Supplementary Material
Affordances from Human Videos as a Versatile Representation for Robotics

Shikhar Bahl[⋆1,2] Russell Mendonca[⋆1] Lili Chen[1] Unnat Jain[1,2] Deepak Pathak[1]

[1]CMU [2]Meta AI

1. Result Videos

Videos are available in the supplementary material and our website: https://vision-robotics-bridge.github.io.

2. Affordance Model Setup

Data Extraction: Our training setup involves learning from EpicKitchens-100 Videos [1]. This dataset contains many hours of videos of humans performing different kitchen tasks. We use each sub-action video (such as 'open door' or 'put cup on table') as training sequences. Consider a video (V) consisting of T frames, $V = \{I_1, ..., I_T\}$. Using 100 DOH annotations [12] (available alongside the dataset), we find all of the hand-object contact points and frames for each hand in the video. As mentioned in Section 3, let model output $f_{\text{hand}}(I_t) = \{h_t^l, h_t^r, o_t^l, o_t^r\}$, where o^l, o^r are the contact variables and h^l, h^r are the hand bounding boxes. We find the first contact timestep and select the active hand (left or right) as the hand side to consider for the whole trajectory. This is found by first binning o_t and looking for all types that have contact with 'Portable' or 'Fixed' objects. These are assigned 1, while all others are assigned 0. We smooth the set of contact variables using a Savitzky–Golay filter [11] using a threshold of 0.75 (with window size 7). This should eliminate any spurious detections. We use the skin segmentation approach from [7], to find the contact points, $\{c_i\}^N$, at the contact timestep around the active hand. We then fit a GMM with $k = 5$ to the set of contact points to determine $\mu_1, ..., \mu_5$. We found that learning without a covariance, Σ, was more stable thus we only aim to learn the μ_1. The input image becomes the first image before the contact where the hand is not visible. If the contact points or trajectory are not in the frame of this initial image (if the camera has moved), we then discard the trajectory. We use crops of size 150x150 (full image size is 456 x 256), which improves robustness at test time. We train on around 54K image-trajectory-contact point tuples. We include visualizations of the affordance model outputs on our website: https://vision-robotics-bridge.github.io/.

Architecture:

We use the ResNet18 encoder from [10] as g_ϕ, as our visual backbone. Our model has two heads, a trajectory head and a contact point head. We use the spatial features from the ResNet18 encoder (before the average pooling layer) as an input to three deconvolutional layers and two convolutional blocks with kernel sizes of 2 and 3 respectively, and channels: [256, 128, 64, 10, 5]. We use a spatial softmax to obtain $\hat{mu}_k$ for where $k = 1, ..., 5$. Our trajectory network is a transformer encoder with 6 self-attention layers with 8 heads each, and uses the output of the ResNet18 encoder (flattened), which has dimension 512. The output of the transformer encoder is used to predict a trajectory of length 5, using an MLP with two layers with hidden size 192.

Training: We train our model for 500 Epochs, using a learning rate of 0.0001 with cosine scheduling, and the ADAM [5] optimizer. We train on 4 GPUs (2080Ti) for about 18 hours.

3. Robotics Setup

Hardware setup: For all the tasks we assume the following structure for robot control for each trajectory. We first sample a rotation configuration for the gripper. The arm then moves to the contact point c, closes its gripper, and moves to the points in the post-contact trajectory τ. For the initial rotation of the Franka, joints 5 and 6 can take values in [0, 30, 45] degrees, while joint4 is fixed to be 0 degrees. For the Hello-Robot, the roll of the end-effector is varied in the range of [0, 45, 90] degrees. Once the orientation is chosen for the trajectory, we perform 3DOF end-effector control to move between points. Given two points a and b, we generate a sequence of waypoints between them to be reached using impedance control for the Franka. The Hello-Robot is axis aligned and has a telescoping arm, thus we did not need to build our own controller. We do not constrain the orientation to be exactly the same as what was selected

⋆equal contribution

in the beginning of the trajectory, since this might make reaching some points infeasible. For all tasks and methods we evaluate success rate by manual inspection of proximity to the goal image after robot execution (for imitation learning, goal reaching and affordance as an action space), and evaluate coincidental success for exploration using manual inspection of whether the objects noticeably move over the course of the robot's execution trajectory. We provide larger versions of the result plots of successes presented in the main paper in Figures 2 and 3.

Affordance Model to Robot Actions Reusing terminology from Section 3, the affordance model output is $f_\theta(I_t) = \hat{p}_c, \hat{\tau}$, where $\hat{p}_c = \sum_{k=0}^{K} \alpha_k \mathcal{N}(\hat{\mu_k}, \hat{\Sigma_k})$, and $\hat{\tau} = \{w_i\}^M$. We can convert this into a 3D set of waypoints using a hand-eye calibrated camera, and obtain a 3D grasp point from $\hat{p}_c$, and a set of 3D waypoints from $\hat{\tau}$.

Imitation from Offline Data Collection - We use our affordance model to collect data for different tasks, and then evaluate whether this data can be used to reach goal images using k-NN and Behavior cloning. As mentioned in Sec 3.3.1, given an image I_t, the affordance model produces $(c, \tau) = f_\theta(I)$. In addition to storing I_t, c and τ, we also store the sequence of image observations (queried at a fixed frequency) seen by the robot when executing this trajectory $O_{1:k}$, where k is the total number of images in the trajectory. k varies across different trajectories (since it depends on c and τ). These intermediate images O_i enable us to determine how close a trajectory is to the given goal image. For each trajectory, the distance to goal image I_g is given by $\min_i ||\psi(I_g) - \psi(O_i)||_2^2$, where ψ is the R3M embedding space. We then use this distance to produce a set of K trajectories with smallest distances to the goal I_g. For k-NN, we simply run (c, τ) from each of these filtered trajectories. For Behavior cloning, we first train a policy that predicts (c, τ) given image I using this set of trajectories, and then run the policy π on the robot. We summarize this is Algorithm 1. We fix the number of top trajectories K to be 10 for k-NN and 20 for behavior cloning. The number of trajectories for initial data collection used for each task is listed in 2. For k-NN, the success is averaged across all K runs on the robot. For behavior-cloning, we parameterize the policy π using a CVAE, where the image is the context, the encoder and decoder are 2 layer MLPs with 64 hidden units and the latent dimension is 4. During inference, we sample from the CVAE given the current image as context, and report success averaged across 10 runs. The quality of data collected by the robot using VRB which is used for imitation can be in seen in the videos on our website: https://vision-robotics-bridge.github.io/.

Although many of our household object categories might be present in the videos of Epic-Kitchens [1], specific *instances* of objects do not appear in training, thus every object our approach is evaluated on is new. To test generalization to "rare" (held-out) objects and evaluate the grasping success using VRB's affordances, see Table 1. VRB consistently outperforms our most competitive baseline, Hotspots [9].

Object	VRB	Hotspots
VR Controller	**0.27**	0.13
Chain	**0.33**	0.20
Hat	0.07	**0.20**
Tape	**0.13**	0.00
Cube	0.00	0.00
Sanitizer	**0.27**	0.20
Stapler	**0.53**	0.20
Shoe	**0.33**	0.13
Mouse	**0.27**	0.00
Hair-Clip	**0.47**	0.20

Table 1. VRB for grasping held-out "rare" objects

Algorithm 1 Imitation from Offline Data Collection

Require: Dataset of trajectories $\{(I_t, O_{1:k}, c, \tau)\}$
Require: Number of top trajectories K
Require: Goal Image I_g
Require: R3M embedding space ψ
For each trajectory $\mathcal{T}$, compute $d_\mathcal{T} = \min_i ||\psi(I_g) - \psi(O_i)||_2^2$
Rank trajectories in ascending order of $d_\mathcal{T}$. Create set $\mathcal{K} = \{(c, \tau)\}$ of the top K ranked trajectories.
if k**-NN then**
    Execute $\mathcal{K}$ on the robot.
else
    Assert **behavior cloning**
    Train a policy $\pi(c, \tau | I)$ using $\mathcal{K}$.
    Execute $c, \tau \sim \pi(.|I)$ on the robot.
end if

	Cabinet	Knife	Veg	Shelf	Pot	Door	Lid	Drawer
N_0	150	100	50	50	50	50	30	40
N_s	50	50	30	30	30	50	30	40

Table 2. Number of trajectories collected for various tasks, for Initial Data Collection (N_0) and for each subsequent fitting iteration for either goal reaching or exploration (N_s)

Exploration & Goal Reaching: We apply our affordance model in the paradigms of exploration as well as goal reaching, where the robot uses the collected data to improve its behavior. As described in Section 3.3, we use a *environment change* visual model to obtain intrinsic reward for exploration, while for goal-reaching we use *distance to the goal* in a feature space like the R3M embedding space. For exploration, we want to *maximize* the change between the first and last images of the trajectory, since

(a) VRB - Cabinet (b) HAP - Cabinet (c) VRB - Veggies (d) HAP - Veggies

Figure 1. Visualization for Affordance as an Action Space for VRB and HAP [3], on the Cabinet and Veggies Tasks

Algorithm 2 Exploration / Goal Reaching

Require: Number of iterations J
Require: Number of top trajectories K
Require: Number of initial trajectories N_0, and for subsequent fitting iterations N_s
Require: Affordance model f_θ
Require: Tradeoff probability p
Require: Visual change model Φ (only for **exploration**)
Require: R3M embedding ψ (only for **goal reaching**)
Require: Goal Image I_g (only for **goal reaching**)
initialize: World model $\mathcal{M}$, Replay buffer $\mathcal{D}$,
Execute $(c, \tau) = f_\theta(I)$ on the robot for N_0 iterations to collect initial dataset $\mathcal{D}$ = $\{(I, O_{1:k}, c, \tau)\}$
for iteration 1:J **do**
  For each trajectory $\mathcal{T}_{0:k}$, compute
  if exploring then
    compute $\text{EC}_\mathcal{T} = ||\phi(O_1) - \phi(O_k)||_2$
    Rank trajectories in descending order of $\text{EC}_\mathcal{T}$
  else
    Assert **goal reaching**
    compute $\text{d}_\mathcal{T} = \min_i ||\psi(I_g) - \psi(O_i)||_2$
    Rank trajectories in ascending order of $d_\mathcal{T}$
  end if
  Create set $\mathcal{K}$ = $\{(c, \tau)\}$ of top K ranked trajectories.
  Compute $\hat{c}, \hat{\tau}$ = mean($\mathcal{K}$)
  For N_s iterations, set (c, τ) = $f_\theta(I)$ with probability p, otherwise set $(c, \tau) = (\hat{c}, \hat{\tau})$.
  Execute (c, τ) on the robot and append data to $\mathcal{D}$
end for

Algorithm 3 Affordance as Action Space

Require: Affordance Model f_θ
Require: Number of initial queries q
Require: Number of clusters for c, N_c and for τ, N_τ
Require: Goal Image I_g
Require: RL algorithm with discrete action-space RLA
Require: R3M embedding space ψ
Query f_θ on the image of the scene q times to obtain a dataset $\{(c, \tau\}$
Fit a GMM G_c with N_c centers to $\{c\}$, and a GMM G_τ and N_τ centers to $\{\tau\}$
Create mapping $\mathcal{M}$ from $\mathcal{A}$ = [1..N_c*N_τ] to values in the cross-product space of the centers of G_c and G_τ
Initialize Dataset $\mathcal{D}$ = $\{\}$, and RLA with discrete action space $\mathcal{A}$ and random policy π.
Run **Sampling** and **Training** asynchronously
while Sampling do
  Run π on the image to get a_d.
  $(c, \tau) = \mathcal{M}(a_d)$, execute on the robot and collect initial and final images I_0 and I_T
  Compute reward $r = ||\psi(I_T) - \psi(I_g)||_2$.
  Store $(\psi(I_0), a_d, \psi(I_T), r)$ in $\mathcal{D}$
end while
while Training do
  Sample data $\sim \mathcal{D}$, pass to RLA for training and updating π.
end while

greater perturbation of objects can lead to the discovery of useful manipulation skills. For goal-reaching, we *minimize* the distance between the trajectory and the goal image, since this achieves the desired object state. In each case (exploration and goal-reaching), we rank the trajectories in the dataset using the appropriate metric, and then fit $(\hat{c}, \hat{\tau})$ to the $\{(c, \tau)\}$ values of the top ranked trajectories. For subsequent data collection iterations, we use the affordance model f_θ with some probability p, but otherwise use $(\hat{c}, \hat{\tau})$ for execution on the robot. The newly collected data is then aggregated with the dataset, and the entire process repeated. We present this procedure in Algorithm 2. The number of initial trajectories N_0 and trajectories for subsequent iterations N_s for different tasks are listed in 2. For all experiments, we set p = 0.35, K = 10, J = 2. We include videos on our website (https://vision-robotics-bridge.github.io/) which show that as our system sees more data, its performance improves for both explo-

ration and goal-reaching.

Intrinsic Reward Model We train a visual model which given a pair of images (I_i, I_j), produces a binary image that captures how *objects* move, and is not affected by changes in the robot arm or body position. Specifically, this model comprises the following -

$$\phi(I_i, I_j) = g(||m(I_i) - m(I_j)||_2, \\ ||\Psi(m(I_i)) - \Psi(m(I_j))||_2) \quad (1)$$

Here m is a masking network which removes the robot from the image. We train this using around 100-200 hand-annotations of the robot in various scenes, and use this data to finetune a pretrained segmentation model Ψ [4]. We evaluate the l2-losses above only on **non-masked** pixels. Further, we also take into account distance in the feature space of the segmentation model to reduce sensitivity to spurious visual artifacts. The function g applies heurestics including gaussian blurring to reduce effects of shadows, and a threshold for the change at each pixel, to limit false positives.

Affordance as an Action Space: For this learning setup, we parameterize the action space for the robot with the output distribution of our affordance model. We first query the model a large number of times, and then fit Gaussian Mixture Models (GMMs) separately to the c and τ predictions, with N_c and N_τ centers respectively. We then define a discrete action space of dimension N_c*N_τ, where each action maps to a value in the cross-product space of the centers of the two GMMs. We can now use discrete action-space RL algorithms. We asynchronously sample from the discrete action-space policy, and train it using the RL algorithm. This procedure is described in Algorithm 3. We note that it is important to reset the environment so that images the policy sees are close to the initial image for which the action space was defined. Across experiments we set N_c = N_τ = 4, q = 2000. For the RL algorithm RLA we use the Deep Q-Network (DQN) [8] implementation from the d3rlpy [13] library. We include a visualization of the action space by plotting the (c, τ) values in the cross-product space of the centers of the two GMMs, for VRB and HAP [3] in Figure 1. We see that for VRB a larger number of the discretized actions are likely to interact with the objects.

4. Baselines and Ablations

Baselines The baselines we compare to include the approaches from from Liu et al. [7] (HOI), Goyal et al. [3] (HAP) and Natarajan et al., (Hotspots) [9]. In each of these baselines, we used the provided pretrained model. Specficially, for Hotspots [9], we employ the model trained on EpicKitchens [1], as this is what our approach is also trained on. Similarly, for HAP [3] we use the trained model on EpicKitchens also. HOI predicts both a contact point and trajectory, which we execute at test time. The other two approaches predict likely contact regions, from which we sample, as well as a random post contact trajectory.

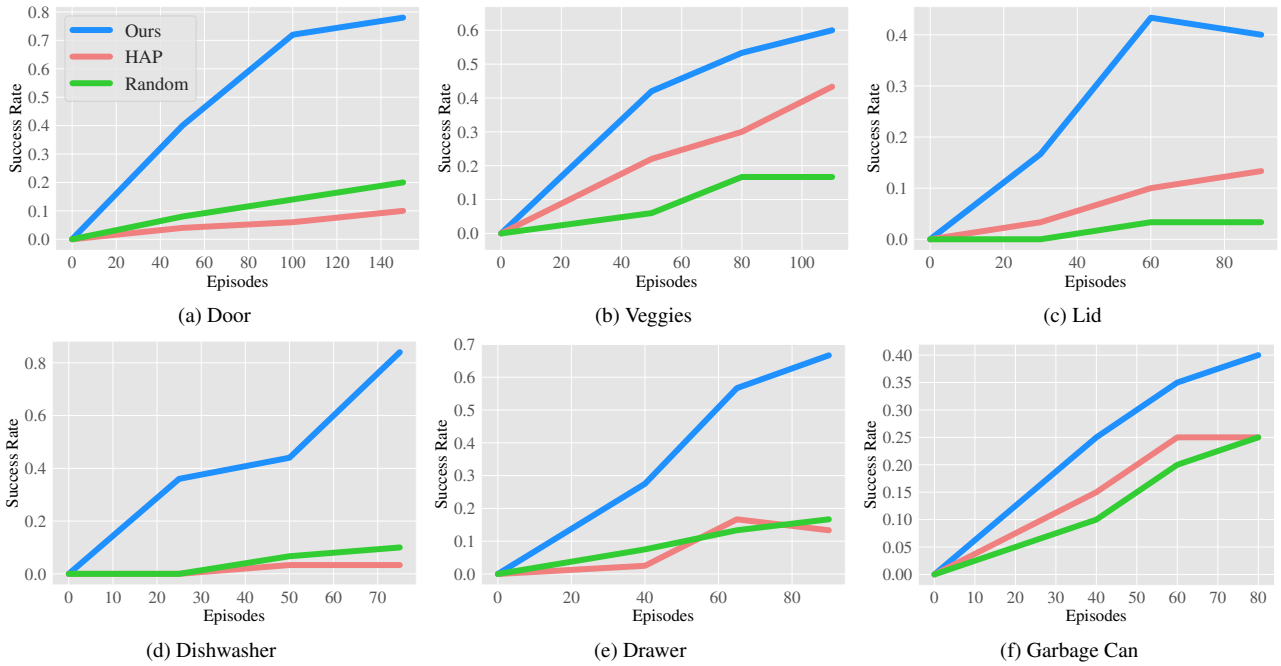
Visual Representation Analysis (Finetuning): For the visual representation finetuning experiments we performed in Section 4.5, we use the Imitation Learning Evaluation Framework from R3M [10], which aims to evaluate the effectiveness of frozen visual representations for performing behavior cloning for robotic control tasks. Following their procedure, we evaluate on three simulated tasks from the Franka Kitchen environment: (1) microwave, (2) slide-door, and (3) door-open. We train the policy using left camera images from their publicly available demonstration dataset, which is collected by an expert state-based reinforcement learning agent and then rendered as image observations.

For behavior cloning with the R3M encoder, we freeze the pretrained R3M encoder (which uses a ResNet50 base architecture) and finetune a policy on top of it. For behavior cloning with the VRB encoder, we instead use an R3M model which was finetuned for 400 steps with affordance model training as in Section 3.2. Note that this finetuning was performed separately from behavior cloning, and during policy learning our representations are also frozen before being used as input for the downstream policy. For both R3M and VRB, we concatenate the visual embedding and proprioceptive data for input to the downstream policy, and then use a BatchNorm layer followed by a 2-layer MLP to output an action. The downstream policy is trained with a learning rate of 0.001 and a batch size of 32 for 2000 steps.

Visual Representation Analysis (Feature space distance): For the feature space distance experiments, we compare an R3M model with a VRB model. Both use a ResNet50 base architecture, and the VRB model is obtained by finetuning an R3M model for 100 steps using affordance model training as in Section 3.2. The distances in Figure 8 are computed as the (squared) L2 distances between the features produced by each model for the goal image and current image.

5. Simulation

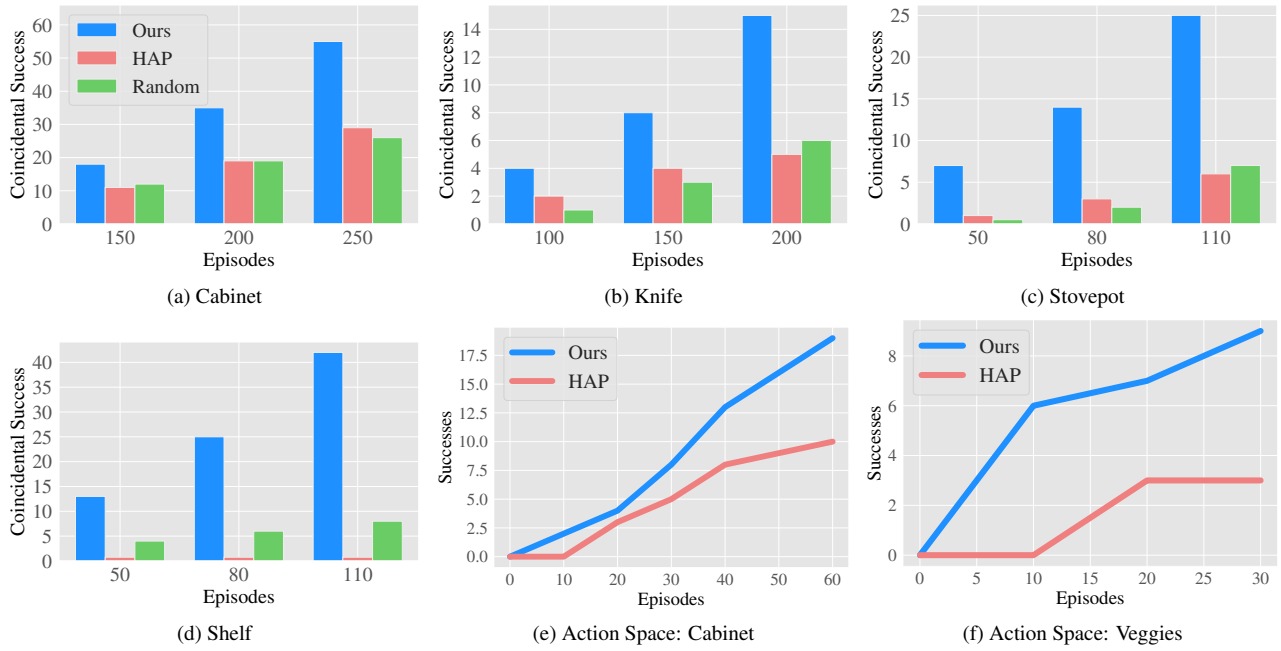
We also provide a simulation environment benchmark to test our affordances. This is modeled after the Franka-Kitchen environment from the D4RL [2]. In this benchmark, the robot observes images and predicts 3D positions to manipulate, in the exact same way as we deploy the robot in the real world. An image of this environment can be seen in Figure 4. There are three different tasks: turning the light on, opening the microwave and lifting the kettle. These are standard tasks in the D4RL benchmark [2]. We run Paradigm 1 (offline data collection) and provide the success rates for VRB and baselines in Table 3. We can see that VRB significantly outperforms the baselines.



(a) Door (b) Veggies (c) Lid

(d) Dishwasher (e) Drawer (f) Garbage Can

Figure 2. **Goal-conditioned Learning**: Success rate for reaching goal configuration for six different tasks. Sampling via VRB leads to faster learning and better final performance.



(a) Cabinet (b) Knife (c) Stovepot

(d) Shelf (e) Action Space: Cabinet (f) Action Space: Veggies

Figure 3. **Exploration** and **Action Space Parameterization**: Coincidental success (stumbling onto goal configurations) increases multiple folds with VRB in comparison to random exploration or the exploration based on HAP [3] in a-d. In e-f, we see the success numbers of using DQN with the discretized action space, for reaching a specified goal image.

6. Codebases

We use the following codebases:

https://github.com/epic-kitchens/epic-kitchens-100-hand-object-bboxes for extracting detections from 100 DOH [12] for EpicKitchens [1].

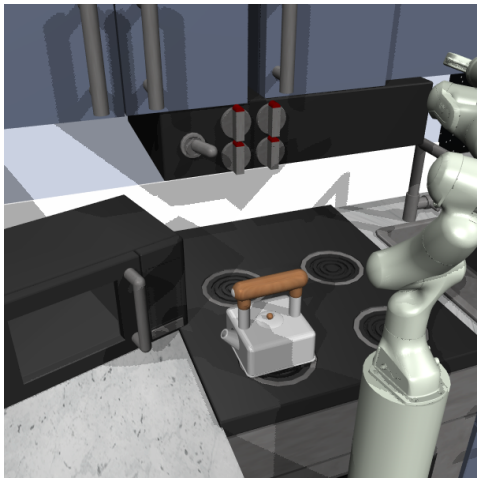


Figure 4. Simulation Environment from [2]

Method	Light	Microwave	Kettle
Random	0.20	0.15	0.20
HAP	0.30	0.20	0.45
HOI	0.60	0.45	0.40
Hotspots	0.35	0.35	0.25
VRB	**0.75**	**0.60**	**0.55**

Table 3. VRB on simulation benchmarks.

https://github.com/stevenlsw/hoi-forecast for Skin segmentation code and HOI baseline [7].

https://github.com/uiuc-robovision/hands-as-probes for HAP baseline [3].

https://github.com/Tushar-N/interaction-hotspots for Hotspots baseline [9].

https://github.com/facebookresearch/r3m for R3M visual features [10].

https://github.com/wkentaro/labelme for getting masks for robot and https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html for a Mask-RCNN [4] implementation.

https://github.com/takuseno/d3rlpy [13] for DQN [8] implementation.

https://github.com/facebookresearch/fairo/tree/main/polymetis [6] as the base for the controller for the Franka Arm.

References

[1] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 4, 5

[2] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 4, 6

[3] Mohit Goyal, Sahil Modi, Rishabh Goyal, and Saurabh Gupta. Human hands as probes for interactive object understanding. In *CVPR*, 2022. 3, 4, 5, 6

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 4, 6

[5] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 1

[6] Yixin Lin, Austin S. Wang, Giovanni Sutanto, Akshara Rai, and Franziska Meier. Polymetis. https://facebookresearch.github.io/fairo/polymetis/, 2021. 6

[7] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *CVPR*, 2022. 1, 4, 6

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. 4, 6

[9] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *ICCV*, 2019. 2, 4, 6

[10] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. 1, 4, 6

[11] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8), 1964. 1

[12] 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, 5

[13] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021. 4, 6