Single Depth View Based Real-Time Reconstruction of Hand-Object Interactions

HAO ZHANG, YUXIAO ZHOU, YIFEI TIAN, JUN-HAI YONG, and FENG XU, Tsinghua University

Reconstructing hand-object interactions is a challenging task due to strong occlusions and complex motions. This article proposes a real-time system that uses a single depth stream to simultaneously reconstruct hand poses, object shape, and rigid/non-rigid motions. To achieve this, we first train a joint learning network to segment the hand and object in a depth image, and to predict the 3D keypoints of the hand. With most layers shared by the two tasks, computation cost is saved for the real-time performance. A hybrid dataset is constructed here to train the network with real data (to learn real-world distributions) and synthetic data (to cover variations of objects, motions, and viewpoints). Next, the depth of the two targets and the keypoints are used in a uniform optimization to reconstruct the interacting motions. Benefitting from a novel tangential contact constraint, the system not only solves the remaining ambiguities but also keeps the real-time performance. Experiments show that our system handles different hand and object shapes, various interactive motions, and moving cameras.

CCS Concepts: • **Computing methodologies** → *Perception*;

Additional Key Words and Phrases: Single depth camera, hand tracking, object reconstruction, hand-object interaction

ACM Reference format:
Hao Zhang, Yuxiao Zhou, Yifei Tian, Jun-Hai Yong, and Feng Xu. 2021. Single Depth View Based Real-Time Reconstruction of Hand-Object Interactions. *ACM Trans. Graph.* 40, 3, Article 29 (July 2021), 12 pages.
https://doi.org/10.1145/3451341

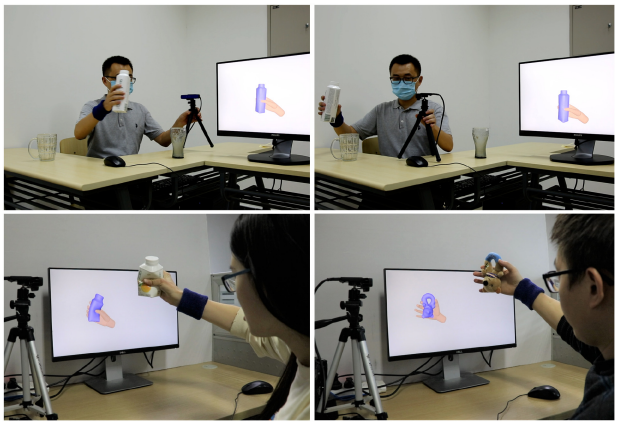

Fig. 1. Our system reconstructs hand poses, a complete object model, and the object's rigid/non-rigid motions using only one depth camera. It is robust to camera motions (top), different users (bottom), and different types of objects (e.g., the object with a hole at the bottom right).

1 INTRODUCTION

Dynamic reconstruction aims to reconstruct complex human/object motions and can be used in many applications, including VR/AR, character animation, and behavior/motion analysis. Interactive motion happens frequently in our daily lives—for example, we use hands to interact with various objects in working, eating, entertainment, and so on. Therefore, the reconstruction of hand-object interactions is important, and it is very useful in robotics, HCI, and remote control, among others.

There are many challenges in the reconstruction of hand-object interactions. At first, hand tracking itself is difficult because of the similarities of fingers in shape and appearance, the high dimensionality of the skeleton motions, and the self-occlusions [Tzionas et al. 2016]. Second, the objects manipulated by a hand vary a lot in colors, shapes, and motions. Besides rigid motions, some objects, such as stuffed toys and cloth, will have nonrigid motions that are in a high dimensional space and are complex to solve. Third, there are severe occlusions between hands and objects. All of these challenges prohibit previous works from achieving full reconstruction of hand-object interactions in real time, not to mention achieving the task with a single depth camera.

Many previous works have studied this task and made their contributions to solving this problem. Ballan et al. [2012] and Wang et al. [2013] achieve high fidelity reconstruction of hand-object interactions but can only track the rigid motion of an object through a multi-view system. Tzionas et al. [2016] and Tsoli and Argyros [2018] estimate 3D motions of a non-rigid object and an articulated hand with one RGBD camera but require an initial object model and rely on an offline system. Sridhar et al. [2016] track a hand and an object in real time but also need a pre-defined object model. Panteleris et al. [2015] can reconstruct hand poses and the motion and shape of an unknown object, but it can only handle rigid objects with low time performance. The work of Zhang et al. [2019] is the state-of-the-art real-time work that estimates both the 3D motions of an articulated hand and the geometry and motion of a manipulated object, but it demands two depth cameras placed face-to-face and a specialized calibration step. In brief, there is no existing work that can use a single sensor to simultaneously reconstruct the pose of a human hand, the 3D rigid and non-rigid motions of a manipulated object, and its 3D shape in real time.

This work was supported by the National Key R(&)D Program of China (2018YFA0704000), the NSFC (61822111 and 61727808), and Beijing Natural Science Foundation (JQ19015).
Authors' address: H. Zhang, Y. Zhou, Y. Tian, J.-H. Yong, and F. Xu (corresponding author), BNRist and School of Software, Tsinghua University, 30 Shuangqing Road, Haidian District, Beijing, China, 100084; emails: {zhanghao16, zhou-yx19, tyf18}@mails.tsinghua.edu.cn, {yongjh, feng-xu}@tsinghua.edu.cn.

0730-0301/2021/07-ART29 $15.00
https://doi.org/10.1145/3451341

In this article, we propose a novel technique to achieve the full reconstruction of hand-object interactions in real time by a single depth camera. To solve the mentioned challenges, we first design a neural network to segment the hand and the object, and estimate the 3D positions of the keypoints (skeleton joints and fingertips) of the hand from a single view depth stream. The predicted 3D keypoints and the segmented hand depth are both used to guide the hand tracking, whereas the segmented object depth is used to reconstruct and track the object. To train the neural network, we build a hybrid dataset that consists of a real data subset and a synthetic data subset. The real subset conveys the distribution information of the real depth data of hand-object interactions. The synthetic subset explores data variation by involving different object shapes, various hand-object interactive motions, and multiple perspective views, and contributes to the accuracy of the trained model as this subset contains ground truth. Finally, we use an optimization framework to jointly solve for the motions of the hand and the object. The energy function here considers the predicted 3D hand keypoints, the segmentation of the hand and the object, and a tangential contact effect between the hand and the object. The tangential contact constraint here is a novel component that involves the motion of the hand to solve the ambiguities of object tracking when the geometry feature is insufficient. Our formulation of this tangential contact constraint is efficient, and our whole system runs in real time. In summary, our contributions include the following:

- To the best of our knowledge, this is the first real-time system that uses only one depth camera to reconstruct hand-object interactions including solving the hand poses and 3D shape and rigid/non-rigid motions of an unknown object.
- We propose a joint learning network to simultaneously estimate 3D hand keypoints and segment the hand and the object, which keeps high prediction accuracy and fits the real-time scenario. We also build a hybrid dataset to train the network.
- We propose a new interactive term that considers the tangential contact effect between the hand and the object, which further helps to solve the severe ambiguities in this task.

2 RELATED WORKS

2.1 Hand or Object Reconstruction in Interactions

Many works have studied hand-object interactions. Some works focus on hand pose estimation in interactions. Cho et al. [2018], Hamer et al. [2009], and Taylor et al. [2017] track the hand mainly based on the generative paradigm. In addition, discriminative methods have also been used to estimate the hand pose in interaction [Romero et al. 2010; Rogez et al. 2015]. With the CNN being used to recover the continuous hand pose by Tompson et al. [2014], more and more works use deep neural networks to recover hand poses in interactions [Choi et al. 2017; Mueller et al. 2017, 2018; Zhou et al. 2020]. Zhou et al. [2020] recover hand pose and shape from a single RGB image at very high frame rates, which adopts a ResNet50 to extract features and compact CNN layers to estimate 2D and 3D hand keypoints. Different from the works focusing on the hand, some works recover the object information in hand-object interactions. Rusinkiewicz et al. [2002], Tzionas and Gall [2015], Wang and Hauser [2019], and Weise et al. [2008, 2011] reconstruct the models of rigid objects in interactions. In addition, Petit et al. [2018] obtain non-rigid motions of objects manipulated by hands.

2.2 Joint Hand-Object Reconstruction in Interactions

Different from the works focusing on either the hand or the object reconstruction in interactions, other works reconstruct both of them. Many of these techniques reconstruct hand-object interactions by tracking the articulated motion of the hand and the rigid/non-rigid motions of the object based on optimization methods. Ballan et al. [2012] and Wang et al. [2013] use a multi-view setup to track the 3D motions of the hand and the object in interactions, whereas Panteleris and Argyros [2017] use a stereo camera. Recently, RGBD cameras have become more and more popular in the study of hand-object interactions because they supply range cues directly. Kyriazis and Argyros [2013, 2014] use an RGBD camera to reconstruct complex hand-object interactions through an offline system. Schmidt et al. [2015] and Sridhar et al. [2016] achieve real-time tracking of the hand and the manipulated rigid object. Tzionas et al. [2016] and Tsoli and Argyros [2018] estimate the 3D non-rigid motion of the object in addition to the articulated hand. Even though these works are impressive, they require a pre-defined 3D model of the object. On the contrary, Oikonomidis et al. [2011] and Panteleris et al. [2015] recover the motions of the hand and the object without requiring pre-defined object models. They can even estimate the shape parameters of objects or reconstruct water-tight object models. However, they can only handle rigid objects with low time performance. Recently, Oberweger et al. [2019] proposed to use a deep neural network with a feedback loop to estimate the 3D poses of the hand and the object from a depth image.

Besides using depth, some works try to reconstruct hand-object interactions directly from color images by use of neural networks. Tekin et al. [2019] take a sequence of color frames as input and output per-frame 3D hand poses, object poses, object classes, and action categories, as well as per-sequence interaction classes. Hasson et al. [2019] directly recover the hand pose and the shape of a non-rigid object from a single color image, heavily depending on a synthetic dataset. However, it is difficult to reconstruct various objects with precise geometries due to the limited coverage of the training dataset and the domain gap between the synthetic data and real data.

The work most similar to ours is InteractionFusion [Zhang et al. 2019]. It simultaneously recovers the 3D poses of an articulated hand, a water-tight object model, and the rigid/non-rigid motions of the object in real time. However, it demands two depth cameras placed face-to-face. A specialized calibration step is inevitable, which largely limits its usage in real applications.

3 OVERVIEW

Our system uses only one depth camera to record hand-object interactive motions. The system runs frame by frame and achieves real-time performance. It contains three steps. In the first step, we propose a joint learning network to predict 3D hand keypoints and segment object from the hand simultaneously. The second step is

to optimize a unified energy function to estimate accurate hand poses and rigid/non-rigid motions of the object. Finally, the newly segmented depth of the object is fused into a static object model based on the estimated object motions, obtaining a smoother and more complete object model. The obtained hand pose, object motion, and object model are all used in the processing of the next input depth.

4 PRELIMINARIES

In this section, we present the hand and object representations and other notations used in this article. The input depth sequence is denoted as $\mathcal{D}^t$, where t is the frame index. A synchronized color sequence is also captured along with the depth sequence, but it is only used to detect the hand-object depth data, as Tkach et al. [2016] did. Similar to Zhang et al. [2019], we use sphere-mesh [Tkach et al. 2016] to model the hand and use the **Truncated Signed Distance Function (TSDF)** and node graph to model the object shape and motions [Newcombe et al. 2015; Guo et al. 2017].

Specifically, sphere-mesh uses two types of sphere block to construct a hand model. One type is determined by two spheres together, which is used to approximate fingers. These two spheres define the shape of the block and are named *end spheres*. In between the two end spheres, there are many intermediate spheres smoothly changing their positions and radius from one end sphere to the other. These spheres are called *middle spheres*. With similar definitions, the other type of sphere block is determined by three end spheres, which is used to approximate palm and wrist. Therefore, the hand model $\mathcal{H}$ is determined by all of the end spheres, which can be formulated as follows:

$$\mathcal{H} = \mathcal{F}\left\{\cup_{j=0}^{J-1}\mathcal{B}_j\left[\cup_{i=0}^{I-1}(r_i, \boldsymbol{c}_i)\right]\right\}, \tag{1}$$

where $\mathcal{B}[\cup_{i=0}^{I-1}(r_i, \boldsymbol{c}_i)]$ constructs different sphere blocks from I end spheres and $\mathcal{F}\{\cup_{j=0}^{J-1}\mathcal{B}_j\}$ obtains the hand mesh by extracting the surface of all J blocks. From Equation (1), the sphere-mesh can fit any hand shape by giving the end spheres proper radius and center positions. The sphere-mesh also defines a hand skeleton to describe hand pose θ. Specifically, there are 17 joints in the skeleton of sphere-mesh and 28 valid degrees of freedom in total. The end spheres are attached to the joints and then the locations of the end spheres $\boldsymbol{c}$ are determined by a 28 degrees-of-freedom vector that is exactly the hand pose θ. In this article, we track the hand by estimating θ. For more details of sphere-mesh, please refer to the work of Tkach et al. [2016].

For object modeling, we follow the work of Newcombe et al. [2015]. Specifically, we use TSDF to represent the static model of an object $\mathcal{M} = \{d(\boldsymbol{x}), w(\boldsymbol{x})\}$ in the canonical frame and use the warping field $\mathcal{W}(\boldsymbol{x})$ to represent the motion of the object, by which the static model can be deformed to each frame in the sequence. $\boldsymbol{x}$ is a coordinate in the canonical frame, $d(\boldsymbol{x})$ is the signed distance from $\boldsymbol{x}$ to its closest surface, and $w(\boldsymbol{x})$ is the confidence of $d(\boldsymbol{x})$. The static mesh of the object O_s can be formulated as

$$O_s = \left\{(\boldsymbol{v}, \boldsymbol{n}) | d(\boldsymbol{v}) = 0, \boldsymbol{n} = \frac{\nabla d(\boldsymbol{v})}{\| \nabla d(\boldsymbol{v})\|_2}\right\}, \tag{2}$$

and the dynamic mesh (a mesh deformed from the static mesh) of the object O_l as

$$O_l = \{(\boldsymbol{v}_l, \boldsymbol{n}_l) | \boldsymbol{v}_l = \mathcal{W}(\boldsymbol{v})\boldsymbol{v}, \boldsymbol{n}_l = \mathcal{W}(\boldsymbol{v})\boldsymbol{n}\}. \tag{3}$$

Here, O_l is the dynamic mesh of the object aligned with one input depth frame. In our implementation, we use discrete voxels to store TSDF and the warping field. In addition, we use a marching cube to get the static mesh of the object O_s from $\mathcal{M}$. For the warping field, we use a node-graph based representation. Specifically, we sample nodes uniformly on O_s. Each node has a coordinate $\boldsymbol{p}$ in the canonical frame and a transformation $\boldsymbol{T}$ in world coordinates. Thus, the node-graph based warping field $\mathcal{W}_N$ can be represented as a set of nodes $\mathcal{W}_N = \cup_{m=0}^{M-1}(\boldsymbol{p}_m, \boldsymbol{T}_m)$ and their spatial adjacent information ϵ. The motion of a vertex $\mathcal{W}(\boldsymbol{v})$ can be obtained by interpolating transformations of the nearest nodes. Please refer to the work of Newcombe et al. [2015] for more details. In this article, we track rigid/non-rigid motions of the manipulated object by solving $\mathcal{W}_N$.

5 METHOD

Our system works frame by frame, and thus before processing a new frame, the results of all previous frames are known. For the first frame, the hand pose is estimated by our hand tracking method (to be formulated in detail in Section 5.3) without considering the object model, and the depth of the object is directly used as the initial object model. The warping field is initialized to $\mathbf{0}$. For a new frame t, we have an input depth $\mathcal{D}^t$, the previous object model O_l^{t-1} and warping field $\mathcal{W}_N^{t-1}$, and the hand poses of the previous frames $\{\theta^0, \theta^1, \ldots, \theta^{t-1}\}$.

5.1 Hand Keypoints Prediction and Hand-Object Segmentation

In this step, the input depth is fed into our neural network to predict the 3D keypoints of the hand and the hand-object segmentation. Segmentation is also performed by Zhang et al. [2019] to guide the reconstruction of the hand and object by their own depths, respectively. But Zhang et al. [2019] do not need 3D keypoints, as the hand depth recorded by the two cameras is almost complete and sufficient for the following hand pose estimation. In this work, as we only use one depth camera, the ambiguity is much more severe. We therefore propose to use a single view based hand keypoints prediction technique to constrain the hand pose estimation. As the keypoints distribution is learned from a large amount of data, they can be used to solve the ambiguity. Furthermore, as we aim for real-time performance, we propose to use one network to perform the two tasks (keypoints prediction and segmentation) together.

In Figure 2, we illustrate our network structure, which is inspired by the work of Zhou et al. [2020]. It comprises two modules: a ResNet-based feature extraction and keypoints prediction module, and a decoder-based segmentation module. The structure of the feature extraction and keypoints prediction module is similar to the DetNet of Zhou et al. [2020]. To be specific, the valid pixel values in an input depth image (320×240) are first normalized to $[0, 1]$ using $z = (z' - d_{min}) * s$, where $s = (d_{max} - d_{min})^{-1}$. Here, z' and z are the original and normalized depth values, and d_{min}

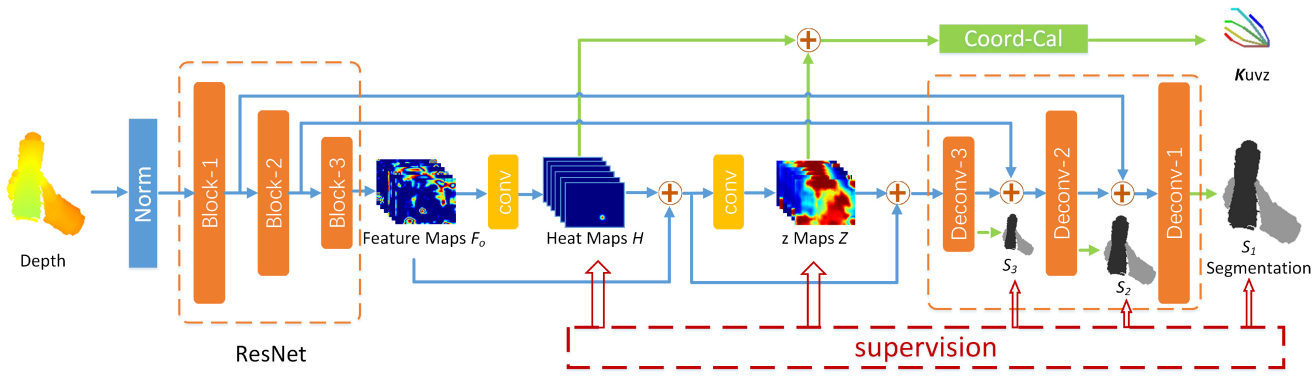

Fig. 2. Our network first uses a ResNet to extract features of the input depth image. Then, heat maps and z maps of the hand keypoints are predicted by two compact CNN layers. The 3D information of the hand keypoints can be extracted from the heat maps and z maps directly. Finally, a decoder is appended to predict the segmentation.

and d_{max} are the minimum and maximum of valid depth values. Then the normalized image is fed into ResNet. The network outputs a feature volume F_o of size $40 \times 30 \times 256$ along with intermediate features in different resolutions (80×60, 160×120). Then, a compact CNN takes the feature volume F_o as input and outputs a heat map volume of 21 pre-defined hand keypoints, H ($40 \times 30 \times 21$). The heat map volume H and the feature volume F_o are concatenated and fed into another compact CNN, which outputs a z map volume Z ($40 \times 30 \times 21$), indicating the estimated z values of the hand keypoints. The heat maps of the keypoints and the z maps are used to calculate the 3D positions of the keypoints $\boldsymbol{K}_{uvz}$. Specifically, we first get a 2D keypoint (u, v) by finding the pixel with the largest value in each channel of H, and the predicted z value can be retrieved at $Z(u, v, :)$. Then, $(u', v') = 8 * (u, v)$ is the 2D position in the image of the original resolution, and $z' = z * s^{-1} + d_{min}$ is the depth value of the keypoint. Given the intrinsic parameters Π of the depth camera, the 3D position (x_h, y_h, z_h) can be obtained by

$$(x_h, y_h, z_h)^{\mathrm{T}} = z' \Pi^{-1} (u', v', 1)^{\mathrm{T}}. \tag{4}$$

Finally, a decoder is added to get the segmentation masks of the hand $\mathcal{I}_h$ and the object $\mathcal{I}_o$, which takes the feature volume F_o, the heat map volume H, the z map volume Z, and the intermediate features as input.

Even though our network is similar to that of Zhou et al. [2020], we have the following modifications. First, our network takes depth maps as input, whereas that of Zhou et al. [2020] takes color images. Second, Zhou et al. [2020] output the keypoints in a normalized coordinate system, which is unable to be combined with the depth in the camera coordinate system for the following optimization step. So we use uv plus z in the camera coordinate system to establish the combination. Third, our network outputs a segmentation by appending a decoder. This is also important, as the two tasks are majorly achieved by the same network, which saves a lot of computation cost for our real-time scenario.

Supervisions are applied to the heat maps H, z maps Z, and segmentation maps (S_1, S_2, S_3) to train this network. The loss function consists of three terms, namely heat map loss $\mathcal{L}_{hm}$, z map loss $\mathcal{L}_{zm}$, and segmentation map loss $\mathcal{L}_s$. $\mathcal{L}_{hm} = \|H - H^{GT}\|_F^2$ tries to ensure that the predicted heat maps H are close to the ground truth H^{GT}, where $\|\cdot\|_F$ is Frobenius norm. H^{GT} is generated by 21 Gaussian functions centered at 21 2D hand keypoints with standard deviation $\sigma = 1$. $\mathcal{L}_{zm} = \|H^{GT} \odot (Z - Z^{GT})\|_F^2$ tries to ensure that the predicted z values around 2D hand keypoints are close to the ground truth, where $\odot$ means the element-wise matrix product. Z^{GT} is constructed by tiling normalized z values of the ground truth of keypoints positions. $\mathcal{L}_s = \sum_{idx=1}^{3} \|S_{idx} - S_{idx}^{GT}\|_1$ is to supervise the predicted segmentation maps (including final output and two intermediate low-resolution segmentation maps), where $\|\cdot\|_1$ is entry-wise L1 norm. S^{GT} is generated by setting the values of the hand pixels to 1 and other pixels to 0.

5.2 Hybrid Dataset Construction

To train the aforementioned network, we need a dataset containing depth images of interactive motions with their corresponding segmentation masks of hands and objects, as well as the ground truth keypoints positions of the hands. This dataset is not easy to obtain due to the strong requirement on the ground truth information. Manually labeling is impractical, as the labeling effort in this task is huge, and it is difficult to label occluded 3D keypoints. In this case, we implement the state-of-the-art interaction reconstruction technique [Zhang et al. 2019] to get the labeling by two streams of depth. After that, we manually select the frames with diverse interactive poses and remove the ones with significant errors to create the real dataset. The remaining labeling error is left in the data. However, it is still costly to create a large dataset covering various objects, and the labeling is not perfect.

Consequently, we create a hybrid dataset that contains not only the aforementioned real data with coarse labels but also a large amount of synthetic data with perfect segmentation and keypoints labeling. To construct the synthetic training dataset, we utilized a public hand simulation library, *CLAP* [Verschoor et al. 2018], and a simulation environment, *Unreal Engine*, to build a hand-object interaction system. By using the LeapMotion toolkit, the system obtains the user's hand motion, and the physical simulation components use the hand motion information to drive the interaction between a virtual hand and a virtual object in the system. As the interacting motion is rendered to the user in real time, the user can adjust his/her hand motion to realize various interactive

actions of the virtual hand and the virtual object in the system. In this manner, we can freely interact with objects of different types, shapes, and scales, and get rendered depth images from arbitrary viewpoints. Furthermore, we have all of the ground truth information to perform the supervised training of the method in the previous section. In general, our hybrid dataset contains 14 real motion sequences (each has two viewpoints, 6,703 frames in total) with 12 types of objects manipulated by one user. However, we have 25 synthetic motion sequences (each has five viewpoints, 58,764 frames in total) containing 13 different objects and a hand model. Our hybrid dataset covers much diversity in hand motions, including large move trajectory and various manipulating poses (grab, hold, pinch, support, etc.). The objects in the dataset have various shapes (cube, sphere, cylinder, etc.) and sizes (from 4 to 22 cm). Both rigid and non-rigid interactions are performed. Please refer to the supplementary document for more details. Furthermore, even though we do not use color images, we still recorded or synthesized the color image for each frame. The dataset will be released for future research.

5.3 Joint Hand-Object Tracking

This step takes the depth and the results of our neural network as input, and outputs the hand pose and the rigid/non-rigid motion of the object. Given depth $\mathcal{D}^t$, we solve for the hand pose θ^t and the warping field $\mathcal{W}_N^t$ by using O_l^{t-1}, $\mathcal{W}_N^{t-1}$, $\{\theta^{t-1}, \theta^{t-2}, \ldots, \theta^0\}$, and the newly predicted hand keypoints $\boldsymbol{K}_{uvz}^t$, and segmentation masks $\mathcal{I}_h^t$ and $\mathcal{I}_o^t$. Similar to Zhang et al. [2019], we solve for θ^t and $\mathcal{W}_N^t$ by optimizing a unified energy:

$$E_{\text{tol}}(\mathcal{W}_N^t, \theta^t) = \omega_{\text{uvz}} E_{\text{uvz}}(\theta^t) + \omega_{\text{tac}} E_{\text{tac}}(\mathcal{W}_N^t) + E_{\text{ori}}(\mathcal{W}_N^t, \theta^t), \tag{5}$$

where $E_{\text{uvz}}(\theta^t)$ is our keypoints-based hand tracking term and $E_{\text{tac}}(\mathcal{W}_N^t)$ is our tangential contact term. $E_{\text{ori}}(\mathcal{W}_N^t, \theta^t)$ is the original energy terms in the work of Zhang et al. [2019]. It includes the depth to hand surface term E_{d2m}, the hand silhouette term E_{m2d}, the hand pose and joint priors terms E_{pose} and E_{lim}, the hand collision term E_{coll}, the hand joints temporal term E_{temp}, the LSTM-based hand pose prediction term E_{lstm} , the depth to object surface term $E_{\text{o-dep}}$, the object silhouette term $E_{\text{o-silh}}$, the object non-rigid regularization $E_{\text{o-reg}}$, and the hand-object contact term E_{itc}. We implement the original energy terms as in the work of Zhang et al. [2019] and use the same data as their work to train the LSTM-based network for hand pose prediction. For more details of the original energy terms, please refer to the work of Zhang et al. [2019]. In the following, we will formulate the novel terms in our system.

5.3.1 Keypoints-Based Hand Tracking. $E_{\text{uvz}}(\theta^t)$ involves the predicted keypoints to solve the ambiguity in single view hand pose estimation. It is formulated as

$$E_{\text{uvz}}(\theta^t) = \|\boldsymbol{K}(\theta^t) - \boldsymbol{K}_{uvz}^t\|_2^2, \tag{6}$$

where $\boldsymbol{K}(\theta)$ stands for the keypoints positions of the sphere-mesh at pose θ. This term is necessary for single view based hand pose tracking. We know that in hand-object interactions, a large part of the hand may not be observed due to the heavy occlusions. In this case, none of the terms in the work of Zhang et al. [2019] can help to solve for a good hand pose. However, in our system, we have a neural network that is trained on a large amount of realistic and synthetic hand-object interaction data accompanying the corresponding ground truth of hand keypoints positions. Thus, the network can learn the prior knowledge of the keypoints distribution even though the hand is partially occluded by the object. By involving the output of the network in the optimization, our system is able to estimate plausible hand poses from the single view input.

Note that there is inevitably some noise in the predicted keypoints, which aggravates finger jitters. To alleviate this problem, two strategies are implemented in our experiments. First, ω_{uvz} is set to 10 initially and decreased by $10/iter_{\max}$ after each iteration, where $iter_{\max}$ is the maximum iteration for one frame and is set to 5. Then, a 1-euro filter [Casiez et al. 2012] is used to smooth the hand pose further after $iter_{\max}$ iterations.

5.3.2 Tangential Contact Term for Object Tracking. Besides the hand tracking, the object tracking also becomes more difficult in single view due to insufficient information. Occlusion caused by the hand makes the situation worse, especially for some objects with fewer geometry features. Our idea here is to use the motion of the hand to constrain the motion of the object. Since the object is manipulated by the hand, the motion of the object has a strong correlation with the hand motion. Zhang et al. [2019] have utilized the correlation by considering the push effect in the normal direction of the contact zone. Our method further considers the friction effect in the tangential direction, which can also be used to track the object. Therefore, we introduce a new term called *tangential contact term* here to help to track the object. Given the paired contact points, the tangential contact term E_{tac} is formulated as

$$E_{\text{tac}}(\mathcal{W}_N^t) = \sum_{(\boldsymbol{v}_l, \boldsymbol{v}_s) \in C_{\text{itc}}} \|P_{\parallel}(\boldsymbol{v}_l(\mathcal{W}_N^t) - \boldsymbol{v}_l(\mathcal{W}_N^{t-1})) - P_{\parallel}(\Delta \boldsymbol{v}_s)\|_2^2, \tag{7}$$

where $\boldsymbol{v}_l$, $\boldsymbol{v}_s$ are the paired contact points on the object mesh O_l and the hand mesh $\mathcal{H}$, $P_{\parallel}$ is a projection operation to remove the motion along $\boldsymbol{v}_l$'s normal $\boldsymbol{n}_l$. Thus, $P_{\parallel}(\Delta \boldsymbol{v}_s)$ is the projection of $\boldsymbol{v}_s$'s motion on the tangential direction of the object surface and formulated as follows:

$$P_{\parallel}(\Delta \boldsymbol{v}_s) = (\mathbf{I} - \boldsymbol{n}_l \boldsymbol{n}_l^{\mathrm{T}})(\boldsymbol{v}_s(\theta^t) - \boldsymbol{v}_s(\theta^{t-1})). \tag{8}$$

Notice that here the contact points are detected on the reconstructed hand and object of the last frame $t-1$ and will be updated according to the result of this frame for the tracking of the next frame. So the contact points need to be updated frame by frame. Otherwise, the contact points will keep fixed and the object will always move with the initial contact points on the hand. From our experiments, we notice that updating the contact points once for one frame is good enough even though the contact points may be continuously changed with the motion. So we use this updating manner to trade off the accuracy and efficiency. We also find that a value between 0.1 and 0.5 is good for ω_{tac}. Another problem is that, from Equation (8), the motion of the contact point on the hand is related to the variable θ^t that is still unknown before finishing the calculation of this frame. To handle this, we slightly change the optimization strategy of Equation (5), which is formulated in Section 5.4.

Notice that for the tangential contact term, the manner of detecting and maintaining the contact points is largely different from that in the work of Zhang et al. [2019]. To be specific, Zhang et al. [2019] just detect the contact points on the object surface. But to formulate the tangential contact term, it is necessary to detect and maintain the contact points on not only the object but also the hand surface. The method to detect and maintain the contact point pairs is introduced in Section 5.3.3.

5.3.3 Contact Point Detecting. To construct the tangential contact term, we need to find the contact point pairs on both the object model and the hand model, and estimate the motion of the contact point on the hand between two consecutive frames as shown in Equation (8). In this case, when updating contact point pairs, for each vertex $\boldsymbol{v}_l$ on the object model O_l, we first calculate the minimum TSDF value $\mathcal{T}^b(\boldsymbol{v}_l)$ to sphere blocks of the hand model, where b indicates a block with the minimum TSDF value. If $\mathcal{T}^b(\boldsymbol{v}_l) \leq 0$, we treat $\boldsymbol{v}_l$ as a contact point on the object and calculate its corresponding contact point on the sphere-mesh $\boldsymbol{v}_s$ by simply projecting $\boldsymbol{v}_l$ to the surface of the sphere block b:

$$\boldsymbol{v}_s = \boldsymbol{c}(\boldsymbol{v}_l) + \begin{cases} r_c \frac{\boldsymbol{v}_l - \boldsymbol{c}(\boldsymbol{v}_l)}{\|\boldsymbol{v}_l - \boldsymbol{c}(\boldsymbol{v}_l)\|_2}, & \boldsymbol{n}_l \cdot (\boldsymbol{v}_l - \boldsymbol{c}(\boldsymbol{v}_l)) < 0 \\ -r_c \boldsymbol{n}_l, & \boldsymbol{n}_l \cdot (\boldsymbol{v}_l - \boldsymbol{c}(\boldsymbol{v}_l)) = 0 \\ -r_c \frac{\boldsymbol{v}_l - \boldsymbol{c}(\boldsymbol{v}_l)}{\|\boldsymbol{v}_l - \boldsymbol{c}(\boldsymbol{v}_l)\|_2}, & \boldsymbol{n}_l \cdot (\boldsymbol{v}_l - \boldsymbol{c}(\boldsymbol{v}_l)) > 0 \end{cases}, \quad (9)$$

where $\boldsymbol{c}(\boldsymbol{v}_l)$ is the center of the closest sphere (an end sphere or a middle sphere) of $\boldsymbol{v}_l$ on the sphere block b, and r_c is the radius of the closest sphere.

Revisiting Equation (8), now we know $\boldsymbol{v}_s(\theta^{t-1})$ but still do not know $\boldsymbol{v}_s(\theta^t)$, as we do not know $\boldsymbol{v}_s$'s corresponding position in frame t. To find the corresponding position of $\boldsymbol{v}_s$ in θ^t, we use a local coordinate representation relative to each sphere block to represent the contact point on the sphere-mesh, which is invariant to the hand pose θ but can be transferred to the world coordinate representation by θ. To compute the local coordinate, a local coordinate system is built for each sphere block at the pose $\theta = \mathbf{0}$. The local coordinate system of a sphere block can be originated at any position with three orthogonal axes, but it should be bound to its corresponding block rigidly. In this case, given a hand pose θ, the transformation from a local coordinate system to the world coordinate system $\boldsymbol{T}^b_{L\to G}(\theta)$ can be fully determined. Thus, a contact point $\boldsymbol{v}^b_s$ in the local coordinate system of block b can be transformed into the world coordinate system as

$$\boldsymbol{v}_s = \boldsymbol{T}^b_{L\to G} \boldsymbol{v}^b_s. \quad (10)$$

We can also perform the inverse transformation as

$$\boldsymbol{v}^b_s = \boldsymbol{T}^b_{G\to L} \boldsymbol{v}_s. \quad (11)$$

Given these definitions, we can calculate $\boldsymbol{v}_s(\theta^t)$ by first estimating its local coordinate $\boldsymbol{v}^b_s$ from its global coordinate in frame $t-1$ ($\boldsymbol{v}_s(\theta^{t-1})$), and then transfer the local coordinate $\boldsymbol{v}^b_s$ to the world coordinate of frame t:

$$\boldsymbol{v}_s(\theta^t) = \boldsymbol{T}^b_{L\to G}(\theta^t) \boldsymbol{T}^b_{G\to L}(\theta^{t-1}) \boldsymbol{v}_s(\theta^{t-1}). \quad (12)$$

To make the tangential contact term efficient, we utilize several strategies in the detection of contact points. For the hand, we only calculate the contact points on fingers since the hand-object interaction is mainly controlled by the fingers. In this case, the kinematic frames of finger joints defined in the work of Tkach et al. [2016] are used as the local coordinate systems of finger blocks in our implementation. As for the object, we allow at most one contact point for each triangle. In addition, we set the maximum number of contact points to 5,000. As suggested by our experiments, these operations achieve a good trade-off between computation efficiency and accuracy.

5.4 Optimization Strategy and Fusion

We use the Gauss-Newton approach to solve the object motion field $\mathcal{W}^t_N$ and the hand pose θ^t by minimizing the total energy of Equation (5). For each frame, we use a two-stage strategy to optimize the energy. At the first stage, we solve $\mathcal{W}^t_N$ and θ^t iteratively without considering the tangential contact term. Then, we track the object for the second time with the tangential contact term, which considers the friction effect of the moving fingers on the object. In our system, each stage executes only once for one frame.

After tracking the object, the depth of the object is fused into the static model of the object $\mathcal{M}$ that is then used to construct O^t_l for the tracking of the next frame.

6 EXPERIMENTS

In this section, we first report the implementation details of our system. Then, we evaluate the key parts of our work and compare our work with previous state-of-the-art methods quantitatively and qualitatively. Finally, we show more results of our work and discuss some issues in this work.

6.1 Implementation

Our system uses one Intel RealSense SR300 to capture a depth stream of hand-object interactions. The intrinsic parameters of the depth camera are read from the device directly. Two NVIDIA TITAN Xp GPUs are used for algorithm execution. One GPU runs the network for hand keypoints prediction and hand-object segmentation, whereas the other GPU runs the energy optimization. We implement pipelines in our system for data preprocessing, hand keypoints prediction and hand-object segmentation, and energy optimization. Finally, we achieve 25 fps by our implementation based on C++, CUDA, and the TensorFlow library.

6.2 Evaluations

6.2.1 Evaluation of the Network. We evaluate our network on the newly built hybrid dataset. Specifically, we select the real data of nine objects and all of the synthetic data as our training dataset and the remaining real data as our evaluation dataset (1,454 frames in total).

At first, we evaluate our network by checking whether the joint design has any effect on the performance of the two tasks. To do this evaluation, we train two sub-networks. One sub-network is to predict hand keypoints (the KPT network) by throwing away the appended decoder for hand-object segmentation. The other sub-network is to do hand-object segmentation (the SEG network) by discarding the parts of keypoints prediction. The percentage of

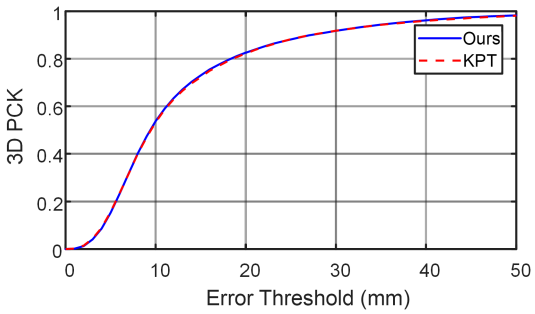

Fig. 3. Evaluation of the network on keypoints prediction.

Table 1. Evaluation of the Network on Keypoints Prediction, Segmentation, Time Performance, and Trainable Variables

Network	3D Error (mm)	MIoU	Runtime	Trainable Var.
Ours	13.1±11.2	0.943	20 ms	15.49M
KPT	13.3±11.7	—	13 ms	11.75M
SEG	—	0.947	17 ms	14.06M

Table 2. Quantitative Evaluation of the Hybrid Dataset on Hand Keypoints Prediction and Hand-Object Segmentation

Network	3D Error (mm)	MIoU
Without syn	14.8±13.0	0.923
With syn	13.1±11.2	0.943

We show performances of the networks trained with and without the synthetic dataset.

correct 3D keypoints (PCK) of hand keypoints prediction is illustrated in Figure 3 and the mean error is shown in Table 1. The results of **Mean Intersection over Union (MIoU)** for hand-object segmentation of the two networks are also shown in Table 1, where the runtime and memory consumption of each network are also provided. From the results, we find that the designed joint network achieves almost the same performance with the special networks for hand keypoint prediction and hand-object segmentation. Furthermore, our joint network saves one-third of the runtime and one-half of the trainable variables. The reason for the equal performance of joint learning and specific learning may be related to the consistency of the two tasks. We guess that the two tasks are inherently consistent, and thus many features can be shared.

6.2.2 Evaluation of the Hybrid Dataset. We demonstrate the effectiveness of our hybrid dataset by quantitative and qualitative experiments. We train a network using only a real dataset and compare it with the network trained by our hybrid dataset. We first test these two networks on the real test dataset. The quantitative result is shown in Table 2. We find that the synthetic dataset improves the performance of hand keypoints prediction and hand-object segmentation. This results in better hand pose estimation as shown in Figure 4. In addition, more accurate hand pose and object segmentation leads to better object reconstruction as shown in Figure 5. To quantitatively demonstrate this, we use the model obtained by

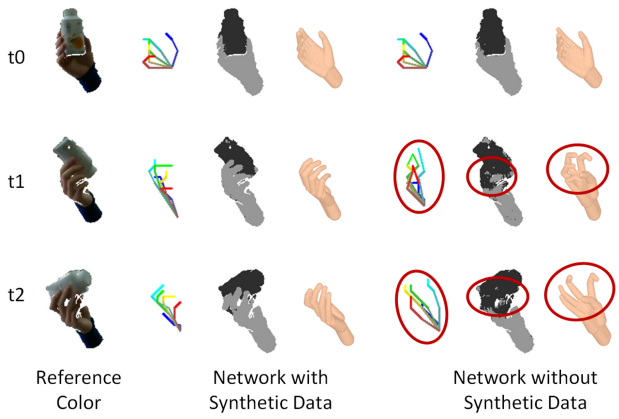

Fig. 4. Qualitative evaluation of the hybrid dataset on hand tracking. In this figure, the hand pose is optimized from the predicted hand keypoints and the segmented hand depth. Without training on the synthetic data, the tracking contains more errors. For each group of results, the left shows the predicted 3D keypoints, the middle shows the segmentation result. and the right shows the tracked hand.

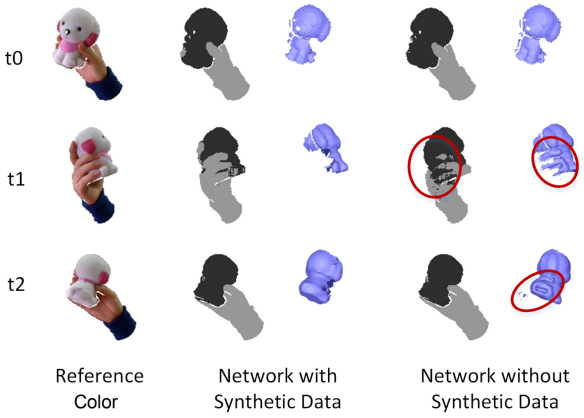

Fig. 5. Qualitative evaluation of the hybrid dataset on object reconstruction. Without training on the synthetic data, the segmentation contains more errors, leading to more artifacts in model fusion.

Zhang et al. [2019] as a reference to evaluate the numerical errors of the fused models. We first align the reconstructed model with the reference model manually and then refine the alignment by ICP. Finally, we measure the surface distance from the aligned fused model to the reference model. The comparison results are illustrated in Figure 6, and the mean distance is shown in Table 3. From these results, we find that the synthetic dataset also improves the model reconstruction of the object.

6.2.3 Evaluation of the Tangential Contact Term. We have introduced a tangential contact term on object tracking. This term considers the friction effect of the hand on object surfaces in interactions. To quantitatively evaluate the tangential contact term, we record one sequence and annotate it by labeling five points on the object manually. The quantitative results are shown in Figure 7. We also show four qualitative results of this sequence at frames

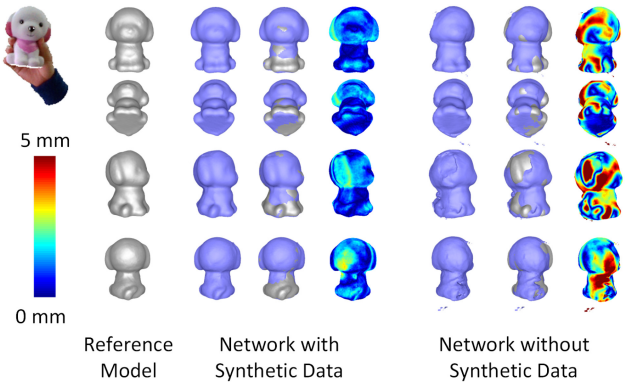

Fig. 6. Qualitative evaluation of the hybrid dataset on object reconstruction. Without training on the synthetic data, the resulting fused model contains more artifacts and the surface of the fused model is farther from the reference model obtained by Zhang et al. [2019].

Table 3. Mean Vertex Distance (mm) from the Reconstructed Model to the Reference Model

Network	Without Syn	With Syn
Mean distance (mm)	2.4	1.0

195, 205, 226, and 235 in Figure 8. We find that the tangential contact term helps to constrain the object tracking when lacking geometry features (frame 205), and it has no side effect when there are sufficient geometry features (frame 195). We also presented three additional results in our accompanying video to further verify that the tangential contact term can truly improve the tracking of the object when geometry features are missing.

6.3 Comparisons

6.3.1 Comparisons with Zhang et al. [2019] and Bo et al. [2010]. In the literature, the work of Zhang et al. [2019] is the only one aiming at the same goal as ours, but it uses two depth cameras, whereas we use only one. We compare our system with Zhang et al. [2019] on their annotated dataset with ground truth. The same as Zhang et al. [2019], we project the five tracked fingertips on to the reference color image and measure the average errors to the ground truth. Since the dataset contains images from two opposite directions (*View0* and *View1*), we evaluate the hand tracking of our system in the two views, whereas only the depth stream of *View0* is fed into our system. The quantitative results are shown in Figure 9 and Table 4. The qualitative results are shown in Figure 10 and the supplementary video. It can be seen that our system achieves comparable hand tracking with Zhang et al. [2019] in *View0*. For *View1*, even though the projected fingers are not visible to our system, we still give a reasonable result with satisfactory accuracy.

We also compare our network with DenseAttentionSeg [Bo et al. 2020] on hand-object segmentation, as it has already shown good performance by Zhang et al. [2019]. The comparison with Bo et al. [2020] on our hybrid dataset is shown in Table 5. It can be seen that our network is slightly better than DenseAttentionSeg on segmentation accuracy, and our model is much smaller than it is.

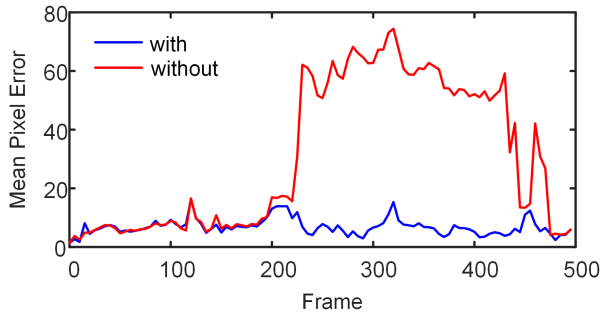

Fig. 7. Quantitative evaluation of the tangential contact term. The tracking of the object fails without the tangential contact term, leading to large tracking errors.

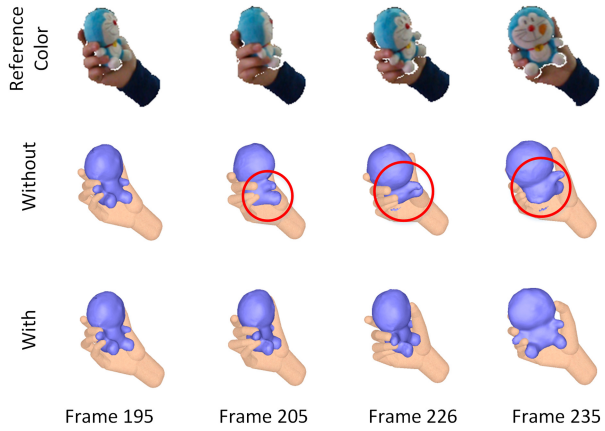

Fig. 8. Qualitative evaluation of the tangential contact term at frames 195, 205, 226, and 235.

Table 4. Average Pixel Error of Different Methods for Different Sequences

	View0		View1	
	Ours	Zhang et al.	Ours	Zhang et al.
RotatePepper	9.2	16.0	13.9	10.8
PourBottle	6.3	6.0	9.3	6.5
ReconstructCat	12.1	11.9	16.2	11.0

The average error is calculated from all of the frames shown in Figure 9.

6.3.2 Comparison with Zhou et al. [2020]. The network used in our system is inspired by the DetNet of Zhou et al. [2020]. As Zhou et al. [2020] also reconstruct a 3D hand model from keypoints positions by an additional IKNet, we compare our method with their work on hand reconstruction with their released code. The complete comparison is shown in our accompanying video. Here we present some snapshots in Figure 11. It can be seen that by reconstructing the object model jointly with the hand reconstruction, our system gives more robust and accurate hand poses when interacting with a large object. Please note that Zhou et al. [2020] only reconstruct the hand, whereas we additionally reconstruct

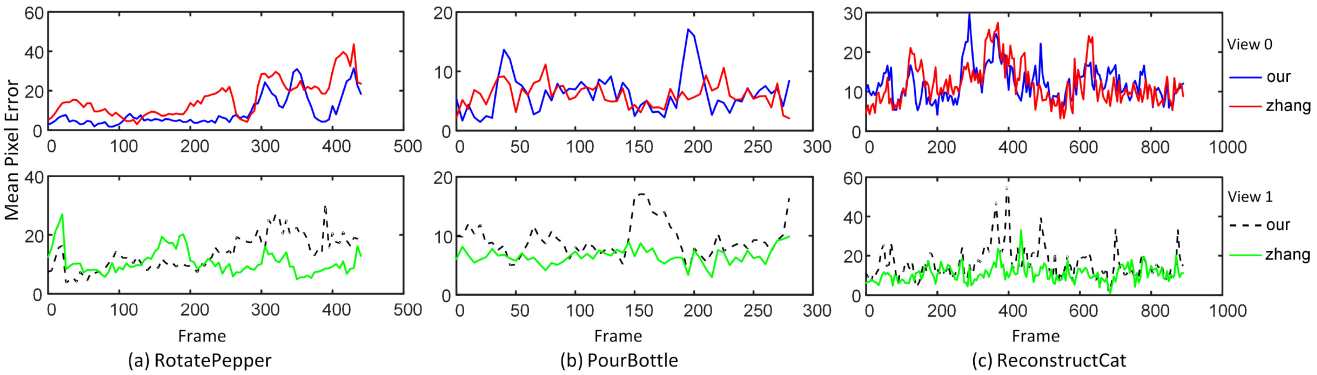

Fig. 9. Quantitative comparison of hand tracking between our work and [Zhang et al. 2019]. Note that even though we show results of two views, we only use the depth of *View0* as input.

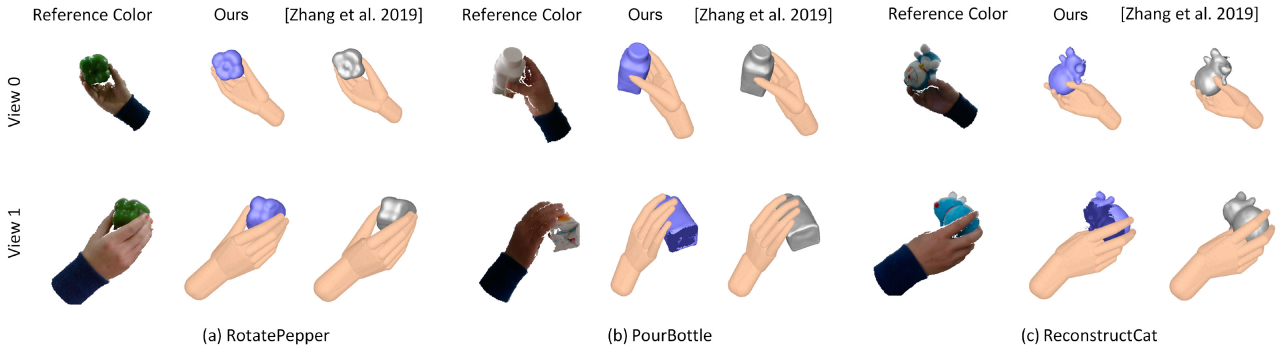

Fig. 10. Qualitative results of our system and that of Zhang et al. [2019]. Note that we only use the depth of *View0* as input.

Table 5. Comparison of the Segmentation between Our Network with Bo et al. [2020] on Our Hybrid Dataset

Network	MIoU	Runtime	Trainable Var.
Ours	0.943	20 ms	15.49M
Bo et al. [2020]	0.935	25 ms	39.91M

the shape and rigid/non-rigid motions of the manipulated object in the hand. In addition, Zhou et al. [2020] take in three-channel RGB, whereas ours uses a single channel depth.

6.3.3 Comparison with Mueller et al. [2017]. Finally, we compare our method with that of Mueller et al. [2017], which reconstructs the hand pose in interactions with an RGBD input. They design a network with a two-step architecture to predict hand keypoints and optimize the hand pose by fitting a hand skeleton to the predicted keypoints.

We first compare our network with the network of Mueller et al. [2017] on hand keypoints prediction quantitatively. As Mueller et al. [2017] did, we use their *SynthHands* as the training dataset and *EgoDexter* as the testing dataset. Specifically, we retrain the KPT network (one sub-network of our full network) on *SynthHands* and compare it with the network in the work of Mueller et al. [2017] on *EgoDexter*. The results are illustrated in Figure 12.

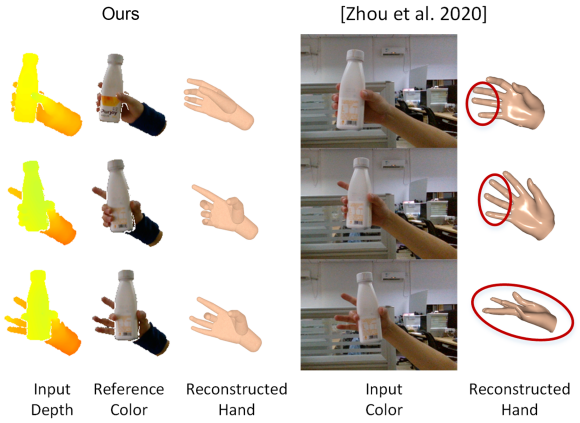

Fig. 11. Comparison with Zhou et al. [2020] on hand reconstruction.

It can be seen that our network has performance comparable to theirs (our network is better in small error threshold but slightly worse in large error threshold). It should be noted that we only use depth information, whereas Mueller et al. [2017] use RGBD information.

Then, we compare our method with that of Mueller et al. [2017] on hand pose reconstruction qualitatively. The comparison is

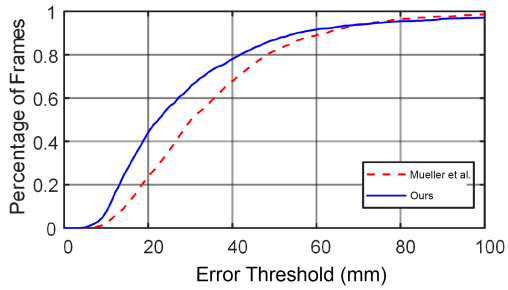

Fig. 12. Comparison of the hand keypoints prediction between our depth-based network with the two-step RGBD CNN architecture of Mueller et al. [2017] on *EgoDexter*.

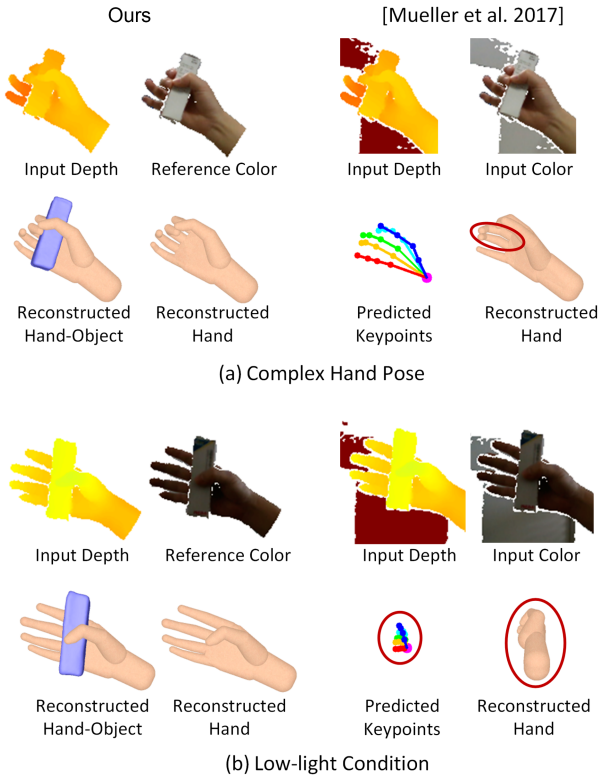

Fig. 13. Comparison of the hand pose reconstruction between our depth-based method and the RGBD-based method of Mueller et al. [2017].

conducted on two hand-object interaction sequences. To implement their work, we use their released network to predict the hand keypoints. Since they did not release their hand skeleton, we solve the hand pose by fitting the kinematic skeleton of sphere-mesh to the predicted keypoints. The results are illustrated in Figure 13 and our accompanying video. As illustrated in Figure 13(a), our method obtains more accurate hand poses than Mueller et al. [2017] because our method also uses depth data to optimize the hand pose besides the predicted hand keypoints. Since we only use depth information, our method works well in the low-light condition, whereas that of Mueller et al. [2017] fails, as shown in Figure 13(b). In addition, our method reconstructs the object in addition to tracking the hand.

6.4 More Results

We show more results in Figure 14, Figure 15, and the accompanying video to demonstrate the generalization ability of our method. It handles various object shapes (regular shapes and the object with a hole), diverse interactive motions (rigid and nonrigid motions, object moving in between fingers, putting down and picking up an object), and different users (male and female). Due to the usage of a depth camera, our system is also robust to various object textures and illumination changes.

6.5 Discussion

Our method cannot handle very tiny or very big objects. For the tiny ones, they may be fully occluded by hands, leading to failure in object tracking and fusion. However, for big objects, the hand may be severely occluded, leading to failure in hand tracking. Since our method uses depth as input, it cannot handle the object similar to the hand in geometry, like the glove in Figure 16(a). Incorporating color information may solve this problem. Our method cannot deal with thin objects either, because it is hard to reconstruct its back surface from a single view, leading to wrong results, as shown in Figure 16(b). For some very challenging interactions like manipulating a rotationally symmetric object, our method cannot avoid a slow drift for object tracking over time. In this case, a more sophisticated technique considering long-time contact continuity may be helpful. Furthermore, our method cannot handle topology changes. To handle this, fusion technologies like those of Slavcheva et al. [2017, 2018] could be involved. In addition, reconstructing the hand-object interaction from RGB images is an interesting direction [Hasson et al. 2019]. However, for RGB images, there is a large gap between synthetic and real data, leading to the difficulty in using synthetic data to solve this problem. Yet our method could provide real training data for this task.

7 CONCLUSION

We present a method for interactive hand and object reconstruction using only one depth camera. Even though it is very challenging because of severe occlusions, our method reconstructs both the hand poses and the object's rigid/non-rigid motions, and fuses the geometry model of the object in real time. Thanks to the following contributions, our method achieves comparable accuracy with the state of the art and works with various objects, different hands, and changing viewpoints. First, with the guidance of the hand keypoints and the hand-object segmentation predicted by a proposed joint network, an accurate pose for visible fingers and a plausible pose for invisible fingers are obtained. The joint learning structure saves computation cost for real-time performance without sacrificing accuracy. Second, with a novel tangential contact term to further constrain the object motion relative to the hand, the ambiguities in object reconstruction are also solved. Third, a large synthetic dataset improves the performance of the joint network on top of a real dataset. The hybrid (real plus synthetic) dataset explores the distribution of real data and covers much diversity

Fig. 14. More results of our system. For each result, we show a reference image and the corresponding reconstructed hand and object.

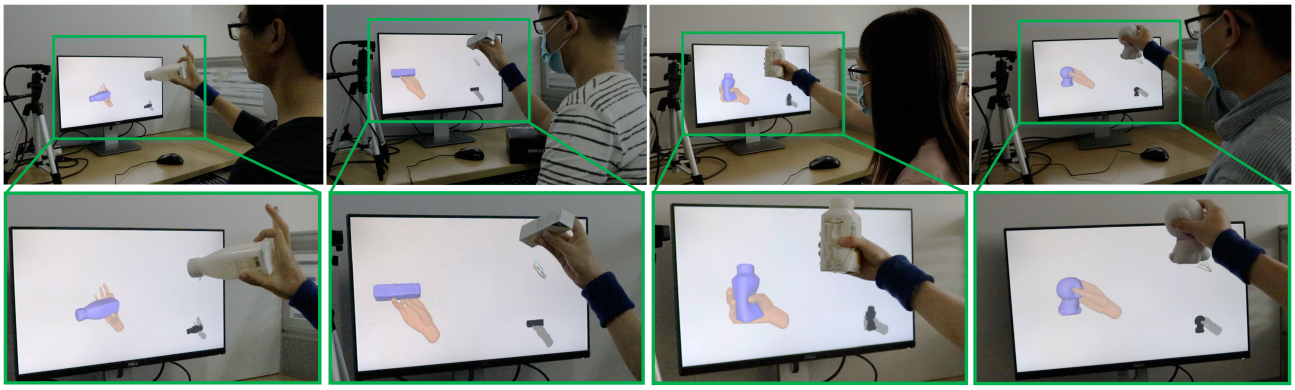

Fig. 15. More results of our system. It shows four different users are manipulating four different objects with different motions.

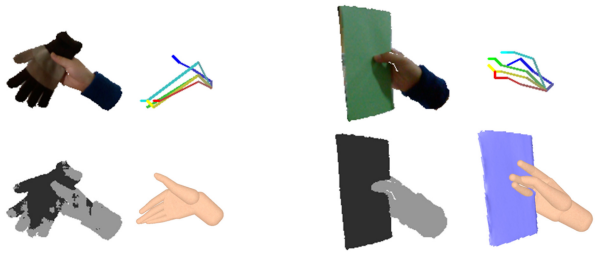

Fig. 16. Some failure cases of our system.

in hand motions, objects, and interactions, which we believe will benefit future research.

REFERENCES

Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. In *Proceedings of the European Conference on Computer Vision*. 640–653.

Zi-Hao Bo, Hao Zhang, Jun-Hai Yong, Hao Gao, and Feng Xu. 2020. DenseAttention-Seg: Segment hands from interacted objects using depth input. *Applied Soft Computing* 92 (2020), 106297.

Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2527–2530.

Woojin Cho, Gabyong Park, and Woontack Woo. 2018. Tracking an object-grabbing hand using occluded depth reconstruction. In *Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct'18)*. IEEE, Los Alamitos, CA, 232–235.

Chiho Choi, Sang Ho Yoon, Chin-Ning Chen, and Karthik Ramani. 2017. Robust hand pose estimation during the interaction with an unknown object. In *Proceedings of the IEEE International Conference on Computer Vision*. 3123–3132.

Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. 2017. Real-time geometry, albedo, and motion reconstruction using a single RGB-D camera. *ACM Transactions on Graphics* 36, 4 (2017), 1.

Henning Hamer, Konrad Schindler, Esther Koller-Meier, and Luc Van Gool. 2009. Tracking a hand manipulating an object. In *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision*. IEEE, Los Alamitos, CA, 1475–1482.

Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. 2019. Learning joint reconstruction of hands and manipulated objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 11807–11816.

Nikolaos Kyriazis and Antonis Argyros. 2013. Physically plausible 3D scene tracking: The single actor hypothesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9–16.

Nikolaos Kyriazis and Antonis Argyros. 2014. Scalable 3D tracking of multiple interacting objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3430–3437.

Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2018. GANerated hands for real-time 3D hand tracking from monocular RGB. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 49–59.

Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2017. Real-time hand tracking under occlusion from an egocentric RGB-D sensor. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 1284–1293.

Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. 2015. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 343–352.

Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. 2019. Generalized feedback loop for joint hand-object pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2019), 1898–1912.

Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. 2011. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proceedings of the 2011 International Conference on Computer Vision*. IEEE, Los Alamitos, CA, 2088–2095.

Paschalis Panteleris and Antonis Argyros. 2017. Back to RGB: 3D tracking of hands and hand-object interactions based on short-baseline stereo. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 575–584.

Paschalis Panteleris, Nikolaos Kyriazis, and Antonis A. Argyros. 2015. 3D tracking of human hands in interaction with unknown objects. In *Proceedings of the 26t British Machine Vision Conference (BMVC'15)*. 123.

Antoine Petit, Stéphane Cotin, Vincenzo Lippiello, and Bruno Siciliano. 2018. Capturing deformations of interacting non-rigid objects using RGB-D data. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'18)*. IEEE, Los Alamitos, CA, 491–497.

Grégory Rogez, James S. Supancic, and Deva Ramanan. 2015. First-person pose recognition using egocentric workspaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4325–4333.

Javier Romero, Hedvig Kjellström, and Danica Kragic. 2010. Hands in action: Real-time 3D reconstruction of hands in interaction with objects. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*. IEEE, Los Alamitos, CA, 458–463.

Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. 2002. Real-time 3D model acquisition. *ACM Transactions on Graphics* 21, 3 (2002), 438–446.

Tanner Schmidt, Katharina Hertkorn, Richard Newcombe, Zoltan Marton, Michael Suppa, and Dieter Fox. 2015. Depth-based tracking with physical constraints for robot manipulation. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA'15)*. IEEE, Los Alamitos, CA, 119–126.

Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. 2017. Killingfusion: Non-rigid 3D reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1386–1395.

Miroslava Slavcheva, Maximilian Baust, and Slobodan Ilic. 2018. SobolevFusion: 3D reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2646–2655.

Srinath Sridhar, Franziska Mueller, Michael Zollhöfer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. 2016. Real-time joint tracking of a hand manipulating an object from RGB-D input. In *Proceedings of the European Conference on Computer Vision*. 294–310.

Jonathan Taylor, Vladimir Tankovich, Danhang Tang, Cem Keskin, David Kim, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. 2017. Articulated distance fields for ultra-fast tracking of hands interacting. *ACM Transactions on Graphics* 36, 6 (2017), 1–12.

Bugra Tekin, Federica Bogo, and Marc Pollefeys. 2019. H+ O: Unified egocentric recognition of 3D hand-object poses and interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4511–4520.

Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. 2016. Sphere-meshes for real-time hand modeling and tracking. *ACM Transactions on Graphics* 35, 6 (2016), 1–11.

Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. 2014. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics* 33, 5 (2014), 1–10.

Aggeliki Tsoli and Antonis A. Argyros. 2018. Joint 3D tracking of a deformable object in interaction with a hand. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 484–500.

Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. 2016. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision* 118, 2 (2016), 172–193.

Dimitrios Tzionas and Juergen Gall. 2015. 3D object reconstruction from hand-object interactions. In *Proceedings of the IEEE International Conference on Computer Vision*. 729–737.

Mickeal Verschoor, Daniel Lobo, and Miguel A. Otaduy. 2018. Soft hand simulation for smooth and robust natural interaction. In *Proceedings of the 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR'18)*. IEEE, Los Alamitos, CA, 183–190.

Fan Wang and Kris Hauser. 2019. In-hand object scanning via RGB-D video segmentation. In *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA'19)*. IEEE, Los Alamitos, CA, 3296–3302.

Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. 2013. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics* 32, 4 (2013), 1–14.

Thibaut Weise, Bastian Leibe, and Luc Van Gool. 2008. Accurate and robust registration for in-hand modeling. In *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Los Alamitos, CA, 1–8.

Thibaut Weise, Thomas Wismer, Bastian Leibe, and Luc Van Gool. 2011. Online loop closure for real-time interactive 3D scanning. *Computer Vision and Image Understanding* 115, 5 (2011), 635–648.

Hao Zhang, Zi-Hao Bo, Jun-Hai Yong, and Feng Xu. 2019. InteractionFusion: Real-time reconstruction of hand poses and deformable objects in hand-object interactions. *ACM Transactions on Graphics* 38, 4 (2019), 1–11.

Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. 2020. Monocular real-time hand shape and motion capture using multi-modal data. arXiv:2003.09572

Received October 2020; revised December 2020; accepted February 2021