Learning from Different Expert Agents

Joanne Truong* and Joel Ye* and Naoki Yokoyama*

***Abstract*— While several works on visual navigation have shown success through using deep reinforcement learning to train agents modeled with simpler dynamics, training robots with more complex dynamics proves to be a difficult and open problem. Here we study how to leverage more easily trained robots (*e.g.*, a cylindrical robot with a discrete action space) with simple dynamics models to teach more complex robots with a different dynamics model (*i.e.*, continuous action space). Such transfer must overcome the correspondence problem in differing robot capabilities. We first study demonstrate how the teacher robot can be leveraged as an onboard expert, using a heuristic mapping between robot action spaces. Then, we consider how to learn a mapping which overcomes the correspondence problem. To this end, we propose an alignment module which can be jointly learned in the inverse reinforcement learning setting, which addresses dynamics mismatch by transforming a sequence of teacher states into corresponding student states.**

I. Introduction

How can we train a robot to navigate around a house? Such a problem is many layers of realism harder than teaching a simple agent in a toy grid-world environment. However, advances in embodied AI have allowed us to tackle several of these problems: training in photorealistic environments [1], [2], incorporating realistic sensor noise and deploying such robots to the real world [3], [4]. One prominent remaining issue is that these works have assumed a robot with a limited discrete action space (move-forward 0.25m, pivot $\pm 10°$), resulting in point-turn dynamics, in which the robot must come to a full halt to turn. Instead, we propose to lift the action space to continuous actions, which better exemplifies the dynamics model of popular mobile robotics platforms (e.g., LoCoBot [5], TurtleBot [6], Fetch [7], etc.).

However, a continuous action space is considerably harder to learn [8], and learning this control in addition to the complex navigation task further complicates learning. To make learning easier, one potential approach is to transfer the knowledge learned from a teacher with a simpler morphology to a student robot with a more complex morphology. In this work, we study two aspects of this approach. First, if we can leverage domain knowledge to directly map teacher behavior to student behavior, how much do student policies benefit? Second, if we cannot manually connect teacher and student, how can we learn leverage expert knowledge?

To make learning a policy with a continuous action space easier, in this work, we investigate how the actions of a discrete action agent can be used to guide a student agent that utilizes a continuous action space. We use a behavioral cloning approach that is inspired by DAgger [9], in which the student learns navigation behavior from the teacher using supervised learning while also interacting with the environment.

For the more general case, we cannot directly transfer robot policies given semantic mismatches between robot action spaces. To overcome this, Hejna *et al*. [10] proposes to learn a hierarchical policy, and directly transfers the teacher's high-level policy while aligning the student's low level policy to match the teacher's with an adversarial discriminator. However, this discriminator assumes that different robots have comparable transitions in task-relevant states, *e.g.*, the same movement speed for locomotion. This is an unrealistic assumption for many cases of robot-to-robot transfer; for instance, if one robot has a more powerful motor or longer legs than another, it can span the same distance in a shorter time frame.

To relax this assumption, we propose to use an alignment module which is jointly trained with the student policy. The alignment module is responsible for converting a student's sequence of states to a new sequence which more closely resembles the teacher's state sequence. With this module, we enable a student with a vastly different morphology to learn from the teacher's demonstrated behavior while not being constrained to identically replicate the teacher's pace.

II. Related Work

A. Robot-to-robot transfer

Several other works also involve distilling the knowledge of an expert policy to a student policy. In the work by Chen et al. [11], a teacher agent trained with privileged information is used to label the actions of a student agent as it trained. The student however, only has access to a subset of the observations that were available to the teacher. In contrast to this work, both our teacher and student policies receive the same set of observations, but have different action spaces. The approach that we take is most similar to the the work done by OpenAI for robust manipulation of a Rubik's Cube [12], in which a DAgger-like approach is also used to help initialize new policies that attain levels of performance very close to the teacher policy, before being fine-tuned with RL. This is the same pipeline that we adopt for policy distillation, with a focus on changing the modality of the action distribution from a discrete space to a continuous space.

B. Adversarial Alignment

The problem we study is how a student policy π_S can learn from a teacher policy π_T despite having different mor-

*All authors contributed equally. Emails: truong.j@gatech.edu joelye9@gmail.com nyokoyama@gatech.edu

phologies and action spaces A_S, A_T. Specifically, we aim to enable such transfer when the distinct action spaces provide the robots with very different capabilities. For example, if A_S consists of actions of discrete 1 meter steps, but A_T only contains discrete 0.25 meter steps, we would like the student to still be able to learn from the teacher's demonstrations.

A core LfD method for aligning student and teacher behavior is adversarial alignment, *e.g*. AIRL from Fu et al. in [13]. In this approach, a discriminator must distinguish whether a given state sequence comes from π_S or π_T. In both the original formulation and its adaptation to sequences by Hejna et al. [10], these state sequences are assumed to be of the same length and spanning the same time frame, simplifying direct comparison between the teacher and the student's actions. A similar assumption has seen success in unsupervised robot learning from human demonstrations, as in AVID [14], *i.e*. that the human moves with the same action constraints as the robot. However, agent constraints (from different morphologies, hardware capabilities) may prevent the student from directly mimicking the teacher. In the case of different agent velocities, the AIRL discriminator could quickly distinguish student and teacher trajectories by measuring distances between states, and thus would not provide a good learning signal.

To resolve this, we propose to learn a sequence to sequence alignment module which transforms teacher states $s_1^T \dots s_{t_T}^T$ to student states $s_1^S \dots s_{t_S}^S$. This is a domain translation task, and can be inspired by works in *e.g*. image translation. A critical design choice to make is whether this alignment is paired or not. Provided arbitrary teacher demonstrations and student attempts and thus no correspondence between the two domains, we can use unpaired alignment as in CycleGAN [15]. On the other hand, our problem setting allows us to simulate teacher demonstrations for any given student episode. This is reminiscent of the one-shot imitation learning setting [16], where we have a student trajectory which can be used to "cue" a reference demonstration for every episode.

III. Experimental Setup

In this section, we recap the Point Goal Navigation [17] task, and provide details on our agents in simulation.

A. Task: PointGoal Navigation

We study this problem for the task of photorealistic Point Goal Navigation (PointNav). In PointNav, the robot is initialized in a previously unseen environment at a random starting location and must navigate to a goal location specified in relative coordinates. An episode is considered successful if the robot navigates within 0.2m (the radius of the LoCoBot) of the goal location. The robot is given as input a depth image and an egomotion sensor which localizes the robot relative to its spawn location. The robot is not provided a map of the environment. The sensory suites will be held constant across robots. Our teacher policy is an idealized cylindrical robot which takes discrete actions consisting of `turn-left 10°`, `turn-right 10°`, `forward 0.25m`. We use the pre-trained policy provided by [1], which has been trained for 2.5 billion simulator frames in the Habitat simulator [18]. We consider two student robots: (1) a robot which has a continuous action space; and (2) a robot which takes discrete actions consisting of `turn-left 5°`, `turn-right 5°`, `forward 0.15m`.

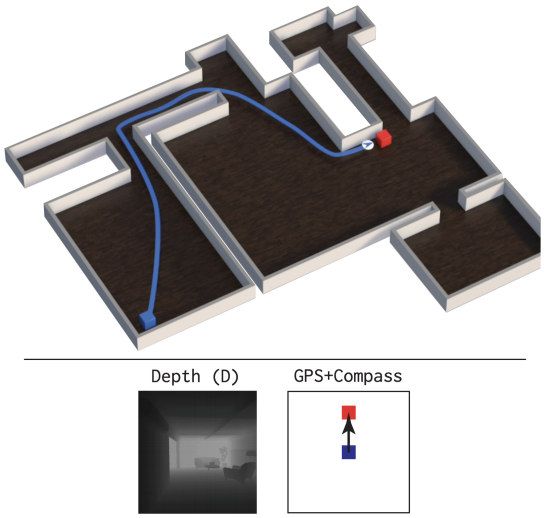

Fig. 1: We consider the task of PointGoal Navigation, in which an agent must navigate from a random starting location (blue) to a target location (red). The agent is placed in a previously unseen environment, and does *not* have access to a map.

B. Evaluation Metrics

To evaluate the navigation performance of our agents, we use success rate and Success weighted by Path Length (SPL) [17]. SPL is similar to success rate, except that it is lower if the agent travels a longer distance than the geodesic distance from the start to the goal point. It is defined as follows:

$$\text{SPL} = S\frac{L}{\max(P, L)} \tag{1}$$

where S is 0 or 1 depending on whether the agent successfully completed the episode, P is the length of the agent's path, and L is the length of the shortest path from the start point to the goal point.

However, in some applications it may be preferable to follow not the shortest path to the goal. Instead it may be preferable to follow the fastest path, which may not always the same (Figure 2). Due to the nature of the discrete action space used by Wjimans et al., their agent is only able to move in a point-turn manner. In contrast, the continuous action space of the student agent allows it to make smoother turns when navigating. In other words, it can move forward and turn simultaneously, allowing it to round corners faster than at the cost of traveling a slightly longer distance.

Thus, we also consider Success weighted by Completion Time (SCT) [19] as a metric, which is lower if the agent takes longer to complete the episode (irrespective of path length) compared to the fastest possible completion time. To approximate the fastest completion time, we multiply the

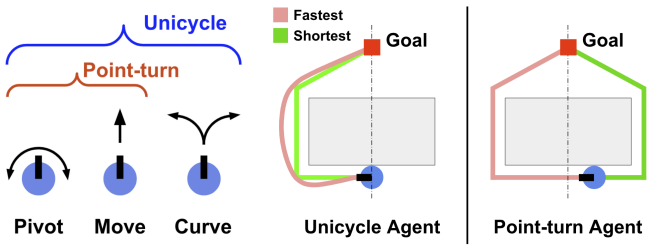

Fig. 2: The shortest path is not always the fastest. *Left:* In addition to pivoting and moving straight like a point-turn model, a unicycle-cart dynamics model can move forward and turn (curve) simultaneously. *Right:* A path faster than the shortest path may exist for both types of dynamics. The fastest path depends on the agent's maximum linear and angular velocities, and initial heading.

length of the shortest path by the agent's maximum linear velocity. As this time does not consider the amount of time necessary to make a turn (angular velocity), it is an upper bound. SCT is defined by

$$\text{SCT} = S\frac{T}{\max(C,T)} \tag{2}$$

where C is the agent's completion time, and T is the fastest completion time to reach the goal point from the start point while circumventing obstacles based on the agent's dynamics.

C. Action Space

We consider both discrete and continuous action spaces for our robots. The discrete action space consists of `turn-left 10°`, `turn-right 10°`, `forward 0.15m or 0.25m`, and `STOP` actions, as used by Savva et al. [18] and Wijmans et al. [1]. The continuous action space for LoCoBot contains center of mass (CoM) linear (foward) and angular (yaw) velocities (v_x, ω_t). We limit the linear velocity to be between 0 and 0.25 m/s, and the angular velocity to be between -10°/s and 10°/s. Note that since LoCoBot is non-holonomic, it does not have access to `strafe-left`, `strafe-right`, or left and right velocity (v_y) actions. We constrain the linear velocity to be non-negative (forward only), as we have found that allowing the full velocity range (forward and backward) results in the robot excessively moving backwards. This hurts the performance of the robot, as it is moving without using the visual input from its forward-facing camera which leads to extensive collisions with the environment.

D. Policy Architecture

We use the same architecture described in [1] for our student agents. The architecture has a visual encoder and a policy. The visual encoder is a convolutional neural network based on ResNet-50 [20], and takes the depth image as input. The policy is a 2-layer LSTM recurrent neural network with a 512-dimensional hidden state, taking the visual encoder's features, the relative distance and heading to the goal, the previous action, and the previous hidden state as input (Figure 3). The goal distance and heading are represented

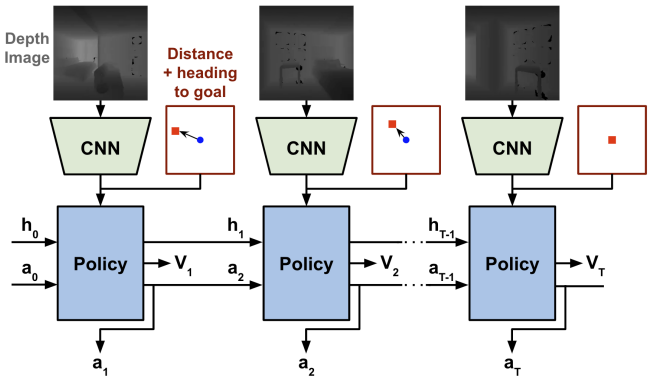

Fig. 3: Network architecture for our student agents, comprised of a convolutional encoder and a recurrent policy. The agent receives an egocentric depth image, its current distance and heading relative to the goal point, and its previous action at each time step. It outputs an action and an estimate of the value function.

as $[r,\ cos(\theta),\ sin(\theta)]$ to avoid the discontinuity at 180°, and the previous action is represented as a pair of values that represent linear and angular velocity. The final layer of the policy 4 outputs, which parameterizes a Gaussian action distribution (two means and two variances) from which to sample a linear and angular velocity from. The policy also has a critic head that outputs an estimate of the state's value, which we use for fine-tuning via reinforcement learning. The full network architecture is shown in Figure 3.

E. Simulators and Datasets

Fig. 4: Example renders of RGB and Depth vision within the photorealistic Habitat simulator. In addition to rich visual details, Habitat also allows us to provide the agent with a variety of sensors, including a GPS+Compass sensor which localizes the robot relative to its spawn location.

We train our agents in the Habitat [18] simulator, a high-performance photorealistic 3D simulator, using the Gibson-4+ dataset [18], which consists of 86 high quality 3D scans that have been rated 4 or above in quality (using the 0 to 5 quality scale) from the full Gibson dataset [2]. This ensures that the 3D scans used are free from significant reconstruction artifacts such as holes or cracks in floor surfaces. Examples of RGB and Depth images from the simulator are

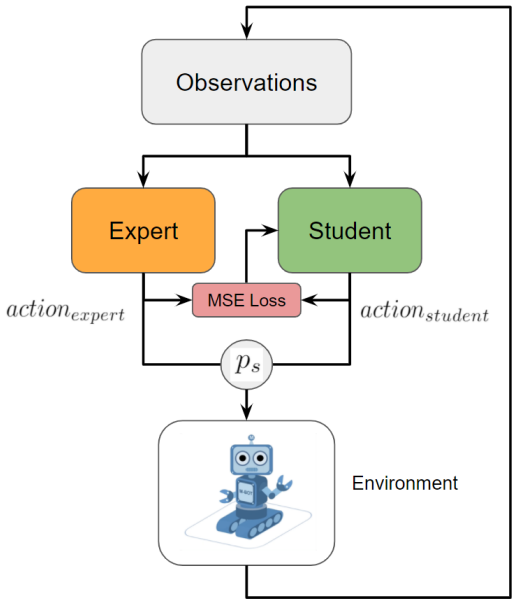

Fig. 5: Our framework for teaching the student agent via behavioral cloning. Both the expert and the student receive the same set of observations (egocentric depth images and egomotion sensor). The student uses the actions of the expert agent as labels, updating its weights via supervised learning. With probability p_s, the environment is stepped using the actions that are outputted by the student agent, otherwise the expert's actions are used.

shown in Figure 4. The dataset includes scans of real-world environments (apartments, offices, houses, etc.), containing furniture (chairs, desks, sofas, tables, etc.). The training split contains over 3M episodes across 72 scenes, and the validation split contains 994 episodes across 14 scenes.

IV. Methods

We study how to leverage different experts from two angles. First, we assume a method of mapping expert to student behavior, and study how expert demonstrations can be leveraged for the Embodied AI setting. Second, we study how to learn such a mapping in a data-driven manner, if it cannot be heuristically or analytically constructed. In the following, we describe the experimental context for our agent training, and then discuss our approach to trajectory alignment.

A. Discrete to Continuous action space

Pre-trained Teacher Agent. For our first set of experiments, our aim is to teach a student agent that utilizes a continuous action space by leveraging a pre-trained expert agent [1] that utilizes a discrete action space as a teacher. The pre-trained expert agent is trained for 2.5 billion simulator frames using 64 GPUs, and reaches near-perfect performance; the agent learns to successfully travel to the goal over 96% of the time in previously unexplored scenes. Furthermore, the expert agent achieves an average SPL score of 0.92 (out of 1.0), which exemplifies how closely it follows the shortest path from the start to the goal (i.e., no detours, wrong turns, etc.).

We adopt a DAgger-like approach [9], in which the student agent learns from both the trajectories of the teacher, as well as the trajectories that it generates itself by interacting with the environment. At every step, the teacher and student both receive the same set of observations, and both output an action to execute. The teacher's actions are used as labels for the student's actions. The teacher's actions are mapped to student labels using the following mapping:

Discrete		*Continuous*
stop	<-> 0 m/s, 0°/s	
forward	<-> 0.25 m/s, 0°/s	
left	<-> 0 m/s, 10°/s	
right	<-> 0 m/s, -10°/s	

For this supervised learning approach, we use Mean Squared Error (MSE) as the loss function for updating weights. To move the robot and step the environment, either the action of the expert or the student is selected. At each time step, the action of the student is selected with probability p_s. We set p_s to begin with a value of 0 at the start of training, and linearly increase it to a value of 1 as training progresses. This translates to letting the teacher assume total control of driving the robot at the beginning of training, with the student slowly gaining control as training progresses.

For this behavioral cloning phase, we use 1 GPU (Nvidia RTX 2080Ti), with 72 parallel environments being stepped concurrently, for about one hour (2 million steps). This is a very modest compute budget compared to the budget used to train the expert agent (64 GPUs, 3 days, 2.5 billion steps). Gradients are accumulated over every `batch_size` steps before updating the weights with backpropagation. We choose a `batch_size` of 4.

In our experiments, we also test training the student with an MSE critic loss, in which the value estimate outputted by the expert agent is used to label value estimates of the state that the student outputs. We did this to see if supervising the critic head of the student boost performance for the later RL fine-tuning step, but found that using the critic loss does not help performance significantly. This implies that it may be possible to teach the student agent using an expert that only outputs actions and not value estimates, such as a classical shortest-path follower.

Oracle Teacher Agent. Following the results of the previous experiment, we investigate using a non-learning approach in lieu of a learned expert agent. Specifically, we use an oracle shortest-path follower to provide actions along the shortest path to the goal, which serve as labels for the student agent. The shortest path follower is a classical approach that uses A* on the map of the environment to compute the shortest path to the goal. Using a shortest path follower obviates the need for a pre-trained expert agent, which may not always be readily available. In addition, since the shortest path follower is a non-learning approach, the scaling of actions can be easily modified to accommodate any limits on linear and angular velocity the student may have, and can be applied

to both discrete and continuous action spaces. This increases the generality of the approach, and allows us to train student agents without the added overhead of pre-training expert agents.

Super-expert performance via RL fine-tuning. We expect that fine-tuning with RL will allow the student agent to learn more diverse motions that allow it to deviate from the simple point-turn dynamics of the discrete action expert and the shortest path follower which also has a point turn dynamics model (Figure 2). This will lead to student agents that can fully exploit their continuous action spaces to learn to reach goals much faster than the original expert agent.

B. Learning an Alignment Module

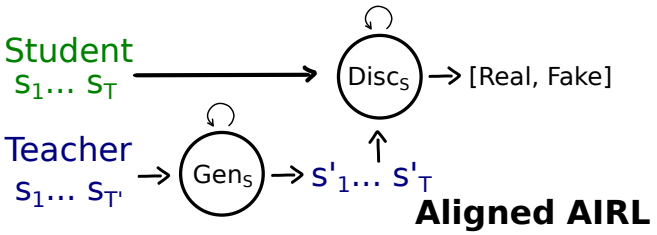

Fig. 6: We propose that the AIRL discriminator can also be used for adversarially training an alignment module which translates teacher to student trajectories.

We reason that an alignment module can take a demonstration's state transitions and output a new trajectory that preserves coarse semantics (*e.g.* moving in the same directions) but adapts the student's low-level transition dynamics (*e.g.* velocity). We hope to learn this translation jointly with AIRL, as displayed in Fig. 6, such that the alignment module and agent both learn to produce expert trajectories with student dynamics. A separate loss should be used to ground alignment module output to its input. How should we design the architecture to accommodate both potential roles? To start with, the AIRL discriminator would typically contrast with full agent states (including egocentric observations), but the alignment module would then have the challenging task of generating photorealistic egocentric observations. One sensible choice, as used in [10], is to simply consider the most task-relevant state, GPS-location and heading. However, while this may be easier for the alignment module, a tuple of two successive locations is not sufficient to distinguish expert from novice behavior in navigation in novel rooms. We hypothesize that AIRL may better identify expert behavior by receiving longer state trajectories. In the extreme, for example, we may translate full expert trajectories (from spawn to goal), and AIRL would judge a student based on their likeness to these full trajectories. This choice is a tradeoff. Longer trajectories provide more context, from which it may be semantically easier to identify expert behavior. However, longer trajectories also make credit assignment harder for the agent, and would task the alignment generator with a difficult sequence to sequence problem. It is worth noting that earlier works such as [10], [14] also perform IRL on sequences but reduce task horizons by operating on subtasks provided by human or hierarchical agent design.

A second design choice to consider is whether alignment output semantics should reflect the teacher or the student. For example, if an immature student explores entirely different rooms than the teacher, it may reduce covariate shift to ask alignment output to reflect student behavior. In the standard domain translation setting, there is no student to match and so alignment output should strongly reflect the teacher's behavior. However, in the simulated robot-to-robot transfer setting, teacher demos are provided for every student episode (and even synthesized on the fly, as studied in the previous onboard expert setting). This exact pairing could potentially provide more precise grounding.

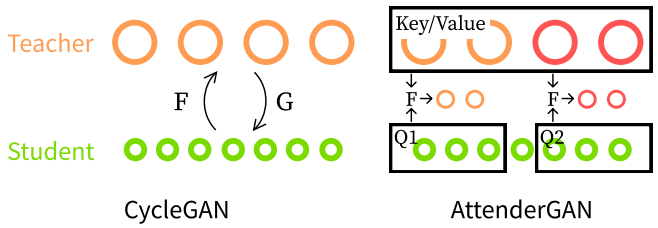

Fig. 7: Two approaches to trajectory alignment. The CycleGAN-based model does sequence encoding-decoding of the full trajectories. AttenderGAN is a novel architecture, which extracts relevant parts of a relevant trajectory.

Considering these two choices, we propose two models for our alignment module, with schematics provided in Fig. 7. The first model is based on CycleGAN [15]; it translates entire teacher trajectories into entire student trajectories, using an RNN encoder-decoder architecture (rather than the CNN-based encoder-decoder used for vision). Alignment output is grounded to the input teacher trajectory through a mean-squared error reconstruction loss. The second model is a novel architecture which we name AttenderGAN, most closely related to attentive architecture in [16]. It enables translation to shorter segments of an episode (CycleGAN cannot produce such segments and fully reconstruct its input). Duan *et al.* [16] uses a single state to attend to the reference demonstration (in this case the teacher trajectory) and extract the relevant behavior; AttenderGAN encodes the cue to attend to the reference and produce a plausible output. Specifically, a context vector is formed from the cue's attention to the reference trajectory; this context vector initializes an RNN generator. To ground the output, we use a contrastive loss inspired by Contrastive Unpaired Translation (CUT) [21]. Grounding towards the student trajectory is achieved by pairing cues with the output it extracts, and negative pairs are formed by other cues to the same reference. Grounding towards the teacher trajectory can use the same approach, but using a cue's attended steps of the teacher demonstration instead of the cue itself. Note that CycleGAN reconstruction loss requires bidirectional translation (two distinct models generate outputs in each of student and teacher domains), while AttenderGAN is one-directional. We omit further architecture details as we do not achieve a

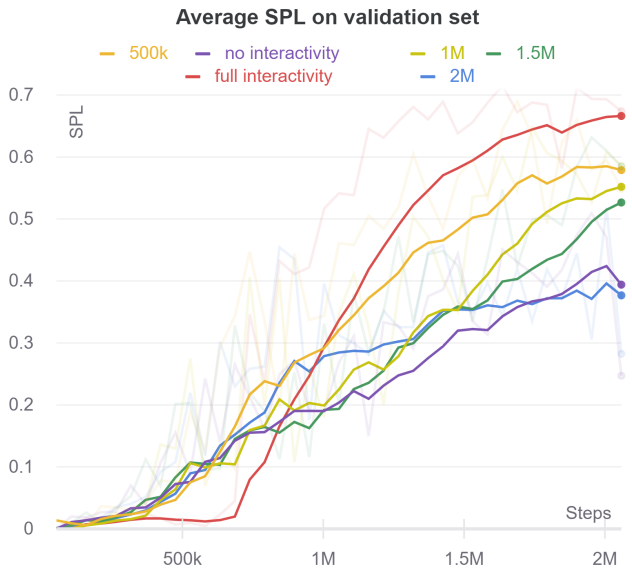

Fig. 8: Average SPL on the validation set. Legend indicates when p_s reaches a value of 1. We find that allowing the student driver to take full control of the robot from the beginning of training to the end yields the best results. Other student agents that allow the teacher to provide trajectories to learn from towards the beginning of training do not converge as quickly in terms of navigation performance.

positive result.

Alignment experimental setup. Before attempting to jointly learn an alignment module in an AIRL setting, we want to validate the difficulty of only domain translation. To test this, we prepare a dataset of 50000 paired (same spawn/goal/scene) trajectories. We compare GPS and heading in states, but pre-encode the heading angle with a sinusoidal embedding. The trajectories are recorded from two agents that were trained entirely with RL. One uses a 0.15 step and the other uses a 0.25m step. The full episodes are on the order of 100 steps; we also evaluate on an abbreviated "short" horizon, which takes the final 15 steps of the 0.15m agent and final 9 steps of 0.25m. Since this is a novel domain translation task, there are no established means of evaluating translation quality. However, we did not find the results qualitatively good enough to warrant defining heuristic evaluation methods.

V. Results

In this section, we aim to address the following questions:

1) Does the behavioral cloning phase significantly accelerate training via reinforcement learning?
2) Can we use a non-learning based agent as an expert to learn from?
3) Can we learn an alignment module to overcome the correspondence problem?

A. Behavioral Cloning Results

Interactivity Study. We test how the amount of interactivity the student is allowed with the environment affects its final performance on the held out validation set of unseen environments. To test various levels of interactivity, we train several variants of the agents that use different schedules for linearly increasing the probability p_s, where p_s starts at 0 and grows to 1 by a certain time step of training.

Accelerating Learning: Pre-trained Teacher. Our results show that with a very modest compute budget, our student agent approaches the asymptotic performance of the expert agent much faster than an agent trained from scratch only using RL with a much more extravagant budget. For the case in which the teacher agent is the learned discrete expert, we found that as the amount of interactivity the student had with environment grew, performance improved (see Figure 8). In fact, full interactivity, with the teacher only being used as a labeler rather than to generate trajectories, yielded the best results in terms of SPL on the validation set.

Accelerating Learning: Oracle Teacher. We also experimented with the amount of interactivity the student had with the environment when trained with an oracle teacher. Upon evaluating variants of the student agents with varying interactivity on the validation set, we found that variant with less interactivity with the environment outperformed agents with high interactivity with the environment (0.259 SPL vs. 0.166 SPL), as shown in Table I. This is the opposite of the conclusion from the previous experiment of learning from the pre-trained teacher agent. We attribute this to the fact that the oracle teacher will almost always attempt to pivot the robot towards the next waypoint along the shortest path to the goal when the students agent's action causes it to stray from the shortest path. On the other hand, the pre-trained teacher has learned that exploration is beneficial, and will allow the student agent to deviate from the shortest path occasionally go down non-optimal paths to scope out rooms and visually verify plausible routes to the goal. Thus, when learning from the oracle teacher, it is best for the teacher to interact with the environment and avoid letting the student derail the robot from the shortest path, letting the student passively learn from the actions that the teacher dictates.

TABLE I: Performance comparison of agents trained with behavioral cloning using an oracle teacher agent with varying amounts of environment interactivity. Mean and 95% confidence interval on the validation scenes are reported.

Interactivity	Success ↑	SPL ↑	SCT ↑
1.0 @t=0	0.175±0.024	0.166±0.023	0.030±0.004
1.0 @t=500k	0.120±0.020	0.118±0.020	0.021±0.004
1.0 @t=1M	0.115±0.020	0.112±0.019	0.027±0.005
1.0 @t=1.5M	0.095±0.018	0.089±0.017	0.018±0.004
1.0 @t=2M	**0.269±0.028**	**0.259±0.027**	**0.055±0.006**
0.0	0.223±0.026	0.153±0.019	0.037±0.005

Super-expert performance via RL fine-tuning. We fine-tune both the expert-trained student agent and the oracle-trained student agent with reinforcement learning using PPO for 10M simulator steps. The performance of the oracle-trained student agent drastically improves with fine-tuning, and we see that in Figure 9, both student agents are able

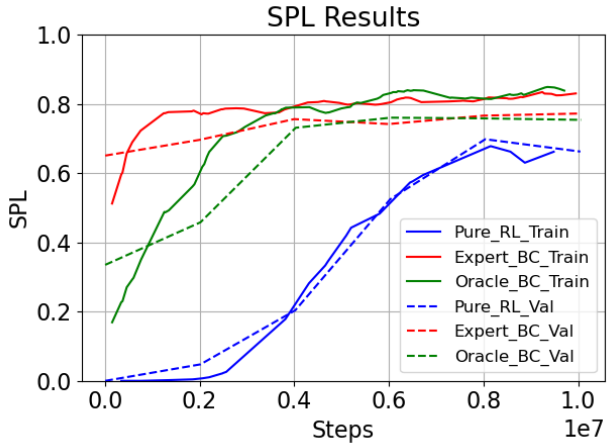

Fig. 9: Average SPL on the training and validation set after fine-tuning. The student agents achieve a higher SPL than an agent trained using reinforcement learning from scratch. However, the student agents seem to overfit slightly to the training environment, as evidenced by the larger gap in performance between the train and validation set.

to achieve a much higher SPL much faster than the agent that was trained from scratch using reinforcement learning. However, we notice that the performance of the student agents drop slightly when evaluated on the validation set, whereas the performance of the pure reinforcement learning agent seems to stay constant. This implies that the despite only being trained for 2M steps during the behavioral cloning phase, the agents seem to be over-fitting to the training data much more than the pure RL agent despite being trained for 10M more steps in the RL fine-tuning phase.

Fig. 10: Average SCT on the validation set after fine-tuning. The student agents are able to achieve a higher SCT than the expert agent it learned from, indicating that the students learn to leverage their continuous action space, although it learned from an expert agent with a discrete action space.

Additionally, we measure SCT to compare the completion times between the students and the expert discrete teacher agent. From Figure 10, we notice that the student agents are able to achieve a much higher SCT than the expert agent it learned from. This demonstrates that the student agent is able to learn to leverage its continuous action space to reach goals via a faster path over the discrete teacher agent it originally learned from.

B. Adversarial Alignment Results

In this section, we provide translation samples from our most stable alignment models, summarize experimental findings, and discuss means for improvement. Intuitively, we expect the module to be learn to interpolate its inputs. As we discuss, this is not achieved and training was unstable, and so our adversarial alignment was not tested in the joint setting.

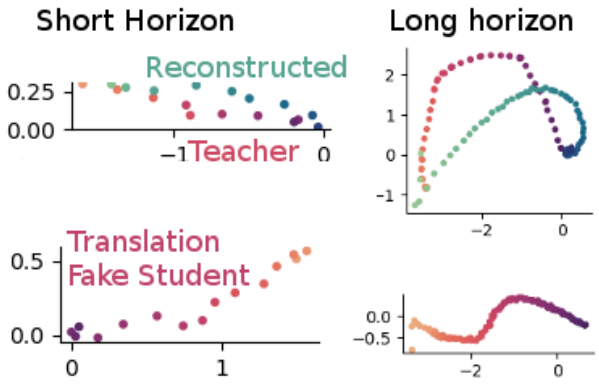

Fig. 11: **CycleGAN outputs.** We plot a teacher trajectory (its top-down GPS locations), its student translation, and the reconstructions of the teacher trajectories. Translation is has denser spacing and is not erratic, but trajectories are arbitrarily rotated. Training collapses on long horizons.

CycleGAN variants. The basic CycleGAN implementation successfully produces denser outputs when translating from a 0.25m teacher to a 0.15m student as shown in Fig. 11. However, the output is additionally rotated relative to the input. The degree of rotation varies across multiple runs and episodes (not shown). Nonetheless, besides this quirk, adversarial training is stable and suggests scaling to longer trajectories. However, when translating 100+ step episodes, training collapses when the discriminator overwhelms the generator. We experimented with allowing the generator to operate with state displacements instead of the state itself, as well as allowing the generator to begin outputing values before reading the whole input sequence; neither produced successful results. It is possible that the short-horizon model would suffice for AIRL, but if not, CycleGAN appears to fail to scale.

AttenderGAN variants. We repeat the short and long-horizon experiments, presenting results in Fig. 12. In the short horizon, outputs are extracted from consistent locations

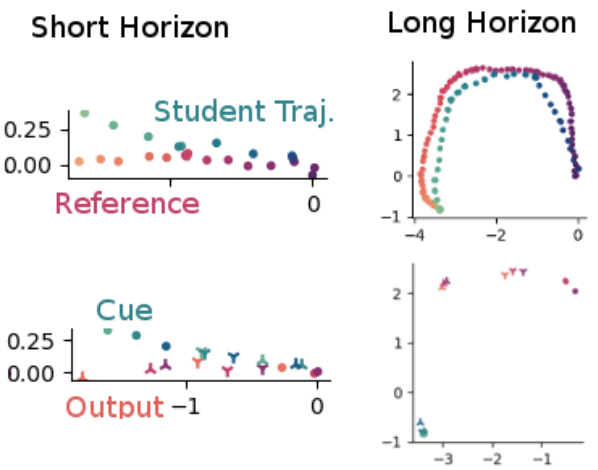

Fig. 12: **AttenderGAN outputs.** We plot a reference teacher trajectory (red) along with the paired student trajectory (green). The second plot shows three different student cues with their corresponding AttenderGAN outputs, distinguished with different markers. AttenderGAN outputs are shuffled relative to the input cues. Outputs are entirely ungrounded on long horizons.

in the expert trajectory. Notably, however, there is no apparent pattern between a cue's location and the trajectory it extracts. This lack of intuitive grounding results in uninterpretable extractions in the long-horizon setting. One remedy for this mismatch is to provide the discriminator a part of the cue, *e.g.* the first state of the cue. However, this makes the generation problem harder and results again in disriminator collapse (not shown).

We unsuccessfully try several variants of the architecture, including:

- Using a CNN or RNN as the cue encoder.
- Adding contrastive objectives to ground the output to the cue.
- Providing the generator a sequence of contextualized inputs instead of an initial condition.

It is possible that multi-scale contrastive losses (as in [21]) or batch-wise contrastive losses could provide better grounding without destabilizing the adversarial training.

VI. Conclusion

In this work, we demonstrate that policy distillation via our DAgger-like behavioral cloning approach enables us to train navigation agents that reach higher performance levels much faster than navigation agents trained with pure reinforcement learning. We find that this holds true even in the absence of a learned expert agent, showing that using an oracle shortest path follower as the teacher agent yields similar levels of performance after RL fine-tuning. We find that in the case of using a learned expert agent as the teacher, more interactivity improves the performance of the student agent, whereas limiting interactivity improves the performance of the student agent when using the oracle as the teacher.

On the other hand, adversarial alignment was more challenging than expected. Alignment fails to scale to full length trajectories, which would be necessary to provide meaningful adversarial rewards. It is possible the existing approach could be made to work as the task does not appear challenging for humans and indeed an non-learning baseline (e.g. dynamic time warping) could produce reasonable results. Alternately, there may be aspects of the alignment (*e.g.* which semantics are important and which should be translated) that are poorly formalized and warrant more attention.

References

[1] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, *et al.*, "DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames," in *ICLR*, 2020.

[2] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *CVPR*, 2018.

[3] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, *et al.*, "Sim2real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, p. 6670–6677, Oct 2020. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.3013848

[4] J. Truong, S. Chernova, and D. Batra, "Bi-directional domain adaptation for sim2real transfer of embodied navigation agents," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2634–2641, 2021.

[5] "Locobot: An open source low cost robot," https://locobot-website.netlify.com/, 2019.

[6] "Willow garage," https://www.turtlebot.com/.

[7] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on autonomous mobile service robots*, 2016.

[8] Y. Tang and S. Agrawal, "Discretizing continuous action space for on-policy optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5981–5988.

[9] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[10] D. Hejna, L. Pinto, and P. Abbeel, "Hierarchically decoupled imitation for morphological transfer," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 13–18 Jul 2020, pp. 4159–4171.

[11] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," *CoRR*, vol. abs/1912.12294, 2019. [Online]. Available: http://arxiv.org/abs/1912.12294

[12] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, *et al.*, "Solving rubik's cube with a robot hand," *CoRR*, vol. abs/1910.07113, 2019. [Online]. Available: http://arxiv.org/abs/1910.07113

[13] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," 2018.

[14] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine, "Avid: Learning multi-stage tasks via pixel-level translation of human videos," 2020.

[15] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2020.

[16] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, *et al.*, "One-shot imitation learning," 2017.

[17] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, *et al.*, "On Evaluation of Embodied Navigation Agents," *arXiv preprint arXiv:1807.06757*, 2018.

[18] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, *et al.*, "Habitat: A Platform for Embodied AI Research," in *ICCV*, 2019.

[19] N. Yokoyama, S. Ha, and D. Batra, "Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation," *arXiv preprint arXiv:2103.08022*, 2021.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[21] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," 2020.