

DexYCB: A Benchmark for Capturing Hand Grasping of Objects

Yu-Wei Chao¹ Wei Yang¹ Yu Xiang¹ Pavlo Molchanov¹ Ankur Handa¹ Jonathan Tremblay¹
 Yashraj S. Narang¹ Karl Van Wyk¹ Umar Iqbal¹ Stan Birchfield¹ Jan Kautz¹ Dieter Fox^{1,2}

¹NVIDIA, ²University of Washington

{ychao, weiy, yux, pmolchanov, ahanda, jtremblay, ynarang, kvanwyk, uiqbal, sbirchfield, jkautz, dieterf}@nvidia.com



Figure 1: Two captures (left and right) from the DexYCB dataset. In each case, the top row shows color images simultaneously captured from three views, while the bottom row shows the ground-truth 3D object and hand pose rendered on the darkened captured images.

Abstract

We introduce DexYCB, a new dataset for capturing hand grasping of objects. We first compare DexYCB with a related one through cross-dataset evaluation. We then present a thorough benchmark of state-of-the-art approaches on three relevant tasks: 2D object and keypoint detection, 6D object pose estimation, and 3D hand pose estimation. Finally, we evaluate a new robotics-relevant task: generating safe robot grasps in human-to-robot object handover.¹

1. Introduction

3D object pose estimation and 3D hand pose estimation are two important yet unsolved vision problems. Traditionally, these two problems have been addressed separately, yet in many critical applications, we need both capabilities working together [36, 6, 13]. For example, in robotics, a reliable motion capture for hand manipulation of objects is crucial for both learning from human demonstration [12] and fluent and safe human-robot interaction [46].

State-of-the-art approaches for both 3D object pose [37, 20, 34, 27, 40, 26, 19] and 3D hand pose estimation [49, 23, 18, 2, 9, 14, 31] rely on deep learning and thus require large datasets with labeled hand or object poses for training.

¹Dataset and code available at <https://dex-ycb.github.io>.

Many datasets [48, 49, 16, 17] have been introduced in both domains and have facilitated progress on these two problems in parallel. However, since they were introduced for either task separately, many of them do not contain interaction of hands and objects, i.e., static objects without humans in the scene, or bare hands without interacting with objects. In the presence of interactions, the challenge of solving the two tasks together not only doubles but multiplies, due to the motion of objects and mutual occlusions incurred by the interaction. Networks trained on either of the datasets will thus not generalize well to interaction scenarios.

Creating a dataset with accurate 3D pose of hands and objects is also challenging for the same reasons. As a result, prior works have attempted to capture accurate hand motion either with specialized gloves [10], magnetic sensors [48, 8], or marker-based mocap systems [3, 35]. While they can achieve unparalleled accuracy, the introduction of hand-attached devices may be intrusive and thus bias the naturalness of hand motion. It also changes the appearance of hands and thus may cause issues with generalization.

Due to the challenge of acquiring real 3D poses, there has been an increasing interest in using synthetic datasets to train pose estimation models. The success has been notable on object pose estimation. Using 3D scanned object models and photorealistic rendering, prior work [16, 34, 38, 5, 17] has generated synthetic scenes of objects with high fidelity in appearance. Their models trained only on synthetic data

can thus translate to real images. Nonetheless, synthesizing hand-object interactions remains challenging. One problem is to synthesize realistic grasp poses for generic objects [4]. Furthermore, synthesizing natural looking human motions is still an active research area in graphics.

In this paper, we focus on marker-less data collection of real hand interaction with objects. We take inspiration from recent work [11] and build a multi-camera setup that records interactions synchronously from multiple views. Compared to the recent work, we instrument the setup with more cameras and configure them to capture a larger workspace that allows our human subjects to interact freely with objects. In addition, our pose labeling process utilizes human annotation rather than automatic labeling. We crowdsource the annotation so that we can efficiently scale up the data labeling process. Given the setup, we construct a large-scale dataset that captures a simple yet ubiquitous task: grasping objects from a table. The dataset, DexYCB, consists of 582K RGB-D frames over 1,000 sequences of 10 subjects grasping 20 different objects from 8 views (Fig. 1).

Our contributions are threefold. First, we introduce a new dataset for capturing hand grasping of objects. We empirically demonstrate the strength of our dataset over a related one through cross-dataset evaluation. Second, we provide in-depth analysis of current approaches thoroughly on three relevant tasks: 2D object and keypoint detection, 6D object pose estimation, and 3D hand pose estimation. To the best of our knowledge, our dataset is the first that allows joint evaluation of these three tasks. Finally, we demonstrate the importance of joint hand and object pose estimation on a new robotics relevant task: generating safe robot grasps for human-to-robot object handover.

2. Constructing DexYCB

2.1. Hardware Setup

In order to construct the dataset, we built a multi-camera setup for capturing human hands interacting with objects. A key design choice was to enable a sizable capture space, where a human subject can freely interact and perform tasks with multiple objects. Our multi-camera setup is shown in Fig. 2. We use 8 RGB-D cameras (RealSense D415) and mount them such that collectively they can capture a tabletop workspace with minimal blind spots. The cameras are extrinsically calibrated and temporally synchronized. For data collection, we stream and record all 8 views together at 30 fps with both color and depth of resolution 640×480 .

2.2. Data Collection and Annotation

Given the setup, we record videos of hands grasping objects. We use 20 objects from the YCB-Video dataset [44], and record multiple trials from 10 subjects. For each trial, we select a target object with 2 to 4 other objects and place

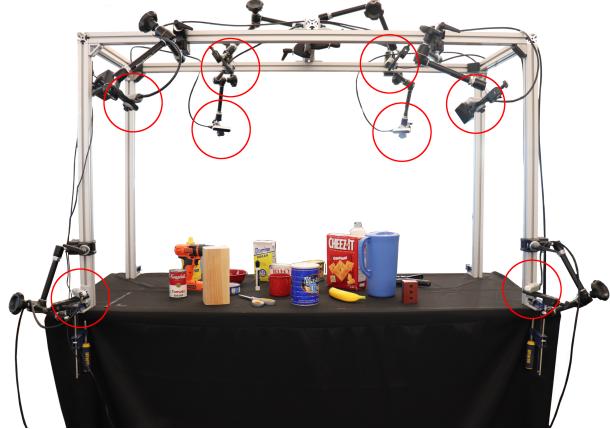


Figure 2: Our setup with 8 RGB-D cameras (red circle).

them on the table. We ask the subject to start from a relaxed pose, pick up the target object, and hold it in the air. We also ask some subjects to pretend to hand over the object to someone across from them. We record for 3 seconds, which is sufficient to contain the full course of action. For each target object, we repeat the trial 5 times, each time with a random set of accompanied objects and placement. We ask the subject to perform the pick-up with the right hand in the first two trials, and with the left hand in the third and fourth trials. In the fifth trial, we randomize the choice. We rotate the target among all 20 objects. This gives us 100 trials per subject, and 1,000 trials in total for all subjects.

To acquire accurate ground-truth 3D pose for hands and objects, our approach (detailed in Sec. 2.3) relies on 2D keypoint annotations for hands and objects in each view. To ensure accuracy, we label the required keypoints in RGB sequences fully through human annotation. Our annotation tool is based on VATIC [39] for efficient annotation of videos. We set up annotation tasks on the Amazon Mechanical Turk (MTurk) and label every view in all the sequences.

For hands, we adopt 21 pre-defined hand joints as our keypoints (3 joints plus 1 tip for each finger and the wrist). We explicitly ask the annotators to label and track these joints throughout a given video sequence. The annotators are also asked to mark a keypoint as invisible in a given frame when it is occluded.

Pre-defining keypoints exhaustively for every object would be laborious and does not scale as the number of objects increases. Our approach (Sec. 2.3) explicitly addresses this issue by allowing user-defined keypoints. Specifically, given a video sequence in a particular view, we first ask the annotator to find 2 distinctive landmark points that are easily identified and trackable on a designated object, and we ask them to label and track these points throughout the sequence. We explicitly ask the annotators to find keypoints that are visible most of the time, and mark a keypoint as invisible whenever it is occluded.

2.3. Solving 3D Hand and Object Pose

To represent 3D hand pose, we use the popular MANO hand model [28]. The model represents a right or left hand with a deformable triangular mesh of 778 vertices. The mesh is parameterized by two low-dimensional embeddings (θ, β) , where $\theta \in \mathbb{R}^{51}$ accounts for variations in pose (i.e. articulation) and $\beta \in \mathbb{R}^{10}$ in shape. We use the version from [14], which implements MANO as a differentiable layer in PyTorch that maps (θ, β) to the mesh together with the 3D positions of 21 hand joints defined in the keypoint annotation. We pre-calibrate the hand shape β for each subject and fix it throughout each subject's sequences.

Since our objects from YCB-Video [44] also come with texture-mapped 3D mesh models, we use the standard 6D pose representation [16, 17] for 3D object pose. The pose of each object is represented by a matrix $T \in \mathbb{R}^{3 \times 4}$ composed of a 3D rotation matrix and a 3D translation vector.

To solve for hand and object pose, we formulate an optimization problem similar to [50, 11] by leveraging depth and keypoint annotations from all views and multi-view geometry. For a given sequence with N_H hands and N_O objects, we denote the overall pose at a given time frame by $P = (P_H, P_O)$, where $P_H = \{\theta_h\}_{h=1}^{N_H}$ and $P_O = \{T_o\}_{o=1}^{N_O}$. We define the pose in world coordinates where we know the extrinsics of each camera. Then at each time frame, we solve the pose by minimizing the following energy function:

$$E(P) = E_{\text{depth}}(P) + E_{\text{kpt}}(P) + E_{\text{reg}}(P). \quad (1)$$

Depth The depth term E_{depth} measures how well the models given poses explain the observed depth data. Let $\{d_i \in \mathbb{R}^3\}_{i=1}^{N_D}$ be the total point cloud merged from all views after transforming to the world coordinates, with N_D denoting the number of points. Given a pose parameter, we denote the collection of all hand and object meshes as $\mathcal{M}(P) = (\{\mathcal{M}_h(\theta_h)\}, \{\mathcal{M}_o(T_o)\})$. We define the depth term as

$$E_{\text{depth}}(P) = \frac{1}{N_D} \sum_{i=1}^{N_D} |\text{SDF}(d_i, \mathcal{M}(P))|^2, \quad (2)$$

where $\text{SDF}(\cdot)$ calculates the signed distance value of a 3D point from a triangular mesh in mm. While E_{depth} is differentiable, calculating E_{depth} and also the gradients is computationally expensive for large point clouds and meshes with a huge number of vertices. Therefore, we use an efficient point-parallel GPU implementation for it.

Keypoint The keypoint term E_{kpt} measures the reprojection error of the keypoints on the models with the annotated keypoints, and can be decomposed by hand and object:

$$E_{\text{kpt}}(P) = E_{\text{kpt}}(P_H) + E_{\text{kpt}}(P_O). \quad (3)$$

For hands, let $J_{h,j}$ be the 3D position of joint j of hand h in the world coordinates, $p_{h,j}^c$ be the annotation of the

same joint in the image coordinates of view c , and $\gamma_{h,j}^c$ be its visibility indicator. The energy term is defined as

$$E_{\text{kpt}}(P_H) = \frac{1}{\sum \gamma_{h,j}^c} \sum_{c=1}^{N_C} \sum_{h=1}^{N_H} \sum_{j=1}^{N_J} \gamma_{h,j}^c \|\text{proj}^c(J_{h,j}) - p_{h,j}^c\|_2^2, \quad (4)$$

where $\text{proj}^c(\cdot)$ returns the projection of a 3D point onto the image plane of view c , and $N_C = 8$ and $N_J = 21$.

For objects, recall that we did not pre-define keypoints for annotation, but rather asked annotators to select distinctive points to track. Here, we assume an accurate initial pose is given at the first frame where an object's keypoint is labeled visible. We then map the selected keypoint to a vertex on the object's 3D model by back-projecting the keypoint's position onto the object's visible surface. We fix that mapping afterwards. Let $K_{o,k}^c$ be the 3D position of the selected keypoint k of object o in view c in world coordinates. Similar to Eq. (4), with $N_K = 2$, the energy term is

$$E_{\text{kpt}}(P_O) = \frac{1}{\sum \gamma_{o,k}^c} \sum_{c=1}^{N_C} \sum_{o=1}^{N_O} \sum_{k=1}^{N_K} \gamma_{o,k}^c \|\text{proj}^c(K_{o,k}^c) - p_{o,k}^c\|_2^2. \quad (5)$$

To ensure an accurate initial pose for keypoint mapping, we initialize the pose in each time frame with the solved pose from the last time frame. We initialize the pose in the first frame by running PoseCNN [44] on each view and select an accurate pose for each object manually.

Regularization Following [24, 13], we add an ℓ_2 regularization to the low-dimensional pose embedding of MANO to avoid irregular articulation of hands:

$$E_{\text{reg}}(P) = \frac{1}{N_H} \sum_{h=1}^{N_H} \|\theta_h\|_2^2. \quad (6)$$

To minimize Eq. (1), we use the Adam optimizer with a learning rate of 0.01. For each time frame, we initialize the pose P with the solved pose from the last time frame and run the optimizer for 100 iterations.

3. Related Datasets

3.1. 6D Object Pose

Most datasets address instance-level 6D object pose estimation, where 3D models are given a priori. The recent BOP challenge [16, 17] has curated a decent line-up of these datasets which the participants have to evaluate on. Yet objects are mostly static in these datasets without human interactions. A recent dataset [41] was introduced for category-level 6D pose estimation, but the scenes are also static.

3.2. 3D Hand Pose

We present a summary of related 3D hand pose datasets in Tab. 1. Some address pose estimation with depth only,

dataset	visual modality	real image	marker-less	hand-hand int	hand-obj int	3D obj	3D hand	3D pose shape	resolution	#frames	#sub	#obj	#views	motion	#seq	dynamic grasp	label
BigHand2.2M [48]	depth	✓	✗	✗	✗	✗	—	✗	640×480	2,200K	10	—	1	✓	—	magnetic sensor	
SynthHands [25]	RGB-D	✗	—	✗	✓	✗	✗	✗	640×480	220K	—	7	5	✗	—	synthetic	
Rendered Hand Pose [49]	RGB-D	✗	—	✗	✗	—	✗	✗	320×320	44K	20	—	1	✗	—	synthetic	
GANerated Hands [23]	RGB	✗	—	✗	✓	✗	✗	✗	256×256	331K	—	—	1	✗	—	synthetic	
ObMan [14]	RGB-D	✗	—	✗	✓	✓	✓	✓	256×256	154K	20	3K	1	✗	—	synthetic	
FPHA [8]	RGB-D	✓	✗	✗	✓	✓	✓	✗	1920×1080	105K	6	4	1	✓	1,175	✓	magnetic sensor
ContactPose [3]	RGB-D	✓	✗	✗	✓	✓	✓	✓	960×540	2,991K	50	25	3	✓	2,303	✗	mocap
GRAB [35]	—	—	✗	✗	✓	✓	✓	✓	—	1,624K	10	51	—	✓	1,335	✓	+ thermal mocap
Mueller et al. [24]	depth	✗	—	✓	✗	—	✗	✗	640×480	80K	5	—	4	✓	11	—	synthetic
InterHand2.6M [22]	RGB	✓	✓	✓	✓	✗	—	✓	512×334	2,590K	27	—	>80	✓	—	—	semi-auto
Dexter+Object [32]	RGB-D	✓	✓	✗	✓	✓	✓	✗	640×480	3K	2	2	1	✓	6	✓	manual
Simon et al. [30]	RGB	✓	✓	✓	✓	✓	✗	✗	1920×1080	15K	—	—	31	✓	—	✓	automatic
EgoDexter [25]	RGB-D	✓	✓	✗	✓	✓	✗	✗	640×480	3K	4	—	1	✓	4	✓	manual
FreiHAND [50]	RGB	✓	✓	✗	✓	✓	✗	✓	224×224	37K	32	27	8	✗	—	—	semi-auto
HO-3D [11]	RGB-D	✓	✓	✗	✓	✓	✓	✓	640×480	78K	10	10	1–5	✓	27	✓	automatic
DexYCB (ours)	RGB-D	✓	✓	✗	✓	✓	✓	✓	640×480	582K	10	20	8	✓	1,000	✓	manual

Table 1: Comparison of DexYCB with existing 3D hand pose datasets.

e.g., BigHand2.2M [48]. These datasets can be large but lack color images and capture only bare hands without interactions. Some others address hand-hand interactions, e.g., Mueller et al. [24] and InterHand2.6M [22]. While not focused on interaction with objects, their datasets address another challenging scenario and is orthogonal to our work. Below we review datasets with hand-object interactions.

Synthetic Synthetic data has been increasingly used for hand pose estimation [25, 49, 23, 14]. A common downside is the gap to real images on appearance. To bridge this gap, GANerated Hands [23] was introduced by translating synthetic images to real via GANs. Nonetheless, other challenges remain. One is to synthesize realistic grasp pose for objects. ObMan [14] was introduced to address this challenge using heuristic metrics. Yet besides pose, it is also challenging to synthesize realistic motion. Consequently, these datasets only offer static images but not videos. Our dataset captures real videos with real grasp pose and motion. Furthermore, our motion data from real can help bootstrapping synthetic data generation.

Marker-based FPHA [8] captured hand interaction with objects in first person view by attaching magnetic sensors to the hands. This offers a flexible and portable solution, but the attached sensors may hinder natural hand motion and also bias the hand appearance. ContactPose [3] captured grasp poses by tracking objects with mocap markers and recovering hand pose from thermal imagery of the objects surface. While the dataset is large in scale, the thermal based approach can only capturing rigid grasp poses but not motions. GRAB [35] captured full body motion together with hand-object interactions using marker-based mocap. It provides high-fidelity capture of interactions but does not

come with any visual modalities. Our dataset is marker-less, captures dynamic grasp motions, and provides RGB-D sequences in multiple views.

Marker-less Our DexYCB dataset falls in this category. The challenge is to acquire accurate 3D pose. Some datasets like Dexter+Object [32] and EgoDexter [23] rely on manual annotation and are thus limited in scale. To acquire 3D pose at scale others rely on automatic or semi-automatic approaches [30, 50]. While these datasets capture hand-object interactions, they do not offer 3D pose for objects.

Most similar to ours is the recent HO-3D dataset [11]. HO-3D also captures hands interacting with objects from multiple views and provides both 3D hand and object pose. Nonetheless, the two datasets differ both quantitatively and qualitatively. Quantitatively (Tab. 1), DexYCB captures interactions with more objects (20 versus 10), from more views (8 versus 1 to 5), and is one order of magnitude larger in terms of number of frames (582K versus 78K)² and number of sequences (1,000 versus 27).³ Qualitatively (Fig. 8), we highlight three differences. First, DexYCB captures full grasping processes (i.e. from hand approaching, opening fingers, contact, to holding the object stably) in short segments, whereas HO-3D captures longer sequences with object in hand most of the time. Among the 27 sequences from HO-3D, we found 17 with hand always rigidly attached to the object, only rotating the object from the wrist with no finger articulation. Second, DexYCB captures a full tabletop workspace with 3D pose of all the objects on the table, whereas in HO-3D one object is held close to the camera each time and only the held object is labeled with pose. Finally, regarding annotation, DexYCB leverages keypoints

²We count the number of frames over all views.

³We count the same motion across different views as one sequence.



Figure 3: Qualitative comparison of HO-3D [11] (top) and DexYCB (bottom). DexYCB captures full grasping processes in a tabletop workspace with 3D pose of all the objects on the table.

fully labeled by humans through crowdsourcing, while HO-3D relies on a fully automatic labeling pipeline.

4. Cross-Dataset Evaluation

To further assess the merit of our DexYCB dataset, we perform cross-dataset evaluation following prior work [50, 29, 47]. We focus specifically on HO-3D [11] due to its relevance, and evaluate generalizability between HO-3D and DexYCB on single-image 3D hand pose estimation. For DexYCB, we generate a train/val/test split following our benchmark setup (“S0” in Sec. 5.1). For HO-3D, we select 6 out of its 55 training sequences for test. We use Spurr et al. [31] (winner of the HAND 2019 Challenge) as our method, and train the model separately on three training sets: the training set of HO-3D, the training set of DexYCB, and the combined training set from both datasets. Finally, we evaluate the three trained models separately on the test set of HO-3D and DexYCB.

Tab. 2 shows the results in mean per joint position error (MPJPE) reported on two different alignment methods (details in Sec. 5.4). We experiment with two different backbones: ResNet50 and HRNet32 [33]. Unsurprisingly, when training on a single dataset, the model generalizes better to the respective test set. The error increases when tested on the other dataset. When we evaluate the DexYCB trained model on HO-3D, we observe a consistent increase from $1.4 \times$ to $1.9 \times$ (e.g., for ResNet50, from 18.05 to 31.76 mm on root-relative). However, when we evaluate the HO-3D trained model on DexYCB, we observe a consistent yet more significant increase, from $3.4 \times$ to $3.7 \times$ (e.g., for ResNet50, from 12.97 to 48.30 mm on root-relative). This suggests that models trained on DexYCB generalize better than on HO-3D. Furthermore, when we train on the combined training set and evaluate on HO-3D, we can further reduce the error from only training on HO-3D (e.g., from 18.05 to 15.79 mm). However, when tested on DexYCB, the error rather rises compared to training only on DexYCB (e.g., from 12.97 to 13.36 mm). We conclude that DexYCB complements HO-3D better than vice versa.

train test	HO-3D [11]	DexYCB	HO-3D [11] + DexYCB
HO-3D	18.05 / 10.66	31.76 / 15.23	15.79 / 9.51
DexYCB	48.30 / 24.23	12.97 / 7.18	13.36 / 7.27
HO-3D	17.46 / 10.44	33.11 / 15.51	15.89 / 9.00
DexYCB	46.38 / 23.94	12.39 / 6.79	12.48 / 6.87

Table 2: Cross-dataset evaluation with HO-3D [11] on 3D hand pose estimation. Results are in MPJPE (mm) (root-relative / Procrustes aligned). Top: [31] + ResNet50. Bottom: [31] + HRNet32.

5. Benchmarking Representative Approaches

We benchmark three tasks on our DexYCB dataset: 2D object and keypoint detection, 6D object pose estimation, and 3D hand pose estimation. For each task we select representative approaches and analyze their performance.

5.1. Evaluation Setup

To evaluate different scenarios, we generate train/val/test splits in four different ways (referred to as “setup”):

- **S0 (default).** The train split contains all 10 subjects, all 8 camera views, and all 20 grasped objects. Only the sequences are not shared with the val/test split.
- **S1 (unseen subjects).** The dataset is split by subjects (train/val/test: 7/1/2).
- **S2 (unseen views).** The dataset is split by camera views (train/val/test: 6/1/1).
- **S3 (unseen grasping).** The dataset is split by grasped objects (train/val/test: 15/2/3). Objects being grasped in the test split are never being grasped in the train/val split, but may appear static on the table. This way the training set still contain examples of each object.

5.2. 2D Object and Keypoint Detection

We evaluate object and keypoint detection using the COCO evaluation protocol [21]. For object detection, we consider 20 object classes and 1 hand class. We collect ground truths by rendering a segmentation mask for each instance in each camera view. For keypoints, we consider 21 hand joints and collect the ground truths by reprojecting each 3D joint to each camera image.

We benchmark two representative approaches: Mask R-CNN (Detectron2) [15, 43] and SOLOv2 [42]. Mask R-CNN has been the de facto for object and keypoint detection and SOLOv2 is a state-of-the-art on COCO instance segmentation. For both we use a ResNet50-FPN backbone pre-trained on ImageNet and finetune on DexYCB.

Tab. 3 shows results for object detection in average precision (AP). First, Mask R-CNN and SOLOv2 perform similarly in mean average precision (mAP). Mask R-CNN has a slight edge on bounding box (e.g., 75.76 versus 75.13 mAP on S0) while SOLOv2 has a slight edge on segmentation (e.g., 71.56 versus 69.58 mAP on S0). This is because

	S0 (default)				S1 (unseen subjects)				S2 (unseen views)				S3 (unseen grasping)			
	Mask R-CNN		SOLOv2		Mask R-CNN		SOLOv2		Mask R-CNN		SOLOv2		Mask R-CNN		SOLOv2	
	bbox	segm	bbox	segm	bbox	segm	bbox	segm	bbox	segm	bbox	segm	bbox	segm	bbox	segm
002_master_chef_can	83.87	82.68	84.90	85.20	82.41	80.69	82.74	81.67	85.82	84.44	83.52	83.39	83.96	83.72	84.53	85.44
003_cracker_box	85.85	83.72	89.08	88.07	82.12	80.18	84.26	84.39	88.74	87.60	89.61	87.85	85.51	83.68	79.75	88.00
004_sugar_box	81.30	77.71	80.45	79.26	78.35	74.75	77.17	76.66	83.72	79.53	81.91	79.25	78.06	75.68	74.87	77.40
005_tomato_soup_can	76.03	74.75	75.70	75.33	73.65	71.33	72.66	71.35	77.66	76.70	75.25	73.82	76.70	76.29	76.50	76.48
006_mustard_bottle	81.56	79.40	79.94	81.87	78.76	75.41	77.76	75.92	81.12	79.06	79.87	78.29	81.74	82.37	82.24	84.45
007_tuna_fish_can	68.08	67.14	67.76	67.34	68.99	67.76	68.07	67.59	73.37	72.06	71.72	70.68	75.76	77.70	75.76	77.61
008_pudding_box	73.60	70.39	74.05	72.58	69.77	68.04	69.78	68.20	78.88	76.74	79.01	76.58	71.32	71.43	65.11	70.94
009_gelatin_box	69.51	68.43	68.64	68.09	67.37	64.99	66.16	63.14	72.87	71.45	73.34	69.36	61.19	58.41	60.98	59.49
010_potted_meat_can	75.63	72.66	77.28	75.80	73.86	71.95	70.87	71.76	78.96	78.35	79.63	76.84	77.48	78.03	78.40	79.24
011_banana	70.53	63.43	69.67	65.49	68.07	60.50	67.31	60.57	72.46	58.18	70.80	70.68	71.67	67.54	73.25	70.28
019_pitcher_base	87.17	84.67	89.95	88.98	86.21	82.00	90.07	86.38	90.47	86.67	86.88	87.02	85.62	83.69	90.22	89.91
021_bleach_cleanser	80.98	77.52	75.52	79.71	79.59	76.68	78.77	78.52	84.75	81.71	82.79	81.75	77.36	71.54	77.01	75.54
024_bowl	80.62	78.12	75.79	80.30	78.75	76.42	78.06	77.39	83.25	80.12	81.54	79.72	81.12	79.69	81.91	82.99
025_mug	76.01	71.95	76.35	74.35	74.64	69.64	72.72	69.93	80.14	75.99	78.24	74.13	75.51	73.19	76.78	75.86
035_power_drill	81.83	73.80	82.85	77.20	76.89	69.02	77.90	71.50	82.57	74.66	82.42	74.09	83.67	77.62	84.06	80.77
036_wood_block	83.75	81.41	85.72	85.59	80.45	78.15	79.45	79.32	83.78	83.40	41.24	64.85	77.52	73.84	78.01	77.78
037_scissors	64.07	29.36	59.00	32.63	55.05	20.68	50.85	26.51	63.16	17.93	38.00	16.93	58.01	24.35	57.92	31.11
040_large_marker	52.69	42.42	50.95	42.46	50.14	38.46	44.94	36.46	53.98	36.47	50.78	33.58	55.35	41.23	53.66	45.22
052_extra_large_clamp	73.41	54.03	71.71	55.38	68.12	51.26	65.26	52.33	75.61	52.77	61.14	53.59	75.24	57.28	72.52	58.87
061_foam_brick	72.68	72.85	72.54	72.93	68.34	66.62	67.98	65.72	75.24	74.10	73.01	71.05	71.36	71.32	71.50	71.59
hand	71.85	54.83	66.41	54.27	64.88	46.86	58.33	46.66	70.23	54.70	59.28	52.08	71.23	54.89	66.33	53.51
mAP	75.76	69.58	75.13	71.56	72.69	66.26	71.48	67.24	77.94	70.60	72.37	68.66	75.02	69.69	74.35	72.05

Table 3: 2D object detection results in AP (%) on the four setups. We compare Mask R-CNN (Detectron2) [15, 43] with SOLOv2 [42] on both bounding box (bbox) and segmentation (segm).

Mask R-CNN predicts bounding boxes first and uses them to generate segmentations. Therefore the error can accumulate for segmentation. SOLOv2 in contrast directly predicts segmentations and uses it to generate bounding boxes. Second, the AP for hand is lower than the AP for objects (e.g., for Mask R-CNN, 71.85 for hand versus 75.76 mAP on S0 bbox). This suggests that hands are more difficult to detect than objects, possibly due to its larger variability in position. Finally, performance varies across setups. For example, mAP in S1 is lower than in S0 (e.g., for Mask R-CNN, 72.69 versus 75.76 on bbox). This can be attributed to the challenge introduced by unseen subjects. Tab. 4 shows the AP for keypoint detection with Mask R-CNN. We observe a similar trend on the ordering of AP over different setups.

5.3. 6D Object Pose Estimation

We evaluate single-view 6D pose estimation following the BOP challenge protocol [16, 17]. The task asks for a 6D pose estimate for each object instance given the number of objects and instances in each image. The evaluation computes recall using three different pose-error functions (VSD, MSSD, MSPD), and the final score is the average of the three recall values (AR).

We first analyze the challenge brought by hand grasping by comparing static and grasped objects. We use PoseCNN

	all	static	grasped
	52.68	56.53	41.65

Table 5: PoseCNN (RGB) [44] results in AR (%) on S0 (default).

(RGB) [44] as a baseline and retrain the model on DexYCB. Tab. 5 shows the AR on S0 evaluated separately on the full test set, static objects only, and grasped objects only. Without surprise the AR drops significantly when only considering the grasped objects. This recapitulates the increasing challenge of object pose estimation under hand interactions. To better focus on this regime, we evaluate only on the grasped objects in the remaining experiments.

We next evaluate on the four setups using the same baseline. Results are shown in Tab. 6 (left). On S1 (unseen subjects), we observe a drop in AR compared to S0 (e.g., 38.26 versus 41.65 on all) as in object detection. This can be attributed to the influence of interaction from unseen hands as well as different grasping styles. The column of S3 (unseen grasping) shows the AR of the three only objects being grasped in the test set (i.e. “009_gelatin_box”, “021_bleach_cleanser”, “036_wood_block”). Again, the AR is lower compared to on S0, and the drop is especially significant for smaller objects like “009_gelatin_box” (33.07 versus 46.62). Surprisingly, AR on S2 (unseen views) does not drop but rather increases slightly (e.g., 45.18 versus 41.65 on all). This suggests that our multi-camera setup has a dense enough coverage of the scene such that training on certain views can translate well to some others.

Finally, we benchmark five representative approaches: PoseCNN [44], DeepIM [20], DOPE [38], PoseRBPF [5],

	S0	S1	S2	S3	PoseCNN [44] RGB	PoseCNN [44] + depth ref	DeepIM [20] RGB	DeepIM [20] RGB-D	DOPE [38] RGB	PoseRBPF [5] RGB	PoseRBPF [5] RGB-D	CosyPose [19] RGB
002_master_chef_can	47.47	44.53	51.36	—	44.53	48.24	60.74	68.40	19.82	35.19	58.04	77.21
003_cracker_box	61.04	57.46	60.65	—	57.46	62.62	79.43	84.56	53.50	43.62	67.37	88.39
004_sugar_box	45.11	36.00	51.73	—	36.00	42.45	51.29	59.34	32.06	30.55	56.55	69.61
005_tomato_soup_can	36.68	34.88	45.44	—	34.88	42.99	46.87	55.46	20.52	23.76	40.03	52.76
006_mustard_bottle	52.56	42.89	51.96	—	42.89	48.98	55.52	63.31	23.66	28.98	54.93	67.11
007_tuna_fish_can	32.70	29.58	31.96	—	29.58	36.17	39.18	46.98	5.86	20.13	36.77	49.00
008_pudding_box	44.24	42.60	50.81	—	42.60	51.26	58.23	67.18	14.48	27.24	47.95	70.22
009_gelatin_box	46.62	34.39	44.38	33.07	34.39	41.71	47.89	55.54	21.81	31.27	46.14	57.63
010_potted_meat_can	37.41	42.10	45.08	—	42.10	51.64	60.19	69.40	14.15	31.17	46.43	65.37
011_banana	38.33	36.71	44.42	—	36.71	40.21	41.82	48.63	16.15	21.39	39.15	35.48
019_pitcher_base	53.49	46.03	53.51	—	46.03	49.62	54.27	63.46	7.02	19.94	51.38	52.68
021_bleach_cleanser	49.41	42.54	56.03	38.52	42.54	46.29	51.54	61.44	16.70	22.94	53.50	63.62
024_bowl	57.42	54.25	58.15	—	54.25	57.05	60.30	69.40	—	37.00	62.49	74.74
025_mug	40.68	37.28	43.70	—	37.28	38.69	41.15	49.51	—	18.21	34.87	48.48
035_power_drill	47.93	41.86	50.79	—	41.86	45.79	58.10	64.28	19.60	30.67	51.41	49.22
036_wood_block	40.11	38.66	49.50	40.53	38.66	44.08	49.91	65.39	—	23.80	51.63	67.14
037_scissors	21.93	22.87	25.90	—	22.87	25.29	27.48	32.65	14.16	17.18	29.40	24.36
040_large_marker	35.09	32.12	38.19	—	32.12	38.73	33.27	46.63	—	19.05	29.66	50.58
052_extra_large_clamp	30.48	31.89	34.42	—	31.89	33.60	39.63	47.04	—	22.99	35.99	50.78
061_foam_brick	12.80	13.04	15.75	—	13.04	16.70	18.14	27.57	—	17.40	32.42	30.12
all	41.65	38.26	45.18	37.41	38.26	43.27	48.99	57.54	—	26.24	46.48	57.43

Table 6: 6D object pose estimation results in AR (%). Left: PoseCNN (RGB). Right: performance of representative approaches on S1.

and CosyPose [19] (winner of the BOP challenge 2020). All of them take RGB input. For PoseCNN, we also include a variant with post-process depth refinement. For DeepIM and PoseRBPF, we also include their RGB-D variants. For CosyPose, we use its single view version. For DeepIM and CosyPose, we initialize the pose from the output of PoseCNN (RGB). We retrain all the models on DexYCB except for DOPE and PoseRBPF, which trained their models solely on synthetic images. For DOPE, we train models for an additional 7 objects besides the 5 provided. Tab. 6 (right) compares the results on the most challenging S1 (unseen subjects) setup. First, we can see a clear edge on those which also use depth compared to their respective RGB only variants (for PoseCNN, from 38.26 to 43.27 AR on all). Second, refinement based methods like DeepIM and CosyPose are able to improve significantly upon their initial pose input (e.g., for DeepIM RGB-D, from 38.26 to 57.54 AR on all). Finally, CosyPose achieves the best overall AR among all the RGB-based methods (57.43 AR on all).

5.4. 3D Hand Pose Estimation

The task is to estimate the 3D position of 21 hand joints from a single image. We evaluate with two metrics: mean per joint position error (MPJPE) in mm and percentage of correct keypoints (PCK). Besides computing error in absolute 3D position, we also report errors after aligning the predictions with ground truths in post-processing [49, 50, 11]. We consider two alignment methods: root-relative and Procrustes. The former removes ambiguity in translation by replacing the root (wrist) location with ground truths. The latter removes ambiguity in translation, rotation, and scale, thus focusing specifically on articulation. For PCK we re-

		absolute	root-relative	Procrustes
S0	[31] + ResNet50	53.92 (0.307)	17.71 (0.683)	7.12 (0.858)
	[31] + HRNet32	52.26 (0.328)	17.34 (0.698)	6.83 (0.864)
	A2J [45]	27.53 (0.612)	23.93 (0.588)	12.07 (0.760)
S1	[31] + ResNet50	70.23 (0.240)	22.71 (0.601)	8.43 (0.832)
	[31] + HRNet32	70.10 (0.248)	22.26 (0.615)	7.98 (0.841)
	A2J [45]	29.09 (0.584)	25.57 (0.562)	12.95 (0.743)
S2	[31] + ResNet50	83.46 (0.208)	23.15 (0.566)	8.11 (0.838)
	[31] + HRNet32	80.63 (0.217)	25.49 (0.530)	8.21 (0.836)
	A2J [45]	23.44 (0.576)	27.65 (0.540)	13.42 (0.733)
S3	[31] + ResNet50	58.69 (0.281)	19.41 (0.665)	7.56 (0.849)
	[31] + HRNet32	55.39 (0.311)	18.44 (0.686)	7.06 (0.859)
	A2J [45]	30.99 (0.608)	24.92 (0.581)	12.15 (0.759)

Table 7: 3D hand pose estimation results in MPJPE (mm). Numbers in parentheses are AUC values.

port the area under the curve (AUC) over the error range [0, 50 mm] with 100 steps.

We benchmark one RGB and one depth based approach. For RGB, we select a supervised version of Spurr et al. [31] which won the HANDS 2019 Challenge [1]. We experiment with the original ResNet50 backbone as well as a stronger HRNet32 [33] backbone, both initialized with ImageNet pre-trained models and finetuned on DexYCB. For depth, we select A2J [45] and retrain the model on DexYCB.

Tab. 7 shows the results. For [31], we can see that estimating absolute 3D position solely from RGB is difficult (e.g., 53.92 mm absolute MPJPE with ResNet50 on S0). The stronger HRNet32 backbone reduces the errors over ResNet50 but only marginally (e.g., 52.26 versus 53.92 mm absolute MPJPE on S0). Similar to in object pose, the errors on S1 (unseen subjects) increase from S0 (e.g., 70.10 versus 52.26 mm absolute MPJPE for HRNet32) due to the impact of unseen hands and grasping styles. However, unlike the trend in Tab. 6 (left), the errors on S2 (unseen views) also in-

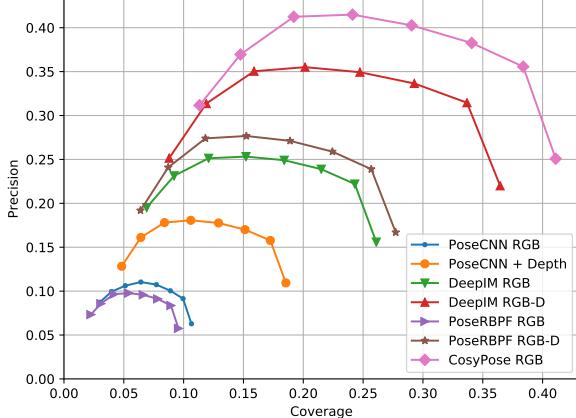


Figure 4: Precision-coverage curves for grasp generation on S1.

crease pronouncedly and even surpass the other setups (e.g., 80.63 mm absolute MPJPE for HRNet32). This is because unlike objects, the subjects are always facing the same direction from a situated position in this setup. This further constrains the possible hand poses observed in each view, making cross view generalization more challenging. Unsurprisingly, A2J [45] outperforms [31] on absolute MPJPE due to the depth input, but falls short on estimating articulation as shown by the errors after alignment (e.g., 12.07 versus 6.83 mm Procrustes MPJPE for HRNet32 on S0).

6. Safe Human-to-Robot Object Handover

Task Given an RGB-D image with a person holding an object, the goal is to generate a diverse set of robot grasps to take over the object without pinching the person’s hand (we refer to as “safe” handovers). The diversity of grasps is important since not all the grasps are kinematically feasible for execution. We assume a parallel-jaw Franka Panda gripper and represent each grasp as a point in $SE(3)$.

Evaluation We first sample 100 grasps for each YCB object using farthest point sampling from a diverse set of grasps pre-generated for that object in [7]. This ensures a dense coverage of the pose space (Fig. 15). For each image, we transform these grasps from the object frame to camera frame using the ground-truth object pose, and remove those collided with the ground-truth object and hand mesh. This generates a reference set of successful grasps \mathcal{R} .

Given a set of predicted grasps χ , we evaluate its diversity by computing the *coverage* [7] of \mathcal{R} , defined by the percentage of grasps in \mathcal{R} having at least one matched grasp in χ that is neither collided with the object nor the hand. Specifically, two grasps g, h are considered matched if $|g_t - h_t| < \sigma_t$ and $\arccos(|\langle g_q, h_q \rangle|) < \sigma_q$, where g_t is the translation and g_q is the orientation in quaternion. We use $\sigma_t = 0.05$ m and $\sigma_q = 15^\circ$.

One could potentially hit a high coverage by sampling grasps exhaustively. Therefore we also compute *precision*,

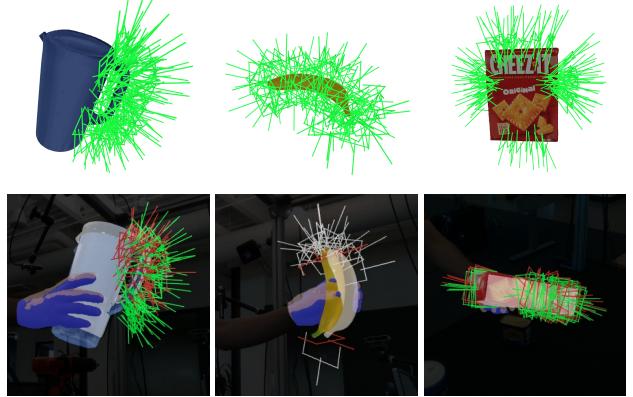


Figure 5: Top: 100 successful grasps sampled from [7]. Bottom: predicted grasps generated by the predicted object pose (textured model) and hand segmentation (blue masks). Green ones denote those covering successful grasps, red ones denote those collided with the object or hand, and gray ones are failures not covering any successful grasps in the reference set. Ground-truth objects and hands are shown in translucent white and brown meshes.

defined as the percentage of grasps in χ that have at least one matched grasp in \mathcal{R} .

Baseline We experiment with a simple baseline that only requires hand segmentation and 6D object pose. Similar to constructing \mathcal{R} , we transform the 100 grasps to the camera frame but using the estimated object pose, then remove those that are collided with the hand point cloud obtained by the hand segmentation and the depth image. Specifically, a grasp is collided if the distance of a pair of points from the gripper point cloud and the hand point cloud is less than a threshold ϵ . The gripper point cloud is obtained from a set of pre-sampled points on the gripper surface. We use the hand segmentation results from Mask R-CNN (Sec. 5.2).

Results We evaluate grasps generated with different object pose methods at different threshold $\epsilon \in [0, 0.07]$ m and show the precision-coverage curves on S1 in Fig. 4. We see that better object pose estimation leads to better grasp generation. Fig. 15 shows qualitative examples of the predicted grasps. We see that most of the failure grasps (red and gray) are due to inaccurate object pose. Some are hand-colliding grasps caused by a miss detected hand when the hand is partially occluded by the object (e.g., “003_cracker_box”). This can be potentially addressed by model based approaches that directly predict the full hand shape.

7. Conclusions

We have introduced DexYCB for capturing hand grasping of objects. We have shown its merits, presented a thorough benchmark of current approaches on three relevant tasks, and evaluated a new robotics-relevant task. We envision our dataset will drive progress on these crucial fronts.

References

- [1] Anil Armagan, Guillermo Garcia-Hernando, Seungryul Baek, Shreyas Hampali, Mahdi Rad, Zhaojun Zhang, Shipeng Xie, MingXiu Chen, Boshen Zhang, Fu Xiong, Yang Xiao, Zhiguo Cao, Junsong Yuan, Pengfei Ren, Weiting Huang, Haifeng Sun, Marek Hruž, Jakub Kanis, Zdeněk Krňouš, Qingfu Wan, Shile Li, Linlin Yang, Dongheui Lee, Angela Yao, Weiguo Zhou, Sijia Mei, Yunhui Liu, Adrian Spurr, Umar Iqbal, Pavlo Molchanov, Philippe Weinzaepfel, Romain Brégier, Grégory Rogez, Vincent Lepetit, and Tae-Kyun Kim. Measuring generalisation to unseen viewpoints, articulations, shapes and objects for 3D hand pose estimation under hand-object interaction. In *ECCV*, 2020. 7
- [2] Adnane Boukhayma, Rodrigo de Bem, and Philip H.S. Torr. 3D hand shape and pose from images in the wild. In *CVPR*, 2019. 1
- [3] Samarth Brahmbhatt, Chengcheng Tang, Christopher D. Twigg, Charles C. Kemp, and James Hays. ContactPose: A dataset of grasps with object contact and hand pose. In *ECCV*, 2020. 1, 4
- [4] Enric Corona, Albert Pumarola, Guillem Alenyà, Francesc Moreno-Noguer, and Grégory Rogez. GanHand: Predicting human grasp affordances in multi-object scenes. In *CVPR*, 2020. 2
- [5] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao-Blackwellized particle filter for 6D object pose estimation. In *RSS*, 2019. 1, 6, 7, 14, 15, 19
- [6] Bardia Doosti, Shujon Naha, Majid Mirbagheri, and David J. Crandall. HOPE-Net: A graph-based model for hand-object pose estimation. In *CVPR*, 2020. 1
- [7] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. A billion ways to grasps: An evaluation of grasp sampling schemes on a dense, physics-based grasp data set. In *ISRR*, 2019. 8, 14
- [8] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with RGB-D videos and 3D hand pose annotations. In *CVPR*, 2018. 1, 4
- [9] Luhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 3D hand shape and pose estimation from a single RGB image. In *CVPR*, 2019. 1
- [10] Oliver Glauser, Shihao Wu, Daniele Panozzo, Otmar Hilliges, and Olga Sorkine-Hornung. Interactive hand pose estimation using a stretch-sensing soft glove. In *SIGGRAPH*, 2019. 1
- [11] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. HOnnoteate: A method for 3D annotation of hand and object poses. In *CVPR*, 2020. 2, 3, 4, 5, 7, 12
- [12] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. DexPilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *ICRA*, 2020. 1
- [13] Yana Hasson, Bugra Tekin, Federica Bogo, Ivan Laptev, Marc Pollefeys, and Cordelia Schmid. Leveraging photometric consistency over time for sparsely supervised hand-object reconstruction. In *CVPR*, 2020. 1, 3
- [14] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *CVPR*, 2019. 1, 3, 4
- [15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 5, 6, 13, 18
- [16] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. BOP: Benchmark for 6D object pose estimation. In *ECCV*, 2018. 1, 3, 6
- [17] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labb  , Eric Brachmann, Frank Michel, Carsten Rother, and Ji   Matas. BOP challenge 2020 on 6D object localization. *ECCV Workshops*, 2020. 1, 3, 6
- [18] Umar Iqbal, Pavlo Molchanov, Thomas Breuel, Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5D heatmap regression. In *ECCV*, 2018. 1
- [19] Yann Labb  , Justin Carpentier, Mathieu Aubry, and Josef Sivic. CosyPose: Consistent multi-view multi-object 6D pose estimation. In *ECCV*, 2020. 1, 7, 13, 14, 15, 19
- [20] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep iterative matching for 6D pose estimation. In *ECCV*, 2018. 1, 6, 7, 13, 14, 15, 19
- [21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll  , and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5, 16
- [22] Gyeongsik Moon, Shouo-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. InterHand2.6M: A dataset and baseline for 3D interacting hand pose estimation from a single RGB image. In *ECCV*, 2020. 4
- [23] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. GANerated Hands for real-time 3D hand tracking from monocular RGB. In *CVPR*, 2018. 1, 4
- [24] Franziska Mueller, Micah Davis, Florian Bernard, Oleksandr Sotnychenko, Mickeal Verschoor, Miguel A. Otaduy, Dan Casas, and Christian Theobalt. Real-time pose and shape reconstruction of two interacting hands with a single depth camera. In *SIGGRAPH*, 2019. 3, 4
- [25] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time hand tracking under occlusion from an egocentric RGB-D sensor. In *ICCV*, 2017. 4
- [26] Kiru Park, Timothy Patten, and Markus Vincze. Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In *ICCV*, 2019. 1
- [27] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-wise voting network for 6DoF pose estimation. In *CVPR*, 2019. 1
- [28] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. In *SIGGRAPH Asia*, 2017. 3

- [29] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F. Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, 2020. 5
- [30] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017. 4
- [31] Adrian Spurr, Umar Iqbal, Pavlo Molchanov, Otmar Hilliges, and Jan Kautz. Weakly supervised 3D hand pose estimation via biomechanical constraints. In *ECCV*, 2020. 1, 5, 7, 8, 13, 20
- [32] Srinath Sridhar, Franziska Mueller, Michael Zollhoefer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. Real-time joint tracking of a hand manipulating an object from RGB-D input. In *ECCV*, 2016. 4
- [33] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 5, 7
- [34] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3D orientation learning for 6D object detection from RGB images. In *ECCV*, 2018. 1
- [35] Omid Taheri, Nima Ghorbani, Michael J. Black, and Dimitrios Tzionas. GRAB: A dataset of whole-body human grasping of objects. In *ECCV*, 2020. 1, 4
- [36] Bugra Tekin, Federica Bogo, and Marc Pollefeys. H+O: Unified egocentric recognition of 3D hand-object poses and interactions. In *CVPR*, 2019. 1
- [37] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6D object pose prediction. In *CVPR*, 2018. 1
- [38] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *CoRL*, 2018. 1, 6, 7
- [39] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 101(1):184–204, Jan 2013. 2
- [40] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martin-Martin, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. In *CVPR*, 2019. 1
- [41] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *CVPR*, 2019. 3
- [42] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic, faster and stronger. In *NeurIPS*. 2020. 5, 6
- [43] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019. 5, 6, 13, 18
- [44] Yu Xiang, Tanner Schmidt, Venkatraman Nafurayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *RSS*, 2018. 2, 3, 6, 7, 13, 14, 15, 19
- [45] Fu Xiong, Boshen Zhang, Yang Xiao, Zhiguo Cao, Taidong Yu, Joey Tianyi Zhou, and Junsong Yuan. A2J: Anchor-to-joint regression network for 3D articulated pose estimation from a single depth image. In *ICCV*, 2019. 7, 8
- [46] Wei Yang, Chris Paxton, Maya Cakmak, and Dieter Fox. Human grasp classification for reactive human-to-robot handovers. In *IROS*, 2020. 1
- [47] Zhixuan Yu, Jae Shin Yoon, In Kyu Lee, Prashanth Venkatesh, Jaesik Park, Jihun Yu, and Hyun Soo Park. HUMBI: A large multiview dataset of human body expressions. In *CVPR*, 2020. 5
- [48] Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhant Jain, and Taekyun Kim. BigHand2.2M benchmark: Hand pose dataset and state of the art analysis. In *CVPR*, 2017. 1, 4
- [49] Christian Zimmermann and Thomas Brox. Learning to estimate 3D hand pose from single RGB images. In *ICCV*, 2017. 1, 4, 7
- [50] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max J. Argus, and Thomas Brox. FreiHAND: A dataset for markerless capture of hand pose and shape from single RGB images. In *ICCV*, 2019. 3, 4, 5, 7, 12

Appendices

Contents

A Constructing DexYCB	11
A.1 Optimizing E_{depth}	11
B Properties of DexYCB	11
B.1 Visualization	11
B.2 Analysis on 3D Annotation Accuracy	12
B.3 Diversity of Hand Pose	12
B.4 Why Black Background?	12
B.5 Why Generalize Better over HO-3D?	12
C Benchmarking Representative Approaches	12
C.1 Setup Details	12
C.2 Qualitative: 2D Object and Keypoint Detection	13
C.3 Qualitative: 6D Object Pose Estimation	13
C.4 Qualitative: 3D Hand Pose Estimation	13
C.5 Full Quantitative: 6D Object Pose Estimation	14
D Safe Human-to-Robot Object Handover	14
D.1 Pre-Generated Grasps	14
D.2 Additional Qualitative Results	14
D.3 Full Quantitative Results	14
E Results on In-the-Wild Images	15

A. Constructing DexYCB

A.1. Optimizing E_{depth}

Here we provide the details on optimizing the depth term E_{depth} in solving 3D hand and object pose (Sec. 2.3). Below we first describe our efficient forward computation based on a point-parallel GPU implementation. Then we show that E_{depth} is differentiable and derive its gradient.

Recall that $\{d_i \in \mathbb{R}^3\}_{i=1}^{N_D}$ denotes the entire point cloud where d_i is a single 3D point, and $\mathcal{M}(P)$ is the triangular mesh of our model. Without loss of generality, we assume that our model contains only one single rigid object, i.e. $P \in \text{SE}(3)$. Recall that the depth term is defined as

$$E_{\text{depth}}(P) = \frac{1}{N_D} \sum_{i=1}^{N_D} |\text{SDF}(d_i, \mathcal{M}(P))|^2, \quad (7)$$

where $\text{SDF}(\cdot)$ calculates the signed distance value of a 3D point d from the mesh \mathcal{M} . One naive way to compute this value is to exhaustively compute the distance from the query point d to all the triangular faces on the mesh, and find the minimum value among them. To perform a more efficient computation, we leverage the bounding volume hierarchy (BVH) technique from graphics. Specifically, we build an axis-aligned bounding box tree (AABB tree) to index the triangular faces of the mesh at the start of each forward computation of Eq. 7. The AABB tree allows us to efficiently traverse the triangular faces of the mesh and compute distances from the query d without going through all the faces exhaustively. Furthermore, since Eq. 7 repeats the same computation for all the query points $\{d_i\}$, we can achieve further speedup by parallelizing over query points using a GPU implementation.

Now let us turn to the backward pass of Eq. 7, since we optimize $E_{\text{depth}}(P)$ using a gradient-based method. During the forward computation, for a query point d , once we find its closest point $q \in \mathbb{R}^3$ on the mesh, we represent q using barycentric coordinates:

$$q = u \cdot a + v \cdot b + w \cdot c, \quad (8)$$

where $a, b, c \in \mathbb{R}^3$ denotes the three vertices of the triangular face which q lies on, and $u, v, w \in \mathbb{R}$ with $0 \leq u, v, w \leq 1$ and $u + v + w = 1$. Now we can further represent $E_{\text{depth}}(P)$ with

$$\begin{aligned} E_{\text{depth}}(P) &= \frac{1}{N_D} \sum_{i=1}^{N_D} |\text{SDF}(d_i, \mathcal{M}(P))|^2 \\ &= \frac{1}{N_D} \sum_{i=1}^{N_D} \|d_i - q_i\|_2^2. \end{aligned} \quad (9)$$

We can thus derive the gradient as

$$\frac{\partial E_{\text{depth}}(P)}{\partial P} = \frac{1}{N_D} \sum_{i=1}^{N_D} \frac{\partial \|d_i - q_i\|_2^2}{\partial P}. \quad (10)$$

Let $\mathcal{V}_n = (\mathcal{V}_{n,x}, \mathcal{V}_{n,y}, \mathcal{V}_{n,z}) \in \mathbb{R}^3$ denote the n th vertex of the mesh \mathcal{M} with N_V vertices. Using the multivariable chain rule, we get

$$\frac{\partial \|d - q\|_2^2}{\partial P} = \sum_{n=1}^{N_V} \sum_{m \in \{x, y, z\}} \frac{\partial \|d - q\|_2^2}{\partial \mathcal{V}_{n,m}} \cdot \frac{\partial \mathcal{V}_{n,m}}{\partial P}. \quad (11)$$

With the barycentric representation of q from Eq. 8, we can write

$$\begin{aligned} \|d - q\|_2^2 &= \|d - (u \cdot a + v \cdot b + w \cdot c)\|_2^2 \\ &= (d_x - u \cdot a_x - v \cdot b_x - w \cdot c_x)^2 \\ &\quad + (d_y - u \cdot a_y - v \cdot b_y - w \cdot c_y)^2 \\ &\quad + (d_z - u \cdot a_z - v \cdot b_z - w \cdot c_z)^2. \end{aligned} \quad (12)$$

Finally, with Eq. 11 and 12, we can derive the gradient with respect to the vertices as

$$\begin{aligned} \frac{\partial \|d - q\|_2^2}{\partial \mathcal{V}_{n,m}} &= \\ \left\{ \begin{array}{ll} -2 \cdot u \cdot (d_x - u \cdot a_x - v \cdot b_x - w \cdot c_x) & \text{if } \mathcal{V}_{n,m} = a_x, \\ -2 \cdot u \cdot (d_y - u \cdot a_y - v \cdot b_y - w \cdot c_y) & \text{if } \mathcal{V}_{n,m} = a_y, \\ -2 \cdot u \cdot (d_z - u \cdot a_z - v \cdot b_z - w \cdot c_z) & \text{if } \mathcal{V}_{n,m} = a_z, \\ -2 \cdot v \cdot (d_x - u \cdot a_x - v \cdot b_x - w \cdot c_x) & \text{if } \mathcal{V}_{n,m} = b_x, \\ -2 \cdot v \cdot (d_y - u \cdot a_y - v \cdot b_y - w \cdot c_y) & \text{if } \mathcal{V}_{n,m} = b_y, \\ -2 \cdot v \cdot (d_z - u \cdot a_z - v \cdot b_z - w \cdot c_z) & \text{if } \mathcal{V}_{n,m} = b_z, \\ -2 \cdot w \cdot (d_x - u \cdot a_x - v \cdot b_x - w \cdot c_x) & \text{if } \mathcal{V}_{n,m} = c_x, \\ -2 \cdot w \cdot (d_y - u \cdot a_y - v \cdot b_y - w \cdot c_y) & \text{if } \mathcal{V}_{n,m} = c_y, \\ -2 \cdot w \cdot (d_z - u \cdot a_z - v \cdot b_z - w \cdot c_z) & \text{if } \mathcal{V}_{n,m} = c_z, \\ 0 & \text{otherwise.} \end{array} \right. \end{aligned} \quad (13)$$

Since we can also compute $\partial \mathcal{V}_{n,m} / \partial P$ for rigid objects in Eq. 11, we can obtain the gradient of $E_{\text{depth}}(P)$. In fact, we can compute the gradient of $E_{\text{depth}}(P)$ as long as $\mathcal{V}_{n,m}(P)$ is differentiable. This holds true for the YCB objects (rigid) and also the MANO hand model (deformable). Therefore we can optimize Eq. 7 with both our object and hand models. Similar to the forward computation, the backward pass can also be parallelized using a GPU implementation.

B. Properties of DexYCB

B.1. Visualization

For a better visualization of data and annotations please see the supplementary video.⁴

⁴<https://youtu.be/Q4wyBaZeBw0>.

thumb tip	index tip	middle tip	ring tip	pinky tip
4.27 ± 3.42	4.99 ± 3.67	4.33 ± 3.53	4.03 ± 3.56	3.90 ± 3.45

Table 8: Reprojection error (pixels) of the five finger tips.



Figure 6: Qualitative examples of annotations with high reprojection errors.

B.2. Analysis on 3D Annotation Accuracy

To analyze the accuracy of 3D annotation in DexYCB, we compute the reprojection error of ground-truth 3D keypoints with human labeled 2D keypoints in each image. Tab. 8 reports the reprojection errors of the five finger tips. The mean errors are all below 5 pixels. As shown in Fig. 6, the large errors are often produced by a fast moving hand (top) and occasionally ambiguous annotations (bottom: ring labeled as pinky).

B.3. Diversity of Hand Pose

DexYCB pushes prior hand datasets to an unprecedented scale on the diversity of grasps. To analyze the diversity of hand pose, we extract the PCA representation of the MANO hand model for each hand pose and visualize the distribution in 2D space using the first two PCA coefficients. Fig. 7 compares the distribution of hand pose in DexYCB with HO-3D [11]. While DexYCB grows the number of objects as well as the number of grasps per object over prior datasets, the main edge actually lies in the diversity of captured grasps as shown by the distribution of hand pose. This diversity marks an essential challenge of pose estimation in hand-object interactions and downstream applications like object handover to robots.

B.4. Why Black Background?

The main focus of DexYCB is on the diversity of grasps, not the diversity of backgrounds. Capturing hand-object interaction is challenging already in controlled settings. Besides, DexYCB is no lesser than prior datasets in this respect: FreiHAND [50] used a green background; while HO-

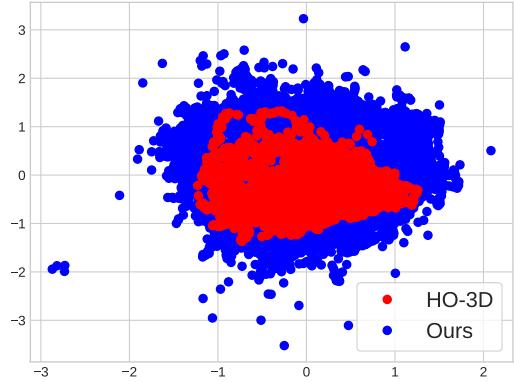


Figure 7: Distribution of hand pose represented by the first two MANO PCA coefficients.



Figure 8: The two captured scene backgrounds (top and bottom) in HO-3D [11].

3D [11] used more natural backgrounds, the diversity is not much better since the choice of background scenes in HO-3D is still only two (Fig. 8).

B.5. Why Generalize Better over HO-3D?

Following B.3 and B.4, DexYCB generalizes better due to a better diversity of grasps given a similar diversity of scene background. This explains the edge of DexYCB in the cross-dataset evaluation in Sec. 4.

C. Benchmarking Representative Approaches

C.1. Setup Details

Tab. 9 shows the statistics of the four evaluation setups: S0 (default), S1 (unseen subjects), S2 (unseen views), and S3 (unseen grasping). The first row (all) represents the full dataset, and each sub-table below shows the statistics of one particular setup, divided into train/val/test splits. For each split, we list the number of subjects (“#sub”), objects

	#sub	#obj	#view	#seq	#image	#obj anno				#hand anno		
						all	grasped	6D eval	handover eval	all	right	left
all	10	20	8	1,000	581,968	2,317,312	581,968	–	–	508,384	254,000	254,384
S0 (default)												
train	10	20	8	800	465,504	1,846,144	465,504	–	–	406,888	203,096	203,792
val	2	20	8	40	23,200	97,456	23,200	–	–	22,728	11,296	11,432
test	8	20	8	160	93,264	373,712	93,264	94,816	1,280	78,768	39,608	39,160
S1 (unseen subjects)												
train	7	20	8	700	407,088	1,620,128	407,088	–	–	356,600	177,656	178,944
val	1	20	8	100	58,592	226,288	58,592	–	–	47,656	23,848	23,808
test	2	20	8	200	116,288	470,896	116,288	119,192	1,600	104,128	52,496	51,632
S2 (unseen views)												
train	10	20	6	750	436,476	1,737,984	436,476	–	–	381,288	190,500	190,788
val	10	20	1	125	72,746	289,664	72,746	–	–	63,548	31,750	31,798
test	10	20	1	125	72,746	289,664	72,746	73,383	1,000	63,548	31,750	31,798
S3 (unseen grasping)												
train	10	15	8	750	436,416	1,742,672	436,416	–	–	381,288	189,712	191,576
val	10	2	8	100	58,192	221,664	58,192	–	–	50,736	25,864	24,872
test	10	3	8	150	87,360	352,976	87,360	89,392	1,200	76,360	38,424	37,936

Table 9: Statistics of the four evaluation setups: S0 (default), S1 (unseen subjects), S2 (unseen views), and S3 (unseen grasping).

(“#obj”), views (“#view”), sequences (“#seq”), and image samples (“#image”).

We also list the number of object annotations (“#obj anno”) in each split, including the full object set (“all”) and the subset with only the grasped objects (“grasped”). For 6D object pose estimation, we follow the BOP challenge to speed up the evaluation by evaluating on subsampled keyframes. We use a subsampling factor of 4. The column “6D eval” lists the number of object annotations in the keyframes of the test split. For object handover, the test images need to capture a person with an object in hand ready for handover. Therefore we use the last frame of each video as the test samples, since during capture the subjects are instructed to hand over the object to someone across at the end. The column “handover eval” lists the number of in-hand object annotations (which is also the number of the last frames from all videos since each image contains one in-hand object) in the test split for the handover evaluation.

Finally, we list the number of hand annotations (“#hand anno”) in each split, including the full hand set (“all”) and the subsets with only right (“right”) and left (“left”) hands.

C.2. Qualitative: 2D Object and Keypoint Detection

Fig. 11 shows qualitative results of 2D object detection and keypoint detection with Mask R-CNN (Detronet2) [15, 43] on the S0 setup. We highlight some failure examples in the last two rows. In many failure cases we see false object detections due to occlusions either by other objects (e.g., “036_wood_block” in row 5 and column 2) or by hand interaction (e.g., “021_bleach_cleanser” in row 6

and column 1). We also see inaccurately detected hand keypoints when the hand is interacting with objects (e.g., row 6 and column 4).

C.3. Qualitative: 6D Object Pose Estimation

Fig. 12 shows qualitative results of 6D object pose estimation on the S1 setup. The first row shows the input RGB images and the following rows show the estimated pose from each representative approach. We render object models given the estimated poses and overlay them on top of a darkened input image. We can see the challenge when the object is severely occluded by the hand (e.g., the left-most example of PoseCNN [44] (RGB)). We also see that refinement based approaches like DeepIM [20] and Cosy-Pose [19] are able to improve upon their coarse estimate input (i.e., PoseCNN (RGB)) and generate more accurate final predictions (e.g., the second-left example of DeepIM (RGB)).

C.4. Qualitative: 3D Hand Pose Estimation

Fig. 13 shows qualitative results of 3D hand pose estimation using Spurr et al.’s method [31] (HRNet32) on the S0 setup. The method is able to generate sensible articulated pose even under object occlusions. This is consistent with the quantitative results reported in Tab. 7, where the mean per joint position error after Procrustes alignment is less than 1cm (6.83mm). As suggested in that table, the major source of error comes from translation, rotate, and scale (Absolute), rather than articulation. Nonetheless, we still see errors in local articulation when objects are in close contact (e.g., the middle finger in row 2 and column 1) and

	S0 (default)						
	PoseCNN [44]		DeepIM [20]		PoseRBPF [5]		CosyPose [19]
	RGB	+ depth ref	RGB	RGB-D	RGB	RGB-D	RGB
002_master_chef_can	47.47	51.04	63.53	71.69	36.70	55.23	78.97
003_cracker_box	61.04	64.44	79.82	86.53	48.54	68.47	88.74
004_sugar_box	45.11	49.90	59.30	70.98	34.44	58.95	78.45
005_tomato_soup_can	36.68	46.23	52.84	61.28	29.54	42.60	57.89
006_mustard_bottle	52.56	56.10	60.92	71.00	33.86	59.20	71.57
007_tuna_fish_can	32.70	36.37	39.78	46.81	19.14	35.03	46.31
008_pudding_box	44.24	51.72	56.55	68.26	29.64	52.34	68.52
009_gelatin_box	46.62	56.15	62.49	72.60	32.08	49.88	70.38
010_potted_meat_can	37.41	43.42	55.48	63.33	32.98	44.39	59.23
011_banana	38.33	42.41	46.47	53.30	22.53	42.96	38.01
019_pitcher_base	53.49	56.39	60.01	71.27	18.66	59.32	55.43
021_bleach_cleanser	49.41	54.87	59.13	71.73	32.03	60.40	68.80
024_bowl	57.42	58.90	63.92	73.15	34.35	61.07	77.12
025_mug	40.68	43.52	49.17	58.27	21.88	37.69	54.70
035_power_drill	47.93	51.59	62.10	72.30	37.54	58.86	52.02
036_wood_block	40.11	46.76	51.77	68.28	25.29	53.23	68.46
037_scissors	21.93	24.11	27.57	33.59	22.32	32.31	23.82
040_large_marker	35.09	42.49	35.35	49.04	25.05	36.03	53.61
052_extra_large_clamp	30.48	34.75	39.94	49.96	22.07	36.74	52.18
061_foam_brick	12.80	15.96	17.50	25.82	18.33	35.10	34.34
all	41.65	46.40	52.25	62.03	28.90	49.09	59.99

Table 10: 6D object pose estimation results of representative approaches in AR (%) on S0.

when the fingers are largely occluded by the held object (e.g., the index finger in row 6 and column 4).

C.5. Full Quantitative: 6D Object Pose Estimation

For 6D object pose estimation, since in Tab. 6 we report benchmark results only on S1, we now include the results on the other three setups. Tab. 10, 11, and 12 show the results in AR on S0, S2, and S3, respectively. Overall, we observe a similar trend as on S1 (see Sec. 5.3), where DeepIM (RGB-D) [20] and CosyPose [19] are the two most competitive approaches in estimation accuracy.

D. Safe Human-to-Robot Object Handover

D.1. Pre-Generated Grasps

Fig. 14 visualizes the 100 grasps for each object used in our evaluation. These grasps are sampled from the pre-generated grasps for YCB objects in [7]. Note that here we only show the grasps for 18 out of 20 objects in DexYCB, since the remaining two objects (“002_master_chef_can” and “036_wood_block”) do not have any feasible grasps using the gripper of choice (i.e., Franka Panda).

D.2. Additional Qualitative Results

Fig. 15 shows additional qualitative results of the predicted grasps for human-to-robot object handover. Interestingly, larger objects like “003_cracker_box” (row 1 and column 1) are less tolerant to errors in pose estimation, since

most successful grasps requires the gripper to be fully open and barely fitting the object. Therefore a slight error in the estimated pose will cause the gripper to collide with the object. At the same time, errors in object pose estimation may cause the gripper to miss the grasp especially on smaller objects (e.g., gray grasps for “037_scissors” in row 4 and column 3). On the other hand, a hand that is miss or partially detected due to occlusion may cause a potential pinch by the gripper (e.g., red grasps for “061_foam_brick” in row 5 and column 3).

D.3. Full Quantitative Results

Since in Fig. 4 we show the results of grasp generation only on S1, we now include the results on the other three setups. Fig. 9 shows the precision-coverage curves on S0, S2, and S3, respectively. Overall, similar to on S1, we can see that more accurate object pose estimation leads to better performance in grasp generation.

It is worth noting that although both DeepIM (RGB-D) [20] and CosyPose [19] achieved very close performance on 6D object pose (e.g. 57.54 versus 57.43 AR in Tab. 6), the later performs significantly better on grasp generation (e.g. see Fig. 4). This is because that DeepIM (RGB-D) maintains a competitive average recall by only outperforming CosyPose on a smaller set of objects (e.g. 7 on S1) but with larger margins, whereas CosyPose shows an edge over DeepIM (RGB-D) on more objects (e.g. 13 on S1). This shows that a higher performance on 6D pose metrics like

	S2 (unseen views)						
	PoseCNN [44]		DeepIM [20]		PoseRBPF [5]		CosyPose [19]
	RGB	+ depth ref	RGB	RGB-D	RGB	RGB-D	RGB
002_master_chef_can	51.36	57.42	68.86	72.87	37.64	63.04	81.94
003_cracker_box	60.65	67.45	84.10	89.34	54.25	74.63	90.66
004_sugar_box	51.73	60.06	69.68	76.23	47.15	71.17	83.11
005_tomato_soup_can	45.44	52.56	55.36	65.60	32.62	47.60	61.52
006_mustard_bottle	51.96	58.77	64.04	71.64	40.81	65.02	73.55
007_tuna_fish_can	31.96	39.20	44.21	51.25	26.09	44.43	55.65
008_pudding_box	50.81	61.41	64.94	73.88	36.66	62.11	77.75
009_gelatin_box	44.38	52.93	59.64	70.02	37.51	47.61	72.51
010_potted_meat_can	45.08	54.94	66.58	70.41	40.13	51.26	68.60
011_banana	44.42	49.09	48.16	58.30	24.85	49.07	42.50
019_pitcher_base	53.51	56.73	61.36	71.98	26.37	62.30	56.77
021_bleach_cleanser	56.03	62.24	66.20	76.89	38.13	67.77	73.86
024_bowl	58.15	62.15	67.75	72.60	40.76	70.66	80.42
025_mug	43.70	46.79	52.47	58.65	26.81	48.77	59.09
035_power_drill	50.79	57.14	68.25	75.54	40.00	60.70	54.66
036_wood_block	49.50	57.34	57.19	75.54	33.19	60.41	72.40
037_scissors	25.90	28.21	31.42	36.68	27.86	39.38	26.90
040_large_marker	38.19	48.47	38.21	56.44	27.65	35.72	58.60
052_extra_large_clamp	34.42	40.21	42.24	52.72	28.25	46.33	58.73
061_foam_brick	15.75	19.11	20.10	28.39	26.19	40.90	31.58
all	45.18	51.61	56.54	65.25	34.65	55.44	64.04

Table 11: 6D object pose estimation results of representative approaches in AR (%) on S2.

	S3 (unseen grasping)						
	PoseCNN [44]		DeepIM [20]		PoseRBPF [5]		CosyPose [19]
	RGB	+ depth ref	RGB	RGB-D	RGB	RGB-D	RGB
002_master_chef_can	—	—	—	—	—	—	—
003_cracker_box	—	—	—	—	—	—	—
004_sugar_box	—	—	—	—	—	—	—
005_tomato_soup_can	—	—	—	—	—	—	—
006_mustard_bottle	—	—	—	—	—	—	—
007_tuna_fish_can	—	—	—	—	—	—	—
008_pudding_box	—	—	—	—	—	—	—
009_gelatin_box	33.07	43.43	47.68	54.13	29.95	47.71	58.29
010_potted_meat_can	—	—	—	—	—	—	—
011_banana	—	—	—	—	—	—	—
019_pitcher_base	—	—	—	—	—	—	—
021_bleach_cleanser	38.52	45.32	48.32	60.18	25.35	58.31	63.04
024_bowl	—	—	—	—	—	—	—
025_mug	—	—	—	—	—	—	—
035_power_drill	—	—	—	—	—	—	—
036_wood_block	40.53	48.17	50.54	65.74	27.30	55.19	65.31
037_scissors	—	—	—	—	—	—	—
040_large_marker	—	—	—	—	—	—	—
052_extra_large_clamp	—	—	—	—	—	—	—
061_foam_brick	—	—	—	—	—	—	—
all	37.41	45.66	48.86	60.07	27.52	53.78	62.25

Table 12: 6D object pose estimation results of representative approaches in AR (%) on S3.

AR does not necessarily translate to a higher performance on downstream robotics tasks like object handover.

E. Results on In-the-Wild Images

Since DexYCB was captured in a controlled lab environment with a constant background, models trained on the

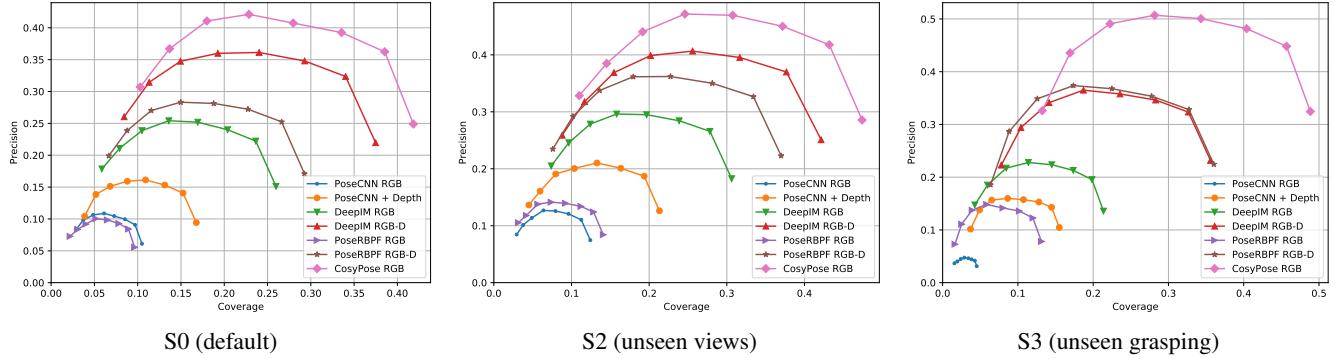


Figure 9: Precision-coverage curves for grasp generation on S0, S2, and S3.

RGB images of DexYCB are not expected to generalize well to in-the-wild images. We tested a DexYCB-trained model on COCO images [21] and observed an expected drop in performance. Fig. 10 shows qualitative examples of 2D hand and keypoint detection. We can see that the detector frequently produces false positives (top left), false negatives (top middle), and inaccurate keypoint detection (top right). Combining DexYCB with other in-the-wild datasets for training will be an interesting follow up work.



Figure 10: Samples of 2D hand and keypoint detection on COCO.

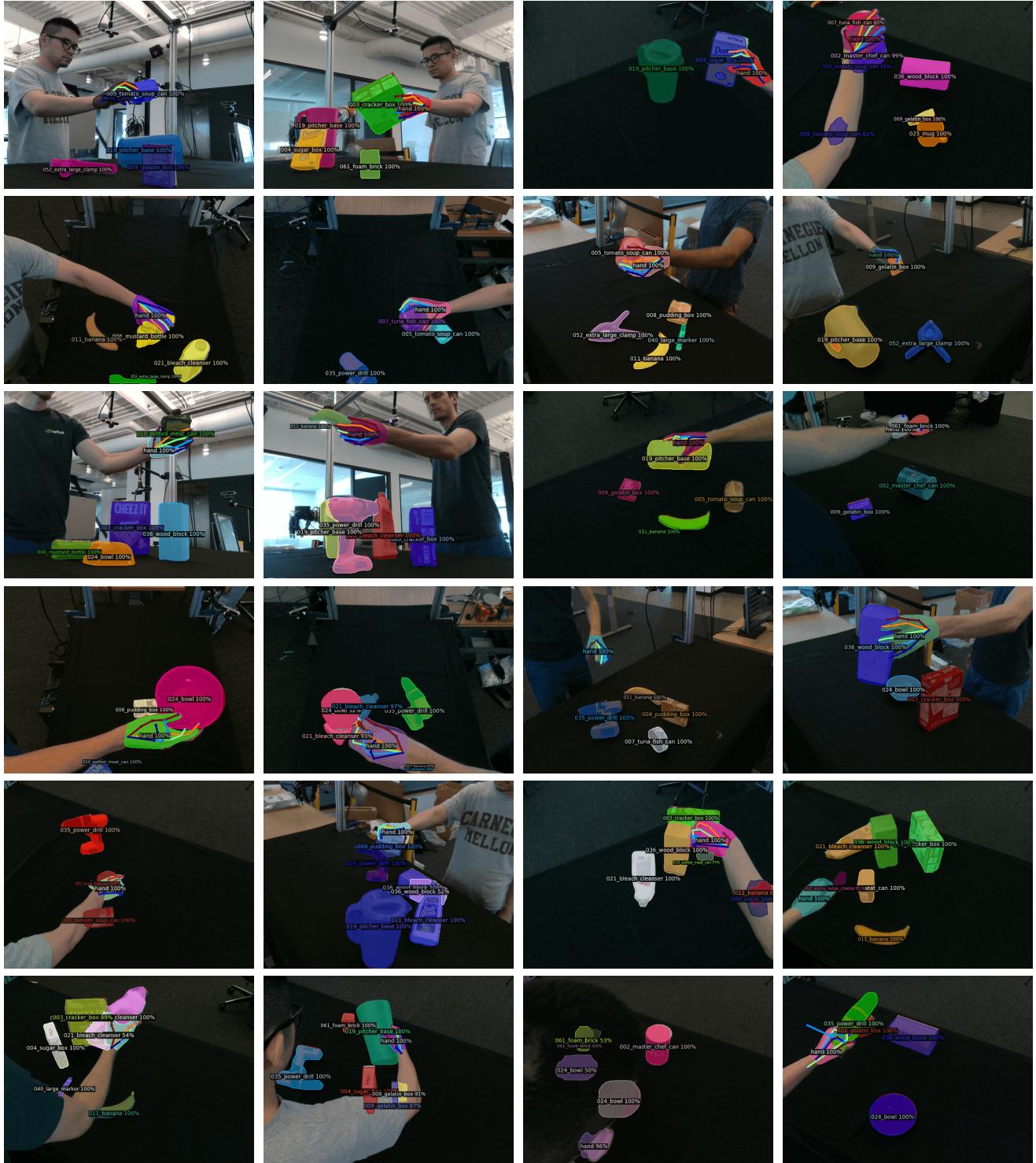


Figure 11: Qualitative results of 2D object and keypoint detection with Mask R-CNN (Detectron2) [15, 43]. The last two rows highlight failure examples.

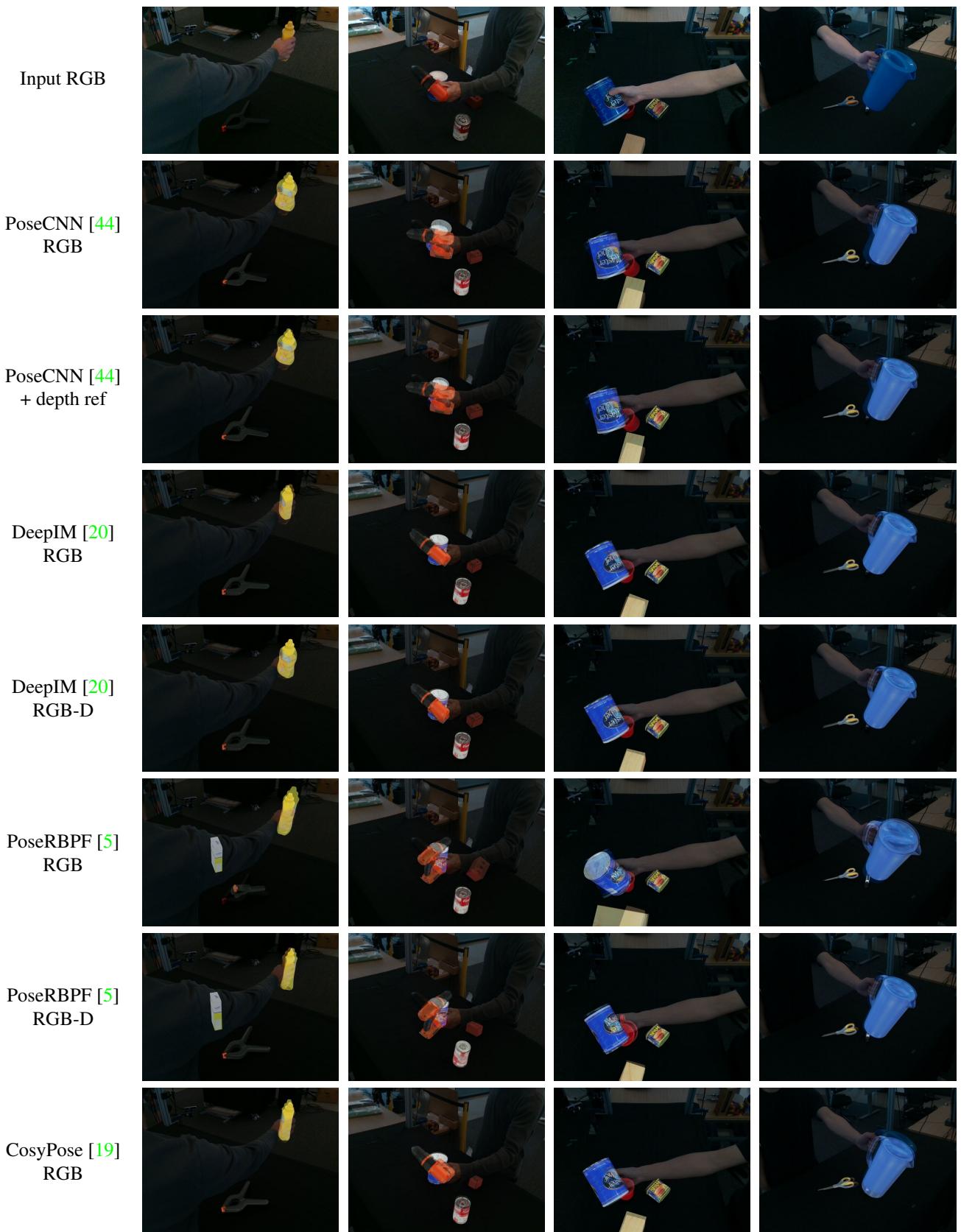


Figure 12: Qualitative results of 6D object pose estimation. We render object models given the estimated poses on a darkened input image.

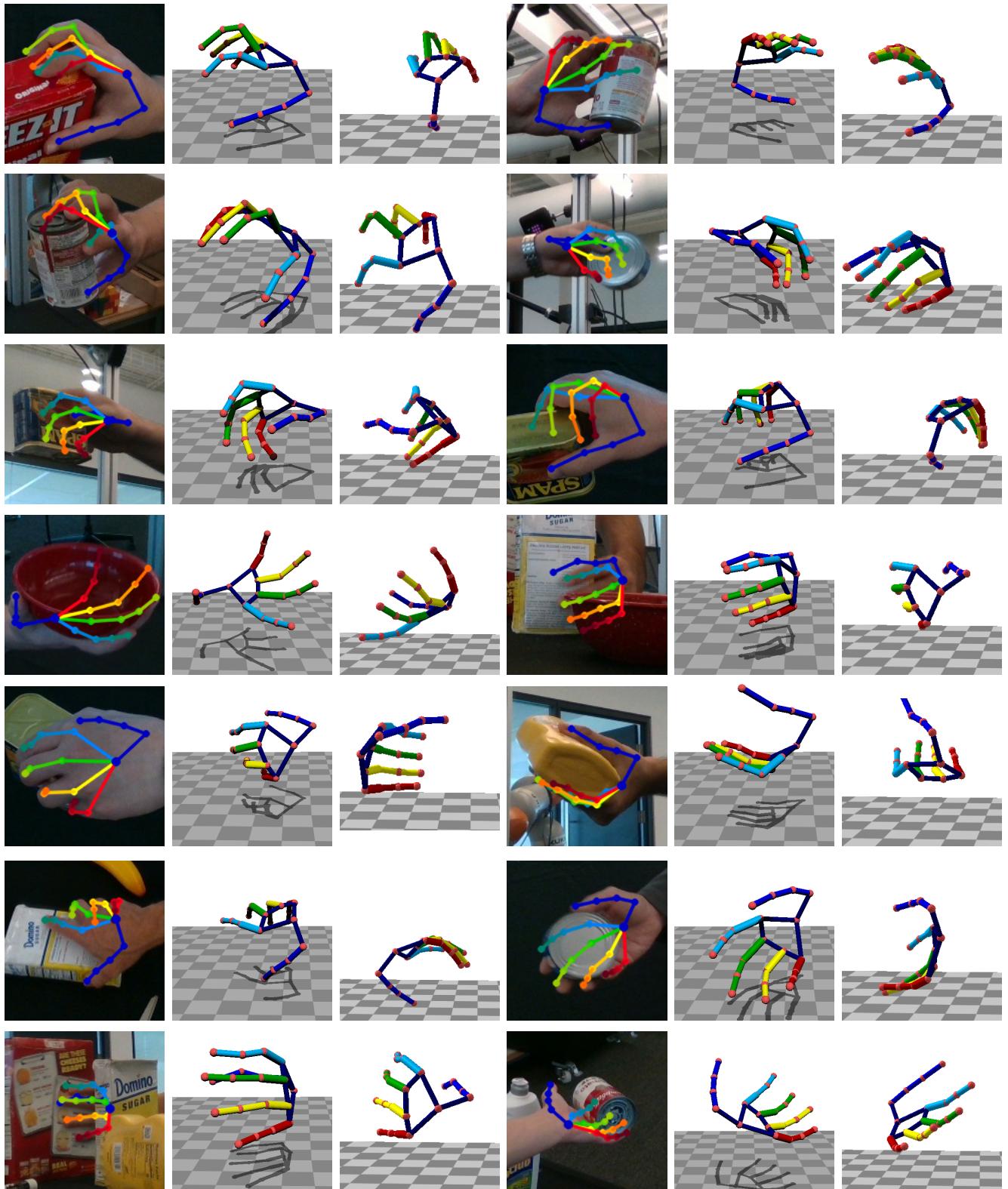


Figure 13: Qualitative results of the predicted 3D hand pose using Spurr et al.'s method [31] (HRNet32). For each image we visualize 3D pose from both front and side views.

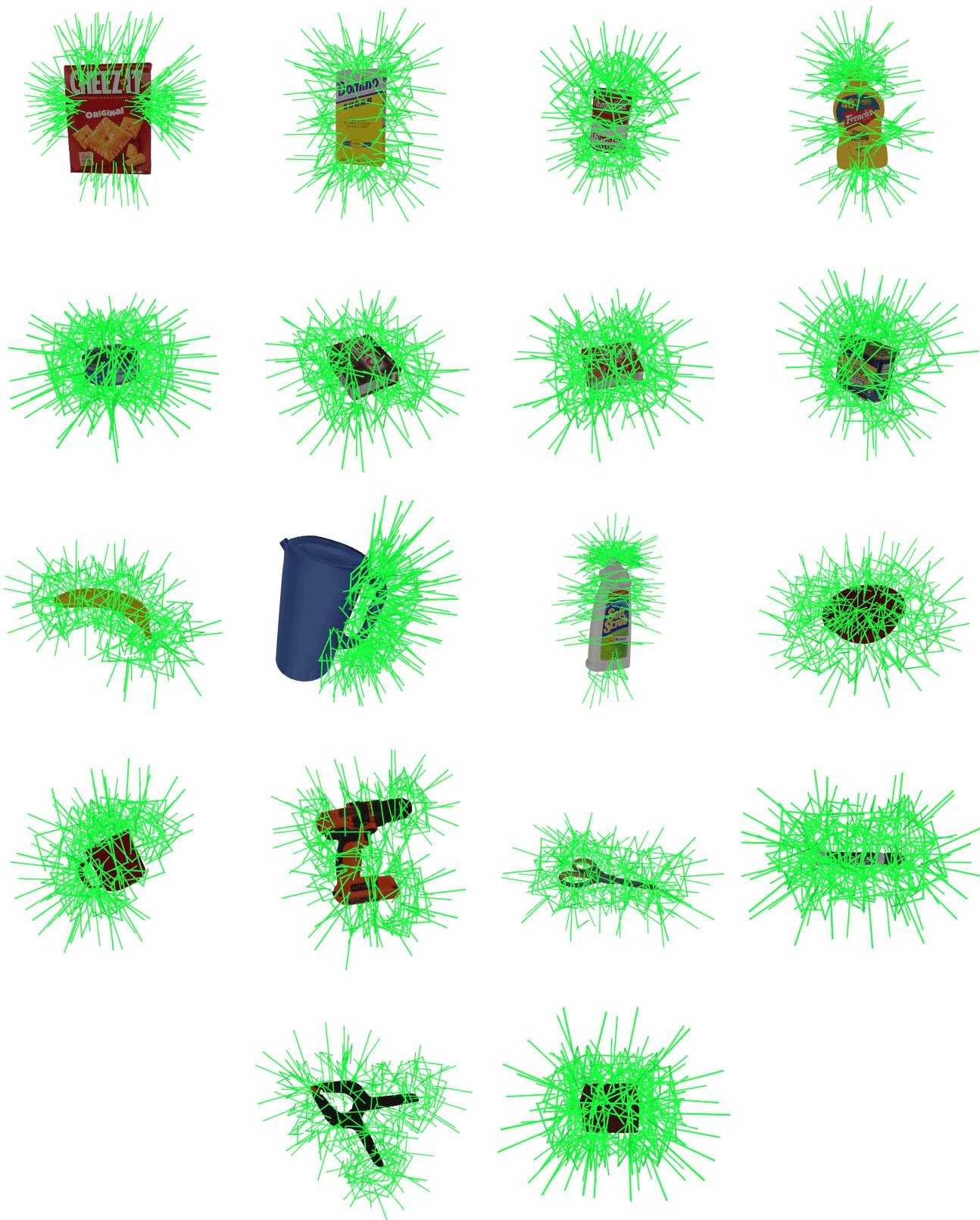


Figure 14: The 100 pre-generated grasps for each object.

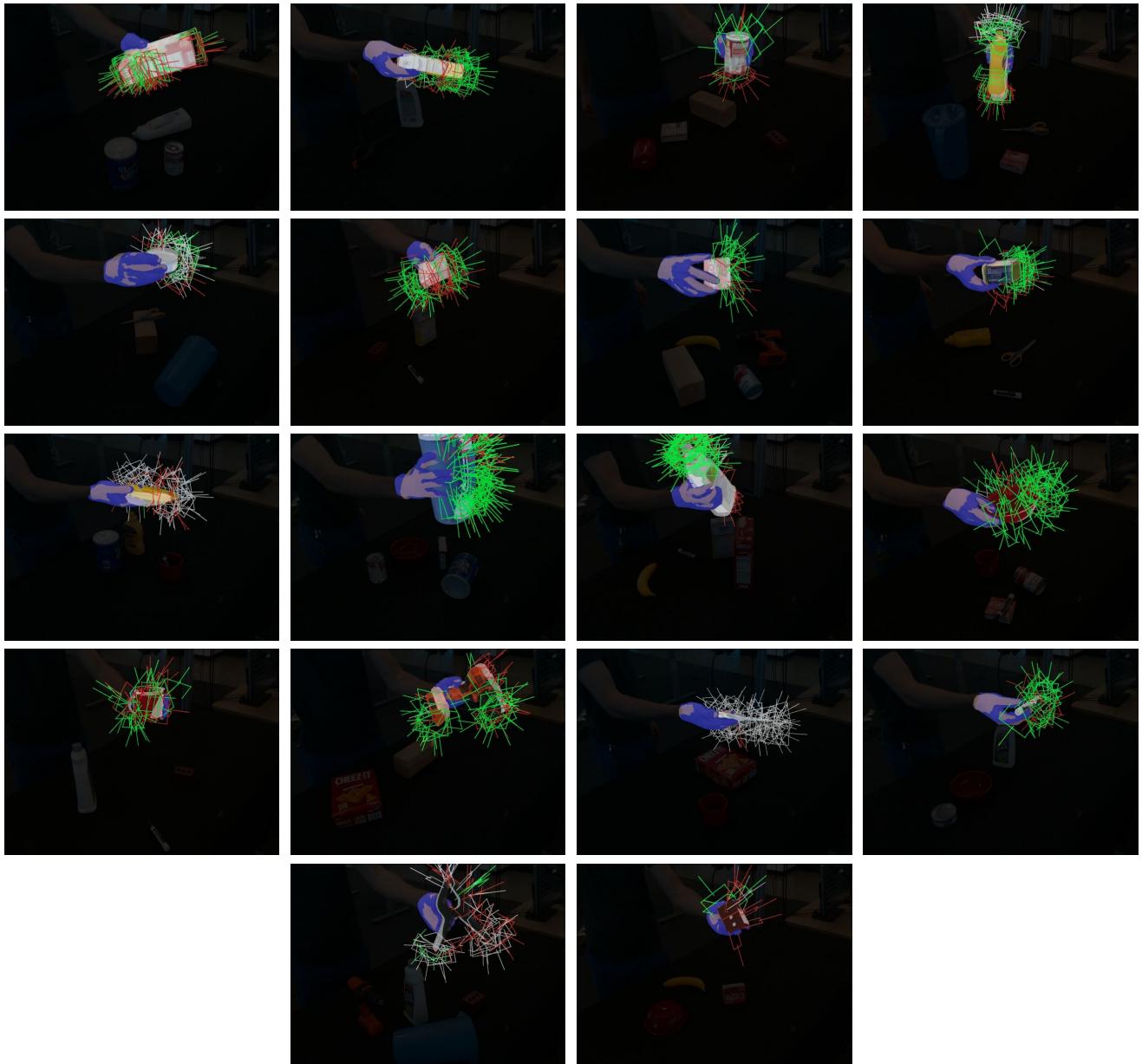


Figure 15: Additional qualitative results of the predicted grasps. Green ones denote those covering successful grasps, red ones denote those collided with the object or hand, and gray ones are failures not covering any successful grasps in the reference set. Predicted object poses are visualized by textured models and hand segmentations are highlighted by blue masks. Ground-truth objects and hands are shown in translucent white and brown meshes.