Appendix

A Additional ACDC Details

A.1 Offline Dataset Generation

Cousins creation requires a large-scale asset set. We adopt BEHAVIOR-1K [4], which includes over 10,000 object assets. The goal of this stage is to preprocess the whole asset set for later usage. Since objects may have occlusion in the input image, common approaches that can estimate the scale and orientation of real objects such as point cloud registration [84, 85] and monocular pose estimation methods [86] are not feasible because these generally require two complete, unobstructed point clouds for a given object. Instead, we choose to represent each asset as a set of visual 2D images, under the expectation that we will use a visual encoder (such as DINOv2) downstream to match geometric correspondences between objects. For our given dataset, we rotate each asset $\mathbf{a}_i$ in the whole asset set and take snapshots from a fixed camera pose $\mathbf{P}_{sim}$, resulting in a set of images $\{\mathbf{i}_{is}\}_{s=1}^{N_{snap}}$ and representative snapshot $\mathbf{I}_i$. Each asset $\mathbf{a}_i$ is pre-annotated with its own semantically-meaningful category $\mathbf{t}_i$. This results in asset tuples $\{\mathbf{a}_i = (\mathbf{t}_i, \mathbf{I}_i, \{\mathbf{i}_{is}\}_{s=1}^{N_{snap}})\}_{i=1}^{N_{assets}}$, where N_{assets} is the total number of assets included in the BEHAVIOR-1K dataset. Note that this stage occurs once offline, and can be cached when running ACDC.

A.2 Mounting Type

We observe that scene objects often serve different semantic roles and fall under difference pose distributions depending on whether an object is fixed with respect to the room. Therefore, as mentioned in Section 2.1, we leverage this inductive bias and prompt GPT to determine if an object is mounted on a wall or not. This distinction helps address a key limitation with our one-shot approach: because of heavy occlusion resulting from a single camera view, objects such as televisions or cabinets that are mounted to walls may only have its frontal face observed from a single camera view, resulting in a insufficient extracted point cloud that does not fully capture its underlying volumetric depth. television or a cabinet fixed on a wall, a frontal view image may only cover the frontal face of the mounted object.

We mitigate this limitation by prompting GPT to classify each object into one of three semantic categories: (1) **Wall Mounted**: An object is fixed on a wall with nothing closely beneath it; (2) **On Floor or On Another Object**: An object is placed on the floor, or on another object, but the object does not touch a wall; (3) **Mixture**: An object is not mounted on a wall, but one of its face touches a wall, like a bookshelf putting on the floor but touches the wall behind it, or a microwave oven putting on a cabinet but its back face touches the wall behind it. In cases (1) and (3), we also require GPT to specify the specific wall on which the object is mounted by feeding all masked walls in the input image generated by Grounded-SAM. Please see Appendix A.3 for how objects with different mounting types are processed.

A.3 Generated Scene Post-Processing

After putting all assets in the correct position in the **Simulated scene generation** stage described in Section 2.1, we post-process each asset for a physically plausible scene. For each asset i, we should have its bounding center position $\mathbf{p}_i^{cen}$, bounding box's top-right vertex position $\mathbf{p}_i^{TR}$, and bounding box's bottom-left vertex position $\mathbf{p}_i^{BL}$. First, we sort all assets from low to high by sorting $\mathbf{p}_i^{cen}$ in ascending order, and project each asset's 3D bounding box to the x-y plane, resulting in a 2D polygon $poly_i$ for each asset i. We then infer "on top" relationships from our sorted asset list. For each asset i, we search over all assets with lower $\mathbf{p}_i^{cen}$ to determine another asset j right beneath it. Whenever the overlapped area between the lower asset j's projected 2D polygon $poly_j$ and the current asset's projected 2D polygon $poly_i$ exceeds 70% of the area of either one of the 2D polygons, i.e., $area(intersect(poly_i, poly_j)) > 0.7 \cdot \min(area(poly_i), area(poly_j))$, we determine that the higher asset i is on top of the lower asset j, and the lower asset j is beneath the higher asset i.

Intuitively, this checks for vertical spatial alignment between two objects. If no matching asset is found, the asset is regarded as being on top of the floor. After all assets have been evaluated in this way, each asset should have another asset or floor beneath it after performing the above searching.

Next, we post-process all assets based on their mounting type: For an asset i with mounting type (1) (Wall Mounted), we first adjust its scale and orientation, and then adjust its position. Since asset i is mounted on a wall, we determine the face of asset i that should be adjusted such that it becomes parallel to the wall. First, we fit a plane to the wall from its corresponding extracted point cloud. We then compute the minimum rotation that aligns either the object's local x or y axis with the normal vector of the wall plane. Finally, we compute the distance between $\mathbf{p}_i^{cen}$ and the wall, and rescale and translate asset i in the x-y plane such that the object's rear face is co-planar with the wall plane and object's front face maintains its same position. Finally, we de-penetrate this object from others by adjusting $\mathbf{p}_i^{cen}$'s z value: We increase $\mathbf{p}_i^{cen}$ by $z(\mathbf{p}_j^{TR}) - z(\mathbf{p}_i^{BL})$, if $z(\mathbf{p}_j^{TR}) > z(\mathbf{p}_i^{BL})$, where $z(\cdot)$ if the z coordinate of a 3D vector, and j is the index of the asset beneath asset i, and then fix asset i on the wall that GPT selected for asset i. When $z(\mathbf{p}_j^{TR}) \leq z(\mathbf{p}_i^{BL})$, we directly fix asset i on the wall without adjusting its position.

For an asset i with mounting type (2) (On Floor or On Another Object), we similarly de-penetrate by placing asset i on top of asset j by adjusting $\mathbf{p}_i^{cen}$ by $|z(\mathbf{p}_j^{TR}) - z(\mathbf{p}_i^{BL})|$. For an asset i with mounting type (3) (Mixture), we adjust the orientation and scale in the same way as assets with mounting type (1), and then adjust $\mathbf{p}_i^{cen}$ in the same way as assets with mounting type (2).

Finally, we check for collisions between the collision meshes of each pair of placed assets and adjust their positions in the x-y plane to avoid any overlap.

A.4 Skill Definition

In order to bootstrap automated demonstration collection, we define a library of analytical and sampling-based skills that can be chained together to solve long-horizon tasks, such as the **Putting Away Bowl** task. For collision-free motion planning, we leverage CuRobo [87]. For sampling-based grasp generation, we leverage Grasp Pose Generator (GPG) [88] [89] based on a given object's sampled point cloud from its analytical mesh. Below, we briefly describe the high-level implementation of each skill:

Open. This skill consists of five steps: **Approach**, which computes a collision-free trajectory towards a point offset in front of the desired handle to articulate, **Converge**, which computes an open-loop straight-line trajectory to the actual grasping point on the handle, **Grasp**, which closes the gripper to grasp the handle, **Articulate**, which computes an open-loop analytical trajectory to articulate the link, and **Ungrasp**, which opens the gripper to release the handle.

For a given articulated object, we leverage ground-truth knowledge of its geometric affordances to compute a corresponding trajectory. Given a specific articulated asset $\mathbf{a}$ and desired link to articulate $\mathbf{l}$, we first infer the link's corresponding handle location by shooting rays towards the link and define the mean handle location as mean location over the rays with the shortest distance. This assumes that the most protruding geometric feature corresponds to the handle. Given handle location, we inspect $\mathbf{l}$'s parent link $\mathbf{j}$'s properties, determining its type (prismatic or revolute) and pose with respect to the handle. Given this information, we can compute a desired analytical trajectory for the handle to open link $\mathbf{l}$. This can easily be transformed into the robot frame, and offset according to the robot's end-effector size.

Close. This implementation is nearly identical to **Open**, though for computing the desired articulation trajectory, the start / end points are reversed.

Pick. This skill consists of three steps: **Move**, which computes a collision-free trajectory towards a sampled grasping point, **Grasp**, which closes the gripper to grasp the object, and **Lift**, which computes an open-loop trajectory to lift the object slightly.

Note that during the **Move** phase, we sample grasping points that are both feasible, collision-free, and minimize robot gripper orientation changes to avoid bad robot configurations.

Place. This skill consists of three steps: **Move**, which computes a collision-free trajectory towards a sampled placement pose, **Ungrasp**, which opens the gripper to release the object, and **Lift**, which computes an open-loop trajectory to lift the gripper slightly.

This skill assumes that an object is already grasped prior to its execution. We assume the desired placement pose is a kinematic predicate relative to another scene object, e.g.: `inside(cabinet)`. Given this predicate, we use rejection sampling to sample collision-free poses for the robot's end-effector and grasped object that satisfy the given predicate, prioritizing poses that minimize end-effector rotation.

A.5 Demonstration Collection

We use fully automated demonstrations using our programmatic skills defined above. For the **Door Opening** and **Drawer Opening** tasks, this simply consists of executing the **Open** skill. For the **Putting Away Bowl** task, this consists of a **Open**, **Pick**, **Place**, **Close** sequence. We use rejection sampling so that our resulting dataset only includes successes, that is, if any skill execution fails midway, we do not save that episode. This allows us to significantly increase the randomization range between episodes without being limited by poor edge cases.

Across all tasks, we randomize the agent's pose as well as scene objects' poses and scales between episodes.

A.6 Using DINOv2 for Digital Cousin Matching

For a given input image $\mathbf{x}$ and set of candidate matching images $\{\mathbf{i}_j\}_{j=1}^N$, we define the top-1 matched candidate through a DINOv2-based voting system. First, we pass both input image $\mathbf{x}$ and all candidate images $\{\mathbf{i}_j\}_{j=1}^N$ through DINOv2, retrieving their feature patches $\mathbf{e}$ and $\{\mathbf{f}_j\}_{j=1}^N$, respectively. Next, we compute the nearest neighbor (defined as the L2-norm) in the DINOv2 feature embedding space for each pixel in $\mathbf{e}$ over all pixels across all candidate feature embeddings $\{\mathbf{f}_j\}_{j=1}^N$, and record the running count of nearest neighbors across all candidates $j \in \{1, ..., N\}$. The top-1 matched candidate is then the candidate with the highest count of per-pixel nearest neighbors – i.e.: the candidate image $\mathbf{i}_j$ that has the highest number of closest visual feature correspondences to input image $\mathbf{x}$. For top-k matched candidates, we repeat the process iteratively, selecting the top-1 each time and subsequently removing the selected $\mathbf{i}_j$ during proceeding iterations. We leverage GPU-accelerated nearest neighbor computations using the open-source faiss [90] package.

Given a matched pair of images $\mathbf{x}$, $\mathbf{i}_j$, we define the DINOv2 embedding distance as the average nearest neighbor L2-distance between each pixel in corresponding input feature map $\mathbf{e}$ and all pixels in corresponding matched feature map $\mathbf{f}_j$. Note that we exclude the largest 10% of nearest neighbor distances in this calculation, as we find empirically that the sorted results across matched candidates are more salient with these outliers removed.

A.7 Additional Real-to-Sim Details

In this subsection, we provide additional implementation details of ACDC real-to-sim pipeline:

Depth image and point cloud processing. One key design decision in ACDC is our decision to use synthetic depth via Depth-Anything [14], instead of a dedicated depth camera. This decision is guided by our observation that it performs more consistently on reflective surfaces. However, this synthetic depth approach still generates artifacts occurring near object boundaries and the image periphery, and so to mitigate this issue we post-process the output of Depth-Anything by cropping to the center 90% of a given image and applying an erosion kernel to segmented object masks $\mathbf{m}_i$.

To further remove noise in object point clouds, we apply DBSCAN clustering [91] on each object point cloud $\mathbf{p}_i$ to filter out noisy points.

Orientation Refinement. DINO performs a rough estimation of asset orientations, which for most objects the orientation is sufficiently accurate. However, ACDC additionally provides an option to further refine the orientation refinement based on an object's extracted point cloud. By computing the z-aligned minimum bounding box of the given point cloud, we can apply an additional z-rotation to DINO's outputted estimated orientation so that the matched asset's canonical xy-axes aligns with the computed minimum bounding box frame. We find this is especially useful for object's that have sharp geometric boundaries, such as furniture objects.

Heuristics for articulated objects. In this project, articulated objects refer to those with doors (revolute) and drawers (prismatic). To ensure the selected digital cousins of an articulated object are also articulated, so that door opening or drawer opening demos can be collected on all digital cousins, we propose to search digital cousins for articulated objects only among articulated assets. Because we have ground-truth information for all of our dataset assets, we know *apriori* which assets are articulated. During the **Real-world extraction** stage, we additionally prompt GPT to determine whether objects are articulated.

An optional heuristics is to apply a door/drawer count threshold on digital cousin creation of articulated objects. During the **Offline Dataset Generation** stage, we can count the number of doors (revolute joints) and drawers (prismatic joints). When creating cousins, we only search among assets with "similar" number of drawers and doors. This threshold is open to users to set. In all of our real-to-sim results, we set the threshold to 2 in the nearest cousin selection too guarantee affordance preservation, but do not apply this heuristic to the rest of the scenes.

GPT API Usage. We use GPT-4o for the real-to-sim pipeline, but experiments in Section 3.1 and Appendix B.1 are done by GPT-4v, as GPT-4o was not released at the time.

B Additional Experimental Details

B.1 Ablation Study

In this subsection, we extend Section 3.1 of our main paper by conducting an ablation study on the real-to-sim pipeline in a sim-to-sim setting. We seek to evaluate whether DINO is sufficient for digital cousin matching, or if applying GPT to finetune DINO's selections can result in improved performance. Our quantitative and qualitative results cover the following comparisons: **(a)** DINO Model Selection & GPT Orientation Selection; **(b)** DINO Model Selection & DINO Orientation Selection; **(c)** GPT Model Selection & GPT Orientation Selection; **(d)** GPT Model Selection & DINO Orientation Selection.

DINO Model Selection involves selecting an asset $\mathbf{A}_c$ as the best digital cousin of an object based solely on the DINOv2 embedding distances between the masked object RGB $\mathbf{x}_i$ and all assets' representative model snapshots $\mathbf{I}_j$ within the nearest k_{cat} categories. While DINO Model Selection generally yields reasonable results, the default scale when capturing representative model snapshots can affect the selection of the best digital cousin. To refine this process, we propose **GPT Model Selection**, which first uses DINOv2 embedding distances to select k_{model} candidate models and then prompts GPT to choose the best one, with $k_{model} = 10$ in practice.

To select the best orientation $\mathbf{q}_c$ of $\mathbf{A}_c$, we first identify k_{ori} candidate orientations based on DINOv2 embedding distances between $\mathbf{x}_i$ and all snapshots $\{\mathbf{i}_{is}\}_{s=1}^{N_{snap}}$ of the selected digital cousin $\mathbf{A}_c$. **DINO Orientation Selection** involves reorienting the asset $\mathbf{A}_c$, rescaling it, placing it in the scene as described in Section 2.1, normalizing its bounding box, and retaking a snapshot with the same relative position to the viewer camera as detailed in Appendix A.1. The best orientation $\mathbf{q}_c$ is then selected based on DINOv2 embedding distances with the retaken snapshots and $\mathbf{x}_i$. However, orientation can be defined for objects within the same category based on key features, even

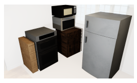
Input Scene	ACDC Output	Scale (m)		Cat.	Mod.	L2 Dist (cm) ↓	Ori. Diff. ↓	Bbox IoU ↑	Ori. Bbox IoU ↑
		3.68	(a)	5/5	5/5	5.27 ± 2.85	0.07 ± 0.07	0.75 ± 0.14	0.64 ± 0.07
			(b)			5.27 ± 2.85	0.07 ± 0.07	0.75 ± 0.14	0.64 ± 0.07
			(c)	5/5	5/5	5.27 ± 2.85	0.07 ± 0.07	0.75 ± 0.14	0.64 ± 0.07
			(d)			5.27 ± 2.85	0.07 ± 0.07	0.75 ± 0.14	0.64 ± 0.07
		3.42	(a)	6/6	4/6	**4.79 ± 1.52**	**0.07 ± 0.03**	0.54 ± 0.32	0.52 ± 0.28
			(b)			4.85 ± 1.52	0.08 ± 0.01	0.54 ± 0.32	0.52 ± 0.28
			(c)	6/6	**6/6**	4.87 ± 1.83	0.12 ± 0.13	0.54 ± 0.31	0.53 ± 0.28
			(d)			5.03 ± 1.53	**0.10 ± 0.11**	0.54 ± 0.32	0.52 ± 0.29
		2.91	(a)	6/6	4/6	**5.51 ± 2.71**	0.03 ± 0.00	0.64 ± 0.16	**0.60 ± 0.17**
			(b)			5.97 ± 2.56	0.03 ± 0.00	0.64 ± 0.16	**0.60 ± 0.17**
			(c)	6/6	**5/6**	7.13 ± 4.77	0.16 ± 0.19	0.54 ± 0.24	0.51 ± 0.21
			(d)			5.60 ± 2.92	**0.10 ± 0.10**	**0.65 ± 0.18**	**0.60 ± 0.17**
		3.54	(a)	5/5	2/5	**6.47 ± 2.79**	0.04 ± 0.01	**0.64 ± 0.23**	**0.65 ± 0.28**
			(b)			6.83 ± 3.20	**0.03 ± 0.01**	0.63 ± 0.24	0.64 ± 0.30
			(c)	5/5	**3/5**	6.51 ± 2.77	0.03 ± 0.01	**0.64 ± 0.22**	0.60 ± 0.20
			(d)			6.90 ± 3.21	0.03 ± 0.01	0.62 ± 0.24	0.58 ± 0.22
		3.24	(a)	5/6	3/6	6.64 ± 3.34	0.24 ± 0.20	0.57 ± 0.21	0.58 ± 0.15
			(b)			5.92 ± 3.32	**0.06 ± 0.03**	**0.69 ± 0.14**	0.66 ± 0.14
			(c)	5/6	**5/6**	6.00 ± 3.79	0.06 ± 0.05	0.68 ± 0.14	0.65 ± 0.14
			(d)			**5.33 ± 3.77**	**0.04 ± 0.03**	**0.69 ± 0.15**	**0.67 ± 0.15**
		4.17	(a)	8/8	3/8	**7.81 ± 4.87**	0.05 ± 0.00	0.65 ± 0.18	**0.64 ± 0.17**
			(b)			7.91 ± 5.04	0.05 ± 0.00	0.65 ± 0.19	**0.64 ± 0.16**
			(c)	8/8	**6/8**	7.94 ± 5.01	0.05 ± 0.00	**0.66 ± 0.17**	0.63 ± 0.17
			(d)			7.94 ± 5.01	0.05 ± 0.00	**0.66 ± 0.17**	0.63 ± 0.17
		6.89	(a)	12/12	6/12	7.27 ± 5.51	0.46 ± 0.50	0.54 ± 0.30	0.55 ± 0.25
			(b)			6.31 ± 4.78	**0.24 ± 0.49**	0.64 ± 0.26	0.62 ± 0.25
			(c)	12/12	**11/12**	**6.15 ± 4.50**	0.25 ± 0.47	0.67 ± 0.22	0.62 ± 0.20
			(d)			5.96 ± 4.35	**0.05 ± 0.04**	**0.70 ± 0.18**	**0.63 ± 0.18**
		10.23	(a)	15/15	12/15	17.40 ± 9.05	0.17 ± 0.17	0.49 ± 0.20	0.50 ± 0.19
			(b)			**17.01 ± 9.34**	**0.15 ± 0.14**	0.51 ± 0.20	0.50 ± 0.20
			(c)	15/15	**15/15**	17.16 ± 9.31	0.16 ± 0.17	0.51 ± 0.22	0.51 ± 0.21
			(d)			17.09 ± 9.26	**0.11 ± 0.10**	**0.55 ± 0.21**	**0.53 ± 0.21**

Table 2: Quantitative evaluation of scene reconstruction using our ACDC method in a sim-to-sim scenario. This table is an extension of Table 1 in the main paper. 'Cat.' indicates the ratio of correctly categorized objects to the total number of objects in the scene. 'Mod.' shows the ratio of correctly modeled objects to the total number of objects in the scene. 'L2 Dist' provides the mean and standard deviation of the Euclidean distance between the centers of the bounding boxes in the input and reconstructed scenes. 'Ori. Diff.' represents the mean and standard deviation of the orientation magnitude difference of each non-uniformly symmetric object. 'Bbox IoU' presents the Intersection over Union (IoU) for axis-aligned 3D bounding boxes. 'Ori. Bbox IoU' displays the IoU for oriented 3D bounding boxes.

under different scales. For example, a taller cabinet can be considered to have the same orientation as a shorter cabinet if their frontal faces align. Motivated by this, we propose **GPT Orientation Selection**, where GPT is prompted to directly select the best orientation among the k_{ori} candidate orientations, with $k_{ori} = 4$ in practice.

Table 2 presents a quantitative evaluation of our ACDC in the sim-to-sim setting, while Fig. 6 provides qualitative visualizations of the output scenes for each pipeline. To ensure diversity at the object level, no model is present in more than one test scene.

Based on the category and model matching accuracy, we observe that prompting GPT to select the nearest neighbor from a list of candidates outperforms pure DINOv2 embedding distance selection. This advantage likely stems from DINO being influenced by factors such as lighting conditions, occlusions, and changes in object scale and orientation. In contrast, GPT focuses better on geometry matching given proper prompting, which is crucial in our real-to-sim setting where an exact digital twin of an object is not always available in the simulator. Although GPT occasionally selects an incorrect model, such as the bookshelf in the sixth row of Fig. 6, it still chooses a reasonable substitute that can be appropriately scaled, oriented, and positioned to represent the target object.

Comparing (d) with (c), and (b) with (a) in terms of orientation difference and IoU-related metrics, we find that the performance of GPT Orientation Selection and DINO Orientation Selection is generally comparable. This represents a trade-off between time and robustness. Prompting GPT to

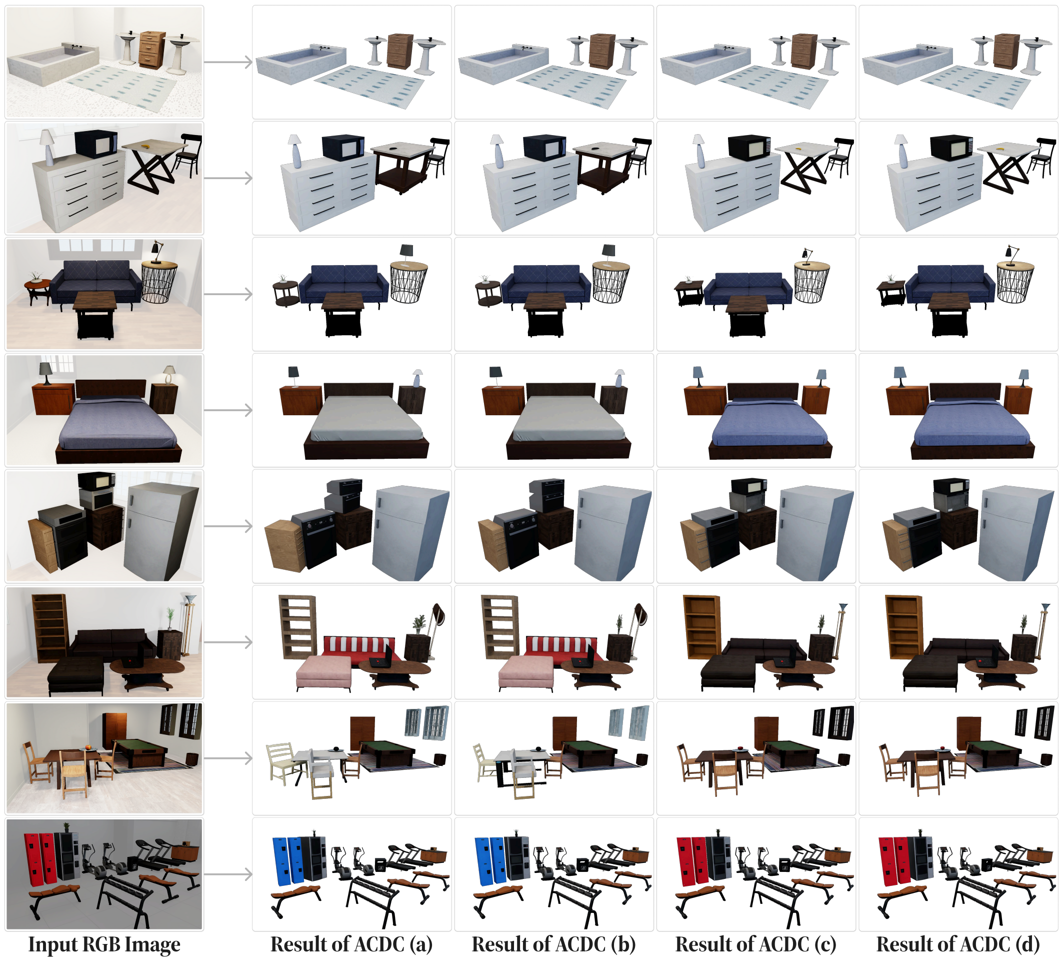


Figure 6: **Qualitative ACDC sim-to-sim scene reconstruction results.** Overall, pipeline (d) gives the best scene reconstruction results, while pipeline (c) balances inference time and reconstruction quality.

select the best orientation takes less than 10 seconds per object, whereas the DINO-based method, which involves rescaling, reorienting assets, taking snapshots, and computing DINO scores, takes about 60 seconds per object but is more robust and accurate. Given that orientation will be randomized during policy training, we recommend GPT Orientation Selection for practical use. For all real-to-sim results, we adopt GPT Orientation Selection.

When comparing (b) with (d), the differences in orientation difference and IoU metrics are minimal, indicating that ACDC can reconstruct high-quality scenes even when the assets in the simulated scene are close approximations (cousins) rather than exact replicas (twins) of the target objects.

Finally, examining the L2 Dist column in Table 2, we see that ACDC places each asset very close to the ground truth position. The average L2 distance errors are less than 10 cm for the first seven test scenes, and is only 17 cm for the eighth scene whose scale is 10.23 m.

B.2 Real-to-Sim Scene Generation: Additional Results

Additional qualitative results of our ACDC real-to-sim cousin creation and scene reconstruction pipeline are presented in Fig. 7. For multi-view visualizations, please refer to our accompanying video and website.

Our ACDC real-to-sim pipeline has the potential to create cousins and reconstruct scenes from a single RGB image without requiring ground truth camera intrinsics. We employ the Paramnet-

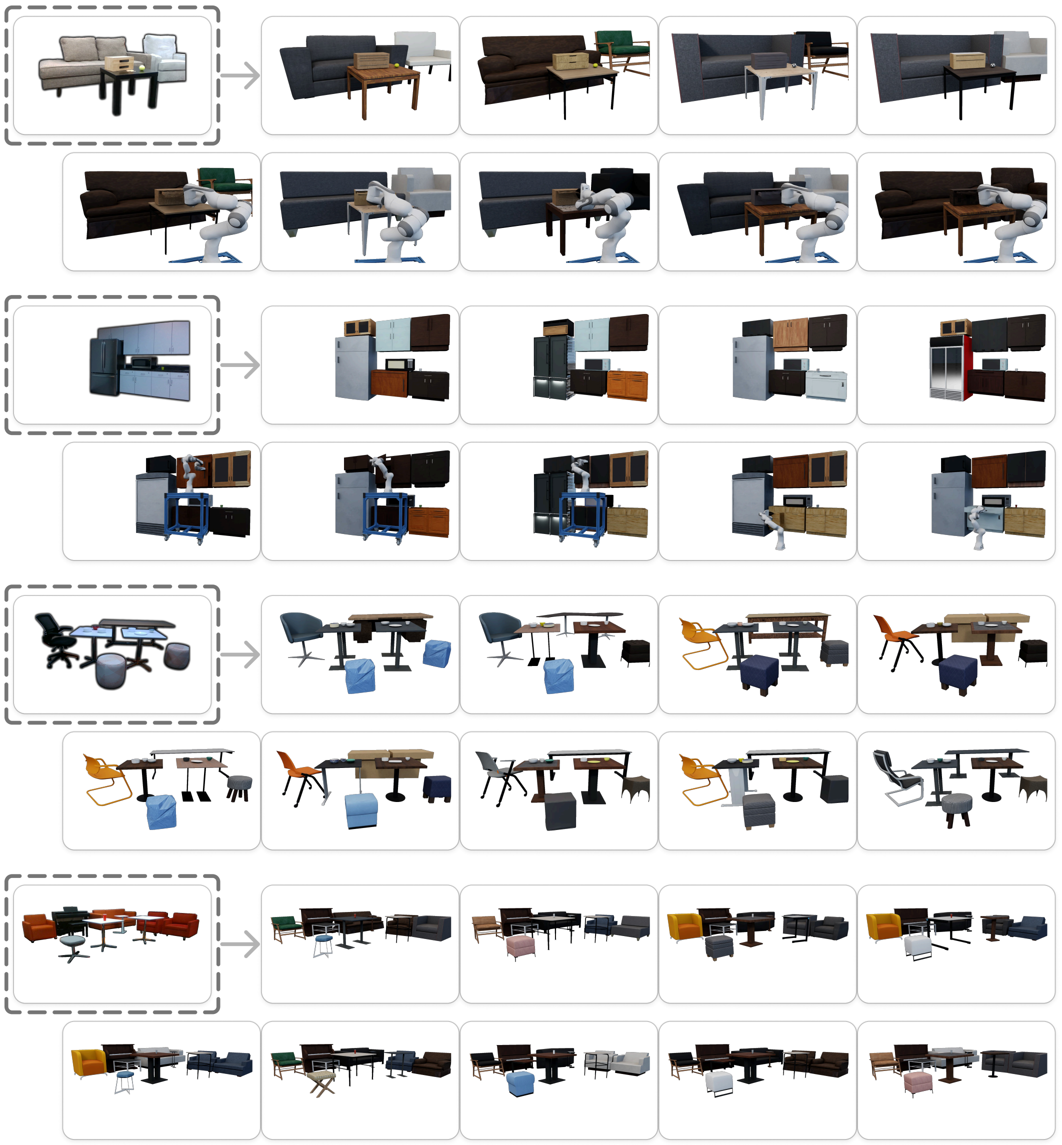


Figure 7: **Qualitative ACDC real-to-sim scene reconstruction results.** Multiple cousins are shown with a robot collecting demonstrations. Images cropped by dashed squares are input RGB images.

360Cities-edina-uncentered model from PerceptiveFields [92] to estimate camera intrinsic matrix $\mathbf{K}$ from the input RGB image. Fig. 8 presents the ACDC real-to-sim scene reconstruction results using the estimated $\mathbf{K}$. This capability may enable large-scale demonstration collection in the future by leveraging in-the-wild web images that lack ground truth camera intrinsics.

B.3 Policy Training Details

We train robot policies using the demonstrations collected (see Appendix A.5. Our action space is delta end-effector actions, expressed as a 6-dimensional (dx, dy, dz) delta position and (dax, day, daz) delta axis-angle orientation command. The commands are then executed via Inverse Kinematics (IK). Our observation space consists of {end-effector position, end-effector orientation, end-effector gripper joint state} proprioception, and a unified point cloud.

The point cloud is computed by first converting all depth images into a single point cloud with a unified frame (in our case, the robot frame), with all non-task relevant objects such as the robot

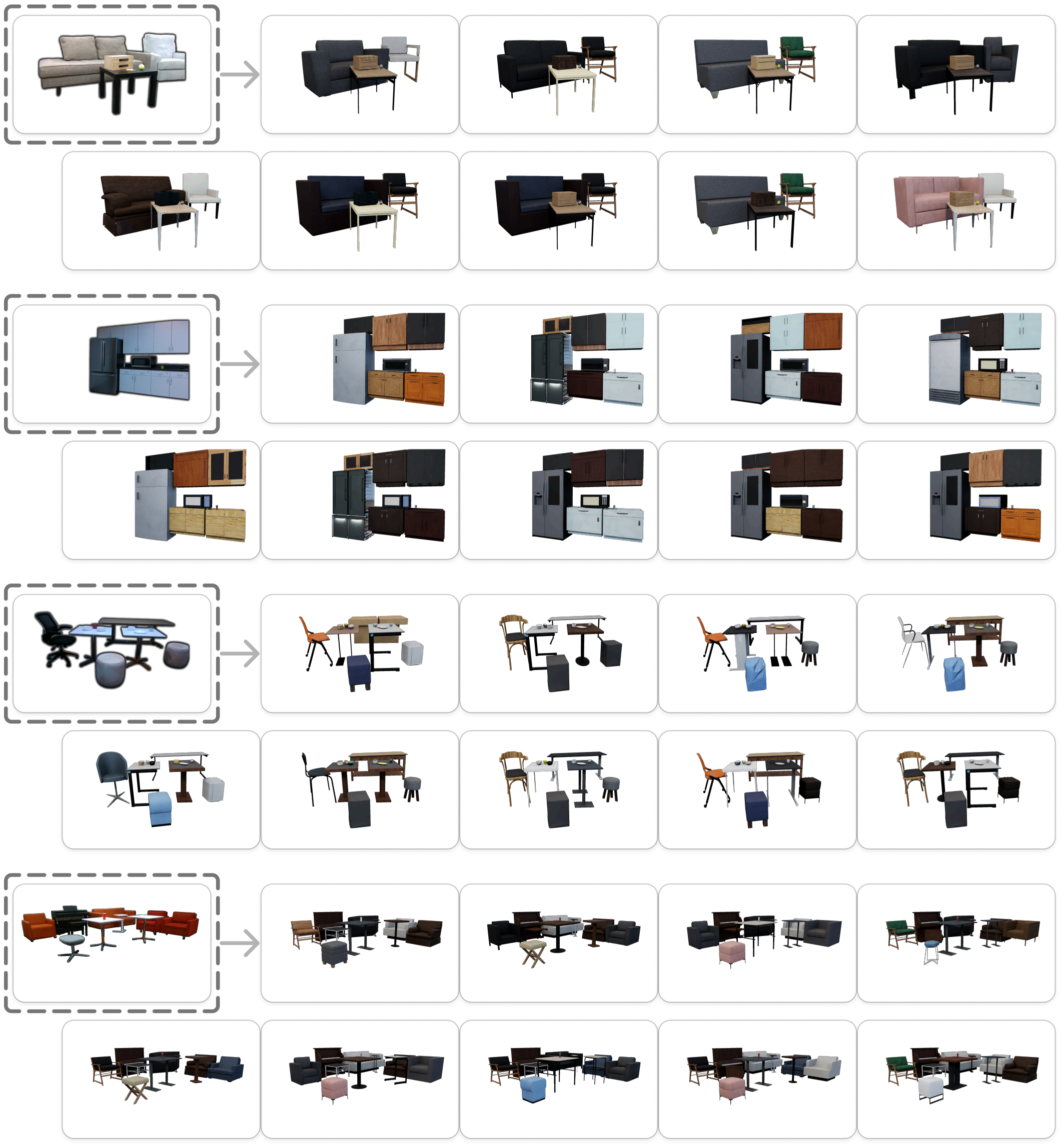


Figure 8: Qualitative **ACDC** real-to-sim scene reconstruction results **without ground truth camera intrinsics K**. Images cropped by dashed squares are input RGB images.

and background masked out. For the real-world setting, we efficiently mask out and track all non-task relevant objects using XMem [93], allowing us to align the sim- and real-world point clouds. We then additionally add a pre-computed point cloud representation of the robot's gripper fingers, placed at the known ground-truth location using the robot's onboard proprioception and forward kinematics. In addition to the (x, y, z) per-point values, we additionally add a fourth binary value $e \in \{0, 1\}$, classifying whether that point belongs to either the scene or the robot's gripper fingers. Finally, we downsample the point cloud to a fixed size using farthest point sampling (FPS). Note that with the exception of the **Putting Away Bowl** task, the point cloud is generated from a single, over-the-shoulder camera. In the **Putting Away Bowl** task, we additionally add another over-the-shoulder camera on the other side of the robot, as well as a wrist camera, since this task exhibits much heavier occlusion during different stages compared to the other tasks.

All of our policies are trained using Behavioral Cloning with an RNN to capture the prior history of actions and a GMM to capture the distribution over demonstrations. We use a 2-layer, 512-dimension PointNet [94] encoder to encode our raw point cloud observations, which undergo further random {downsampling, translation, noise jitter} before being passed to the actor network. We also convert the binary e value into a 128-dimensional learned embedding, to better enable the network to differentiate useful features between the robot fingers and the scene. Our policies use an RNN horizon of 10, RNN hidden dimension 512, are optimized using AdamW [95].

During evaluation, we take the best performing checkpoint for a given run and evaluate it 100 times. These results are then aggregated across multiple runs to give us our finalized results.

B.4 Sim-to-Sim Policy Learning with Digital Cousins

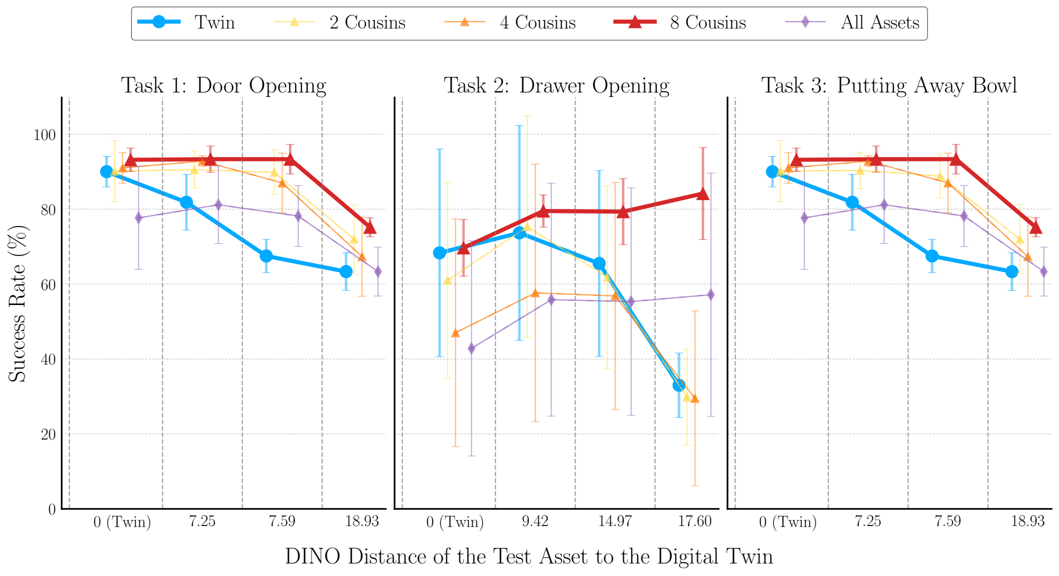


Figure 9: Average success rates (with standard deviations) of policies trained on demonstrations collected from the exact twin, different numbers of cousins, and all assets in the three nearest categories. Success rates are reported for three tasks: Door Opening, Drawer Opening, and the composite task of Putting Away Bowl. Policies are tested on four assets (from left to right in each line plot): the exact digital twin, the second unseen cousin, the sixth unseen cousin, and a more dissimilar asset, to quantify out-of-domain generalization ability. The DINO embedding distance to the digital twin is used as the quantitative metric to rank assets and select cousins. Error bars indicate the standard deviation, reflecting the stability of policy training.

Task	DINO Dist.	Training Models				
		Twin	2 Cousins	4 Cousins	8 Cousins	All Assets
Door Opening	0	96 92 91 91 87 83	100 96 95 92 90 74	95 94 94 92 87 84	97 95 95 94 90 88	94 93 86 67 66 60
	7.25	91 67 87 81 83 82	95 91 96 86 93 82	91 95 94 93 93 91	95 91 88 99 95 92	96 89 88 71 75 68
	7.59	69 60 66 65 73 72	98 85 93 87 95 81	76 91 77 95 96 87	91 96 98 91 87 97	86 83 88 65 73 74
	18.93	58 63 72 64 66 57	74 80 85 57 71 65	47 68 62 78 77 72	80 72 73 75 75 76	72 68 68 53 60 59
Drawer Opening	0	87 86 85 73 71 8	85 80 72 69 53 7	81 67 63 61 7 3	84 73 71 65 63 62	73 68 58 51 6 1
	9.42	87 91 89 80 85 10	85 86 95 93 83 10	92 73 75 86 8 12	86 78 81 82 78 72	73 80 79 79 12 12
	14.97	81 80 78 56 84 14	60 61 88 84 65 13	91 76 71 73 16 14	90 84 86 69 81 66	76 67 82 81 16 10
	17.6	38 36 45 30 32 17	37 35 41 42 13 11	79 32 20 23 15 8	97 88 94 74 90 62	81 75 84 80 8 15
Putting Away Bowl	0	33 14 14 11 9	-	-	11 10 9 8 5	-
	14.17	0 0 0 0 0	-	-	10 8 1 4 3	-
	14.44	3 0 0 0 0	-	-	31 14 24 31 19	-
	17.73	0 0 0 0 0	-	-	0 0 0 0 0	-

Table 3: Success rates (%) of all policies used in Fig. 4 and Fig. 9. "DINO Dist." shows the DINOv2 embedding distances between test assets and the original digital twin.

As an extension of Fig. 4, Fig. 9 presents the average and standard deviations of success rates of policy rollouts on the original digital twin and multiple unseen assets. The success rates of all runs used to generate Fig. 4 and Fig. 9 are detailed in Table 3. For each training set, we train policies with different hyperparameters and select the best two combinations based on the rollout success rate on the original digital twin asset. We then train policies using these best two combinations with three different seeds, resulting in six policies. The results reported in Fig. 4, Fig. 9, and Table 3 are based on these six policies. We note that for the third **Putting Away Bowl** task, we only evaluate on five runs due to resource constraints.

An unexpected behavior is observed in the **Drawer Opening** task, where the 4-cousin policies perform sub-optimally. We believe this is due to the limited number of cabinets with drawers available for cousin selection. Among the four cousins, the first two are geometrically similar, as are the last two, but there is a significant similarity gap between the second and third cousins. This is partially illustrated by their DINO embedding distances to the digital twin: 7.78, 9.32, 14.10, and 14.90. The demonstrations collected on these four assets may not form a high-quality distribution for training. In contrast, the 4-cousin policy in the **Door Opening** task yield decent results, likely because there are more than 40 assets available for cousin selection, allowing ACDC to form a relatively narrower distribution. The geometric similarities between the four cousins in the **Door Opening** task are more continuous in terms of DINO similarity to the digital twin, with DINO distances being 6.49, 7.51, 8.13, and 9.66. However, 8-cousin policies still performed well in this relatively limited category, much better than all-assets policies and twin policies. A key takeaway is that: (1) when there are a sufficient number of assets to choose cousins from, all cousin policies can outperform twin policies on held-out cousins, and (2) more cousins should be found when the number of available assets is relatively small for the target category.

Digital Cousins Improve Policy Training Stableness. Comparing the standard deviation of policies trained on the digital twin, 8 digital cousins, and all assets in Fig. 9, we find that all-assets policies are the most unstable, followed by twin policies, while 8-cousin policies are the most stable. This highlights another advantage of training digital cousin policies: the policy training process on demonstrations collected from a set of high-quality cousins can be more stable, i.e., more robust against different random seeds and requiring less tuning.

References

[1] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv: Arxiv-1709.07857*, 2017.

[2] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. Retinagan: An object-aware approach to sim-to-real transfer. *arXiv preprint arXiv: Arxiv-2011.03148*, 2020.

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots, 2021.

[4] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, M. Anvari, M. Hwang, M. Sharma, A. Aydin, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, S. Savarese, H. Gweon, K. Liu, J. Wu, and L. Fei-Fei. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 80–93. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/li23a.html.

[5] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlavac, S. Y. Min, V. Vondruš, T. Gervet, V.-P. Berges, J. M. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

[6] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[7] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, and R. Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *NeurIPS*, 2022. Outstanding Paper Award.

[8] C.-C. Hsu, Z. Jiang, and Y. Zhu. Ditto in the house: Building articulation models of indoor scenes through interactive perception. *arXiv preprint arXiv: Arxiv-2302.01295*, 2023.

[9] Z. Zhang, L. Zhang, Z. Wang, Z. Jiao, M. Han, Y. Zhu, S.-C. Zhu, and H. Liu. Part-level scene reconstruction affords robot interaction, 2023.

[10] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv: Arxiv-2403.03949*, 2024.

[11] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[12] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges,

C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph. Gpt-4 technical report, 2024.

[13] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[14] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao. Depth anything: Unleashing the power of large-scale unlabeled data, 2024.

[15] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021.

[16] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012. doi:10.1177/0278364911434148. URL `https://doi.org/10.1177/0278364911434148`.

[17] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv: Arxiv-2003.08934*, 2020.

[18] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa. Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv: Arxiv-2302.04264*, 2023.

[19] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *arXiv preprint arXiv: Arxiv-2308.04079*, 2023.

[20] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv: Arxiv-1709.06158*, 2017.

[21] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv: Arxiv-1712.05474*, 2017.

[22] F. Xia, A. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. *arXiv preprint arXiv: Arxiv-1808.10654*, 2018.

[23] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. Tchapmi, A. Toshev, L. Fei-Fei, R. Martín-Martín, and S. Savarese. Interactive gibson benchmark (igibson 0.5): A benchmark for interactive navigation in cluttered environments. *arXiv preprint arXiv: Arxiv-1910.14442*, 2019.

[24] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[25] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, A. Gokaslan, V. Vondruš, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 251–266. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/021bbc7ee20b71134d53e20206bd6feb-Paper.pdf.

[26] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, and R. Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *arXiv preprint arXiv: Arxiv-2206.06994*, 2022.

[27] A. Raistrick, L. Lipson, Z. Ma, L. Mei, M. Wang, Y. Zuo, K. Kayan, H. Wen, B. Han, Y. Wang, A. Newell, H. Law, A. Goyal, K. Yang, and J. Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12630–12641, June 2023.

[28] Y. Yang, F.-Y. Sun, L. Weihs, E. VanderBilt, A. Herrasti, W. Han, J. Wu, N. Haber, R. Krishna, L. Liu, C. Callison-Burch, M. Yatskar, A. Kembhavi, and C. Clark. Holodeck: Language guided generation of 3d embodied ai environments. *arXiv preprint arXiv: Arxiv-2312.09067*, 2023.

[29] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5616–5626, June 2022.

[30] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv: Arxiv-2405.11656*, 2024.

[31] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, Z. Erickson, D. Held, and C. Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv: Arxiv-2311.01455*, 2023.

[32] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[33] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao,

M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv: Arxiv-2212.06817*, 2022.

[34] O. X.-E. Collaboration. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv: Arxiv-2310.08864*, 2023.

[35] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv: Arxiv-2403.12945*, 2024.

[36] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, A. Wahid, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv: Arxiv-2010.14406*, 2020.

[37] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. *arXiv preprint arXiv: Arxiv-2109.12098*, 2021.

[38] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.041. URL `https://doi.org/10.15607/RSS.2023.XIX.041`.

[39] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *arXiv preprint arXiv: Arxiv-2010.01083*, 2020.

[40] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox. Imitating task and motion planning with visuomotor transformers. *arXiv preprint arXiv: Arxiv-2305.16309*, 2023.

[41] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. *arXiv preprint arXiv: Arxiv-2307.14535*, 2023.

[42] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. *arXiv preprint arXiv: Arxiv-2211.11744*, 2022.

[43] Y. Chen, C. Wang, L. Fei-Fei, and C. K. Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv: Arxiv-2309.00987*, 2023.

[44] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. General in-hand object rotation with vision and touch. *arXiv preprint arXiv: Arxiv-2309.09979*, 2023.

[45] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv: Arxiv-2310.17596*, 2023.

[46] R. Hoque, A. Mandlekar, C. Garrett, K. Goldberg, and D. Fox. Intervengen: Interventional data generation for robust and data-efficient robot imitation learning. *arXiv preprint arXiv: Arxiv-2405.01472*, 2024.

[47] Z. Chen, S. Kiami, A. Gupta, and V. Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. *arXiv preprint arXiv: Arxiv-2302.06671*, 2023.

[48] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, D. M, J. Peralta, B. Ichter, K. Hausman, and F. Xia. Scaling robot learning with semantically imagined experience. *arXiv preprint arXiv: Arxiv-2302.11550*, 2023.

[49] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *arXiv preprint arXiv: Arxiv-1910.07113*, 2019.

[50] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023. doi:10.1126/scirobotics.adc9244. URL `https://www.science.org/doi/abs/10.1126/scirobotics.adc9244`.

[51] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. *arXiv preprint arXiv: Arxiv-2210.04887*, 2022.

[52] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *arXiv preprint arXiv: Arxiv-1810.05687*, 2018.

[53] S. Kozlovsky, E. Newman, and M. Zacksenhouse. Reinforcement learning of impedance policies for peg-in-hole tasks: Role of asymmetric matrices. *IEEE Robotics and Automation Letters*, 7(4):10898–10905, 2022. doi:10.1109/LRA.2022.3191070.

[54] D. Son, H. Yang, and D. Lee. Sim-to-real transfer of bolting tasks with tight tolerance. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9056–9063, 2020. doi:10.1109/IROS45743.2020.9341644.

[55] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. S. Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.039. URL `https://doi.org/10.15607/RSS.2023.XIX.039`.

[56] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka. Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning. In *7th Annual Conference on Robot Learning*, 2023. URL `https://openreview.net/forum?id=gFXVysXh48K`.

[57] X. Zhang, M. Tomizuka, and H. Li. Bridging the sim-to-real gap with dynamic compliance tuning for industrial insertion. *arXiv preprint arXiv: Arxiv-2311.07499*, 2023.

[58] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg. Planar robot casting with real2sim2real self-supervised learning. *arXiv preprint arXiv: Arxiv-2111.04814*, 2021.

[59] W. Zhou and D. Held. Learning to grasp the ungraspable with emergent extrinsic dexterity. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 150–160. PMLR, 2022. URL `https://proceedings.mlr.press/v205/zhou23a.html`.

[60] M. Kim, J. Han, J. Kim, and B. Kim. Pre- and post-contact policy decomposition for non-prehensile manipulation with zero-shot sim-to-real transfer. *arXiv preprint arXiv: Arxiv-2309.02754*, 2023.

[61] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv: Arxiv-2405.10315*, 2024.

[62] X. Zhang, S. Jain, B. Huang, M. Tomizuka, and D. Romeres. Learning generalizable pivoting skills. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 5865–5871. IEEE, 2023. doi:10.1109/ICRA48891.2023.10161271. URL `https://doi.org/10.1109/ICRA48891.2023.10161271`.

[63] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv: Arxiv-1804.10332*, 2018.

[64] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: rapid motor adaptation for legged robots. In D. A. Shell, M. Toussaint, and M. A. Hsieh, editors, *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021. doi:10.15607/RSS.2021.XVII.011. URL `https://doi.org/10.15607/RSS.2021.XVII.011`.

[65] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv: Arxiv-2309.05665*, 2023.

[66] R. Yang, G. Yang, and X. Wang. Neural volumetric memory for visual locomotion control. *arXiv preprint arXiv: Arxiv-2304.01201*, 2023.

[67] H. Benbrahim and J. A. Franklin. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, 22(3):283–302, 1997. ISSN 0921-8890. doi:https://doi.org/10.1016/S0921-8890(97)00043-2. URL `https://www.sciencedirect.com/science/article/pii/S0921889097000432`. Robot Learning: The New Wave.

[68] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid. Reinforcement learning-based cascade motion policy design for robust 3d bipedal locomotion. *IEEE Access*, 10:20135–20148, 2022. doi:10.1109/ACCESS.2022.3151771.

[69] L. Krishna, G. A. Castillo, U. A. Mishra, A. Hereid, and S. Kolathaya. Linear policies are sufficient to realize robust bipedal walking on challenging terrains. *arXiv preprint arXiv: Arxiv-2109.12665*, 2021.

[70] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *arXiv preprint arXiv: Arxiv-2105.08328*, 2021.

[71] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *arXiv preprint arXiv: Arxiv-2303.03381*, 2023.

[72] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *arXiv preprint arXiv: Arxiv-2401.16889*, 2024.

[73] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 2023. doi:10.1038/s41586-023-06419-4. URL `https://doi.org/10.1038/s41586-023-06419-4`.

[74] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82): eadg1462, 2023. doi:10.1126/scirobotics.adg1462. URL `https://www.science.org/doi/abs/10.1126/scirobotics.adg1462`.

[75] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv: Arxiv-1710.06537*, 2017.

[76] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. V. Wyk, A. Zhurkevich, B. Sundaralingam, and Y. S. Narang. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 5977–5984. IEEE, 2023. doi:10.1109/ICRA48891.2023.10160216. URL `https://doi.org/10.1109/ICRA48891.2023.10160216`.

[77] J. Wang, Y. Qin, K. Kuang, Y. Korkmaz, A. Gurumoorthy, H. Su, and X. Wang. Cyberdemo: Augmenting simulated human demonstration for real-world dexterous manipulation. *arXiv preprint arXiv: Arxiv-2402.14795*, 2024.

[78] L. Ljung. *System Identification*, pages 163–173. Birkhäuser Boston, Boston, MA, 1998. ISBN 978-1-4612-1768-8. doi:10.1007/978-1-4612-1768-8_11. URL `https://doi.org/10.1007/978-1-4612-1768-8_11`.

[79] P. Chang and T. Padir. Sim2real2sim: Bridging the gap between simulation and real-world in flexible object manipulation. *arXiv preprint arXiv: Arxiv-2002.02538*, 2020.

[80] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 8973–8979. IEEE, 2019. doi:10.1109/ICRA.2019.8793789. URL `https://doi.org/10.1109/ICRA.2019.8793789`.

[81] J. P. Hanna and P. Stone. Grounded action transformation for robot learning in simulation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4931–4932. AAAI Press, 2017.

[82] E. Heiden, D. Millard, E. Coumans, and G. S. Sukhatme. Augmenting differentiable simulators with neural networks to close the sim2real gap. *arXiv preprint arXiv: Arxiv-2007.06045*, 2020.

[83] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023.

[84] W. Guan, W. Li, and Y. Ren. Point cloud registration based on improved icp algorithm. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1461–1465, 2018. doi:10.1109/CCDC.2018.8407357.

[85] P. Li, R. Wang, Y. Wang, and W. Tao. Evaluation of the icp algorithm in 3d point cloud registration. *IEEE Access*, 8:68030–68048, 2020. doi:10.1109/ACCESS.2020.2986470.

[86] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects, 2024.

[87] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. V. Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023.

[88] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt. High precision grasp pose detection in dense clutter, 2016.

[89] H. Liang. Python binding for grasp pose generator (pygpg), Aug. 2021. URL `https://doi.org/10.5281/zenodo.5247189`.

[90] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[91] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[92] L. Jin, J. Zhang, Y. Hold-Geoffroy, O. Wang, K. Matzen, M. Sticha, and D. F. Fouhey. Perspective fields for single image camera calibration, 2023.

[93] H. K. Cheng and A. G. Schwing. XMem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, 2022.

[94] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv: Arxiv-1612.00593*, 2016.

[95] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019.