DexYCB: A Benchmark for Capturing Hand Grasping of Objects Supplementary Material

Yu-Wei Chao[1] Wei Yang[1] Yu Xiang[1] Pavlo Molchanov[1] Ankur Handa[1] Jonathan Tremblay[1]
Yashraj S. Narang[1] Karl Van Wyk[1] Umar Iqbal[1] Stan Birchfield[1] Jan Kautz[1] Dieter Fox[1,2]

[1]NVIDIA, [2]University of Washington

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

Contents

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 of the main paper). 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

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

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. 1. 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. 1 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. 1, 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:

$$q = u \cdot a + v \cdot b + w \cdot c, \tag{2}$$

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

$$\begin{aligned} E_{\text{depth}}(P) &= \frac{1}{N_D} \sum_{i=1}^{N_D} |\text{SDF}(d_i, \mathcal{M}(P))|^2 \\ &= \frac{1}{N_D} \sum_{i=1}^{N_D} ||d_i - q_i||_2^2. \end{aligned} \tag{3}$$

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

We can thus derive the gradient as

$$\frac{\partial E_{\text{depth}}(P)}{\partial P} = \frac{1}{N_D} \sum_{i=1}^{N_D} \frac{\partial ||d_i - q_i||_2^2}{\partial P}. \tag{4}$$

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

$$\frac{\partial ||d - q||_2^2}{\partial P} = \sum_{n=1}^{N_V} \sum_{m \in \{x,y,z\}} \frac{\partial ||d - q||_2^2}{\partial \mathcal{V}_{n,m}} \cdot \frac{\partial \mathcal{V}_{n,m}}{\partial P}. \tag{5}$$

With the barycentric representation of q from Eq.2, we can write

$$\begin{aligned} ||d - q||_2^2 &= ||d - (u \cdot a + v \cdot b + w \cdot c)||_2^2 \\ &= (d_x - u \cdot a_x - v \cdot b_x - w \cdot c_x)^2 \\ &\quad + (d_y - u \cdot a_y - v \cdot b_y - w \cdot c_y)^2 \\ &\quad + (d_z - u \cdot a_z - v \cdot b_z - w \cdot c_z)^2. \end{aligned} \tag{6}$$

Finally, with Eq. 5 and 6, we can derive the gradient with respect to the vertices as

$$\frac{\partial ||d - q||_2^2}{\partial \mathcal{V}_{n,m}} = \begin{cases} -2 \cdot u \cdot (d_x - u \cdot a_x - v \cdot b_x - w \cdot c_x) & \text{if } \mathcal{V}_{n,m} = a_x, \\ -2 \cdot u \cdot (d_y - u \cdot a_y - v \cdot b_y - w \cdot c_y) & \text{if } \mathcal{V}_{n,m} = a_y, \\ -2 \cdot u \cdot (d_z - u \cdot a_z - v \cdot b_z - w \cdot c_z) & \text{if } \mathcal{V}_{n,m} = a_z, \\ -2 \cdot v \cdot (d_x - u \cdot a_x - v \cdot b_x - w \cdot c_x) & \text{if } \mathcal{V}_{n,m} = b_x, \\ -2 \cdot v \cdot (d_y - u \cdot a_y - v \cdot b_y - w \cdot c_y) & \text{if } \mathcal{V}_{n,m} = b_y, \\ -2 \cdot v \cdot (d_z - u \cdot a_z - v \cdot b_z - w \cdot c_z) & \text{if } \mathcal{V}_{n,m} = b_z, \\ -2 \cdot w \cdot (d_x - u \cdot a_x - v \cdot b_x - w \cdot c_x) & \text{if } \mathcal{V}_{n,m} = c_x, \\ -2 \cdot w \cdot (d_y - u \cdot a_y - v \cdot b_y - w \cdot c_y) & \text{if } \mathcal{V}_{n,m} = c_y, \\ -2 \cdot w \cdot (d_z - u \cdot a_z - v \cdot b_z - w \cdot c_z) & \text{if } \mathcal{V}_{n,m} = c_z, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Since we can also compute $\partial \mathcal{V}_{n,m} / \partial P$ for rigid objects in Eq. 5, 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. 1 with both our object and hand models. Similar to the forward computation, the backward pass can also be parallelized using a GPU implementation.

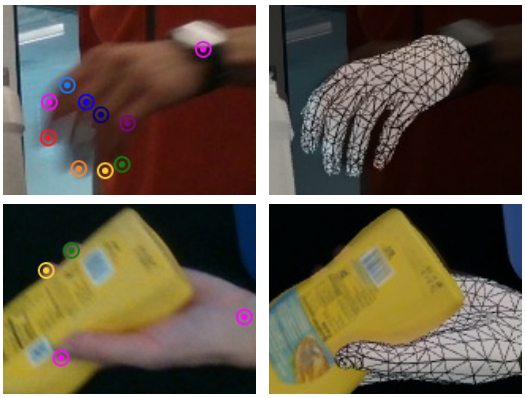


Figure 1: Qualitative examples of annotations with high reprojection errors.

B. Properties of DexYCB

B.1. Visualization

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

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. 1 reports the reprojection errors of the five finger tips. The mean errors are all below 5 pixels. As shown in Fig. 1, 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. 2 compares the distribution of hand pose in DexYCB with HO-3D [3]. 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.

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

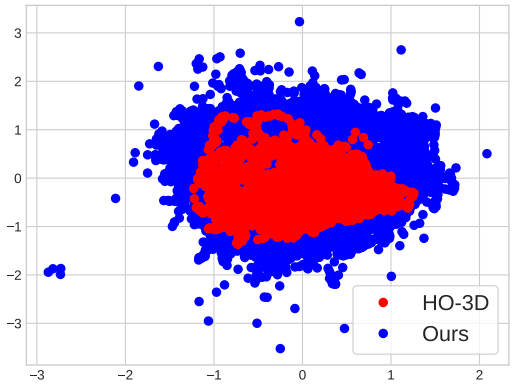


Figure 2: Distribution of hand pose represented by the first two MANO PCA coefficients.

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

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

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 of the main paper.

C. Benchmarking Representative Approaches

C.1. Setup Details

Tab. 2 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 ("#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 chal-

lenge 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. 6 shows qualitative results of 2D object detection and keypoint detection with Mask R-CNN (Detectron2) [4, 9] 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. 7 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 leftmost example of PoseCNN [10] (RGB)). We also see that refinement based approaches like DeepIM [6] and CosyPose [5] 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. 8 shows qualitative results of 3D hand pose estimation using Spurr et al.'s method [8] (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 of the main paper, 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 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 the main paper (Tab. 4) we report benchmark results only on S1, we now include the results on the other three setups. Tab. 3, 4, and 5 show the results in AR on S0, S2, and S3, respectively. Overall, we observe a similar trend as on S1 (see Sec. 5.3 of the main paper), where DeepIM (RGB-D) [6] and CosyPose [5] are the two most competitive approaches in estimation accuracy.

D. Safe Human-to-Robot Object Handover

D.1. Pre-Generated Grasps

Fig. 9 visualizes the 100 grasps for each object used in our evaluation. These grasps are sampled from the pre-generated grasps for YCB objects in [2]. 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. 10 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 the main paper (Fig. 4) we show the results of grasp generation only on S1, we now include the results on the other three setups. Fig. 4 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) [6] and CosyPose [5] achieved very close performance on 6D object pose (e.g. 57.54 versus 57.43 AR in Tab. 4 of the main paper), the later performs significantly better

	S0 (default)						
	PoseCNN [10]		DeepIM [6]		PoseRBPF [1]		CosyPose [5]
	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 3: 6D object pose estimation results of representative approaches in AR (%) on S0.

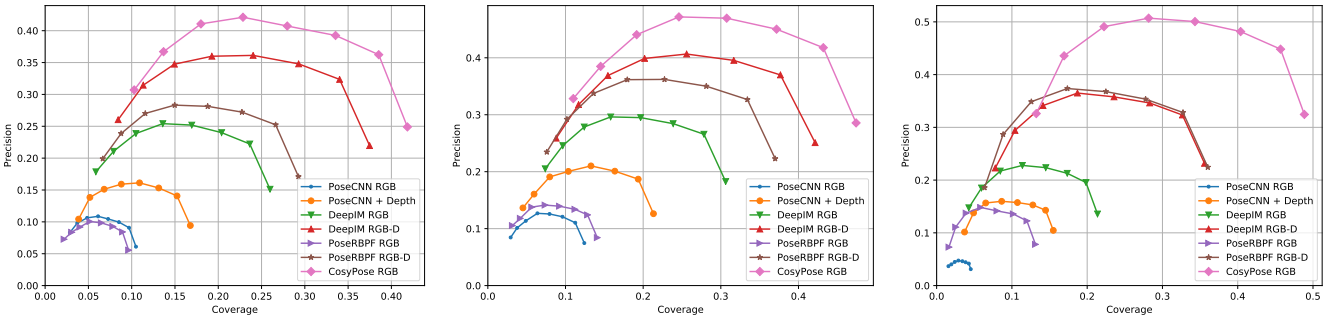
	S2 (unseen views)						
	PoseCNN [10]		DeepIM [6]		PoseRBPF [1]		CosyPose [5]
	RGB	+ depth ref	RGB	RGB-D	RGB	RGB-D	RGB
002_master_chef_can	51.36	57.42	68.86	72.87	37.64	63.04	**81.94**
003_cracker_box	60.65	67.45	84.10	89.34	54.25	74.63	**90.66**
004_sugar_box	51.73	60.06	69.68	76.23	47.15	71.17	**83.11**
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 4: 6D object pose estimation results of representative approaches in AR (%) on S2.

on grasp generation (e.g. see Fig. 4 of the main paper). 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 AR does not necessarily translate to a higher performance on downstream

	S3 (unseen grasping)						
	PoseCNN [10]		DeepIM [6]		PoseRBPF [1]		CosyPose [5]
	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 5: 6D object pose estimation results of representative approaches in AR (%) on S3.



S0 (default) S2 (unseen views) S3 (unseen grasping)

Figure 4: Precision-coverage curves for grasp generation on S0, S2, and S3.

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 RGB images of DexYCB are not expected to generalize well to in-the-wild images. We tested a DexYCB-trained model on COCO images [7] and observed an expected drop in performance. Fig. 5 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.

References

[1] 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. 5, 6, 9

[2] 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. 4

[3] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. HOnnotate: A method for 3D annotation of hand and object poses. In *CVPR*, 2020. 2, 3

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 4, 8

Figure 5: Samples of 2D hand and keypoint detection on COCO.

[5] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. CosyPose: Consistent multi-view multi-object 6D pose estimation. In *ECCV*, 2020. 4, 5, 6, 9

[6] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep iterative matching for 6D pose estimation. In *ECCV*, 2018. 4, 5, 6, 9

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 6

[8] Adrian Spurr, Umar Iqbal, Pavlo Molchanov, Otmar Hilliges, and Jan Kautz. Weakly supervised 3D hand pose estimation via biomechanical constraints. In *ECCV*, 2020. 4, 10

[9] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 4, 8

[10] 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. 4, 5, 6, 9

[11] 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

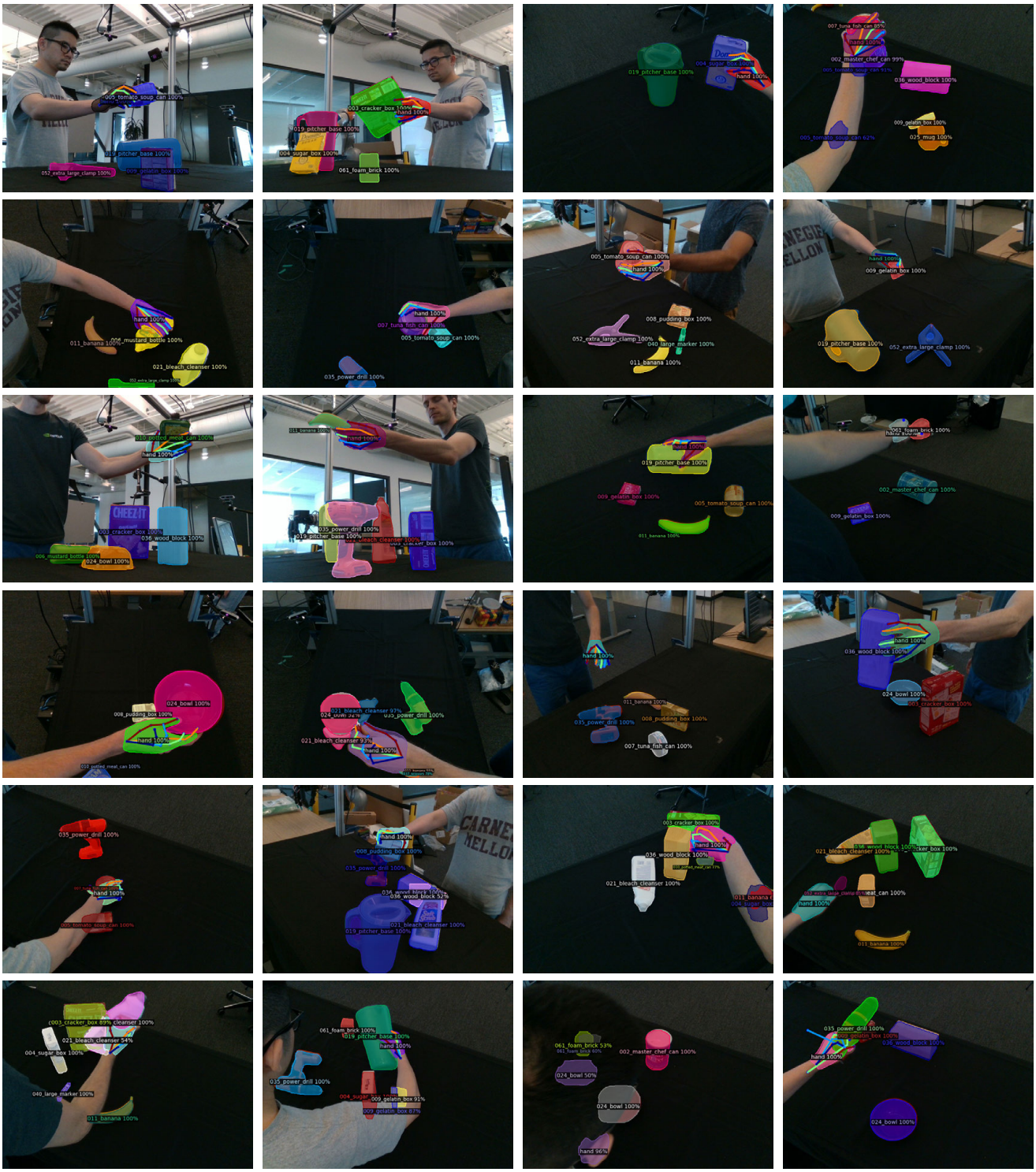


Figure 6: Qualitative results of 2D object and keypoint detection with Mask R-CNN (Detectron2)) [4, 9]. The last two rows highlight failure examples.

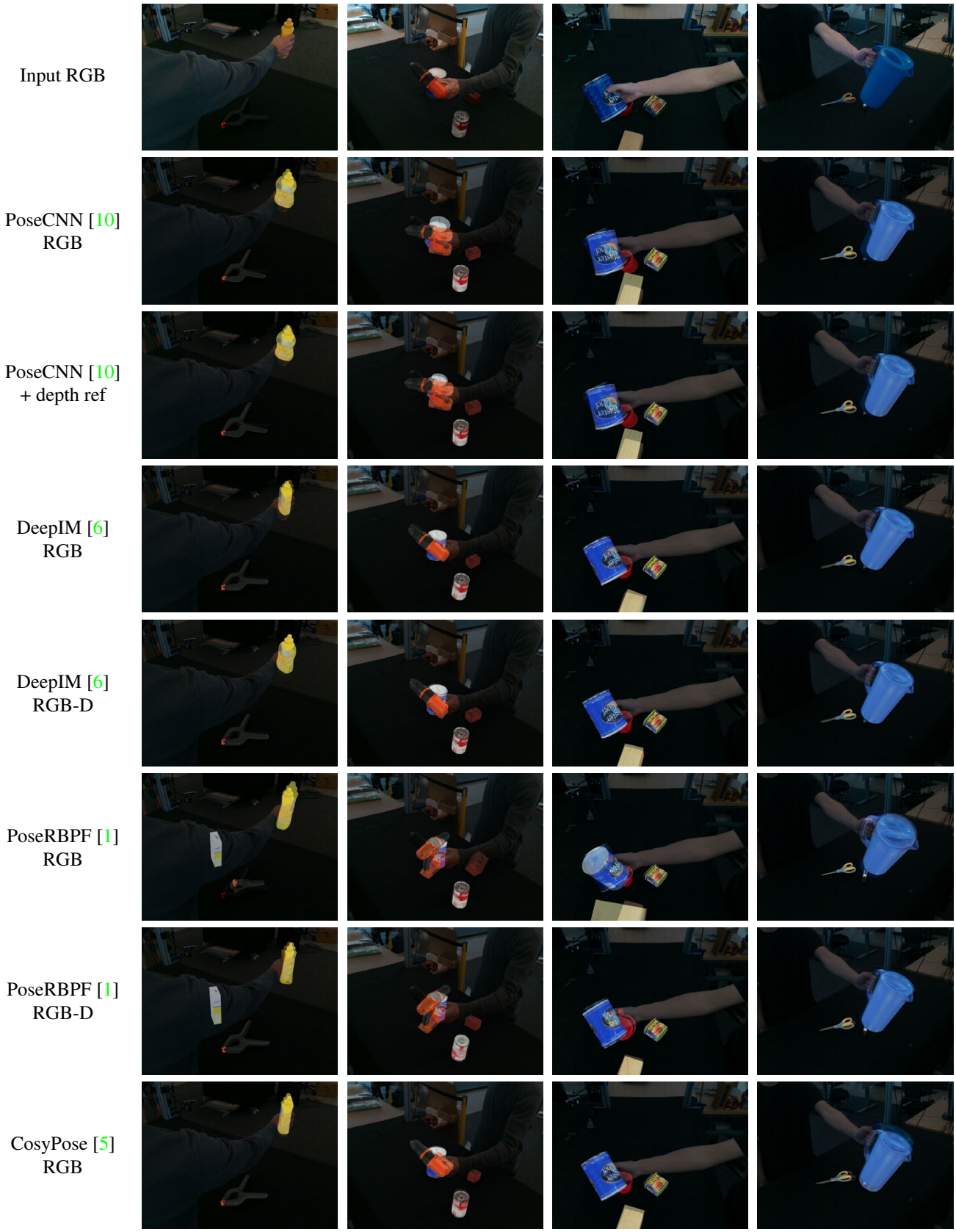


Figure 7: Qualitative results of 6D object pose estimation. We render object models given the estimated poses on a darkened input image.

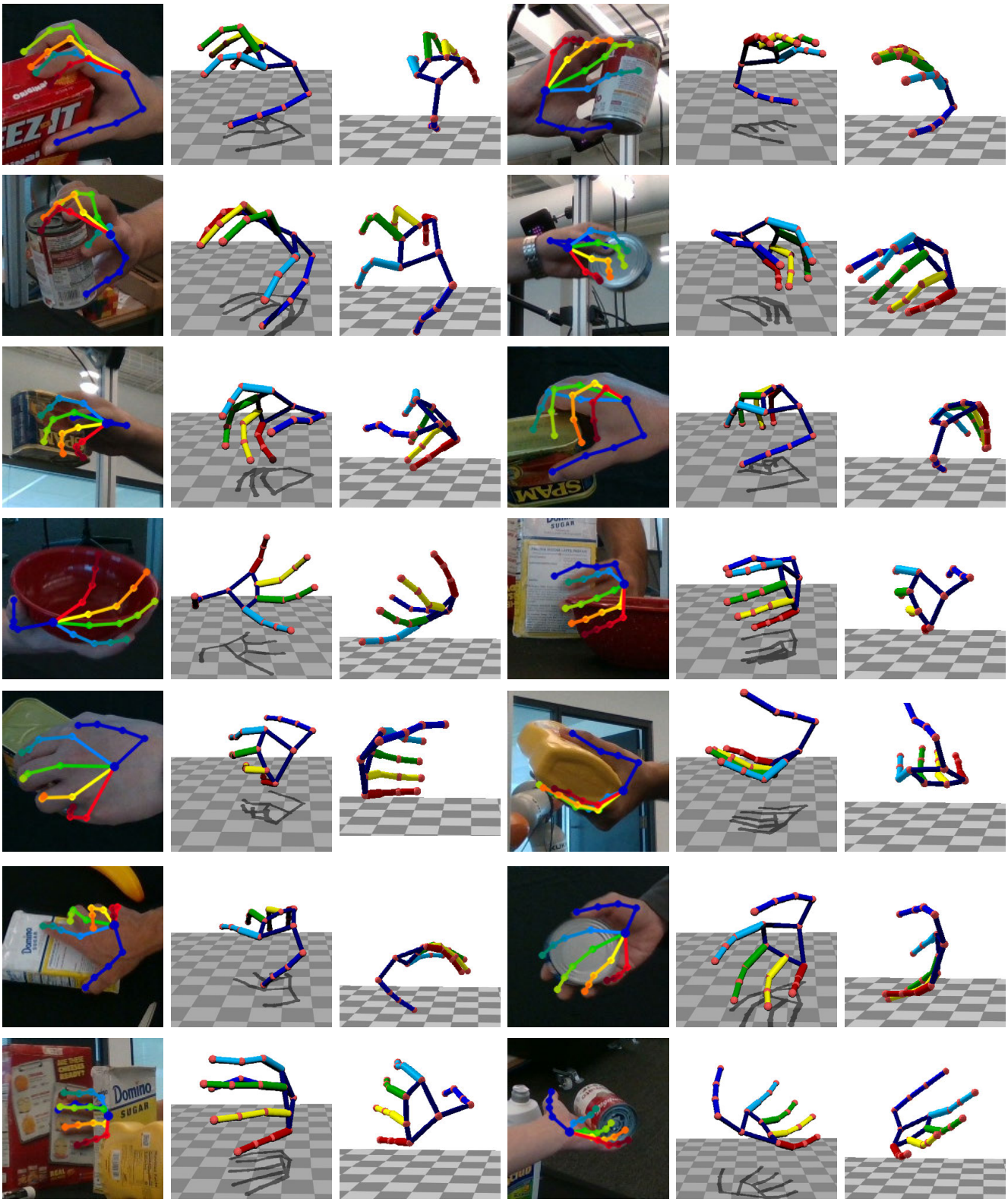


Figure 8: Qualitative results of the predicted 3D hand pose using Spurr et al.'s method [8] (HRNet32). For each image we visualize 3D pose from both front and side views.

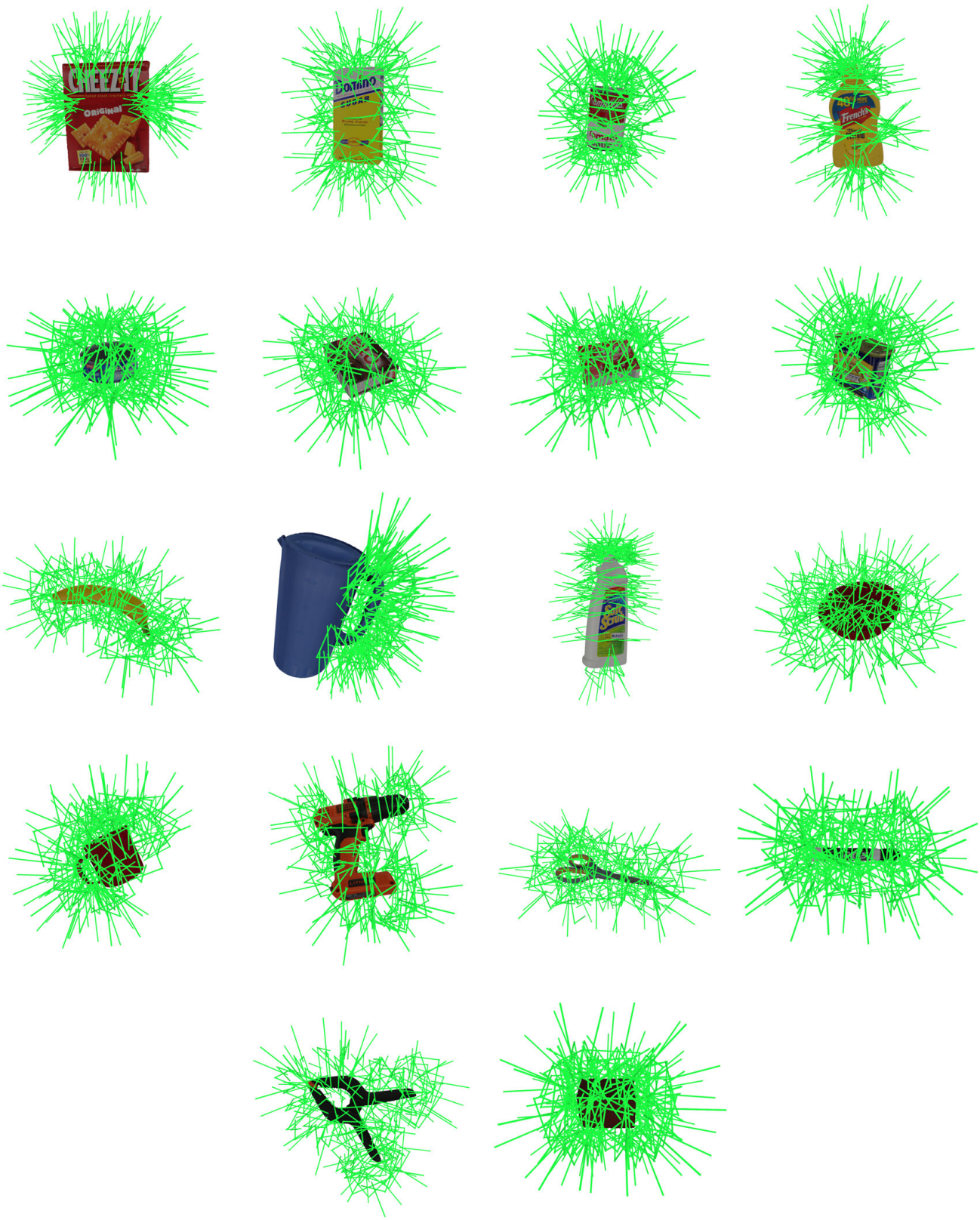


Figure 9: The 100 pre-generated grasps for each object.

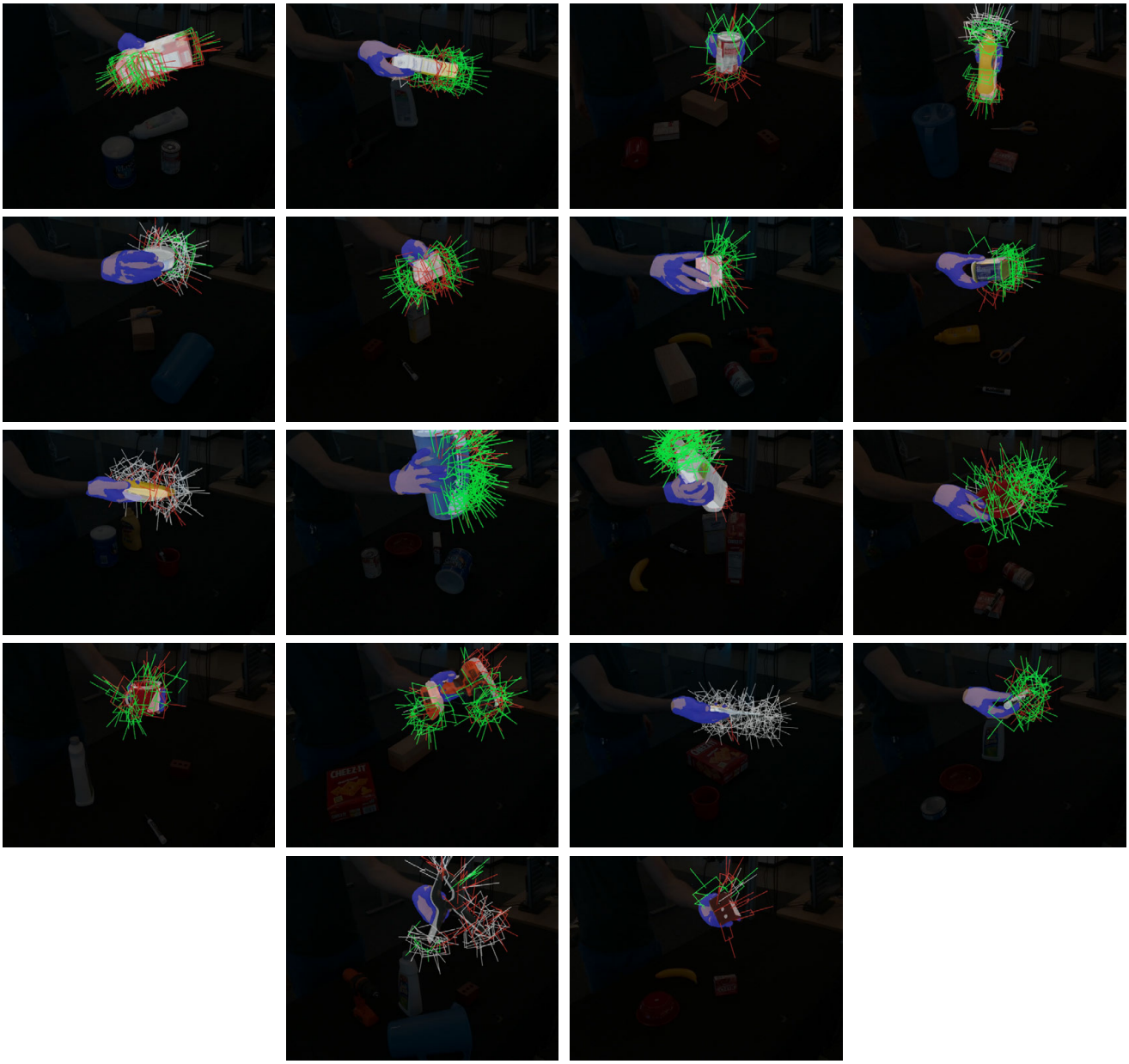


Figure 10: Additional qualitative results of the predicted grasps. Green ones denote those covering successful grasps, red ones denote those collided with the object or hand, and gray ones are failures not covering any successful grasps in the reference set. Predicted object poses are visualized by textured models and hand segmentations are highlighted by blue masks. Ground-truth objects and hands are shown in translucent white and brown meshes.