Estimating 2D Multi-hand Poses from Single Depth Images

Le Duan[1(✉)], Minmin Shen[1,3P], Song Cui[2,4P], Zhexiao Guo[1], and Oliver Deussen[1]

[1] INCIDE Center, University of Konstanz, Konstanz, Germany
{duan.le,zhexiao.guo,oliver.deussen}@uni-konstanz.de, mmshenntu@gmail.com
[2] Institute of High Performance Computing, Singapore, Singapore
songcui@acm.org
[3] Amazon Alexa, San Jose, USA
[4] Cisco Systems, San Jose, USA

Abstract. We present a novel framework based on Pictorial Structure (PS) models to estimate 2D multi-hand poses from depth images. Most existing single-hand pose estimation algorithms are either subject to strong assumptions or depend on a weak detector to detect the human hand. We utilize Mask R-CNN to avoid both aforementioned constraints. The proposed framework allows detection of multi-hand instances and localization of hand joints simultaneously. Our experiments show that our method is superior to existing methods.

Keywords: Multi-hand pose estimation · Pictorial Structure
Mask R-CNN

1 Introduction

Accurate hand pose estimation from depth images or videos plays an essential role in human-computer interaction, as well as virtual and augmented reality. However, challenges with estimating hand pose can arise from self-similarity, self-occlusion, and large view-point variation. Although much progress has been made in this area [8,18,23–27], multi-hand pose estimation is still mostly unsolved. A good solution, however, would provide more flexibilities and possibilities in many HCI applications.

Compared to single-hand pose estimation, estimating poses of multiple hands from a single depth image is more difficult because it requires the correct detection of all hand instances while also precisely localizing the corresponding hand joints. A straightforward way to solve this problem is to follow the common

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-11024-6_17) contains supplementary material, which is available to authorized users.

two-stage strategy [25] that first uses a traditional method (e.g., a random forest [2]) to extract regions of an image that contains a hand object. Having these regions, single-hand pose estimation methods are applied to each of them. However, a general framework with more powerful detectors that can fulfill multi-hand instance detection and hand joint localization simultaneously could be more reliable and convenient in real-world applications.

Recently, convolutional Neural Networks (CNN) have become a mainstream technique in computer vision tasks such as image classification [14], pose estimation [4,9] and object detection [22]. In [11], a multi-task learning framework named Mask R-CNN [11] was proposed for simultaneous object detection and instance segmentation. Mask R-CNN is a generic multi-task learning pipeline that can be generalized to multi-human pose estimation. Because minimal domain knowledge for human pose estimation is exploited, Mask R-CNN is not applicable to model joint relationships explicitly. Moreover, as pointed out in [3], key points might not be localized accurately in complex situations.

In this paper, we propose a Pictorial Structure (PS) [1] model-based framework to address limitations of methods based on Mask R-CNN by refining the output from these networks with a learned global structure of the current hand pose during the test stage. The overall structure of our proposed method is shown in Fig. 1. Our framework is composed of two stages: first, Mask R-CNN is adopted to predict possible key point locations (Fig. 1c) and segments each hand from the given images (Fig. 1d). Then, we utilize the instance segmentation output of Mask R-CNN to approximate the pose prior of each hand (Fig. 1d–g) and add this constraint in pose space. Finally, key point locations are estimated via combining local information and global constraints (Fig. 1f).

The main contributions of our work are:

- a new method for 2D multi-hand pose estimation from a single depth image.
- a PS model-based method to find global structure constraints of a hand pose online and two ways to implement the method.
- two multi-hand datasets, dexter2Hands and NYU2Hands, that are based on the popular single-hand datasets dexter1 [23] and NYU hand pose dataset [25].

2 Related Work

In this section, we first briefly review some relevant hand pose estimation algorithms with CNN. Because estimating body and hand pose share some similarities, algorithms for one object can be extended to serve the other. Further, related multi-human pose estimation methods are also reviewed. Finally, we introduce the Mask R-CNN framework, which serves as the baseline for our research.

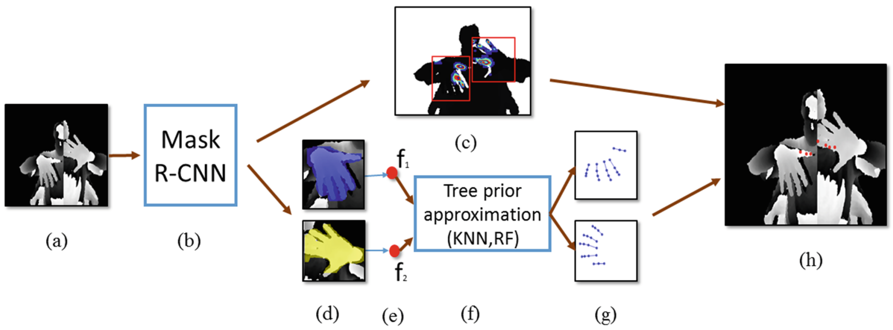

Fig. 1. Example of how our method localizes joints of left thumb finger and right index finger. Given an input image (a), we first use Mask R-CNN (b) to detect bounding boxes, possible joint positions (c), and hand segmentations (d). Then, we extract global features (e) of each hand from Mask R-CNN and find hand poses similar to input hands from training data (f). Afterwards, we compute global constraints of input hands (g). Final hands joint positions are localized by combining the local information and global constraints (h).

2.1 Hand Pose Estimation

More recently, CNN has been widely used in hand pose estimation. Authors in [25] first used CNN for predicting heat maps of joint positions, and this method was improved in [8] by predicting heat maps on three orthogonal views to better utilize the depth information. In [18], a multi-stage CNN that enforces priors to hand poses was presented to directly regress hand joints. Authors in [9] presented a 3D CNN that regresses 3D hand joint positions directly. In [27], a three-stage approach that can estimate 3D hand poses from regular RGB images was proposed. In that approach, the hand is first located by a segmentation network and serves as input to another network for 2D hand pose estimation. The final 3D hand joint positions are localized via combining the estimated 2D positions and the 3D pose prior information.

2.2 Multi-human Pose Estimation

In [7], a PS model-based framework was proposed for estimating poses of multiple humans, but it relies on an additional human detector and simple geometric body part relationships. Similarly, the model proposed in [15] also requires a human detector for initial human hypotheses generation, and the estimation of key points positions and instances are divided into two stages. Unlike previous strategies that need to first detect people and subsequently estimate their poses, the method proposed in [21] utilizes CNN for body part hypotheses generation and is able to jointly solve the task of detection and pose estimation. This work was extended in [13] with stronger part detectors and more constraints in the problem formulation.

2.3 Mask R-CNN

Mask R-CNN is a general framework for object instance segmentation and human pose estimation. It consists of two stages. In the first stage, candidate object bounding boxes are proposed by the Region Proposal Network (RPN). In the second stage, features of each candidate bounding box are extracted and classification, bounding box regression, instance segmentation and key point detection are performed. Unlike methods proposed in [5,16,20] whereby classification depends on mask prediction, Mask R-CNN applies a parallel strategy that can simultaneously solve tasks in stage two. The overall network architecture of Mask R-CNN contains a convolutional backbone used to extract features over the whole image and three parallel network heads: one for classification and bounding box regression, and two for the remaining tasks.

3 Problem Formulation

Mathematically, our objective is to estimate hand poses $\mathbf{P} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_M\}$ from a single image I, where $\mathbf{X}_i$ denotes the pose of an instance and M is the number of instances in I. Following [1], we assume that a hand can be decomposed into a set of parts, the pose of a hand is defined as $\mathbf{X}_i = \{\mathbf{x}_i^n | 1 \leq n \leq N, \forall \mathbf{x}_i^n \in \Re^3\}$, where the state of part n is formulated as $\mathbf{x}_i^n = \{\mathbf{y}_i^n, t_i^n\}$. $\mathbf{y}_i^n = \{x_i^n, y_i^n\}$ is the position of the key point in image coordinate system and $t_i^n = \{0, 1\}$ denotes the state indicating the presence of part n.

We formulate the multi-hand poses estimation problem as finding the maximum posteriori of poses given an image I, i.e., $p(\mathbf{P}|I)$, which can be approximated as

$$p(\mathbf{P}|I) \propto p(I|\mathbf{P})p(\mathbf{P}), \tag{1}$$

where $p(I|\mathbf{P})$ is the likelihood of the image evidence given particular poses, and the $p(\mathbf{P})$ corresponds to poses prior. We assume that all hands are independent for simplicity, Eq. 1 can be factorized as

$$p(\mathbf{P}|I) \propto \prod_{i=1}^{M} p(I|\mathbf{X}_i)p(\mathbf{X}_i), \tag{2}$$

where $p(I|\mathbf{X}_i)$ is the likelihood of the image evidence given a particular pose, and the $p(\mathbf{X}_i)$ corresponds to a kinematic tree prior according to the Pictorial Structure [1] (PS) model, though this may not always hold when fingers of different hands are crossed. We propose a general framework based on PS model and utilize Mask R-CNN [11] to solve Eq. 2.

4 Mask R-CNN for Hand Pose Estimation

In this work, we use ResNet-50 [12] with Feature Pyramid Network (FPN) [17] as the backbone to extract features of the entire image. For details of ResNet

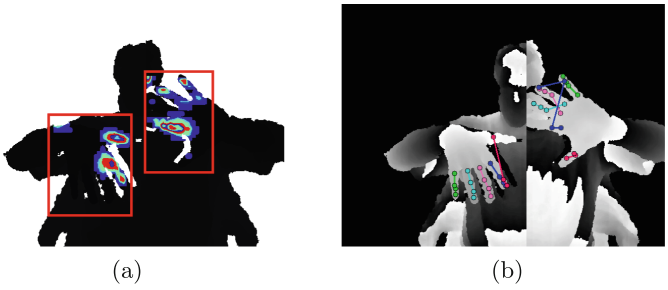

Fig. 2. (a) Confidence maps of left thumb finger joints and right index finger joints. (b) Mask R-CNN detection result.

and FPN, we refer readers to [12,17]. For the network head, we follow the three-parallel-branches architecture presented in [11] whereby one branch is for bounding box classification and regression, one for instance mask prediction and one for key point detection. In general, given a training image, features of the entire image are first extracted by the ResNet-FPN backbone. Based on the features, RPN generates a set of ROIs. Each positive ROI is fed into three parallel branches of the network head: one branch for bounding box classification and the other two for remaining tasks. The loss function is defined as $L = L_{cls} + L_{box} + L_{mask} + L_{kpt}$, where the classification loss L_{cls} is log loss over two classes (hand vs. background). The bounding box regression loss L_{box} is identical as that defined in [10]. The mask loss L_{mask} is the binary cross-entropy loss over predicted hand mask and groundtruth and the key point mask loss L_{kpt} is the average cross-entropy loss over the predicted N joints and N groundtruth points.

At test time, Mask R-CNN key point head branch outputs confidence maps of all joints. Figure 2(a) shows an example of confidence maps of left thumb finger joints and right index finger joints. Because relationships among hand joints are only implicitly learned during the training process, localizing key point positions via finding locations with maximum probabilities could lead to large pixel error. As shown in Fig. 2(b), two joints of the left thumb finger are estimated incorrectly on the left index finger. Similarly, joints of the right index finger are incorrectly predicted as the ring and little finger. Moreover, if we cannot guarantee the correctness of confidence maps, they cannot be used alone to infer the presence or visibility of joints. Inspired by PS models by which the poses of objects can be estimated by combing global structure constraints (which encode part relationships) and part confidence maps, we utilize the output of Mask R-CNN mask head to learn kinematic structures of hands explicitly. Learned kinematic structures are used to refine confidence maps of corresponding hands and infer presences of joints.

5 Confidence Refinement

Confidence maps provide probabilities of each joint position, which can be viewed as $p(I|\mathbf{P})$ in Eq. 2. According to the PS model, the prior $p(\mathbf{X}_i)$ is supposed to encode probabilistic constraints on part relationships and capture the unified global structure of objects in the training data. We present a conceptually simple method to approximate the tree prior $p(\mathbf{X}_i)$ and two methods to implement it.

5.1 Tree Prior Approximation

As illustrated in Fig. 1(d), masks predicted by Mask R-CNN mask head capture global structures of hand instances, but they lack information on part relationships (e.g., neighbouring joints of the same finger should lie close to each other). Our idea is to find a training subset $\mathbf{S}_i$ that has a similar mask as the i_{th} test hand mask, then the kinematic tree prior that encodes part relationships of the test hand can be learned from $\mathbf{S}_i$.

Before we introduce how we find out $\mathbf{S}_i$, there is one critical question: can we make masks comparable when they may have different scale and size? In Mask R-CNN, the mask head branch would first predict a fixed size mask for each instance and the predicted mask is further resized to have the true size of the corresponding instance. We reshape the fixed size mask into a feature vector so that every hand instance can be represented in a comparable form. This feature representation projects the instances to the feature space that visually similar instances are close to each other. Feature vectors of all hand instances in training data are extracted by the same procedure and stored on disk for future use.

Unsupervised Learned Tree Prior Approximation. Given that the i_{th} hand instance can be represented by a feature vector $\mathbf{f}_i$, we use K nearest neighbours (KNN) search to find features of training images that lie close to $\mathbf{f}_i$ in the feature space, $\mathbf{S}_i$ is composed of those corresponding training images. In order to learn $p(\mathbf{X}_i)$ from $\mathbf{S}_i$, for simplicity, we assume that all hand parts are independent, the prior $p(\mathbf{X}_i)$ is approximated as

$$p(\mathbf{X}_i) \approx p(\mathbf{x}_i^1, \mathbf{x}_i^2, ..., \mathbf{x}_i^N|\mathbf{f}_i) = \prod_{j=1}^{N} p(\mathbf{x}_i^j|\mathbf{f}_i) \tag{3}$$

where $p(\mathbf{x}_i^j|\mathbf{f}_i)$ is the j_{th} part prior of i_{th} hand instance based on the feature vector $\mathbf{f}_i$. Let $coord = (x, y)$ denote the coordinate of a pixel in image, $p(\mathbf{x}_i^j|\mathbf{f}_i)$ is computed as

$$p(\mathbf{x}_i^j|\mathbf{f}_i) = \begin{cases} 1 & ||coord - mean_{\mathbf{S}_i}^j||_p \leq d \\ 0.5 & \text{otherwise} \end{cases} \tag{4}$$

where $|| \bullet ||_p$ is the Minkowski distance between two points and $mean_{\mathbf{S}}^j$ is the mean coordinate of the j_{th} part in $\mathbf{S}_i$. d is a hyper-parameter that adjusts the

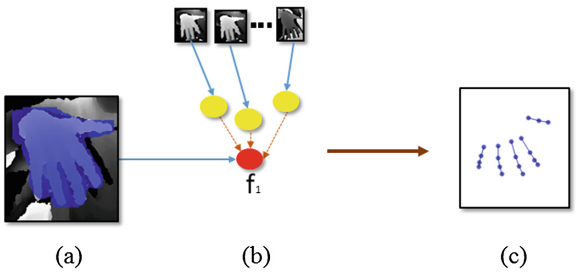

Fig. 3. KNN for hand instance kinematic prior approximation. A hand instance (a) is expressed by a feature vector $\mathbf{f}_1$, training data with similar features are found out by KNN search (b). The kinematic structure of the hand instance are learned from those training data (c).

influence of $p(\mathbf{X}_i)$. We adopt this formulation because it allows faster computation than other common probabilistic distributions and it is mainly defined to refine joint confidence maps. Though in our formulation we assume that all joints are independent, joint relationships are implicitly preserved by the subset of training data in $\mathbf{S}_i$. Figure 3 shows an example of this process. The absence of joint(s) is inferred by the absent joints in $\mathbf{S}_i$, e.g., if the number of absent tips of the ring finger from $\mathbf{S}_i$ result is greater than a threshold τ, the ring finger tip is deemed as invisible for the i_{th} hand instance.

Because the whole process needs to be repeated for every hand instance, KNN-based tree prior approximation method is computationally heavy. Moreover, features of training data need to be stored, which may require a large amount of space. These limitations motivate us to find $\mathbf{S}_i$ via other methods that require less computation and storage.

Supervised Learned Tree Prior Approximation. It is possible to use a supervised learning method to find out $\mathbf{S}_i$, which should be faster than KNN, provided that a labelling method could be found that is able to distinguish different hand poses. In our framework, a hand instance is assigned a label $L = \{j_1, j_2, ..., j_N\}$, where the index of j_i in the label vector indicates the joint name and N is the number of joints. We first compute distances between each hand joint and the origin. Those computed distances are stored in a vector $\mathbf{v}$, then we sort $\mathbf{v}$ in decent order. The value of j_i is determined by the index of the corresponding joint plus one in sorted $\mathbf{v}$. For example, if a sorted $\mathbf{v}$ is of the form $\mathbf{v} = \{dist(joint_2, org), ..., dist(joint_1, org)\}$, where $dist$ is the function computes the distance between two points, $joint_i$ is the coordinate of a joint and org is the coordinate of the origin, the values of j_1 and j_2 are N and 1 in the label vector L . If a joint is not visible, the corresponding entry in L is set to 0. In most cases, the presented labelling method is able to distinguish different hand poses and preserve joint spatial relationships, especially when we need to localize all joints and tips of a hand.

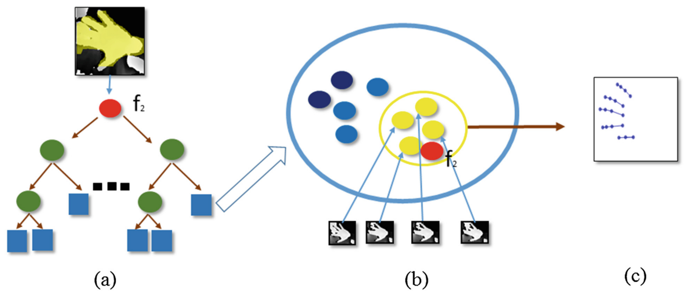

Fig. 4. Random forest for hand instance kinematic prior approximation. The feature vector $\mathbf{f}_2$ of hand instance is classified into a class by RF (a). Training data of the same label in nodes that $\mathbf{f}_2$ falls into are selected (b) and used to compute the kinematic prior (c).

The next step is to choose a proper classifier. We select Random Forest [2] (RF) because it is naturally designed for multi-class classification and it provides soft decision boundaries. Moreover, RF is able to handle high dimensional input data efficiently, which allows fast computation at test time. Figure 4 shows an example of how we use RF to predict the kinematic tree prior of a test hand. Feature vector $\mathbf{f}_2$ of the test hand goes through all trees and falls into some leaf nodes (Fig. 4a). It is assigned a label l by RF and we select training data with the same label l (Fig. 4b), which is actually the training subset $\mathbf{S}_i$. The kinematic tree prior (Fig. 4c) is estimated by Eq. 3. In practice, kinematic tree priors learned from each leaf node can be computed offline and it is only necessary to store joint coordinates, bounding box width and height, i.e., totally $N \times 2 + 2$ numbers, which requires much less storage space compared to our KNN method. Absences of joints or tips can be directly predicted by RF (entry in the label vector is 0).

5.2 Final Localization

Given $p(I|\mathbf{P})$ and $p(\mathbf{X}_i)$, the posterior probability $p(\mathbf{P}|I)$ can be computed by Eq. 2. Joints locations are estimated by finding image positions with highest probabilities. Note that both our tree prior approximation methods are able to detect presences of joints; if Mask R-CNN failed to detect the j_{th} joint of the i_{th} hand, the position of the j_{th} joint is estimated by $mean^j_{\mathbf{S}_i}$.

6 Data Preparation

We generated two 2-hands datasets, dexter2Hands and NYU2Hands, based on depth images of the popular single hand datasets dexter1 [23] and NYU hand pose dataset [25]. For the dexter2Hands dataset, we randomly selected 2504 images from 3154 images in the dexter1 dataset as a training set, and the remaining 600 images were equally split into a validation set and test set.

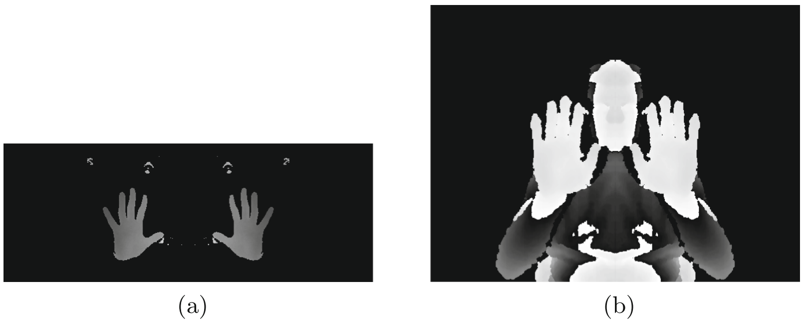
Fig. 5. (a) Sample image of Dexter2Hands dataset. (b) Sample image of NYU2Hands dataset.

Because images of dexter1 only contained hands and the image size was relatively small (320×240), images in the final training data of dexter2Hands are of size 640×240, and are generated by the concatenation of randomly selected left and right hand images from (mirrored-)training set. Same processes are applied to generate validation data and test data of dexter2Hands dataset. In our experiments, dexter2Hands training data contained 57404 images, validation data contained 14025 images and test data contained 9925 images. The key point number of a hand instance is 5, which are thumb finger tip, index finger tip, middle finger tip, ring finger tip and little finger tip, respectively. Figure 5(a) shows an example of images in dexter2Hands. Hand masks of the dexter2Hands dataset are generated by setting pixel values of hand object in each image to 1 and background to 0.

Processes used to generate the NYU2Hands dataset are similar, and we use only depth images from the view-point 1. However, the image size of NYU2Hands is the same as NYU, which is 640×480. Training data and validation data of NYU2Hands are generated by copying the mirrored left side hand (in image coordinate) to be the corresponding right side hand. The key point number of a hand instance is 19, which are little finger tip (LT), little finger joint 1 (L1), little finger joint 2 (L2), little finger joint 3 (L3), ring finger tip (RT), ring finger joint 1 (R1), ring finger joint 2 (R2), ring finger joint 3 (R3), middle finger tip (MT), middle finger joint 1 (M1), middle finger joint 2 (M2), middle finger joint 3 (M3), index finger tip (IT), index finger joint 1 (I1), index finger joint 2 (I2), index finger joint 3 (I3), thumb finger tip (TT), thumb finger joint 1 (T1) and thumb finger joint 2 (T2), respectively. Figure 5(b) shows a sample image of NYU2Hands. Because there are 75157 images in the NYU hand pose dataset with the same background, we randomly selected 62727 images to generate training data and 10000 images to generate validation data. We applied the same strategy of generating dexter2Hands training data to generate test data of NYU2Hands, which contained 6038 images. Synthetic depth images provided by the NYU dataset are used to generate training hand masks of the NYU2Hands dataset.

7 Implementation Details

7.1 Mask R-CNN

Training: In our experiments, parameters of Mask R-CNN backbone are initialized by Imagenet [6] pre-trained weights. Training depth images are converted into 3-channel images by replication. We train the model on 50K iterations for dexter2Hands and 60K iterations for NYU2Hands, starting from a learning rate of 0.002 and reducing it by 10 at 15K and 35K iterations. Models are trained on 4 Nvidia GTX 1080 GPUs. Each batch has 1 image per GPU and each image has 128 sample ROIs. Other implementations are identical as [11].

Inference: At test time, the bounding box branch directly predicts bounding boxes of hand instances. The instance segmentation branch predicts a mask of size 28×28 and the key point mask branch outputs a $56 \times 56 \times$ N joint mask for each hand instance. N is 5 for dexter2Hands and 19 for NYU2Hands. Those masks are further resized to the size of the bounding box, and binarized at a threshold t to obtain the final detection result. t is 0.1 for instance masks and 0.5 for key point masks. Instance threshold is chosen at a low value because we hope the estimated mask could cover hand finger tips. The feature vector of a hand instance is generated by reshaping the 28×28 mask into a vector of 1×784.

7.2 Tree Prior Approximation

For KNN search, we set $K = 10$ and threshold $\tau = 4$ for both datasets. For our RF approach, we use the RF implementation provided by [19] to construct a 10-tree RF and do not change other parameters. Each tree has a depth of around 30 and around 6000 leaf nodes. We choose Manhattan distance to compute part prior $p(\mathbf{x}_i^j|\mathbf{f}_i)$ in Eq. 4 since it is relatively fast and d is set to 30 for dexter2Hands dataset and 40 for NYU2Hands dataset.

8 Experiments

8.1 Evaluation

We evaluate our methods on test data of Dexter2Hands and NYU2Hands. Results of our methods are compared with two versions of Mask R-CNN, i.e., keypoint only and keypoint & mask, as well as groundtruth joint positions. Mask R-CNN keypoint only indicates that joint positions are localized via finding positions of joint confidence maps with maximum probabilities. Mask R-CNN keypoint & mask restricts keypoints lying on estimated masks. We employ two metrics to evaluate the performance of our proposed method. The first metric is the average Euclidean distance in pixels between the results and the groundtruth. The second metric is the percentage of success frames in which all joint errors are below a certain threshold. In addition, we compute the false positives (FP) rate and false negatives (FN) rate to infer the presence of each joint to validate the adequacy of our methods. In our experiments, we found that Mask R-CNN is able to correctly detect almost all hand instances, with fewer than 5 frames being wrongly detected.

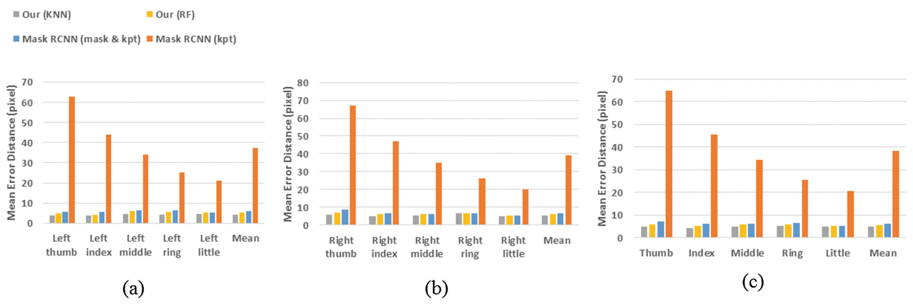

Fig. 6. Per-joint mean error distance in pixels on dexter2Hands. (a) Left hand. (b) Right hand. (c) Both hands.

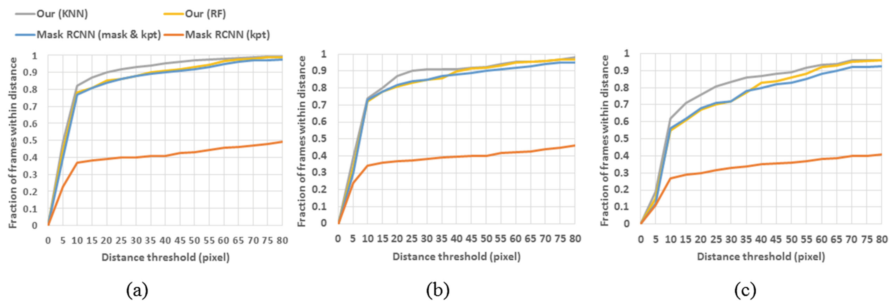

Fig. 7. Fraction of frames within distance on dexter2Hands. (a) Left hand. (b) Right hand. (c) Both hands.

8.2 Results and Discussion

Figure 6 shows the comparison results of our methods and Mask R-CNN on Dexter2Hands dataset. In all cases, we can see that our methods produce fewer pixel errors of each tip on each individual hand and both hands. Because the image background of this dataset is relatively clean, estimating joint locations via finding positions with maximum probabilities without constraint is noise sensitive. This is the reason for the large pixel errors in the method of Mask R-CNN keypoint only. As shown in Table 1, the average joint pixel error over all frames of our KNN method is 4.8, which is better than our RF method (5.7) and Mask R-CNN keypoint & mask method (6.2). The fraction of good frames over a different threshold for each individual hand and both hands is shown in Fig. 7. For the left hand, our KNN method achieves the best good frame rate (82%) when the threshold is 10 pixels, while the good frame rate is 78% for our RF method and 77% for mask R-CNN keypoint & mask. Similarly, the performance of our KNN and RF methods outperform other methods on the right hand (Fig. 7b) and both hands (Fig. 7c).

Table 1. Quantitative evaluation on Dexter2Hands.

Method	Position error (pixels)	FN (%)	FP (%)
Our (KNN)	4.8	1	1
Our (RF)	5.7	0	1
Mask RCNN (kpt & mask)	6.2	2	3
Mask RCNN (kpt)	38.5	2	3

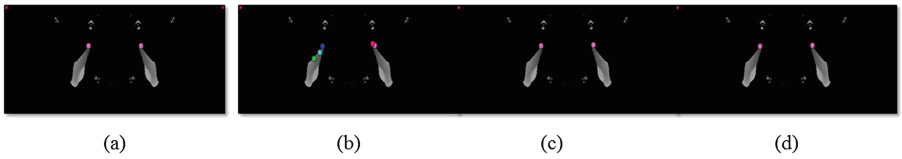

Fig. 8. Examples of our methods compared to Mask-RCNN on Dexter2Hands dataset. (a) Groundtruth. (b) Outputs of Mask RCNN (with mask). (c) Outputs of of our KNN method. (d) Outputs of our RF method.

Another advantage of our methods is that they are able to infer the presence of joint visibilities. Figure 8 shows a typical example. Given an input image with groundtruth that only the middle fingers of both hands are visible (Fig. 8a), Mask R-CNN wrongly predicts that pinky, ring, middle and index finger tips are visible on the left hand. Similarly, all finger tips are estimated to be overlapping on the right hand (Fig. 8b). Our methods successfully detect the presence of joints and correctly predict visible joint position (Fig. 8c, d). Both versions of Mask R-CNN produce FN rates of 2% and FP rates of 3%, while FN rate of our KNN and RF methods are 1% and 0%. The FP rate of our methods are 1%.

We also compare our methods with Mask R-CNN on the NYU2Hands dataset, which is more challenging since there are 19 joints on each hand. As shown in Fig. 9, our methods achieve fewer pixel errors than Mask R-CNN in all cases. Mean pixel errors of the left middle finger tip (MT in Fig. 9) of Mask R-CNN are 27.3 (keypoint only) and 22.4 (keypoint & mask), while mean pixel errors of our methods for that joint are 11.2 (KNN) and 12.2 (RF). For the right hand, though on some joints (e.g., Fig. 9b: L1, RT, MT, etc.) Mask R-CNN keypoint & mask has fewer pixel errors than our RF method, the largest margin is on the right middle finger tip, which is 1.4 (11.1 vs 12.5). Table 2 shows the averaged position errors in pixel for different methods. Mean joint pixel errors over all frames of our methods are 9.3 (KNN) and 10.1 (RF), which is better than Mask R-CNN keypoint & mask (12.4) and keypoint only (16.2). The proportion of good frames over different error thresholds is shown in Fig. 10, and we can see a clear order of performance of the four methods: our KNN method is better than our RF method and the proposed methods outperform Mask R-CNN. The FN and FP rates of our methods are all 0%, while FN rates of both versions of Mask R-CNN are 2% and FP rates are 0%. Some qualitative results for the

NYU2Hands dataset are shown in Fig. 11. As can be seen, our proposed methods can better preserve hand joint relationships and provide a more accurate estimation.

Table 2. Quantitative evaluation on NYU2Hands.

Method	Position error (pixels)	FN (%)	FP (%)
Our (KNN)	9.3	0	0
Our (RF)	10.1	0	0
Mask RCNN (kpt & mask)	12.4	1	0
Mask RCNN (kpt)	16.2	1	0

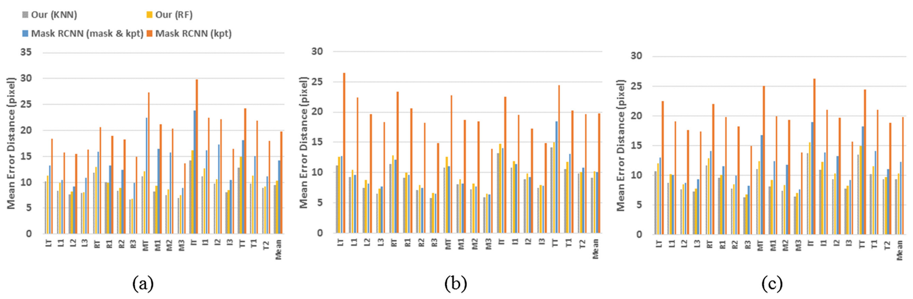

Fig. 9. Per-joint mean error distance in pixels on NYU2Hands. (a) Left hand. (b) Right hand. (c) Both hands.

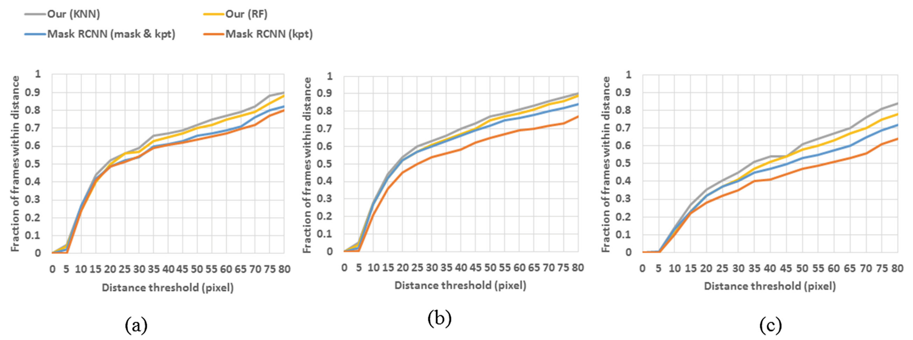

Fig. 10. Fraction of frames within distance on NYU2Hands. (a) Left hand. (b) Right hand. (c) Both hands.

Runtime: The runtime of both versions of Mask R-CNN to process a test image of dexter2hands dataset is 0.45 s on average, and it takes 0.5 s for our KNN method and 0.46 s for our RF method. For the test image of NYU2Hands dataset, the averaged process time of both versions of Mask R-CNN is 0.5 s because the image size is two times larger than test images of dexter2Hand and needs to locate more joints. The process time of our KNN method for the NYU2Hands dataset is around 0.85 s per image, including 0.25 s for the calculation of mean joint positions for each joint in KNN search result. Compared to our KNN method, our RF method is much faster because mean joint positions are already stored after training, which requires 0.55 s to process a NYU2Hands test image.

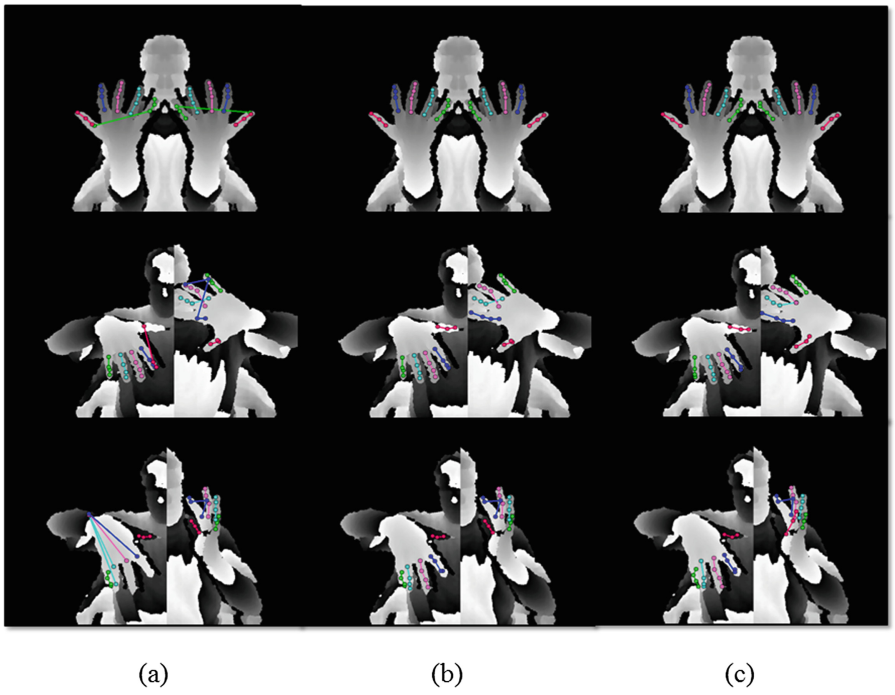

Fig. 11. Examples of our methods compared to Mask-RCNN on NYU2Hands dataset. (a) Outputs of Mask RCNN (with mask). (b) Outputs of our KNN method. (c) Outputs of our RF method.

9 Conclusion and Future Work

We present a new algorithm based on the PS model for estimating 2D multi-hand poses from single depth images. The proposed framework utilizes Mask R-CNN to learn the mapping from local informations of joints and global structures of hands to their corresponding poses. We formulate a new utilization of the segmentation output of Mask R-CNN and propose two ways to approximate pose priors of test instances. The estimated pose priors could be used to infer the presences of joints. Our method addresses issues of interchangeable estimations by solely using Mask R-CNN for the detection of hand key points. We also present interplays between Mask R-CNN and the PS model, as well as Mask R-CNN and random forests. The performance of our algorithm has been validated on two self-generated datasets with two hands that can also serve as a baseline for future research.

Future work will encompass generating a real multi-hand dataset with accurate labelling that not only labels the joint position but also provides visibility information of occluded joints. Our system could be extended to 3D multi-hand pose estimation and an improved method could be designed to model the relationships of joints, both in network structure design and the tree prior approximation step.

References

1. Andriluka, M., Roth, S., Schiele, B.: Pictorial structures revisited: people detection and articulated pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 1014–1021. IEEE (2009)
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
3. Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., Sun, J.: Cascaded pyramids network for multi-person pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7103–7112. IEEE (2018)
4. Chu, X., Ouyang, W., Li, H., Wang, X.: Structured feature learning for pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4715–4723 (2016)
5. Dai, J., He, K., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3150–3158 (2016)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: CVPR (2009)
7. Eichner, M., Ferrari, V.: We are family: joint pose estimation of multiple persons. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6311, pp. 228–242. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15549-9_17
8. Ge, L., Liang, H., Yuan, J., Thalmann, D.: Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3593–3601 (2016)
9. Ge, L., Liang, H., Yuan, J., Thalmann, D.: 3D convolutional neural networks for efficient and robust hand pose estimation from single depth images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, p. 5 (2017)
10. Girshick, R.: Fast R-CNN. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448. IEEE (2015)
11. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the International Conference on Computer Vision (ICCV) (2017)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

13. Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., Schiele, B.: DeeperCut: a deeper, stronger, and faster multi-person pose estimation model. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 34–50. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_3
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
15. Ladicky, L., Torr, P.H., Zisserman, A.: Human pose estimation using a joint pixel-wise and part-wise formulation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3578–3585 (2013)
16. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2359–2367 (2017)
17. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR, vol. 1, p. 4 (2017)
18. Oberweger, M., Wohlhart, P., Lepetit, V.: Hands deep in deep learning for hand pose estimation. arXiv preprint arXiv:1502.06807 (2015)
19. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
20. Pinheiro, P.O., Collobert, R., Dollár, P.: Learning to segment object candidates. In: Advances in Neural Information Processing Systems, pp. 1990–1998 (2015)
21. Pishchulin, L., et al.: Deepcut: joint subset partition and labeling for multi person pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4929–4937 (2016)
22. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
23. Sridhar, S., Oulasvirta, A., Theobalt, C.: Interactive markerless articulated hand motion tracking using RGB and depth data. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), December 2013
24. Tang, D., Jin Chang, H., Tejani, A., Kim, T.K.: Latent regression forest: structured estimation of 3D articulated hand posture. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3786–3793 (2014)
25. Tompson, J., Stein, M., Lecun, Y., Perlin, K.: Real-time continuous pose recovery of human hands using convolutional networks. ACM Trans. Graph. **33**, 169 (2014)
26. Yuan, S., et al.: Depth-based 3D hand pose estimation: From current achievements to future goals. In: IEEE CVPR (2018)
27. Zimmermann, C., Brox, T.: Learning to estimate 3D hand pose from single RGB images. In: International Conference on Computer Vision (2017)