Supplementary: Human-to-Robot Imitation in the Wild

A. *Videos*

Videos of our results can be found at: https://human2robot.github.io

B. *Implementation Details*

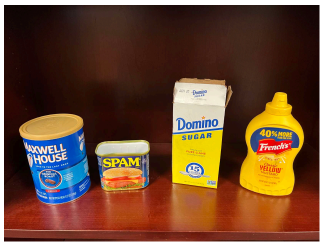
1) *Real Robot Setup*

We use use the Stretch robot from Kemp et al. [2]. This is a robot with a mobile base and a wrist with suction cups as fingertips. All 6 degrees of the freedom of the robot are controllable, and we use code provided in https://github.com/hello-robot to control the robot. To capture videos of humans and the robot we use an Intel Realsense D415. We obtain both depth and RGB images from this camera setup. Each human demonstration takes about 30 seconds. Similarly, each robot episode also takes 30-45 seconds. Overall training takes anytime between 4 and 6 hours, including time to compute inpainted videos (which is the bottleneck in terms of software).

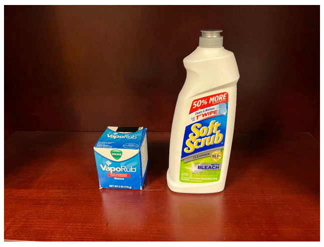
Our real robot tasks are all in the wild (i.e. outside labs). We perform tasks using everyday objects and in every day locations such as kitchens etc. Due to torque limits on the robots, we had to use non-standard objects for some settings, such as the ball in hoop task, as the robot's gripper cannot grasp a heavier and bigger basketball. In Figure 2 we present a full list of tasks with images. We show details on train and test objects for our shelf pick-and-place task in Figure 1.

2) *Data Collection*

Human demonstrations are very easy obtain and each takes about 30 seconds to collect. The tasks presented in Figure **??** are trained from one demonstration. During each iteration, 30 samples were taken, and we used the top 10 ranking ones to fit the policy. The Stretch robot [2] took less than 1 minute per episode, thus around 20 minutes per iteration.



(a) Training Objects



(b) Pouring

Fig. 1: Images of the objects we used in our shelf pick-and-place task. Objects a-d are train objects, objects e and f are test objects

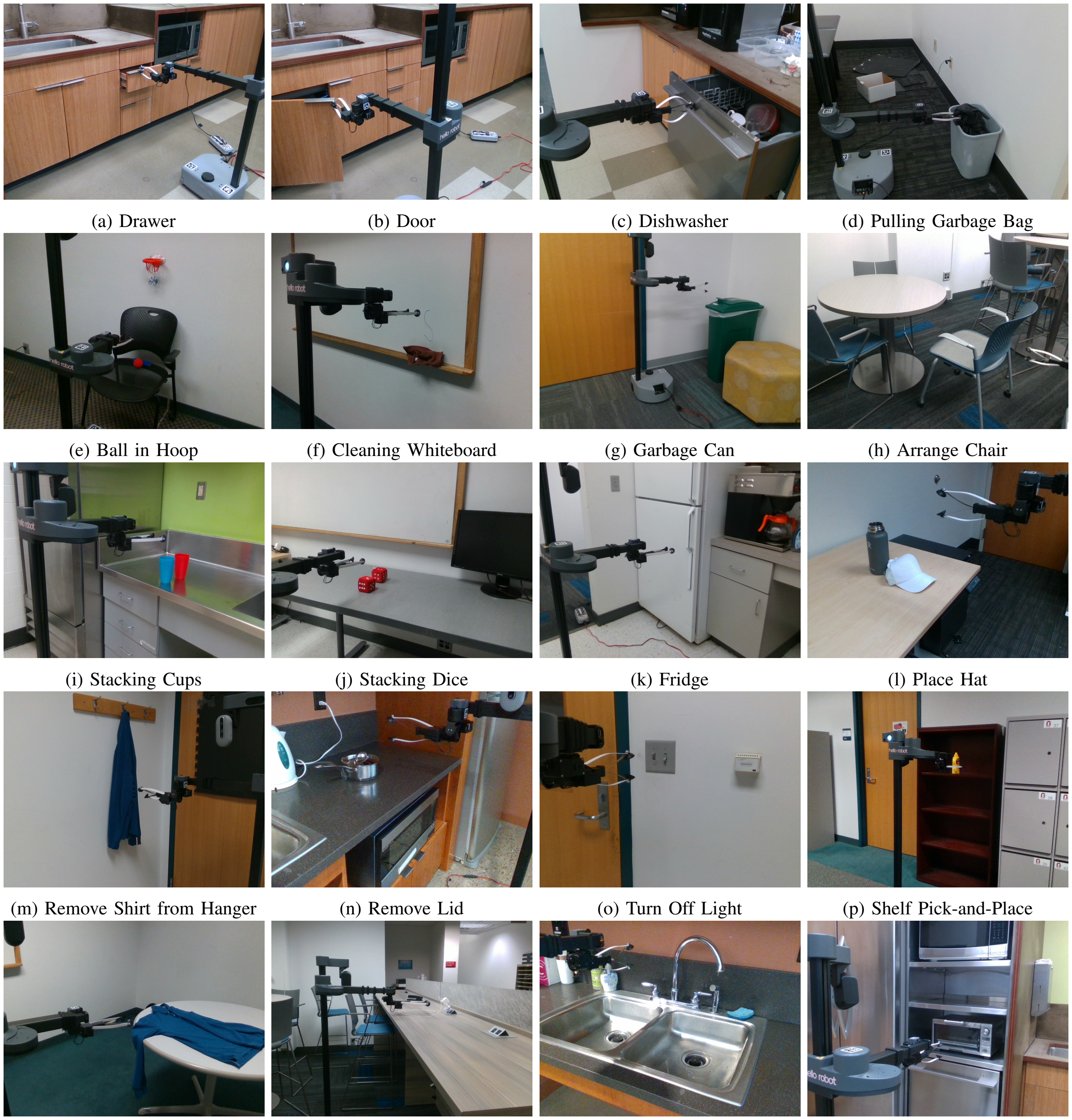
3) *Hyper-parameters and Design Choices*

Our policy is a 4 layer MLP, which takes as input an embedding of a demonstration video as well as the prior. The output of the policy is the residual to the prior. We process the human video using an action-recognition pipeline, using features from state-of-the-art pretrained action recognition models such as SlowFast 3D ResNets [1]. We train the policy as a Variational Auto-Encoder, using a KL-divergence loss with weight 0.0005 to train the model and latent dimension = 4. However, larger latent dimensions work well too, but this should be dependent on the size of the action space for the robot (which in our case is 13 dimensional). Our exploration policy is structured in a similar manner. For the human prior, we use the hand-object interaction detector from Shan et al. [7]. We employ Copy-Paste Networks [3] for inpainting humans and robots from videos. We use action recognition models such as Multi-Moments [4] and SlowFast 3D ResNets [1] as our representation space (Φ) for aligning humand and robot videos. Furthermore, for measuring "change" in the environment we used features from the VGG16 [8] network. Our exploration policy uses the same exact architecture. We use the following video augmentations for our representations: Salt-Pepper Jittering, Random Crops, Gaussian Blurs, Vertical and Horizontal Flips). For the hand-object and wrist detection modules we used default parameters provided by the respective codebases. To smooth the predictions we used the filter from https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol_filter.html. We used about 200 labeled images of the robot to train the instance segmentation module, with the default network sizes from the codebase. All of our image and video sizes were 640 x 480. For the policy training module, our optimization approach fits to the top 10 (out of 30) samples.

4) *Codebases*

We use the following codebases:

- For hand-object detection we use the codebase from Shan et al. [7] , https://github.com/ddshan/hand_detector.d2
- For wrist detection we use FrankMocap [5], https://github.com/facebookresearch/frankmocap
- For the instance segmentation module we use code from https://github.com/wkentaro/labelme to label robot instances, and use code from Detectron2 [10] (https://github.com/facebookresearch/detectron2) code provided in https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html to train an instance segmentation model.
- For the TCN [6] baseline we use code from https://github.com/kekeblom/tcn
- For the CycleGAN baseline we use code from https://


(a) Drawer (b) Door (c) Dishwasher (d) Pulling Garbage Bag

(e) Ball in Hoop (f) Cleaning Whiteboard (g) Garbage Can (h) Arrange Chair

(i) Stacking Cups (j) Stacking Dice (k) Fridge (l) Place Hat

(m) Remove Shirt from Hanger (n) Remove Lid (o) Turn Off Light (p) Shelf Pick-and-Place

(q) Fold Shirt (r) Pull Plug from Socket (s) Open Tap (t) Toaster

Fig. 2: Images of our 20 tasks.

github.com/Lornatang/CycleGAN-PyTorch

- Our Inpainting model [3]: https://github.com/shleecs/Copy-and-Paste-Networks-for-Deep-Video-Inpainting
- We use the model from Monfort et al. [4] (https://github.com/zhoubolei/moments_models) to compute representations for our cost functions
- Our offline RL baselines are from Takuma Seno [9] (https://github.com/takuseno/d3rlpy)
- https://github.com/okankop/vidaug

REFERENCES

[1] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 1

[2] Charles C Kemp, Aaron Edsinger, Henry M Clever, and Blaine Matulevich. The design of stretch: A compact,

lightweight mobile manipulator for indoor human environments. *arXiv preprint arXiv:2109.10892*, 2021. 1

[3] Sungho Lee, Seoung Wug Oh, DaeYeun Won, and Seon Joo Kim. Copy-and-paste networks for deep video inpainting. In *ICCV*, 2019. 1, 2

[4] Mathew Monfort, Bowen Pan, Kandan Ramakrishnan, Alex Andonian, Barry A McNamara, Alex Lascelles, Quanfu Fan, Dan Gutfreund, Rogerio Feris, and Aude Oliva. Multi-moments in time: Learning and interpreting models for multi-action video understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1, 2

[5] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In *CVPR (ICCV) Workshops*, pages 1749–1759, October 2021. 1

[6] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018. 1

[7] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9869–9878, 2020. 1

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[9] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021. 2

[10] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 1