DexVIP: Learning Dexterous Grasping with Human Hand Pose Priors from Video: Supplementary Material

Priyanka Mandikal[1,2] **and Kristen Grauman**[1,2]
[1]Department of Computer Science, The University of Texas at Austin
[2]Facebook AI Research
{mandikal,grauman}@cs.utexas.edu

Note: Please see the supplementary video for example episodes.

Contents

A Noisy sensing and actuation

We are so far unable to deploy our system on a real robot, since we lack access to a dexterous hand robot. Instead, we provide experiments with a popular realistic simulator and further stress-test our approach with noisy sensing and actuation. Those results appear in the main paper; here we elaborate on the noise models.

Robots can encounter a number of non-ideal scenarios when executing policies in the real world. The ever changing nature of the real world coupled with faults in hardware systems can pose a daunting challenge to real world deployment. These discrepancies often occur in the form of sensing and actuation failures. Sources for noise include variations in sensory systems such as perception modules as well as fluctuations in actuation control. Before robots can successfully be deployed into the real world, they must be capable of handling such variations.

5th Conference on Robot Learning (CoRL 2021), London, UK.

In Section 4 of the main paper, we describe the setup for inducing noise into our agent's sensory and actuation modules during training and testing following prior work [1, 2, 3, 4]. Here, we further describe each of the noise sources in detail.

1. Proprioceptive noise: We apply additive Gaussian noise of mean 0 and standard deviation 0.01 on the robot's joint angles and angular velocities. This simulates the sensing and signal failures that can arise in the system. Thus training with such an induced noise source improves the likelihood of the robot being more robust to hardware failures during actuation.
2. Actuation noise: Similar to the proprioceptive noise, we apply additive Gaussian noise on the robot actuation values. Such a noise model accounts for fluctuations in actuation control experienced in real-world deployment.
3. Perception noise: For each RGB image I_t^r that is processed by the vision module f_V, we apply pixel perturbations in the range $[-5, 5]$ and clip all pixel values between $[0, 255]$. This more closely resembles noise arising during camera sensing [1].
4. Tracking noise: The object tracking points are operated on by a Gaussian noise model of mean 0 and standard deviation 1 cm. Additionally, we also freeze these tracking points for 20 frames at random intervals to further challenge our system. This simulates the effect of tracking failures arising from momentary occlusions of the object during interaction.

As can be seen in Fig. 5 (left) of the main paper, even under substantial noise, our proposed method DEXVIP still yields a high grasp success rate comparable to its performance under noise-free conditions, and even outperforms noise-free models of the other methods. This demonstrates the robustness of the trained policy to non-ideal realistic conditions. Since our lab does not have access to a real robot, we perform all our experiments in simulation. However, using the noise-induction techniques described above, we are able to test our method for real-world compliance under realistic conditions.

Some areas that could still vary between simulation and the real world can include friction coefficients, damping factors, and so forth, which could further be accounted for using automatic domain randomization techniques as in [3]. Despite the lack of access to a real robot, the encouraging performance of DEXVIP in a noise-induced simulation environment lends support for potentially transferring the learned policies to the real world [3, 2, 5] were we to gain access to a robot.

B Reward function details

We describe the various components of the reward function in Eq.1 of the main paper in more detail.

1. R_{succ}: This is a positive reward determining if the object has been grasped by the agent. At a particular time step t, if there is contact established between the hand and the object and no contact between the object and the table, the agent gets a $+1$ reward for that time step. This ensures that scenarios where the object is resting on the table as well those in which the object is in the air but out of the agent's reach do not get counted as grasp successes.
2. R_{aff}: This is a negative reward denoting the hand-affordance contact distance between points on the hand and object affordance regions. Following [4], we compute R_{aff} as the negative of the Chamfer distance between M points on the hand and N points on the object. It is defined as follows:

$$R_{aff} = -d_{Chamfer}(M, N) = -\sum_{m \in M} \min_{n \in N} ||m - n||_2^2 - \sum_{n \in N} \min_{m \in M} ||m - n||_2^2. \quad (1)$$

   We set $M = 10$ and $N = 20$ in our experiments, following [4]. In essence, the agent experiences a higher penalty if it is away from the object affordance region, which drops to 0 as it gets closer. This encourages the agent to explore useful object regions during exploration.
3. R_{pose}: As discussed in the main paper, R_{pose} is a mean per-joint angle error applied on the robot joint angles p_t^r upon making contact with the object, so that it matches the human expert pose p_c^*. We ignore the azimuth and elevation values of the arm in the pose error since they are specific to the object orientation in I^h, which may be different from the robot's

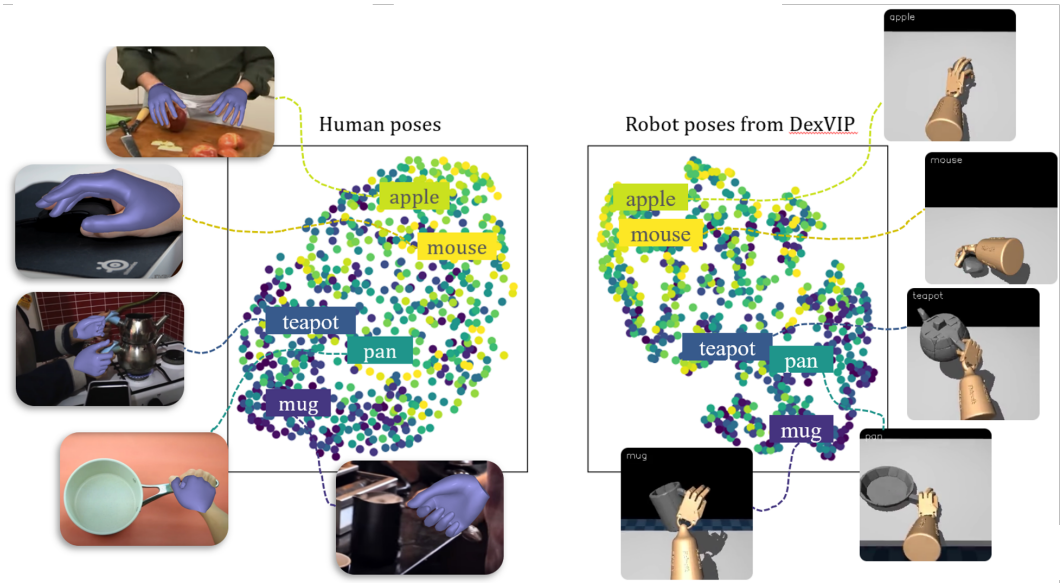


Figure A: **TSNE over hand poses.** A good distribution of human and robot poses across morphologies is observed (e.g. clenched fist - mug, pan, teapot vs. loose fist - apple, mouse).

viewpoint I^r. This joint error is also hierarchically weighted over the joint angles such that errors on the parent joints are more heavily penalized compared to those on the child joints. This encourages the agent to align parent joints first before aligning their children and thus adopts a global to local approach for pose matching. R_{pose} is therefore defined as follows:

$$R_{pose} = \gamma_1 l_{wrist} + \sum_{j=1}^{5} \left(\gamma_2 l^j_{knuckle} + \gamma_3 l^j_{middle} + \gamma_4 l^j_{distal}\right). \tag{2}$$

Here, j spans all five fingers of the hand and $l^j_i$is the error between joint i of the j^{th} finger of the robot pose p^r_t and target robot pose $p^r_{c^*}$. In our experiments, we set $\gamma_1 = 1.0, \gamma_2 = 0.75, \gamma_3 = 0.5, \gamma_4 = 0.25$. In this way, we can have the poses align faster starting from the root joint. R_{pose} is a negative reward that penalizes the agent for having poses that are distant from the human pose p^*_c. Furthermore, R_{pose} is applied only when 30% of the robot's touch sensors T^r are activated. This encourages the robot to assume the target hand pose once it is close to the object and in contact with it.

4. $R_{entropy}$: This reward is used while training the PPO agent so as to encourage exploration of the action space. This is implemented by maximizing the entropy over the target action distribution.

C TSNE on hand poses

We perform a TSNE analysis on all the human hand poses $p^h \in \mathcal{P}^h$ in our curated YouTube dataset and the hand poses of our trained grasping agent in Fig. A. We observe a meaningful distribution across object morphologies (e.g. clenched fist – mug, pan, teapot, vs. loose fist – apple, mouse). This behavior is also reflected in the trained DexVIP agent for which we analyze the robot's pose at the last time step, p^r_T of all successful grasps. The distribution for DexVIP is also more clustered since it uses the cluster center per object category p^*_c as the target pose during training. This shows that the proposed approach of injecting human hand pose priors derived from in-the-wild object interaction videos can successfully guide dexterous robotic agents.

D Hand pose retargeting from FrankMocap to Adroit

To use the human pose inferred from FrankMocap in our simulator, we need to re-target the pose from FrankMocap to Adroit. Although both the human hand and the robot hand share a common five-finger morphology, their joint hierarchy trees are different. Fig. B.i shows the two kinematic chains. We briefly describe each morphology below:

- **FrankMocap**: It uses the hand model from SMPL-X [4] to represent the human hand pose inferred from video frames. It consists of three ball joints in each of the five fingers, each having 3 DoF. This yields 15 ball joints in total with 45 DoF. Additionally, the root joint at the base of the hand j_0^h has 6 DoF. The joint space of the human hand in 3D is thus represented as $\boldsymbol{J}^h \in \mathbb{R}^{21\times 3}$ — a wrist, 15 finger joints, and 5 finger tips. Note that the finger tips are not joint locations, but are used for computing joint angles for their parent joints
- **Adroit**: The Adroit hand in the simulator is actuated by 24 revolute joints having 1 DoF each, resulting in 24 DoF in total. The hand is attached to a 6 DoF robotic arm, yielding 30 DoF in total. The joint space of the robot hand is thus represented by $\boldsymbol{J}^r \in \mathbb{R}^{30}$.

As we can see, the FrankMocap model has many more degrees of freedom compared to the Adroit hand. Keeping these differences in mind, we design a joint retargeting mechanism to bridge the gap between the FrankMocap and Adroit models and effectively use the human pose to train the robot.

The hand pose retargeting mechanism is depicted in Fig. B.ii. The different stages of this retargeting pipeline are as follows:

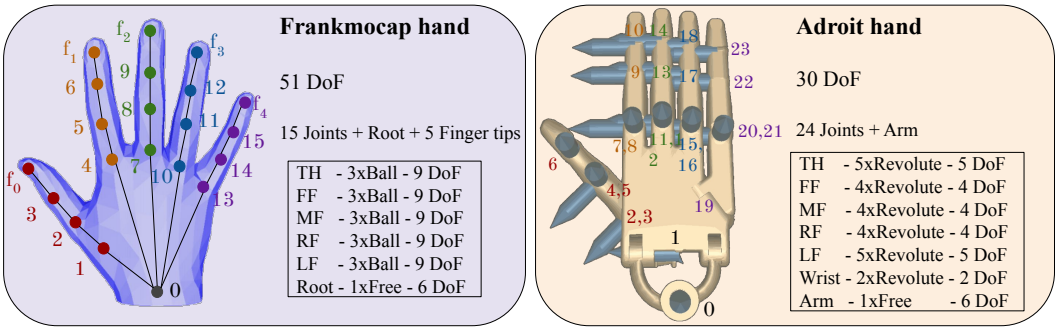
a) Hand pose in world coordinates: We infer the FrankMocap pose from the input video frame I^h and obtain the human hand pose $p^h \in \boldsymbol{J}_w^h$ as a set of 3D joint key-points in world coordinates.

b) World to root relative: We convert raw X, Y, Z positions of the joints in world coordinates to root-relative coordinates by shifting the origin to the wrist joint j_0^h to obtain p_{rr}^h in root relative pose $\boldsymbol{J}_{rr}^h \in \mathbb{R}^{21\times 3}$. To determine the orientation of the Adroit arm, we first construct a plane through the wrist j_0^h, fore finger knuckle j_4^h and ring finger knuckle j_{10}^h. The palm is taken to lie in this plane. The palmar plane along with its normal defines the orientation of the arm. The axis angles obtained during this transformation are used to set the rotational joints of the Adroit arm in $\boldsymbol{J}^r \in \mathbb{R}^{30}$.

c) Root relative to parent relative: A sequential rotational transform is applied about X, Y and Z axes, with the angular changes α, β and γ respectively, on the joint positions in the Root Relative Coordinate System to get the corresponding skeleton in Parent Relative Coordinate System. These angular changes are computed such that the Z-axis lies along the child joint and the Y-axis points outward at every finger joint. At every level of the joint hierarchy, coordinate transformations of the parent are applied to the child joints so that after successively parsing through the entire tree, the root relative coordinates p_{rr}^h are transformed into a parent-relative coordinate frame p_{pr}^h in $\boldsymbol{J}_{pr}^h \in \mathbb{R}^{21\times 3}$. Here every joint j_i is expressed relative to a coordinate frame defined at its parent joint $P(j_i)$.

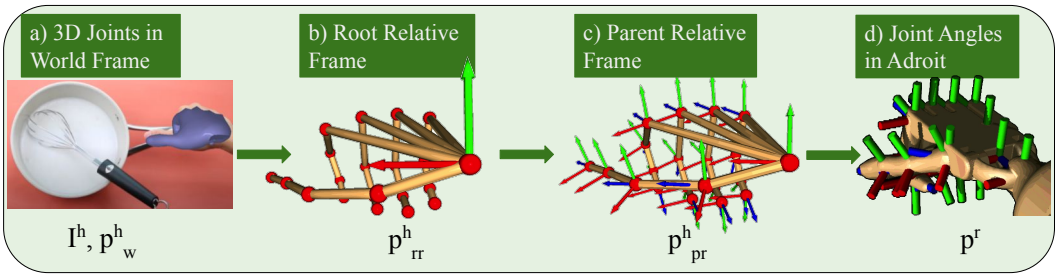
d) Joint angle transfer: The polar coordinates (azimuth and elevations) computed in the parent relative system yield local joint angles that are mapped onto the revolute joints in the Adroit space $\boldsymbol{J}^r \in \mathbb{R}^{30}$. Most revolute joints in Adroit can be mapped from the azimuth or elevation values of different joints in p_{pr}^h. As an example, consider the middle joint on the fore finger in Adroit i.e. j_9^r in Fig. B.i. This joint angle can be obtained by computing the elevation of j_6^h with respect to the coordinate frame defined at its parent joint j_5^h in FrankMocap, while ignoring the azimuth and tilt angles. Other joints can similarly be obtained from the joint definitions in p_{pr}^h. For the little finger metacarpel j_{19}^r which is not modelled in FrankMocap, we set it to be 0.25 of the elevation at j_{13}^h.

Using the above re-targeting scheme, we are able to successfully transfer the FrankMocap pose from the human pose space to the Adroit pose space. Samples can be seen in Fig. 3, main paper. While the mapping is approximate—due to the inherent differences in kinematic chains as discussed above—we find that this re-targeting mechanism generates Adroit poses that closely match the human pose and works well for our purpose.



TH=Thumb, FF = Fore finger, MF = Middle finger, RF = Ring finger, LF = Little finger

i) Kinematic chain for FrankMocap (left) and Adroit (right)



ii) Hand pose retargeting mechanism between FrankMocap and Adroit

Figure B: **Pose retargeting from FrankMocap to Adroit**. **i)** Joint hierarchies for Frankmocap (left) ad Adroit (right). Frankmocap has 15 ball joints with 3 DoF, root with 6 DoF and 5 finger tips. Adroit has 24 revolute joints with 1 DoF and an arm (not shown) with 6 DoF. **ii) Pose retargeting mechanism for transforming Frankmocap joints to the Adroit joint space.** a) We first obtain the Frankmocap pose i.e. 3D joint locations in the world coordinate frame b) This is converted to a root relative coordinate frame through a simple coordinate translation. c) We then compute the palmar plane in Frankmocap to obtain the arm orientation for Adroit. Subsequently, the structure of the kinematic tree is used to successively transform the root relative coordinate frame to a parent relative frame centered at each joint. d) The polar coordinates (azimuth and elevations) computed in the parent relative system yield local joint angles that are mapped onto the revolute joints in Adroit.

E Physical parameters in the simulator

E.1 Parameter settings

We report the physical parameters setting for the simulated environment in Table A. All environment physical parameters are taken from [6]. The sliding friction acts along both axes of the tangent plane. The torsional friction acts around the contact normal. The rolling friction acts around both axes of the tangent plane. Regular friction is present between the hand, object and table. The joints within the hand are assumed to be friction-less with respect to each other.

E.2 Robustness to parameters

To further demonstrate the robustness of the trained policy to different masses and scales of the objects, we evaluate the trained policy on objects with varying masses and scales. Specifically, we vary the mass between $0.5kg$ and $1.5kg$ and the scale between $0.8x$ and $1.2x$ of the training size. Results can be seen in Figure C. We observe that DexVIP remains fairly robust to large variations in these physical properties. It is easier to grasp lighter object than heavier ones as expected.

Parameter	Value
Sliding friction	$1N$
Tortional friction	$0.5N$
Rolling friction	$0.01N$
Hand wrist damping	$0.5N$
Hand fingers damping	$0.05N$
Object rotational damping	$0.1N$
Object mass	$1kg$

Table A: **Physical parameter settings used in the simulator**

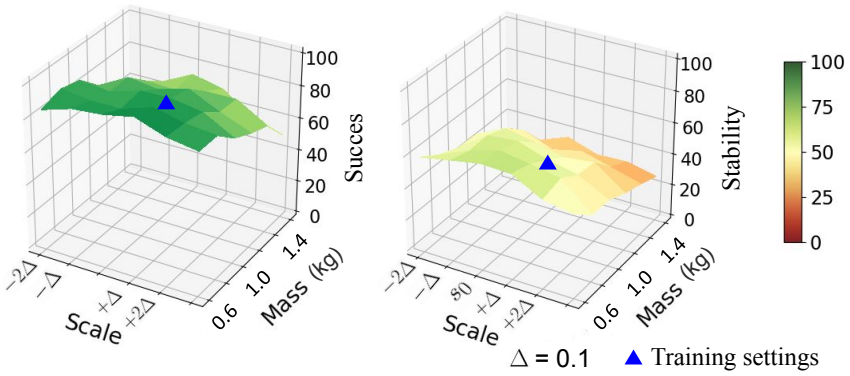


Figure C: **Robustness to changes in physical parameters**. We evaluate DexVIP on a range of object mass and scale values. DexVIP remains fairly robust to large variations in such physical object properties.

F Hand-Affordance distance

The distance input d_t^r is the pairwise distance between the agent's hand and the object affordance region as defined in GRAFF [4]. Here the object affordance region is a 2D binary affordance map that is inferred from a model from [4] that is trained for affordance anticipation on ContactDB objects. Like in [4], we find that this model works reasonably well for objects outside ContactDB as well. We obtain affordance points by back-projecting the affordance map to 3D points in the camera coordinate system using the depth map at t_0. We sample $M = 20$ points from these back-projected points. We then track these points throughout the rest of the episode. In the noise experiments, in addition to the stated noise models, we also induce tracking failure on the affordance points to relax the tracking assumption as in [4]. For points on the hand, we sample $N = 10$ regular points on the surface of the palm and fingers. The number of points at every time step remains the same across all objects.

G Additional results

G.1 Effect of pose prediction on performance

DexVIP uses hand poses inferred from video frames for obtaining target pose priors. Since these poses are inferred, there can be errors in these predictions. To examine the effects of those errors on our policies, we train a model on ground truth (GT) hand poses from ContactPose [7] captured using mocap for all ContactDB objects. Using these GT poses to train the DexVIP policy provides an upper bound for grasp performance with perfect pose. Results are reported in Table B. We find that the policy trained using inferred poses performs comparably to the one trained on GT poses, showing that DexVIP is fairly robust to errors in pose predictions. We further note that the hand pose clustering process that we perform is able to effectively filter out bad/outlier poses so that we obtain a representative hand pose for each object (Fig. D).

Pose Supervision	Success	Stability	Functionality	Posture
Inferred pose	68	51	64	62
GT pose	70	53	65	65

Table B: **Effect of hand pose supervision on grasping performance**. DEXVIP uses hand poses inferred from video frames for supervision. Using ground truth poses captured using mocap to train the DEXVIP policy provides an upper bound for grasp performance. The policy trained using inferred poses performs comparably to the one trained on GT poses indicating robustness to pose errors.

Figure D: **Outlier poses.** The clustering mechanism effectively filters out outlier poses which are produced due to errors in pose prediction.

G.2 Effect of object shape variation on performance

We use keywords containing the object class label for obtaining grasp images for each object. We use the same object category in simulation as well. Note that we do not require an exact matching object instance for the one in the video. A generic object mesh from the same category works quite well. To illustrate this, we show samples of a few objects in the video frame and within the simulator along with their grasp success rate in Fig. E. We find that DEXVIP remains fairly robust to variations in the object shape between the video and simulator. For instance, even though objects like teapot, flashlight and saucepan do not have an exact match in the simulator, the grasp policy works quite well on these objects.

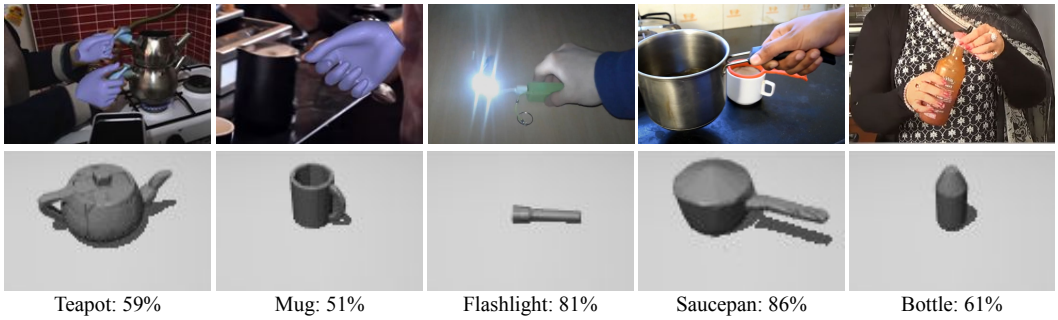


Figure E: **Effect of object shape variation on performance.** DEXVIP remains fairly robust to variations in the object shape between the video and simulator. For instance, even though objects like teapot, flashlight and saucepan don't have an exact match in the simulator, the grasp policy works quite well on these objects.

Model	Success	Stability	Functionality	Posture
Affordance only	60	41	63	48
Affordance + Touch	63	45	64	49
Affordance + Touch + Pose prior (full DEXVIP model)	**68**	**51**	**64**	**62**

Table C: **Metrics for DEXVIP ablations.** The full DEXVIP policy is able to leverage the touch-informed pose prior to perform well across all metrics.

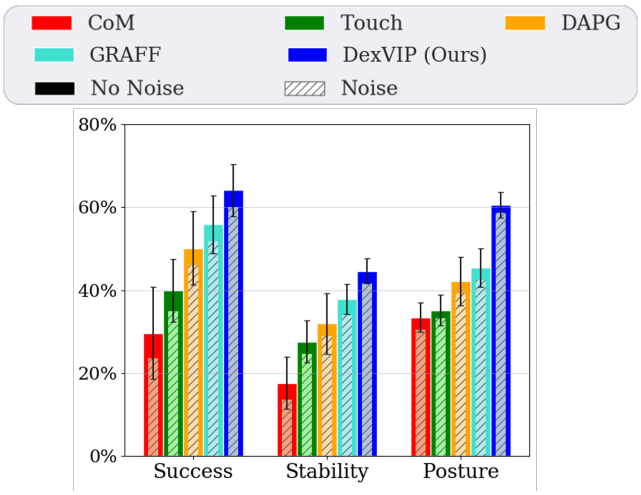


Figure F: **Grasping results on additional non-ContactDB objects.** Compared to the performance on ContactDB, DEXVIP is able to maintain its performance and experiences only a marginal drop whereas the other methods take substantial hits to performance. While the success rate of DAPG and GRAFF drop by 15% and 12% respectively, the drop for DexVIP is only 4%. These results indicate that DEXVIP can effectively leverage hand poses for a variety of different objects.

G.3 Ablation evaluation

We report all metrics for the ablations of the main paper in Table C. We observe that the full DEXVIP model gains substantially in success, stability and posture metrics, while maintaining the functionality score of the policy that is trained using only affordance.

G.4 Performance on additional objects

In addition to comparing performance on non-ContactDB objects with DAPG, we provide a comparison against all methods in Figure F. Note that all 11 non-ContactDB objects belong to object classes not found in ContactDB. When compared to performance on ContactDB, we observe that DEXVIP is able to maintain its performance and experiences only a marginal drop whereas the other methods undergo substantial drops in performance. As reported in L318-320, DAPG's success rate drops from 59%-50%, a 15% drop in performance, while DEXVIP sees a marginal 4% drop from 68% to 65%. Furthermore GRAFF also sees a large 12% drop from 60% to 53%. The results indicate that DEXVIP can effectively leverage hand poses for a variety of objects.

References

[1] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[2] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 2020.

[3] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.
[4] P. Mandikal and K. Grauman. Dexterous robotic grasping with object-centric visual affordances. In *arXiv preprint arXiv:2009.01439*, 2020.
[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.
[6] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Robotics: Science and Systems (RSS)*, 2018.
[7] S. Brahmbhatt, C. Tang, C. D. Twigg, C. C. Kemp, and J. Hays. ContactPose: A dataset of grasps with object contact and hand pose. In *The European Conference on Computer Vision (ECCV)*, 2020.