the labels in the sample set K. Then, the candidate tuple that yields the maximum score is chosen for the particular node.

In classification, a pixel $(I, \boldsymbol{x})$ is pushed down the tree until a leaf node is reached. At each leaf node, a histogram represents the posterior probabilities $P(c_i|I, \boldsymbol{x})$ for each class c_i learned during the training phase. The final decision is made by averaging the posterior probabilities estimated by the trees of the forest:

$$P(c_i|I, \boldsymbol{x}) = \frac{1}{N} \sum_{n=1}^{N} P_n(c_i|I, \boldsymbol{x}) \tag{6}$$

where N is the number of trees in the forest.

2.2.2. Joint Position Estimation using the Classified Pixels

For a given depth image, the RDF-C algorithm yields posterior probabilities of each pixel for each class after classification. The resulting probability surfaces are generally multi-modal. Thus, simple averaging is not a suitable operation. For overcoming the high impact of misclassified pixels on the centroid of the pixel locations of the same class, a method that is more robust to false positives than averaging, must be used. In this situation, mode finding is preferred instead of averaging. A local mode finding approach, such as mean shift, can be used.

First, a Gaussian kernel centered on a random point on the probability image is placed. Then, a weighted mean of the probability image under this

Gaussian kernel is calculated. Weight indicates the importance of the pixel and is an estimate of the area the pixel covers. Weights are calculated as

$$w_{I,\boldsymbol{x},c_i} = P(c_i|I,\boldsymbol{x})I(\boldsymbol{x})^2. \tag{7}$$

The newly calculated mean point is used as the starting point of the next iteration. This is repeated until converging to a local maximum. For finding the global maximum, the algorithm runs several times, each time starting at a different initial point and the highest peak is selected.

2.3. Hand Pose Estimation using Randomized Decision Forest for Regression (RDF-R)

Girshick et al. (2011) proposed a new method for the human body pose estimation problem. This technique directly infers the joint coordinates using random decision trees without an intermediate pixel classification representation, hence making it more robust against occlusion. This algorithm is suitable for application to human hand pose estimation. We adapt Randomized Decision Forest for Regression (RDF-R) for directly inferring hand joint positions from the depth image without the intermediary per pixel classification phase. RDF-R can learn and estimate the joint positions even under self-occlusions. Unlike RDF-Cs, RDF-Rs depend on the mean shift algorithm in the training phase, as well.

2.3.1. Training of the Joint Positions

The structure of the trees, namely the features selected in the tree nodes, is the same as described in Section 2.2.1. Therefore, the structure is learned in such a way that leaves are favored to store pixels belonging to the same part of the hand shape.

RDF-Rs do not store hand part histograms at each leaf node l but a distribution over the relative 3D offsets, called a vote, to each hand joint j. These votes are the positions of joints relative to the pixel in question. Each training pixel q is propagated through the tree branches until it reaches a leaf node l. The pixel then casts a relative vote for each distinct joint j. The relative vote can be evaluated using as:

$$\Delta_{iq\rightarrow j} = \mathbf{z}_{ij} - \mathbf{x}_{iq}, \tag{8}$$

where $\mathbf{z}_{ij}$ is the ground truth joint position, and $\mathbf{x}_{iq}$ is the 3D pixel position for a pixel q belonging to image i. For a leaf node l, a relative vote to joint j evaluated for pixel q belonging to image i is then stored in set R_{lj}. An example multi-modal vote distribution is illustrated in Figure 2.

We prefer to use large training sets since we want to infer joint positions of different hand pose configurations. All the information represented by a vote distribution of a leaf cannot be stored in memory. A consensus of relative votes has to be reached per tree leaf for information compression. Unfortunately the vote distributions are not unimodal. Representing the votes by fitting a Gaussian is therefore not suitable. The different clusters of votes have to be distinguished as a preliminary phase. Mean shift algorithm is a proper candidate for the task. After selecting a suitable kernel, the mean shift algorithm finds the number of different clusters and their means. The percentage of relative votes belonging to a cluster is the weight of that cluster. Unfortunately the training phase requires handling of great numbers of votes per tree leaf. In order to learn the relative votes in a reasonable time, the vote distribution R_{lj} is sub-sampled using reservoir sampling of Vitter (1985). Reservoir sampling is a single-pass $O(N)$ algorithm that facilitates speeding up the long training phase. Sub-sampling, once a reasonable sample size is chosen, does not affect the modes of the vote distributions, thus providing a considerable performance increase during the training phase without compromising quality.

Consequently, we initialize a set $R_{lj} = \emptyset$ for all leaf nodes l and joints j. A depth image pixel q is propagated to its respective tree leaf and casts a vote which is stored in R_{lj}. All the reservoir sampled pixels of a leaf l cumulatively represent vote distributions for all different joints j. For all leaf nodes l and joints j we cluster the reservoir sampled distributions R_{lj} using mean shift and take top K weighted modes as V_{lj}.

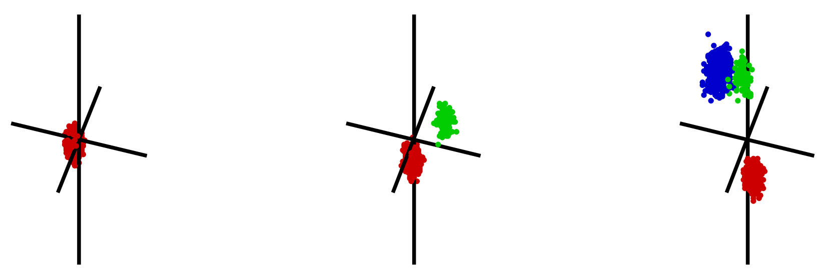


Figure 2: Sample multi-modal vote distributions for three different joints. Vote distributions may have multiple modes. Different colors indicate pixel clusters that are assigned to the same mode found by mean shift.

2.3.2. Direct Joint Position Estimation using RDF-R

We start by initializing $Z_j = \emptyset$ for all joints j. All the pixels in a test depth image are propagated to the tree leaves by starting at the root node and assigning the pixel either to the left or to the right child recursively until a leaf node l is reached. 3D pixel position of the depth image pixel is recalled from the depth image using $\mathbf{x}_q = (x_q, y_q, z_q)^\mathsf{T}$. Each test-time depth pixel casts its per joint vote as represented by the stored weighted relative vote set V_{lj}. Absolute vote coordinate is evaluated using $\mathbf{z} = \Delta_{ljk} + \mathbf{x}_q$. The vote is not cast if $\|\Delta_{ljk}\|_2 \geq \lambda_j$, where λ_j is a distance threshold learned for each different hand joint. To aggregate the absolute votes Z_j, for each joint j we define a continuous distribution over world space using a Gaussian Parzen density estimator such as

$$P_j(z') = \sum_{(z,w)\in Z_j} w.\exp\left(-\left\|\frac{z'-z}{b_j}\right\|_2^2\right), \tag{9}$$

where b_j is a learned per-joint bandwidth. Running mean shift using Equation 9 produces the weighted modes as final hypotheses.

2.4. Hierarchical Mode Selection using Geometry Constraints (RDF-R+)

RDF-R method outputs posterior distributions of possible joint locations. Even though using the global modes of the multi-modal distributions seems to be the most straightforward approach, the correct position of the joint often corresponds to a local mode. However, considering local modes for the joint positions results in multiple skeletal configuration candidates, instead of a single one given by the global modes. For selecting a suitable configuration, we introduce a hierarchical mode selection method and define a constraint function based on our prior knowledge about the hierarchical structure of the 3D model. We penalize each candidate skeletal configuration according to our constraints and select the one with the smallest penalty to be the hand pose.

The hierarchy of a skeleton can be defined as

$$H = \{(c, p) : p \text{ is the parent of } c\} \tag{10}$$

where c and p are joints.

Let j be a joint and $P(\cdot|j)$ be the posterior distribution of j's position. Assuming that the distribution has N_j modes, we define a mode of this distribution as $\boldsymbol{x}_j^{k_j}$ where $k_j \in \{1, \ldots, N_j\}$ and $P(\boldsymbol{x}_j^{k_j}|j) \geq P(\boldsymbol{x}_j^{k_j+1}|j)$. With this condition modes are ordered decreasingly according to their probabilities.

For finding the skeletal configuration, we want to select a mode for each $P(\cdot|j)$. This selection is performed by dynamic programming so that the total penalty of the hierarchical model is minimized for a given penalty function f,

$$(k'_1, \ldots, k'_J) = \operatorname*{argmin}_{k_1,\ldots,k_J} \sum_{(c,p)\in H} f\left(\boldsymbol{x}_c^{k_c}, \boldsymbol{x}_p^{k_p}\right) \tag{11}$$

where J is the total number of joints and $k_1, \ldots, k_J$ are the mode indices of the related joints. Then, the most suitable skeletal configuration can be represented as

$$S' = (\boldsymbol{x}_1^{k'_1}, \ldots, \boldsymbol{x}_J^{k'_J}). \tag{12}$$

By defining a penalty function, different hierarchical constraints can be imposed. Let us define two different penalty functions

$$f_1\left(\boldsymbol{x}_c^{i_c}, \boldsymbol{x}_p^{i_p}\right) = \begin{cases} 0, & i_c = 1 \text{ and } i_p = 1 \\ 1, & \text{otherwise} \end{cases} \tag{13}$$

$$f_2\left(\boldsymbol{x}_c^{i_c}, \boldsymbol{x}_p^{i_p}\right) = \left(||\boldsymbol{x}_c^{i_c} - \boldsymbol{x}_p^{i_p}|| - b_{cp}\right)^2 \tag{14}$$

where b_{cp} is the expected length of the bone between the joints c and p.

Since $\boldsymbol{x}_j^k$s are ordered according to their probability, the global mode of the posterior distribution is $\boldsymbol{x}_j^1$. Thus, f_1 behaves as the global mode finding approach used by Girshick et al. (2011). f_2 tries to select the modes such that the distance between them is as close as possible to the expected bone length.

An example of the improvement made by RDF-R+ method is shown in Figure 3 which shows the vote distribution for tip of the index finger. As clearly seen, there are two different local modes in the distribution. The global mode is not the correct one to be selected. RDF-R wrongly selects the global mode whereas RDF-R+ considers the hierarchical dependencies of the joints and finds the appropriate local mode successfully.

3. Experiments

In Section 3.1, we give the details of the hand data generation step and introduce the datasets that we use. Section 3.2 discusses the fine-tuning of the essential training parameters, namely the forest size, the tree depth, the probe distance, the depth threshold, and the mean shift bandwidth. In Section 3.3, the fine-tuned RDFs are tested on four different datasets. We first test on the training dataset to gather information about the characteristics

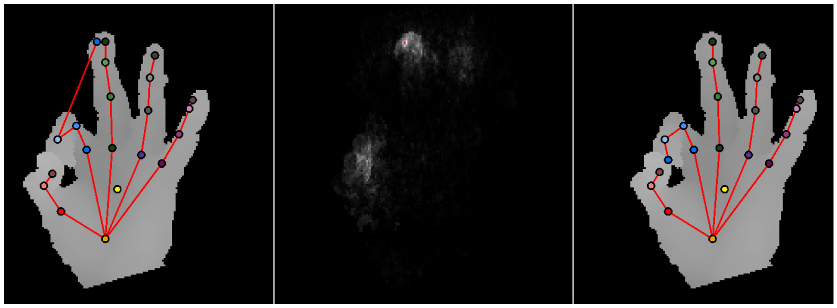


Figure 3: Improvement of RDF-R+ method over RDF-R is shown for tip of the index finger: left) incorrect fingertip estimation of RDF-R, middle) accumulated votes for index fingertip, right) correct fingertip estimation of RDF-R+.

of the methods, and then extend our tests to two different synthetic and one real test datasets.

3.1. Data Generation and Datasets

Training the decision trees for classification and regression requires a great amount of training data. Capturing such a big dataset and labeling different parts of the depth images is a problem on its own. In order to cope with this problem, a synthetic 3D hand mesh is modeled and a realistic skeleton is rigged. The produced skeletal object is animated. In this approach, the difficult pixel labeling problem reduces to creating a label texture that is mapped to the hand mesh. We generate the label texture such that each skeleton joint is at the center of one of the labeled parts. The variation of the human hand across different individuals is significantly less than the body. An average sized hand is selected whose length from the bottom of the palm to the tip of the middle finger is 20 cm. Each different joint of the hand is restricted in a manner to mimic the constraints of the human hand. The synthesizer program designed for this study allows for different poses to be stored in keyframes along a timeline. Once an animation is designed, we animate it using linear interpolation between different keyframes.

In this study, we use 40 different hand poses. Poses are mainly selected from the American Sign Language alphabet. 26 poses represent the letters from A to Z, and 10 poses are for the numbers between 1 and 10. We also include four widely used hand poses, namely the closed-hand, open-hand, approval gesture, and all finger tips touching pose. For each pose, the model is rotated up to 32, 64, and 64 degrees along the x, y, and z axes, respectively. Various samples are collected for different angles with steps of six degrees. Center of the palm is always aligned with the center of the images created. We also add Gaussian depth noise to the depth image pixels with a mean of 10 mm, and a standard deviation of 5 for improving the generalization of the trained classifiers. During hand pose configuration setup, Gaussian noise is introduced to the angles of the skeleton for the unconstrained degree of freedoms with a mean of 2 degrees and a standard deviation of 1 degree. The resulting training (TRAIN) dataset consists of 29766 image samples. Rendered depth and label images are 160x160 pixels in resolution. In Figure 4, we show sample frames from the TRAIN dataset.

Training and validation are done on the TRAIN dataset using 10–fold cross validation. For testing the performance of different methods, we created two different datasets. The first one is the cropped (CROP) dataset. It is generated by retaining the center 80x80 pixels and erasing the outer pixels of each image from the TRAIN dataset. On the average, 81.09% of the pixels from each image are kept with a standard deviation of 8.91. The CROP dataset is used for testing the performance of different methods under missing

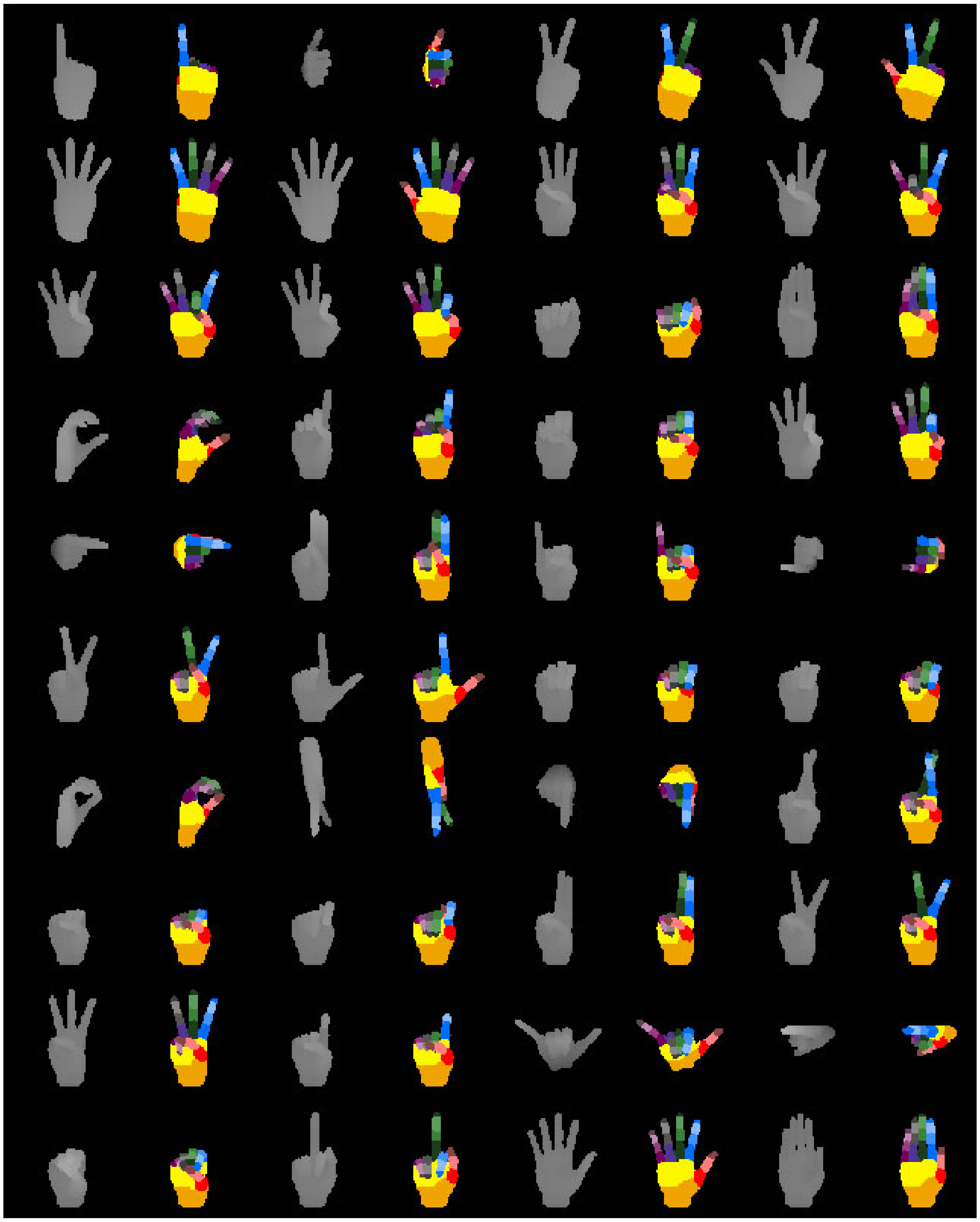


Figure 4: Sample frames from the TRAIN dataset

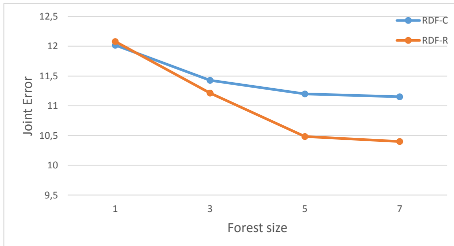
data conditions.

The second test dataset is the Rock–Paper–Scissors–Lizard–Spock (RPSLS) dataset. The Rock–Paper–Scissors–Lizard–Spock game is invented by Kass (1995). The RPSLS dataset is a completely new dataset synthesized from 5 different well-known poses not contained in the training set. All possible transitions between pose pairs are considered and animated using 3 frames per transition. The same rotation conditions are also applied to the pose animations, resulting in a dataset of 30492 images with 160x160 resolution. The poses of the RPSLS dataset are not used during the training of the decision trees. This dataset is used to benchmark the generalization performance of different methods.

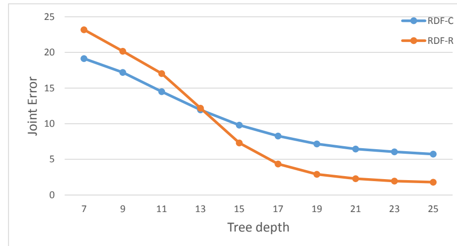
Additionally, we tested the methods on a subset of ASL Finger Spelling Dataset of Pugeault and Bowden (2011) for reporting the performance on real data. The dataset contains sample frames of 5 annotators for 24 finger spelling signs. Since 3D annotation on depth data is a tedious task, we only annotated a 55 frame subset of the dataset. The annotated subset consists of one sample for a, d, e, f, i, l, s, u, v, w, and y signs of each annotator.

3.2. Parameter Selection

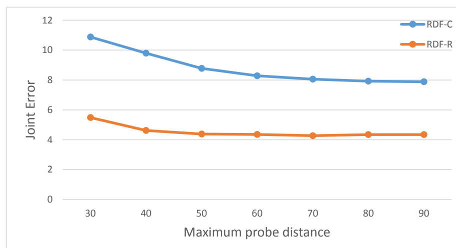
We do 10–fold cross validation for fine-tuning the essential training parameters. We investigate the effects of the forest size, the tree depth, the probe distance, the depth threshold, and the mean shift bandwidth parameters.



(a) The effect of forest size



(b) The effect of tree depth



(c) The effect of probe distance

Figure 5: Effects of training parameters

3.2.1. The Effect of the Forest Size

An advantage of random forests is that the inference performance can be enhanced by combining multiple random trees. Both the generalization capability and the accuracy of the inference improves as the number of trees used is increased. There is a trade-off between inference time and the inference accuracy. Figure 5a shows the effect of number of trees on the accuracy of the system. Accuracy, recognition time, and memory usage are jointly optimized when the number of trees is selected to be 5.

3.2.2. The Effect of the Tree Depth

The depth of the trees is also an essential parameter. The representation capability increases as the depth of the trees increase. Unfortunately, increasing the depth of the trees also increases the inference time. Selecting an optimal depth is important for balancing the inference accuracy and the recognition time. In addition to that, the memory requirement also increases exponentially. If the training data complexity and size are not adequate for the utilization of the desired tree depth, numerous empty sub-branches in the forest structure appear. This under-utilization causes ineffective use of the allocated memory. For instance, selecting a tree depth of 20 consumes approximately 256 MB of memory per tree.

The validation performance of both RDF-C and RDF-R stabilizes after a tree depth of 20 levels. They both slowly converge to their maximum accuracy as we increase the tree depth. However, it is not feasible to further increase the tree depth as the memory required increases exponentially. We selected a tree depth of 21 for our tests as it is a good trade-off between the accuracy, recognition time, and memory requirement.

Another interesting behavior is that RDF-Cs perform better than RDF-Rs when tree depth is less than 13 as shown in Figure 5b. This behavior is due to the fact that RDF-Cs store class label histograms whereas RDF-Rs store 3D relative votes. For a high quality histogram, leaves should have numerous pixels, which is the case in a shallow tree. On the other hand, RDF-Rs work better with just enough data.

3.2.3. The Effect of the Probe Distance

Probe distance is an important parameter which defines the learning amount from spatial relations. As we increase the probe distance, even more distant depth pixel couples are utilized for inference. With a small probe distance value, a more localized recognizer is trained which cannot infer successfully using correlation of distant parts or joints of the model. On the other hand, selecting a bigger than needed maximum probe distance value increases the training time. Both the RDF-C and RDF-R methods converge to their optimal performances when probe distance is selected to be 60 for our dataset as seen in Figure 5c.

3.2.4. The Effect of the Depth Threshold

Depth threshold is similar to the probe distance. It controls the amount of learning based on depth variations. A very small value forces the learning not to depend on depth differences which produces a silhouette learner. A big value may learn the noise along the depth axis. We selected 30 as the depth threshold.

3.2.5. The Effect of the Mean Shift Bandwidth

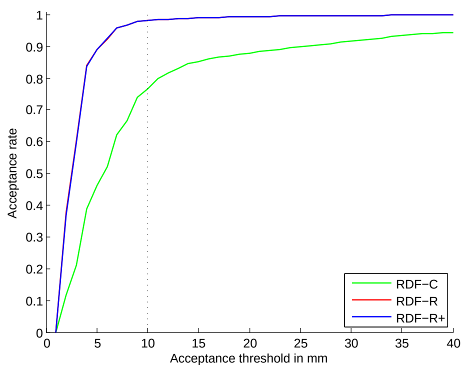
We used a shared bandwidth parameter for all joints. RDF-C algorithm produces less multi-modal and smoother distributions when compared to RDF-R. RDF-Rs form multi-modal distributions where the distribution peaks are clearly distinguished. Selecting an appropriate bandwidth has a greater influence on the performance of RDF-Rs. Performance of RDF-Cs do not change for reasonable values of bandwidth values whereas the performance of RDF-Rs are clearly dependent on the bandwidth parameter. We selected the bandwidth parameter to be 8.

3.3. Hand Pose Estimation Test Results

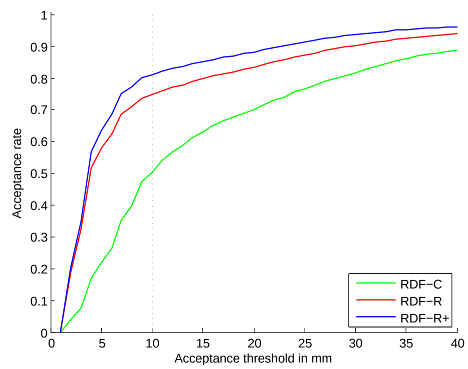
We start by testing all methods with the training dataset for demonstrating the amount of learning each method can achieve. We use CMC curves that report successful joint localization versus an acceptance threshold. For instance, the acceptance rate at 10 mm shows the percentage of joints that are closer than 10 mm to ground truth locations. RDF-C achieves a performance rate of 76.6% at 10 mm acceptable distance threshold. For RDF-R and RDF-R+, this performance increases to 98.0% and 98.1% respectively. These results clearly show that RDF-C under-performs due to self-occlusions of the hand poses. Both regression forest based methods are robust to self-occlusion, hence, can learn all of the joints with a high accuracy. Figure 6a shows the CMC curve for the TRAIN dataset.

3.3.1. CROP Dataset

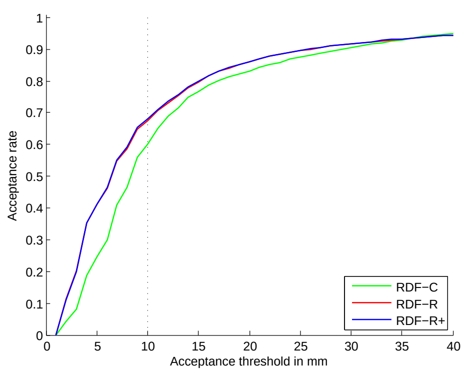
The hand is a limb that inherently produces self-occlusions. Moreover, depth data may be partly missing due to various other reasons. Parts of the hand may be out of sight of the camera. Similarly hand may be very close to the camera. Depth cameras such as Kinect have zero planes. The pixels closer than the zero plane are clipped. Another common cause of missing data is occlusion imposed by other objects in the environment. Depth cameras provide qualitative information where significant depth differences among neighboring regions occur. Given a depth image, segmentation algorithms are able to mark those regions occluded by other objects with a high accuracy. When those occluded regions are removed, the resulting depth image is an image where some of the valuable data is missing. The CROP dataset is specially designed for testing the inference performance of methods in the case of large amount of occlusions. The RDF-C method is by design not robust against missing data. It produces a transient state of pixel classifications where the valuable information about occluded parts are lost. RDF-R is implemented to cope with this problem. It is robust to occlusion



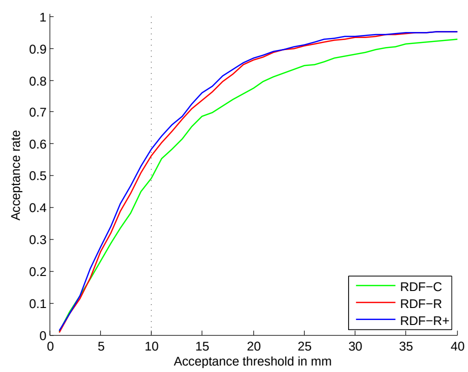
(a) The TRAIN dataset



(b) The CROP dataset



(c) The RPSLS dataset



(d) ASL Finger Spelling Dataset

Figure 6: CMC curves for all datasets

by design. In addition, RDF-Rs provide multi-modal posterior distributions that are suitable for imposing structural constraints. The prior information provided by structural constraints of the hand is applied to create a new algorithm, namely RDF-R+.

Figure 6b shows the CMC curve for the CROP dataset. The performance plots clearly demonstrate the strength of RDF-R over RDF-C. 50.4% of all joints recognized by the RDF-C method are in a neighborhood of 10 mm of ground-truth coordinates. RDF-R method enhances this performance to 74.8%, which is a very significant improvement. Applying skeletal constraints still improves the results. RDF-R+ performs significantly better than RDF-R, increasing the performance to 81.3%.

3.3.2. Rock-Paper-Scissors-Lizard-Spock (RPSLS) Dataset

The RPSLS dataset is a difficult dataset to recognize by the trees trained with the TRAIN dataset. It is chosen to evaluate the sensitivity of the methods against extreme situations. The poses used during the creation of the dataset are either completely new poses or very similar but different poses. In either case the exact poses are not included in the training dataset.

Training dataset includes a thumb-up pose which means alright in English body language. This pose is similar to the rock-pose of RPSLS dataset. The open-hand pose of training dataset is similar to the Spock-pose of RPSLS dataset. The open-hand pose is not regarded as similar to the paper pose due to their different alignments around z-axis. Training set rotates the poses around z-axis only 32 degrees which cannot cover 90 degree difference between open-hand and paper poses. None of the poses in RPSLS dataset is included in training. During this test, we check for the generalization limits of different methods.

Figure 6c shows the CMC curve for the RPSLS dataset. The performance plots demonstrate the similar relative performance improvements between RDF-C, RDF-R. RDF-R+, however, behaves similar to RDF-R. 60.2% of all joints recognized by the RDF-C method are in a neighborhood of 10 mm of ground-truth coordinates. RDF-R method increases the performance to 67.5%. An interesting result is that the performance of RDF-C is better on RPSLS dataset compared to the CROP dataset with their respective values of 60.2% and 50.4%. This is caused due to successful recognition rate on poses that are similar to the training poses and absence of cropping. Inferring in case of missing data is where RDF-C is weak. Applying skeletal constraints this time improves the results not so significantly, as its performance is 67.7%. When we examine the multi-modal posterior distributions of different joints, we see that the joint configurations are either detected with a high confidence or not detected at all with a high variance. Constraints cannot improve the performance significantly since similar poses are already detected well enough.

3.3.3. ASL Finger Spelling Dataset (SURREY)

Figure 6d shows the CMC curve for the ASL Finger Spelling Dataset. The performance of all algorithms are lower than those for the CROP and RPSLS datasets. This performance degradation is due to several factors: The first, is the significantly different characteristics of the training and test datasets: The test dataset contains many different unseen poses, and variations. The second is the presence of 5 different subjects, with different hand geometries. To see which of these factors weighs more in performance degradation, we have looked at performance on different shapes and on different subjects. Instead of giving CMC curves, we provide performance figures at the 10 mm acceptance threshold in Table 1 and Table 2, for different shapes and different subjects, respectively.

Table 1: Percentage of joints that are closer than 10 mm to the ground truth for each sign on ASL Finger Spelling Dataset

ASL Letter	RDF-C	RDF-R	RDF-R+
a	46.3	44.2	**48.4**
d	49.5	**61.1**	**61.1**
e	41.1	**53.7**	52.6
f	48.4	47.4	**53.7**
i	41.1	52.6	**58.9**
l	54.7	65.3	**67.4**
s	35.8	**47.4**	45.3
u	55.8	**67.4**	63.2
v	58.9	58.9	**62.1**
w	51.6	64.2	**65.3**
y	58.9	55.8	**62.1**

As observed from Table 1 and Table 2, the performance varies among different hand shapes and subjects. For example, the ASL letters a and s perform the worst: Upon inspection, it is seen that these ASL letters have been performed differently than they are rendered in the training database. If one excludes these shapes from the test set, performance increases by 2.5%. Different subjects, on the other hand, affect performance; somewhat

Table 2: Percentage of joints that are closer than 10 mm to the ground truth for each subject on ASL Finger Spelling Dataset

Subject	RDF-C	RDF-R	RDF-R+
Subject 1	52.2	56.9	**61.7**
Subject 2	**50.2**	49.3	47.8
Subject 3	46.4	51.2	**56.5**
Subject 4	45.9	**57.9**	56.9
Subject 5	51.7	65.6	**67.9**

less. The hand size of the subject is an important factor. For instance, RDF-C performs better for subject 2. When subject 2's live samples are examined, it is seen that she has a considerably bigger palm and shorter fingers than the synthetic model used in training. In some rare cases RDF-R+ algorithm decreases the performance of RDF-R. It is due to imposing bone length constraints which are not very compatible with the test data that is estimated. It is apparent that the system would further benefit from more rigorous training, with poses closer to those in the test set, with different hand shapes, and with different hand sizes.

Overall live data performance of RDF-C, RDF-R, and RDF-R+ at 10 mm acceptance threshold are 49.3%, 56.2%, and 58.2% respectively. Comparing the different algorithms, we observe that RDF-R algorithms perform significantly better than RDF-C; and RDF-R+ has a 2% advantage over RDF-R.

Figure 7 illustrates sample problematic cases for different methods. The average joint estimation performance rates of methods for all four datasets are shown in Table 3.

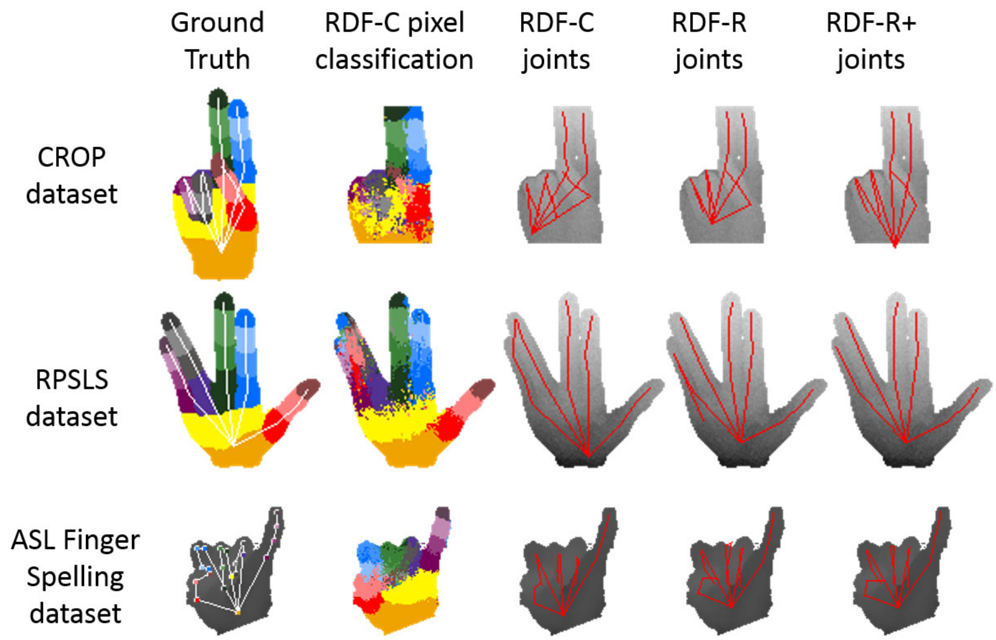


Figure 7: Joint estimation illustrations on test datasets

4. Conclusions

We have demonstrated an implementation of regression forests for estimating the articulated 3D pose of the human hand. Previous attempts at articulated hand pose estimation used RDF-Cs. We have adapted RDF-Rs to this problem and implemented an improved hierarchically constrained version for further enhancing the robustness against heavy occlusion by implementing an algorithm that exploits the prior knowledge about the hierarchy of human hand. Considering the hierarchical dependencies of the joints improved the joint position accuracy significantly. Skeletal constraints are

Table 3: Acceptance rates for the threshold of 10 mm. In all datasets, RDF-R+ method outperforms the other methods.

Dataset	RDF-C	RDF-R	RDF-R+
TRAIN	76.6	98.0	**98.1**
CROP	50.4	74.8	**81.3**
RPSLS	60.2	67.5	**67.7**
SURREY	48.7	55.8	**57.6**

exhaustively evaluated using dynamic programming in real-time. Tests with real data have shown us that although performance is lower with tests on real data, the results are consistent and performance is still acceptable. In order to improve performance, more rigorous training with i) more poses, ii) different hand geometries and, iii) real data is left as future work. The inference algorithms altogether run with an approximate speed of 200 FPS on a conventional notebook computer (Core i7 Quad 2.7 Ghz). Moreover the approach only uses a single depth image for inference. Temporal information can still be utilized for extra performance in future studies. Being able to detect the hand configuration without using a prior calibration step is important for commercial applications. Although this method works with a high accuracy, it can also be used as an initialization and/or observation step for a temporal domain tracker. For future studies, other skeletal constraints can be used and combined. Distances between all different joint pairs can be learned from the dataset for applying more restrictive hand configurations. Posterior distribution of joints can also be used as an observation step of a particle filter that fits a skeleton with a fast local search.

References

Athitsos, V., Sclaroff, S., 2003. Estimating 3D hand pose from a cluttered image. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. , II–432–9doi:10.1109/CVPR.2003.1211500.

Billiet, L., Oramas M., J., Hoffmann, M., Meert, W., Antanas, L., 2013. Rule-based hand posture recognition using qualitative finger configurations acquired with the Kinect, in: Proceedings of the 2nd International Conference on Pattern Recognition Applications and Methods, pp. 1–4.

de Campos, T., Murray, D., 2006. Regression-based Hand Pose Estimation from Multiple Cameras. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06) , 782–789doi:10.1109/CVPR.2006.252.

de La Gorce, M., Fleet, D.J., Paragios, N., 2011. Model-Based 3D Hand Pose Estimation from Monocular Video. IEEE transactions on pattern analysis and machine intelligence , 1–14doi:10.1109/TPAMI.2011.33.

Doliotis, P., Athitsos, V., 2012. Hand Shape and 3D Pose Estimation Using Depth Data from a Single Cluttered Frame, in: International Symposium on Visual Computing, pp. 148–158.

Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., Twombly, X., 2007. Vision-based hand pose estimation: A review. Computer Vision and Image Understanding 108, 52–73. doi:10.1016/j.cviu.2006.10.012.

Gallo, L., Placitelli, A.P., 2012. View-independent Hand Posture Recognition from Single Depth Images Using PCA and Flusser Moments, in: 2012 Eighth International Conference on Signal Image Technology and Internet Based Systems, Ieee. pp. 898–904. doi:10.1109/SITIS.2012.133.

Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A., 2011. Efficient Regression of General-Activity Human Poses from Depth Images, in: 2011 International Conference on Computer Vision, Ieee. pp. 415–422. doi:10.1109/ICCV.2011.6126270.

Kass, S., 1995. Rock paper scissors spock lizard. URL: `http://www.samkass.com/theories/RPSSL.html`.

Keskin, C., Kirac, F., Kara, Y., Akarun, L., 2011. Real-time hand pose estimation using depth sensors, in: Proceedings of Thirteenth IEEE International Conference on Computer Vision Workshops. ICCV 2011, IEEE Comput. Soc. pp. 1228—-1234.

Lee, C., Chun, S., Park, S., 2012. Articulated Hand Configuration and Rotation Estimation using Extended Torus Manifold Embedding, in: 21st International Conference on Pattern Recognition (ICPR 2012), pp. 441–444.

Liang, H., Yuan, J., Thalmann, D., 2012. Hand Pose Estimation by Combining Fingertip Tracking and Articulated ICP, in: Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry - VRCAI '12, ACM Press, New York, New York, USA. pp. 87–90. doi:10.1145/2407516.2407543.

Liu, X., Fujimura, K., 2004. Hand gesture recognition using depth data. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings. , 529–534doi:10.1109/AFGR.2004.1301587.

Malassiotis, S., Strintzis, M., 2008. Real-time hand posture recognition using range data. Image and Vision Computing 26, 1027–1037. doi:10.1016/j.imavis.2007.11.007.

Mo, Z., Neumann, U., 2006. Real-time Hand Pose Recognition Using Low-Resolution Depth Images. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06) , 1499–1505doi:10.1109/CVPR.2006.237.

Oikonomidis, I., Kyriazis, N., Argyros, A., 2011. Markerless and efficient 26-DOF hand pose recovery, in: Proceedings of the 10th Asian conference on Computer vision-Volume Part III, Springer. pp. 744–757.

Pugeault, N., Bowden, R., 2011. Spelling it out: Real-time ASL fingerspelling recognition. 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops) , 1114–1119doi:10.1109/ICCVW.2011.6130290.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A., 2011. Real-time human pose recognition in parts from single depth images, in: CVPR.

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

Suarez, J., Murphy, R.R., 2012. Hand Gesture Recognition with Depth Images: A Review, in: 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, Ieee. pp. 411–417. doi:10.1109/ROMAN.2012.6343787.

Suryanarayan, P., Subramanian, A., Mandalapu, D., 2010. Dynamic Hand Pose Recognition Using Depth Data. 2010 20th International Conference on Pattern Recognition , 3105–3108doi:10.1109/ICPR.2010.760.

Thippur, A., Ek, C., Kjellström, H., 2013. Inferring Hand Pose: A Comparative Study of Visual Shape Features, in: International Conference on Face and Gesture.

Vitter, J.S., 1985. Random sampling with a reservoir. ACM Transactions on Mathematical Software 11, 37–57. doi:10.1145/3147.3165.

Yao, Y., Fu, Y., 2012. Real-Time Hand Pose Estimation from RGB-D Sensor, in: 2012 IEEE International Conference on Multimedia and Expo, Ieee. pp. 705–710. doi:10.1109/ICME.2012.48.