Supplementary Material

Anonymous Author(s)
Affiliation
Address
email

Abstract: This document serves as the supplementary materials for ”AR2-D2: Training a Robot without a Robot.” It provides comprehensive coverage of the task design process, implementation of AR2-D2, and the resulting outcomes. Details can also be found on our anonymized project page at the following URL: https://anonymousar2d2.github.io/AR2D2.github.io/.

1 Task Details

Personalized Object	Foundational Skill Type	Language Template
`Computer mouse`	Press	“Press the computer mouse”
`Minecraft torch`	Press	“Press the minecraft torch switch button”
`Buzzer`	Press	“Press the buzzer button”
`LEGO train`	Push	“Push the LEGO train”
`8 ball`	Push	“Push the 8 ball”
`Drawer`	Push	“Push the drawer”
`Queen piece`	Pick up	“Pick up the queen piece”
`Plastic bowl`	Pick up	“Pick up the Plastic Bowl”
`Takeaway bag`	Pick up	“Pick up the takeaway bag”

Table 1. Language-Conditioned Tasks in Real-world across the 9 personalized objects for three foundational task.

1.1 Computer mouse

Data filename: `computer_mouse`

Task: Press the computer mouse.

Objects: 1 computer mouse.

Success Metric: Press the computer mouse clicks either left or right has been pressed.

1.2 Minecraft torch

Data filename: `minecraft_torch`

Task: Press the minecraft torch.

Objects: 1 minecraft torch.

Success Metric: Press the minecraft torch switch at the specfic location which is at the bottom of the torch.

1.3 Buzzer

Data filename: `buzzer`

Task: Press the buzzer button.

Objects: 1 buzzer.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

Success Metric: Press the buzzer with the cross on it.

1.4 LEGO train

Data filename: `LEGO_train`

Task: Push the LEGO train.

Objects: 1 LEGO toy train

Success Metric: Push the LEGO toy train from the rear side until it moved even slightly.

1.5 8 ball

Data filename: `8_ball`

Task: Push the 8 ball.

Objects: 1 8 ball

Success Metric: Push the 8 ball until it moved even slightly.

1.6 Drawer

Data filename: `drawer`

Task: Push the top drawer.

Objects: 1 drawer

Success Metric: Push the top drawer until it moved slightly.

1.7 Queen piece

Data filename: `queen_piece`

Task: Pick up the queen chess piece.

Objects: 1 queen chess piece

Success Metric: Pick up the queen chess piece at any part of the body and raise it above ground.

1.8 Plastic bowl

Data filename: `plastic_bowl`

Task: Pick up the plastic bowl.

Objects: 1 plastic bowl

Success Metric: Pick up the plastic bowl from the edge and raise it above ground.

1.9 Takeaway bag

Data filename: `takeaway_bag`

Task: Pick up the takeaway bag.

Objects: 1 takeaway bag

Success Metric: Pick up the takeaway bag from the edge or the string handler and raise it above ground.

Methods	Obj_1	Obj_2	Obj_3	Obj_4	Obj_5	Obj_6	Obj_7	Obj_8	Obj_9
Simulation	2,1,1	1,1,0	4,3,2	1,0,0	3,1,2	3,0,2	3,1,0	2,1,3	1,0,0
VR interface (w/o personalized objects)	1,0,0	1,1,0	2,2,1	0,0,0	3,1,1	2,2,0	2,1,1	1,1,1	0,0,1
VR interface (with personalized objects)	8,5,7	8,3,8	8,7,6	5,4,6	5,7,6	6,7,5	3,1,5	8,6,7	5,4,3
AR2-D2 (Ours)	6,4,7	7,4,5	9,9,4	5,5,4	6,6,4	7,7,5	6,3,6	4,3,9	3,4,0

Table 2. The full result table for the main experiments for all three trials for each distractor scene during test time.

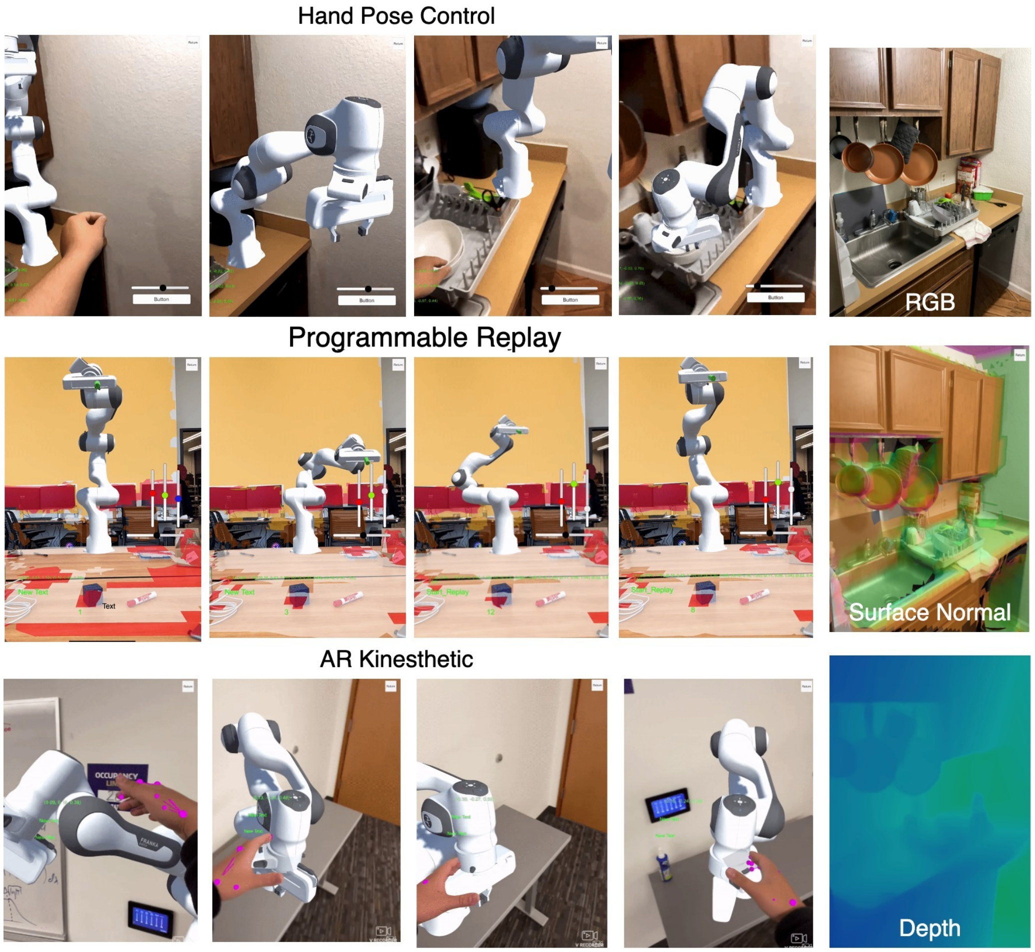

Figure 1. Different approaches of collecting robot demonstration via JARVIS and the different input modalities. (Top) Hand pose control. (Middle) Programmable replay. (Bottom) AR Kinesthetic.

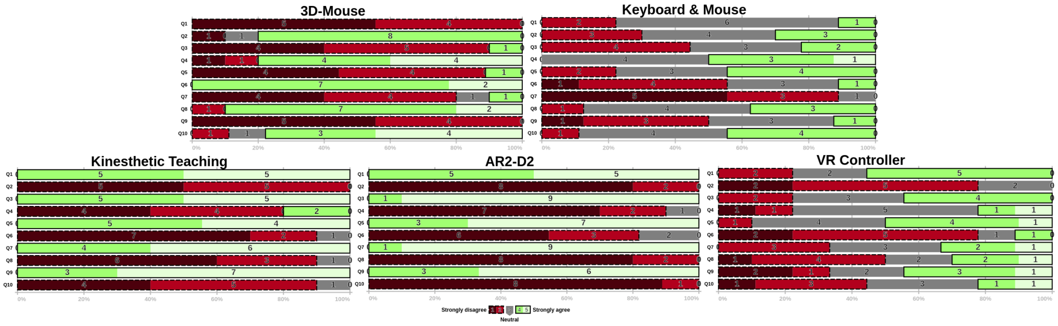

Figure 2. The Likert Scale Plot for the ten questions asked in the SUS Survey across the different methods of demonstrations collection.

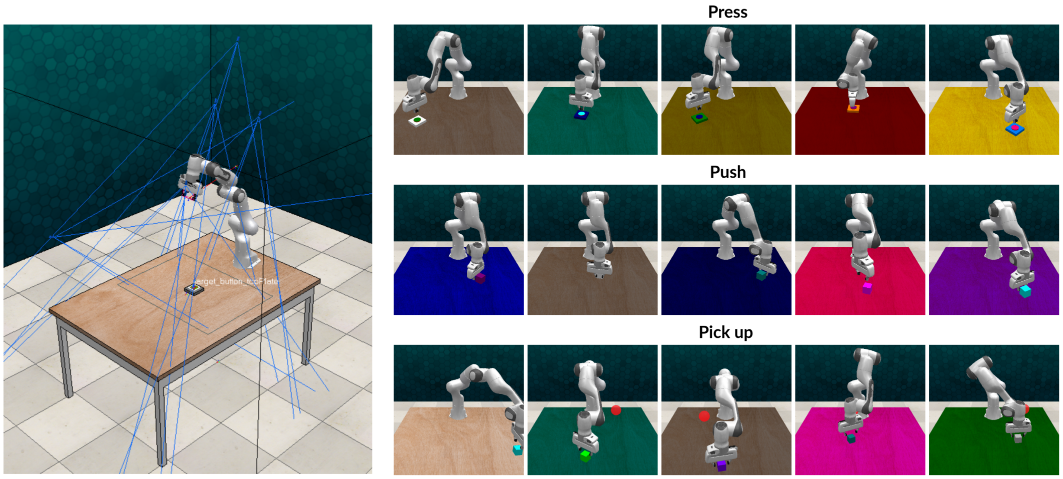

Figure 3. Simulation demonstrations. All of the demonstrations in simulation are collected via RLBench [1] with domanin randomization to help bridge the sim2real gap. (**Left**) The scene setup in the RLBench simulator. (**Right**) The three foundational skills collected via the Key-frames techniques [2] in RLBench.

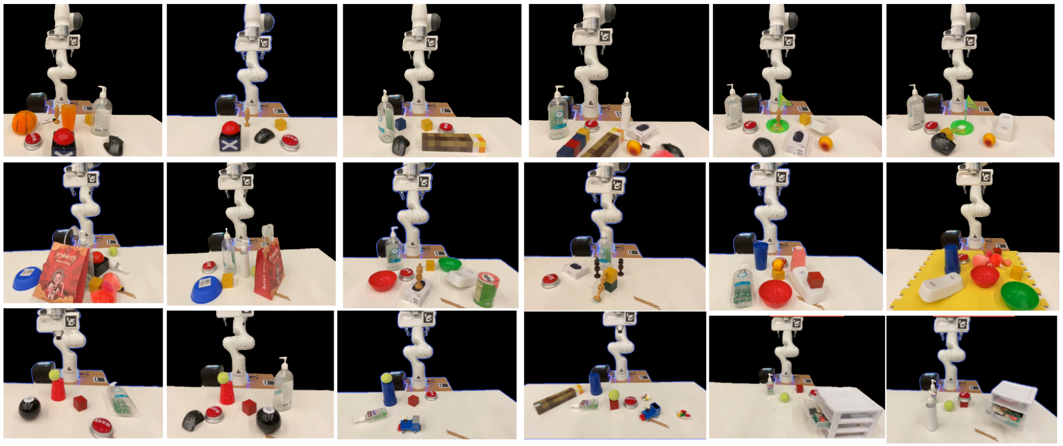

Figure 4. More examples of the different scenes with distractors being used for evaluation of the results.

2 Detailed on AR2-D2 implementation

Setup. AR2-D2 collected demonstrations of 9 highly personalized objects that are difficult to obtain. The demonstrations were primarily collected on two fixed scenes: one on the same table used for deploying the real robot and another on a separate different looking tabletop to introduce diversity. For the collected demonstrations, AR2-D2 will provide the depth frames alongside with the RGB frames, both are 512x424 pixels in dimension, and we then use it to generate an input 3D voxel grid of 100x100x100 for training a real robot via PerAct[3].

We have developed several approaches, as illustrated in Figure 1, to achieve the movement of the end effector to the desired key poses. These approaches can be categorized into three methods:

- Hand pose control: Users directly utilize their hands to reach the desired pose, and the system records the corresponding translation, rotation, and open/close state.
- Programmable replay: Users employ the controller interface on the iPad to control the joint pose of the virtual robot.
- AR kinesthetic: Users manipulate the virtual robot's body using their hand, simulating the control of a real-world robot through kinesthetic teaching.

Detailed training strategy

In this section, we present the detailed implementations of the Behavior Cloning (BC) models. Our approach includes an image-based BC model and 3D BC model PerAct. The image-based BC utilizing a CNN and siamese network, inspired by the BC-Z implementation proposed in Jang et al. [4], which is an image-to-action agent. As for the primary BC model, we employ PerAct [3], a multi-task behavior cloning agent. It takes as input a voxel observation consisting of a 100^3 voxel grid with 10 channels, along with 4 scalar values: gripper open state, finger joint position, left finger joint position, and timestep. Additionally, our AR2-D2 framework provides 3 tuples of values: translation, rotation, and gripper state. PerAct utilizes a Perceiver Transformer as its main backbone, employing 6 self-attention layers in its implementation. During the fine-tuning phase, we choose to freeze all 6 self-attention layers exclusively, focusing on refining other parts of the model. Furthermore, for simulation data collection, we use RLBench [1] and the idea of generating key-frames [2] to generate the required dataset for training PerAct, we further employ domain randomization by rendering different textures across the demonstrations in attempt to bridge the sim2real gap when transferring into the real world robot (see Figure 3).

Setup. In order to ensure a fair comparison, we carefully arranged the objects in relative distances across all three domains - AR, real-world, and simulation - for the two tasks used in our user study. For the SUS (System Usability Scale) score survey, we employed a modified version of the default question template commonly utilized in similar projects. The survey consists of ten questions, designed to assess the participants' perception of the system:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

We have also included a list of likert scale plot for each questions across the 4 other methods of demonstration collection as shown Figure 2.

References

[1] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[2] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022.

[3] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv:2209.05451*, 2022.

[4] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.