Digital Twin Catalog: A Large-Scale Photorealistic 3D Object Digital Twin Dataset

Supplementary Material

We provide the following content in the supplementary materials:

- We provide a comprehensive dataset statistics of DTC dataset in Sec. A, including the full category of DTC objects, the list of objects used in DLSR and egocentric recording respectively. We also provided visualizations for examples of DTC objects, DSLR and egocentric data.
- We include complementary details of benchmark in Sec. B. First, as indicated by the main paper, we include a sparse view setting of benchmark. We further provide the baseline comparisons and analysis for DSLR benchmark and egocentric benchmark. We use the same baselines and dataset split as described in the main paper.
- We provide details of the simulation and experiments for our robotics experiments in Sec. C.

A. DTC Dataset Statistics & Details

In this section, we present the category list of DTC 3D objects (Sec. A.1) and showcase additional examples of DTC digital-twin-quality 3D objects (Fig. 9). We also provide the list of selected objects used for creating the DSLR evaluation dataset (Sec. A.2) and the egocentric evaluation dataset (Sec. A.3). Furthermore, we include additional examples of real-world captures and recordings to better illustrate the DSLR (Sec. A.4) and Egocentric (Sec. A.5) evaluation datasets.

A.1. Categories of DTC Scanned Objects

In DTC dataset, we selected and scanned 2,000 physical-world objects across 40 carefully curated categories from the taxonomy of LVIS [27]. These categories were chosen to ensure a diverse representation of common daily objects while remaining compatible with the scanner's capabilities. Table 5 provides an overview of these categories, including the number of models, average vertex count, minimum/maximum vertex numbers for each category and corresponding category label in LVIS taxonomy.

For each scanned model, we generated high-quality 4K-resolution PBR material maps—including albedo, roughness, metallic, and normal maps—to achieve a photorealistic appearance. Fig. 9 showcases additional examples of DTC scanned digital-twin-quality 3D models, each accompanied by a full set of PBR material maps (Fig. 10, 11, 12, 13, 14).

Category	# Models	avg # vert / std	min / max	LVIS Category
airplane	15	129,153 / 15,535	80,719 / 144,587	airplane
axe	3	133,588 / 1,922	132,096 / 136,301	ax
basketball	55	132,260 / 3,013	128,652 / 140,985	basketball
birdhouse	102	147,577 / 11,057	98,405 / 182,594	birdhouse
bowl	106	127,456 / 1,019	125,883 / 130,964	bowl
building blocks	129	96,592 / 46,377	28,181 / 301,475	toy
calculator	38	131,244 / 3,638	126,379 / 139,970	calculator
candle	31	108,221 / 58,406	15,897 / 309,601	candle
candle holder	1	132,285 / 0	132,285 / 132,285	candle_holder
cast iron	31	165,954 / 127,424	128,074 / 711,536	pan
cup	58	137,154 / 43,605	126,346 / 417,343	cup
cutting board	16	133,796 / 11,244	127,543 / 173,141	chopping_board
dino	103	82,403 / 29,477	13,796 / 151,148	animal
dish	51	129,732 / 4,130	127,014 / 147,451	dish
dumbbell	39	156,915 / 165,650	126,593 / 1,177,907	dumbbell
fake food can	79	117,829 / 22,772	78,646 / 270,271	can
fakefruit	96	124,037 / 9,724	100,833 / 151,870	fruit
figurine	77	57,152 / 47,753	15,482 / 140,812	figurine
football	48	132,323 / 2,953	126,719 / 138,459	football
gargoyle	50	137,921 / 5,019	130,206 / 151,325	gargoyle
gravestone	24	85,245 / 55,760	10,750 / 150,966	gravestone
hammer	33	133,413 / 36,272	46,808 / 320,845	hammer
hardcover_book	17	174,884 / 102,168	130,267 / 500,735	hardback_book
key	2	60,456 / 41,100	19,357 / 101,556	key
keyboard	25	146,684 / 12,414	129,360 / 174,488	computer_keyboard
knife	10	126,359 / 8,118	102,569 / 133,250	knife
mallard (fake duck)	48	99,728 / 45,926	9,000 / 131,058	mallard
marker	54	93,394 / 39,460	34,796 / 306,272	marker
miscellaneous	44	149,799 / 168,694	27,772 / 1,208,696	NA
mouse	52	144,184 / 46,099	121,879 / 304,533	mouse
pistol	2	133,587 / 91	133,496 / 133,678	pistol
pottery	46	143,284 / 162,783	12,354 / 1,206,223	pottery
remote	2	42,993 / 17,068	25,925 / 60,061	remote_control
shampoo	45	143,639 / 75,261	126,285 / 604,491	shampoo
shaver	20	142,231 / 57,736	38,246 / 306,105	shaver
shoes	121	139,363 / 4,171	130,604 / 148,465	shoe
speaker	40	201,013 / 152,410	127,154 / 852,714	speaker
spoon	34	107,649 / 56,459	29,712 / 302,920	spoon
teapot	99	145,149 / 100,792	85,289 / 926,273	teapot
vase	101	142,927 / 142,946	61,488 / 1,568,710	vase
volleyball	52	133,921 / 4,056	129,653 / 144,770	volleyball

Table 5. Categories of DTC Scanned Objects. We include the number of models per object (#Models), the average number of vertices per object categories (avg # vert) and its standard deviation (std), the minimum and maximum number of vertices within the object category (min/max), and the label name in LVIS category taxonomy.

A.2. List of Models for DSLR Evaluation Data

The full list of object models selected for DSLR evaluation data is as follows, and each model is captured under 2 different environment lighting conditions. For the 15 object recordings that are used in our benchmark evaluations, we highlight them in **bold**.

- `Airplane_B097C7SHJH_WhiteBlue`
- `Airplane_B0B2DC5QBP_BlueGray`
- **`BirdHouse`**
- `BirdHouse_B0B8F27TFK_BrownRoofYellowWalls`
- `BirdHouseRedRoofYellowWindows`
- `BirdHouseWoodenRoofGreenWall`
- `Bowl_B0BQR77WRW_LightGrey_1_TU`
- `Box_ADTIR_DecorativeBoxHexLarge_Green`
- `Car_38330969_Toy`
- `CaramicBowlBluewithBrown`
- `CeramicBowlBigWhite`
- **`Cup_B08TWHJ33Q_Tan`**
- `Cup_B0B3JKZW76_Brown`

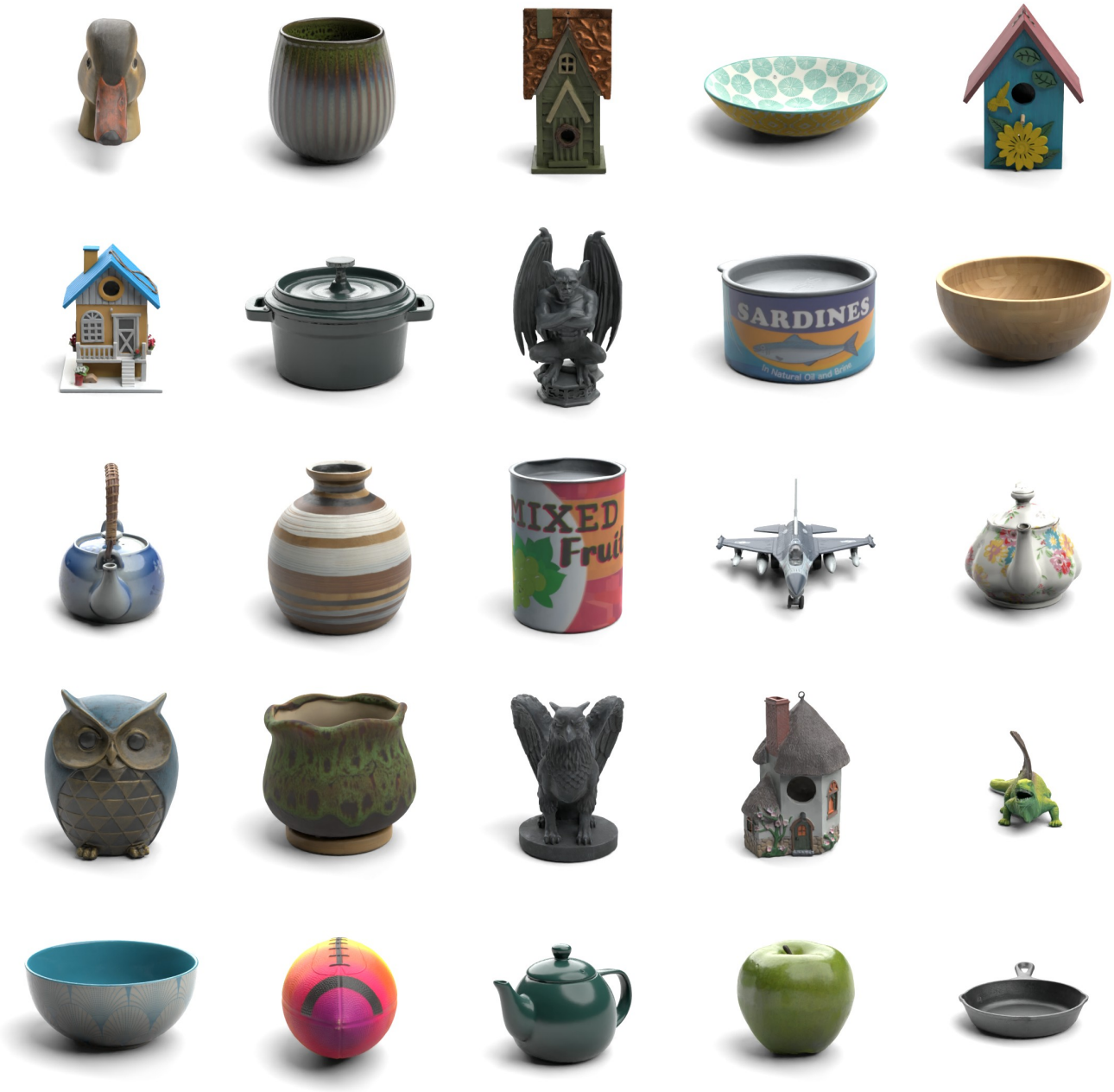

Figure 9. Examples of DTC 3D models. The PBR Materials of each object are presented in the following figures from Fig. 10 to Fig. 14.

- Cup_B0CQXPND8L_Stripes
- Dutch_Oven_B0B916N11D_Black
- **Figurine_B08FYFNYP4_LionKing**
- Figurine_B0983CQ2HH_Angel
- Figurine_B0CR3Y5T3K_Gnome
- Gargoyle_B005KDPAFW_BatWings
- **Gargoyle_B08SQMBDXY_HandsOnKnees**
- **Gargoyle_B0C2PNF2C1_Meditating**
- Gravestone_B08TBJQ5XP_LightGrayKitty
- Hammer_B000FK3VZ6_Wood
- Home_ADTIR_A6116F6_DraganBoxSmall_Wood
- Home_ADTIR_L041OV8_Rinnig_PlateHolder
- Keyboard_B07P6K5GMY_Black
- Keyboard_B0CL8S2DW9_Pink
- Kitchen_Spoon_B008H2JLP8_LargeWooden
- **Mallard_B082D168CK_MintGreen**

Figure 10. PBR Materials of the example DTC objects (the list of objects in Fig.9 **Row 1**), From left to right: albedo map, roughness map, metallic map, normal map, and PBR rendering.

- **Mallard_B09LV16HD5_LightBrown2**
- **Mallard_B0BPY18VHR_White**
- Mallard_B0C6MQWM21_BlackWhite
- **Mouse_B0CHNVBBLF_Honeycomb_1**
- Pan_B0CFQWYJZ8_BlackWoodHandle
- Pan_B0CHW1KK8Z_Black
- **Planter_B0C4G81ZPF_Cat**
- Pottery_B097S319TR_Woman
- **Pottery_B0CJJ59SLH_BlueHairFairy**
- **Shoe_B000ZP6MIY_Navy7L_TU**
- Spoon_B08M3XNKYR_Slotted
- TeaPot_B074ZQYRP7_BrownDragonShaped
- TeaPot_B07GL8MH3X_PinkFlamingo
- TeaPot_B07QP5MFQ1_BlackCastIron
- TeaPot_B084G3K8TD_YellowBlackSunflowers
- **TeaPot_B08HSDHBM4_BlackGoldLeaves**
- **TeaPot_B00ESU7PFG_WhiteRoseFlowers**
- **TeaPot_B01KFCZB2Y_WhiteWoodHandle**

Figure 11. PBR Materials of the example DTC objects (the list of objects in Fig.9 **Row 2**). From left to right: albedo map, roughness map, metallic map, normal map, and PBR rendering.

- `Vase_B09ZGXSVTT_White_TU`
- `Vase_B0BV44B4R4_BlueBirdsYellowBirds`
- `Vase_Corrected`

A.3. List of Models for Egocentric Evaluation Data

The full list of models selected that contains pairs of egocentric data is as follows, and each model is captured with both *active* and *passive* trajectories. For the 15 object recordings that are used in our benchmark evaluations, we highlight them in **bold**.

- `Airplane_B097C7SHJH_WhiteBlue`
- `Airplane_B09WN2RN15_Black_1`
- `BasketPlasticRectangular`
- `BirdHouseToy`
- **`BirdHouse_B0B8F27TFK_BrownRoofYellowWalls`**
- `BirdHouse_B004HJE8AS_WhiteWallsTwoPorches_2`
- `BirdHouseRedRoofYellowWindows`
- `BirdHouseWoodenRoofGreenWall`

Figure 12. PBR Materials of the example DTC objects (the list of objects in Fig.9 **Row 3**). From left to right: albedo map, roughness map, metallic map, normal map, and PBR rendering.

- BirdHouse_B08GYBKJ8N_RedBarn
- Birdhouse_B09FJYJYDQ_BoatHouse
- BirdHouseMetalRoofYellowWall_1_TU
- BirdHouse_A79823645_BearWithLogStump
- BirdHouseWoodenRoofRedWall
- BlackCeramicDishLarge
- BlackCeramicMug
- Block_B007GE75HY_RedBlue
- Bowl_B0BQR77WRW_LightGrey_1_TU
- Bowl_B07ZNJ5RQV_Orange_TU
- CandleDishSmall
- CrateBarrelBowlRed
- **CeramicBowlBigWhite**
- CelebrateBowlPink
- Car_38330969_Toy
- CaramicBowlBluewithBrown
- Candle_B0B2JQWNNQ_White
- Candle_B0B764F39X_Turquoise

Figure 13. PBR Materials of the example DTC objects (the list of objects in Fig.9 **Row 4**). From left to right: albedo map, roughness map, metallic map, normal map, and PBR rendering.

- Candle_B09MP8NDML_White
- Calculator_B0C7GP2D5C_Purple
- **Cup_B08TWHJ33Q_Tan**
- Cup_B0B3JKZW76_Brown
- Cup_B0CQXPND8L_Stripes
- Cup_B08TWHJ33Q_Gray
- Cup_B09QCYR1SL_Pitcher_1
- Cup_B01LYONYPB_SkyBlue
- Dumbbell_B00PY62Y90_Black
- Dumbbell_B0CN56CQPS_PurpleCoolGray
- Dutch_Oven_B0B916N11D_Black
- FakeFruit_B076H96CS1_Banana
- FakeFruit_B09992T572_WatermelonSlice
- FakeFruit_B0815W7RKC_StarFruit
- Figurine_B08FYFNYP4_LionKing
- **Figurine_B0983CQ2HH_Angel**
- Figurine_B0CR3Y5T3K_Gnome
- **Flask**

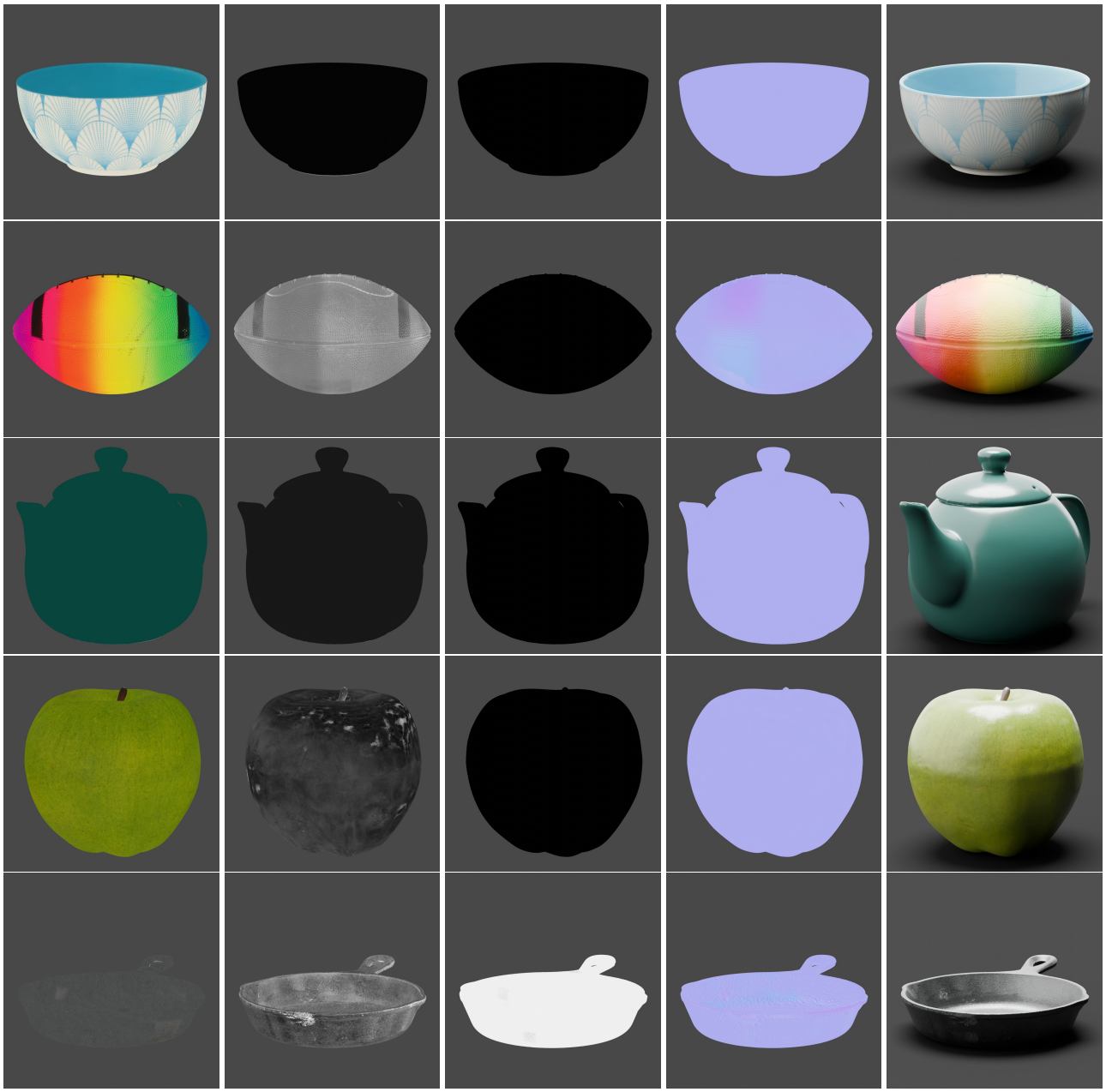

Figure 14. PBR Materials of the example DTC objects ((the list of objects in Fig.9 **Row 5**)), From left to right: albedo map, roughness map, metallic map, normal map, and PBR rendering.

- Gargoyle_B07GHVQ3C4_BatCat
- **Gargoyle_B08SQMBDXY_HandsOnKnees**
- Gargoyle_B0C3RQ5254_Bronze
- Gargoyle_B0BYTQT173_Dragon
- Gargoyle_B0C2PNF2C1_Meditating
- Gargoyle_B00N08IU24_Dog
- Gravestone_B08TBJQ5XP_LightGrayKitty
- Gravestone_B07WDGH3NR_GrayAngel
- Hammer_B000FK3VZ6_Wood
- Hammer_B0BN6FXDQ7_BlueHandle
- Hammer_B01N63ONKY_DrillingSledge
- Kitchen_Cup_B09G2WNN61_DarkBlue
- Kitchen_Spoon_B008H2JLP8_LargeWooden
- Kitchen_Spoon_D146567C_Green_1
- LargeLightGreyDish
- Mallard_B082D168CK_MintGreen
- **Mallard_B09LV16HD5_LightBrown2**
- **Mallard_B0BPY18VHR_White**

- Mallard_B0C6MQWM21_BlackWhite
- Mallard_B00RTSJU7K_Red
- Mouse_B0CHNVBBLF_Honeycomb_1
- Pan_B0CFQWYJZ8_BlackWoodHandle
- Pan_B0CHW1KK8Z_Black
- Planter_B0C4G81ZPF_Cat
- PlasticBowlGreen_1_TU
- **Pottery_B097S319TR_Woman**
- Pottery_B0CJJ59SLH_BlueHairFairy
- Pottery_B07TGC6TGL_White
- Pottery_B075SX9GVK_White
- Pottery_B0CF8FW987_Man
- **Shoe_B000ZP6MIY_Navy7L_TU**
- Shoe_B094ZCQK75_Red6HL_TU
- **Spoon_B08M3XNKYR_Slotted**
- TeaPot_B074ZQYRP7_BrownDragonShaped
- TeaPot_B07GL8MH3X_PinkFlamingo
- TeaPot_B07QP5MFQ1_BlackCastIron
- TeaPot_B084G3K8TD_YellowBlackSunflowers_TU
- **TeaPot_B08HSDHBM4_BlackGoldLeaves**
- **TeaPot_B00ESU7PFG_WhiteRoseFlowers**
- **TeaPot_B01KFCZB2Y_WhiteWoodHandle**
- TeaPot_B07RT7BYXL_WhiteSnowOwl_TU
- **Vase_B09ZGXSVTT_White_TU**
- Vase_B0BV44B4R4_BlueBirdsYellowBirds
- Vase_Corrected
- Vase_B0BNX2CWW8_YellowTall
- Vase_B00858OOXI_Green_1
- Vase_B09FYBCM1R_BeeSunflowers
- Vase_B0BY8PYLLC_PinkPineapple
- Vase_B0BNX2CWW8_YellowShort
- WhiteContainerBox
- WhiteClip_1
- WoodBlocks_B00FIX22YQ_BlueArch
- WoodBlocks_B098PHYN3P_SmallOctagon
- WoodenBoxSmall
- WoodenFork

A.4. An Example of DSLR Data

As described in Sec. 3.3, we used three DSLR cameras mounted on a rig to capture 360-degree inward-facing photos of each object. As shown in Fig. 15, these DSLR cameras were positioned at different elevation angles and captured images circularly around the object with a 9-degree interval between shots. Each camera captured 40 photos, resulting in a total of 120 photos per object. For each photo, we provide the corresponding camera pose and an accurate foreground object mask (2nd row overlaid on the original image in Fig. 15) to facilitate straightforward and reliable evaluation. Additionally, to support the evaluation of relightable reconstruction, we utilized a mirror ball to capture and reconstruct an HDR environment map (bottom row in Fig.15).

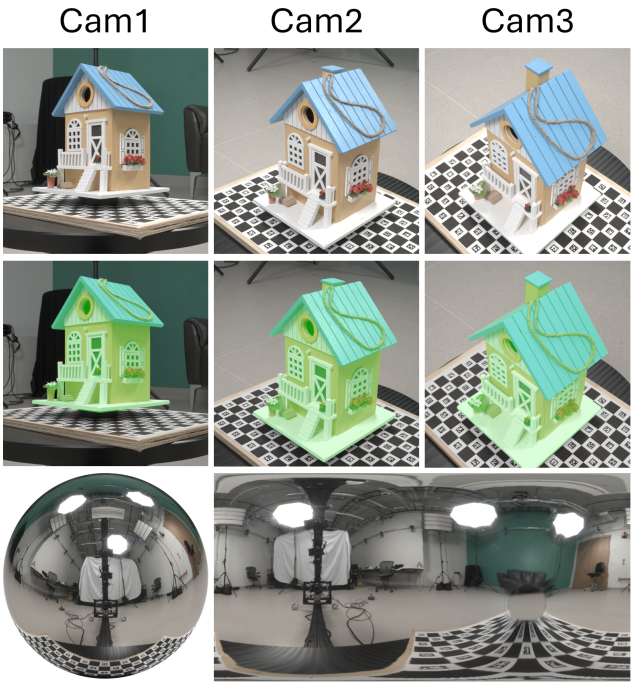

Figure 15. Example DSLR evaluation data: captured images with 3 DSLR cameras from different elevation angles (first row), the object mask overlaid on each image (second row) and the recovered HDR envmap (last row).

A.5. An Example of Egocentric Data

Fig. 16 provides an example visualization of the egocentric recording observed from its RGB camera and the rendered ground truth using the 3D aligned digital twin model using the method described in Sec. 3.4. The raw RGB input from Project Aria is a wide angle fisheye camera, which limits the amount of existing off-the-shelf baselines using pin-hole camera models. To resolve this limitation, we rectify all the raw images using a linear camera model, and render the corresponding ground truth of the linear camera model as well. All the evaluations on egocentric recordings are performed using the linear rectified images as input and output. We use the focal length 1200 and 2400x2400 resolution which retains the original pixel resolution and field of view from the raw image input.

It is worth noting the image input from Project Aria is gravity aligned. Both the raw image and rectified image will appear as 90 degree counter-clock wise rotated according to the gravity direction without explicitly updating the camera calibration information. For the reconstruction and evaluation, we do not rotate the image or update the calibration. We only rotate the rendered images 90 degree clockwise for the benefit of visualization purpose (e.g. Fig. 22)

We provided two different type of recordings to benchmark reconstruction from egocentric views. Fig. 17 shows a

Figure 16. An example of visualization for the egocentric recordings. From left to right, we present the original egocentric view from the fisheye RGB camera input, the rectified image view, object mask, rendered depth, and rendered normal. The egocentric recording are gravity aligned respect to the global trajectory coordinate. We use the rectified image for all evaluations.

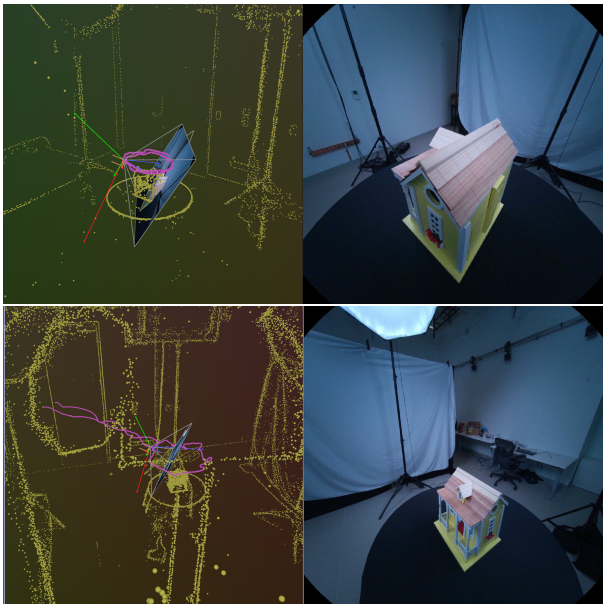

Figure 17. An snapshot visualization of the active (above) and passive (below) of egocentric recordings in the 3d space. The left of image shows the trajectory (visualized in Violet color) of the camera movement in each scenario. The yellow points are the scene point cloud.

snapshot of the egocentric recording. We demonstrate each type of camera trajectory overlaid in the 3D space using the scene point cloud. The active trajectory represents a common object-centric 360 view of the objects while the passive trajectory represents a more casual walk around the object from different distances.

B. Complementary Benchmarks

In this section, we provide complementary results on the different applications of benchmarking in Sec. 4. First, we include the qualitative comparisons of baselines in the inverse rendering application using the DTC dataset. We use the same baselines described in Sec. 4.1. Second, as indicated in the main paper, we include a new baseline evaluation for sparse-view reconstruction in Sec. B.2 for DSLR data and Sec. B.3 for egocentric data. Third, we present the qualitative comparisons of egocentric active reconstruction baselines described in Sec. 4.2.

Explanations on Evaluation Metrics. In Sec. 4.1, we evaluate the quality of *geometry reconstruction* by comparing the rendered depth maps, normal maps, and predicted 3D meshes to the ground truth. For depth maps, we use the Scale-Invariant Mean Squared Error (**SI-MSE**) as the evaluation metric. Normal maps are evaluated using **Cosine Distance**, and for the 3D meshes, we compute the **Bi-directional Chamfer Distance** between sampled surface points. For *novel view synthesis* and *novel scene relighting*, we utilize widely adopted metrics: Peak Signal-to-Noise Ratio for both HDR (**PSNR-H**) and LDR (**PSNR-L**), Structural Similarity Index Measure (**SSIM**) [67], and Learned Perceptual Image Patch Similarity (**LPIPS**) [86]. We follow PhySG [82] and adapt all these metrics to be scale-invariant, addressing the ambiguity in material and lighting decomposition. We use the same metrics for novel-view synthesis in the egocentric novel-view synthesis applications as well. The same evaluation metrics are used in StanfordORB [39].

B.1. Inverse Rendering Baseline Comparisons

We provide qualitative comparisons as a supplement to the inverse rendering baseline comparisons in Sec. 4.1. Our DTC dataset contains objects with varying geometric complexity, ranging from simpler ones, *e.g.*, cups (Fig. 18 Right), to complex ones, *e.g.*, toy birdhouses (Fig. 18 Left). The surface material ranges from diffuse ones, *e.g.*, toy houses, to highly reflective ones, *e.g.*, cups. For the two examples in Fig. 18, we observed that methods using surface-based representations, *i.e.*, PhySG and InvRender, and hybrid representations, *i.e.*, NVDiffRec and NVDiffRecMC, tend to produce smoother geometry compared to methods using NeRF [52]-based representations, *i.e.*, Neural-PIL and NeRD.

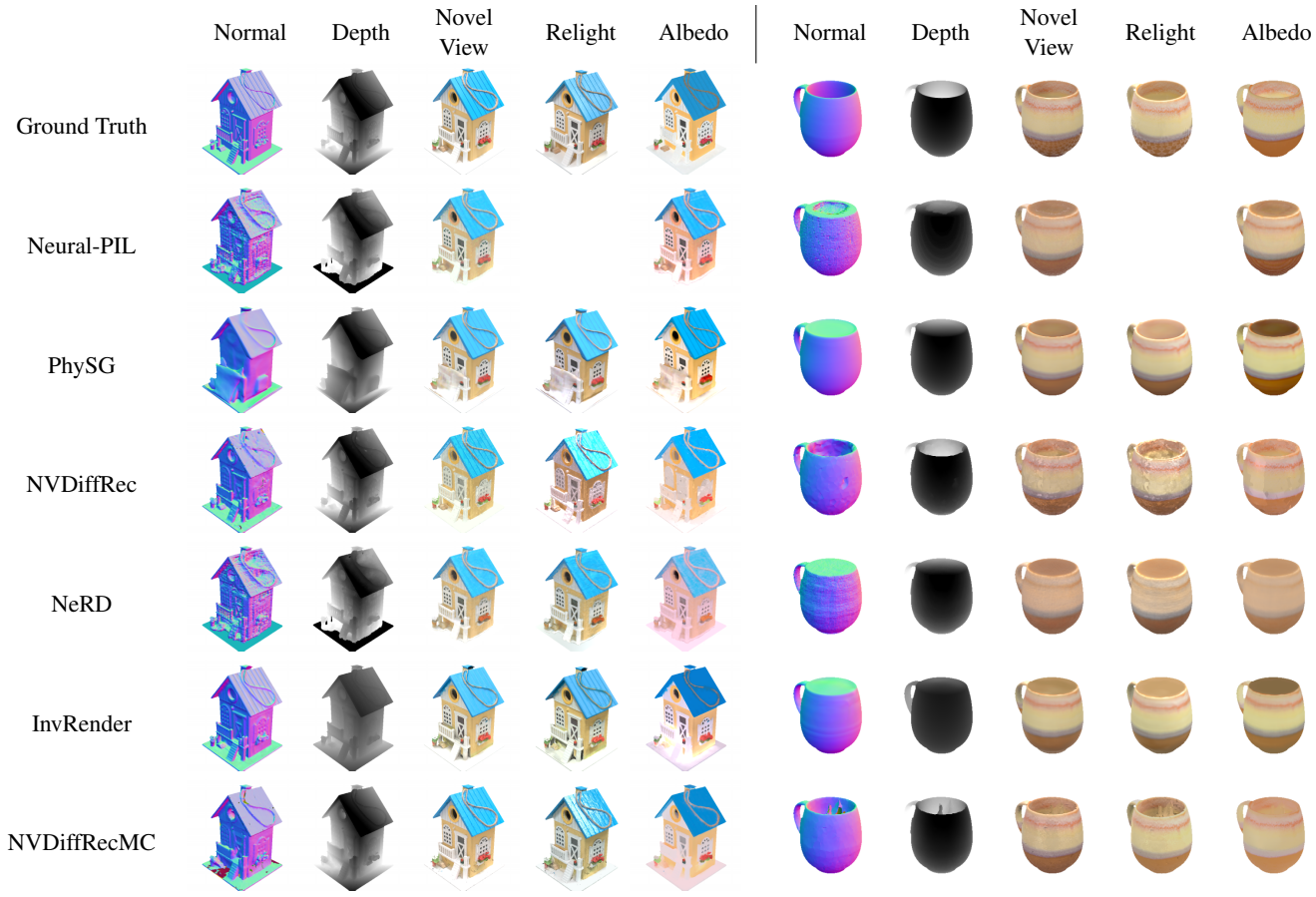

Figure 18. **Qualitative Comparisons of Baseline Methods**. We show qualitative comparisons of baseline methods, used in Table 2, on two different DTC models `BirdHouseToy` (Left) and `Cup_B08TWHJ33Q_Tan` (Right).

Table 6. Quantitative evaluation of trained LRM-VolSDF baseline for view synthesis on **GSO** dataset with different number of images. It achieves the state-of-the-art performance compared to prior work.

LRM-VolSDF	PSNR (↑)	SSIM (↑)	LPIPS (↓)
MeshLRM [70]	28.13	0.923	0.093
4 images	28.72	0.940	0.070
8 images	30.19	0.947	0.061

Table 7. Quantitative results on novel view synthesis for sparse-view reconstruction using DTC DSLR data.

	PSNR-H (↑)	PSNR-L (↑)	SSIM (↑)	LPIPS (↓)
LRM-VolSDF	20.53	21.16	0.993	0.006

Figure 19. LRM view synthesis results from DSLR data.

B.2. Sparse-view Reconstruction Application for DSLR data

We provide a sparse-view reconstruction baseline on our DSLR data using learning based reconstruction method. To build the baseline, we train an LRM model similar [70] that achieves state-of-the-art mesh reconstruction results as demonstrated on public synthetic dataset on GSO[21] dataset in Table 6.

LRM baseline Our LRM baseline has the same transformer architecture and tri-plane representation as [70]. It consists of 24 self-attention blocks with feature dimension 1024 and 16 heads of attention. The tri-plane token number is 32×32. Each token is decoded to a 8×8 feature patch through a

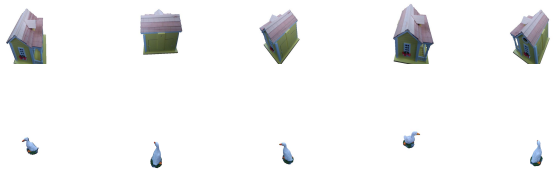

Figure 20. Example input data of egocentric sparse-view reconstruction. We used the masked pixels from the rectified egocentric passive data as input. Some objects can appear very small due to the challenges in free-view movement in egocentric passive trajectory.

Table 8. Quantitative results on novel view synthesis for sparse-view reconstruction using egocentric *passive* data.

	PSNR-H (↑)	PSNR-L (↑)	SSIM (↑)	LPIPS (↓)
LRM-VolSDF	20.03	21.72	0.971	0.065

linear layer, which leads us to final tri-plane resolution of 256×256. During training, our LRM directly outputs SDF value instead of density and we use the volume ray tracing method proposed in [78] to render images for supervision. Compared to the NeRF representation used in [70], our LRM baseline is robust to train and can achieve accurate geometry reconstruction. We evaluate our LRM-VolSDF baseline on synthetic GSO dataset. Results are summarized in Table 6. With 8 input views, our baseline achieves reconstruction accuracy comparable to the state-of-the-arts.

Experiments and evaluation We randomly sample 16 images from the 120 training views as inputs to our LRM-VolSDF network. The testing view is the same as the dense-view inverse rendering experiments. The quantitative numbers for view synthesis are summarized in Table 7. We observe that all three metrics are much worse compared to the results from synthetic GSO dataset, indicating the existence of a domain gap between synthetic and real data. Fig. 19 visualize some of our reconstruction results. We observe that for relative simple shape, our model generalizes well and achieves highly detailed reconstruction that closely match the ground-truths. However, it struggles at reconstructing more complicated object, such as the Birdhouse. Further investigation is required to understand the gap on real world dataset.

B.3. Sparse-view Reconstruction Application for Egocentric data

We use the egocentric *passive* recordings in this evaluation, which is a challenging sparse-view reconstruction setting where we only have casual observation of an object captured by an egocentric RGB camera. Compared to the sparse-view reconstruction for DSLR data where the object always

Figure 21. LRM view synthesis results from egocentric data.

appear in the image center and occupies a large portion of the image, the egocentric recording has natural human moving trajectory and object can appear small in many views.

Fig. 20 shows several examples of the masked objects that we use as input to this evaluation. We can see the camera poses are much more diverse and object only occupies a very small portion of the image. This setting will defy any classical optimization-based 3D reconstruction method. We build a baseline using the same LRM-VolSDF model as mentioned in Sec. B.2. We tested the model on the challenging data without any fine-tuning.

Experiments and Evaluation. For each sequence, we randomly sample 24 views from the middle 1/3 of the whole sequence as this is when our Aria glasses are relatively close to the object. We use 16 of the 24 views as inputs and 8 views as ground-truth for testing. Similar to the DSLR experiment, we use centralized cropping to place the object in the center of the image and modify the plucker rays representation accordingly. Qualitative and quantitative results are summarzed in Fig. 21 and Tab. 8 respectively. We observe that despite that the camera poses distribution of input views is very different from our training data, the LRM-VolSDF generalizes to the new type of input data. Our PSNR numbers for the masked region are much lower than those of synthetic data but still achieve reasonable performance. We calculate the LPIPS and SSIM based on the full image which results in higher numbers. This evaluation expose the domain gap in learning based reconstruction map for egocentric data input. We hope our current experimental results can be a promising starting point.

B.4. Dense-view Reconstruction for Egocentric data

As a complement to Sec. 4.2, we provided qualitative comparisons of novel view synthesis in Fig. 22. We rendered the depth from both baselines in a normalized depth range. For 3D-GS, we acquire the normal from its point cloud ray casted from each pixels. For 2D-GS, we directly use the predicted normal from the rasterizer.

Results analysis: Both 3D-GS[37] and 2D-GS [31] can

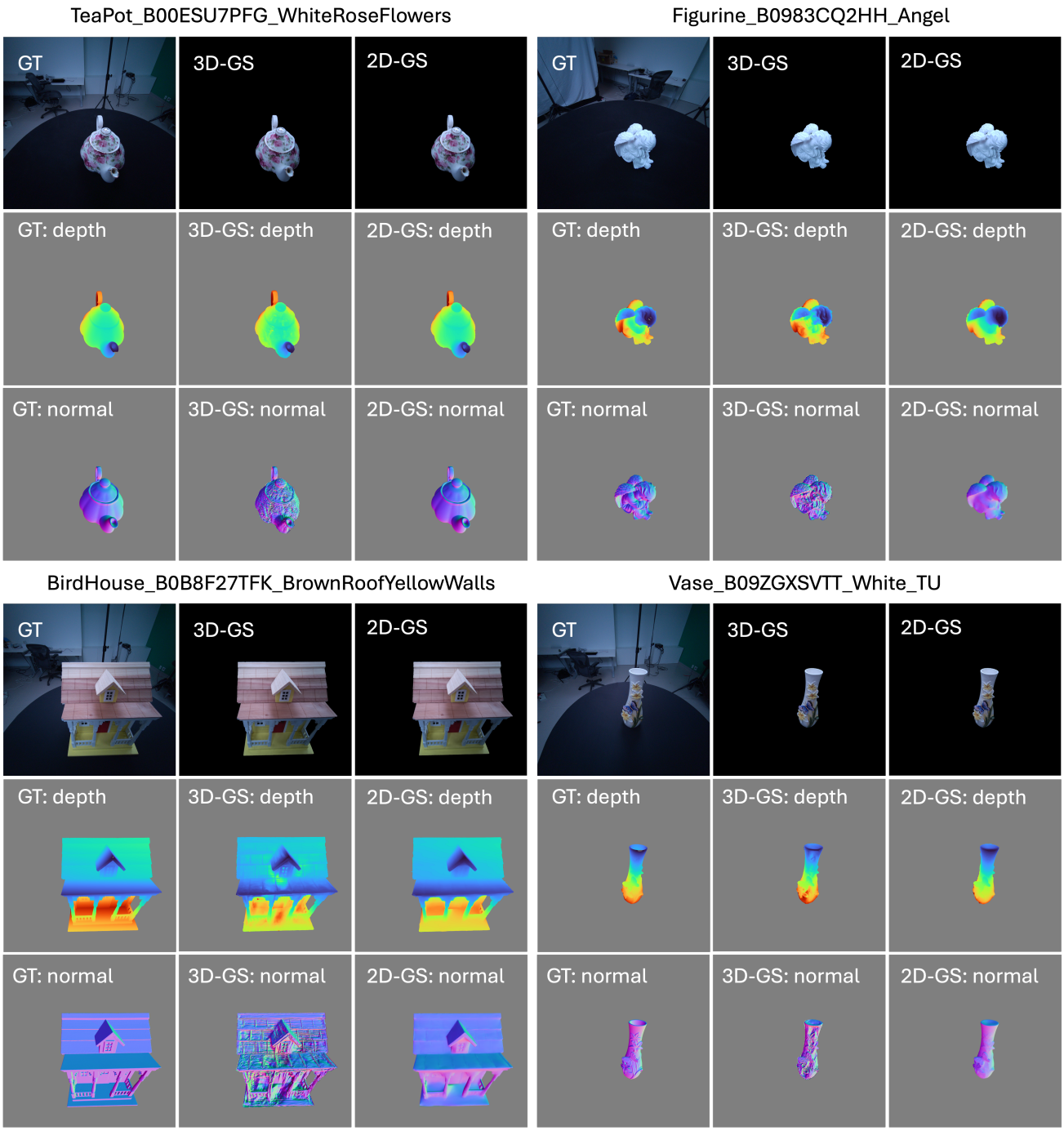

Figure 22. Qualitative comparisons of baselines reconstructions on egocentric recordings. We compare 3D-GS[37] and 2D-GS[31] to the ground truth using the modalities of rendered images, depth and normal. Both baselines can provide near photoreal view synthesis on the held-out validation view compared to ground truth. However, the comparison in their geometry (depth and normal) indicate the existing methods still fall short to recover the details in digital twin reconstruction. For the benefit for visualization, we rotate all the images 90 degree clockwise.

provide near photo-real view synthesis of the objects with high quality view-dependent effect. 3D-GS perform slightly better in PSNR metric consistent across all objects. In terms of shape reconstruction, 2D-GS performs better in particular to recover surface normals. Compared to the ground truth rendering, the estimated depth from 3D-GS is significantly

noiser which also leads to visible artifacts in its normal map, while the 2D-GS tends to predict smoother depth and normal and can ignore certain details. For objects with simple shape (e.g., `Teapot_B00ESU7PFG_WhiteRoseFlowers`), both methods perform well. However, for objects with complex shapes, both methods fail to recover the complex geometric details in the object shape (e.g., `BirdHouse_B0B8F27TFK_BrownRoofYellowWalls`). The challenge in shape reconstruction indicates the direction for future research work in this area.

C. Experimental Details For Robotic Experiments

C.1. Experimental Objects

The specific objects that we use from the DTC and Objaverse-XL datasets [2] are as follows:

DTC dataset:

- `Cup_B01LYONYPB_SkyBlue`
- `Cup_B0BR43SPKJ_Blue`
- `Cup_B0CJBZT7N5_Black`
- `Cup_B0CMPB8FNY_MountainBluebell`
- `Cup_B0CYL5PSR3_Gray`
- `Cup_B08PTSRWF8_Green`
- `Cup_B0CMD4LX4D_DarkBlue`
- `Cup_B08TWHJ33Q_Tan`
- `Cup_B0CNJP2KZF_GreenOrange`
- `Cup_B094NQH2YM_BlackGold`
- `Cup_B0BXB21T7Q_Blue`
- `Cup_B0CPX832ZP_Floral`
- `Cup_White`
- `Cup_B09L8DS2ZB_DarkBrown`
- `Cup_B0C3X3WY2Q_SageGreen`
- `Cup_B0CQTF6GF1_RedWhite`
- `Cup_B09QCYR1SL_Pitcher_1`
- `Cup_B0C81BCSXQ_WhiteRainbow`
- `Cup_B0CQXPND8L_Stripes`
- `Cup_B09YDPVRM7_RedTeaCup`
- `Cup_B0CDWXPDK1_Zebra`
- `Cup_B0CR45H24G_Blue`
- `Cup_B0B3JKZW76_Brown`
- `Cup_B0CHJYN6F1_Black`

For Objaverse-XL, we collect a subset of 24 objects by filtering objects by the STL file format and by the word `cup` appearing in the provided metadata tags, and then manually filtering the results by inspecting the object files to only include objects that can be considered "cups" by a human judge.

[2]The version of Objaverse-XL used in this work excludes all 3D models sourced from Sketchfab.

C.2. Simulation and Data Collection

When importing objects into the PyBullet simulator, we obtain separate collision meshes by performing convex decomposition of each object with CoACD [69] with a concavity threshold of 0.01, and rescale each object to have a maximum bounding box side length of 0.137 m (to match that of the test object). Since not all Objaverse-XL objects come with textures, we randomly color Objaverse-XL objects uniformly in RGB space.

For data collection, we uniformly randomly select one object from the considered object set to be placed into the scene. We uniformly randomly place the object within a 0.4×0.4 m box centered at the scene origin at a z height of 0.10m, and allow it to fall to the floor workspace. If the object rolls away (further than a 0.8×0.8 m bound), we re-sample the initial position and re-attempt the object placement.

Pushing: For pushing, we generate pseudo-random trajectories by executing a scripted policy in two stages: First, the robot samples a starting position from the push by randomly sampling an angle θ and radius r. It then takes steps to move its end-effector to the x-y location $(x_{obj} + r\sin(\theta), y_{obj} + r\cos(\theta))$, where (x_{obj}, y_{obj}) is the initial object position. After reaching within 2cm of this location, the robot begins to push the object, by moving from its current end-effector position toward the object location. All steps are normalized to have a magnitude of 0.03m, and the action at each step is affected by uniform random noise drawn from $\mathcal{U}(-0.05, 0.05)$ in each axis. The trajectory ends after 35 steps.

Grasping: For grasping, we generate successful grasps by sampling grasping candidates and filtering only the successful ones. After adding an object to the scene, we uniformly randomly sample a grasping position (x, y, z) where $x = x_{obj} + \mathcal{U}(-0.05, 0.05)$, $y = y_{obj} + \mathcal{U}(-0.05, 0.05)$, $z = 0.2 + \mathcal{U}(-0.1, 0.1)$ where (x_{obj}, y_{obj}) is the initial object position. The robot then performs four steps: (1) It moves in the air to the target grasp position x, y coordinates, (2) it moves vertically downwards to a height of z, (3) it closes the gripper, and (4) it lifts to a height of 0.5m. The grasp is considered successful if the object is at least 0.1m above the floor after step (4).

C.3. Policy Training

We train convolutional neural network policies to regress actions representing xy position changes and the (x, y, z, θ) grasp position for pushing and grasping respectively. For pushing, we train goal-image-conditioned policies, relabeling goals using a hindsight sampling mechanism. We sample goals uniformly randomly from future timesteps in the trajectory of a sampled initial state, and use the first action after the initial state as the label.

We use similar architectures for pushing and grasping.

The architecture consists of an encoder and an MLP head for action prediction. Below we describe the architecture for goal-conditioned pushing, with differences for the grasping policy noted. The input image size for both initial and goal images is $256 \times 256 \times 3$.

For pushing, we normalize action labels by multiplying each dimension by 100, and for grasping, we normalize the last dimension (grasp angle) by dividing by 10 such that it has similar magnitudes to the other action label dimensions.

Encoder:

- `Conv2d(3, 32, 3, stride=2, padding=1)`
- ReLU
- `Conv2d(32, 64, 3, stride=2, padding=1)`
- ReLU
- `Conv2d(64, 64, 3, stride=2, padding=1)`
- ReLU
- Flatten
- `Linear((image_size/8)`$^2 \times 64$`, 512)`
- ReLU

Note that for the pushing policy, the observation and goal image share encoder weights.

Action Prediction Head:

- `Linear(`$512 \times 2, 256$`)` (the input feature dimension is 512 for grasping, which has a single image input as opposed to 1024 and two image inputs for pushing, which have their features concatenated together).
- ReLU
- `Linear(256, 256)`
- ReLU
- `Linear(256, 2)` (for grasping, the final output size is 4)

We train policies using a mean-squared-error loss using the Adam optimizer with a learning rate of $3e-4$ and a batch size of 32, holding out 5% of the data for validation. We train for 200 epochs and take the checkpoint with lowest validation loss for evaluation. Training uses a single NVIDIA Titan RTX GPU.

C.4. Evaluation

For evaluation, we generate test sets of 100 initial conditions for each task using the "`cup_scene003`" object from the StanfordORB dataset. For pushing, we generate trajectories in the same way as during data collection and use the initial state and final state as the initial condition and target goal for the policy. We roll out the policy for 35 steps and compute the final position error of the object in the x, y dimensions. For grasping, we generate initial object positions in the same way as in data collection and the policy performs a single grasp attempt, which is determined successful if the object ends at least 0.1m above the working surface.

In Figure 8, we present detailed results for the pushing experiments, plotting the success rates for each policy with respect to varying success thresholds based on final object position error.