BEHAVE: Dataset and Method for Tracking Human Object Interactions (Supplementary Material)

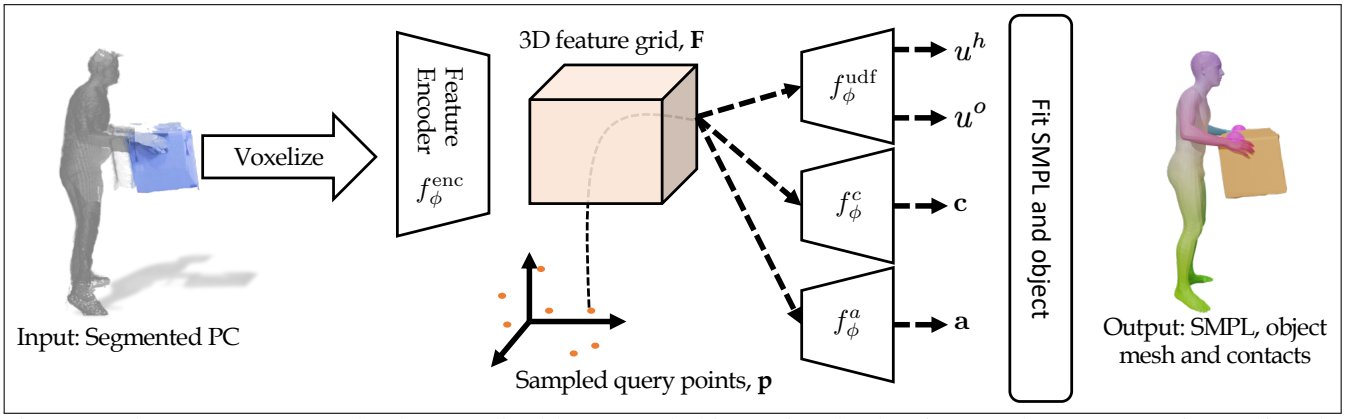

Figure 1: The BEHAVE network takes voxelized human and object point cloud as input and generates an input aligned 3D feature grid, $\mathbf{F}$. We then sample 3D query points and for each input point $\mathbf{p}$, we use the point feature $\mathbf{F}(\mathbf{p})$ to predict unsigned distance to human and object surfaces, u^h, u^o, correspondences to the SMPL model $\mathbf{c}$ and object orientation $\mathbf{a}$. We use these predictions to fit SMPL and object meshes to the input, explicitly taking the contacts between them into account.

1. Network Architecture

The input to our method is multi-view segmented point cloud of the human and the object. We voxelize it and feed it to our feature encoder f_ϕ^{enc} to obtain a grid aligned set of features. The network f_ϕ^{enc} comprises of $4 \times \{2 \times \text{Conv3D} + \text{ReLU} + \text{BN}\}$ layers. We use a stride of 2 in our convolutional layers.

We then train three separate decoders $f_\phi^{\text{udf}}, f_\phi^{\text{corr}}$ and f_ϕ^a to predict (i) unsigned distances to the human u^h, and the object u^o, surfaces; (ii) correspondences to the SMPL model $\mathbf{c}$ and (iii) object orientation $\mathbf{a}$, respectively. We use the same architecture for all our decoders, $3 \times \{\text{Conv1D} + \text{ReLU}\}$, followed by a regression layer, Conv1D.

Our network structure can be seen in Fig. 1. We will release our code for further research in this direction.

2. Data collection and annotation

In this section, we discuss in more detail about our data collection, annotation and registration process.

2.1. Data capture system setup

We use four Azure Kinect RGB-D cameras [1] placed at four corners of a square to capture human-object interactions. We use checkerboard to calibrate the relative poses between different kinects in a pairwise manner. Specifically, we capture 20 pairs of RGB-D images from two kinects and then register each color image with corresponding depth image such that they have the same resolution. We then use OpenCV to extract the checkerboard corners in the color images and obtain their 3D camera coordinates utilizing the registered depth map. Finally, we perform a Procrustes registration on these ordered 3D checkerboard corners to obtain the relative transformation between two kinects. We obtain 3 pairs of relative transformation for 4 kinects and combine them to compute the transformation under a common world coordinate.

2.2. Data preprocess and manual annotation

Both color and depth videos are captured at 30fps. Kinect cameras are synchronized through audio cables and the exact capture time of each image is saved for later processing. We extract synchronized frames and run depth-color registrations so that each depth image has the same resolution as color image. Frames are extracted at 10fps for better SMPL registration but our manual annotation is done at 1fps to maximize the diversity of annotated frames.

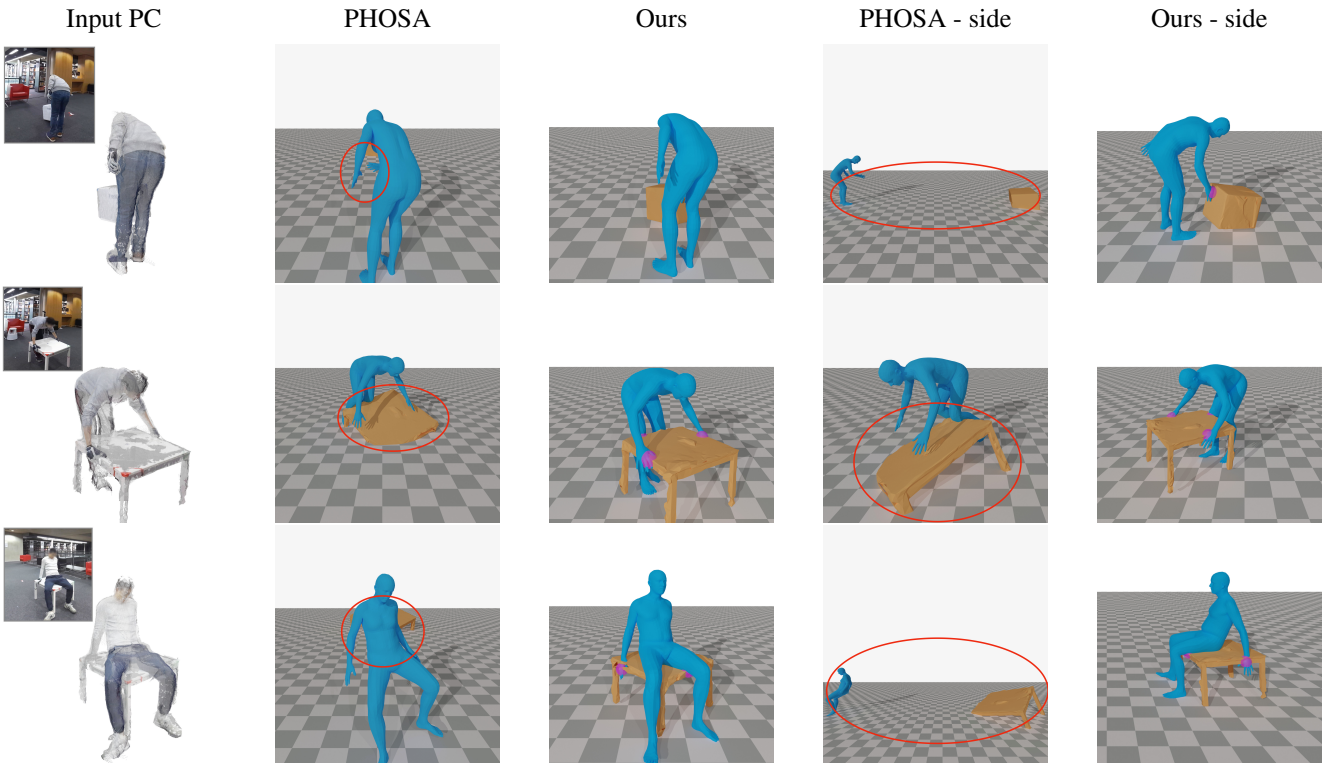

Figure 2. We compare our method with PHOSA [9] and show that our approach generates noticeably better quality results.

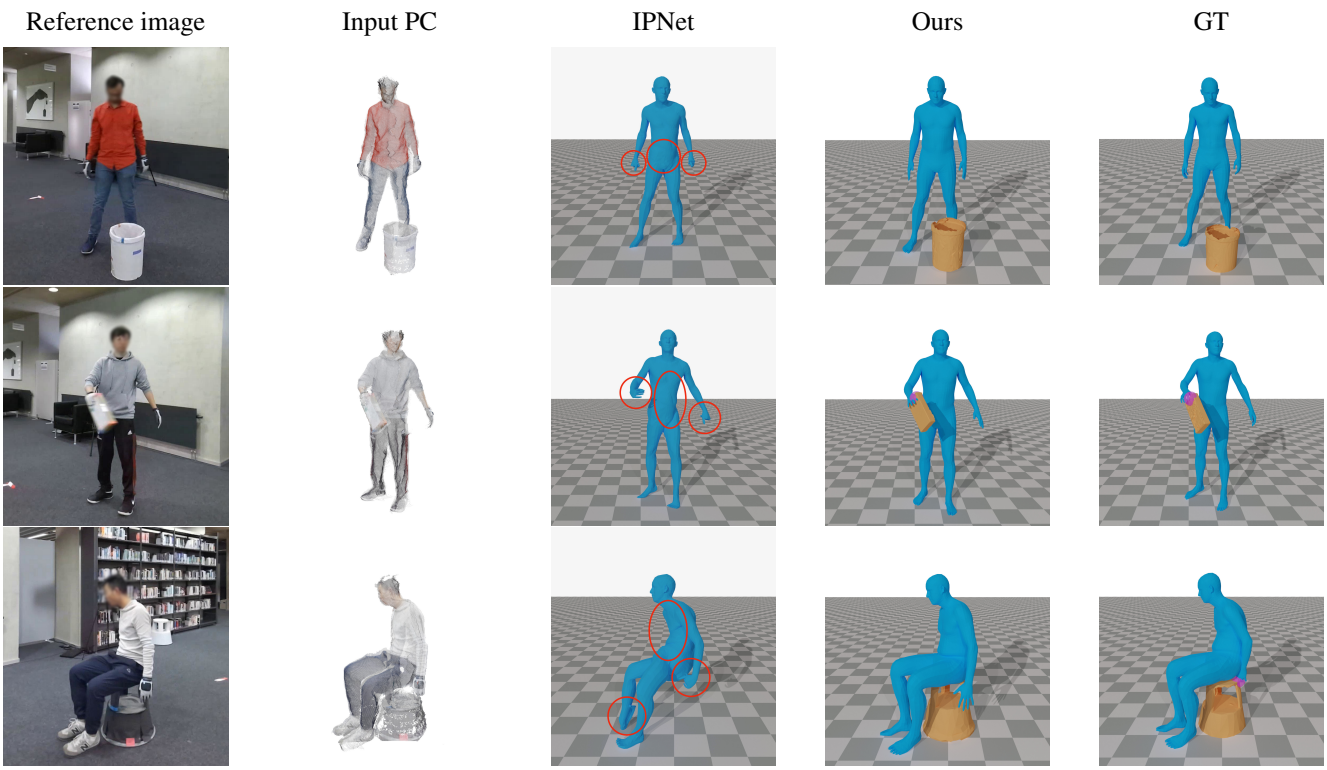

Figure 3. We compare our SMPL registration with IPNet [3] and show superior results. IPNet cannot register objects that our approach can.

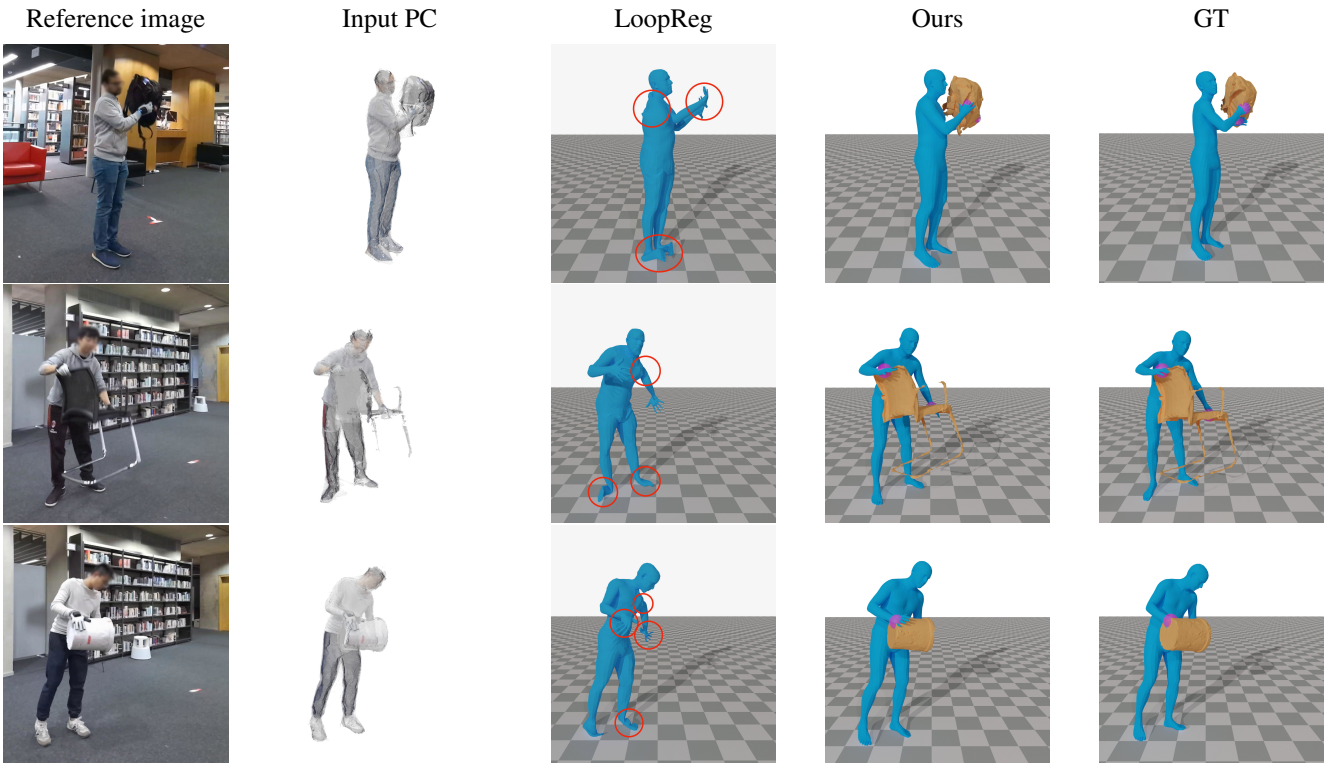

Figure 4. We compare our SMPL registration with LoopReg [4] and show superior results. LoopReg cannot register objects that our approach can.

Annotating human mask. Accurate human point cloud segmentation is required for good SMPL registration, which we achieve by obtaining accurate human masks in color images. We first run Detectron2 [8] to obtain the human masks for each image. This automatic segmentation usually works well when the person is not heavily occluded and can be easily distinguished from the background. However, fine-grained details such as hands and legs are often missing, especially when these parts have similar color with the background. We correct these minor errors through manual annotation from Amazon Mechanical Turk (AMT). Workers are asked to place 3-6 points on the erroneous segmentation regions. We use these clicks and run a recently proposed interactive segmentation method [6] to compute the corrected mask. We finally manually go over all corrected masks to filter out noisy segmentation.

Annotating object keypoints. Obtaining object segmentation mask is more expensive and the resultant point clouds are still noisy and incomplete, hence we adopt keypoints annotation for object registration. For each object, we predefine 4-8 keypoints depending on the complexity of the object geometry. We show multi-view images of the captured human-object interaction frame together with example photos of our predefined object keypoints to AMT annotators and ask them to annotate only the visible keypoints in images, see one example in Fig. 7. To ensure the annotation quality, we also manually go over all frames to filter out bad annotations.

For all our AMT annotations, we first run a test batch to select good annotators and release the full batch only to the selected annotators.

2.3. SMPL and object registration

SMPL registration. We use the corrected person segmentation mask discussed above to segment multi-view depth maps and obtain human point clouds. We initialize the SMPL pose from FrankMocap [5] and use the multi-step registration method from [2] to fit SMPL to the segmented human point clouds.

Object registration. Given template object meshes and annotated 2D keypoints from multi-views, we register the mesh to the images by optimizing the reprojection loss, similar to the method used in Pix3D registration [7]. To favor convergence, we downsample the original scans to around 2000 faces and initialize the translation parameter as the center of human point clouds.

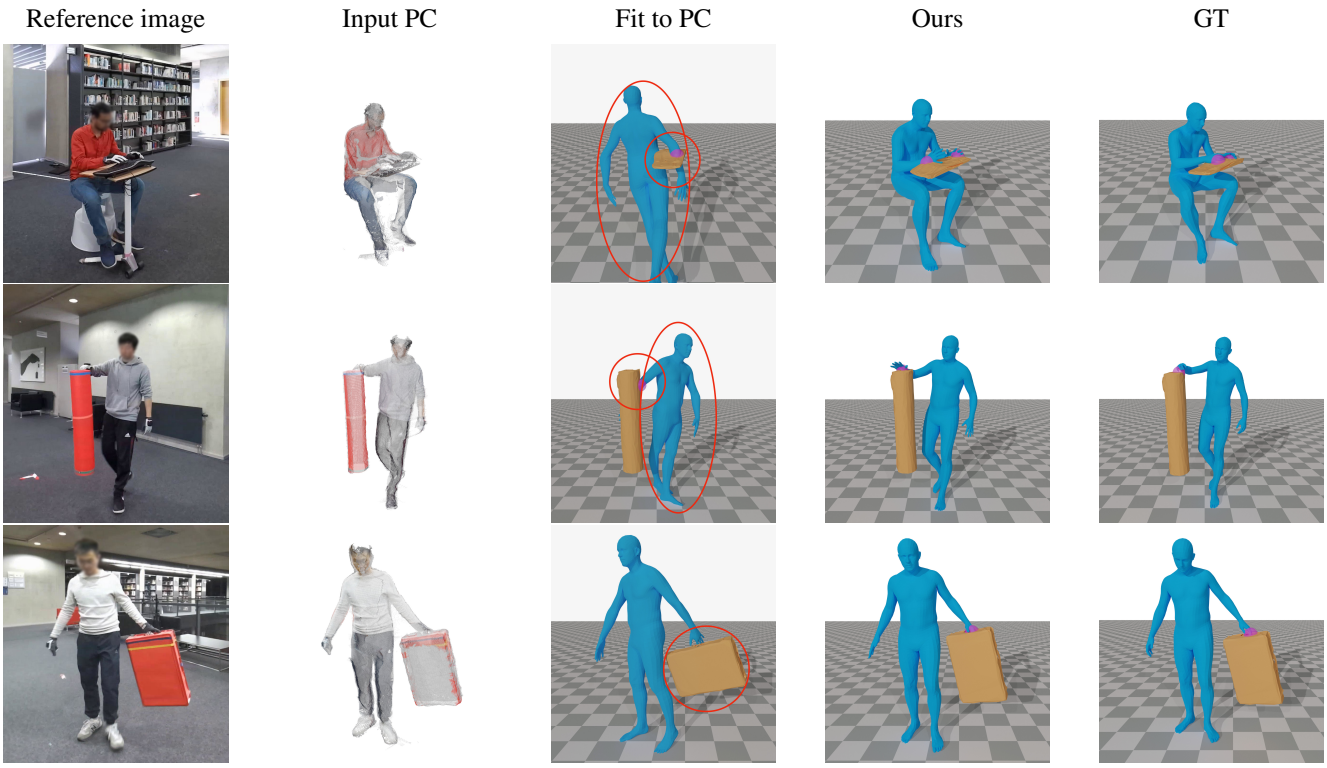

Figure 5. We show that registering SMPL and object meshes directly to input point cloud results in inaccurate fitting. Our neural predictions corresponding to human and object unsigned distance fields, SMPL correspondence field and object orientation field are key to good fitting.

3. Comparison with PHOSA [9]

We provide more qualitative comparisons with PHOSA and show that our approach easily outperforms it. PHOSA uses pre-defined fixed contact locations and heuristic based instance specific optimization. Our method on the other hand learns contact prediction and model fitting from the data, leading to superior performance and improved scalability. We show qualitative results for the same in Fig. 2

4. Comparison with IPNet [3]

IPNet can only register SMPL and not objects, but we still find its idea of combining implicit reconstruction and parametric model fitting interesting. As discussed in Sec. 5.3 (main paper), our formulation for fitting SMPL to the input is superior than IPNet as it alleviates the requirements for watertight surfaces, marching cubes and high number of query points. Our method is the first approach that can directly fit SMPL to the distance field prediction without explicitly predicting the 3D mesh. We show more qualitative comparisons in Fig. 3 and show that we outperform IPNet (trained on our data) on the task of SMPL fitting to the input point cloud.

5. Comparison with LoopReg [4]

Although LoopReg cannot fit objects, we still find their idea of predicting correspondences and then fitting the SMPL model useful. As discussed in Sec. 5.3 (main paper) our formulation is superior to LoopReg as it can handle noisy input and use off-surface input points for registration where as Loopreg can use points only on the input surface. This allows us to handle incomplete point clouds, which is common in the case of heavy occlusions during human-object interactions. We show more qualitative comparisons in Fig. 4 and show that we outperform LoopReg (trained on our data) on the task of human registration with noisy and incomplete input.

6. Comparison with direct fitting to multi-view input

Fitting SMPL and object meshes directly to the input point cloud is another intuitive baseline. This experiment highlights the importance of our network predictions and Fig. 5 shows that without our predictions the optimization gets stuck in local minima leading to poor fitting.

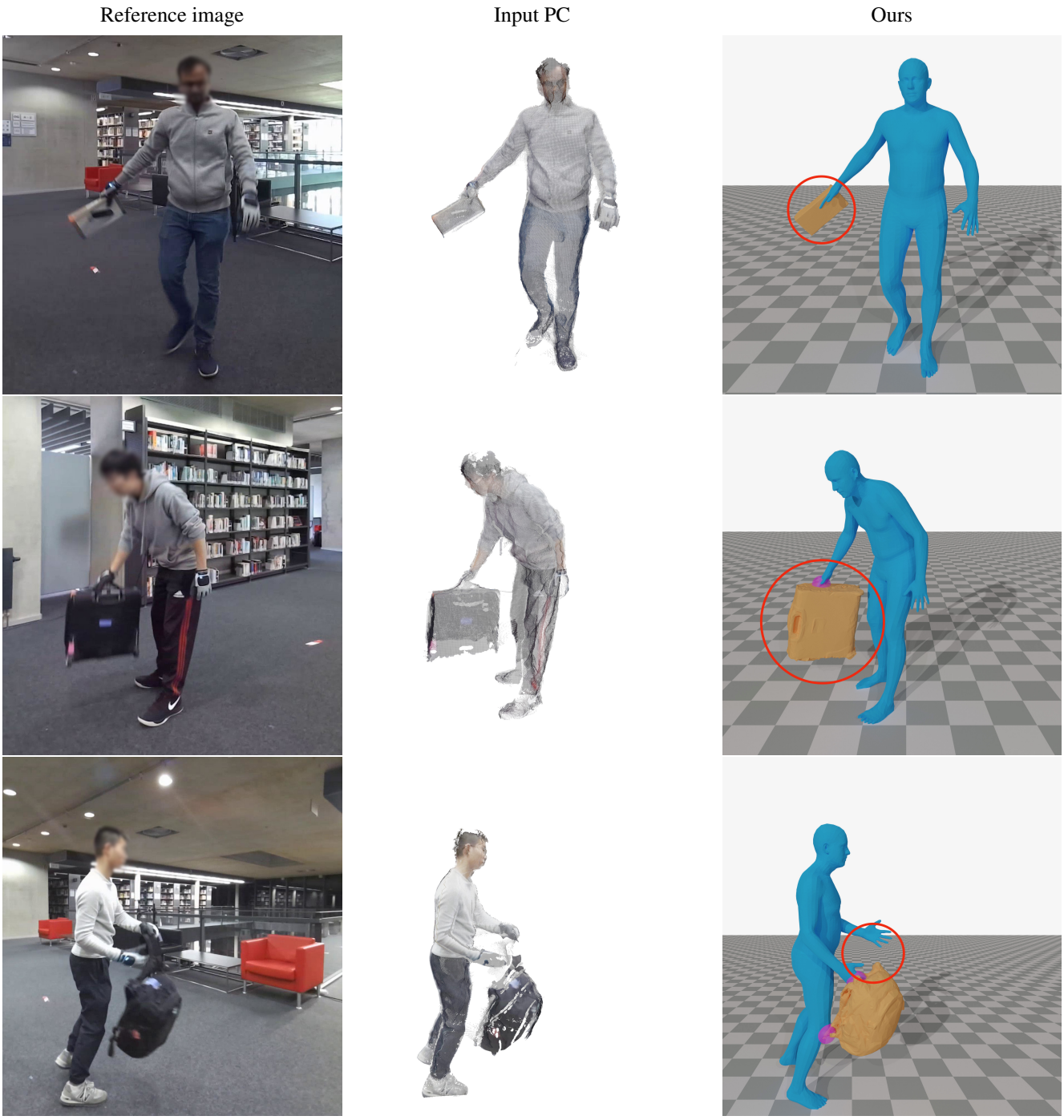

Figure 6. We show some limitations of the proposed method. (Top) The input point cloud from the Kinect is often noisy which makes tracking fine grained details like hands challenging. (Mid) Our method also struggles to fit symmetric objects such as a square suitcase accurately. It can be seen that although the object mesh fits the point cloud quite well, the orientation is wrong. (Bottom) A lot of real world objects are non-rigid, making their tracking difficult with rigid templates. It can be seen here that the deformed strap of the backpack is not tracked.

7. Limitations and Future Works

We discuss the limitations of our current method in Fig. 6. We find that network struggles sometimes to predict

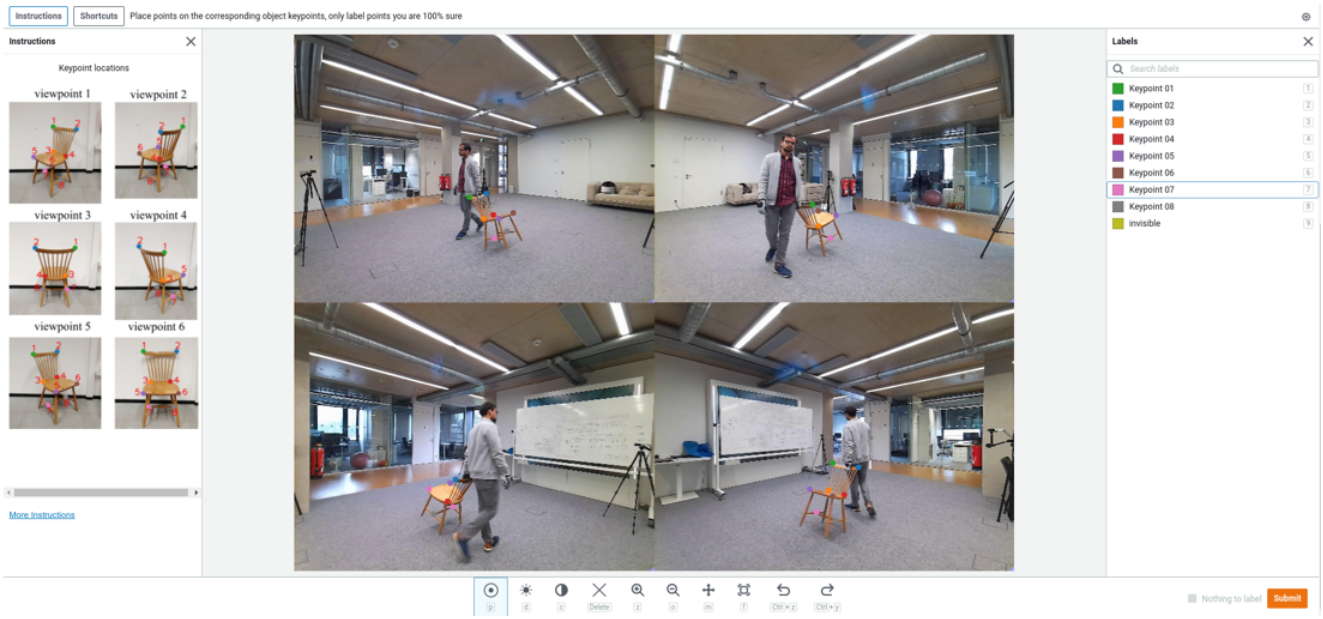

Figure 7. Example frame of our object keypoint annotation. Our predefined object keypoints are shown in the left, the frame to be annotated is shown in the middle. Annotators are asked to place keypoint labels (right panel) at their corresponding locations in the image.

correct orientation of objects that are symmetric such as a square suitcase. We also observe that since Kinect data is noisy, we cannot model fine grained hand interactions, this results in interpenetrations between the hand and the object and sometimes unrealistic grasps. More interesting limitation arises from the fact that we assume the objects to be rigid which is not the case in reality. Objects like backpacks when grabbed from the straps are not accurately registered as the deformations in this case are non-rigid. All these are challenging scenarios with no straightforward solution and these directions warrant further research.

References

[1] https://azure.microsoft.com/en-us/services/kinect-dk/. 1

[2] Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single RGB camera. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[3] Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. Combining implicit function learning and parametric models for 3d human reconstruction. In *European Conference on Computer Vision (ECCV)*. Springer, August 2020. 2, 4

[4] Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. Loopreg: Self-supervised learning of implicit surface correspondences, pose and shape for 3d human mesh registration. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020. 3, 4

[5] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In *IEEE International Conference on Computer Vision Workshops*, 2021. 3

[6] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8623–8632, 2020. 3

[7] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[8] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 3

[9] Jason Y. Zhang, Sam Pepose, Hanbyul Joo, Deva Ramanan, Jitendra Malik, and Angjoo Kanazawa. Perceiving 3d human-object spatial arrangements from a single image in the wild. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 4