Learning from Imperfect Demonstrations from Agents with Varying Dynamics

Zhangjie Cao[1] and Dorsa Sadigh[2]

***Abstract*—Imitation learning enables robots to learn from demonstrations. Previous imitation learning algorithms usually assume access to optimal expert demonstrations. However, in many real-world applications, this assumption is limiting. Most collected demonstrations are not optimal or are produced by an agent with slightly different dynamics. We therefore address the problem of imitation learning when the demonstrations can be sub-optimal or be drawn from agents with varying dynamics. We develop a metric composed of a feasibility score and an optimality score to measure how useful a demonstration is for imitation learning. The proposed score enables learning from more informative demonstrations, and disregarding the less relevant demonstrations. Our experiments on four environments in simulation and on a real robot show improved learned policies with higher expected return.**

***Index Terms*—Imitation Learning, Learning from Demonstrations, Robot Learning**

I. Introduction

Today's imitation learning algorithms often assume access to expert demonstrations [1]–[4]. However, this assumption is limiting in many real-world environments. In practice, it is expensive to obtain large number of expert demonstrations, but we usually have access to a plethora of imperfect demonstrations — demonstrations that can range from random noise or failures to expert or even optimal demonstrations, and demonstrations that are collected from agents with different dynamics, such as different embodiments, body schema, or joint or rigid body structures. This is even evident when collecting human demonstrations on a robot as we often have access to more suboptimal data from humans due to factors such as their bounded rationality [5] or difficulty of controlling robots with different or high degrees of freedom [6].

Prior work has studied learning from suboptimal demonstrations by assigning an optimality score for each demonstration, and only relying on learning from the optimal data [7]–[10]. In this paper, we go beyond this assumption and consider imperfect demonstrations that can also be collected from agents with different dynamics, e.g., demonstrations on a robot with different number of joints than our target agent. In this setting, simply considering optimality can fail because some demonstrations assessed as optimal might not even be feasible on the target robot. Our key insight is that we need to measure *optimality* along with *feasibility* of demonstrated trajectories to effectively learn policies for a target domain. This further requires us to go beyond existing optimality measures, as an optimal demonstration in another dynamics may not be optimal for our target agent.

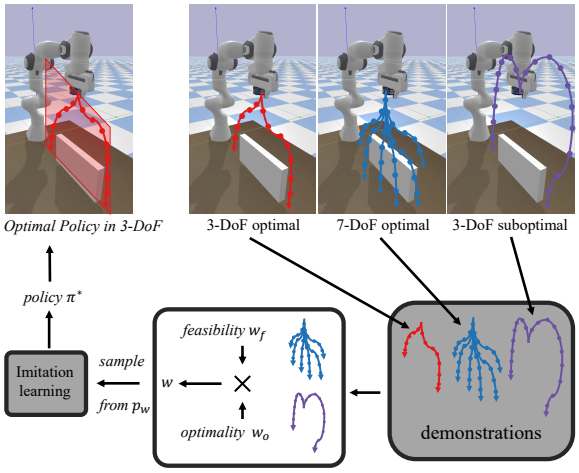


Fig. 1. Learning from Imperfect Demonstrations from Agents with Varying Dynamics. The target agent is a 3-DoF robot arm, and the demonstrations can be: optimal demonstrations in the same dynamics (*3-DoF optimal*), optimal demonstrations from another dynamics (*7-DoF optimal*), or suboptimal demonstrations in the same dynamics (*3-DoF suboptimal*). We develop a score w derived from a feasibility w_f and an optimality w_o to define a distribution p_w over state transitions in demonstrations. p_w assigns higher probability to desirable demonstrations to enable learning an optimal policy.

We introduce an algorithm for learning from agents with varying dynamics. Our algorithm leverages a new metric that combines a feasibility score and an optimality score to assess the informativeness of the demonstrations. The feasibility score measures the likelihood of a trajectory to be achieved by a target agent. The optimality score measures optimality under different dynamics as opposed to prior work that only assess optimality based on the demonstrator's dynamics. Leveraging our new metric, we can learn from demonstrations that are both more likely to be feasible and have higher expected return. The proposed score only changes how we sample from the demonstrations, and our algorithm is agnostic to the choice of imitation learning algorithm used after scoring the demonstrations.

The main contribution of the paper can be summarized as:

- We propose a more realistic problem setting of learning from imperfect demonstrations, where the demonstrations may not be optimal and can be drawn from agents with different dynamics.

Manuscript received: October 15, 2020; Revised: January 26, 2021; Accepted: February 23, 2021.

This paper was recommended for publication by Editor Dana Kulic upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by NSF Award Number 1849952 and the FLI grant RFP2-000 project.

[1]caozj@cs.stanford.edu, and [2]dorsa@stanford.edu. The authors are with the Department of Computer Science, Stanford University, Stanford, CA 94305, USA

Digital Object Identifier (DOI): see top of this page.

- We propose a scoring function composed of a feasibility score and an optimality score to measure how informative each imperfect demonstration is. The feasibility score measures how likely it is for a state transition to be achieved by the target agent, while the optimality score measures how optimal a state transition is with respect to the expert behavior according to its expected return.
- We conduct experiments in four simulation environments and an obstacle avoidance task on a robot. We show that our algorithm outperforms other methods when learning from imperfect demonstrations with varying dynamics.

II. Related work

Imitation Learning. The goal of imitation learning is to find a policy that best imitates expert demonstrations. Behavior cloning (BC) directly learns the policy from a sequence of state-action pairs [1]. However, BC can result in compounding errors that is usually addressed by dataset aggregation [11] or policy aggregation [12], [13]. On the other hand, Inverse reinforcement learning (IRL) aims to first recover the reward function from the demonstrations and then find a policy through reinforcement learning [2], [14]–[16]. Generative Adversarial Imitation Learning (GAIL) was proposed for learning the expert behavior by matching the occupancy measure between the policy and the demonstrations [3]. GAIL has been extended to improve the divergence measurement [17], [18], but these extensions rely on access to a sequence of states and actions. When only state observations are available, imitation learning first recovers the actions [19], or directly matches the state distribution [20]–[22].

Learning from Demonstrations with Different dynamics. The above imitation learning works assume the demonstrator shares the same MDP as the target agent. However, in reality, the demonstrator may have a different dynamics than the target agent, which introduces the correspondence problem between the demonstrator and the target agent [23], [24]. Englert *et al.* address the correspondence problem by aligning the state trajectory distributions of the expert demonstrations and of the policy [25]. Calinon *et al.* model the demonstrations as a Gaussian mixture model within a projected lower-dimensional subspace, and reproduce the optimal behavior by optimizing a time-dependent similarity measurement for different robot contexts [26]. Eppner *et al.* learn a task description as the relation of objects and robot body parts using a dynamic Bayesian network and reproduce the joint action by maximizing the likelihood [27]. Liu *et al.* aim to address the different dynamics for imitation learning by matching the state sequence distribution to learn a state policy and recover the actions using inverse dynamics [28]. These works need a correspondence between the demonstrations and the target agent, but under different dynamics, there could exist situations that the demonstrations are not even achievable by the target agent, and the correspondence may not exist. Our feasibility metric is designed to address exactly this problem. Also, our work unlike prior works does not assume access to optimal demonstrations.

Learning from Imperfect Demonstrations. Previous learning from demonstration works usually assume that the demonstrations are provided by an optimal agent who is successful at achieving the task [3], [11], [14], [16], [29]. In practice, most demonstrations may be sub-optimal or even failures due to reasons, such as the difficulty of providing expert data on robots with high degrees of freedom, or noisy demonstrations due to bounded rationality of humans [6], [30]–[32]. Recently, several works have addressed the problem of learning from imperfect demonstrations including learning a confidence weight to model the optimality of state-action pairs [7], or modeling the probability of optimal demonstrations [10]. Zhu *et al.* utilize a sparse reward function to measure the optimality of each demonstration [9]. Grollman *et al.* assume access to only low-quality demonstrations, and treat the data as negative examples [8]. All of these works assume the same MDP between the provided demonstrations and the target agent. This results in simplified optimality metrics based on the demonstrator's dynamics. However, when the dynamics of a demonstrator and the target agent are different, the previous optimality measurements are not applicable anymore. Here, we introduce a new optimality measure combined with a feasibility metric to tackle the problem of learning from imperfect demonstrations from agents with varying dynamics.

III. Problem Statement

We consider the standard Markov decision process (MDP): $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, \mathcal{R}, \rho_0, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability, ρ_0 is the initial state distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor.

A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ defines a probability distribution over the space of actions in a given state. An optimal policy π^* maximizes the expected return $\eta_\pi = \mathbb{E}_{\xi \sim \pi}[G_t(\xi)] = \mathbb{E}_{s_0 \sim \rho_0, \pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})\right]$, where t indicates the time step.

We formalize the problem of learning the optimal policy π^* for the MDP $\mathcal{M}$ as an imitation learning problem: Given a set of demonstration trajectories $\Xi = \{\xi_1, \ldots, \xi_D\}$, where each trajectory is a sequence of state-action pairs $\xi = \{s_0, a_1, s_1, a_2, \ldots, a_N, s_N\}$, we aim to learn a policy π^* that best matches the demonstration trajectories. Note that we focus on the offline imitation setting as employed in [3], [7], where a set of demonstrations are provided ahead of time instead of gradually incrementing our dataset as in online imitation learning [33].

One of the core assumptions in prior works in imitation learning is that the demonstrations are drawn from the expert policy of the MDP $\mathcal{M}$, which is referred as the *target* MDP [3], [16], [34]. Here, we are interested in a variant of the imitation learning problem, where we do not make the strong assumption of having access to *expert* demonstrations. Instead, we consider a setting, where the demonstrations can be drawn from either a *sub-optimal policy* or from a *different MDP,* $\mathcal{M}^d$*,* that has a different dynamics from the target MDP $\mathcal{M}$. We note that the transition probability of an MDP is affected by changes both in the environment and in the agent's dynamics. The *demonstrator* MDP, $\mathcal{M}^d$, differs from the *target* MDP, $\mathcal{M}$, only in the agent's dynamics. For example, in a driving environment, we might have demonstrations from a vehicle with different vehicle

dynamics that drive in the same environment. Unlike prior imitation learning work, we can only rely on state-transitions as opposed to state-action sequences since the two MDPs might not share the same transition probabilities, and some actions in the demonstrator's MDP might not even be feasible in the target MDP. We thus assume the reward is only a function of state transitions, i.e., $\mathcal{R} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$, and instead of imitating the state-action trajectories, *we discard actions in the demonstration trajectories and let the agent imitate the state transitions*, i.e., (s_t, s_{t+1}) pairs.

Challenges. The core challenge in learning from imperfect demonstrations from agents with varying dynamics is assessing how beneficial a given trajectory is for learning a policy for the target MDP. This requires measuring two main characteristics of a given trajectory: 1) whether the trajectory is achievable in the target MDP (*feasibility*), and 2) the expected return of the trajectory in the target MDP (*optimality*).

We argue that learning a policy can only benefit from a trajectory if and only if neither the feasibility nor the optimality are low for that given trajectory. If a trajectory has low feasiblity, it is not achievable by the target agent and thus the target agent cannot imitate the behavior. Similarly, if a trajectory has low optimality, imitating it can induce a sub-optimal policy leading to low expected return.

IV. Learning from Imperfect Demonstrations from Agents with Varying Dynamics

In this section, we first introduce our method that incorporates a scoring mechanism for assessing feasibility and optimality, and then discuss our algorithm.

A. Feasibility

Feasibility measures how likely it is for each state transition in the demonstration to be generated by the target MDP. Previous learning from imperfect demonstration works do not consider a measure of feasibility since they often assume the demonstrator follows the same dynamics as the target MDP [7], [10]. Formally, a state transition (s_t, s_{t+1}) is *feasible* if and only if there exists an action $a_t \in \mathcal{A}$ in the target MDP $\mathcal{M}$ satisfying $p(s_t, a_t, s_{t+1}) > 0$.

However, we cannot simply apply this definition to our demonstrations to verify feasibility. First, the transition probability function is usually unknown or difficult to learn in most realistic robotics tasks, and thus we cannot directly use the function to check for the feasible actions. If we do not use the transition probability function, but assume access to a simulator, where we can sample the next state given a state and action – as is assumed by prior works [3], [7], [16], [34] – we would still need to search over a large space of possible actions to verify existence of an action a_t for the state transition (s_t, s_{t+1}) to be successful in the target MDP.

We instead propose computing the most probable action using an inverse dynamics model of the target MDP $f_{\text{id}} : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{A}$, which takes a state transition pair (s_t, s_{t+1}) that could be generated by either the target MDP or the demonstrator MDP as an input and outputs all the actions a_t that satisfy $p(s_t, a_t, s_{t+1}) > 0$. For every ξ from the set of demonstrations provided by the demonstrator MDP, we initialize the agent at the state s_0, which is the initial state of ξ. We then make the agent take an action a'_t generated by f_{id} at each time step to generate a new state trajectory $\xi' = \{s'_0, s'_1, s'_2 \ldots, s'_N\}$ that is achievable by the target MDP:

$$\begin{aligned} &s'_0 = s_0, \quad a'_t = f_{\text{id}}(s'_{t-1}, s_t), t \geq 1 \\ &s'_t \sim p(s'_{t-1}, a'_t, s'_t), t \geq 1 \end{aligned} \tag{1}$$

Learning Inverse Dynamics. If we assume access to an accurate inverse dynamics model f_{id} – meaning that f_{id} gives the correct action for a feasible state transition and outputs "None" for an infeasible state transition, we can detect the infeasible trajectories based on the "None" output. However, such a binary feasibility metric cannot differentiate between nearly feasible and totally infeasible trajectories, where the former is useful for learning and the latter does not have much learning value. So instead we aim to learn a f_{id} that outputs the correct action if the state transition is feasible while still outputting the closest action, when the state transition is infeasible. We model f_{id} as a neural network, and train it on a randomly sampled set of trajectories $\Xi_f = \{\xi_f\}$ from the target MDP. We emphasize that similar to prior imitation learning works [3], [7], [16], [34], we assume access to a simulator for the target MDP, where we can sample feasible but random trajectories. We learn f_{id} using a regression loss:

$$\min_{f_{\text{id}}} \mathbb{E}_{(s_t, a, s_{t+1}) \sim \xi_f, \xi_f \sim \Xi_f} L_{1;\text{smooth}}(f_{\text{id}}(s_t, s_{t+1}), a). \tag{2}$$

Computing the Feasibility Score. Here, the trained f_{id} will output similar actions if the input state transition is similar to a state transition seen in the training data. With the learned f_{id}, for every trajectory ξ – even if it is infeasible – we can compute a corresponding trajectory ξ'. In addition, we can compute a distance metric between the two trajectories ξ and ξ', $F(\xi, \xi')$, where lower distances correspond to higher feasibility scores. We compute the feasibility score by normalizing this distance within a range $[d_{\text{min}}, d_{\text{max}}]$:

$$w_f(\xi) = \begin{cases} 1 & F(\xi, \xi') < d_{\text{min}} \\ 1 - \frac{F(\xi, \xi') - d_{\text{min}}}{d_{\text{max}} - d_{\text{min}}} & d_{\text{min}} \leq F(\xi, \xi') \leq d_{\text{max}} \\ 0 & F(\xi, \xi') > d_{\text{max}} \end{cases} \tag{3}$$

Here, ξ' is a corresponding trajectory to ξ generated by the inverse dynamics f_{id}. Further, we can choose $F(\xi, \xi')$ to be any sequence metric between the two trajectories. In our experiments, we computed the mean of the l_2 distance between each pair of states in ξ and ξ'. The proposed continuous feasibility score can measure different degrees of feasibility and preserve nearly feasible trajectories that can be useful for learning a policy for the target agent.

Discussion of Feasibility Score. Using our feasibility measurement, the more infeasible transitions contained in ξ are, the deviation between ξ' and ξ becomes larger, and hence it would be less likely that the trajectory is drawn from the target MDP. A possible drawback of our feasibility score is compounding errors from our inverse dynamics model f_{id}. To address this problem, we normalize the distance between the trajectories $F(\xi, \xi')$ using d_{min} and d_{max}. Here, d_{min} models the

lowest compounding error that could be achieved by a feasible trajectory, while $d_{\max}$ is the highest compounding error. Thus, our feasibility measure can assign non-zero feasibility to any feasible trajectories even with compounding errors from the inverse dynamics. We include the method to set the thresholds in Section II of the supplementary materials.

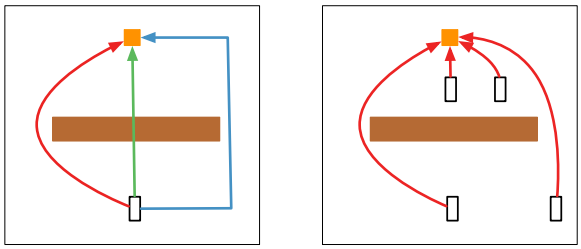
B. Optimality

We would like to define an optimality score that not only measures the instance reward received by the state transition but also the influence of a given state transition on the future state transitions, e.g., whether the given state transition leads to higher expected return trajectories. Previous works evaluate optimality based on pre-defined measurements provided by the demonstrator, such as confidence score [7] and probability of taking an optimal action [10]. However, these scores are evaluated based on the demonstrator's dynamics.

Here, the demonstrator follows a different dynamics as the target agent, and thus the set of state transitions achievable by the target agent may be different from those achievable by the demonstrator. For example, consider a navigation setting in Fig. 2(a), where we allow three possible dynamics: (1) unlimited turns, (2) no turns are allowed, and (3) can only turn in $\frac{\pi}{2}$ increments. We set the reward as driving to the orange goal as fast as possible without colliding with the brown obstacle. Here, the red, green and blue trajectories are the optimal trajectories achieved by the dynamics of *unlimited turns*, *no turns allowed*, and *only turning in $\frac{\pi}{2}$ increments* respectively. Note that the trajectory with *unlimited turns* receives higher expected return than the other trajectories. So a state transition that is optimal under one dynamics may be suboptimal or even useless for another agent. Thus, we need an optimality measurement that can handle the more general case of learning from demonstrations with varying dynamics.

We compute the expected return received by each state trajectory $\xi = \{s_1, s_2, \ldots, s_N\}$: $\eta_\xi = \sum_t \gamma^t \mathcal{R}(s_t, s_{t+1})$ as an estimate for how optimal a trajectory is. Since the reward function only depends on state transitions, the expected return of a trajectory can be naturally generalized across different dynamics. We emphasize that we do not assume access to the reward function, and only assume access to the expected return of a trajectory similar to prior work that leverage a given confidence measure for each demonstration [7].

Effects of Initial States on Optimality. The expected return is general but it still suffers from some problems. Is a trajectory with higher expected return really more useful for the policy learning of the target agent? If the trajectories are starting from the same initial state, this claim is correct, because higher expected return means we have a more optimal path from this initial state. However, when the trajectories start from different initial states, the claim may not be correct. Imagine the case shown in Fig. 2(b), where the car navigates to the goal from different initial states. If the car is close to the goal, e.g., one step from the goal, the navigation seems easier while if the car is far away from the goal, the navigation is more difficult. The optimal trajectory for the close initial location receives much higher reward than the optimal trajectory for the far one and thus is assigned with a much higher optimality if we only use the expected return as our measure of optimality. However, both trajectories (starting close or far) are important for the target agent since they give the agent guidance on how to navigate to the goal from different initial locations. In fact, we should assign similar optimalilty to both trajectories. Therefore, naïve optimality scores based on the expected return may negatively influence the diversity of the demonstrations.



(a) Different Dynamics (b) Different Initial States

Fig. 2. Fig. (a) shows optimal trajectories of navigation to the orange goal without colliding with the brown obstacle under different dynamics. The green trajectory is for a car that can never turn and only go straight. The blue trajectory is for a car that can only turn in $\frac{\pi}{2}$ increments. The red trajectory is for a car that can turn in any direction. We can see the optimal trajectories for different dynamics under the same reward can be different. Fig. (b) shows the optimal trajectories from different initial states with the same dynamics under the navigation task to reach the orange goal as fast as possible. So the car receives positive reward for reaching the goal but is punished with negative reward for every time step. The expected returns received by the optimal trajectories are different for different initial states.

The Optimality Score. To address this problem, we rectify the expected return to make it conditioned on the initial states. Our key idea is that if demonstration trajectories are from the same initial states, we should learn from the one with higher expected return. Based on this idea, we define a function $f_{\text{rec}} : \mathcal{S} \to \mathbb{R}$. The input is an initial state s_0 and the output is the highest expected return of a demonstration starting from s_0. We then define the optimality of a trajectory ξ as:

$$w_o(\xi) = \exp\left(-\frac{(\eta_\xi - f_{\text{rec}}(s_0))^2}{2\sigma^2}\right). \tag{4}$$

We compute the distance between the expected return of each trajectory $\xi = \{s_0, s_1, \ldots, s_N\}$ and the best expected return the demonstration can achieve by starting from s_0, and this can determine the optimality score assigned to a trajectory. However, the distance $\eta_\xi - f_{\text{rec}}(s_0)$ is smaller than 0 but the lower bound depends on the reward function, which may have highly different scales from the feasibility. So we normalize this score in the range of $[0, 1]$. We use a Gaussian function for normalization instead of the widely-used min-max normalization [35], which can more effectively filter out extremely low return trajectories.

Learning the Rectifying Function. The key to the designed optimality is estimating f_{rec} – the highest expected return for an initial state. Given that we only have access to a small number of trajectories, where the initial states and the expected returns are known, we need a set of assumptions for generating training data to learn f_{rec}. For example, if the initial states of two trajectories ξ_1 and ξ_2 are very close, the policy learned from one trajectory can also perform on the other and achieve

similar expected return. So we may assume that f_{rec} of the two close initial states should be the same or at least similar. A realistic assumption for f_{rec} is the neighborhood property: $\exists \delta > 0, \forall \xi = \{s_0, s_1, \ldots, s_N\}$, we set the highest expected return for s_0 as:

$$f_{\text{rec}}(s_0) = \max_{\xi' \in \Xi, |s'_0 - s_0| < \delta, w_f(\xi) > 0} \eta_{\xi'}, \tag{5}$$

where s'_0 is the initial state of ξ' and Ξ is the set of all demonstrations. We set the highest expected return for a initial state as the highest expected return of neighboring initial states of feasible trajectories ($w_f(\xi) > 0$), because the rectify function will be influenced by infeasible trajectories with high expected returns otherwise. Using this rectifying function f_{rec}, we can compute the rectified optimality, which induces more diverse optimal demonstrations.

C. Algorithm

Using the defined notions of feasibility and optimality, we now can compute the final score of how useful each state transition is for learning a policy for the target agent. We compute the final measurement as the product of the feasibility and optimality scores:

$$w(\xi) = w_f(\xi) \times w_o(\xi). \tag{6}$$

We then re-weigh each state transition in the demonstration by the score $w(\xi)$. The weighting-based work for learning from imperfect demonstrations often directly incorporate the weight in the imitation loss [7]. However, such re-weighing can suffer from numerical issues, when the weight of some state transitions are too large. Instead, for a more stable training, we define a distribution p_w over state transitions based on the score $w(\xi)$. For every state transition in a demonstration from the dataset, we assign a weight w_i (for the i-th state transition) equal to the weight assigned to the trajectory $w(\xi)$, i.e., $\forall i \quad w_i = w(\xi)$. We compute a probability distribution over the state transitions as $p_{w_i} = \frac{w_i}{\sum_i w_i}$. Using the sampling distribution p_w, we can embed our method into any imitation learning algorithm to enable it to learn from imperfect demonstrations from various dynamics.

Note that given a trajectory, we assign equal scores to each state-transition within the trajectory, which fails to emphasize the most important transitions. However, this is not a problem as the key transitions repeatedly appear in high score trajectories, and everytime they appear, their sampling probability p_w accumulates. This ensures the key transitions are considered. The detailed description of the algorithm is shown in Section I of the supplementary materials.

V. Experiment

We conduct experiments in five environments including three MuJoCo environments: Reacher, Swimmer, and Ant, one driving, and a Robot Arm environment, which are all kinematic without force interactions. However, our approach is not specific to kinematics, and can also be applied to settings with force interactions as long as we can interact with the environment to generate trajectories. We include the implementation details, parameter selection, and results on varied demonstration ratios in the supplementary materials.

Source of Demonstrations. We collect imperfect demonstrations using a policy trained with only 10 epochs of reinforcement learning, i.e., the policy is not converged. These generated trajectories range from random trajectories to near-optimal trajectories. We also collect optimal demonstrations with an optimal policy. For Reacher, Swimmer, and Ant environments, we train the optimal policy by TRPO [36]. For Driving and Panda Robot Arm, it is difficult for TRPO to learn an optimal policy, so we hand-craft an optimal policy.

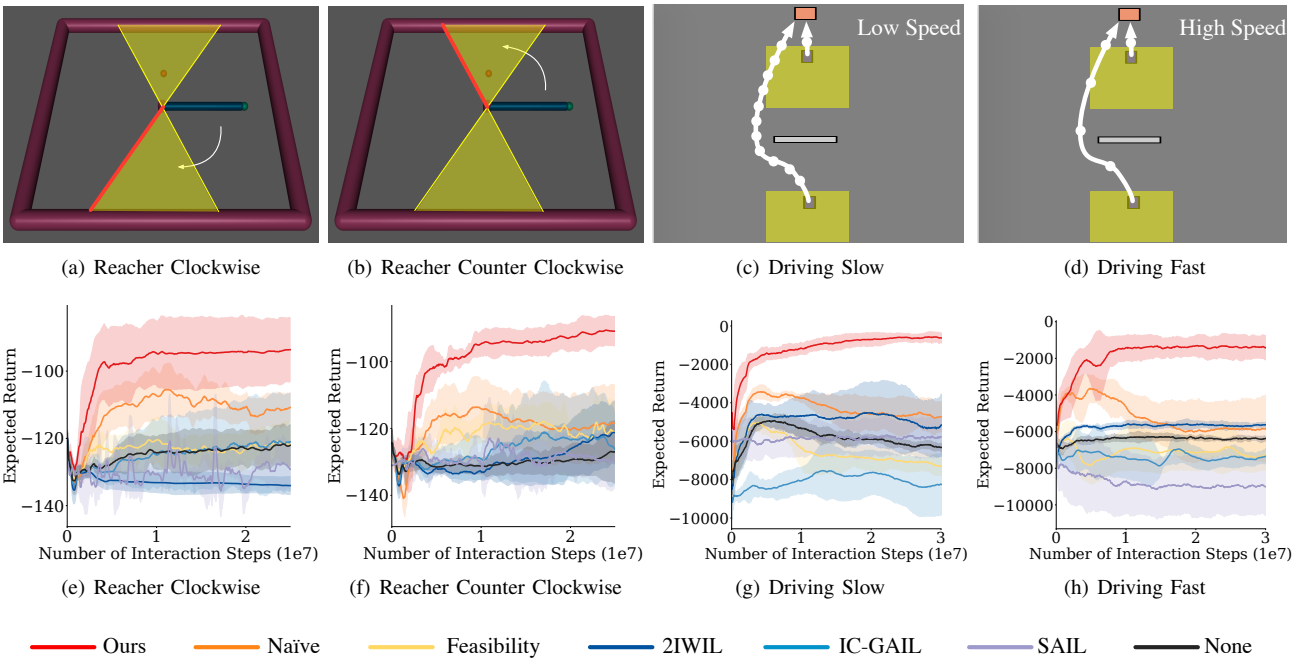
Composition of Demonstrations. We follow the experiment setting of the state-of-the-art weighting-based learning from imperfect demonstration work [7]. We do not make extra assumptions on the quality of demonstrations, and consider the setting, where demonstrations can range from random to fully optimal. Therefore, we employ a mixture of perfect and imperfect demonstrations, where some demonstrations are useful and informative, while others are not as useful or informative. Specifically, we construct a mixture of three types of data: demonstrations of the optimal policy of the target environment, demonstrations of the sub-optimal policy of the target environment, and the demonstrations from an agent with different dynamics.

In each experiment, we use a set of 1000 demonstrations for each environment. To achieve a non-trivial imitation learning problem, we select ratios of each type of demonstration appropriately to ensure that the problem cannot easily be solved by vanilla imitation learning. The specific composition of demonstrations in each environment is discussed below.

Other Methods. We compare our proposed approach with the most relevant learning from demonstration works: imitation learning (specifically we compare with Generative Adversarial Imitation Learning (GAIL)) [3], imitation learning from imperfect demonstration: 2IWIL and IC-GAIL [7], and imitation learning from different dynamics: SAIL [28].

A. Environments with Return Influenced by the Initial State

Reacher. We experiment with openAI gym's reacher environment [37], [38], where the end effector of the agent is supposed to reach a final location. We modify the original reacher environment, where the agent only has one joint and one link as shown in Fig. 3(a) and 3(b). In addition, we limit the final goal location (the red ball) within a particular area, which is shown as the shaded yellow area composing of two triangles, where the vertex angle of the triangle at the center is $53.1°(2\arctan(0.5))$. Here, we consider two possible dynamics: (1) the agent can only rotate clockwise with its joint limit reaching at most the red line shown in Fig. 3(a) and (2) the agent can only rotate counter-clockwise with its joint limit reaching at most the red line shown in Fig. 3(b). We use the shaded yellow area and the limit on rotation to make the optimal trajectories for one dynamics to be quite different from the other. For instance, for a goal location in the up triangle, the optimal trajectories are quite different: for the first dynamics the optimal trajectory is to stay still, while the optimal trajectory for the second dynamics is to rotate



(a) Reacher Clockwise (b) Reacher Counter Clockwise (c) Driving Slow (d) Driving Fast

(e) Reacher Clockwise (f) Reacher Counter Clockwise (g) Driving Slow (h) Driving Fast

Fig. 3. (a-b) are the illustrations of Reacher environments with clockwise and counter-clockwise dynamics. (c-d) are the illustrations of Driving environments with low speed and high speed dynamics. (e,f,g,h) are the expected return results on environments (a,b,c,d) respectively. One x-axis unit is 10^7 steps.

counter-clockwise toward the goal. So it is difficult for agents with different dynamics to learn from each other.

Driving. We consider a simple 2D driving environment. As shown in Fig. 3(c) and 3(d), the blue car aims to drive to the goal while avoiding the white obstacle. As introduced in Section IV-B, when different initial states induce different highest expected returns, the optimality based on the raw expected return may fail. To test the difficulty, we set the initial states to be in the yellow areas and define the reward function (just for evaluation) as $\mathcal{R}(s_t, s_{t+1}) = \alpha_1 \mathbf{1}_{\text{goal}}[s_{t+1}] + \alpha_2 \mathbf{1}_{\text{obstacle}}[s_{t+1}] + \alpha_3 \mathbf{1}_{\text{out}}[s_{t+1}] + \alpha_4$, where $\mathbf{1}_{\text{goal}}$ is an indicator of when the next state s_{t+1} reaches the goal, $\mathbf{1}_{\text{obstacle}}$ is an indicator of when s_{t+1} collides with the obstacle, and $\mathbf{1}_{\text{out}}$ is an indicator of when s_{t+1} moves out, and α_i are appropriate weights for each indicator. Then the initial location closer to the goal has higher optimal expected return. Here, we fix the car speed and only control the steering of the car. As shown in Fig. 3(d) and Fig. 3(c), we create different dynamics by setting a high and a low speed for the car.

Compostion of Demonstrations. The ratio of demonstrations from the optimal policy of the target environment, the sub-optimal policy of the target environment, and the optimal policy in a different environment is $5\%, 90\%, 5\%$ respectively for the Reacher environment and $30\%, 60\%, 10\%$ respectively for the Driving environment.

Results. Fig. 3 shows that our proposed approach outperforms 2IWIL, IC-GAIL, and SAIL. 2IWIL and IC-GAIL focus on sub-optimal demonstrations while SAIL focuses on demonstrations with varying dynamics. When both kinds of demonstrations are present, 2IWIL, IC-GAIL, and SAIL cannot learn a high return policy from the demonstrations. Our method employs feasibility and optimality to filter harmful demonstrations and enable successful imitation learning.

We also perform ablation studies on our method. In the driving and Reacher environments, since the expected return of the trajectory is influenced by the initial state, the proposed f_{rec} is needed. So the optimality depends on the feasibility as shown in Eqn. (5), and we cannot remove feasibility while preserving optimality. We demonstrate the necessity of f_{rec} by removing f_{rec} and using a naïve optimality purely based on the reward (Naïve). We observe that our method outperforms Naïve with a large margin in both environments. We then remove the optimality and only use the feasibility score. Our method outperforms Feasibility in both Reacher and Driving environments, indicating the importance of optimality to filter sub-optimal demonstrations. Note that Naïve performs similar to Feasibility. This is because in the environments where the expected return is influenced by the initial state, directly using expected return as the optimality tends to select initial states inducing high expected returns but ignore other useful initial states. This can influence the diversity of the demonstrations and make the policy fail on low expected return initial states.

B. Environments with Similar Return across Initial States

Swimmer. In the Swimmer environment, we have an agent with three links and two joints. The goal for the swimmer is to move forward by rotating the joints. As shown in Fig. 4(a) and Fig. 4(b), we create two different dynamics: (1) the blue joint is disabled (BackDisabled) and (2) the red joint is disabled (FrontDisabled), where disabled means the joint angle is fixed to 0 (the two links are in a line).

Ant. In the Ant environment, we have an ant with four legs where each leg consists of two links and two joints. The goal for the ant is to move forward in the x-axis direction as fast as possible. As shown in Fig. 4(c) and Fig. 4(d), we create different dynamics by setting the legs at different angles relative to each other: (1) the four legs can only rotate within $[0°, 53°]$

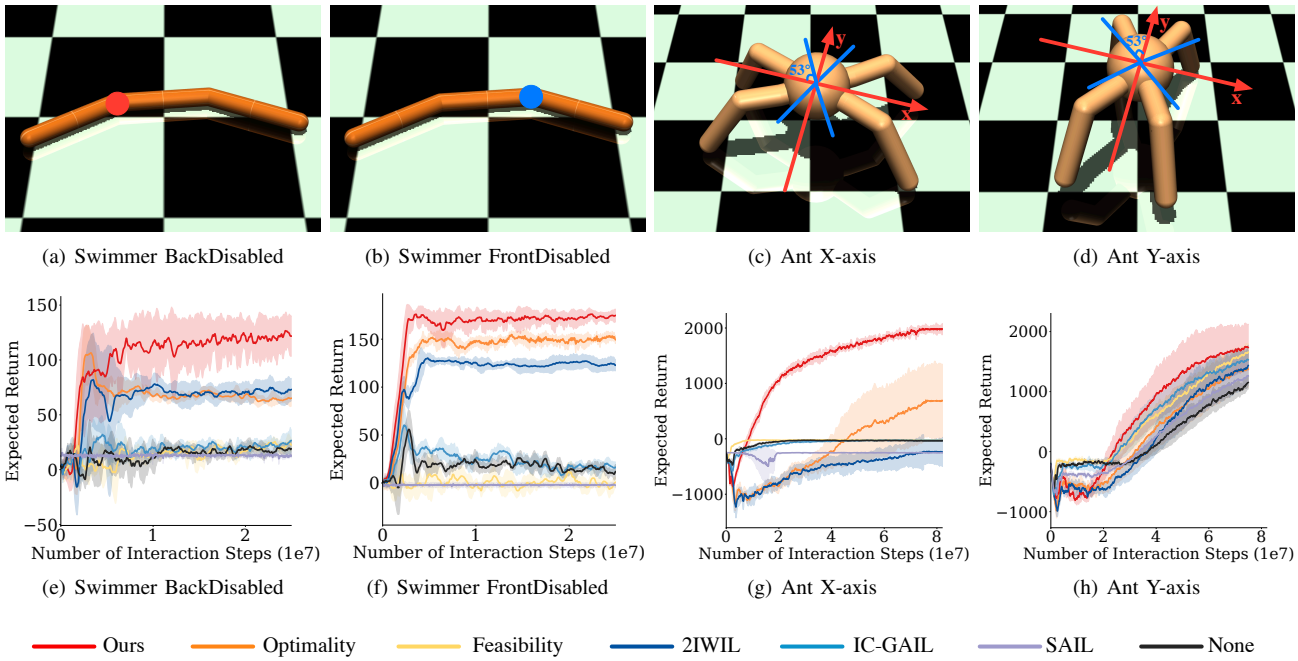


Fig. 4. (a-b) are the illustrations of Swimmer with disabled back joint and disabled front joint. (c-d) are the illustrations of Ant environment with dynamics x-axis aligned and y-axis aligned dynamics. (e,f,g,h) are the expected return results on environments (a,b,c,d) respectively. One x-axis unit is 10^7 steps.

from the x-axis. and (2) the four legs can only rotate within $[0°, 53°]$ from the y-axis.

Composition of Demonstrations. The Swimmer and the Ant environments are two other interesting environments that we study. The ratio of demonstrations from optimal policy of the target environment, the sub-optimal policy of the target environment, and the optimal policy of a different environment is $1\%, 49.5\%, 49.5\%$ respectively for the Swimmer environment and for the Ant environment.

Results. In Fig. 4, we observe that the proposed method outperforms all the other methods with large margin in the expected return while having similar converging speed even on these complex tasks. 2IWIL significantly outperforms other baseline methods. This is because the influence of the sub-optimal demonstrations are larger than demonstrations from different dynamics. 2IWIL tends to filter more sub-optimal demonstrations, and thus performs better. However, 2IWIL still performs worse than our proposed method since our method further filters infeasible demonstrations, and learns from more feasible demonstrations from different dynamics.

The expected return of the trajectory is not influenced by the initial state for the Swimmer and the Ant environments, so the optimality does not depend on the feasibility. Therefore, our proposed optimality equals to using the raw expected return as the optimality score. Here, we run an ablation study, where we remove optimality and feasibility individually, denoted as Optimality and Feasibility. We observe that our approach outperforms both Feasibility and Optimality, which demonstrates that both feasibility and optimality are important elements in selecting useful demonstrations.

C. Robot Experiments

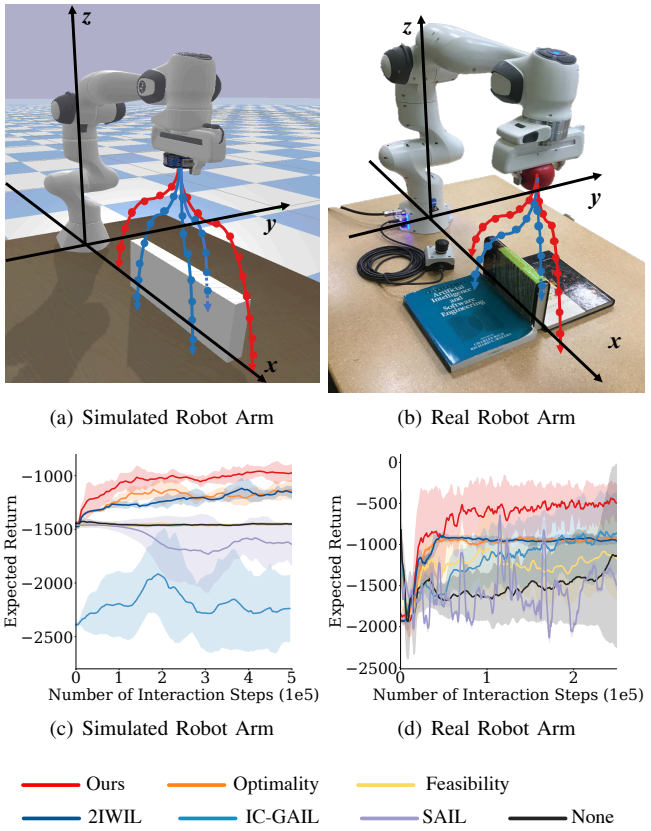
The Panda Robot Arm environment uses a 7 Degrees of Freedom (DoF) Franka Panda Robot arm [39]. We create two dynamics: (1) the original 7-DoF arm and (2) a 3-DoF arm by fixing all the joints that rotate around the z-axis. As shown in Fig. 5(a) and Fig. 5(b), we conduct experiments on a task, where we want the robot arm to take an object from a starting configuration above the table and move to the table without colliding with the obstacle. The 7-DoF arm can have various paths shown in blue trajectories while the 3-DoF robot can only move along the x-axis resulting in either one of the two red trajectories to avoid the obstacle. So different dynamics can introduce different optimal trajectories.

Composition of Demonstrations. The ratio of demonstrations from optimal policy of the target environment, the suboptimal policy of the target environment, and the optimal policy of a different environment is $1\%, 49.5\%, 49.5\%$.

Results. In the robot environment the initial state does not influence the highest expected return much, so we compare the proposed method with Feasibility and Optimality similar to the Swimmer and Ant environments. We conduct experiments both on a simulated panda robot arm with pybullet [40] and a real robot arm. In Fig. 5, we observe that the proposed method outperforms all the baselines with a large margin. Our method outperforms the variants with only optimality or only feasibility considered individually, which demonstrates the necessity of both feasibility and optimality in realistic environments.

VI. Conclusion

Summary. In this paper, we propose an algorithm to learn from imperfect demonstrations from agents with different dynamics, where the demonstrations can be drawn from non-expert policies or from agents with different dynamics. We design a feasibility score and an optimality score to select demonstrations that are useful for the target agent. We show that the policy learned from the selected demonstrations outperforms other imitation learning methods on various environments and different composition of demonstrations.



(a) Simulated Robot Arm (b) Real Robot Arm

(c) Simulated Robot Arm (d) Real Robot Arm

Fig. 5. (a-b) are the illustrations of a simulated robot arm and a real robot arm, where the blue curves are demonstrated by the 7-DoF arm, while the 3-DoF arm can only demonstrate the red curves. (c-d) are the corresponding expected returns. One x-axis unit is 10^5 steps.

Limitations and Future Work. Although our work relaxes stringent assumptions placed on imitation learning algorithms, it is also limited in a few ways: To compute feasibility, our method uses the inverse dynamics function and the environment for trajectory sampling, which might not be accessible in some settings. To compute the rectify function f_{rec}, we need the neighborhood property, which may also be violated in particular environments. For example, in navigation, if two nearby locations are separated by an obstacle, their navigation route can differ significantly and the highest expected return might also be quite different. In the future, we plan to address these challenges by incorporating structures about the problem that can help us better learn the inverse dynamics or rectify functions that do not rely on the neighborhood property.

VII. Acknowledgements

We would like to thank FLI grant RFP2-000 and NSF Award Number 1849952 for their support.

References

[1] M. Bain and C. Sammut, "A framework for behavioural cloning." in *Machine Intelligence 15*, 1995.
[2] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*, 2008.
[3] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NeurIPS*, vol. 29, 2016.
[4] Z. Cao, E. Bıyık, W. Z. Wang, A. Raventos, A. Gaidon, G. Rosman, and D. Sadigh, "Reinforcement learning based control of imitative policies for near-accident driving," *arXiv:2007.00178*, 2020.
[5] H. A. Simon, *Models of bounded rationality: Empirically grounded economic reason*. MIT press, 1997.
[6] D. P. Losey, K. Srinivasan, A. Mandlekar, A. Garg, and D. Sadigh, "Controlling assistive robots with learned latent actions," in *ICRA*, 2020.
[7] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama, "Imitation learning from imperfect demonstration," in *ICML*, 2019.
[8] D. H. Grollman and A. Billard, "Donut as i do: Learning from failed demonstrations," in *ICRA*, 2011.
[9] Z. Zhu, K. Lin, B. Dai, and J. Zhou, "Learning sparse rewarded tasks from sub-optimal demonstrations," *arXiv:2004.00530*, 2020.
[10] V. Tangkaratt, B. Han, M. E. Khan, and M. Sugiyama, "Variational imitation learning with diverse-quality demonstrations," in *ICML*, 2020.
[11] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.
[12] H. Daumé, J. Langford, and D. Marcu, "Search-based structured prediction," *Machine learning*, 2009.
[13] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *AISTATS*, 2010.
[14] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004, pp. 1–8.
[15] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning." in *ICML*, 2000, pp. 663–670.
[16] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *ICLR*, 2018.
[17] H. Xiao, M. Herman, J. Wagner, S. Ziesche, J. Etesami, and T. H. Linh, "Wasserstein adversarial imitation learning," *arXiv preprint arXiv:1906.08113*, 2019.
[18] L. Ke, M. Barnes, W. Sun, G. Lee, S. Choudhury, and S. Srinivasa, "Imitation learning as f-divergence minimization," in *Workshop on the Algorithmic Foundations of Robotics*, 2020, pp. 313–329.
[19] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *IJCAI*, 2018, pp. 4950–4957.
[20] Y. Schroecker and C. L. Isbell, "State aware imitation learning," in *NeurIPS*, 2017.
[21] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *Imitation, Intent, and Interaction (I3) Workshop at ICML*, 2019.
[22] W. Sun, A. Vemula, B. Boots, and D. Bagnell, "Provably efficient imitation learning from observation alone," in *ICML*, 2019.
[23] C. L. Nehaniv, K. Dautenhahn, *et al.*, "The correspondence problem," *Imitation in animals and artifacts*, vol. 41, 2002.
[24] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, 2009.
[25] P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth, "Addressing the correspondence problem by model-based imitation learning," in *ICRA Workshop on Autonomous Learning*, 2013.
[26] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *TSMC*, 2007.
[27] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard, "Imitation learning with generalized task descriptions," in *ICRA*, 2009.
[28] F. Liu, Z. Ling, T. Mu, and H. Su, "State alignment-based imitation learning," in *ICLR*, 2019.
[29] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *ICRA*, 2018.
[30] C. Basu, Q. Yang, D. Hungerman, M. Singhal, and A. D. Dragan, "Do you want your autonomous car to drive like you?" in *HRI*, 2017.
[31] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *IJSR*, 2012.
[32] M. Kwon, E. Biyik, A. Talati, K. Bhasin, D. P. Losey, and D. Sadigh, "When humans aren't optimal: Robots that collaborate with risk-aware humans," in *HRI*, 2020.
[33] J. Lee, C.-A. Cheng, K. Goldberg, and B. Boots, "Continuous online learning and new insights to online imitation learning," *arXiv preprint arXiv:1912.01261*, 2019.
[34] A. H. Qureshi, B. Boots, and M. C. Yip, "Adversarial imitation via variational inverse reinforcement learning," in *ICLR*, 2019.
[35] J. Zhang, Z. Ding, W. Li, and P. Ogunbona, "Importance weighted adversarial nets for partial domain adaptation," in *CVPR*, 2018.
[36] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015.
[37] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv:1606.01540*, 2016.
[38] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*, 2012.
[39] F. E. GmbH. Introduction. [Online]. Available: https://www.franka.de/
[40] E. Coumans, Y. Bai, and J. Hsu, "Pybullet physics engine," 2018.