Skeletal Feature Compensation for Imitation Learning with Embodiment Mismatch

Eddy Hudson[1], Garrett Warnell[1,2], Faraz Torabi[1] and Peter Stone[1,3]

Abstract— Learning from demonstrations in the wild (e.g. YouTube videos) is a tantalizing goal in imitation learning. However, for this goal to be achieved, imitation learning algorithms must deal with the fact that the demonstrators and learners may have bodies that differ from one another. This condition — "embodiment mismatch" — is ignored by many recent imitation learning algorithms. Our proposed imitation learning technique, SILEM (Skeletal feature compensation for Imitation Learning with Embodiment Mismatch), addresses a particular type of embodiment mismatch by introducing a learned affine transform to compensate for differences in the skeletal features obtained from the learner and expert. We create toy domains based on PyBullet's HalfCheetah and Ant to assess SILEM's benefits for this type of embodiment mismatch. We also provide qualitative and quantitative results on more realistic problems — teaching simulated humanoid agents, including Atlas from Boston Dynamics, to walk by observing human demonstrations.

I. INTRODUCTION

Endowing artificial agents with the ability to learn new behaviors by watching humans is a lofty goal in the field of machine learning. Artificial agents that can learn in this way, i.e, perform *learning from demonstration* (LfD) or *imitation learning* (IL), have enormous learning potential. First, IL enables machines to learn new skills from demonstrators that are experts in particular tasks of interest (e.g., operating construction equipment or cooking) rather than from researchers who are experts at writing computer code or designing cost functions. Second, this paradigm of learning provides a channel of skill acquisition even in cases where it is currently prohibitively difficult to induce the desired behavior by writing computer code or specifying a cost function. Finally, the greatest allure of being able to perform *LfD in the wild*—in which one seeks to enable an artificial agent to imitate new behaviors using unconstrained video demonstrations of those behaviors (e.g., YouTube videos)— is that there already exists a vast and relatively untapped amount of demonstration data that we can immediately use to drive behavior acquisition in machines. Here, we are motivated by the problem of performing LfD in the wild for robot behavior acquisition, and we seek to make progress toward that goal in a very specific way.

While a great deal of recent research progress has been made on particular variants of imitation learning, relatively little recent work has considered the problem of *embodiment mismatch*, i.e., the situation that arises when a demonstrating agent's body is different than that of the imitator. For example, a student in a fitness class who is 6'3" is able to easily imitate a 5'2" instructor doing jumping jacks. While the specific signals sent to the student's muscles are much different, the jumping-jack motion induced is qualitatively the same. For IL algorithms to be applied to unconstrained videos, they will similarly need to be able to deal with embodiment mismatch of this kind. We envision a two-stage pipeline where computer vision techniques such as keypoint extraction and pose-estimation are used to extract *expert* data from unconstrained videos, and an IL algorithm capable of dealing with the embodiment mismatch in the expert data trains a learner to acquire the skills originally found in the video.

In this paper, we focus our attention on the second stage of such a pipeline, and propose a new imitation learning algorithm called SILEM (**S**keletal feature compensation for **I**mitation **L**earning with **E**mbodiment **M**ismatch) that can help reliably train a learner to imitate an expert with a different body. Central to SILEM is a learned affine transform that compensates for differences in the skeletal features (e.g. joint angles, height of head, etc.) derived from the expert and learner.

What sets our work apart from prior work in this problem setting [8], [15], [16] is that 1) we do not require access to simulations of the expert's body, and 2) through a series of controlled ablation studies (Figures 2, 3), we provide empirical evidence of the detrimental effect embodiment mismatch has in the domains we consider before proceeding to address the issue.

II. BACKGROUND

Our ultimate goal in this work is to learn a controller that solves a sequential decision making problem. Such problems are typically formulated in the context of a Markov decision process (MDP), i.e., a tuple $\mathcal{M} = <\mathcal{S}, \mathcal{A}, T, R, \gamma>$, where $\mathcal{S}$ denotes an agent's state space, $\mathcal{A}$ denotes the agent's action space, $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the environment model which maps state-action pairs to a distribution over the agent's next state, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function

[1]*The University of Texas at Austin* Austin, Texas, USA {eddyhudson, faraztrb}@utexas.edu, pstone@cs.utexas.edu

[2]*Army Research Laboratory* Austin, Texas, USA garrett.a.warnell.civ@army.mil

[3]*Sony AI* Austin, Texas, USA

that provides a scalar-valued reward signal for state-action-next-state tuples, and $\gamma \in [0, 1]$ is a discount factor that specifies how the agent should weight short- vs. long-term rewards. Solutions to sequential decision making problems are often specified by reactive policies $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, which specify agent behavior by providing a mapping from the agent's current state to a distribution over the actions it can take. Machine learning solutions to problems described by an MDP typically search for policies that can maximize the agent's expected sum of future rewards.

The IL problem is typically formulated using an MDP *without* a specified reward function, i.e., $\mathcal{M} \setminus R$. Instead of reward, the agent is provided with *demonstration* trajectories—typically assumed to have been generated by an expert—that specifies the desired behavior, i.e., $\tau_E = (s_0, a_0, s_1, a_1, ...)$. Imitation from observation (IfO) is a sub-problem of IL in which the agent does not have access to the actions taken during the demonstration trajectories, i.e., $\tau_E = (s_0, s_1, ...)$. Techniques designed to solve the IL problem seek to use observed demonstrations to find policies that an imitating agent can use to imitate the demonstrator.

Adversarial imitation learning (AIL) is a particular way to perform IL that has recently come to the fore. AIL is loosely based on GANs [7], in that both involve the same min-max game with discriminators and generator networks. The discriminator D, is trained to distinguish between the demonstration trajectories and trajectories generated by the imitator. In particular, the goal of updating D is to drive $\mathbb{E}_{o \sim \tau_E}[D(o)]$ toward 1 and $\mathbb{E}_{o \sim \tau}[D(o)]$ toward 0, where o is a segment of the trajectory, and τ represents trajectories recently generated by the imitator. In the seminal AIL algorithm GAIL [9], $o = (s_t, a_t)$, whereas in GAIfO [21], $o = (s_t, s_{t+1}, \ldots, s_{t+n})$. Merel *et al.* [12] deviate from this paradigm by preventing the discriminator from accessing actual state information. Instead, they let $o = (g(s_t), g(s_{t+1}), \ldots, g(s_{t+n}))$, where g is an abstract function that extracts features based on the agent's state. In their IL experiments involving 2 and 3 link arms, g simply returned the end effector's location. This representation allowed Merel *et al.* [12] to train a 3-link arm to imitate a 2-link arm. In this work, we refer to the features extracted by g as skeletal features. The generator, which in AIL algorithms is the imitator's policy π, is trained to induce behavior that elicits large output from D, i.e., to "fool" D into thinking that the imitator's trajectories came from the demonstrator. By iteratively updating D and π as described, AIL approaches are able to find imitator policies that successfully mimic the demonstrated behavior.

Key to this work is our observation that, to date, most AIL approaches that have been proposed require the expert and the learner to have the same embodiment. The embodiment mismatch problem we consider here occurs when the demonstrator is structurally different from the learner. For example, in the case of dog-like agents, the demonstrator may be a tall Great Dane and the learner may be an elongated Dachshund. In such cases, it might be impossible to match τ with τ_E. AIL algorithms such as GAIL and GAIfO will tend to suffer degraded performance in this scenario. Therefore,
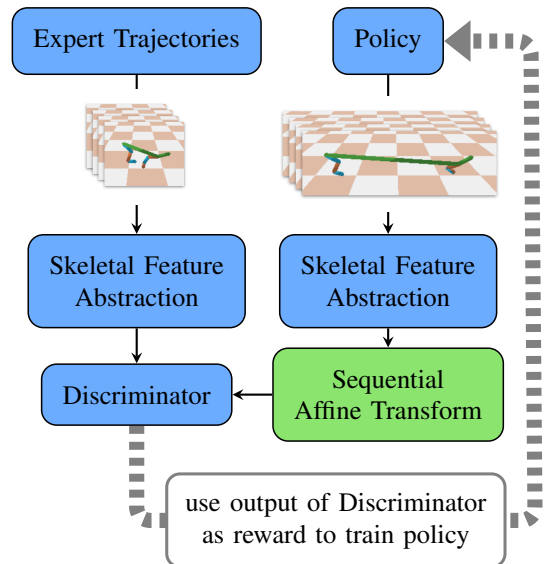

Fig. 1: A high level overview of the proposed technique, SILEM. The policy and discriminator are learned through a mini-max game as in prior AIL algorithms, while the sequential affine transform is learned by backpropagating through the discriminator (Algorithm 1).

solutions must be designed to explicitly account for this kind of embodiment mismatch. Note that our goal is not to train a single policy that is robust to changes in embodiment. Our aim is to train a policy specific to a particular embodiment by leveraging demonstrations from another embodiment.

III. SILEM

In this work, we propose a new AIL algorithm capable of training useful policies in the presence of a particular type of embodiment mismatch, and term it SILEM (**S**keletal feature compensation for **I**mitation **L**earning with **E**mbodiment **M**ismatch). SILEM follows the lead of prior AIL algorithms GAIL and GAIfO in its core operations (Algorithm 1, Figure 1) with one critical exception — the sequential affine transform.

The sequential affine transform is an affine transform with the following functional form:

$$T(o_t) = T\Big(g(s_t), g(s_{t+1}), \ldots, g(s_{t+n})\Big)$$
$$= \Big(f(g(s_t)), f(g(s_{t+1})), \ldots, f(g(s_{t+n}))\Big)$$

where T is the sequential affine transform, and f is an affine transform operating on the skeletal features from single states. i.e., $f(g(s)) = \mathbf{A}g(s) + b$, where $\mathbf{A}$ is a diagonal matrix and g is an abstract function that extracts skeletal features from single states. We restrict our attention to IfO to more seamlessly learn from human demonstrations, which generally do not come with actions.

The sequential affine transform's purpose is to compensate for differences in the skeletal features obtained from expert

and learner bodies, which arise due to embodiment mismatch. It is only applied to skeletal features from the learner, and is optimized by minimizing the following loss function: $L = -\log(D(T(o)))$, while keeping the weights of the discriminator D fixed. T is the sequential affine transform, and $o = (g(s_t), g(s_{t+1}), \ldots, g(s_{t+n}))$ is a sequence of skeletal features from the learner. This loss function is similar to that employed for training Conditional GANs [13]. We frame the sequential affine transform as a conditional GAN that aims to generate skeletal features pertaining to the demonstrator given skeletal features from the learner. Combining this additional step with the essential components of GAIL/GAIfO, each iteration of SILEM comprises three main steps: (Line 9) update the discriminator D, (Line 10) update the sequential affine transform T, and (Line 11) update the policy π.

Algorithm 1 SILEM and SILEM$^-$ (our ablation without the sequential affine transform). Lines in green to be executed only for SILEM (3, 7, 8, 10).

1: Initialize parametric policy π
2: Initialize parametric discriminator D
3: Initialize sequential affine transform T
4: Obtain state-only expert demonstration data $\tau_E = \{o_t\} = \{(s_t, s_{t+1}, \ldots, s_{t+n})\}$
5: **while** π improves **do**
6: Using π, collect learner trajectories $\tau = \{o_t\} = \{(s_t, s_{t+1}, \ldots, s_{t+n})\}$
7: Generate a copy of τ called τ_c
8: Replace each element o_t in τ with $T(o_t)$
9: Update D using the loss: $-\Big(\mathbb{E}_{o \sim \tau_E}[log(D(o))] + \mathbb{E}_{o \sim \tau}[log(1 - D(o))] \Big)$
10: Update T using the loss: $-\mathbb{E}_{o \sim \tau_c}[log(D(T(o)))]$
11: Update π by performing PPO updates with reward function $D(o)$, where $o \in \tau$
12: **end while**

Note that our choice of objective function to train the sequential affine transform runs the risk of interfering with policy learning. In the presence of embodiment mismatch, skeletal features from the learner can differ from the expert both due to embodiment mismatch and due to imperfect imitation. Left to its own devices, this objective function might encourage the sequential affine transform to go above and beyond its purpose by converting skeletal features from faltering learner states to skeletal features from excellent expert states – as opposed to merely compensating for embodiment mismatch. However, we find empirically that, because we use an affine transform rather than a more powerful network (e.g., a multi-layer perceptron (MLP)), this scenario does not occur (Table I).

At first glance, the restriction of having to use an affine transform might appear quite severe. It requires a certain degree of similarity between the expert and learner. However, as we show in the results section, by choosing g appro-

priately, the space of problems solvable by SILEM widens to include potentially impactful problems such as training humanoid robots from human demonstrations. In our experiments involving human demonstrations, we define a separate g for the demonstrator (*expert*) and the learner, which allows us to address instances of embodiment mismatch where there exists a mismatch in the number of joints (see the humanoid walking experiments in Section V). Naturally, as the difference between the expert and learner bodies becomes more and more drastic, the process for designing g will become more and more involved. Fortuitously, as the expert differs from the learner in ever more ways, it becomes less and less useful to the learner since the feedback it can offer the learner also drops precipitously. For instance, humanoid robots can learn a lot more from human demonstrations than from demonstrations by quadrupedal agents.

IV. RELATED WORK

Our work broadly fits in the realm of learning from demonstrations (LfD), or imitation learning. In LfD, the learner is trained to imitate the behavior of an expert demonstrator. One use case for LfD is when it is difficult to specify the target behavior using a scalar reward signal. e.g. performing backflips [2]. This is in contrast to many impressive results in the past decade using deep RL techniques such as in Atari games [14], or Go [18], where the target behavior can be more easily achieved by specifying a scalar reward signal.

In LfD, the expert and learner might face different environments and have different embodiments. Prior work [4], [11] has shown promise in correcting the difference in dynamics faced by the learner and expert. However, these approaches have failed to show concrete evidence that they combat embodiment mismatch.

Manually matching the feature-space of the learner and expert can resolve embodiment mismatch to a certain extent [12], [16]. However, methods such as these may be difficult to scale and often arrive at suboptimal solutions when the embodiment mismatch gets noticeably large. Peng *et al.* [16] also require control over the agent's starting state distribution, thus precluding application of their technique on real robots. Some methods for LfD sidestep the problem of constructing a mapping by using temporal consistency to learn a reward function that is invariant to the environment that the input state belongs to [17]. However, Torabi *et al.* [21] show that it is difficult to learn periodic locomotion behaviors using such reward functions.

There have been a few attempts at automatically creating a mapping between the state-spaces of two different agents [1], [3], [8], [10], [15]. Gupta *et al.* [8] require both the expert and the learner to have solved at least one common task beforehand. This makes it impossible to learn from new demonstrators in the wild, effectively ruling out any application for the LfD in the wild problem. Gamrian *et al.* [3] and Kim *et al.* [10] use a framework based on GANs to map between various domains. However, they train the mapping using a random policy. In many robotics domains, random policies do not provide useful information: a random

Humanoid agent will fall down immediately. Thus a mapping trained using such a policy is unlikely to generalize to walking gaits. Along those lines, Ariki *et al.* [1] also require random explorations, and they do show successful results using a humanoid robot (the NAO robot). However, they fail to evaluate their approach on more complex humanoid agents such as the ones we have used, where random policies fail to provide any useful information. Peng *et al.* [15] use inverse kinematics to approximately match keypoints on the expert with keypoints on the learner. Such an approach, apart from having to be laboriously tuned, has only been shown to work for relatively simple examples of embodiment mismatch. Our best attempt to use the techniques of Peng *et al.* [15] to train the humanoid agents to imitate human demonstrations fails to produce useful policies.
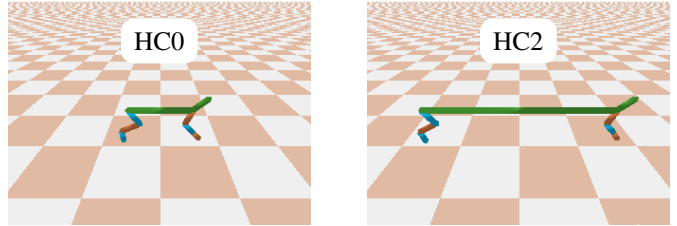
SILEM is closest to the work of Stadie *et al.* [19], Third Person Imitation Learning (TPIL). TPIL employs one discriminator for policy improvement, and another separate discriminator to *cancel* out embodiment mismatch using gradient reversal [5]. The success of the entire approach hinges on the hypothesis that differences in single states reflect only embodiment mismatch and not differences in policy quality. However, for complex domains such as humanoid walking, this hypothesis fails to hold [12]. Our results also confirm that TPIL fails to reliably help humanoid agents imitate human demonstrations.

Within the specific problem setting that we operate, learning locomotion skills in the presence of embodiment mismatch, we are the first to provide definitive evidence that our approach is capable of handling embodiment mismatch. Neither Peng et al. [15] nor Peng et al. [16] perform controlled ablation studies showing that their approaches handle meaningful embodiment mismatch. Gangwani *et al.* [4] perform such ablation studies, but they study transition dynamics mismatch. Ghadirzadeh *et al.* [6] and Yu *et al.* [22] provide impressive results, but they do so on robotic arms, where the challenge is in dealing with a wide diversity of physical objects while the actual task remains simple: reaching and picking.
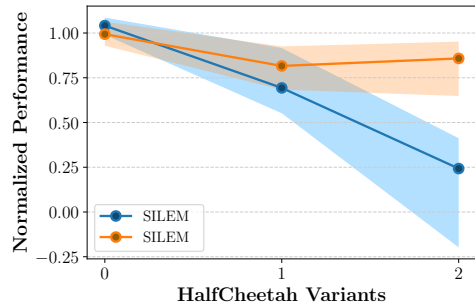
V. EXPERIMENTS AND RESULTS

Our experiments involve a series of HalfCheetah-based bodies (Figure 2a), a series of Ant-based bodies (Figure 2c), and three simulated humanoid agents Humanoid, HumanoidR, and Atlas (Figure 3a). The input to the policy network and Q function consist of the default low level state information (for example joint angles, velocities, etc.).
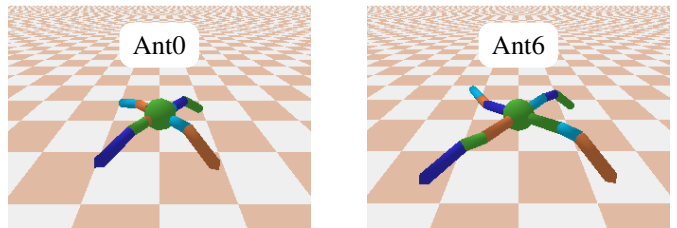
For the HalfCheetah-based and Ant-based bodies, g is simply the identity function, while for the humanoid agents, it is a function that abstracts out the list of features defined in Figures 4a and 4b. In our experiments involving human demonstrations, there exists an asymmetry in the definition of g. For example, due to differences in the skeletal structure between the demonstrator and learner (Figure 3a), it is not possible to use the same function to compute the relative position of the right hand in the demonstrator and learner.

(a) The bodies we design based on HalfCheetah

(b) Results for the HalfCheetah bodies

(c) The bodies we design based on Ant

(d) Results for the Ant bodies

Fig. 2: The toy domains we created to assess SILEM's benefits (**2b, 2d**) Results showing that SILEM is able to prevent degradation of imitation performance due to embodiment mismatch. SILEM$^-$ is an ablation of SILEM with the sequential affine transform removed from the training structure. The y-axis shows performance normalized by body-specific expert-level performance — a score of 1.0 by an algorithm for a body indicates that that algorithm can match the performance of PPO on that body. The plot shows minimum, average, and maximum performance over 5 independent trials.

(a) The simulated humanoids considered in this paper

(b) Results in Atlas, Humanoid, and HumanoidR

Fig. 3: The experiments we performed involving human demonstrations. **(3b)** Results showing that SILEM is able to reliably learn from human demonstrations, while TPIL, Peng20, and our ablation without the sequential affine transform (SILEM$^-$) are unable to do so. The y-axis uses the built-in reward function from the DeepMind Control Suite, which rewards the agent for maintaining an upright posture, matching the hum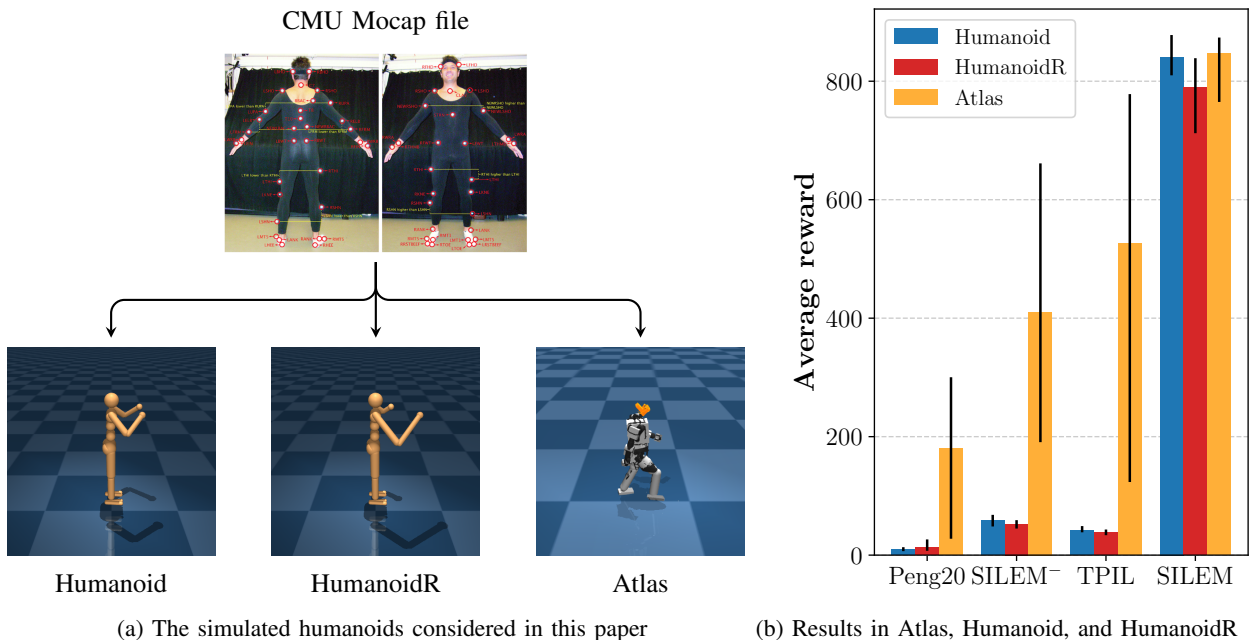an's speed, and minimizing energy expenditure. The maximum reward possible is 1000. The plot shows minimum, average, and maximum performance over 5 independent trials.

Thus, we use different g functions for the demonstrator and learner (that eventually end up computing the same quantity).

We generated the three HalfCheetah-based bodies HC0, HC1, and HC2 by setting the torso length at 1, 2, and 3 respectively. HC0 is the original HalfCheetah provided by PyBullet. We then train HC0, HC1, and HC2 to imitate an expert HC0 that is running as fast as possible. We generated the Ant-based bodies by increasing the length of the link closest to the original Ant's body in steps of 0.05 from 0.2 to 0.5. This resulted in a total of 7 Ant-based bodies, from Ant0 to Ant6, where Ant0 is the original Ant provided by PyBullet. We then train all the Ant-based bodies to imitate an expert Ant0 that is running as fast as possible. The first humanoid agent, Humanoid, is the simple humanoid from the DeepMind Control Suite [20]. The second humanoid, HumanoidR, is an asymmetric humanoid that we generated by elongating Humanoid's right arm and shortening its left arm. The third humanoid is a simulated version of Atlas, a robot from Boston Dynamics. The humanoid agents are trained to imitate the first demonstration from Subject 8 in the CMU Mocap Library (mocap.cs.cmu.edu). The demonstration is of a human walking forward. Note that we do not train HumanoidR to imitate Humanoid. They both, along with the simulated Atlas, are trained to imitate the human demonstration.

Can the sequential affine transform address embodiment mismatch? To answer this question, we first create an ablation of SILEM, SILEM$^-$ (Algorithm 1), where the sequential affine transform is removed from the training structure. We then apply SILEM$^-$ and SILEM to the toy domains we designed and to train the three humanoids (Humanoid, HumanoidR, and Atlas). Our results in Figure 2 and 3b and the attached supplementary video shows that SILEM is able to consistently generate stable gaits for all of the agents we test, while SILEM$^-$ fails to do so.

As further evidence that the sequential affine transform addresses embodiment mismatch, we plot the affine transforms used by the best performing policies of Humanoid and HumanoidR in Figure 4a and 4b. The results show that the sequential affine transforms apply targeted compensation for the asymmetric arms, thus providing evidence that SILEM works by correcting embodiment mismatch.

Comparison with TPIL and Peng20 on training humanoid agents from human demonstrations: Note that we refer to our implementation of Peng et al. [15] as Peng20. We compare SILEM with Peng20 since our application of the sequential affine transform for humanoid agents is similar to their application of inverse kinematics for dog-like agents. Peng20 matches end effectors between demonstrator and learner, while we also additionally match the elbow and knee angles. We also compare SILEM to TPIL since we found TPIL to be the closest available alternative to SILEM — like SILEM, TPIL is an AIL algorithm that addresses mismatch between expert and learner in an online fashion.

Our quantitative results in Figure 3b and our qualitative results in the attached supplementary video show that SILEM outperforms both Peng20 and TPIL by a large margin at the task of imitating the human demonstration. Surprisingly,

(a) Values in the diagonal matrix, **A**
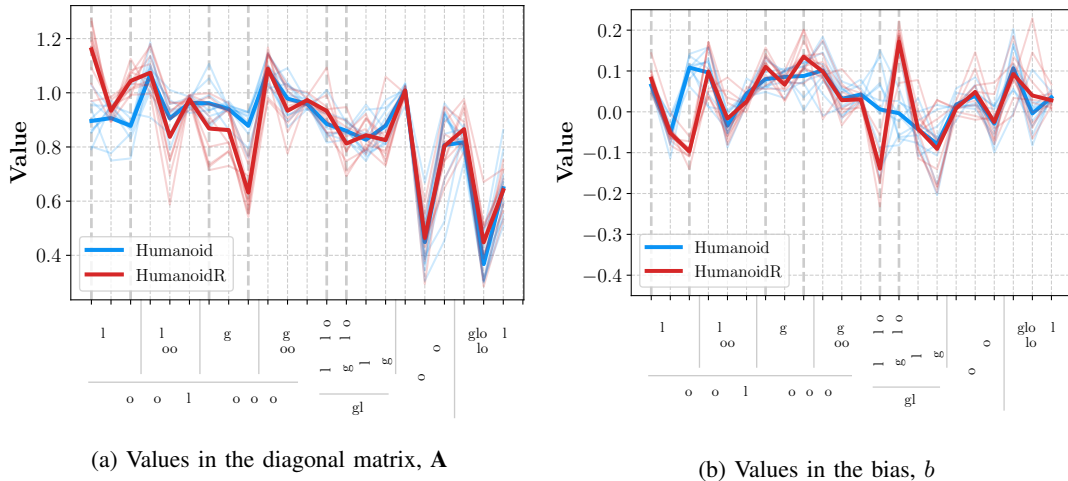

(b) Values in the bias, b

Fig. 4: The values of the affine transform from the top 10 policies in both Humanoid and HumanoidR. Thicker plotlines show the mean. The relevant features have been highlighted with thicker vertical gridlines. Note that positive x faces forward from the agent, and positive z points upwards. (**4a**) As expected, the relative position of the left and right hands get scaled up and down respectively (the hands' z-values are always negative). (**4b**) The left arm maintains a more obtuse angle than the right arm in order to match the counterbalance provided by the right arm (see supplementary video). The sequential affine transform allows for this adaptive correction by modifying the left and right elbow angles accordingly.

Peng20, TPIL, and SILEM$^-$ perform much better on Atlas than they do on the other humanoid agents. We posit that this is because Atlas is closer in dimensions to the demonstrator than the other humanoid agents. The feet are 1.13, 1.26, and 0.99 units away from the torso for the demonstration, Humanoid, and Atlas respectively. However, the hands are 0.74, 0.4 and 0.91 units away from the torso for the demonstration, Humanoid, and Atlas respectively.

Why does the sequential affine transform not interfere with the policy's learning? The objective function used to learn the sequential affine transform has the potential to conflict with the policy's learning objective — the sequential affine transform could, in theory, generate skeletal features corresponding to good expert states from skeletal features corresponding to poor learner states. We hypothesize that such a scenario does not arise because the sequential affine transform consists of affine transforms and not more powerful deep networks such as MLPs. Our experiments support the claim (Table I). For context, the average ground truth reward obtained by a random policy and expert is -1555 and 2249 respectively. On the other hand, the discriminator reward can vary from 0 to 1000. Using an MLP (1 hidden layer with 100 units) results in high discriminator reward, but poor performance by the policy. This finding suggests that the MLP is crafting skeletal features corresponding to good expert states from skeletal features corresponding to poor learner states.

VI. CONCLUSION

In this paper, we have introduced a new AIL algorithm called SILEM (**S**keletal feature compensation for **I**mitation **L**earning with **E**mbodiment **M**ismatch). Using SILEM, we showed improved performance in the challenging problem of

TABLE I: Results from applying SILEM to HC0 where f in the sequential affine transform is either an MLP or an Affine Transform. These results suggest a degeneracy in the MLP not present with the affine transform — the MLP is both compensating for embodiment mismatch and imperfect imitation. All results are the mean over 5 independent trials.

f	Discriminator Reward	Ground Truth Reward
MLP	466	-1979
Affine Transform	451	2234

learning from human demonstrations. We presented evidence that SILEM works by learning a sequential affine transform capable of compensating for differences in the skeletal features of the expert and learner that arise due to embodiment mismatch.

We hypothesize that SILEM is also capable of handling simple cases of *environment* mismatch. E.g., the human walking demonstration is from a flat surface, but the humanoid agent is learning to walk on an inclined surface. By scaling and shifting the skeletal features appropriately, it might be possible to make the walking behavior on an inclined surface resemble that on a flat surface. An interesting line of future work would be to exhaustively evaluate the capabilities of SILEM with regards to environment mismatch. This could be followed by trying to successfully replace the sequential affine transform with a more sophisticated architecture, such as an ensemble of affine transforms, thus opening the door for SILEM to tackle ever more complex instances of embodiment and environment mismatch.

References

[1] Ariki, Y., Matsubara, T., Hyon, S.H.: Latent kullback-leibler control for dynamic imitation learning of whole-body behaviors in humanoid robots. In: 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids). pp. 946–951 (2016).

[2] Christiano, P.F., Leike, J., Brown, T., Martic, M., Legg, S., Amodei, D.: Deep reinforcement learning from human preferences. In: Advances in Neural Information Processing Systems. pp. 4299–4307 (2017)

[3] Gamrian, S., Goldberg, Y.: Transfer learning for related reinforcement learning tasks via image-to-image translation. CoRR **abs/1806.07377** (2018), http://arxiv.org/abs/1806.07377

[4] Gangwani, T., Peng, J.: State-only imitation with transition dynamics mismatch. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=HJgLLyrYwB

[5] Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by back-propagation. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. p. 1180?1189. ICML?15, JMLR.org (2015)

[6] Ghadirzadeh, A., Chen, X., Poklukar, P., Finn, C., Björkman, M., Kragic, D.: Bayesian meta-learning for few-shot policy adaptation across robotic platforms. CoRR **abs/2103.03697** (2021), https://arxiv.org/abs/2103.03697

[7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)

[8] Gupta, A., Devin, C., Liu, Y., Abbeel, P., Levine, S.: Learning invariant feature spaces to transfer skills with reinforcement learning. arXiv preprint arXiv:1703.02949 (2017)

[9] Ho, J., Ermon, S.: Generative adversarial imitation learning. In: Advances in neural information processing systems. pp. 4565–4573 (2016)

[10] Kim, K., Gu, Y., Song, J., Zhao, S., Ermon, S.: Domain adaptive imitation learning. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 5286–5295. PMLR (13–18 Jul 2020), http://proceedings.mlr.press/v119/kim20c.html

[11] Liu, F., Ling, Z., Mu, T., Su, H.: State alignment-based imitation learning. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=rylrdxHFDr

[12] Merel, J., Tassa, Y., Srinivasan, S., Lemmon, J., Wang, Z., Wayne, G., Heess, N.: Learning human behaviors from motion capture by adversarial imitation. arXiv preprint arXiv:1707.02201 (2017)

[13] Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)

[14] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)

[15] Peng, X., Coumans, E., Zhang, T., Lee, T.W.E., Tan, J., Levine, S.: Learning agile robotic locomotion skills by imitating animals. In: Robotics: Science and Systems (07 2020).

[16] Peng, X.B., Kanazawa, A., Malik, J., Abbeel, P., Levine, S.: Sfv: Reinforcement learning of physical skills from videos. In: SIGGRAPH Asia 2018 Technical Papers. p. 178. ACM (2018)

[17] Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., Brain, G.: Time-contrastive networks: Self-supervised learning from video. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 1134–1141. IEEE (2018)

[18] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815 (2017)

[19] Stadie, B.C., Abbeel, P., Sutskever, I.: Third-person imitation learning (2017), https://arxiv.org/pdf/1703.01703.pdf

[20] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D.d.L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al.: Deepmind control suite. arXiv preprint arXiv:1801.00690 (2018)

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

[22] Yu, T., Finn, C., Dasari, S., Xie, A., Zhang, T., Abbeel, P., Levine, S.: One-shot imitation from observing humans via domain-adaptive meta-learning. In: Kress-Gazit, H., Srinivasa, S.S., Howard, T., Atanasov, N. (eds.) Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018 (2018)., http://www.roboticsproceedings.org/rss14/p02.html

Skeletal feature compensation for Imitation Learning with Embodiment Mismatch: Supplementary material

1 Hyper-parameter search

For every experiment, we used a grid search to find the best set of hyper-parameters. Specifically, for every configuration of hyper-parameters in the experiment, we ran 5 independent trials. Within each trial, we trained a policy, measured the average reward obtained by that policy over 1000 episodes, and assigned the resulting number as the score for that particular trial. The score for each configuration of hyper-parameters is the average score over the 5 trials corresponding to that configuration. The best configuration of hyper-parameters is then that which maximizes this score.

All instantiations of SILEM, SILEM $^-$, and TPIL, use $n = 4$. I.e., the input to the discriminator is a tuple of size 4.

1.1 General settings for PPO

Throughout this work, the learning rate in PPO is always linearly decayed to 0. We use the Adam optimizer with the default options from OpenAI baselines. The other settings in Table 2 and Table 1 are also used throughout this work unless specified otherwise. Note that both SILEM and SILEM$^-$ use PPO to improve the policy, but instead of the reward function from the environment, they utilize the output of the discriminator as the reward function. Contrary to Table 2, the Ant experts were trained for 1000 PPO iterations.

Table 1: Common hyper-parameters used for PPO in experiments involving Atlas, Humanoid and HumanoidR (The Mujoco environment).

HYPER-PARAMETER	VALUES
NO. TIMESTEPS USED FOR EACH ITERATION	4096
PPO CLIP PARAMETER	0.1
DISCOUNT FACTOR (γ)	0.99
LAMBDA (λ)	0.95
PPO MINIBATCH SIZE	256
NO. PPO ITERATIONS	8000
LEARNING RATE	1E-4

Table 2: Common hyper-parameters used for PPO in experiments involving Ant and HalfCheetah (The Pybullet environments).

HYPER-PARAMETER	VALUES
NO. TIMESTEPS USED FOR EACH ITERATION	2048
PPO CLIP PARAMETER	0.2
DISCOUNT FACTOR (γ)	0.99
LAMBDA (λ)	0.95
NO. PPO EPOCHS	10
PPO MINIBATCH SIZE	64
NO. PPO ITERATIONS	2000
LEARNING RATE	2E-4

1.2 SILEM and SILEM$^-$ for the humanoid agents (Atlas, Humanoid and HumanoidR)

The SILEM experiments involving humanoid agents used L2 normalization for the state transformer's weights. None of our other SILEM experiments used L2 normalization. The diagonal matrix and bias values of the state transformer were normalized with different coefficients.

Table 3: The grid search for SILEM and SILEM$^-$ on Humanoid and HumanoidR was conducted by evaluating all permutations of the following parameter values. Hyper-parameters in bold only apply to SILEM.

HYPER-PARAMETER	VALUES
(N_D) NO. UPDATE STEPS FOR THE DISCRIMINATOR	1, 2
(N_G) **NO. UPDATE STEPS FOR THE STATE TRANSFORMER**	1
(d) **LEARNING RATE DECAY FOR THE STATE TRANSFORMER**	0.996, 1
(w_d) **L2 COEFFICIENT FOR DIAGONAL MATRIX**	1, 10
(w_b) **L2 COEFFICIENT FOR BIAS VALUES**	1, 10
(N_P) NO. PPO EPOCHS	4, 7

Table 4: The combination of parameters that turned out best for each experiment involving SILEM and Humanoid and HumanoidR.

HUMANOID AGENT	N_D	d	w_d	w_b	N_P
HUMANOID	2	1	10	10	7
HUMANOIDR	1	1	1	10	7

Table 5: The combination of parameters that turned out best for each experiment involving SILEM$^-$ and Humanoid and HumanoidR.

HUMANOID AGENT	N_D	N_P
HUMANOID	1	7
HUMANOIDR	1	7

Table 6: The grid search for SILEM and SILEM^{-} on Atlas was conducted by evaluating all permutations of the following parameter values. Hyper-parameters in bold only apply to SILEM.

HYPER-PARAMETER	VALUES
(N_D) No. update steps for the discriminator	1, 2
(N_G) **No. update steps for the state transformer**	2, 3
(w_d) **L2 coefficient for diagonal matrix**	1, 10
(w_b) **L2 coefficient for bias values**	1, 10
(N_P) No. PPO epochs	5, 8, 12
(a) Ankle's range of motion	$\pm0.3, \pm0.1$

Table 7: The combination of parameters that turned out best for each experiment involving SILEM and Atlas.

HYPER-PARAMETER	VALUES
(N_D) No. update steps for the discriminator	2
(N_G) No. update steps for the state transformer	2
(w_d) L2 coefficient for diagonal matrix	10
(w_b) L2 coefficient for bias values	10
(N_P) No. PPO epochs	8
(a) Ankle's range of motion	±0.1

Table 8: The combination of parameters that turned out best for each experiment involving SILEM^{-} and Atlas.

HYPER-PARAMETER	VALUES
(N_D) No. update steps for the discriminator	1
(N_P) No. PPO epochs	8
(a) Ankle's range of motion	±0.1

1.3 TPIL (Third Person Imitation Learning) for Atlas, Humanoid and HumanoidR

Table 9: The grid search for TPIL on Humanoid and HumanoidR was conducted by evaluating all permutations of the following parameter values.

HYPER-PARAMETER	VALUES
(C_g) Gradient reversal coefficient	0.1, 0.3, 0.6, 1, 1.3, 1.6
(N_D) No. update steps for the discriminator	1, 2, 3
(N_P) No. PPO epochs	4, 5, 7

Table 10: The combination of parameters that turned out best for each experiment involving TPIL and Humanoid and HumanoidR.

HUMANOID AGENT	C_g	N_D	N_P
HUMANOID	0.6	1	5
HUMANOIDR	0.3	1	4

Table 11: The grid search for TPIL on Atlas was conducted by evaluating all permutations of the following parameter values.

HYPER-PARAMETER	VALUES
(C_g) GRADIENT REVERSAL COEFFICIENT	0.1, 0.3, 0.6, 1, 1.3, 1.6
(N_D) NO. UPDATE STEPS FOR THE DISCRIMINATOR	1, 2
(N_P) NO. PPO EPOCHS	5, 8, 12
(a) ANKLE'S RANGE OF MOTION	±0.1, ±0.3

Table 12: The combination of parameters that turned out best for each experiment involving TPIL and Atlas.

HYPER-PARAMETER	VALUES
(C_g) GRADIENT REVERSAL COEFFICIENT	1.3
(N_D) NO. UPDATE STEPS FOR THE DISCRIMINATOR	1
(N_P) NO. PPO EPOCHS	8
(a) ANKLE'S RANGE OF MOTION	±0.1

1.4 Peng20 for Atlas, Humanoid and HumanoidR

We first used `ik_humanoid.py` to convert the demonstration trajectory to Humanoid's and HumanoidR's embodiment. Then we used `ik_atlas.py`, to get the demonstration trajectory in Atlas' embodiment. Both the aforementioned python files can be found in the source code. With the demonstration trajectory in the correct embodiments, we now apply GAIfO/SILEM$^-$ to train the respective humanoid agents. We use the same grid search options as in Table **??** and **??**.

Table 13: The combination of parameters that turned out best for each experiment involving Peng20 and Atlas.

HYPER-PARAMETER	VALUES
(N_D) NO. UPDATE STEPS FOR THE DISCRIMINATOR	1
(N_P) NO. PPO EPOCHS	12
(a) ANKLE'S RANGE OF MOTION	±0.3

Table 14: The combination of parameters that turned out best for each experiment involving Peng20 and Humanoid and HumanoidR.

HUMANOID AGENT	N_D	N_P
HUMANOID	1	7
HUMANOIDR	1	4

1.5 SILEM and SILEM$^-$ for the HalfCheetah-based bodies

Table 15: The grid search for SILEM and SILEM$^-$ was conducted by evaluating all permutations of the following parameter values. Hyper-parameters in bold only apply to SILEM.

HYPER-PARAMETER	VALUES
(n) NO. EXPERT TRAJECTORIES	10, 20
(N_D) NO. UPDATE STEPS FOR THE DISCRIMINATOR	5, 10
(N_G) **NO. UPDATE STEPS FOR THE STATE TRANSFORMER**	5, 10
(d) **LEARNING RATE DECAY FOR THE STATE TRANSFORMER**	0.999, 1
(N_P) NO. PPO EPOCHS	5, 10
(N_{Pmb}) PPO MINIBATCH SIZE	64, 128

Table 16: The combination of parameters that turned out best for each experiment involving SILEM and a HalfCheetah-based body. We only ran SILEM using the HalfCheetah0 expert.

HALFCHEETAH-BASED BODY	n	N_D	N_G	d	N_P	N_{Pmb}
HALFCHEETAH0	20	10	5	0.999	10	64
HALFCHEETAH1	20	10	5	0.999	10	64
HALFCHEETAH2	10	10	10	1	10	64

Table 17: The combination of parameters that turned out best for each experiment involving SILEM$^-$ and a HalfCheetah-based body, when the HalfCheetah0 expert is used.

HALFCHEETAH-BASED BODY	n	N_D	N_P	N_{Pmb}
HALFCHEETAH0	20	10	10	64
HALFCHEETAH1	20	5	10	256
HALFCHEETAH2	20	5	5	64

Table 18: The combination of parameters that turned out best for each experiment involving SILEM$^-$ and a HalfCheetah-based body, when the learner and expert have the same HalfCheetah-based body.

HALFCHEETAH-BASED BODY	n	N_D	N_P	N_{Pmb}
HALFCHEETAH0	10	10	10	128
HALFCHEETAH1	20	5	5	64
HALFCHEETAH2	20	5	10	128

1.6 SILEM and SILEM$^-$ for the Ant-based bodies

Table 19: The grid search for SILEM and SILEM$^-$ was conducted by evaluating all permutations of the following parameter values. Hyper-parameters in bold only apply to SILEM.

HYPER-PARAMETER	VALUES
(n) NO. EXPERT TRAJECTORIES	10, 20
(N_D) NO. UPDATE STEPS FOR THE DISCRIMINATOR	5, 10
(N_G) **NO. UPDATE STEPS FOR THE STATE TRANSFORMER**	5, 10
(d) **LEARNING RATE DECAY FOR THE STATE TRANSFORMER**	0.994, 0.996, 0.998, 0.999, 1
(N_P) NO. PPO EPOCHS	5, 10
(N_{Pmb}) PPO MINIBATCH SIZE	64, 128, 256

Table 20: The combination of parameters that turned out best for each experiment involving SILEM and an Ant-based body. We only ran SILEM using the Ant0 expert.

ANT-BASED BODY	n	N_D	N_G	d	N_P	N_{Pmb}
ANT0	20	10	10	1	10	128
ANT1	10	10	5	1	10	128
ANT2	20	10	10	1	10	128
ANT3	10	5	5	1	10	128
ANT4	10	10	10	0.999	10	64
ANT5	10	5	5	1	10	128
ANT6	20	5	5	1	10	256

Table 21: The combination of parameters that turned out best for each experiment involving SILEM$^-$ and an Ant-based body, when the Ant0 expert is used.

ANT-BASED BODY	n	N_D	N_P	N_{Pmb}
ANT0	10	10	10	256
ANT1	10	10	10	128
ANT2	20	10	5	64
ANT3	20	10	10	256
ANT4	20	5	10	256
ANT5	20	5	5	128
ANT6	10	5	5	256

Table 22: The combination of parameters that turned out best for each experiment involving SILEM$^-$ and an Ant-based body, when the learner and expert have the same Ant-based body.

ANT-BASED BODY	n	N_D	N_P	N_{Pmb}
ANT0	20	10	10	128
ANT1	10	10	10	256
ANT2	20	10	10	128
ANT3	10	10	5	128
ANT4	20	10	5	64
ANT5	10	10	10	64
ANT6	10	10	10	256

2 Neural Architectures

All of the neural networks used in this work are multi-layer perceptrons (MLPs) made up of tanh nonlinearities. For the experiments involving humanoid agents, the discriminators had 2 hidden layers with 128 units each while the policies had 2 hidden layers with 256 units each. For all other experiments (involving the Pybullet environments), the discriminators had 3 hidden layers with 128 units each, while the policies had 2 layers with 64 units each.

3 Computing Infrastructure and Run-times

We run our experiments on the CPUs in a computing cluster managed by HTCondor. Each node in the cluster had approximately the following server model with minor variations: Dell PowerEdge M620, and processor: 2x Xeon E5-2670 (8 core) @ 2.60GHz. Each experimental run of SILEM takes approximately 2 days in this setup. Running the grid searches for the Ant-based bodies took approximately a week, while for the humanoid agents, it took around 2-4 days. Note that this timing is dependent on the load of the cluster during job submission. Our source code is based on Open AI baselines and the DeepMind Control Suite (for the humanoid agents), and used the following GitHub repo — `github.com/ywchao/merel-mocap-gail` — as a starting point. Our source code is attached to the supplementary materials. While SILEM incurs additional computational complexity due to the training steps for the state transformer (as compared to SILEM$^-$), we found that it did not affect training time much. The computational cost of simulating the policy's interaction with the environment is overwhelmingly higher than the other steps in an iteration of SILEM.