Learning by Watching: Physical Imitation of Manipulation Skills from Human Videos

Haoyu Xiong[1], Quanzhou Li[1], Yun-Chun Chen[1], Homanga Bharadhwaj[1], Samarth Sinha[1], and Animesh Garg[1,2]
[1]University of Toronto & Vector Institute
[2]Nvidia

***Abstract*—Learning from visual data opens the potential to accrue a large range of manipulation behaviors by leveraging human videos without specifying each of them mathematically, but rather through natural task specification. In this paper, we present Learning by Watching (LbW), an algorithmic framework for imitation from a single human video. The key insights of our method are two-fold. First, since the human arms may not have the same morphology as robot arms, our framework learns unsupervised human to robot translation to overcome the morphology mismatch issue. Second, to capture the details in salient regions that are crucial for learning state representations, our model performs unsupervised keypoint detection on the translated robot videos. The detected keypoints form a structured representation that contains semantically meaningful information and can be used directly for computing reward and policy learning. We evaluate the effectiveness of our LbW framework on five robot manipulation tasks, including reaching, pushing, sliding, coffee making, and drawer closing. More results and analysis are available at `pair.toronto.edu/lbw-kp/`.**

I. Introduction and Related Work

Robotic *Imitation Learning*, also known as *Learning from Demonstration* (LfD), allows robots to acquire manipulation skills performed by expert demonstrations through learning algorithms [13, 1]. While progress has been made by existing methods, collecting expert demonstrations remains expensive and challenging as it assumes access to both observations and actions via kinesthetic teaching [13, 1], teleoperation [29, 4], or crowdsourcing platform [9, 10, 11, 12]. In contrast, humans have the ability to imitate manipulation skills by *watching* third-person performances. Motivated by this, recent methods resort to endowing robots with the ability to learn manipulation skills via physical imitation from human videos [8, 21, 24, 20, 19, 14, 18, 27, 15, 23, 22, 17, 16].

Unlike conventional LfD methods [13, 1, 29, 4], approaches based on imitation from human videos relax the dependencies, requiring *only* human videos as supervision [8, 24, 21]. One of the main challenges of these imitation learning methods is how to minimize the domain gap between humans and robots. To overcome the morphology mismatch issue, existing imitation learning methods [8, 24, 21] typically leverage image-to-image translation models (e.g., CycleGAN [30]) to translate videos from the human domain to the robot domain. However, vanilla image-to-image translation models often capture only the macro features at the expense of neglecting the details in salient regions that are crucial for downstream tasks [3]. Deriving state representations by encoding the translated image observations using a feature encoder would lead to suboptimal performance.

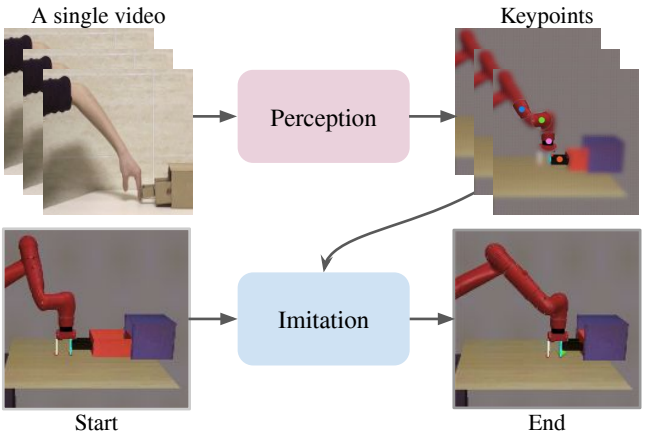

Fig. 1. **LbW.** Given a single human video, our LbW framework learns human to robot translation followed by unsupervised keypoint detection. The resulting keypoint-based representations are semantically meaningful and can be used to guide the robot to learn manipulation skills through physical imitation.

In this paper, we present Learning by Watching (LbW), a framework for physical imitation from human videos for learning robot manipulation skills. As shown in Figure 1, our framework is composed of a perception module and a policy learning module for physical imitation. The perception module aims at minimizing the domain gap between the human domain and the robot domain as well as capturing the details of salient regions by keypoint-based representations. To learn manipulation skills, we cast this as a *reinforcement learning* (RL) problem, where we aim to enable the robot to perform physically viable learning with the objective to imitate similar behavior as demonstrated in the translated robot video under context-specific constraints. We evaluate the effectiveness of our LbW framework on five robot manipulation tasks, including reaching, pushing, sliding, coffee making, and drawer closing in two simulation environments (i.e., the Fetch-Robot manipulation in OpenAI gym [2] and meta-world [28]).

II. Preliminaries

A. Unsupervised Image-to-Image Translation

We adopt MUNIT [6] to learn a model that translates images from a source domain X to a target domain Y *without* paired training data. MUNIT assumes that an image representation can

be disentangled into a domain-invariant content code (encoded by a content encoder E^c) and a domain-specific style code (encoded by a style encoder E^s). The content encoders E^c_X and E^c_Y are shared in the two domains, whereas the style encoders E^s_X and E^s_Y of the two domains do *not* share weights. To translate an image from one domain to the other, we combine its content code with a style code sampled from the other domain. We define the adversarial loss $\mathcal{L}^x_{\text{GAN}}$ in the source domain as

$$\mathcal{L}^x_{\text{GAN}} = \mathbb{E}\Big[\log D_X(x) + \log\Big(1 - D_X\big(G_X(c_y, s_x)\big)\Big)\Big], \quad (1)$$

The adversarial loss $\mathcal{L}^y_{\text{GAN}}$ in the target domain can be similarly defined.

In addition to the adversarial losses, MUNIT applies reconstruction losses on images and content and style codes to regularize the model learning. For the source domain, the image reconstruction loss $\mathcal{L}^x_{\text{rec}}$ is defined as

$$\mathcal{L}^x_{\text{rec}} = \mathbb{E}\Big[\big\|G_X(c_x, s_x) - x\big\|\Big], \quad (2)$$

the content reconstruction loss $\mathcal{L}^{c_x}_{\text{rec}}$ is defined as

$$\mathcal{L}^{c_x}_{\text{rec}} = \mathbb{E}\Big[\big\|E^c_Y\big(G_Y(c_x, s_y)\big) - c_x\big\|\Big], \quad (3)$$

and the style reconstruction loss $\mathcal{L}^{s_x}_{\text{rec}}$ is defined as

$$\mathcal{L}^{s_x}_{\text{rec}} = \mathbb{E}\Big[\big\|E^s_X\big(G_X(c_y, s_x)\big) - s_x\big\|\Big]. \quad (4)$$

The image reconstruction loss $\mathcal{L}^y_{\text{rec}}$, the content reconstruction loss $\mathcal{L}^{c_y}_{\text{rec}}$, and the style reconstruction loss $\mathcal{L}^{s_y}_{\text{rec}}$ in the target domain can be derived similarly.

The total loss $\mathcal{L}_{\text{MUNIT}}$ for training MUNIT is given by

$$\begin{aligned}\mathcal{L}_{\text{MUNIT}} = \mathcal{L}^x_{\text{GAN}} + \mathcal{L}^y_{\text{GAN}} + \lambda_{\text{image}}(\mathcal{L}^x_{\text{rec}} + \mathcal{L}^y_{\text{rec}}) \\ + \lambda_{\text{content}}(\mathcal{L}^{c_x}_{\text{rec}} + \mathcal{L}^{c_y}_{\text{rec}}) + \lambda_{\text{style}}(\mathcal{L}^{s_x}_{\text{rec}} + \mathcal{L}^{s_y}_{\text{rec}}),\end{aligned} \quad (5)$$

where λ_{image}, λ_{content}, and λ_{style} are hyperparameters used to control the relative importance of the respective loss functions.

B. Unsupervised Keypoint Detection

We leverage Transporter [7] to detect the keypoints in each translated video frame. Transporter leverages object motion between a pair of video frames to transform a video frame into the other by transporting features at the detected keypoint locations. Given two video frames x and y, Transporter first extracts feature maps $\Phi(x)$ and $\Phi(y)$ for both video frames using a feature encoder Φ and detects K 2-dimensional keypoint locations $\Psi(x)$ and $\Psi(y)$ for both video frames using a keypoint detector Ψ. Transporter then synthesizes the feature map $\hat{\Phi}(x, y)$ by suppressing the feature map of x around each keypoint location in $\Psi(x)$ and $\Psi(y)$ and incorporating the feature map of y around each keypoint location in $\Psi(y)$:

$$\hat{\Phi}(x, y) = (1 - \mathcal{H}_{\Psi(x)}) \cdot (1 - \mathcal{H}_{\Psi(y)}) \cdot \Phi(x) + \mathcal{H}_{\Psi(y)} \cdot \Phi(y), \quad (6)$$

Next, the transported feature $\hat{\Phi}(x, y)$ is passed to a refinement network R to reconstruct to the video frame y. We define the loss $\mathcal{L}_{\text{transporter}}$ for training Transporter as

$$\mathcal{L}_{\text{transporter}} = \mathbb{E}\Big[\big\|R\big(\hat{\Phi}(x, y)\big) - y\big\|\Big]. \quad (7)$$

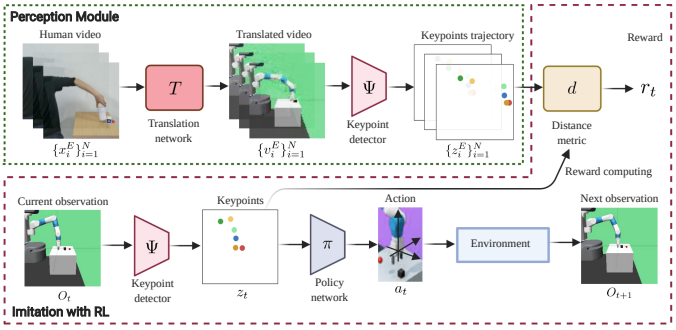

Fig. 2. **Overview of the proposed LbW.** Our LbW framework is composed of three main components: an image-to-image translation network T, a keypoint detector Ψ, and a policy network π.

III. Proposed Method

A. Algorithmic Overview

We consider the task of physical imitation from human videos for learning robot manipulation skills. In this setting, we assume we have access to a *single* human video $V_X = \{x^E_i\}^N_{i=1}$ of length N depicting a human performing a specific task (e.g., pushing a block) that we want the robot to learn from, where $x^E_i \in \mathbb{R}^{H \times W \times 3}$ and $H \times W$ is the spatial size of x^E_i. Our goal is to develop a learning algorithm that allows the robot to *imitate* the behavior demonstrated by the human in the human video V_X.

As shown in Figure 2, given a human video V_X and the current observation $O_t \in \mathbb{R}^{H \times W \times 3}$ at time t, we first apply the image-to-image translation network T to each frame x^E_i in the human video V_X and translate x^E_i to a robot demonstration video frame $v^E_i \in \mathbb{R}^{H \times W \times 3}$. Next, the keypoint detector Ψ takes each translated robot demonstration video frame v^E_i as input and extracts the keypoint-based representation $z^E_i = \Psi(v^E_i) \in \mathbb{R}^{K \times 2}$, where K denotes the number of keypoints. Similarly, we also apply the keypoint detector Ψ to the current observation O_t to extract the keypoint-based representation $z_t = \Psi(O_t) \in \mathbb{R}^{K \times 2}$. To compute the reward for physical imitation, we define a distance metric d that computes the distances between the keypoint-based representation z_t of the current observation O_t and each of the keypoint-based representations z^E_i of the translated robot demonstration video frames v^E_i. Finally, the policy network π takes as input the keypoint-based representation z_t of the current observation O_t to predict an action $a_t = \pi(z_t)$ that is used to guide the robot to interact with the environment. The details of each component are described in the following subsections.

B. Unsupervised Domain Transfer with Keypoints

To achieve physical imitation from human videos, we develop a perception module that consists of a MUNIT model and a Transporter model as shown in Figure 3. To train the MUNIT model, we first collect the training data for the source domain (i.e., human domain) and the target domain (i.e., robot domain). The source domain contains the human video V_X

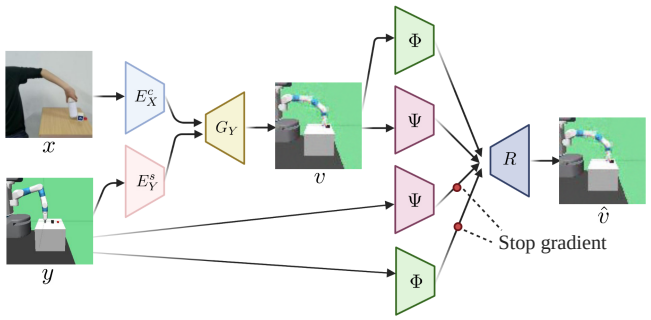

Fig. 3. **Overview of the perception module.** Our perception module is composed of a MUNIT network (left) and a Transporter model (right).

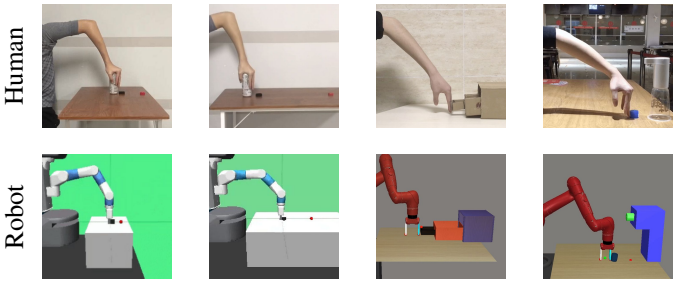

Fig. 4. **Task overview.** We present the sample task scenes and one sample human video frame for the pushing, sliding, drawer closing, and coffee making tasks, respectively.

that we want the robot to learn from. To increase the diversity of the training data in the source domain for facilitating the MUNIT model training, we collect a few *random* data by asking the human to randomly move the hands above the table *without* performing the task. As for the target domain training data, we collect a number of robot videos generated by having the robot execute actions that are randomly sampled from the action space. As such, the collection of the robot videos does *not* require human expertise and effort.

Using the training data from both source and target domains, we are able to train the MUNIT model to achieve human to robot translation using the total loss $\mathcal{L}_{\text{MUNIT}}$ in (5) and following the protocol described in Section II-A. After training the MUNIT model, we are able to translate the human video $V_X = \{x_i^E\}_{i=1}^N$ frame by frame to the robot video $\{v_i^E\}_{i=1}^N$ by combining the content code of each human video frame and a style code randomly sampled from the robot domain.

As mentioned in Section II-B, we aim to learn keypoint-based representations from the translated robot demonstration video in an unsupervised fashion. As illustrated in Figure 3, the Transporter model takes a translated robot demonstration video frame v and a robot video frame y from robot videos collected by a random policy as input, then, extracts their features and detects keypoint locations, respectively. The transporter model then reconstructs the translated robot demonstration video frame. To train the Transporter model, we optimize the total loss $\mathcal{L}_{\text{transporter}}$ in (7). Once the training of the Transporter model converges, we are able to use the keypoint detector Ψ of the Transporter model to extract a keypoint-based representation $z_i^E = \Psi(v_i^E)$ for each frame v_i^E in the translated robot demonstration video to form a keypoints trajectory $\{z_i^E\}_{i=1}^N$ and a keypoint-based representation $z_t = \Psi(O_t)$ for the current observation O_t. We then use both of them to compute the reward r_t and use the keypoint-based representation z_t of the current observation O_t to predict an action a_t.

C. Physical Imitation with RL

We use RL to learn a policy from image-based observations. In our method, we *decouple* the policy learning phase from the keypoint-based representation learning phase. To achieve physical imitation, we aim to minimize the distance between the keypoints trajectory of the agent and that of the translated robot demonstration video. Specifically, we define the reward r_t as

$$r_t = d\big(z_t, z_{t+1}, \{z_i^E\}_{i=1}^N\big) = \lambda_{r_1} \cdot r_1(t) + \lambda_{r_2} \cdot r_2(t), \quad (8)$$

where λ_{r_1} and λ_{r_2} are hyperparameters that balance the importance between the two terms, and the aforementioned goal is imposed on $r_1(t)$ and $r_2(t)$, which are defined by the following equations:

$$r_1(t) = -\min \|z_t - z_m^E\|, \text{ and} \quad (9)$$

$$r_2(t) = -\min \Big(\big\| (z_{t+1} - z_t) - (z_{m+1}^E - z_m^E) \big\| \Big), \quad (10)$$

where $1 \leq m \leq N-1$, $r_1(t)$ aims to minimize the distance between the keypoint-based representation z_t of the current observation O_t and the most similar (closest) keypoint-based representation z_m^E in the keypoints trajectory $\{z_i^E\}_{i=1}^N$ of the translated robot demonstration video $\{v_i^E\}_{i=1}^N$, and $r_2(t)$ is the first-order difference equation of $r_1(t)$.

The policy network π can be trained with any RL algorithms in principle.

IV. Experiments

Through experiments, we aim to investigate the following questions: (1) How accurate is our perception module in handling the human-robot domain gap and in detecting keypoints? (2) How does LbW compare with state-of-the-art baselines in terms of performance on robot manipulation tasks?

A. Experimental Setting

We perform experimental evaluations in two simulation environments, i.e., the Fetch-Robot manipulation in OpenAI gym [2] and meta-world [28]. We evaluate on five tasks: reaching, pushing, sliding, coffee making, and drawer closing. Figure 4 presents the overview of each task, including the task scenes and one sample human video frame for each task.

In the policy learning phase, the robot receives only an RGB image of size $84 \times 84 \times 3$ as the observation. The robot arm is controlled by an Operational Space Controller in end-effector positions As each of the tasks is described by a single human video, we set the initial locations of the object and the target to a fixed configuration.

B. Comparison to Baseline Methods

To evaluate the effectiveness of our perception module, we implement two baseline methods using the same control model as LbW, which is adopted from SAC+AE [26], but with different reward learning methods.

Classifier-reward. We implement a classifier-based reward learning method in a similar way as VICE [5]. For each task, given robot demonstration videos, instead of the human videos, the CNN classifier is pre-trained on goal images with *positive labels* and the remaining images with *negative labels*.

AVID-m. Since AVID [24] is the state-of-the-art method that outperforms prior approaches, including BCO [25] and TCN [19], we focus on comparing our method with AVID. For a fair comparison, we reproduce the reward learning method of AVID and replace the control module with SAC+AE[26]. We denote this method as AVID-m. For each task, given human demonstration videos, we first translate the human demonstration videos to the robot domain using the CycleGAN [30] model. Then the CNN classifier is pre-trained on the *translated* goal images with *positive labels* and the remaining translated images with *negative labels*.

C. Dataset Collection and Statistics

We decouple the training phase of the perception module from that of the policy learning module.

Dataset for perception module training. To train our perception module and the CycleGAN method, we collect human expert videos and videos of a human performing random actions without performing the tasks for the human domain. Note that we do *not* use robot expert videos for training the perception module. For the human domain, we use 1056, 398, 650, 986, 658 images for Reaching, Pushing, Sliding, Drawer Closing, and Coffee Making respectively. For the robot domain, we use 3150, 1220, 2120, 2940, 4007 for the same five tasks respectively.

Dataset for policy learning. For policy learning, we use only one single human expert video to train our policy network. The AVID-m method uses 15 human expert videos, while the classifier-reward approach uses 35 robot expert videos.

D. Performance Evaluations

Following AVID [24], we use *success rate* as the evaluation metric. At test time, the task is considered to be a success if the robot is able to complete the task within a specified number of time steps (i.e., 50 time steps for reaching and pushing, and 300 time steps for sliding, coffee making, and drawer closing). The results are evaluated by 10 test episodes for each task.

Table I reports the success rates of our method and the two baseline approaches on all five tasks. We find that for the sliding, drawer closing, and coffee making tasks, our LbW performs favorably against the two competing approaches.

The difference between the AVID-m method and the classifier-reward approach is that AVID-m leverages CycleGAN for human to robot translation, while the classifier-reward method using ground-truth robot images directly. As shown in Figure 5, the translated images of AVID-m have clear visual artifacts. For instance, the red cube disappears and the robot poses in the translated images do not match those in the human video frames. The comparisons between AVID-m and the classifier-reward method and the visual results of AVID-m in Figure 5 show that using image-to-image translation models alone for minimizing the human-robot domain gap will have negative impact to the performance of the downstream tasks.

Our perception module learns unsupervised human to robot translation as well as unsupervised keypoint detection on the translated robot videos. The learned keypoint-based representation provides semantically meaningful information for the robot, allowing our LbW framework compares favorably two competing approaches.

TABLE I

SUCCESS RATES. COMPARISON OF SUCCESS RATES FOR TEST EVALUATIONS OF OUR LBW FRAMEWORK AND THE BASELINES.

Method	Videos	Reaching	Pushing	Sliding	Drawer closing	Coffee making
Classifier reward	35 robot videos	**100**%	**100**%	30%	70%	50%
AVID-m	15 human videos	**100**%	60%	0%	50%	40%
LbW (Ours)	**1** human video	**100**%	**100**%	**80**%	**80**%	**70**%

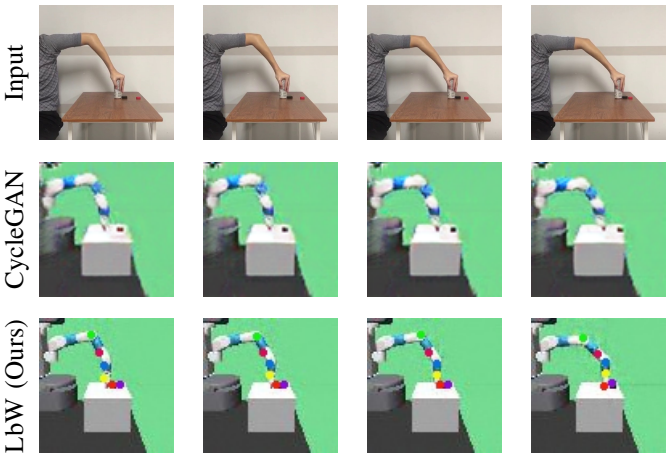

Fig. 5. **Visual results and comparisons on the pushing task.** Given a human video as input in the first row, we present the translated images of CycleGAN [30] in the second row. In the third row, we visualize our translated images and the detected keypoints produced by the perception module. Our perception module accurately detects the robot arm pose and the location of the interacting object.

V. CONCLUSIONS

We introduced LbW, a framework for physical imitation from human videos. Our core technical novelty lies in the design of the perception module that 1) minimizes the human-robot domain gap, 2) learns a semantically meaningful keypoint-based representation in an unsupervised manner. Extensive experimental results on five robot manipulation tasks demonstrate the effectiveness of our approach and the advantage of learning keypoint-based representations over conventional state representation learning approaches. More results, videos, and performance comparisons are available at `pair.toronto.edu/lbw-kp/`.

References

[1] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *HRI*, 2012.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

[3] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *ICCV*, 2019.

[4] Sylvain Calinon, Paul Evrard, Elena Gribovskaya, Aude Billard, and Abderrahmane Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. In *ICAR*, 2009.

[5] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *NeurIPS*, 2018.

[6] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[7] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. In *NeurIPS*, 2019.

[8] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *ICRA*, 2018.

[9] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, A. Tung, J. Gao, John Emmons, Anchit Gupta, Emre Orbay, S. Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *CoRL*, 2018.

[10] Ajay Mandlekar, Fabio Ramos, B. Boots, Li Fei-Fei, Animesh Garg, and D. Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *ICRA*, 2020.

[11] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, S. Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv*, 2020.

[12] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and S. Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv*, 2020.

[13] Peter Pastor, Ludovic Righetti, Mrinal Kalakrishnan, and Stefan Schaal. Online movement adaptation based on previous sensor experiences. In *IROS*, 2011.

[14] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In *ICLR*, 2018.

[15] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *TOG*, 2018.

[16] Vladimir Petrik, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Learning Object Manipulation Skills via Approximate State Estimation from Real Videos. In *CoRL*, 2020.

[17] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. In *CoRL*, 2020.

[18] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv*, 2016.

[19] 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.

[20] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *RSS*, 2020.

[21] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. In *NeurIPS*, 2019.

[22] Maximilian Sieb and Katerina Fragkiadaki. Data dreaming for object detection: Learning object-centric state representations for visual imitation. In *Humanoids*, 2018.

[23] Maximilian Sieb, Zhou Xian, Audrey Huang, Oliver Kroemer, and Katerina Fragkiadaki. Graph-structured visual imitation. In *CoRL*, 2020.

[24] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. In *RSS*, 2020.

[25] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv*, 2018.

[26] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv*, 2019.

[27] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv*, 2018.

[28] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019.

[29] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*, 2018.

[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.