Deep Multi-Task Learning based Fingertip Detection

Jiewen Zhao Ruize Han* Xiaoyu Zhang Liang Wan*
College of Intelligence and Computing, Tianjin University
{zhaojw, han_ruize, xiaoyuzhang, lwan}@tju.edu.cn

Abstract

Human-computer interaction (HCI) is significantly depended on hand behavior detection, e.g., hand action recognition, gesture recognition. However, fingertip detection is more practical in many HCI cases, e.g., finger pointing, clicking. Although the great progress in general object detection, fine-grained target detection, e.g., fingertip detection is still a challenging problem for existing methods because of the limited appearance features of the fingertip. In this paper, we present a simple yet effective approach for fingertip detection by designing a multi-task learning based framework. More specifically, we introduce a point regression loss into the object detection framework to jointly accomplish the hand region detection and fingertip location detection tasks. We then propose a new point regression loss referred to as root loss to pay more attention to small-range errors for more precise fingertip localization. Our framework further achieves number-variant fingertip detection without the default number of fingertips. We use the benchmark EgoGesture for performance evaluation. Experimental results show the superiority of the proposed method and the ablation studies verify the effectiveness of each component. Our method achieves the state-of-the-art fingertip detection accuracy while maintaining the real-time algorithm speed.

1. Introduction

With the advent of artificial intelligence era and the emergence of smart wearable camera, such as Google Glass [47, 7, 14, 13], human-computer interaction (HCI) has become more and more prevalent. As a bridge for HCI, fingertip detection has gradually become a new topic in computer vision research. Given an image containing an arbitrary gesture, fingertip detection is to locate the positions of all the visible fingertips in the entire image. Compared to hand action and gesture recognition, fingertip detection is more practical in many HCI cases. It is common for a person to use his/her fingertips to interact with the computer in both real-world and virtual environments. The localization of fingertip can record the desired position at a specific moment, e.g., when the finger is pointing or clicking. The trajectories of the fingertip can also record the passing curve for a while, which can be a specific shape, even a character representing an interactive instruction. In this paper, we focus on the fingertip detection from a single RGB image.

Fingertip detection can be regarded as a specific object detection task. However, it is challenging to detect the fingertip using a classical object detection method, e.g., SSD [25], Yolo [33] based model, because the fingertip is so small with very limited features. Recently, many fingertip detection methods are proposed by using hand-crafted features [20] [32], or deep features [19] [18] [42]. As shown in the top of Figure 1, all the previous works treat the fingertip detection as two separate tasks – the hand detection and fingertip detection. They first search the hand region in the entire image and detect the fingertip in the cropped hand region. Besides, in the fingertip detection process, previous methods can only handle the specific scenario, e.g., 1) there is only one visible fingertip in the image [20] [32] [19] [18]; 2) the number of fingertips is fixed and given in advance [42], and the test images containing the hands with different numbers of fingertips need different pre-defined models.

In this paper, we treat the fingertip detection as a multi-task learning process. We simultaneously handle two tasks, i.e., hand detection and fingertip detection. As shown in Figure 1, the input images contain the hands with a different number of fingertips. We attempt to obtain the hand region and the fingertip position together due to the two tasks always go hand-in-hand and have inter-related features. Specifically, we first extract the features of the input images using a pre-trained Convolutional Neural Network (CNN). We introduce the multi-task learning concept, which shares features between different related tasks. The extracted features are fed into three branches, i.e., the classification branch, box regression branch, and point regression branch. Among them, the first branch is used to estimate the confidence of a candidate region in containing the hand and fingertips. The box regression branch is to detect the hand region, which is like the classical object detection task. The point regression branch is used for fingertip local-

*Corresponding authors.

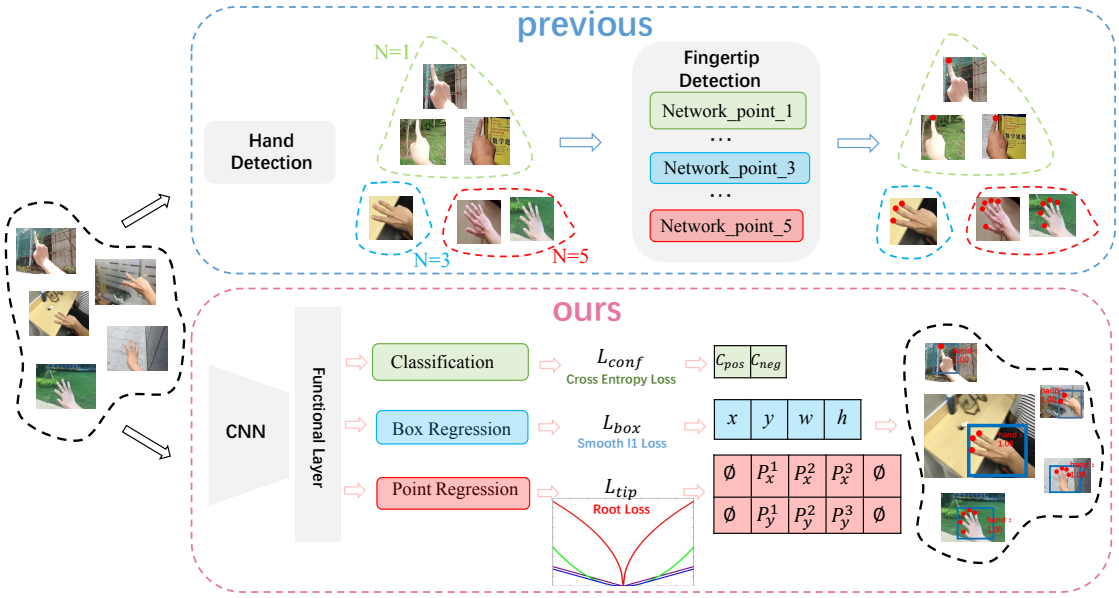

Figure 1: The framework of previous two-stage fingertip detection method (top) and the proposed multi-task learning based fingertip detection method (bottom).

ization. We propose a novel root loss function, as shown in Figure 1, which is more sensitive to small errors and facilitates more precise fingertip localization results. In fingertip detection, we arrange the fingertips in order and divide them into real points (visible fingertips) and null points (invisible fingertips). Then we propose a simple yet effective multi-point regression strategy to uniformly handle the samples with number-variant fingertips. To sum up, given a set of images containing the hands (with different number of fingertips) and a messy background, the proposed method can jointly obtain the hand region and fingertip position with high precision.

Our major contributions are three-fold.

- We propose to use a multi-task learning based framework to simultaneously get the hand and fingertip detection results instead of separately obtaining them using the cascaded network in prior works.
- We develop a new point regression loss for more precise fingertip localization and the multiple fingertip detection strategy to detect the number-variant fingertips.
- We show the proposed end-to-end framework outperforms previous fingertip detection and state-of-the-art object detection methods while maintaining a high running speed of over 60 fps.

The rest of this paper is organized as below. The related works are classified and summarized in Section 2. The proposed method is elaborated in Section 3. The performance of the proposed method is extensively evaluated in Section 4. Finally, a conclusion is discussed in Section 5.

2. Related Work

Object Detection. Fingertip detection can be regarded as a kind of object detection. Almost all of state-of-the-art object detection methods are based on deep learning, which can be divided into two major categories: one-stage detector and two-stage detector. Two-stage detectors first extract candidate regions from the images via a proposal generator, e.g., selective search [41]. Then they classify and predict the object bounding box from each extracted candidate region. These kind of methods, e.g., R-CNN [10], fast R-CNN [9], faster R-CNN[34], and SPPNet [31], are challenging to meet very high speed due to the need for extracting candidate regions before detection. For one-stage detectors, they do not need to generate region proposals and directly generate class probability and position coordinates of different objects through a single shot detection, such as [23, 33, 28, 25, 21]. In Yolo [33], it uses a fully connected layer after convolution layers to get the detection results directly. SSD [25] gets the final detection results with default boxes at different scales and different aspect ratios. Although existing methods have a significant increase in general object detection, they still perform poorly on small objects, e.g., fingertip. Fingertip detection is also similar to hand keypoints detection. However, most

previous approaches on hand keypoints detection rely on depth data [2, 36, 48, 40, 29, 30]. Only recently, Simon et al. [38] used multiview bootstrapping to create a sizeable real-world dataset of hands annotated with 2D keypoint positions. Another work [50] estimates 3D hand gestures and gets 3D hand key points from regular RGB images. Above hand keypoint detection methods focus on the relative positions of hand joints and are used to estimation 3D hand pose/gesture, which is also distinguishing of the fingertip detection task.

Fingertip Detection. Early methods of fingertip detection leverage the relevant properties of the skin, like color information [20]. In this work, the fingertip detection is performed by separating the hand region using the skin color. Raheja et al. [32] generated a binary silhouette of the hand region using a skin filter based on HSV color space. As for these methods, the most serious problem is the heavy dependence on skin color information, which makes these kind of methods unuseful in complex environments. There are also several methods that make use of some other information, e.g., depth information and skeleton information [22] [11]. More recently, CNN based fingertip detection has been studied. In [19], a cascaded CNN framework is proposed by combining a Faster R-CNN based hand detector and a multi-point fingertip detector. The work in [26] replaces the Faster R-CNN based hand detection in [19] with an attention-based hand detector and improves the accuracy of fingertip detection. Huang et al. [18] introduced a HSV space-based color feature for fingertip detection. Previous fingertip detection methods are used for the gesture with only one visible fingertip until YOLSE [42] uses the finger map to achieve multiple fingertip detection. However, above existing fingertip detection methods need the hand bounding box as input, which can be obtained by the manual annotation or an extra hand detector [8, 37].

Multi-Task Learning. Multi-Task Learning (MTL) has recently become a popular topic in computer vision. The basic idea of MTL has been proposed long ago by Caruana in [3]. Since then, MTL has gradually been introduced into various fields [4, 6, 17]. The rationale of MTL is to leverage the inter-relationships between multiple tasks, primarily by jointly using the features of each task [35, 24, 27, 15]. Through the inter-related characteristics, each task can be promoted by other tasks. MTL is very suitable for deep learning tasks because of the features learned from a task can be used in other tasks. It has been proved that MTL can improve the performance of each task in many computer vision problems [44][45]. The combination of MTL and the deep neural network has been widely used in face detection, pose estimation, and landmark localization, e.g., [49, 46, 12, 1]. Zhang et al. [46] implemented facial landmark detection by exploiting the features of other tasks such as head pose estimation and facial attribute inference. Similarly, the main idea of [12] is to use Deep Multi-Task Learning approach to jointly estimate multiple heterogeneous attributes from a single face image. However, there is currently no MTL based method for fingertip detection.

3. The Proposed Method

3.1. Overview

In this paper, we attempt to synergistically get the hand region and fingertip positions from a single RGB image. We propose a fingertip detection method based on deep multi-task learning (MTL) architecture, which is a framework to simultaneously perform hand detection and fingertip detection. We use an end-to-end network to avoid the complex process of detecting the hand first and then locating the fingertip.

In the proposed MTL framework, a CNN is first used to extract feature maps, then the shared output features are fed into three branches for different tasks using corresponding loss function. In order to further improve the precision of fingertip detection, we propose a novel loss function, root loss, by increasing the impact of small errors on the prediction results. The root loss makes the keypoints regression more accurate. Therefore, it is more suitable for fingertip detection. Besides, due to the uncertainty of the number of fingertips, we propose a unified multiple fingertip regression strategy to make the number-variant fingertip detection feasible. In the following, we will detail the multi-task network architecture, proposed regression loss, and the multiple fingertip regression strategy.

3.2. Multi-task Learning Architecture

Given a RGB image, we attempt to simultaneously complete two tasks of hand detection and fingertip detection together by a unified end-to-end network. We implement a multi-task learning architecture that shares the same feature extraction network and uses different branches for multiple tasks as shown in Figure 2. Among them, the classification branch is used to estimate the confidence of a candidate region containing the hand and fingertip. The hand regression branch and fingertip regression branch are used to regress the hand bounding box and fingertip point coordinates, respectively.

We present the detailed network architecture in Figure 2. Considering the speed performance, we employ a fully convolution network. The network takes a RGB image with a uniform size as input. Then it extracts CNN features through the common network structures, e.g., MobileNet [16], VGG [39]. We will discuss the accuracy and speed performance using different CNN structures in Section 4. After feature extraction, we use two feature maps directly from the previous CNN network and four feature maps through extra convolution operations. As shown in

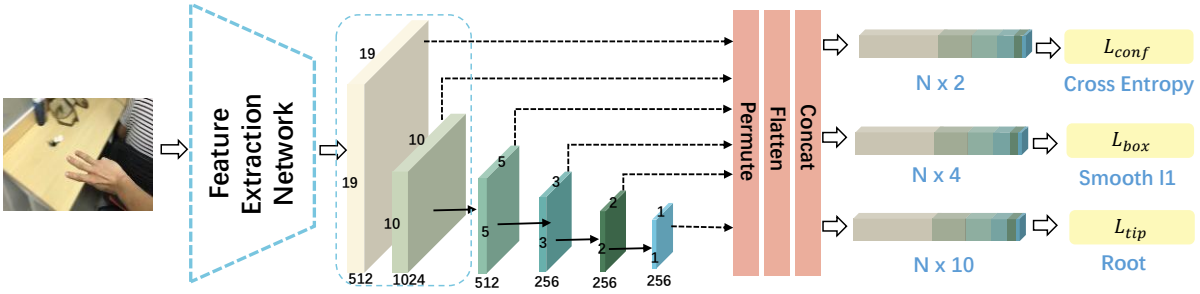

Figure 2: Architecture of our multi-task learning network. The network is composed of a basic feature extraction network, shared convolution layers and three branches. Among them, the classification branch is used to classify the candidate region. The hand and fingertip regression branch are used to regress the hand region and fingertip localization, respectively.

Figure 2, the cubes with different colors and sizes represent the extracted feature maps in different layers. Next, these six feature maps are operated by a convolution kernel with channel number of $C_1 + C_2 + C_3$, as shown by the dotted arrow in Figure 2. All the output features are fed into the permute and flatten layer and then concatenated as three feature vectors. The sizes of three concatenated feature vectors in three branches are $N \times C_i$ $(i = 1, 2, 3)$, where N represents the number of anchors (candidate bounding boxes) in all six feature maps. As shown in Figure 2, the first branch with the vector size of $N \times C_1$ represents the confidence of classification. The parameter C_1 is two because there is only one class, where C_1 taking values of 1 and 0 represents the prediction of yes and no, respectively. The last two branches represent the coordinates of the hand bounding box (four coordinates) and the fingertips (ten coordinates), respectively. Therefore C_2 and C_3 are 4 and 10, respectively.

In these six feature maps, we assume that S_i $(i = 1, 2..6)$ represents the size of the i-th feature map. So, N can be calculated as $N = \sum_{i=1}^{6} S_i \times \alpha$, where α represents the default number of anchors per location in each feature map. The selection of anchors is similar to SSD [25], which are some pre-defined boxes of different sizes and different aspect ratios. A detailed training strategy will be introduced in Section 3.5.1.

3.3. The Proposed Root Loss

At present, the typical loss functions of the regression task are L_1 loss, L_2 loss and smooth L_1 loss, as shown in Figure 3 (a). Among them, the smooth L_1 loss is widely used in the general object detection task, while all the previous fingertip detection methods use L_2 loss as the regression loss. From Figure 3 (a), we can see that the loss values grow faster as the values of the x-axis increase, i.e., L_2 loss is very sensitive to the significant errors. In contrast, it reduces the impact of small errors on the loss value. At this point, some researchers develop to use smooth L_1 loss in object detection, which is a combination of L_1 loss and L_2 loss. From the curve shown in Figure 3 (a) and (b), we can see that the smooth L_1 loss alleviates the excessive sensitivity to the outliers in L_2 loss. However, it does not increase the sensitivity to small errors. Actually, in fingertip detection, we aim to optimize the fingertip point coordinate errors between predicted and ground truth results, in which we should pay more attention to the small and medium errors. This way, we propose a new loss function as shown in Figure 3 (b).

Similar to the usage of piecewise function in smooth L_1 loss, we propose a new loss function for fingertip point regression. The difference from L_1 loss is that for the fewer errors, we use a function whose gradient gradually increases as the coordinate value of x-axis approximates to zero. Such a setting can help the loss function be more sensitive to small errors. In this paper, we consider using the r-th root function $\sqrt[r]{(\cdot)}$ to satisfy above condition.

As discussed above, we propose a piecewise loss function, which is composed of a nonlinear function $\sqrt[r]{(\cdot)}$ for small errors and a linear function $|(\cdot)|$ for large errors. The formulation of the proposed loss function, referred to as Root loss, is defined as follow:

$$\text{Root}\,(x) = \begin{cases} \omega \cdot (\sqrt[r]{\frac{|x|}{\phi} + \epsilon} - \sqrt[r]{\epsilon}), & \text{if } |x| < \omega \\ |x| - C_r, & \text{otherwise} \end{cases} \quad (1)$$

$$C_r = \omega - \omega \cdot (\sqrt[r]{\frac{\omega}{\phi} + \epsilon} - \sqrt[r]{\epsilon}), \quad (2)$$

where ω mainly controls the length of the nonlinear region into $(-\omega, \omega)$. The parameters ϕ and r jointly control the curvature of the function curve. Moreover, we can see that the derivative of $\sqrt[r]{z}$ is $\frac{1}{r}(\frac{1}{z})^{\frac{r-1}{r}}$. The r-th root function $\sqrt[r]{z}$ has a non-derivable condition at zero point $(z = 0)$.

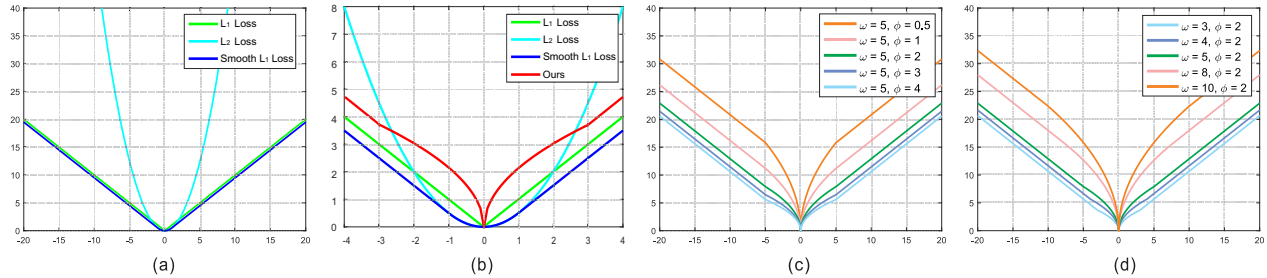

Figure 3: (a) An illustration of three loss functions – L_1 loss, L_2 loss, and smooth L_1 loss. (b) A comparison of our loss and the baseline loss functions within a small range. (c-d) An illustration of the proposed root loss function with different parameters ω and ϕ, where we fix $r = 2$.

Thus we add an offset ϵ to ensure that it can be derivable at any point. The parameter C_r is the link between the linear region and the nonlinear region in the loss function, which can be computed by above four parameters.

In practice, we set $r \geq 2$ in the function $\sqrt[r]{z}$ to satisfy the condition that the smaller z has a larger gradient. The value of ϕ can not be set as too small value. Note that if ϕ is too small, the value of gradient will be huge for small errors, which will cause the exploding gradient problems. The function curves with different parameters are shown in Figure 3 (c) and (d). We will show the advantage of the proposed root loss for fingertip regression and the impact of different parameter selections in Section 4.

3.4. Multiple Fingertip Regression Strategy

Multiple fingertip detection is a task to localize the positions of all the visible fingertips under different gestures, which has a different number of fingertips. Due to the uncertainty of gestures, it is a challenging task to determine the number of fingertips meanwhile localizing the positions of them. As we know, CNN based keypoint detection methods require providing the number of keypoints in advance. Different tasks require different network structures to match the specific number of output points. So, we focus on how to use a unified network to achieve number-variant multiple fingertip detection.

In order to solve the above problem, we propose a multiple fingertip regression strategy. Inspired by the fixed-number keypoints detection, we introduce the regression label of the invisible fingertip point. Specifically, the multiple fingertip regression loss is defined as follows:

$$\mathrm{L}_{\mathrm{tip}} = \sum_{i \in \mathcal{R}} \mathrm{R}(|P_i - P_i^{\mathrm{gt}}|) + \sum_{j \in \mathcal{N}} \mathrm{R}(|P_j - P_j^{\mathrm{gt}}|), \quad (3)$$

where R denotes the proposed root loss, P and P^{gt} represent the predicted and the ground-truth results, respectively. We regard the visible fingertips as *real points* and the invisible fingertips as *null points*, which are represented as

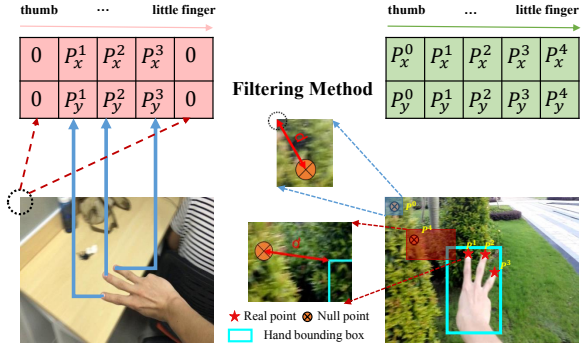

Figure 4: Illustration of the proposed multiple fingertip regression strategy. In the training stage (left), we regress the real points (P^1-P^3) into the positions of fingertips and null points into the upper left corner. In the testing stage (right), we first obtain all the possible points and then filter out the null points by considering two conditions. As shown in the middle, we filter out the detected points closing to the upper left corner or far away from the hand.

$\mathcal{R}$ and $\mathcal{N}$, respectively. For different hands with the arbitrary number of fingertips, we all produce five keypoints (in a hand), including the real points and null points. As mentioned above, the key to complete the regression task is to introduce the Dummy Regression Label (DRL) of the null points. For a training image, the DRLs of the null points are uniformly set as the upper left corner of the training image, as shown in Figure 4 (left). Attributed to the above setting, the regression results of the null points tend to be near the upper left corner. The result of real points will be regressed to the corresponding positions of fingertips.

In the testing stage, after getting the regression results, we need to complete the filtering operations in order to get the true fingertips. We use two methods to filter out the non-finger points, as shown in Figure 4 (right). First, attributed to our multi-task learning architecture, we can constrain the position of the fingertips by the region of the hand bounding box. In most cases, the fingertip will be inside the hand

region. So, we filter the detected points far away from the hand, where the distance d between them is larger than a threshold value δ. Second, we filter out the points quite closing to the upper left corner (null point regression label), i.e., a predicted fingertip is removed if its distance d to the upper left corner is smaller than the threshold value δ.

3.5. The Framework

3.5.1 Training loss

Our framework is designed to complete two different tasks, i.e., hand detection and fingertip detection. As shown in Figure 1, we simultaneously optimize three losses in different branches thus propose to minimize the following objective function

$$\mathrm{L} = \mathrm{L}_{\mathrm{conf}}(\mathcal{C}, \mathcal{C}^{\mathrm{gt}}) + \mathrm{L}_{\mathrm{box}}(\mathcal{B}, \mathcal{B}^{\mathrm{gt}}) + \mathrm{L}_{\mathrm{tip}}(\mathcal{P}, \mathcal{P}^{\mathrm{gt}}). \quad (4)$$

The total loss function of the entire framework is composed of three parts, i.e., the classification loss $\mathrm{L}_{\mathrm{conf}}$, hand regression loss $\mathrm{L}_{\mathrm{box}}$, and fingertip regression loss $\mathrm{L}_{\mathrm{tip}}$. We use the two-class softmax loss function, smooth L_1 loss function and the proposed root loss function for the above three losses.[1] The sets $\mathcal{C}$ and $\mathcal{C}^{\mathrm{gt}}$ indicate the classification confidence and ground-truth class of all anchors. The sets $\mathcal{B}$ and $\mathcal{P}$ indicate the coordinate sets of the predicted hand bounding box and fingertip point of the anchors, $\mathcal{B}^{\mathrm{gt}}$ and $\mathcal{P}^{\mathrm{gt}}$ indicate the corresponding ground truth.

3.5.2 Implementation details

Our model is implemented by PyTorch on an NVIDIA GTX-2080Ti GPU. We use the SGD algorithm as the optimizer for learning the network parameters. During the training phase, the learning rate is set to 1×10^{-3}, and decays by 5×10^{-4} per epoch. We set the batch size as 32 in the training process. In the experiments, we use MobileNet for feature extraction. Similar with [25], we set the number of anchors $\alpha = 6$ in the experiment. We set the parameter r as 2 and the regular term ϵ as 10^{-8} in Eq. (1). The parameters ω and ϕ are experimentally set to 3 and 2, respectively. We will analyze the parameters setting of ω and ϕ in Section 4.4. The threshold value δ for filtering error predictions is set to 30px. Note that all the parameters are fixed for different testing data in our experiments.

[1]Here, we use the smooth L_1 loss for hand bounding box regression rather than root loss because that the main advantage of the proposed root loss is for accurate point regression, e.g., fingertip or other hand/facial landmark detection. However, the hand detection is a region detection task, which regresses both the key points (e.g., the upper-left corner of the hand box) and the box size. Also, the appearance features at the regressed key points, in most cases, contains no discriminative representation of the hand.

4. Experimental Results and Analysis

4.1. Setup

4.1.1 Dataset and metrics

We use the public dataset EgoGesture[42] to evaluate the proposed method, which contains eleven single-hand gestures subsets and five double-hand gestures subsets. The whole dataset is made up of single RGB images with the size of 640×480 pixels captured by the egocentric-view camera. The dataset is collected from many different conditions, e.g., complex backgrounds, lighting changes, different user hands and directions, skin-like backgrounds, etc. The variety of conditions can avoid the unity of training data. In order to meet the needs of common application scenarios, we use five subsets with commonly used gestures (single hand with one to five fingers) in our experiment following [42]. Among them, the subset of single one finger has a total of 3,374 images, including 12 different scenes. The number of images is 300 in most scenes, but it varies in some some individual scenes. For the other four subsets, each has 3,780 images, including seven different scenes, and the number of images in each scene is 540. In order to make the experimental results more robust, that is, the training and test data have relatively obvious differences, we take the last 10% in each scene of each dataset as the test set and the rest as the training set. We do not randomly divide the training/test dataset, which can avoid the situation in which a test image is very similar to the adjacently sampling images in the training set. Finally, we have a total of 16,645 images as the training data and 1,849 images as the testing data.

We apply standard classification and object detection metrics for evaluating fingertip detection performance. We use precision, recall and their comprehensive evaluation index F_1-score to mainly evaluate the performance of fingertip detection. When calculating the precision and recall, we consider the fingertips as correct detection results if their errors are no more than a specified distance error threshold. For the fingertip detection, we tend to be more inclined to detect the precise positions in practical applications. We may be more concerned about the distance precision of each fingertip. We also introduce a metric to measure the distance error of all the detected fingertips – Mean Distance Error (MDE). MDE is calculated as the mean value of all fingertips with distance errors no more than the distance error threshold.

4.1.2 Comparative methods

We choose two approaches as the baseline methods. The first one is YOLSE [42] that is specifically for fingertip detection, and the other is the current classical object detection method SSD [25]. Note, we do not find the pub-

lic code of YOLSE, and the splitting of the training and test set is not reported in the published paper. Therefore, we re-implement this work fully in accordance with the details described in [42]. For a fair comparison, we train/test the method using our well-defined training set and test set. YOLSE is not an end-to-end method to detect fingertips, which needs the ground truth hand area as inputs. Besides, similar to the proposed method, we train a unified network of YOLSE to handle the image with a different number of fingertips, which is different from the training method in [42]. SSD, as a object detection method, requires the target bounding box as input. For a fair comparison, we take the fingertip coordinates as the center and the surrounding five pixels area as the bounding box of the fingertip. When evaluating, the center point coordinate of each predicted box is taken as the predicted position of the fingertip. We evaluate the fingertip detection performance in this way: as for every ground-truth fingertip, we find the fingertip prediction point with the smallest distance from the ground truth among the detected fingertips. If the minimum distance error is within our established threshold, we consider this detection point as the true point corresponding to the ground-truth fingertip. Then we repeat above operation for other fingertips.

4.2. Comparative Results

We compare the proposed method with SSD and YOLSE. Figure 5 shows the variation of Recall, Precision, F_1-score, and MDE of all methods against different *normalized* distance error thresholds [5] in x-axis, which is the given thresholds (discussed in Section 4.1.1) divided by the diagonal length of the image. The value in the legend indicates the area under the curve (AUC) scores, where the area surrounded by the axises is one. We can see that the AUC values of our methods using MobileNet (Ours) and VGG network (Ours_V) are higher than the AUC values of other methods, i.e., SSD_V, SSD_M and YOLSE, as shown in Figure 5 (a), (b) and (c). This demonstrates that the proposed multi-task framework is very effective for detecting fingertips.

As shown in Figure 5 (a), YOLSE has a significant increase in the recall value as the normalized distance error threshold increases. It indicates that YOLSE can roughly detect the fingertips, but the localization results are not accurate enough. It also demonstrates the correctness of our reimplementation of YOLSE. As for the reason why YOLSE has insufficient localization accuracy under small error threshold, there are two reasons as follow: 1) The parameters used to generate the finger map in YOLSE are not reported in the paper. Different parameter settings generate the Gaussian maps with different sizes, which affects the final predicted position of the fingertip. More details can be found in [42]. 2) There may be some tricks in the original implementation of this algorithm, which is also an important factor affecting the final localization accuracy. From Figure 5 (d), we can see that our method is slightly lower than SSD with VGG on the mean distance error (MDE). However, the algorithm efficiency and detection accuracy of SSD with VGG are lower than our method, which will be discussed in Section 4.4.2.

Table 1: Comparison results of SSD, YOLSE and our method. SSD_V, SSD_M denote SSD detector with VGG-16 and MobileNet as the feature extraction network, respectively.

Methods	Precision	Recall	F_1-score	MDE
SSD_V	86.1%	70.0%	77.2%	**3.29**
SSD_M	88.5%	82.2%	85.2%	5.30
YOLSE	41.5%	51.4%	45.9%	12.54
YOLSE @30	61.2%	74.0%	67.0%	16.13
YOLSE @45	71.2%	85.1%	77.5%	18.64
Ours	**97.4%**	**87.7%**	**92.3%**	3.85

In the following, we show the quantitative evaluation results using a specific distance error thresholds, e.g., 20 pixels[2]. As shown in Table 1, the first two rows show the fingertip detection performance of SSD with different backbone networks. We can see that SSD provides an acceptable result, especially using the backbone VGG, the mean detection error (MDE) of SSD is quite small. However, it provides relatively poor fingertip detection accuracy, i.e., precision, recall and F_1-score, compared to our method. Although obtaining acceptable performance, SSD can not determine the number of fingertips in a testing image (e.g., the detected results may exceed 5 points) nor distinguish the order (i.e., from the thumb to little finger) of the detected fingertips.

We also compare the proposed method with YOLSE as shown in the third row of Table 1. Specifically, 'YOLSE' denotes the performance at the position error in default distance error threshold. We can see that neither the F_1-score nor the MDE is good enough. We increase the error threshold into 30 and 45 pixels and evaluate the performance as shown in the fourth and fifth rows in Table 1. We can see that the F_1-score gets better as the error threshold increases. However, we can also see that the MDE score gradually gets worse. The comparison results show that YOLSE is not suitable for accurate fingertip localization. On the contrary, the proposed method has a good tradeoff between detection accuracy and localization precision.

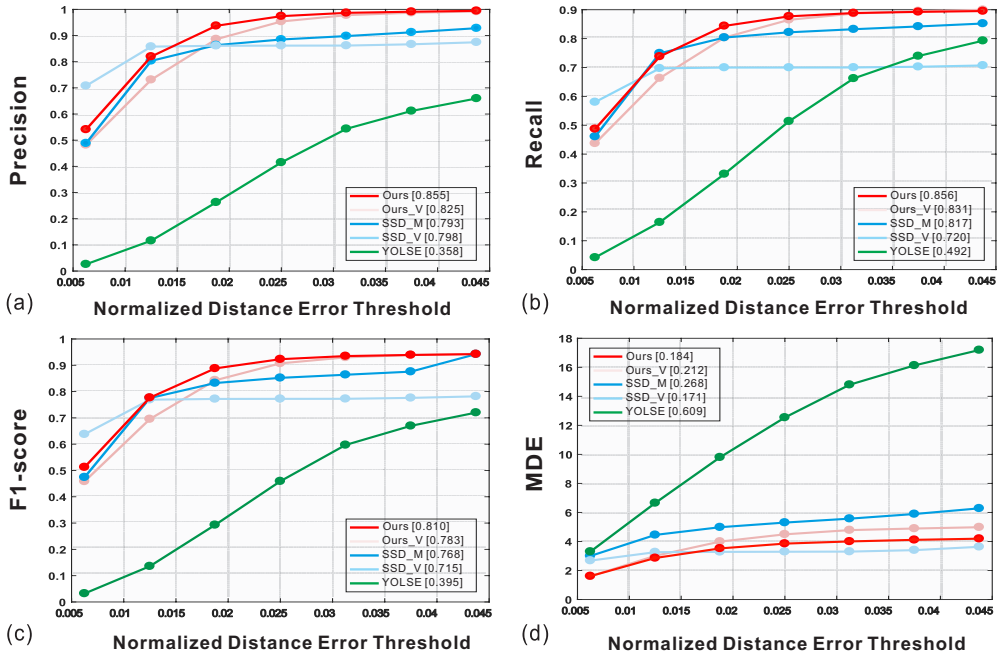

Figure 5: Fingertip detection performance under different distance error thresholds. We use precision, recall, F_1-score and mean distance error (MDE) for performance evaluation. The value in the legend indicates the area under the curve (AUC) scores, the larger scores in (a-c) while smaller scores in (d) indicate the better performance.

Table 2: Ablation study of the proposed method using different feature extraction network, network architecture and regression loss.

Type	Precision	Recall	F_1-score	MDE
w VGG	95.4%	86.5%	90.7%	4.49
w/o L_{box}	94.5%	84.9%	89.4%	4.11
w/o DRL	35.8%	31.7%	33.7%	6.14
w L_1	96.5%	86.6%	91.3%	3.94
w L_2	62.4%	56.2%	59.1%	6.17
w Smooth L_1	96.7%	87.2%	91.7%	4.05
Ours	**97.4%**	**87.7%**	**92.3%**	**3.85**

4.3. Ablation Study

4.3.1 Feature extraction network

We first study the influence of the feature extraction network. We compare two popular networks, i.e., VGG-16 and MobileNet, in our experiments. The VGG-16 is the backbone network used in the SSD framework. We use the VGG-16 network in our framework, i.e., 'w VGG', and evaluate the fingertip detection performance as shown in the first row in Table 2. We can see that the proposed method using MobileNet as the feature extraction network, i.e., 'Ours', generates better detection results.

[2]We set the threshold as 20 pixels in our experiments referring the target center localization error (CLE) threshold defined and used in [43].

4.3.2 Network component

We also evaluate the effectiveness of the hand detection branch in the proposed method. As shown in Table 2, 'w/o L_{box}' denotes that we remove the hand bounding box regression loss in our framework. We can see that both the F_1-score and the MDE score get worse without the box regression loss. The ablation experiment verifies that the integration of hand region regression can not only generate the hand detection results but also promote the performance of fingertip detection. We further verify the validity of the proposed multiple fingertip regression strategy. Specifically, we remove the dummy regression label (DRL), i.e., the second term in Eq. (3), for null points regression. We can see that 'w/o DRL' provides a very poor precision score, which is because the invisible fingertips can not be identified by an unified regressed localization (i.e., the upper-left corner in our method). It also negatively affects the performance of visible fingertip detection therefore provides a low recall score. This indicates the effectiveness of the multiple fingertip regression strategy for number-variant fingertip detection.

Table 3: Results by varying values of w and ϕ. We evaluate fingertip detection results by varying one of these two parameters while fixing another parameter, i.e., $w = 3$, $\phi = 2$.

w	**Precision**	**Recall**	**F_1-score**	**MDE**	ϕ	**Precision**	**Recall**	**F_1-score**	**MDE**
2	97.1%	88.0%	92.3%	3.94	1	96.1%	86.6%	91.1%	4.00
3	**97.4%**	**87.7%**	**92.3%**	**3.85**	**2**	**97.4%**	**87.7%**	**92.3%**	**3.85**
5	95.8%	86.1%	90.7%	4.36	3	96.6%	87.1%	91.7%	3.98

Table 4: The running speed and fingertip detection performance of different methods. SSD_V, SSD_M and Ours_V, Ours_M denote SSD detector and the proposed method with VGG-16 and MobileNet as the feature extraction network, respectively. The speed of YOLSE is only for the fingertip detection in the given hand area.

Method	SSD_V	SSD_M	YOLSE (w Hand)	Ours_V	Ours
Speed (fps)	50.1	66.7	357.1	33.3	66.9
F_1-score	77.2%	85.2%	45.9%	90.7%	92.3%
MDE	3.29	5.30	12.54	4.49	3.85

4.3.3 Fingertip regression loss

Next, we compare the results using different loss functions in the fingertip regression branch. We replace the root loss with three popular loss functions, i.e., L_1 loss, L_2 loss and smooth L_1 loss. The comparative results are shown in the last four rows in Table 2. We can see that the L_2 loss provides poor results in fingertip detection. It is because L_2 loss is very sensitive to the large distance errors, i.e., the loss value has large difference with other losses when the error increases as shown in Figure 3 (a), and it is insensitive to the small errors. This characteristic leads to that L_2 loss is not appropriate in our task. We can also see that the L_1 loss and smooth L_1 loss provide much better results, which are more suitable for precise fingertip detection. Compared to the above three loss functions, the proposed root loss provides the best performance in both F_1-score and MDE score. The comparison results verify the effectiveness of the root loss in keypoint detection.

4.4. Algorithm Analysis

4.4.1 Parameter analysis

As described in Section 3.5, there are two free parameters ω and ϕ in Eq. (4). We select different values for them and see their influence on the final detection accuracy of fingertips detection. The illustration of the function curves using different parameters are shown in Figure 3 (b)(c). Table 3 shows the detection results by varying one of these two parameters while fixing another parameter. We can see that the final detection results are not sensitive to the selected values of ω and ϕ, either the fingertip detection accuracy metrics, i.e., Precision, Recall and F_1-score or the localization error metric, i.e., MDE. We can see that the parameters setting of $\omega = 3$ and $\phi = 2$ provides the best detection results.

4.4.2 Speed analysis

We analyze the speed efficiency of our proposed method. Since different methods use different data processing methods, the time we calculate is only the running time of inference phase. We run our method and all the comparison methods on the same environment – NVIDIA GTX 2080Ti GPU. From Table 4, we can see that our approach using MobileNet as the backbone network runs faster than our approach with VGG. This is mainly because the VGG network is significantly more complicated than the MobileNet. We can also see that our approach using MobileNet runs faster than SSD algorithm using MobileNet/VGG. We notice that our method is much slower than YOLSE. There are two main reasons for this. The first reason is that YOLSE takes the hand area as input, so the search region is smaller than the other methods. The second reason is that YOLSE has fewer network layers and the size of all convolution kernels is 3×3 in our reproduced network, which makes the inference of the network very fast. However, YOLSE produces very poor detection performance compared to other methods.

4.4.3 Qualitative analysis

To further evaluate the advantages of our algorithm, we take some representative cases to qualitatively analyze the results as shown in Figure 6. There are five different gestures with the detection results using three methods, i.e.,

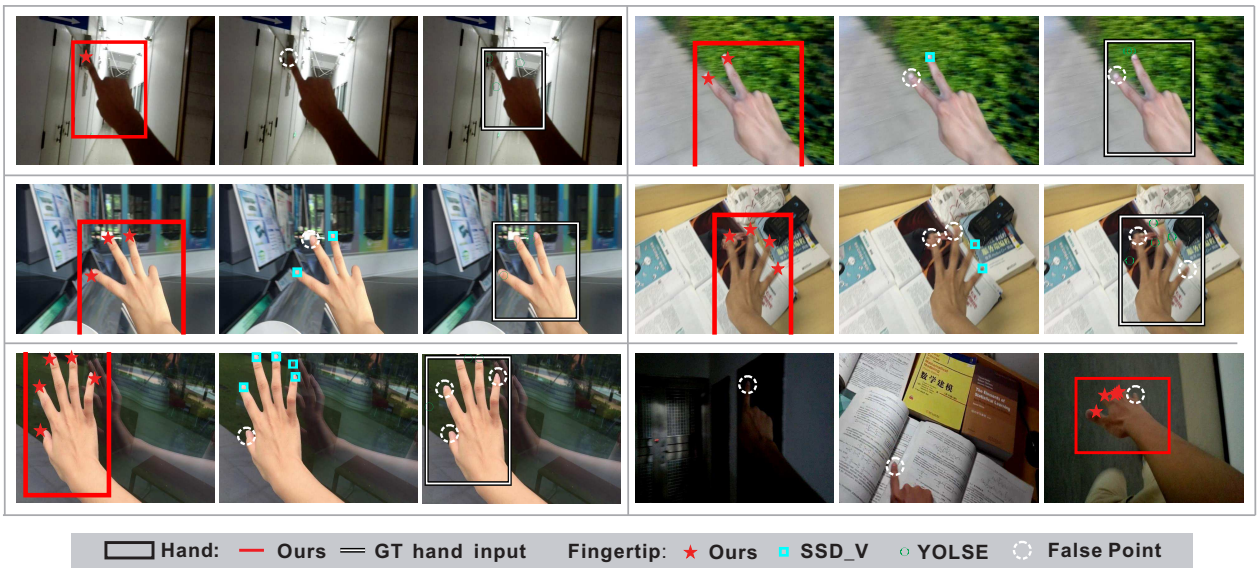

Figure 6: Qualitative analysis and failure case analysis. The first two rows and third row (left) illustrate the comparison results of three methods in five cases. The third row (right) shows three failure cases of the proposed method.

Ours, SSD_V and YOLSE. Each group containing three images represents a kind of gesture, and each image illustrates the detection result of Ours, SSD_V and YOLSE, respectively. The first row (left) shows the scenario where the light is dark, and we can see that the SSD_V shows a poor result while YOLSE detects several wrong points. The first row (right) and the second row show another situation: the background is cluttered, which frequently occurs in the real world. We can notice that all the fingertip points are not detected by SSD_V and YOLSE. SSD_V also gets poor detection results when the background is complexed, e.g., the images shown in the left of the second row. The third row (left) shows another case: there is a mirrored hand in the image. We can notice that SSD_V redundantly detects the points in the mirror, and YOLSE also provides poor results under the influence of the mirror. In contrast, our approach provides the more robust results in these cases. It can be seen from the figure that our method can accurately localize the tip of the finger while detecting the hand region.

4.4.4 Failure case analysis

At last, we present three failure cases, as shown in the third row (right) of Figure 6. We can see that the ambient light is very dark (the first case), and there is only a part of the hand visible in the image (the second case). The proposed method fails to detect the fingertip because the hand is not successfully detected in the first two cases. The last failure case shows a hand with serious deformation. In this case, our method can directly detect the hand area. However, the positions of fingertips can not be accurately detected because most of the fingers are blocked due to serious deformation.

5. Conclusion

In this paper, we have developed a new multi-task learning based framework to detect the fingertip without depending on the pre-given hand region and the number of fingertips. Our approach constructs three branches using the classification loss, box regression loss and point regression loss, respectively, to obtain the hand region and fingertip localization simultaneously. In the fingertip detection, we propose a new loss for more precise point localization and a multiple fingertip detection strategy to handle the number-variant fingertips. On the benchmark dataset EgoGesture, we validates the superiority of our approach over the state-of-the-art competitors. The ablation study verifies the effectiveness of each proposed component. In the future, we consider integrating the network compression and multithreading acceleration strategy to the proposed model, which makes our framework efficiently applied to the mobile device.

References

[1] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia. Multi-task cnn model for attribute prediction. *IEEE TMM*, 17(11):1949–1959, 2015.

[2] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. In *CVPR*, 2003.

[3] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[4] K. Du, X. Lin, Y. Sun, and X. Ma. Crossinfonet: Multi-task information sharing based hand pose estimation. In *CVPR*, 2019.

[5] Z. Feng, J. Kittler, M. Awais, P. Huber, and X. Wu. Wing loss for robust facial landmark localisation with convolutional neural networks. In *CVPR*, 2018.

[6] D. Fourure, R. Emonet, E. Fromont, D. Muselet, N. Neverova, A. Trmeau, and C. Wolf. Multi-task, multi-domain learning. *Neurocomputing*, 251:68–80, 2017.

[7] Y. Gan, R. Han, L. Yin, W. Feng, and S. Wang. Self-supervised multi-view multi-human association and tracking. In *ACM MM*, 2021.

[8] Q. Gao, J. Liu, and Z. Ju. Robust real-time hand detection and localization for space human–robot interaction based on deep learning. *Neurocomputing*, 2019.

[9] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[11] H. Guo, G. Wang, and X. Chen. Two-stream convolutional neural network for accurate rgb-d fingertip detection using depth and edge information. *arXiv preprint arXiv:1612.07978*, 2016.

[12] H. Han, A. K. Jain, S. Shan, and X. Chen. Heterogeneous face attribute estimation: A deep multi-task learning approach. *IEEE TPAMI*, PP(99):1–1, 2017.

[13] R. Han, W. Feng, Y. Zhang, J. Zhao, and S. Wang. Multiple human association and tracking from egocentric and complementary top views. *IEEE TPAMI*, 2021.

[14] R. Han, J. Zhao, W. Feng, Y. Gan, L. Wan, and S. Wang. Complementary-view co-interest person detection. In *ACM MM*, 2020.

[15] E. M. Hand and R. Chellappa. Attributes for improved attributes: A multi-task network utilizing implicit and explicit relationships for facial attribute classification. In *AAAI*, 2017.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[17] B. Hu, J. Lai, and C. Guo. Location-aware fine-grained vehicle type recognition using multi-task deep networks. *Neurocomputing*, 243:60–68, 2017.

[18] Y. Huang, X. Liu, L. Jin, and Z. Xin. Deepfinger: A cascade convolutional neuron network approach to finger key point detection in egocentric vision with mobile camera. In *IEEE International Conference on Systems*, 2016.

[19] Y. Huang, X. Liu, Z. Xin, and L. Jin. A pointing gesture based egocentric interaction system: Dataset, approach and application. In *CVPRW*, 2016.

[20] S. K. Kang, Y. N. Mi, and P. K. Rhee. Color based hand and finger detection technology for user interaction. In *International Conference on Convergence & Hybrid Information Technology*, 2008.

[21] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen. Ron: Reverse connection with objectness prior networks for object detection. In *CVPR*, 2017.

[22] P. Krejov and R. Bowden. Multi-touchless: Real-time fingertip detection and tracking using geodesic maxima. In *IEEE International Conference & Workshops on Automatic Face & Gesture Recognition*, 2013.

[23] X. Li, Y. Mao, Y. Liu, and C. Zhu. Adaptive deep convolutional neural networks for scene-specific object detection. *IEEE TCSVT*, 29(99):1–1, 2017.

[24] S. Liu, E. Johns, and A. J. Davison. End-to-end multi-task learning with attention. In *CVPR*, 2019.

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[26] X. Liu, Y. Huang, Z. Xin, and L. Jin. Fingertip in the eye: An attention-based method for real-time hand tracking and fingertip detection in egocentric videos. In *Chinese Conference on Pattern Recognition*, 2016.

[27] J. Ma, Z. Zhao, J. Chen, A. Li, and L. Hong. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *AAAI*, 2019.

[28] M. Najibi, M. Rastegari, and L. S. Davis. G-cnn: an iterative grid based object detector. In *CVPR*, 2016.

[29] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a feedback loop for hand pose estimation. In *ICCV*, 2015.

[30] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, 2011.

[31] P. Purkait, C. Zhao, and C. Zach. Spp-net: Deep absolute pose regression with synthetic views. *arXiv preprint arXiv:1712.03452*, 2017.

[32] J. L. Raheja, K. Das, and A. Chaudhary. An efficient real time method of fingertip detection. *arXiv preprint arXiv:1108.0502*, 2011.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[34] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[35] B. Schölkopf, J. Platt, and T. Hofmann. Multi-task feature learning. In *NeurIPS*, 2006.

[36] T. Sharp, C. Keskin, D. P. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Human Factors in Computing Systems*, 2015.

[37] J. Si, J. Lin, F. Jiang, and R. Shen. Hand-raising gesture detection in real classrooms using improved R-FCN. *Neurocomputing*, 359:69–76, 2019.

[38] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM TOG*, 33(5):1–10, 2014.

[41] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[42] W. Wu, C. Li, C. Zhuo, Z. Xin, and L. Jin. Yolse: Egocentric fingertip detection from single rgb images. In *ICCVW*, 2018.

[43] Y. Wu, J. Lim, and M. H. Yang. Object tracking benchmark. *IEEE TPAMI*, 37(9):1834–1848, 2015.

[44] X.-T. Yuan, X. Liu, and S. Yan. Visual classification with multitask joint sparse representation. *IEEE TIP*, 21(10):4349–4360, 2012.

[45] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *IJCV*, 101(2):367–383, 2013.

[46] Z. Zhang, L. Ping, C. L. Chen, and X. Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, 2014.

[47] J. Zhao, R. Han, Y. Gan, L. Wan, W. Feng, and S. Wang. Human identification and interaction detection in cross-view multi-person videos with wearable cameras. In *ACM MM*, 2020.

[48] X. Zhou, Q. Wan, Z. Wei, X. Xue, and Y. Wei. Model-based deep hand pose estimation. In *IJCAI*, 2016.

[49] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, 2012.

[50] C. Zimmermann and T. Brox. Learning to estimate 3d hand pose from single rgb images. In *ICCV*, 2017.