J. Vis. Commun. Image R. 70 (2020) 102802

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Hand pose estimation in object-interaction based on deep learning for virtual reality applications ☆,☆☆

Min-Yu Wu [a], Pai-Wen Ting [a], Ya-Hui Tang [a], En-Te Chou [a], Li-Chen Fu [a,b,*]

[a] Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, ROC
[b] Department of Electrical Engineering, National Taiwan University, Taiwan, ROC

ARTICLE INFO

Article history:
Received 29 March 2019
Revised 5 March 2020
Accepted 21 March 2020
Available online 4 April 2020

Keywords:
Hand pose estimation
Deep learning
Convolutional neural network
Spherical part model

ABSTRACT

Hand Pose Estimation aims to predict the position of joints on a hand from an image, and it has become popular because of the emergence of VR/AR/MR technology. Nevertheless, an issue surfaces when trying to achieve this goal, since a hand tends to cause self-occlusion or external occlusion easily as it interacts with external objects. As a result, there have been many projects dedicated to this field for a better solution of this problem. This paper develops a system that accurately estimates a hand pose in 3D space using depth images for VR applications. We propose a data-driven approach of training a deep learning model for hand pose estimation with object interaction. In the convolutional neural network (CNN) training procedure, we design a skeleton-difference loss function, which effectively can learn the physical constraints of a hand. Also, we propose an object-manipulating loss function, which considers knowledge of the hand-object interaction, to enhance performance.

In the experiments we have conducted for hand pose estimation under different conditions, the results validate the robustness and the performance of our system and show that our method is able to predict the joints more accurately in challenging environmental settings. Such appealing results may be attributed to the consideration of the physical joint relationship as well as object information, which in turn can be applied to future VR/AR/MR systems for more natural experience.

1. Introduction

Nowadays, hand pose estimation has become one of the most attractive topics in the computer vision field as its application to plenty of innovative techniques, such as Virtual Reality (VR), Augmented Reality (AR), Mixed Reality (MR), and Human-Computer Interaction (HCI), have gained popularity. Development of powerful devices in VR/AR/MR, such as Microsoft HoloLens, HTC VIVE, or PlayStation VR, has made interaction between human hands and virtual objects possible. In addition, cameras capable of capturing depth images have been invented, and there are some accessible commercial depth cameras, such as Kinect or Intel RealSense, providing the depth stream. With the depth information, one can not only take advantage of the spatial relationship embedded in the image but also transform the pixel-wise depth value to the actual 3D geometry in a space. This important invention offers potential improvement for solving problems that strongly rely on the 3D relationship. From then on, joint prediction studies, such as human pose estimation and hand pose estimation, have flourished.

In real-world applications, we have many opportunities for using our hands holding objects, which means there will be lots of hand-object interactions, and we still need to find the proper hand pose in such cases. Current works about hand pose estimation focus on bare hands only, which means the research has been done only for estimating hand poses in a simple environment without considering other complex conditions. In this paper, we discuss hand-object interaction via two cases. The first case is manipulating a virtual object, which is equivalent to estimating the hand pose only. As there are few articles claiming to address the condition with hand-object interactions, the datasets of this field do not contain the manipulation setting (*i.e.* the grasping pose, the pinching pose, and so on). The other case is interacting with a real object, which can exist in partly VR, AR, or MR applications. Imagine a scenario where, in a virtual scene, one would like to hold a cup to

☆ This paper has been recommended for acceptance by Dr. Zicheng Li.

☆☆ This research was supported by the Joint Research Center for AI Technology and All Vista Healthcare under Ministry of Science and Technology of Taiwan, and Center for Artificial Intelligence & Advanced Robotics, National Taiwan University, under the grant numbers of 108-2634-F-002-016 and 108-2634-F-002-017.

* Corresponding author at: Department of Electrical Engineering and Computer Science and Information Engineering, National Taiwan University, Taiwan, ROC.
E-mail address: lichen@ntu.edu.tw (L.-C. Fu).

https://doi.org/10.1016/j.jvcir.2020.102802
1047-3203/

drink coffee. This task can be straightforward if we just make a virtual cup for him/her. Nevertheless, as emphasized in VR/AR/MR, being natural is the top priority, which gives the reason we have come up with the idea to have real objects be involved in the whole system. Moreover, we think hand-object interaction information can provide some knowledge for the hand pose estimation since people get used to manipulating objects with certain poses in normal cases. Hence, we design a hand pose estimation system that considers hand-object interaction, and we expect the system will enhance quality of life by combining VR applications with the real world for remote control, education, and so on.

Many researchers have spent a lot of effort on this intriguing topic; however, hand pose estimation itself is still a fairly challenging problem. As we already know, a human hand normally has 5 fingers, each of which contains two to three joints, and these fingers are able to move with different rotations and bends simultaneously. Observing the highly dynamic shapes of various poses performed, one of the difficulties of estimation is that a hand is far from being a rigid body, so every one of its configurations is highly articulated, which makes validation of the reconstructed hand model extremely difficult. Also, hand poses easily can change from one to another within very short time period. For example, the facts that the transition from clenching one's fist to opening one's hand can be very rapid and is conducted frequently in daily life make reconstruction difficult, let alone the fact that every individual finger may bend and straighten at any time, which increases the difficulty of the above task. Due to such high variation, traditional tracking methods fail to perform well in solving this tough problem.

Furthermore, when it comes to imaging, occlusion is a common issue, which means that we may not be able to access the information from the input source and that we have to use limited information to achieve the goal of hand pose estimation. Due to the complex configuration of a hand (*i.e.* fingers), parts of a hand tend to block each other from the view of the camera, which is quite common when capturing hand images. If a certain part of a hand is occluded by another part, we call it self-occlusion. When some difficult poses are performed, such as clenching one's fists, self-occlusion will arise severely. Self-occlusion occurs even when there is only one hand in an image, which is quite frequent when various poses are performed. If the camera's view is egocentric, which means from first person's view, related to the AR/VR applications, say, when one is wearing a camera on his/her head, we notice that the fingertips are self-occluded by the palm since the palm will be the nearest part with respect to the camera. As we have mentioned, we must solve this problem first if we want to integrate this system into VR/AR/MR devices.

Since our target is to estimate a hand pose interaction with an object, there is another occlusion problem caused by external objects. This case makes the information less sufficient since hands are blocked directly, and sometimes we even have to describe the shape of the objects to calculate how large the area of occlusion is. As mentioned in [1], this kind of occlusion is almost inevitable when one's hand manipulates objects or holds an object tightly in his/her hand with natural poses because the object is likely to be located in the middle of the hand. Among all the poses, holding an object with one's hand being open is a rare case in daily life.

The occlusion problem caused by the existence of external objects deteriorates even more from the first person's view, and Fig. 1 illustrates some examples of hands in interaction with external objects. We also notice that, while interacting with objects, the depth pixels of hands and of objects will be connected, which is one of the reasons normal hand pose estimation systems cannot work perfectly under this condition. The research in [2] pointed out that straightforward combination of methods for object approach and for hand approach does not lead to satisfactory solutions. Some articles try to simplify hand pose estimation with object interaction to normal hand pose estimation. For instance, in [3], the authors segment the image first to obtain the pixels of the object, remove those parts from the image, and leave only hand pixels for the following hand pose estimation process. Nevertheless, this method can only work for cases where occlusion is not severe, or almost all the depth values will be subtracted completely with no information left.

Although hand pose estimation with object interaction faces many challenges, the conditions apparently are inevitable for real-world applications. In pursuit of good quality hand pose estimation, it is necessary to overcome the obstacles that prevent us from achieving the goal. In this paper, we will provide a novel architecture for hand pose estimation that aims to solve the estimation problem under hand-object interaction conditions, which are especially challenging since the environment becomes more complex. Our target is to integrate this technique with VR devices like Head-Mounting Devices (HMD) and reconstruct a hand in the VR world, which will provide connection between the real world and VR world. Since the application we have mentioned will include many hand-object interactions, it turns out that the method of hand pose estimation we design can improve the rotation performance under this condition, and such promising results will be illustrated towards the last part of the paper.

2. Related work

In recent years, the accomplishments of hand pose estimation have been numerous, given that there has been considerable research dedicated to this field. In this section, we will introduce the research relevant to this paper. There are three parts, including object detection, normal hand pose estimation, and hand pose estimation with object interaction. We will describe each of them in the following sections.

2.1. Object detection

Object detection is the most important and the most popular field in computer vision, and there have been many works proposing a variety of architectures to improve performance. In this section, we explain the object detection approaches based on convolutional neural networks (CNN), which is employed as a component in this paper. Convolutional neural network, a network presented by Krizhevsky et al. [1], dramatically increases the accuracy of object classification, and its efficiency is enhanced considerably by the advancement in GPU technology, which is recognized as the key to CNN. Afterwards, with diverse designs of network architecture, CNN also has been proven to increase performance dramatically; thus, it has been employed in a variety of computer vision tasks was the pioneer in exploiting CNN [5] for object detection, using selective search [6] as the region proposal method. Some researchers, however, soon noticed that some redundant computation could be saved, leading to the proposals of [7] and [8]. The two architectures extract regions of interest (ROI) on the feature map, of which the latter is famous for the ROI pooling layer. In [9], the authors replaced selective search, which could not be implemented on a GPU, with another GPU accelerated architecture, which led to the design of region proposal network (RPN). Such architecture, namely, faster R-CNN, attains great accuracy and efficiency, and it has been employed in various object detection tasks. Nonetheless, some later works, such as [10] and [11], present excellent architectures for solving a few disadvantages of faster R-CNN. Recently, [12] employed a structure based on faster R-CNN, proposing a fully-convolutional network approach that generates position-sensitive score maps and exploits the spatial relationship

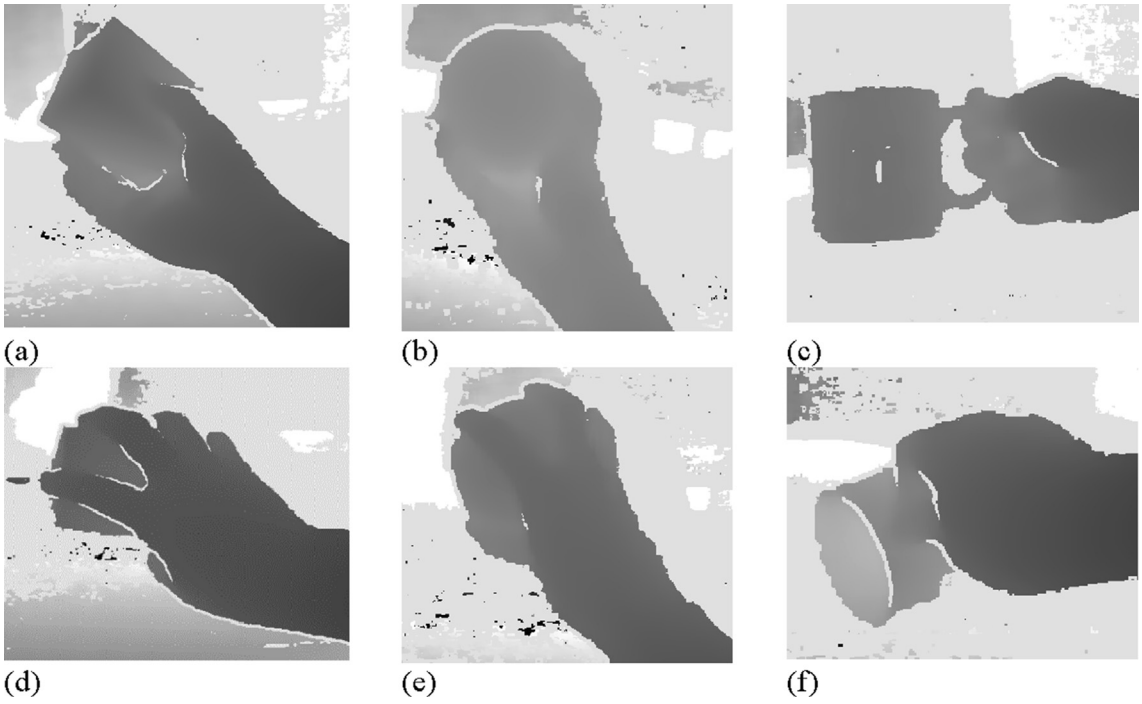

Fig. 1. Some examples of hand-object interaction images. In these depth images, we can see that either the hand or the object tends to be occluded by the other, and the pixels of both may connect together at some points. The object types in the figure are defined as (a) cube, (b) ball, and (c) cup.

on the feature map to obtain the desired ROI. This modification greatly enhances the efficiency of object detection as compared with that of faster R-CNN and shows promising performance, which motivates us to apply this architecture in our object detection part.

2.2. Hand pose estimation

Hand pose estimation is one specific regression problem in computer vision, and its development forms a long history. There have been numerous considerably different solutions to date, and there is no method that seems to dominate this field. The conventional optimization approach based on a good algorithm and a data-driven approach like deep learning alternate in showing performance improvement every year.

For the sake of clear explanation, we share some articles in this field that take advantage of CNN first. In [13], the authors created heat maps to perform pose recovery with inverse kinematics. In the process of recovering the depth image, a CNN is exploited for dense feature extraction, and the obtained heat map is a prototype of hand pose estimation. In [14], the feedback loop architecture of a CNN was proposed to estimate the 3D pose and an outstanding result with excellent efficiency was shown. It is an entirely data-driven approach, whose core is to iteratively correct mistakes. In [15], an architecture was introduced that is composed of multiple CNNs used to refine every joint by the cropped region of multi-scales and to iteratively approach the best prediction. In [16], a multi-stream CNN was proposed to cope with the multi-view problem with depth image and fuse the heat map from different views to estimate the positions of the joints. Also, *Sinha et al.* [17] proposed a two-stage CNN where one CNN in the first stage is expected to globally estimate the primary hand pose. In [18], a whole architecture was built with CNN where hand detection, as well as hand pose estimation, share the convolutional layers. That work next divides the pose into small parts, of which each part has one local hand pose regressor. To show its performance, it is tested through experiments on a public dataset and another synthetic dataset. *Ge et al.*[34] designed real-time 3D hand pose estimation from single depth images using 3D Convolutional Neural Networks (CNNs). Their method achieved 10 mm precision over a public dataset using Mean Euclidean Distance Error as the evaluation metric. In [20], a spherical part model (SPM) was employed to enhance the effect of the training model, and the approach was verified to be efficient and to show great performance. In [36], it demonstrates a steady growth of the research in real-time 3D hand pose estimation. As many researchers have confirmed the power of the CNN for this regression problem, we also turn to a CNN as our basic architecture in this paper.

Besides CNN, physical constraints of a hand are taken into consideration in many works since a hand is distinctly partitioned and the relationship of joints in a part is strong. Some researchers have combined the hierarchy property and the model-based approach (e.g. Predict the palm first, then predict fingers) to achieve better performance. *Sun et al.* [21] proposed a cascaded scheme to train multiple regressors for palm and finger, and the performance is excellent in their experimental results. *Tang et al.* [22] presented the Latent Regression Forest that predicts starting from the root (i.e. the palm joint) to the leaves (i.e. the fingertips) and optimizes each result in every step. The same group introduced a hierarchical sampling optimization approach in [23], which iteratively optimizes the hand configurations in each layer to obtain the final pose. Some works also combine CNN with a hierarchical strategy, such as [17,19], which were shown later to work very well. Wan *et al.* [33] introduced the method of pose parameterization to estimate an offset vector between depth points and hand joints, which makes the estimate translation-invariant and also generalizes to different combinations of finger poses.

2.3. Hand pose estimation with object interaction

Hand Pose Estimation involving object interaction is a relatively rare topic in computer vision, and someof the existing literature,

such as [24], whose resulting system is designed for robots, just pays more attention to object poses. Only a few works so far have focused on hand pose estimation while manipulating objects, which is significant for future applications. In [25], an RGB-image-based nearest neighbor search method was introduced for hand-tracking. In [1], the occlusion problem was introduced in hand-object interaction and a multidimensional optimization approach was proposed for hand-object pose estimation (HOPE) based on particle-swamp-optimization (PSO). That work claims to be the first to demonstrate that hand-object interaction can be exploited as a context to facilitate hand pose estimation. It also conducts experiments on synthetic data to discover the difference between HOPE and pose estimation of hands in isolation (PEHI), which refers to normal hand pose estimation methods. In [26], three kinds of trackers for two hand articulated tracking and object tracking were analyzed, and the proposed ensemble of Collaborative Trackers (ECT) for RGB-D images was shown to be able to get high accuracy. Pham et al. [27] observed hand-objection interactions and showed force sensing from vision (FSV), which helps judge the cases of interaction. In [3,28], both methods scan hands and objects in interaction into point clouds to reconstruct their 3D shapes, with the advantage of estimation of the model of unknown objects. In [29], a method was proposed for hand tracking in various kinds of interactions and detecting the salient points on fingers. Sridhar et al. [2] proposed a real-time approach that tracks hand and object at the same time from RGB-D input source, and they designed a Gaussian Mixture Model representation for a depth map of hands and objects. Their method turns out to be efficient and shows improved performance. As for [29], only fingertip points instead of the whole hand pose were predicted, and the data used in that work contains only simple interaction. [39,39] propose their solutions based on hand-object interaction which are restricted to daily hand actions involving objects. However, there are still missing various kind of 3D hand poses. [37] provides generative methods for hand-object interaction activity, only focusing on virtual objects.

To summarize the survey above, only a few employ RGB-D input images and a majority of them take pure depth image as their input sources.

3. Hand pose estimation in hand-object interaction

In this section, we will describe in details how we design this approach for the hand pose estimation in Hand-Object Interaction system.

As in [2] and [3], a lot of procedures have been done before the pose of hands and the objects are estimated. In order to explain our development more clearly, we divide it into two stages respectively with two levels of difficulties, namely, 'without object-interaction' and 'with object-interaction'. In the following sequel, we will elaborate the scheme for the former case in 3.1, which is the fundamental to this research work, and then exploit it as the basis for the latter case to be revealed in 3.2, which matches the settings closer to real-world application, so as to develop a more practical system.

3.1. Hand pose estimation using SDNet

Hand pose estimation is to predict the correct position of each hand joint on the image. In this section, we focus on normal hand pose estimation, which is the problem setting the same as that for pose estimation of hands in isolation (PEHI) in [1], since it is the fundamental work to which most researches previous articles are dedicated. The architecture we hereby propose so called skeleton-difference network (SDNet), which contains a novel loss function, namely, skeleton-difference loss function, for training in order to consider the physical constraints as well as properties of a hand. We elaborate each step to finish this task. Given a raw image from a depth image sensor, we have to preprocess this data to be the input of the next stage, hand detection, where we detect the bounding box of a hand. Then, we crop the hand according the bounding box result of hand detection, and put it as the input to hand pose estimation stage. Through our proposed SDNet, we can successfully predict the positions of hand joints. We employ deep learning approaches, CNN, for both hand detection and hand pose estimation. We will describe in details the architecture in the following sub-sections.

3.1.1. Data preprocessing

For training the CNN net for hand pose estimation, we first crop the hand region. We set a bounding box from the ground truth label. The labels contain 16 joints on each hand defined by us, which are illustrated in Fig. 2. We use the center of palm joint P and the most inner joint of middle finger M1 as the hand center, crop the bounding box on the image with the method as in [20]. The edge length of the bounding box is 25 mm in real-world space, and adjust it to a bit bigger if any of other joints exceed this region. After cropping, we can use the RoIs to training our network. The 3D bounding box is also able to crop out the depth value so that the image can be well normalized.

3.1.2. Hand detection

Before doing hand pose estimation, cropping a hand out is significant since it determines the quality of the input image for the next stage. The architecture we employed is based on RFCN [12], which adds a layer called position-sensitive RoI-pooling layer to retrieve the position information of region of interest (RoI), and adopts a fully convolutional scheme. Our network is based on ZF-net [30] to leverage its accuracy and efficiency, but we modify it to fit the architecture of RFCN. Our detection framework is illustrated in Fig. 3. In the first five layers, we take the pre-trained model on ImageNet [4] dataset, fine-tune it, and finally train the two added convolutional layers.

This detector is able to detect four classes, including 'hand,' 'mug,' 'cube,' and 'ball,' and output bounding boxes of the detected hand and other objects. Notwithstanding, the CNN of hand pose estimation demands a fixed input size of 224 × 224 pixels since there are fully-connected layers in the network. Therefore, we choose the long edge as the new width and height of the new bounding box. Then, we crop the new image to an aspect ratio of 1:1 and resize it to 224 × 224 to be the input for the next CNN.

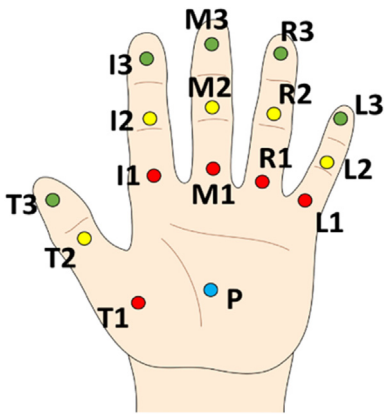

Fig. 2. The defined 16 joints in this thesis. Each finger has three joints from the palm to the fingertip.

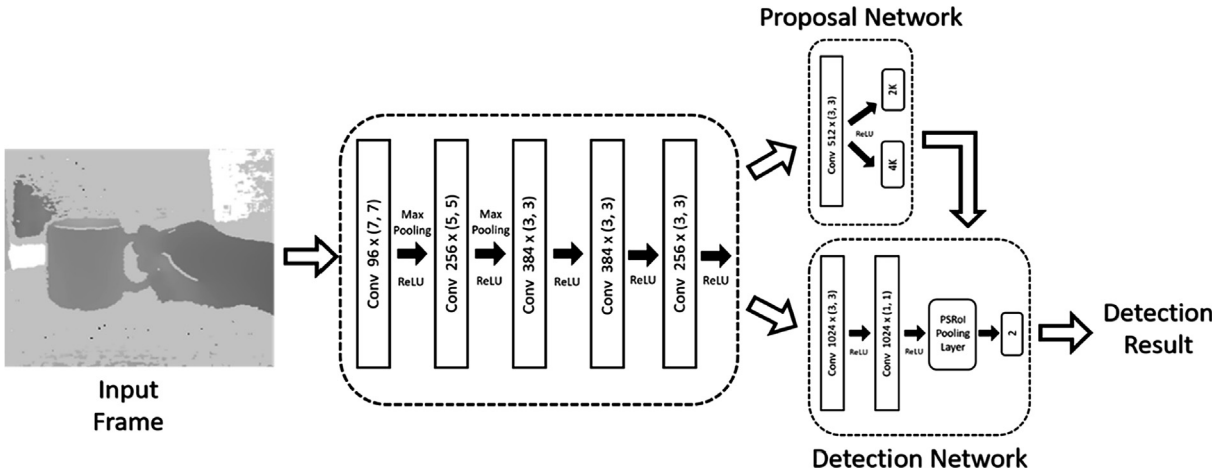

Fig. 3. Our fundamental detection architecture. The detection framework is based on RFCN [12] and the network we employ is ZF-Net [31].

3.1.3. Skeleton-difference loss function

In a normal process for hand pose estimation, only direct depth appearance is taken into consideration, but there are supposed to be some physical constraints to help predict joints. Therefore, we design a loss function, namely skeleton-difference loss function, to model the feasibility of a hand pose.

The structure of the defined skeleton also is shown in Fig. 2. In the skeleton-difference loss function, we compute the loss throughout every bone in the structure, and the loss is derived as follows:

$$\Psi_{SD} = \alpha Loss_a + \beta Loss_b \tag{1}$$

The equation includes two terms which indicate the angle loss term and the bone loss term, respectively, while α and β are the weights of two terms, respectively. The first term represents the angular loss, explaining the angle between two bones, and consequently it is derived as:

$$Loss_a = \sum_{p=1}^{5}\sum_{l=1}^{2}(\omega_1\left|\tan\left(\frac{\theta_{p,l}}{2} - \frac{\theta_{p,l}^{GT}}{2}\right)\right| + \omega_2 F(\theta_{p,l})) \tag{2}$$

where

$$\theta_{p,l} = \cos^{-1}\left(\frac{B_{p,l}\cdot B_{p,l+1}}{\|B_{p,l}\|\|B_{p,l+1}\|}\right) \tag{3}$$

$$F(\theta) = \begin{cases} 1, & \theta \geq threshold \\ 0, & else \end{cases} \tag{4}$$

Here, p is the part number out of 5 fingers on a hand, l is the angle number depending on how many angles we have on a finger, B_i is the vector between two joints (i.e. from $joint_i$ to $joint_{i+1}$, which means $joint_i$ is closer to the palm, and $joint_{i+1}$ is farther) in each part, and θ_i is an angle between the two vectors B_i and B_{i+1} while ω_k is a weight. $F(\theta)$ is a function aiming to measure the feasibility of the estimated angle, and if an angle is greater or smaller than a threshold (e.g. no smaller than 50 degrees in our experiments), it will suffer an additional penalty to suggest the natural bending of a specific hand finger. Fig. 4(a) illustrates how to calculate the angular loss. In Eq. (2), a predicted angle will between 0 and 90 degrees after divided by 2, and the difference of the two angles will between -90 to 90 degrees. Then we use tangent function to reflect bigger loss for a larger angle difference.

The other loss term of this loss function is the bone length loss, also demonstrated in Fig. 4(b), and it is as defined below:

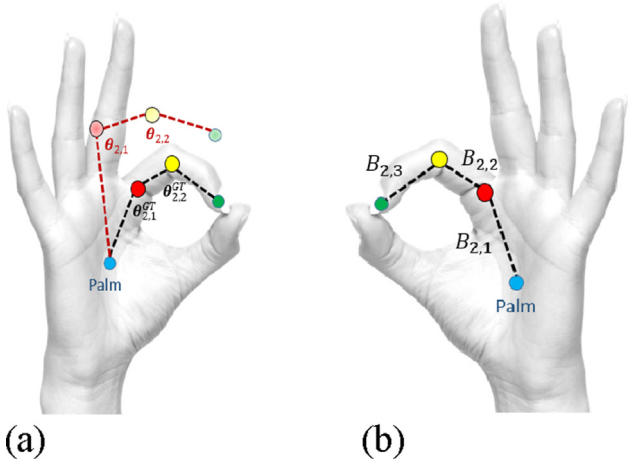

Fig. 4. The illustration of the two loss terms in skeleton-difference loss function. (a) Shows how to compute angular loss. The black angles are ground truth angles in a finger while the red ones are estimated angles, where the angular loss represents the difference. (b) Demonstrates the three bone lengths in one finger, and the bone length ratio is the ratio of the neighboring two bones' lengths. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$Loss_b = \sum_{p=1}^{P}\sum_{l=0}^{L}\|G_{p,l} - G_{p,l}^{GT}\| \tag{5}$$

where

$$G_{m,n} = \left(\frac{\|B_{m,n+1}\|}{\|B_{m,n}\| + \|B_{m,n+1}\|}\right) \tag{6}$$

Here, $B_{m,n}$ means the vector between joint n and joint $n+1$ in part m, which can be physically regarded as one bone. This loss reflects the difference of predicted bone length and ground truth bone length, but we use length ratio instead. The bone ratio is the ratio of two neighboring bone lengths, which can overcome scale problems to be scale-invariant. Fig. 4(b) illustrates the bone vector in a hand pose.

3.2. Hand pose estimation in hand-object interaction

In 3.1, our SDNet is able to estimate a hand pose when no external objects intervene. In this section, the research problem of hand pose estimation is now promoted to an upper level, which means

the estimation needs to proceed for a situation where the hand manipulates some objects. When the impact of objects is considered, the difficulty of the estimation task dramatically rises since there will be a large proportion of a hand region occluded by external objects. Normally, the methods presented in previous works as well as our proposed approach described in 3.1 will have better reconstruction of a hand pose as the information from a frame is more sufficient, which means the occlusion problem in hand-object interaction will probably lead to a disaster for those methods. On the other hand, since some sources of the input image are missing, we ought to look for another additional clues to compensate the lost information. From the observation and the experience of daily life, we assume that people interacts with a certain object with some specific and natural hand poses, which we expect to find out, and that's the relations of the so-called interaction. Therefore, we try to form the relationship between a hand and an object when the former is manipulating the latter, and combine this knowledge into the hand pose estimation task. Likewise, we rely on CNN as our basic structure and design a multi-CNNs approach to solve the task. In the following sub-sections, we will first explain the network structure of the system, then talk about how we define the contact between a hand and an object, and finally describe the proposed object-manipulation layer based on Gaussian Mixture Model (GMM) that aims to enhance the performance of our deep learning process.

3.2.1. System structure overview

The basic idea of the system structure in this section is similar to what we have described in 3.1, where we can divide the whole procedure into two major stages, detection and hand pose estimation, but there are some modifications for both two parts. For detection in this section, the primary difference is that we detect not only hands but also objects in input depth frames. The detection architecture we employ here is the same as that introduced in 3.1.2, which is also R-FCN framework from [12]. Depending on how many objects we want to detect, we have $c + 1$ classes in total for our task, which means the hand and other c of objects. The details of the detection processes can be referred to 3.1.2. Note that the 'object' we will mention in the following articles will refer to 'object type' in order to generalize our model, which means those objects with similar shapes or similar functions will be categorized into the same 'object type'.

For hand pose estimation part here, we employ a multi-CNNs hand pose estimation structure in order to take various object types with which a hand is interacting into consideration. We use deep learning way to train one CNN model for one type of objects with the object-manipulation model for each, and the detail of training can be found in 3.2.3. Besides, the original hand pose estimation model (i.e. SDNet) is also contained for the case when no object exists in the image to make sure that every frame is properly predicted. Hence, there will be $c + 1$ CNNs included in this stage, and for any input frame being processed in this system, it will choose one CNN to predict its final hand pose estimation result according to what object the hand is interacting with. The whole structure of system is as illustrated in Fig. 5, which can reveal the whole testing process. Besides the two stages mentioned above, there is also one crucial component that determines the contact status of a hand and an object when interaction is going to occur, which will later be explained in 3.2.2. Given a raw depth frame from the camera, we can obtain the predicted 3D positions of each joint on a hand, which form a hand pose, after the two stages.

3.2.2. Hand-object contact status estimation

To pick the proper hand pose estimation model of the next stage, we ought to find out what is the object hand is manipulating or is about to manipulate. To estimate whether a hand and an object start to interact, we use a function to achieve the goal. The definition, namely, interacting score, of the function is derived as follows:

$$contact(\boldsymbol{d}_i) = \begin{cases} \text{True}, & if\,\Omega(\boldsymbol{r}_{h,i}, \boldsymbol{r}_{o,i})\ \ is\ \ true \\ \text{False}, & \text{else} \end{cases} \tag{7}$$

where $\boldsymbol{d}_i$ is the depth frame, $\boldsymbol{r}_{h,i}$ and $\boldsymbol{r}_{o,i}$ denote the cropped hand and object region obtained from detector. $\Omega(\cdot,\cdot)$ is the overlapping estimation function. These functions are defined as below:

$$\Omega(\boldsymbol{r}_1, \boldsymbol{r}_2) = \begin{cases} \text{True}, & \Theta(\boldsymbol{r}_1, \boldsymbol{r}_2) > 0 \\ \text{False}, & \text{else} \end{cases} \tag{8}$$

where $\Theta(\cdot,\cdot)$ denotes the intersection area of regions $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, and when the two regions overlap, the overlapping estimation function will be true as in Fig. 6.

3.2.3. Object-manipulation loss function

Object-manipulation loss function is a loss function for increasing hand pose estimation performance. We need to find a model that is capable of memorizing all the possible hand poses that are in interacting status. Thus, we use Gaussian Mixture Model (GMM) to model the hand poses in interaction.

First, we divide a hand into five parts, as shown in Fig. 7. Then, we calculate the relations between each finger and the object contact center, which is the point manually labeled on the surface of an object. The predicted joint will recognize this reference point as an object, and we can model the spatial relationship between hand and object. Fig. 8 shows different object contact centers of each object.

where $\Theta(\cdot,\cdot)$ denotes the intersection area of regions $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, and when the two regions overlap, the overlapping estimation function will be true as in Fig. 6.

The function works when the bounding boxes of a hand and an object intersect each other; otherwise it will return 'no object contact'. The contact status estimation is simple and straightforward. Though the function cannot work perfectly to judge the timing of contact, this is just an estimation, and there are indeed some cases when the estimation is wrong. However, these cases do not bother us too much because of two reasons. First, we train the interacting model with the frames including the cases whenever a hand is close enough to an object. Second, in the next stage, all the hand pose estimation models are able to provide a reasonable prediction basically, and if a frame whose contact status belongs to true but it is estimated as false, it will choose the other models that offer a bit less accurate but still acceptable result. Overall, it tolerates some errors, but moderate accuracy can still be expected.

We then transform the spatial relations into our object-manipulation representation, as in Fig. 9. Each part has one representation, and the representation $T_{\boldsymbol{J},p}$ for part p from a joint set $\boldsymbol{J}$ is defined as below:

$$T_{\boldsymbol{J},p} = (\theta_{p,1}, \theta_{p,2}, \theta_{p,3}) \tag{9}$$

where θ_k is defined as the angle between two vectors. As the hand structure shown in Fig. 9., the first one is the vector from the object contact center to palm center, while the second one is the vector from the object contact center to the joint on the part p. There are three joints in each part, so the three angles are combined as a new representation, which is able to reflect a natural pose that may exist when a finger is interacting with an object, and our job is to collect all the rational poses with GMM and to find their distribution. In [20], the works have confirmed that GMM can be a reliable data-driven method to model the probability of samples among a complex distribution. Following the idea of these papers, we use the representations from training data as training

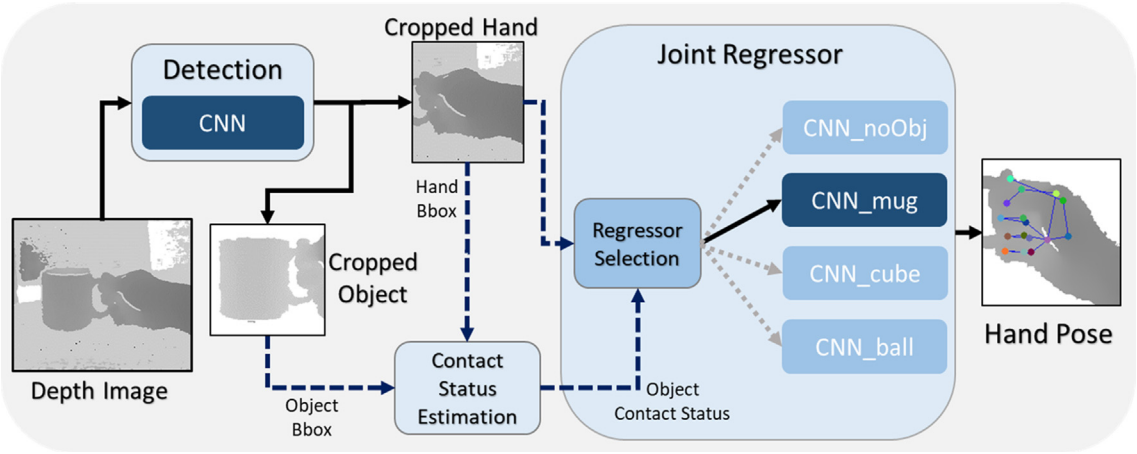

Fig. 5. The whole architecture of our hand pose estimation system, which consists of three main components: detection, contact status estimation and joint regressor stage. In joint regressor stage, the estimation will depend on what kind of object type the hand is interacting, whose status is determined by the contact status estimation. The mug, cube and ball here are three types of objects.

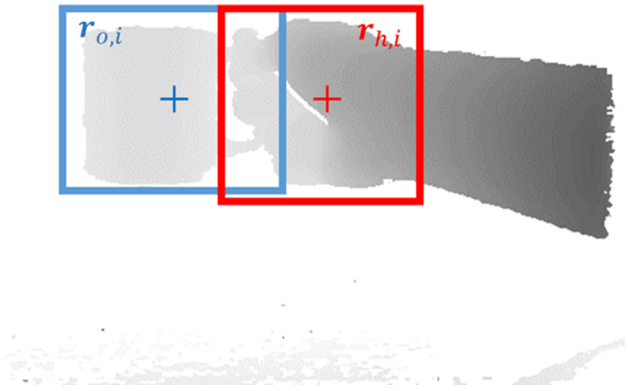

Fig. 6. The overlap of the two bounding boxes of the object and the hand, respectively. The bounding boxes are from the previous stage of detection.

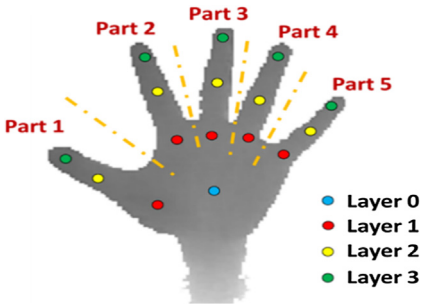

Fig. 7. We divide the hand into 5 parts, each of which has three joints. One part corresponds to one finger. Furthermore, the hand palm is also defined to be in the layer 0, while layer 1 to layer 3 is defined as the joints of each finger from the palm to the fingertips. That is, the five fingertips belong to layer 3.

samples for GMM by EM algorithm, and the expected distribution will be accessed. The number of Gaussian distribution is 3. We obtain five GMMs from five parts, and each GMM is able to predict a score of assembling the distribution given a new sample. That is, the score can offer the rationality of a new hand pose. Therefore, we take this idea into the training process of CNN, and becomes a new loss function, namely object-manipulation loss function, and the loss $\Psi_{OM}(\cdot)$ is derived as following:

$$\Psi_{OM}(\boldsymbol{J}) = \sum_{p=1}^{5} \frac{1}{S(T_{\boldsymbol{J},p}, GMM_p)} \tag{10}$$

where $\boldsymbol{J}$ denotes the predicted joint set in the training process, and $S(\cdot,\cdot)$ offers the evaluated score of joint set $\boldsymbol{J}$ given by GMM_p, which is the GMM for part p. We use inverse to transform the score into loss and sum the loss of every part. Then our training architecture will include the joint loss function, skeleton-difference loss function and object-manipulation loss function, and we can construct the CNN training process as illustrated in Fig. 10, which considers Euclidean loss, physical constraints of a hand pose itself and the rationality of a manipulating hand pose. Eventually, the total loss of the CNN training procedure will be consequently derived as following:

$$\Psi_{Total} = \Psi_D + \omega_{SD}\Psi_{SD} + \omega_{OM}\Psi_{OM} \tag{11}$$

where ω_{SD} and ω_{OM} are the weights of the two individual loss functions.

In the formula above, if no object involves, the model will be trained with the first two terms, which are Euclidean loss function and skeleton-difference loss function, respectively. On the other hand, if hand-object interaction is considered, the third term, object-manipulation loss function, will be added. For one object type, we train a CNN, which serves all objects of similar kinds instead of a specialized object. We believe that a similar type of objects will be held by the similar hand poses, which can be common in our daily life. People always use the similar hand poses to take, to hold and to manipulate stuff of a certain shape. What's more, when some occlusion appears or some parts are invisible owing to an object or the hand itself, the skeleton-difference loss function and the object-manipulation loss function are able to predict a reasonable and natural result from our assumption of natural hand poses. Using the models trained from the whole architecture, we can predict a hand pose directly from an input depth frame.

In the training process of the dataset involving dataset, we use the model that is pre-trained on public dataset without hand-object interaction, and then fine-tune this model on the frames including hand-object interaction. By doing so, the knowledge of the hand pose can be maintained and transferred to the new data.

4. Experimental results

In this section, we will explain the experiments of our hand pose estimation system, including the conditions of a single hand

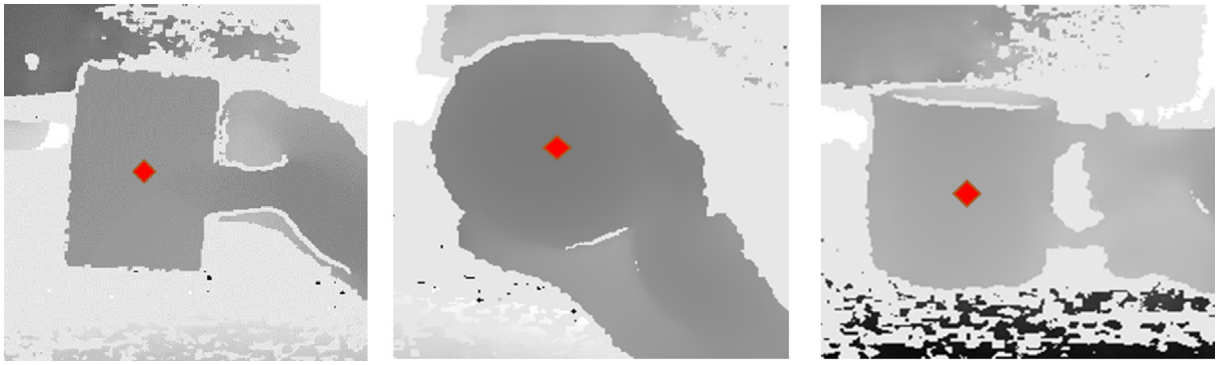

Fig. 8. The object contact center of different object types, which are (a) cube, (b) ball and (c) cup, respectively. An object contact center normally will be labeled at the center of an object's main body.

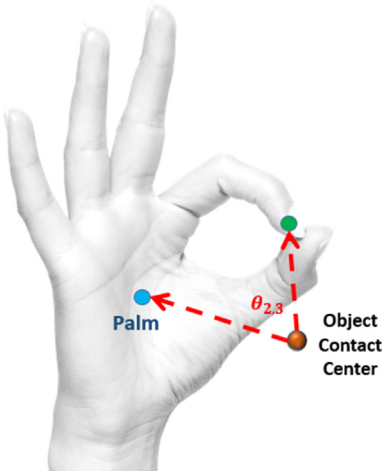

Fig. 9. The illustration of transformation from one finger joint into object-manipulation representation. One angle of this representation is the angle between two vectors from the object contact center.

pose estimation or hand pose estimation in interaction with objects.

4.1. Environmental settings

The experiments of our whole system are conducted on a PC. Table 1 shows some details of the hardware specification of this computer. As for implementation of convolutional neural network (CNN), we employ Caffe [31] library as our deep-learning toolbox. For both detection and hand pose estimation, we implement the system with the python version of Caffe.

Table 1
The Specification of our machine for experiments.

Equipment	Specification
Central Processing Unit (CPU)	Intel Core i5-6500 @3.20 GHz
Random Access Memory (RAM)	24.0 GB
Graphic Processing Unit (GPU)	NVIDIA GeForce GTX 980
Operating System (OS)	Microsoft Win 10
System Bit Type	64 bit

4.2. Datasets

We use two public datasets, ICVL Hand Posture Dataset [22] and NYU Hand Pose Dataset [13]. We also make a dataset ourselves, namely NTU Hand-Object Interaction Dataset, to evaluate the performance of our hand pose estimation in object interaction. We will talk about the contents and the properties of the three individual datasets in the following sections.

4.3. ICVL hand posture dataset

ICVL Hand Posture Dataset [22] is a public dataset released by Imperial Computer Vision & Learning Lab (ICVL). There are about 22,000 images in the training set and 1596 testing frames captured by Intel Creative depth sensor. In the dataset, a hand in each individual frame is annotated with 3D positions of the hand joints in

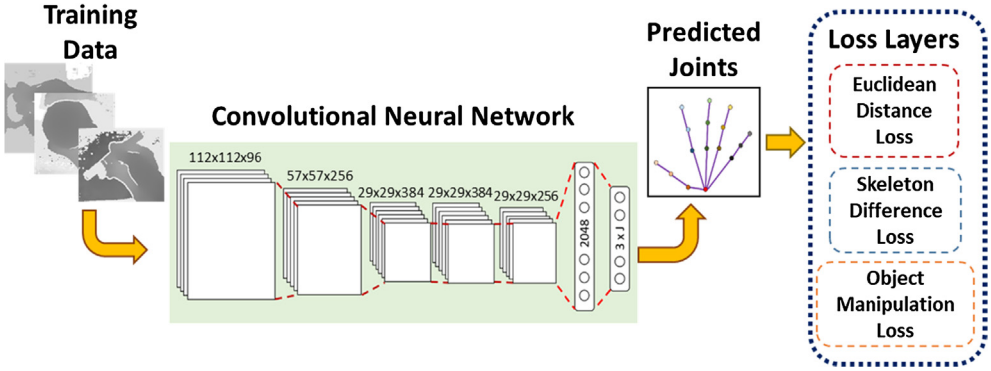

Fig. 10. The flowchart of the training process. Given the training data, we can retrieve the predicted joint positions from the CNN, and we update the parameters of the CNN via the back-propagation from the loss layers. There are three loss functions, including the conventional Euclidean distance loss function, the skeleton-difference loss function and the object-manipulation loss function.

the image space and its depth (i.e. (u, v, d)). There are 16 joints annotated on each frame, as shown in Fig. 11 shows some examples in this dataset.

4.4. NYU hand pose dataset

NYU Hand Pose Dataset is a public dataset released by NYU that contains 8252 test-set and 72,757 training-set frames captured by Microsoft Kinect [32]. The frames were collected as RGB-D from third person's view of three viewpoints and are labeled with ground-truth annotations of hand-pose information. Some examples are shown in Fig. 12.

4.5. First-Person Hand Action (FPHA) dataset

First-Person Hand Action (FPHA) dataset [38] is one publicly available dataset at the time for 3D hand-object interaction recognition that contains labels for 3D hand pose, 6D object pose and action categories. The dataset contains 1175 videos belonging to 45 different activity categories performed by 6 actors in 3 different scenarios - kitchen, office and social. A total of 105,459 RGB-D frames are annotated with accurate hand poses and action categories. A subset of the dataset contains annotations for objects' 6-dimensional poses along with corresponding mesh models for 4 objects involving 10 different action categories. This subset of the dataset is denoted as FPHA hand-object dataset.

4.6. NTU hand-object interaction dataset

This dataset, which is captured by Intel RealSense [35] depth sensor, is a new dataset created by our own for evaluating our hand-object interaction model. The image source includes depth frames and RGB frames that are captured from the egocentric view (i.e. first-person's view) to simulate the camera on a Head-Mounted Display (HMD) as in the setting of Fig. 13, and accordingly can be utilized in further RGB-D hand pose estimation researches or activity recognition researches, and we use only depth source in our following experiments in this thesis. It currently has 7 video sequences in conditions of a bare hand or a hand that is manipulating different objects, including the mug, the ball and the cube. The frame number of this dataset is listed in Table 2.

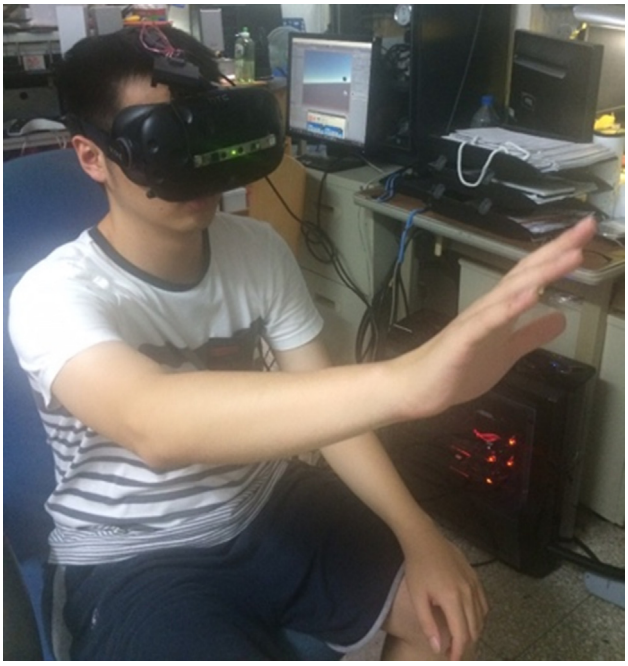

Fig. 13. We integrate a depth sensor (Intel RealSense F200) with a VR device (HTC Vive). We want to construct a hand in virtual world through our hand pose estimation system.

Table 2
The frame number of each class in NTU hand-object dataset.

Object class	Training frames	Testing frames
Bare hand	2214	246
Mug	1289	143
Cube	1858	206
Ball	890	98

For the frames in this dataset, we have manually annotated the bounding box of objects, the 3D position of object contact centers which are used for training our object-manipulation model, and the 3D position of 16 hand joints as defined in Fig. 7. To annotate

Fig. 11. Some examples in NYU Hand Pose Dataset [32]. This dataset contains many images that have serious broken pieces.

Fig. 12. Four frames in ICVL Hand Posture Dataset [13]. This dataset contains many challenging hand poses. These images are normalized to be well presented.

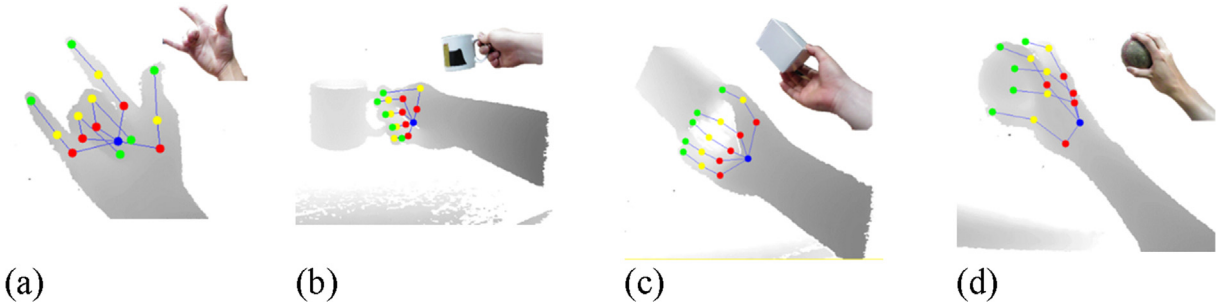

Fig. 14. Some examples of annotated frames in NTU hand-object dataset. The four classes are (a) bare hand, (b) mug, (c) cube, and (d) ball.

Table 3
The result of SDNet and comparisons with previous works on ICVL Hand Posture Dataset.

Method	Average Euclidean Distance Error (mm)
Cascaded [21]	9.800
SPM [20]	8.649
LRF [22]	13.433
Ours (SDNet)	8.452

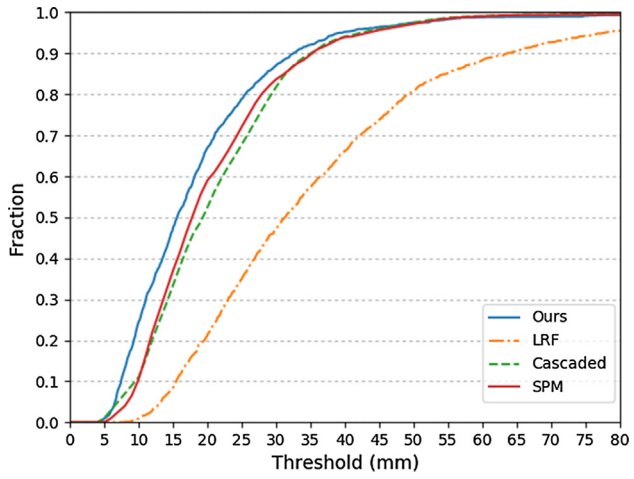

Fig 15. The successful pose estimation fraction of a full hand under the threshold compared with other papers [20–22] on ICVL Hand Posture Dataset.

Table 4
The comparison of setting different weights to skeleton-difference layer. The baseline uses the CNN of [30] without additional knowledge. The weight of Euclidean loss function is set to be 1 and is adjusted to different weights of our skeleton-difference loss function (SDNet).

Network settings (weight, ω_{SD})	Mean Euclidean Distance Error (mm)
Baseline	12.6990
SDNet (0.125)	8.8209
SDNet (0.25)	8.7291
SDNet (0.5)	**8.4520**
SDNet (1)	8.5104

Fig. 17. The samples of detection results from our hand detector on NYU Hand Pose Dataset.

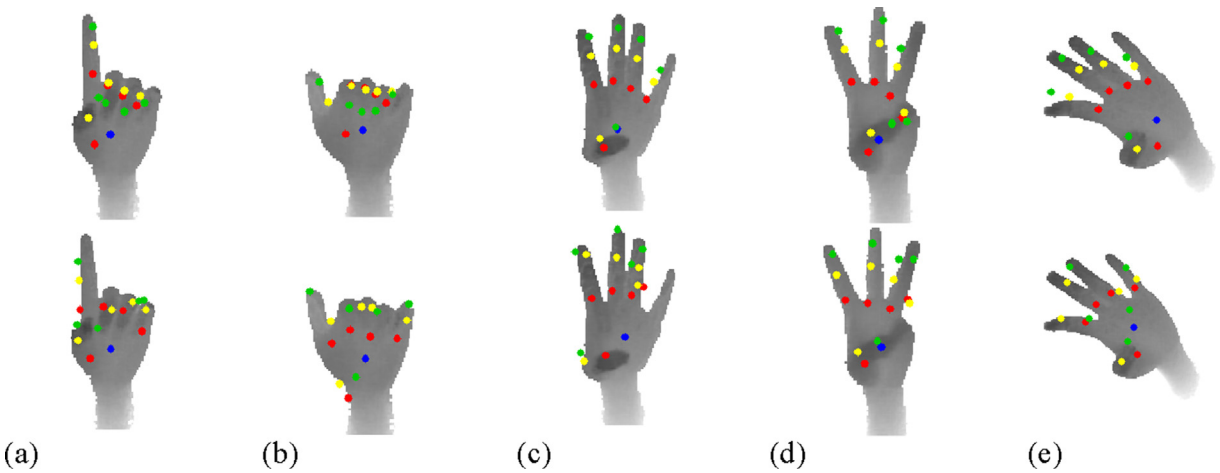

Fig. 16. Some examples of our predicted results (upper row) and of [22] (lower row) on ICVL Hand Posture Dataset. (a)–(c) Our estimated hand poses have better performance than other works even in some difficult cases, such as severe occlusions, and the poses maintain good shape. (d) and (e) Even in some challenging poses, our prediction result may not be absolutely accurate, but our method still makes a hand pose complete and natural.

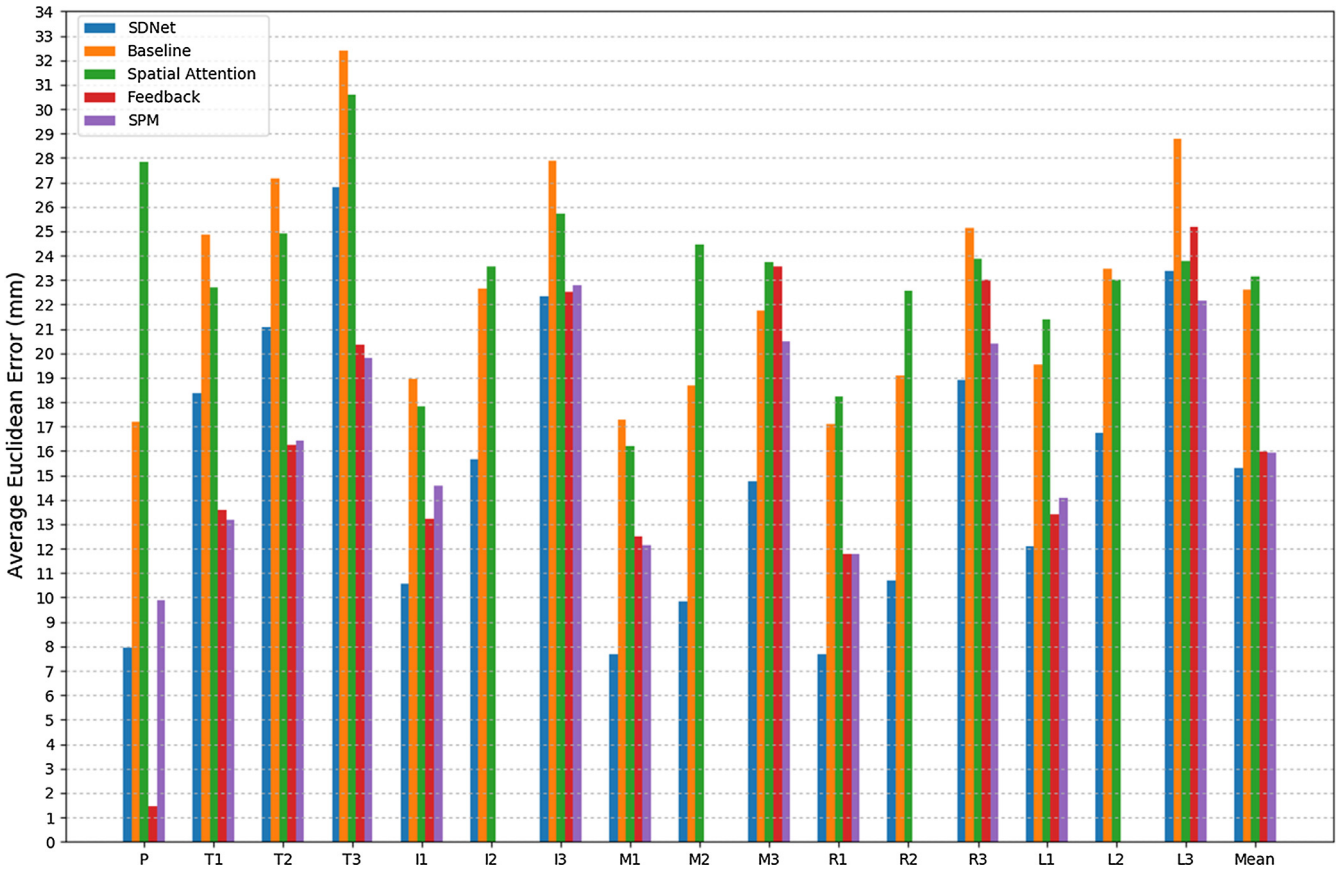

Fig. 18. The average Euclidean Error of each joint. We compare our results (SDNet) with Spatial Attention DeepNet [19], Feedback Loop [14], SPM [20], and our baseline that uses simple CNN architecture on NYU Hand Pose Dataset.

Table 5
Comparisons with previous works on NYU Hand Pose Dataset.

Method	Average Euclidean Distance Error (mm)
Feedback Loop [14]	15.972
SPM [20]	15.909
Spatial Attention DeepNet [19]	23.149
Ours (SDNet)	15.286

Fig. 19. The examples of prediction results on NYU Hand Pose Dataset.

the positions in 3D space from 2D depth image frames, we first transform a depth frame into visualized point cloud frame to label the z-axis value of a joint.

The frames in some sequence of this dataset contain a lot of hand-object interaction, which means external occlusions may occur in many frames. Basically, we assume that these invisible joints will be located at some feasible positions and will form a natural hand pose. Moreover, in order to overcome the problem, we use some rules to enhance quality of annotations. First, an annotated hand pose is supposed to be thought as rational and natural by a human annotator, and we can infer the position of invisible joints from the visible joints. And also, the shape and the appearance of the manipulating objects, which are rigid body, can be helpful for the annotation as well. Secondly, our data are continuous frames, so we can annotate the joints from its previous frame. We initialize the annotations of a new frame with the previous labeled frame, and then adjust the position minor. Third, when frames have been annotated, we will calculate the bone lengths, which means the distance between two neighboring joints, and choose out the poses that contain abnormal bone length to have them re-annotated.

The image frames in this dataset are captured in some very challenging conditions of hand-object interaction such as holding it and rotating it, some of which contain many occlusions. Though this dataset may be too difficult for some researches, the purpose of this dataset is to set in the real-world conditions as much as possible. Some examples of annotated frames in this dataset are shown in Fig. 14.

4.7. Results

4.7.1.1. ICVL hand posture dataset In this dataset, we compare our experimental results with the previous state-of-the-art papers [20–22] and the baseline shown in Table 3 for evaluation of the average Euclidean distance error. Our method performs better than the other three papers in almost all the joints, which verifies that our overall prediction is more accurate.

Fig. 15 show that the comparison in the fraction under threshold evaluation, and we have a remarkable enhancement. From the

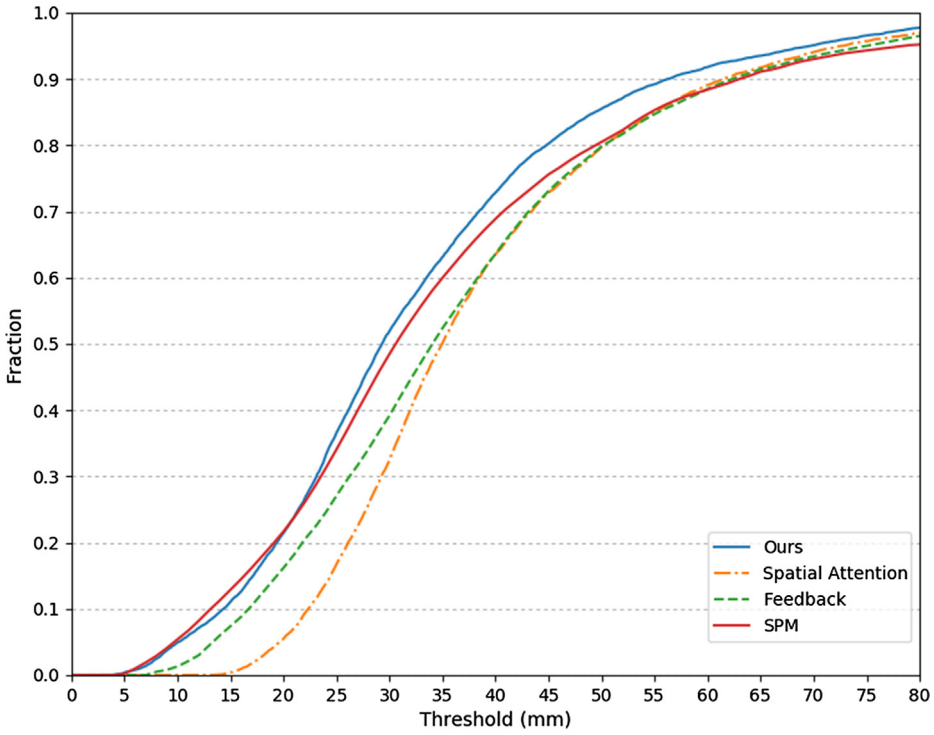

Fig. 20. The successful pose estimation fraction of a full hand under the threshold compared with other papers [14,19,20] on NYU Hand Pose Dataset.

figure provided in [21], their mean Euclidean distance error is about 9.8 mm while ours is 8.45 mm. Nevertheless, the results in Fig. 16 confirm that our predicted hand poses can be maintained in good shape since we impose the relations between joints on our model. Fig. 16 demonstrates the completeness of our estimated hand pose. We compare the predicted results with [22], and we see that their estimated pose can be highly distorted from time to time, while ours is quite natural and intact in shape.

Moreover, Table 4 demonstrates the importance of skeleton-difference loss function with those physical constraints. We set the weight of Euclidean distance error to be 1 and adjust to different weights of skeleton-difference loss function (i.e. SDNet). We notice that, as the weight of skeleton-difference loss function increases, the performance becomes better.

4.7.1.2. NYU hand pose dataset. We evaluate our method on the testing-set of NYU Hand Pose Dataset as well. Fig. 17 shows some examples of the predicted bounding box from our detector of NYU Hand Pose Dataset. Notice that we do not expect to resize or distort the bounding box since we need to forward it to the next stage (i.e. Hand pose estimation), so we try to predict the region of interest (RoI) as a square.

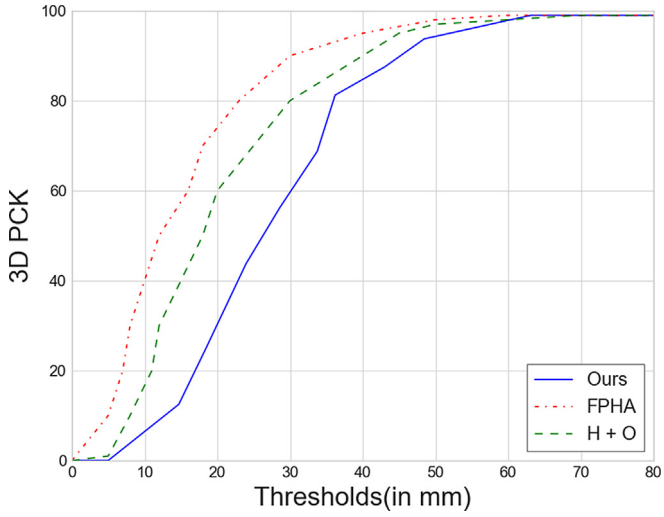

Fig. 21. Comparison of the hand pose estimation results of our work with those of *Garcia-Hernando et al.* [38] *Tekin et al.* [39] on FPHA hand-object dataset.

Table 6
The detection result of our hand and object detector on FPHA hand-object dataset.

Class	Average precision
Hand	0.916
Mug	0.889
Cube	0.931
Ball	0.798
Mean	0.8835

Table 7
The class mapping between FPHA hand-object dataset and Object Class detection for our model.

Object class (NTU)	Action category (FPHA)
Cube	close_milk
	close_juice_bottle
	close_liquid_soap
	open_milk
	open_juice_milk
	open_liquid_soap
	pour_juice_bottle
	pour_liquid_soap
Mug	pour_milk
Ball	put_salt

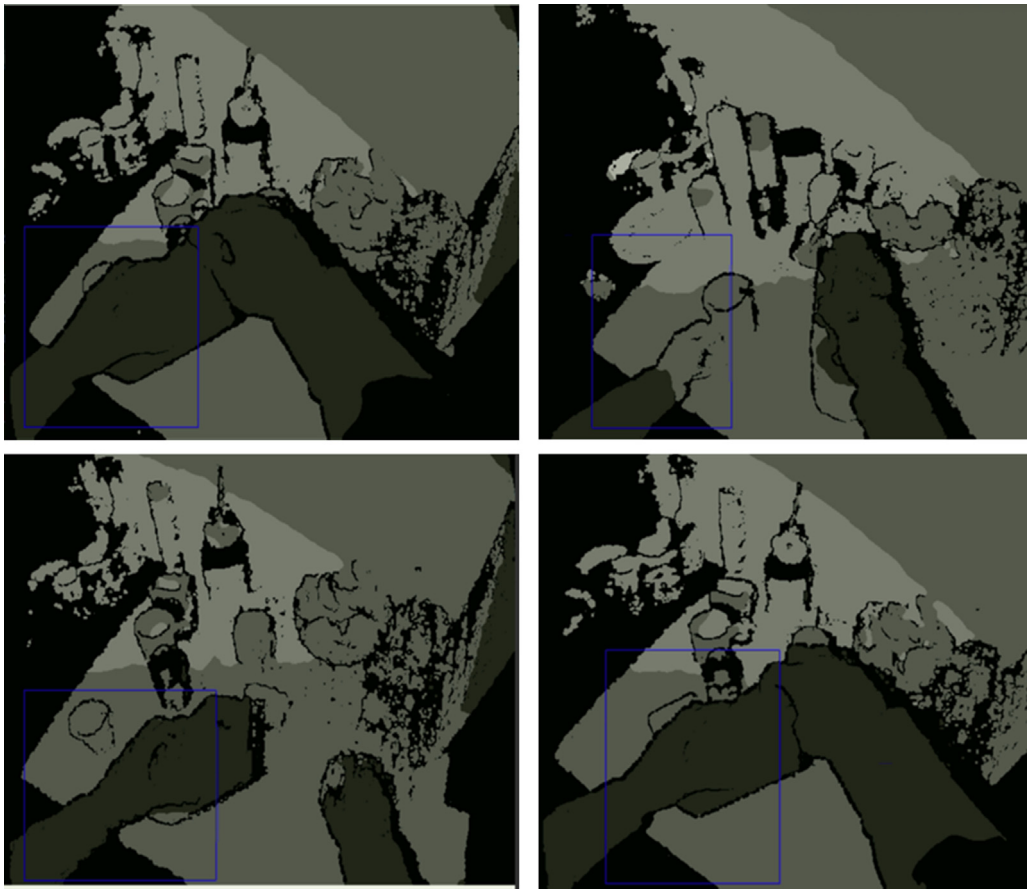

Fig. 22. Error detection happens on FPHA hand-object dataset.

Table 8
The detection result of our hand and object detector on NTU hand-object dataset.

Class	Average precision
Hand	0.9520
Mug	0.9005
Cube	0.7947
Ball	0.9445
Mean	**0.8979**

We show our experimental results on NYU Hand Pose Dataset and compare with the state-of-the-art works [14,19,20] from their released statistics. We also set the same baseline as in the previous section and make a comparison after the skeleton-difference layer is joined. The results of average Euclidean distance are shown in Fig. 18 and Table 5, and one can see that we achieve slightly better results in the Euclidean distance error over previous papers. Fig. 19 shows some examples of results.

We can turn to Fig. 20 to observe the fraction of success, and in this evaluation benchmark, Feedback Loop [14], SPM [20] and ours are quite close.

4.7.1.3. FPHA hand-object dataset. We compare the accuracy of our 3D hand pose predictions to the state-of-the-art research [39,39] on FPHA hand-object dataset in Fig. 21. Besides comparing 3D hand pose predictions, we also do experiments to evaluate the performance of this hand and object detector, Table 6. In order to create a correlation between our detectable object class set and the action category set used by the FPHA dataset, we have to map each of the 10 different action categories of the FPHA hand-object dataset to one of our three object types – 'mug', 'cube' and 'ball – nets, Table 7.

Since our work is focusing on 'right hand' detection at the first stage, some of the data from the FPHA dataset was not compatible with our model. It failed to detect the object because two hands were visible in the dataset and the model couldn't decide which was the correct hand, see Fig. 22. Since such errors decrease the accuracy of our detection, so we exclude most of these edge cases. In future work the system can be adapted to detect both hands. Our work is designed to run in real-time and be less computationally intensive than the compared research *Garcia-Hernando et al.* [38] *Tekin et al.*[39].

4.7.1.4. NTU Hand-Object dataset. This dataset is made by ourselves to evaluate the hand pose estimation when a hand is interacting with objects. We choose three object types, including 'mug', 'cube' and 'ball' in our experiments, and we design some comparisons to see the performance. Table 8 shows the performance of this hand and object detector, and we can observe that all the classes perform quite well.

We also make some bare hand images to show the performance of our SDNet on hand pose estimation. As with the previous two public datasets, we employ two evaluations in comparison with the baseline, and the results are shown in Fig. 23 and Fig. 24. This can also show the improvement of our SDNet on every joint, and the average Euclidean distance error of our SDNet model is 16.365 mm.

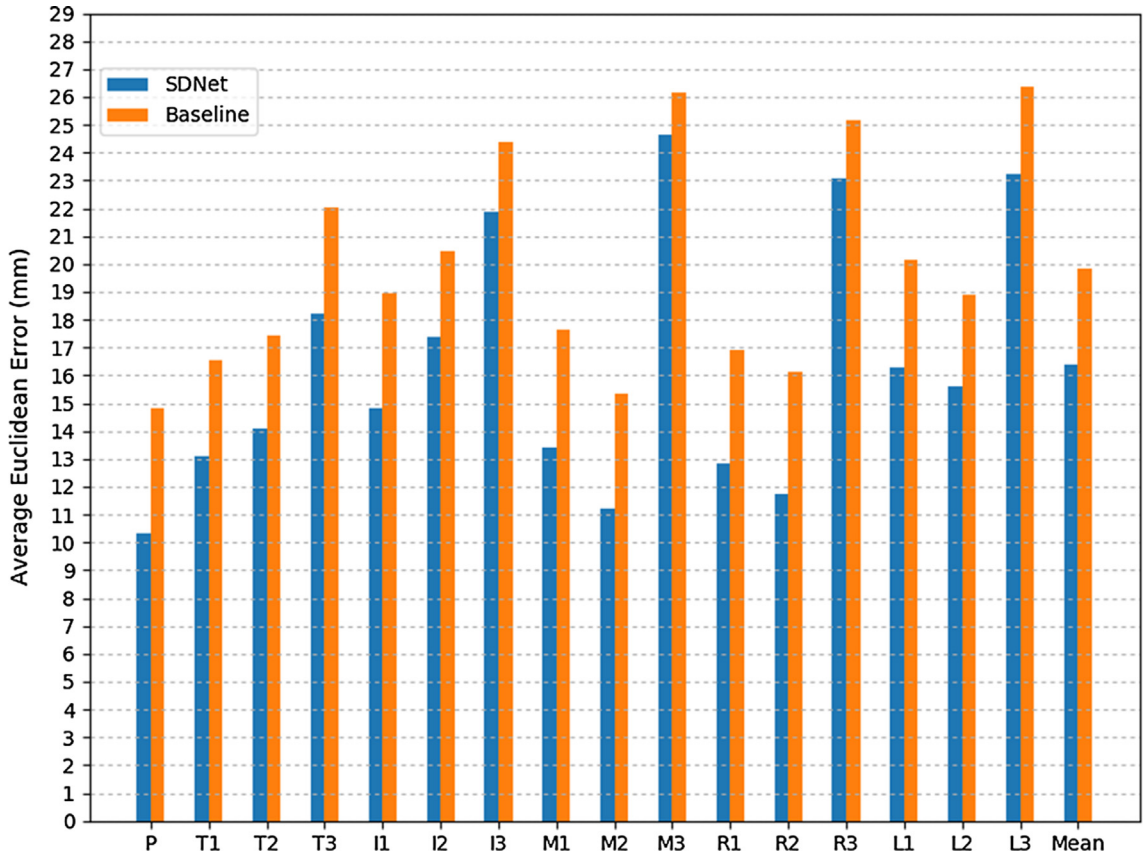

Fig. 23. The average Euclidean Error of each joint. We compare our results (SDNet) and the baseline that uses simple CNN architecture on NTU Hand-Object Dataset.

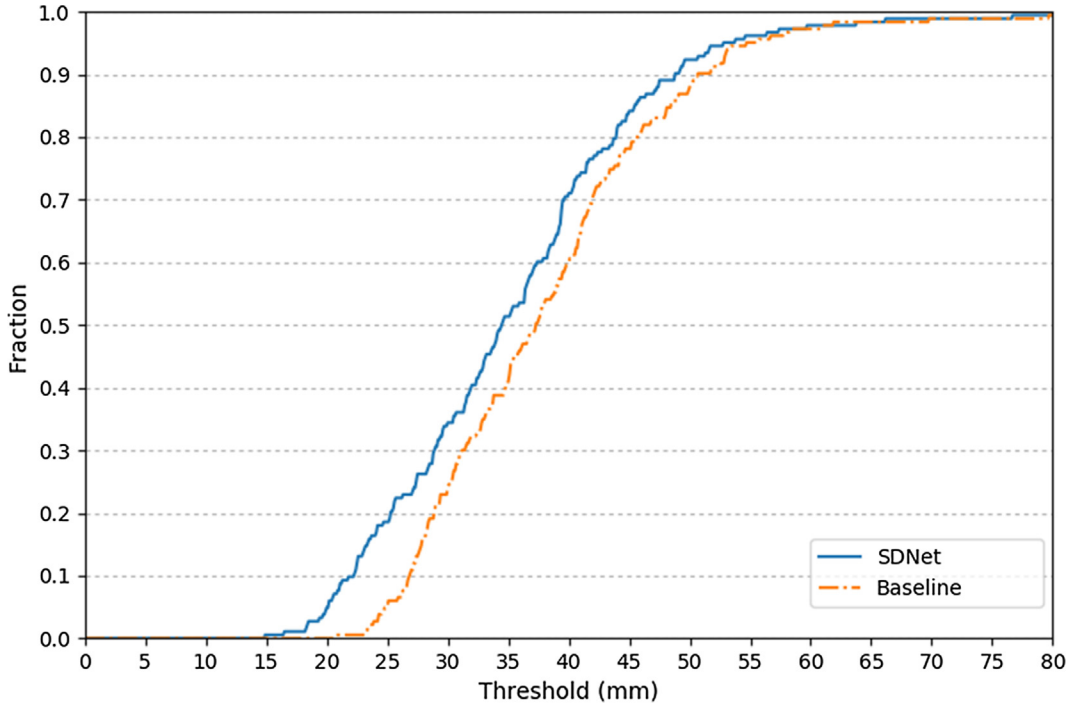

Fig. 24. The successful pose estimation fraction of a full hand under the threshold compared with the baseline on NTU Hand-Object Dataset.

To illustrate the improvement of our method, we can test a data sequence with three kinds of nets. The first one is a CNN without additional knowledge to predict a hand pose based on depth frame. The second one is SDNet, which considers the physical constraints of a hand. We can regard the previous two as baseline, and compare with the SDNet trained with object-manipulation loss

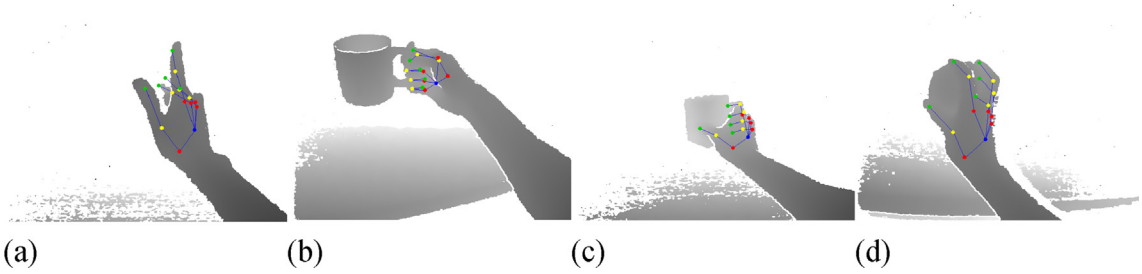

Fig. 25. Some examples of predicted results on NTU Hand-Object Dataset. The four classes are (a) bare hand, (b) mug, (c) cube, and (d) ball.

Table 9
The Euclidean distance error of models on different objects in NTU Hand-Object Dataset.

Object Class	Model		
	Baseline	SDNet	SDNet + OML
Bare Hand	19.8416 mm	16.3651 mm	–
Mug	10.5878 mm	10.5689 mm	8.3031 mm
Cube	20.4631 mm	18.2559 mm	15.2499 mm
Ball	16.1031 mm	12.8315 mm	11.0239 mm
Average	16.7489 mm	14.5054 mm	11.5256 mm

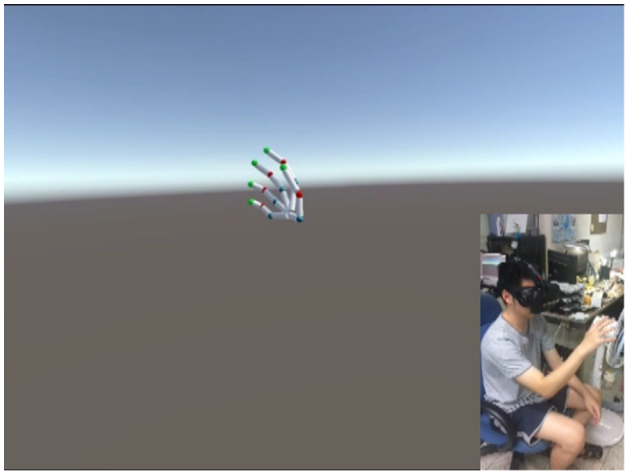

Fig. 26. Our system can be applied to HMD of VR devices.

function (OML), which takes the object knowledge into consideration. We test these models on four classes of objects including 'hand,' 'mug,' 'cube,' and 'ball,' and the results are shown in Table 9. The figure shows that, when more additional knowledge is taken into consideration, the result is more likely to perform better. Fig. 25 shows some examples of results.

Moreover, the detection stage, contact status estimation stage, and joint regression stage takes about 20 ms, 0 ms, and 7 ms, respectively, with our equipment, which shows this system can run real-time and can be applied precisely to future applications, as in Fig. 26. It shows the integration of depth sensor, Head-Mounting Devices (HMD), and our hand pose estimation technique in one system. Afterward, we can use our hand to interact with the virtual world directly through the interface and no longer need external controllers to achieve the goal of human–computer interaction.

5. Conclusion

In this paper, we propose a novel system for 3D hand pose estimation that can predict a human hand pose accurately from vision-based frames. Moreover, either a bare hand or a hand manipulating an external object can be estimated by this system. The experimental results conducted on several datasets show that our method not only performs well but also is robust.

To achieve the goal of accurate and efficient hand pose estimation in these challenging conditions, we designed a deep-learning approach to train a convolutional neural network (CNN) model. First, we developed a skeleton-difference layer that described the physical constraints, such as angles of bones and lengths of a hand. Second, we propose an object-manipulating layer that models the relationship of hand-object interaction. In implementation, we employ a Gaussian Mixture Model (GMM) to compute the distribution of the spatial relations between hand joints and objects and add this knowledge into training the CNN model. With the two layers mentioned above, our CNN model is able to consider the appearance of depth image and the physical constraints of a hand and the hand-object interaction at the same time, so we can train this model end-to-end. Moreover, with this model, we are able to predict positions of the joints very efficiently.

The proposed system includes a hand-object detector and several hand pose estimators for a bare hand and a hand in interaction with other external objects. We compare the effect of those components with other state-of-the-art papers, and see great improvements over previous works and our designed baseline. Therefore, the proposed system can be used in applications, such as Virtual Reality (VR), Augmented Reality (AR), or Mixed Reality (MR), that may highly rely on human's hands as the natural communication between a human and a computer in the future, and it can create a convenient interface to improve our future daily life.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] I. Oikonomidis, N. Kyriazis, A.A. Argyros, Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints, in: IEEE International Conference on Computer Vision (ICCV), (2011) pp. 2088–2095.
[2] S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, C. Theobalt, Real-time joint tracking of a hand manipulating an object from RGB-D input, in: European Conference on Computer Vision, (2016), pp. 294–310.
[3] P. Panteleris, N. Kyriazis, A.A. Argyros, 3D Tracking of Human Hands in Interaction with Unknown Objects, in: British Machine Vision Conference, (2015) pp. 123.1–123.12.
[4] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inform. Process. Syst. (2012) 1097–1105.
[5] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, (2014) pp. 580–587.
[6] J.R. Uijlings, K.E. Van De Sande, T. Gevers, A.W. Smeulders, Selective search for object recognition, Int. J. Comput. Vision 104 (2013) 154–171.

[7] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in: European Conference on Computer Vision, (2014), pp. 346–361.
[8] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.
[9] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, Adv. Neural Inform. Process. Syst. (2015) 91–99.
[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, et al., Ssd: Single shot multibox detector, in: European conference on computer vision, 2016, pp. 21–37.
[11] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[12] Y. Li, K. He, J. Sun, R-fcn: Object detection via region-based fully convolutional networks, in: Advances in Neural Information Processing Systems, 2016, pp. 379–387.
[13] J. Tompson, M. Stein, Y. Lecun, K. Perlin, Real-time continuous pose recovery of human hands using convolutional networks, ACM Trans. Graphics (ToG) 33 (2014) 169.
[14] M. Oberweger, P. Wohlhart, V. Lepetit, Training a feedback loop for hand pose estimation, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3316–3324.
[15] M. Oberweger, P. Wohlhart, V. Lepetit, Hands deep in deep learning for hand pose estimation, arXiv preprint arXiv:1502.06807, 2015.
[16] L. Ge, H. Liang, J. Yuan, D. Thalmann, 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, 2016, pp. 3593–3601.
[17] A. Sinha, C. Choi, K. Ramani, Deephand: Robust hand pose estimation by completing a matrix imputed with deep features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4150–4158.
[18] T.-Y. Chen, M.-Y. Wu, Y.-H. Hsieh, L.-C. Fu, Deep learning for integrated hand detection and pose estimation, in: 23rd International Conference on Pattern Recognition (ICPR), 2016, 2016,, pp. 615–620.
[19] Q. Ye, S. Yuan, T.-K. Kim, Spatial Attention Deep Net with Partial PSO for Hierarchical Hybrid Hand Pose Estimation, in: European Conference on Computer Vision, (2016) pp. 346–361.
[20] T.-Y. Chen, P.-W. Ting, M.-Y. Wu, L.-C. Fu, Learning a Deep Network with Spherical Part Model for 3D Hand Pose Estimation, presented at the IEEE International Conference on Robotics and Automation (ICRA), Singapore, Singapore, 2017.
[21] X. Sun, Y. Wei, S. Liang, X. Tang, J. Sun, Cascaded hand pose regression, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 824–832.
[22] D. Tang, H. Jin Chang, A. Tejani, T.-K. Kim, Latent regression forest: Structured estimation of 3d articulated hand posture, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3786–3793.
[23] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, J. Shotton, Opening the black box: Hierarchical sampling optimization for estimating human hand pose, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3325–3333.
[24] M. Krainin, P. Henry, X. Ren, D. Fox, Manipulator and object tracking for in-hand 3d object modeling, Int. J. Robot. Res. 30 (2011) 1311–1327.
[25] J. Romero, H. Kjellström, D. Kragic, Hands in action: real-time 3D reconstruction of hands in interaction with objects, IEEE International Conference on Robotics and Automation (ICRA) 2010 (2010) 458–463.
[26] N. Kyriazis, A. Argyros, Scalable 3d tracking of multiple interacting objects, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3430–3437.
[27] T.-H. Pham, A. Kheddar, A. Qammaz, A.A. Argyros, Towards force sensing from vision: Observing hand-object interactions to infer manipulation forces, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2810–2819.
[28] D. Tzionas, J. Gall, 3D object reconstruction from hand-object interactions, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 729–737.
[29] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, J. Gall, Capturing hands in action using discriminative salient points and physics simulation, Int. J. Comput. Vision 118 (2016) 172–193.
[30] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European conference on computer vision, 2014, pp. 818–833.
[31] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, et al., Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM international conference on Multimedia, 2014, pp. 675–678.
[32] Z. Zhang, Microsoft kinect sensor and its effect, IEEE Multimedia 19 (2012) 4–10.
[33] C. Wan, T. Probst, L. Van Gool, A. Yao, Dense 3d regression for hand pose estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5147–5156.
[34] L. Ge, H. Liang, J. Yuan, D. Thalmann, Real-time 3D hand pose estimation with 3D convolutional neural networks, IEEE Trans. Pattern Anal. Mach. Intell. 41 (4) (2018) 956–970.
[35] M. Draelos, Q. Qiu, A. Bronstein, G. Sapiro, "Intel realsense= real low cost gaze," in Image Processing (ICIP), IEEE International Conference on 2015 (2015) 2520–2524.
[36] Shanxin Yuan, Guillermo Garcia-Hernando, Björn Stenger, Ju. Gyeongsik Moon, Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, et al., Depth-based 3d hand pose estimation: From current achievements to future goals, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2636–2645.
[37] Markus Oberweger, Paul Wohlhart, Vincent Lepetit, Generalized feedback loop for joint hand-object pose estimation, IEEE Trans. Pattern Anal. Mach. Intell. (2019).
[38] Garcia-Hernando Guillermo, Shanxin Yuan, Seungryul Baek, Tae-Kyun Kim, First-person hand action benchmark with RGB-D videos and 3D hand pose annotations, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2018) pp. 409–419.
[39] Bugra Tekin, Federica Bogo, Marc Pollefeys. H+ O: Unified egocentric recognition of 3D hand-object poses and interactions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2019) pp. 4511–4520.

Min-Yu Wu was born on September 7, 1991. He received the B.S. degree in Chemical Engineering from National Taiwan University, Taipei, Taiwan, in 2013 and the M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan in 2017. His research interests include the area of computer vision, pattern recognition and Human-Computer Interaction (HCI).

Pai-Wen Ting received a B.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 2016 and received the M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan in 2018. His research interests include the areas of computer vision, pattern recognition and virtual reality.

Ya-Hui Tang received the B.S. degree from National Cheng Kung University, Taiwan, in 1996, and the M.S. degree from National Chiao Tung University, Taiwan, in 1999. She is pursuing Ph.D. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan. She worked in IBM Taiwan as Software Engineer before entering National Taiwan University, 2000 to 2014. Her areas of research interest include computer vision, pattern recognition, and virtual reality.

En-Te Chou received a B.S. degree in Electrical and Computer Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2017 and the M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan in 2019. His research interests include the areas of computer vision, pattern recognition, and virtual reality.

Li-Chen Fu received B.S. degree from National Taiwan University, Taiwan, in 1981, and M.S. and Ph.D. degrees from University of California, Berkeley, U.S.A. in 1985 and 1987, respectively. Since 1987, he joined Dept. of Electrical Engineering and Dept. of Computer Science and Information Engineering, National Taiwan University (NTU), Taiwan, R.O.C. as a faculty member, and was awarded Lifetime Distinguished Professorship in 2007. Currently, he serves as Director of NTU Center for Artificial Intelligence (AI) and Advanced Robotics as well as Co-director of MOST (Ministry of Science and Technology)/NTU Joint Research Center for AI Technology and All Vista Healthcare. So far, he has received numerous recognitions, including IEEE Fellow (2004) and IFAC Fellow (2017). Internationally, he is serving as Editor-in-Chief of the Asian Journal of Control, and Advisory Committee member of Asian Control Association. His research interests include social robotics, smart home, visual detection and tracking, virtual reality, and control theory & applications.