Appendix for Learning Feasibility to Imitate Demonstrators with Different Dynamics

Zhangjie Cao[1], **Yilun Hao**[1], **Mengxi Li**[2], **Dorsa Sadigh**[1,2]
[1]Department of Computer Sciences, Stanford University, United States
[2]Department of Electrical Engineering, Stanford University, United States
`caozj@cs.stanford.edu`, `{yilunhao,mengxili}@stanford.edu`, `dorsa@cs.stanford.edu`

In this Appendix, we provide the proposed method under stochastic MDPs in Sec. A. We provide the details of the algorithm in Sec. B. We further explain our experimental details in Sec. C. Finally, we show some extra experimental results in Sec. D.

A The Extension to Stochastic MDPs

In our main text, we discussed the deterministic MDP setting as all our experiments are in the deterministic setting, but here we would like to extend the discussion to the stochastic MDP case. In a stochastic MDP, our goal is still to learn a policy $\pi^i : \mathcal{S} \rightarrow \mathcal{A}^i$ for the imitator to maximally achieve the state transitions in the demonstrations. This means that if the state transition (s^d_t, s^d_{t+1}) from a demonstration is more likely to be feasible, the expected distance between the next state s^i_{t+1} produced by π^i and s^d_{t+1} should be small, i.e., $\mathbb{E}_{s^i_{t+1}\sim p^i(s^d_t,\pi^i(s^d_t))}[f_{\text{dis}}(s^i_{t+1}, s^d_{t+1})]$ should be small. Therefore, the expected distance between s^i_{t+1} and s^d_{t+1} can serve as a measure of feasibility, where a smaller distance corresponds to a higher feasibility.

Under a feasibility metric defined by the expected distance, we can use the same design of f-MDP as the deterministic MDP case: $M^f = \langle \mathcal{S}, \mathcal{A}^i, p^i, \mathcal{R}^f, \rho^f_0, \gamma^f \rangle$. The state space, the action space and the transition probability are all the same as the imitator's. The initial state distribution is defined as $\text{Uniform}(T_0)$. The reward is defined as:

$$s^d_0 \sim \text{Uniform}(T_0), \quad s_0 = s^d_0, \quad s_{t+1} = p^i(s_t, a), \quad R^f(s_t, a, s_{t+1}) = -f_{\text{dis}}(s_{t+1}, s^d_{t+1}). \tag{1}$$

Maximizing the expected return in the f-MDP in such a design will minimize the expected state distance between the learned policy and the demonstrations, which matches our definition of feasibility. After we learn the optimal policy π^* for the f-MDP, we can derive the feasibility for each trajectory ξ with the expected state distance between the demonstration and the policy:

$$w(\xi) = \exp\left(\frac{-\mathbb{E}_{s_t\sim\pi^*}\left[\sum_{t=1}^{N}(\gamma^f)^t f_{\text{dis}}(s_t, s^d_t)\right] - C}{\sigma}\right). \tag{2}$$

In our experiments, we only consider the case where the MDP is deterministic.

B Algorithm

We go through the steps of the algorithm of learning feasibility to imitate demonstrators with different dynamics in Algorithm 1. For the state-based imitation learning algorithm used to learn the final policy from reweighted demonstrations, we use the state-based GAIL as [1]. Lines 1-6 show the process of constructing N f-MDPs (one for each demonstrator) and training the optimal policy for each f-MDP. Line 7 shows how to compute feasibility with the optimal policy for each f-MDP. Lines 8-11 show imitation learning over the newly reweighted demonstrations.

Currently, we consider the state transitions in each trajectory share the same feasibility but do not consider the case where parts of the trajectories are more feasible than some other parts. This is because the state-based GAIL in our algorithm as well as many standard imitation learning algorithms

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

```
Algorithm 1: Algorithm
Input: Demonstrations $\Xi^j$ from each demonstrator $\mathcal{M}_j^d$, $(1 \le j \le N)$.
for j=1 to N do
    Construct the trajectory f-MDP $M_j^f$ based on the demonstration set $\Xi_j$ according to Eqn. (2);
    Train an optimal policy $\pi_j^*$ for the trajectory f-MDP $M_j^f$;
    Compute the feasibility $w(\xi_j)$ for each trajectory $\xi_j \in \Xi_j$ as in Eqn. (3);
    Assign the feasiblity $w(\xi_j)$ to each state transitions in $\xi_j$;
end
Compute $p_w((s_t^d, s_{t+1}^d)) \leftarrow \frac{w((s_t^d, s_{t+1}^d))}{\sum_{\left(s_{t'}^d, s_{t'+1}^d\right) \in T} w((s_{t'}^d, s_{t'+1}^d))}$
while not converging do
    Sample a batch of state transitions from $p_w$;
    Train $\pi$ with the sampled batch of state transitions by an state-based imitation learning
      algorithm: state-based GAIL
        $\min_{\theta_\pi} \max_{\theta_d} \mathbb{E}_{(s_t, s_{t+1}) \sim \pi}(\log(D(s_t, s_{t+1}))) + \mathbb{E}_{(s_t, s_{t+1}) \sim p_w}(1 - \log(D(s_t, s_{t+1}))),$    (3)
      where $\theta_\pi$ is the parameter of $\pi$ and $\theta_d$ is the parameter for the discriminator $D$ in
      state-based GAIL;
end
Output: Learned optimal policy $\pi^*$ for $\mathcal{M}^i$.
```

rely on learning from the full trajectory from the start to the end state. If a segment of the trajectory is far from feasible or harmful, then the remaining part is also not going to be useful for our algorithm. Therefore, we only learn from trajectories that are helpful in all parts.

C Experiment Details

In this section, we discuss experimental details for all of our environments. In addition, we discuss the choice of our discount factor γ^f in C.3

C.1 Mujoco

Table 1: The composition of demonstrations for each environment

Environment	Number of Demonstrations			
	i	ii	iii	iv
Swimmer	50	50	500	500
Walker2d (first setting)	50	50	500	500
Walker2d (second setting)	500	500	10	10
HalfCheetah (first setting)	25	25	500	500
HalfCheetah (second setting)	500	500	25	25
Hopper	50	50	500	500

Composition of Demonstrations. The composition of demonstrations in each environment is shown in Table 1. We design the composition of demonstrations to ensure that directly performing imitation learning on all the demonstrations cannot learn an optimal policy as otherwise we do not need to consider the problem of learning from demonstrations from agents with different dynamics.

Implementation Details. To implement our algorithm, we use TRPO [2] as the RL algorithm to learn the optimal policy from f-MDP, and we use the GAIL from Observation algorithm [3] as our imitation learning technique to learn the optimal policy from the reweighted demonstrations. For each demonstrations, we create a separate f-MDP for its demonstrations and train an optimal policy for the f-MDP to generate the feasibility for its demonstrations.

Compared to the final imitation learning algorithm, which requires about 7×10^7 interactive steps with the environment to converge, learning the optimal policy from f-MDP only needs about 5×10^5 time steps, which is significantly smaller. This indicates that the proposed feasibility learning is efficient even with an additional RL learning process.

C.2 Simulated and Real Robot Arm

Reward Function. In the main text, we evaluate our policy for the two robot arm environments using the expected return. We introduce the details of the reward function here. The exact formula of the reward function is $r = -s - 10000h + 5000g$, where r is the reward, s is the number of steps, $h \in \{0, 1\}$ represents whether the robot hits any object in the environment, and $g \in \{0, 1\}$ represents whether the robot achieves the goal.

Composition of Demonstrations. The demonstration set composed of 5 trajectories from the 3 DoF Panda robot arm with disabled joints and 43 trajectories from the 7-DoF Panda arm for both the simulated and the real arm environments.

Implementation Details. To implement our algorithm, we use TRPO [2] as the RL algorithm to learn the optimal policy from f-MDP. To learn the optimal policy from the reweighted demonstrations, we learn a beta-VAE [4] to imitate the state transition sampled from the reweighted distribution of state transitions p_w. After learning the state transitions, we recover the joint actions from the changes of the end-effector's position using inverse kinematics.

Compared to learning the beta-VAE model, which requires about 2.56×10^4 interactive steps with the environment to converge, learning the optimal policy from f-MDP only needs about 5.12×10^3 time steps, which is negligible with respect to the imitation learning algorithm. This indicates that the proposed feasibility learning is efficient even with an additional RL learning process.

C.3 Choice of Discount Factor

The choice of the discount factor γ^f depends on how fast the compounding error increases with respect to the number of steps in the environment. Faster increasing compounding errors need smaller γ^f. In practice, this depends on the scale of the 'movement' of the agent at each step. For example, for a robot arm, if each joint can only move at a small angle at each step, we can set γ^f to be larger or if each joint can move at a larger angle at each step, γ^f should be set smaller. In our experiments, we fix the $\gamma^f = 0.9$ and empirically it works well for all the environments.

D Experimental Results

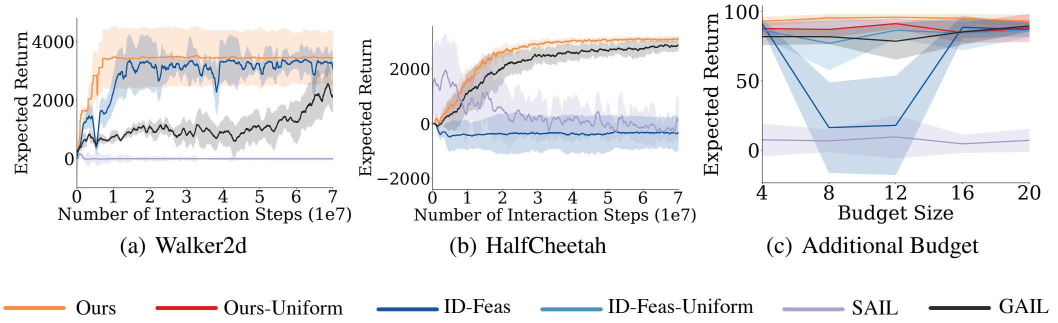
In this section, we provide a number of additional experiments and results including additional results in the Mujoco environment and the simulated robot (D.1, D.3), results demonstrating the effects of varying the number of demonstrations (D.4), discussion and additional results with varying choices of distance metrics (D.5), comparison with a mapping-based method (D.6), and comparison of the expected return of our method with the expected return of the originally collected demonstrations (D.7).

D.1 More Results for Mujoco Environments

We show the results of the second setting for the Walker2d and the HalfCheetah in Fig. 1. We observe that our proposed feasibility achieves the best performance among all the methods. The highest p-value comparing our method to baselines is 0.297 with ID-Feas for the Walker2d environment, and 0.0037 with ID-Feas for the HalfCheetah environment (statistically significant) respectively.

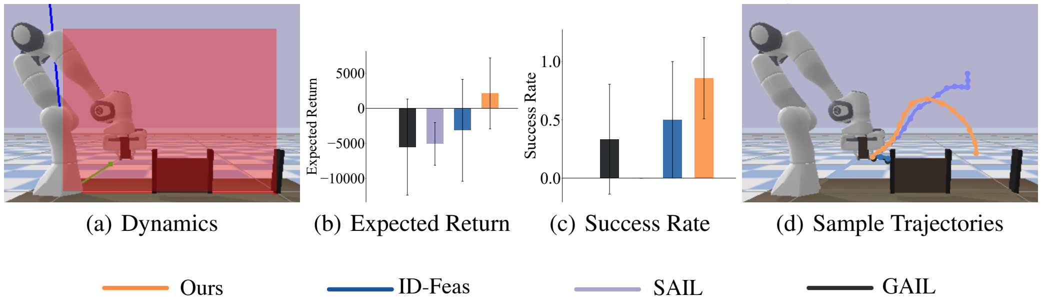
D.2 Performance with a Budget of Additional Demonstrations.

We now consider a setting, where we start with a limited set of demonstrations, but acquire more demonstrations under a limited budget. Our feasibility metric can assess how likely it is for a demonstrator to produce feasible demonstrations, and hence can help us select which demonstrator to query for more demonstrations. We start with one demonstration from each demonstrator in the Swimmer environment and evaluate the performance as we add demonstrations. For our method and



(a) Walker2d (b) HalfCheetah (c) Additional Budget

Figure 1: (a-b) show the expected return for the second setting for the Walker2d and the HalfCheetah environments respectively. (c) The expected return with varying budget of additional demonstrations in Swimmer.



(a) Dynamics (b) Expected Return (c) Success Rate (d) Sample Trajectories

Figure 2: (a) The illustration of different dynamics in the simulated robot arm environment. The 7 DoF robot arm can move in the whole 3D space while the 3 DoF arm can only move in the red plane; (b-c) The bar plot for the expected return and the success rate for the simulated robot arm environment; (d) Sampled trajectories for different methods in the simulated robot arm environment.

ID-Feas, we can acquire demonstrations proportional to the computed feasibility score. We compare the expected return with demonstrations selected based on feasibility (Ours, ID-Feas) to the expected return with demonstrations uniformly acquired from each demonstrator (Ours-Uniform, ID-Feas-Uniform). We further compare with SAIL and GAIL, where no feasibility is defined and we uniformly acquire demonstrations. As shown in Fig. 1(c), Ours outperforms ID-Feas, which demonstrates that the proposed feasibility can better reflect how likely each demonstrator produces feasible demonstrations and acquire more demonstrations from helpful demonstrators. Ours outperforms all the other methods including Ours-Uniform (although not with statistical significance), which indicates that the demonstrations acquired based on the feasibility gain more useful information.

D.3 Results on the Simulated Robot

As shown in Fig. 2, We observe that the proposed approach outperforms the baseline methods both in terms of the expected return and the success rate. The highest p-value for the expected return is 7.252×10^{-9} and for the success rate is 1.047×10^{-8} (both with ID-Feas), which are both statistically significant. The sampled trajectory show that the proposed approach achieves an efficient trajectory successfully moving the book to the empty area of the shelf.

D.4 Varying the Number of All Demonstrations

We vary the number of demonstrations from all demonstrators. We conduct experiments on the first setting of the all the Mujoco environments. For the Swimmer and Walker2d environments, we test with 20% and 50% of the original demonstrations. For the HalfCheetah environment, we test with 40% and 60% of the original demonstrations since we have much less demonstrations (25 vs 50)

Table 2: The performance of varying percentage of demonstrations used with respect to the original setting.

Method	Swimmer			Walker2d			HalfCheetah			Hopper		
	20%	50%	100%	20%	50%	100%	40%	60%	100%	20%	50%	100%
GAIL	40.0±34.3	37.9±35.2	30.9±23.5	276.9±18.3	313.1±63.0	261.1±5.6	2464.9±460.2	2597.2±399.0	2443.6±440.7	2798.3±351.1	2996.6±623.2	3009.6±362.4
SAIL	-7.7±19.7	-5.5±24.6	-3.0±28.4	8.5±17.5	-5.3±56.4	19.0±30.1	-556.7±365.8	-503.3±299.3	-603.0±389.6	-252.6±432.6	-1622.2±1780.1	-7.00±11.4
RAL	23.1±33.9	30.6±28.1	48.2±39.0	244.3±27.4	261.0±46.7	227.0±24.2	2604.4±423.1	2515.3±311.0	2594.4±508.7	2040.3±408.3	2108.9±611.9	1916.1±750.4
Ours	**68.2**±24.7	**76.4**±22.1	**74.3**±20.1	**3137.6**±50.2	**3127.0**±30.7	**3144.3**±23.5	**2716.7**±301.6	**2812.4**±261.2	**2830.6**±292.6	**3273.3**±180.2	**3351.0**±146.3	**3329.6**±115.2

from the demonstrators similar to the imitator. As shown in Table 2, we observe that our approach outperforms all the other methods when having access to different number of demonstrations.

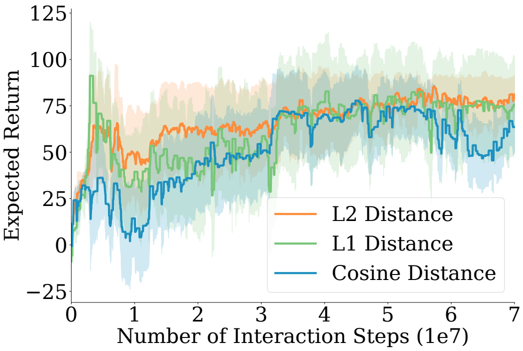


Figure 3: The expected return with respect to the number of steps with different choices of distance metrics.

Method	Expected Return
GAIL [5]	31.20±22.25
SAIL [6]	0.56±4.27
DCC [7]	5.32±3.43
ID-Feas [1]	48.96±38.50
Ours	74.89±19.68

Table 3: The expected return of the learned policy in the Swimmer environment (with standard deviation).

D.5 Discussion of the Choice of Distance Functions

We use L2 distance between states in the reward function in Eqn. (2) and (3) in the main text and in all the experiments. This is because in all our environments, the L2 distance can accurately measure the distance between states. However, this does not mean that the distance metric in our reward of f-MDP is restricted to the L2 distance. We can change the distance depending on the specific state space. For example, for a state space with unit vectors, we can use the cosine distance as the distance metric.

In Fig. 3, we show the expected return of our method by using different distances in the Swimmer environment. We use L1 distance and Cosine distance (the cosine of the angles between two state vectors) as examples. We observe that L1 distance, which is another distance derived by norm, performs close to L2 distance, but Cosine distance performs worse than L2 distance because Cosine distance only cares about the distance on angle but ignores the scale of the vectors, while in Swimmer, the scale of the states is also important. The results show that the choice of this distance function is flexible and depends on the specific choice of the state space in our problem.

D.6 Comparison with Mapping-Based Methods

Mapping-based methods translate the demonstrations across different environments by learning state mappings and action mappings [7], which can be used in our problem setting by mapping the source demonstrations to the target environment. However, our problem setting does not ensure that there exist a mapping between the demonstrators and the imitator, which violates the assumption of the mapping-based methods. We thus do not include any mapping-based methods in our experiments in the main body of our work. However, here as an additional experiment, we compare our method with the state-of-the-art mapping-based method, DCC [7].

DCC requires random trajectories from both the demonstrators and the imitator to learn a mapping, but we do not have access to the demonstrators' environment and only have access to demonstrators' demonstrations. So we use the demonstrations and the imitator's random trajectories as the input to DCC. As shown in Table 3, the performance of DCC is much worse than our method and even worse than GAIL. This is because DCC itself is a good mapping-based method but mapping-based methods are not quite suitable for our problem setting. In fact, there should not exist a mapping between demonstrations and the imitator's random trajectories. Building such a mapping causes severe mismatch between states and actions of different environments and makes the translated demonstrations distort the original demonstrations.

Table 4: The average expected return of demonstrations in different environments and the expected return of our method.

	Swimmer	Walker2d First	Walker2d Second	HalfCheetah First	HalfCheetah Second	Hopper	Simulated Robot	Real Robot
Demonstrations	106±3	3098±118	3720±336	3229±170	3337±67	3460±87	1823±110	2531± 362
Ours	75±20	3147±10	3424±645	2832±291	3142±89	3330±115	2127±5053	2746±2712

D.7 Comparison with the Collected Demonstrations

We compare the expected return of our approach with the demonstrations in Table 4. We observe that in the first setting of Walker2d environment, Hopper environment, the simulated robot and the real robot environments, our approach performs comparably to the expected return of demonstrations, which are optimal demonstrations for different demonstrators. In the Swimmer, the second setting of Walker2d and the HalfCheetah environments, the performance is worse than the demonstrations. This is because only a few demonstrations are feasible for the imitator and that may not be enough to learn an optimal policy. However, the margin between our approach and the demonstrations is still not large. The results show that the proposed feasibility can select useful demonstrations for the imitator to imitate.

References

[1] Z. Cao and D. Sadigh. Learning from imperfect demonstrations from agents with varying dynamics. *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[2] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, 2015.

[3] F. Torabi, G. Warnell, and P. Stone. Generative adversarial imitation from observation. *Imitation, Intent, and Interaction (I3) Workshop at ICML*, 2019.

[4] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[5] J. Ho and S. Ermon. Generative adversarial imitation learning. In *NeurIPS*, volume 29, 2016.

[6] F. Liu, Z. Ling, T. Mu, and H. Su. State alignment-based imitation learning. In *ICLR*, 2019.

[7] Q. Zhang, T. Xiao, A. A. Efros, L. Pinto, and X. Wang. Learning cross-domain correspondence for control with dynamics cycle-consistency. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QIRlze3I6hX.