GRIDToPix: Training Embodied Agents with Minimal Supervision

Unnat Jain1 Iou-Jen Liu1 Svetlana Lazebnik1 Aniruddha Kembhavi2 Luca Weihs2* Alexander Schwing1*

1University of Illinois at Urbana-Champaign 2PRIOR @ Allen Institute for AI

https://unnat.github.io/gridtopix/

Abstract

While deep reinforcement learning (RL) promises freedom from hand-labeled data, great successes, especially for Embodied AI, require significant work to create supervision via carefully shaped rewards. Indeed, without shaped rewards, i.e., with only terminal rewards, present-day Embodied AI results degrade significantly across Embodied AI problems from single-agent Habitat-based PointGoal Navigation (SPL drops from 55 to 0) and two-agent AI2-THOR-based Furniture Moving (success drops from 58% to 1%) to three-agent Google Football-based 3 vs. 1 with Keeper (game score drops from 0.6 to 0.1). As training from shaped rewards doesn’t scale to more realistic tasks, the community needs to improve the success of training with terminal rewards. For this we propose GRIDToPix: 1) train agents with terminal rewards in gridworlds that generically mirror Embodied AI environments, i.e., they are independent of the task; 2) distill the learned policy into agents that reside in complex visual worlds. Despite learning from only terminal rewards with identical models and RL algorithms, GRIDToPix significantly improves results across tasks: from PointGoal Navigation (SPL improves from 0 to 64) and Furniture Moving (success improves from 1% to 25%) to football gameplay (game score improves from 0.1 to 0.6). GRIDToPix even helps to improve the results of shaped reward training.

1. Introduction

The Embodied AI research community has developed a host of capable simulated environments focused on the challenges of navigation [58, 78], interaction [35, 77, 24], manipulation [79, 32, 81], and simulation-to-real transfer [21, 50, 34]. Fast progress has been made within these environments over the past few years, particularly in navigation heavy tasks such as PointGoal navigation [58, 75].

While early applications of deep RL to Embodied AI set down the more challenging path of learning from terminal

---

*denotes equal mentoring by LW and AS
Figure 2. GRIDTOPIX methodology to transfer knowledge from gridworld to visual agents. We propose to train agents in gridworld environments, where perception is simplified and simulation is fast, mirroring visual environments used in Embodied AI research. Learned gridworld policies can then be distilled to visual agents via imitation learning. See Sec. 3.3 for detailed notation.

As a first step towards this goal, we empirically analyze the ability of modern tried-and-true Embodied AI algorithms to learn high-quality policies from only terminal rewards in visual environments (DirectPix). We consider a variety of challenging tasks in three diverse simulators – single-agent PointGoal Navigation in Habitat [58] (SPL drops from 55 to 0), two-agent Furniture Moving in AI2-THOR [35] (success drops from 58% to 1%), and three-agent 3 vs. 1 with Keeper in the Google Research Football Environment [38] (game score drops from 0.6 to 0.1). Given only terminal rewards, we find that performance of these modern methods degrades drastically, as summarized in Fig. 1. Often, no meaningful policy is learned despite training for millions of steps. These results are eye-opening and a reminder that we are still ways off in our pursuit of embodied agents that can learn skills with minimal supervision, i.e., supervision in the form of only terminal rewards.

Driven by this hypothesis, we study GRIDTOPIX, a training routine for embodied agents that decouples the joint goal of planning from visual input into two manageable pieces. Specifically, using general-purpose gridworld environments which generically mirror the embodied environments of interest, we first train a gridworld agent for the desired task. Within a gridworld, an agent has perfect visual capabilities, allowing learning algorithms to focus on long-horizon planning given only terminal rewards. Next, this gridworld agent supervises an agent operating solely on complex visual observations (no gridworld is needed at test time). Importantly, our conceptualization of ‘gridworld’ isn’t restricted to a top-down occupancy map. As we detail in Sec. 3.2, gridworlds are perfect-perception environments, i.e., semantics are explicitly available to the agent.

Across tasks, as summarized in Fig. 1, we observe that GRIDTOPIX significantly outperforms directly comparable methods when training visual agents using terminal rewards: for PointGoal Navigation the SPL metric improves from 0 to 64; for Furniture Moving success improves from 1% to 25%; for 3 vs. 1 with Keeper the game score improves from 0.1 to 0.6. Moreover, GRIDTOPIX even improves benchmarks when trained with carefully shaped rewards. This finding is analogous to the progress made by weakly supervised computer vision approaches which inch towards benchmarks set by fully supervised methods.

2. Why Learn From Terminal Rewards?

Human supervision, hand-written and rule-based expert teachers, shaped rewards, and custom architectures are common when developing increasingly capable embodied agents. Several questions naturally arise when developing agents with only minimal supervision. As rule-based optimal experts are frequently available in to-

\[ \text{GRID AGENT} \] (perfect perception)

\[ \text{Agent Trajectory in Environment} \]

\[ \text{GRID AGENT} \] (perfect perception)

\[ \text{Agent Trajectory in Environment} \]

\[ \text{GRID AGENT} \] (RGB observations)

\[ \text{Agent Trajectory in Environment} \]

\[ \text{GRID AGENT} \] (RGB observations)
day’s simulated environments and tasks, why not use them for dense supervision at every step? Admittedly, many present-day embodied tasks are navigational which permits computing optimal actions easily using shortest path algorithms. The community is, however, quickly moving to more intricate and physics-based tasks, where rule-based experts are computationally expensive or infeasible. Consider, for example, the games of Go, Hanabi, and Football, or the real world tasks of elderly assistance and disaster relief. It is extremely difficult to devise a heuristic expert, let alone an optimal one, for any of those examples. Creating heuristic experts in these settings is just as, if not more, labor intensive as the reward shaping we hoped to avoid. With these long term pursuits in mind, we think it is important to study and develop methods that learn with minimal supervision, including in tasks that are popular today. In this work, optimal actions can be easily computed for the popular Point Navigation task [58]. But for the other two tasks we consider – FurnMove [29] and 3 vs. 1 with Keeper [38] – no experts are available in the literature and creating such an expert appears to be extremely difficult.  

If not rule-based experts, how about collecting human annotations? Deep models need the equivalent of years of expert supervision even for perfect-perception tasks [16]. Embodied AI would undoubtedly require a similar (if not much larger) number of annotations. Collecting these human annotations is labor intensive and must be done for every new task and behavior we wish our agent to complete. This is extremely expensive and intractable for more complex behaviors. In contrast, terminal rewards are easy to provide and allow years of simulated self-play (tractable in wall-clock time). This goal aligns closely with the pursuits of the AI community – pre-training with self-supervision and then learning new skills with minimal human help. Why should we expect that computing terminal rewards is easier than computing shaped rewards? While we can’t “prove” that terminal rewards are easier to compute than shaped rewards, we believe there is good evidence that this is, in general, the case. Shaped rewards are frequently designed so that they provide, at every step, feedback to the agent about its progress towards achieving the goal. Computing such rewards thus requires to approximate a ‘distance,’ in terms of agent actions, between an arbitrary state and the goal. It is not hard to construct tasks where computing such a distance is NP-complete (exponential-time with known algorithms). In comparison, verifying that an agent has reached a goal (and thus deserves its terminal reward) requires no such search, making it, in principle, an easier problem.  

3. GridToPix  

We are interested in learning from only terminal rewards a high-performance policy $\pi^V$ which acts on realistic (e.g., visual) observations within an Embodied AI environment, hereafter called the visual environment. However, empirically and irrespective of the task and environment, we find joint learning of perception and planning from terminal rewards (DirectPix) to be extremely challenging, as summarized in Fig. 1.  

In contrast, due to their perceptual simplicity, gridworlds are generally extremely fast to simulate, and learning is sample efficient as a gridworld policy $\pi^G$ needs to devote little effort to learn accurate perception. This enables gridworld agents to rapidly learn high-quality policies $\pi^G$ from only terminal rewards.  

This advantage of gridworlds is an opportunity to reduce labor-intensive reward shaping efforts: can we first learn policies from terminal rewards within gridworlds that replicate the dynamics of a visual environment and then efficiently transfer these policies to visual agents?  

We refer to the approach which is designed to enable this transfer and which is sketched in Fig. 2 as GridToPix, and discuss it next. In particular, Sec. 3.1 formalizes terminal and shaped rewards, Sec. 3.2 describes how we design gridworlds that replicate a visual environment’s dynamics, and Sec. 3.3 details both how we train gridworld agents and how imitation learning can be used to transfer gridworld policies to visual agents.  

3.1. Terminal vs. Shaped Reward Structures  

The reward structure used during training has a substantial impact on sample efficiency, stability, and quality of the learned policy. Most prior work for the tasks considered in this paper used a shaped reward structure, i.e., the reward obtained by the agent at timestep $t$ can be decomposed into  

$$r^\text{shaped}_t = r^\text{success}_t + r^\text{progress}_t + r^\text{all}_t.$$  

Here, $r^\text{success}_t$ is a sparse terminal reward equalling 0 on all but the last timestep; for our considered tasks,  

$$r^\text{success}_t = \begin{cases} r^\text{success} \cdot (1 - \alpha \cdot \frac{t}{T}) & \text{if goal achieved}, \\ 0 & \text{otherwise}, \end{cases}$$  

where $T \geq 1$ is the maximum allowed episode length and $r^\text{success} > 0$, $\alpha \in [0, 1]$ are task-specific constants. Further, $r^\text{progress}_t$ is a dense, goal-dependent reward, often equaling solutions is easy (polynomial time) but in which finding these solutions is fundamentally harder (unless P=NP).  

3The inclusion of the $\alpha$ parameter in terminal rewards allows to encode a more nuanced measure of success. Note that when $\alpha > 0$, $r^\text{success}_t$ is larger when the agent completes the task in fewer steps.
the change in distance to the goal. Finally, \( r_t^{\text{progress}} \) is a possibly dense but goal-independent reward, meant to encourage behaviors that are generally thought to be “good” regardless of the task, e.g., \( r_t^{\text{progress}} \) often includes a small negative step penalty which gently encourages the agent to complete its task quickly.

As previously discussed, the goal-dependent shaped rewards \( r_t^{\text{goal-reward}} \) must be manually designed, can be computationally expensive, and generally require privileged access to the environment state. Because of this, we often aim to train agents with the simpler reward structure

\[
    r_{t}^{\text{terminal}} = r_{t}^{\text{success}} + r_{t}^{\text{progress}}.
\]

In fact, for our experiments with terminal rewards, we let \( r_{t}^{\text{progress}} = 0 \) in all but the FurnMove task where we keep prior work’s inclusion of a step and failed-coordination penalty.

3.2. Gridworld Mirrors of Visual Environments

In order for policies obtained from gridworld training to be successfully transferable to visual agents, we need these gridworlds to be mirrors of their visual counterparts: a step in the gridworld should be translatable to a step in the visual world. We will now describe the three gridworlds used in our experiments, mirroring the AIHabitat, A2-THOR, and Google Football environments. While these gridworlds work well for the considered tasks, they are also applicable for other tasks trained in these environments. We hope to encourage current and future Embodied AI environment developers to provide comprehensive general purpose gridworlds corresponding to their environments.

**GRID-HABITAT.** We extend the existing functionality of the AIHabitat platform to permit, for the first time, training of gridworld agents. Specifically, we generate an egocentric, top-down, observation with the agent positioned at the bottom facing towards the center (see Fig. 2). This top-down observation contains sensor information about occupancy (i.e., free space and walls) and goal location. This sensor information is sufficient for PointNav and can be easily enriched for other tasks (e.g., by adding semantic channels for ObjectNav). Our gridworld observation is different from the top-down visualization tool provided by AIHabitat which is allocentric and scaled differently for different scenes. We call this new gridworld GRID-HABITAT.

**GRID-A2-THOR.** For the multi-agent FurnMove task, we follow Jain et al. [29] and conduct our study on an A2-THOR-mirroring top-down gridworld. Given the complexity of FurnMove, we include information in separate channels of the top-down tensor with each channel corresponding to the output of a distinct sensor. These sensors specify whether a cell can be occupied by a furniture item, whether the cell is reachable by an agent, whether it was previously visited, the location of the furniture item, and the location of the other agent. We refer to this gridworld as GRID-A2-THOR.

**GRID-GOOGLE-FOOTBALL.** For the multi-agent Google football environment created by Kurach et al. [38], we construct an observation that summarizes important game state information. Specifically, for each controlled player, the gridworld observation is a 1D vector which contains the location of all other players, the ball, and the opponent goal relative to the controlled player’s location.

Importantly, as suggested in Sec. 1, our conceptualization of a gridworld goes beyond a spatial top-down occupancy map. While observations of GRID-HABITAT capture occupancy and goal information, the observation of GRID-A2-THOR is a tensor containing explicit semantics of the environment. Moreover, as in GRID-GOOGLE-FOOTBALL, gridworld observations need not be restricted to have spatial structure. Here, as is common in RL literature, gridworld agents receive a 1D vector capturing perfect-perception observations.

3.3. Supervising Visual Agents via Gridworlds

We now describe how we train gridworld agents and subsequently use imitation learning to transfer their learned policies to visual agents. An overview is given in Fig. 3. Training in Gridworlds. As is standard in RL, we train gridworld agents to maximize the expected \( \gamma \)-discounted cumulative reward of their actions. Task specific details (e.g., algorithms and hyperparameters) are included in Sec. 4. Whenever possible we follow protocols laid out by prior work.

Distilling policies. How can we effectively train the parameters of a visual agent policy \( \pi^V \) given a policy \( \pi^G \) trained in a gridworld? To answer this question, recall that the visual agent \( V \) is interacting with a visual environment for which every step is translatable to a gridworld. The vi-
ual agent can hence be supervised by a gridworld agent $G$. The visual agent $V$ might act using its own policy $\pi_V$ or adopt an exploration policy $\mu$. This leads to on-policy and off-policy variants of imitation learning (IL). To illustrate this, let a rollout of the visual agent $V$ and the gridworld agent $G$ be $(a_1^V, a_2^V, \ldots, a_T^V)$ and $(a_1^G, a_2^G, \ldots, a_T^G)$, respectively. To train the parameters of the visual policy $\pi_V$ via GRIDToPIX, we employ the cross-entropy loss. Formally, suppose that $O_\mu$ is a random variable corresponding to the observation seen by an agent when following policy $\mu$ and suppose that $H_\mu$ is a random variable encoding the history of all observations seen before obtaining observation $O_\mu$. The GRIDToPIX loss is then

$$L_{GRIDToPIX} = \mathbb{E}_{a \sim \pi_\mu(O_\mu, H_\mu)}[-\log \pi_V(a \mid O_\mu, H_\mu)].$$

(4)

The choice of exploratory policy $\mu$ leads to three variants, each widely adopted in IL tasks: student forcing, teacher forcing, and annealed teacher forcing or DAgger (details in Appendix). In our experiments we primarily use DAgger. Importantly, no matter the exploration policy used during training, at test time the visual agent’s policy $\pi_V$ is deployed so that there is no access to gridworld policies at test time.

4. Tasks, Models, and Evaluation

We evaluate GRIDToPIX using three tasks. We chose these tasks as they (a) span both single-agent and multi-agent settings, (b) include tasks that do not have easily constructed rule-based experts (FurnMove and 3 vs. 1 with Keeper Football) as well as a task for which optimal actions can be computed from shortest path computations (PointNav), (c) provide the opportunity to test GRIDToPIX across a variety of different reward structures and model architectures, and (d) employ three different Embodied AI environments. Additionally, in the appendix, we include experiments for a task where perfect reward shaping is intractable – Visual Predator-Prey (in OpenAI multi-particle environment [46, 49]). Across all experimental results, we present standard evaluation metrics after 10% and 100% of training has completed (to provide an understanding of sample efficiency and asymptotic results).

4.1. PointGoal Navigation

PointGoal Navigation (PointNav) is a single agent navigation task specified for the AIHABITAT simulator. An agent is spawned at a random location in the scene and must navigate to a target location specified by coordinates relative to the agent’s current location. The relative goal position is available at every time step and the agent navigates by choosing one of four actions at every step. Shaped rewards. Traditionally trained with shaped rewards [58, 75, 80], the change in shortest-geodesic-path distance to the goal is chosen as the goal-dependent progress reward $r^\text{progress}_t$ (see Eq. (1)). Computing it requires access to the full scene graph and a shortest-path planner. Here the success reward $r^\text{success}_t$ is chosen as in Eq. (2) with $r^\text{success}_t = 10$ and $\alpha = 0$. Finally, $r^\text{fail}_t$ is a constant step penalty of $-0.01$.

Terminal rewards. Here we use the reward structure from Eq. (3) with $r^\text{success}_t$ obtained from Eq. (2) with $r^\text{success}_t = 10$ and $\alpha = 0.9$. We set $r^\text{penalty}_t = 0$.

Model architecture. For a fair comparison, we use the standard CNN-GRU architecture [58, 75, 14, 9, 11, 69]. We adopt the official implementation$^6$ and train with PPO [59], the de-facto standard RL algorithm for PointNav.

Evaluation. PointNav agents are primarily evaluated via Success weighted by Path Length (SPL) [1] and percentage of successful episodes (Success). We set a budget of 50 million environment steps$^7$ and report results on the Gibson validation set of 14 unseen scenes with 994 episodes.

4.2. Furniture Moving

FurnMove is a challenging two-agent furniture moving task set within the AI2-THOR simulator [29]. Two agents collaborate to move an object through the scene and place it above a visually distinct target. Agents may communicate with other agents at each timestep using a low bandwidth communication channel. Each agent can choose from 13 actions. Specifically, in addition to the vanilla navigation, agents may move with the lifted object, move just the object, and rotate the object. Moreover, due to two agents, the joint action space contains 169 actions.

Shaped rewards. Here, change in Manhattan distance to the goal$^8$ is used as the goal-dependent reward $r^\text{progress}_t$. Three goal-independent rewards are suggested in [29] to help learn the coordinated policy for this multi-agent system. In particular, $r^\text{joint}_t$ includes a step penalty, a joint-pass (or do-nothing) penalty, and a failed action penalty.

Terminal rewards. For a head-on comparison with [29] we simply drop $r^\text{progress}_t$. Like prior work [29], the success reward $r^\text{success}_t$ is obtained from Eq. (2) using $r^\text{success}_t = 1$ and $\alpha = 0$. All FurnMove results are based on this. Additional results with $r^\text{fail}_t = 0.01$ are included in the appendix.

Model architecture. We utilize SYNC, the best-performing architecture from prior work [29]. For fairness, we use the same RL algorithm (A3C [47]) as prior work.

Evaluation. FurnMove agents are primarily evaluated via two metrics – % successful episodes (Success) and a Manhattan distance based SPL (MD-SPL). Additional metrics are reported in the appendix. Consistent with [29], we train

$^6$github.com/facebookresearch/habitat-lab

$^7$10.5 days of training using a g4dn.12xlarge AWS instance configured with 4 NVIDIA T4 GPUs, 48 CPUs, and 192 GB memory.

$^8$In FurnMove, computing the shortest-geodesic-path is intractable as each location of the furniture corresponds to over 400k states [29].
for 500k episodes for the visual agents. Metrics are reported on the test set and learning curves on validation.

### 4.3. Football – 3 vs. 1 with Keeper

3 vs. 1 with Keeper is a task introduced by Kurach et al. [38]. In this multi-agent task, three agents collaborate to score against rule-based defenders. One agent starts behind the penalty arc and the other two agents start on the sides of the penalty area. There are two opponent rule-based defenders, including one goalkeeper. At the beginning of each episode, the center agent possesses the ball and faces the defender. The episode terminates when the controlled agents score or the maximum episode length is reached.

**Shaped rewards.** We use the ‘checkpoint’ [38] reward as the goal-dependent reward \( r^\text{progress} \) to encourage controlled agents to move the ball towards the opponent’s goal. Specifically, the area between the initial location of the center agent and the opponent goal is divided into three checkpoint regions according to the distance to the opponent goal. A progress reward of \(+0.1\) is received the first time the controlled players posses the ball in a checkpoint region.

**Terminal rewards.** Following [38], \( r^\text{success} = 1 \) and \( \alpha = 0 \).

**Model architecture.** For a fair comparison, we use the CNN architecture used by [38, 42], and a parallel PPO algorithm [42] to train the agents.

**Evaluation.** Agents are evaluated using the average score obtained in test episodes. Consistent with Liu et al. [42], we set a training budget of 8 million environment steps. Training curves and final metrics are reported on test episodes.

### 5. Experiments

We now provide an overview of the training routines employed to train our visual and gridworld agents, followed by results on the three tasks.

#### 5.1. Training Visual Agents

For each task (in both the terminal reward and shaped reward settings), we train visual agents in three ways:

**DirectPix.** We train visual agents (hence, ‘Pix’) using only reward-based RL losses (i.e., PPO or A3C).

**GridToPix.** Our proposed routine trains visual agents by imitating a corresponding gridworld expert (that we train).

As stated before, the expert is only available at training time.
Table 2. **Quantitative results (shaped reward structure).** Identical to Tab. 1 but all methods have been trained with shaped, rather than terminal, rewards. Even with this denser form of supervision, training routines based on GridToPix can improve performance of all the metrics across the three tasks.

| Tasks | PointGoal Navigation | Furniture Moving | 3 vs. 1 with Keeper |
|-------|----------------------|------------------|---------------------|
|       | SPL @ 10% @ 100% @ 10% @ 100% | MD-SPL @ 10% @ 100% @ 10% @ 100% | Game Score @ 10% @ 100% |
| DirectPix | 35.9 54.7 60.5 79.9 | 2.8 11.2 22.5 58.4 | 0.0 0.6 |
| GridToPix | 57.6 69.0 77.5 86.4 | 8.4 9.7 57.7 62.0 | 0.37 0.65 |
| GridToPix → DirectPix | 53.6 69.1 71.9 84.9 | 7.1 15.3 55.3 68.6 | 0.38 0.61 |

Gridworld expert (upper bound) 85.3 97.5 22.2 76.3 0.95

**5.2. Training Gridworld Experts**

For each task, the gridworld expert utilizes a training routine identical to DirectPix. Gridworld experts observe semantic top-down tensors or 1D state information, unlike the architectures that learn from raw pixels. Hence, we make minimal edits to the CNN that encodes the observations (see appendix for details).

As described in Sec. 3, training in gridworlds is very fast. Moreover, gridworlds can, in principle, be optimized to accelerate the environment transitions. Hence, for gridworld variants of each of the three tasks, we train models to near saturation. Importantly, due to difference in state space, models, and training time, the gridworld expert rows in Tab. 1 and Tab. 2 shouldn’t be compared to visual analogues. They serve as a loose upper-bound for the performance of the visual agents trained with GridToPix.

**5.3. Results**

We report standard evaluation metrics on three tasks. To study sample efficiency, we also show learning curves.

**Terminal rewards** (see Tab. 1 and Fig. 4). With perfect perception, gridworld experts can train to a high performance (e.g., 94% success in PointNav) and guide the learning of visual agents. In sharp contrast, DirectPix doesn’t learn a meaningful policy in any of the tasks – demonstrating present day methods’ inability to learn from terminal rewards in these visual worlds. Our GridToPix variants perform significantly better. For instance, at PointNav, GridToPix obtains a respectable SPL of 0.638, inching towards the 0.788 obtained by the gridworld expert. GridToPix→DirectPix sometimes produces small gains beyond GridToPix. As evidenced by learning curves, GridToPix learning is also efficient – most gains come in the early few million steps (0.45 SPL, i.e., 70% of the final performance in just 5M steps for PointNav). A similar pattern emerges in other tasks – DirectPix cannot learn meaningful behaviors from terminal rewards with a 0.8% success and 0.07 game score in FurnMove and

---

11We investigated the learned policy qualitatively in the PointNav task and found that the DirectPix agent learns the (locally optimal) policy of executing STOP as its first action. We report best number after training over 15 configurations (20M steps each) by varying the entropy coefficient, number of PPO trainers, and random seeds. For 3 vs. 1 with Keeper, the DirectPix agents shoot straight at the goal and are intercepted by the goalkeeper and sometimes pass to another player who fails to receive.
3 vs. 1 with Keeper, respectively. Our training routines achieve the best performance.

In addition to these improvements in the terminal reward structure, we also investigate the training routines in (the more traditional) setting with shaped rewards.

Shaped rewards (see Tab. 2 and Fig. 5). As seen in past works, DirectPix performs well when provided with shaped rewards. Interestingly, GRIDToPix → DIRECTPix also outperforms DirectPix in this reward setting in the first two tasks and performs roughly as well on the third task—an over 25% relative gain in SPL (PointNav), an over 35% relative gain in MD-SPL (FurnMove), and an over 8% relative gain in game score (Football). The efficiency of learning is also high with GRIDToPix → DIRECTPix reaching 83% of its final SPL in just 5M steps for the PointNav task. Since PointNav is single-agent navigation, we trained an agent using the optimal shortest-path actions via IL. GRIDToPix outperforms this baseline with (very) dense supervision.\footnote{By imitating optimal actions, the visual agent can reach a SPL at (10%, 100%) = (0.301, 0.687) and success at (10%, 100%) = (35.5, 76.7).}

These results across three tasks running in three simulators demonstrate the potential of GRIDToPix as we move towards training with terminal rewards. In addition GRIDToPix also provides meaningful gains in our current Embodied AI training setups that employ shaped rewards.

6. Related Work

**Embodied task completion.** Development of visually realistic simulators [10, 35, 4, 77, 58] has led to tremendous progress for embodied agents. Agents are trained for a variety of tasks including multiple variants of indoor navigation [21, 58, 77, 73, 82, 70, 64], question answering [18, 26, 74], instruction following [2, 23, 67, 36, 37, 63], adversarial gameplay [72, 13], and multi-agent collaboration [30, 29, 38, 40, 52]. In this work we have focused on three diverse tasks, particularly, PointNav [58], FurnMove [29], and 3 vs. 1 with Keeper [38]. In contrast to past works that generally aim to maximize success rates at these tasks using any and all possible supervision, we focus on achieving high performance while using only terminal rewards.

**Sparse rewards.** Learning from sparse rewards has long been of interest within the RL community where research has, in large part, focused on visually simple environments. A number of approaches have been proposed: curiosity as intrinsic motivation for exploring diverse states [53], using hindsight to reinterpret “failed” trajectories [3], curriculum learning [8, 22], self-play [62, 6], and learning to shape rewards [66, 51, 25]. Unlike these approaches, we are interested in enabling models to learn from terminal rewards in visually complex environments. Our GRIDToPix method can be further enhanced by incorporating any of the above ideas: rather than using standard RL to train our gridworld experts we could instead use one of the above methods and, potentially, obtain even better results. As our gridworld experts already learn high quality policies without these methods we leave this for future work.

**Accelerating visual reinforcement learning.** Several methods have been proposed to improve the sample and wall-clock efficiency of RL in embodied tasks. For sample efficiency, agents are often optimized with additional auxiliary self-supervised tasks [83, 80]. For faster training, advances for distributed and decentralized scaling of PPO have been proposed [75, 42]. Pertinent to our work, Jain et al. [29] developed an AI2-THOR-aligned gridworld (16× faster than AI2-THOR [35]) to prototype their experiments and scale their tasks to a larger number of agents.

**Imitation learning.** An alternative to reward-based training of agents is imitation of a supervisory policy or demonstrations via behavior cloning [57, 5]. E.g., Data Aggregation (DAgger) [55, 56] mitigates the covariate-shift that troubles classical behaviour cloning approaches. This has been applied to train visual agents via rule-based shortest path experts [27, 19, 30, 71]. For the task of instruction following [60], separate neural modules for perception and action have shown to improve policies learned by imitating human-annotated data [63]. Chen et al. [15] train gridworld agents to predict waypoints for AI driving. Their privileged agents are trained by offline behavior cloning of human-labeled trajectories (with data augmentation). Obtaining human-labeled data for dense supervision for sequence prediction can quickly become costly and time-consuming. In contrast, we train visual agents using self-supervised labels from their gridworld counterparts. Since these experts learn by interacting in the gridworld via minimal supervision of terminal rewards, no human-labeled data is needed.

7. Conclusion

Embodied AI progress is slowed by labor-intensive work to manually tweak model architectures, collect human annotations, and/or shape reward functions. We study how to reduce this effort significantly: for an embodied environment of interest, create a generic gridworld mirror which is visually parsimonious and fast to simulate. Use of this gridworld mirror for GRIDToPix enables one to avoid many task-specific interventions and allows for learning from only terminal rewards: a critical step for complex tasks within physically realistic environments where generating shaped rewards quickly becomes infeasible.

Acknowledgements: This work is supported in part by NSF under Grant #1718221, 2008387, 2045586, MRI #1725729, and NIFA award 2020-67021-32799. We thank Anand Bhattad, Angel X. Chang, Manolis Savva, Martin Lohmann, and Tanmay Gupta for thoughtful discussions and valuable input.
References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757, 2018. 5

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In CVPR, 2018. 1, 8

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, NeurIPS, 2017. 8

[4] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105, 2017. 8

[5] Michael Bain and Claude Sammut. A framework for behavioural cloning. In Machine Intelligence, 1995. 5

[6] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. arXiv preprint arXiv:1909.07528, 2019. 8

[7] S. Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In NeurIPS, 2015. 13

[8] Y. Bengio. Learning Deep Architectures for AI. Foundations and Trends in ML, 2009. 8

[9] Vincent Cartillier, Zhihe Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allo-centric semimaps and representations from egocentric views. arXiv preprint arXiv:2010.01191, 2020. 5

[10] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In 3DV, 2017. 8

[11] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos. In NeurIPS, 2020. 5

[12] Devendra Singh Chaplot, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural mapping. In ICLR, 2020. 1

[13] Boyuan Chen, Shuran Song, Hod Lipson, and Carl Vondrick. Visual hide and seek. In Artificial Life Conference Proceedings. MIT Press, 2020. 8

[14] Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vincenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. In ECCV, 2020. 5

[15] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In CoRL, 2020. 8

[16] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lah lou, Lucas Willems, Chitwan Saharia, Thien Hu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In ICLR, 2019. 2, 3

[17] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. CVPR, 2005. 1

[18] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In CVPR, 2018. 8

[19] A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural Modular Control for Embodied Question Answering. In ECCV, 2018. 7, 8

[20] Abhishek Das, Satwik Kottur, Jose M.F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In ICCV, 2017. 7

[21] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In CVPR, 2020. 1, 8

[22] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In CoRL, 2017. 8

[23] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In NeurIPS, 2018. 8

[24] Chuang Gan, Jeremy Schwartz, Seth Alter, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilis, Abhishek Bhandwalda, Nick Haber, Megumi Sano, et al. Threed世界第一: A platform for interactive multi-modal physical simulation. arXiv preprint arXiv:2007.04954, 2020. 1

[25] Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal-conditioned policies. arXiv preprint arXiv:1811.07819, 2018. 8

[26] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. IQA: Visual Question Answering in Interactive Environments. In CVPR, 2018. 8

[27] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. In CVPR, 2017. 8

[28] Meera Hahn, Jacob Krantz, Dhruv Batra, Devi Parikh, James M. Rehg, Stefan Lee, and Peter Anderson. Where are you? localization from embodied dialog. In EMNLP, 2020. 1

[29] Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander G. Schwing. BabyAI: First steps towards grounded language learning with a human in the loop. In ICLR, 2019. 2, 3

[30] Unnat Jain, Luca Weihs, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G. Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In CVPR, 2019. 8

[31] Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. In ACL, 2019. 7
L. Sifre, George van den Driessche, T. Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 2017. 2, 8

[63] Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. Moca: A modular object-centric approach for interactive instruction following. *arXiv preprint arXiv:2012.03208*, 2020. 8

[64] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Makysmet, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *arXiv preprint arXiv:2106.14405*, 2021. 8

[65] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *CoRL*, 2020. 1

[66] A. Trott, Stephan Zheng, Caiming Xiong, and R. Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. In *NeurIPS*, 2019. 8

[67] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dingshan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019. 1, 7, 8

[68] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *ECCV*, 2018. 7

[69] Saim Wani, Shivansh Patel, Unnat Jain, Angel Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. *NeurIPS*, 2020. 5

[70] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *CVPR*, 2021. 8

[71] Luca Weihs, Unnat Jain, Jordi Salvador, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander Schwing. Bridging the imitation gap by adaptive insubordination. *arXiv preprint arXiv:2007.12173*, 2020. 8

[72] Luca Weihs, Aniruddha Kembhavi, Kiana Ehsani, Sarah M Pratt, Winson Han, Alvaro Herrasti, Eric Kolve, Dustin Schwenk, Roozbeh Mottaghi, and Ali Farhadi. Learning generalizable visual representations via interactive gameplay. In *ICLR*, 2021. 8

[73] Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. Allenact: A framework for embodied ai research. *arXiv preprint arXiv:2008.12760*, 2020. 8

[74] Erik Wijmans, Samyak Datta, Oleksandr Makysmet, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *CVPR*, 2019. 8

[75] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2019. 1, 5, 8

[76] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1989. 13

[77] Fei Xia, William B. Shen, Chengshu Li, Priya Kasimbeg, Micael Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 2020. 1, 8

[78] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. 1

[79] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment. In *CVPR*, 2020. 1

[80] Joel Ye, Dhruv Batra, Erik Wijmans, and Abhishek Das. Auxiliary tasks speed up learning pointgoal navigation. In *CoRL*, 2020. 5, 8

[81] Tianhe Yu, Deirdre Quillen, Zhanpeng He, R. Julian, Karol Hausman, Chelsea Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019. 1

[82] Kuo-Hao Zeng, Luca Weihs, Ali Farhadi, and Roozbeh Mottaghi. Pushing it out of the way: Interactive visual navigation. In *CVPR*, 2021. 8

[83] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR*, 2020. 8

[84] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. In *ICRA*, 2017. 1
A. Appendix of GRIDTOPIX: Training Embodied Agents with Minimal Supervision

In this appendix we include:

A.1 Details of reward structure for {PointNav, FurnMove, 3 vs. 1 with Keeper} × {shaped, terminal} (Tab. A.1).
A.2 An alternate terminal reward structure for FurnMove by simply dropping a component from \( r^\perp \) (Tab. A.2).
A.3 Additional quantitative metrics for FurnMove introduced in [29] (terminal rewards – Tab. A.3 and shaped rewards – Tab. A.4).
A.4 Edits to allow deep net models to encode gridworld observation (vs. visual observations).
A.5 Teacher forcing probability and IL → RL transition for GRIDTOPIX and GRIDTOPIX→DIRECTPIX routines (Tab. A.5).
A.6 Experimental results for Visual Predator-Prey task in OpenAI multi-particle environment [46, 49]. This serves as a testbed wherein reward shaping is not enough (Fig. A.1 and Tab. A.6).
A.7 Qualitative visualizations of policy learned by DirectPix agent in 3 vs. 1 with Keeper (Fig. A.2 and Fig. A.3).

A.1. Reward Structures

In Tab. A.1, we list all the reward components for shaped and terminal rewards for the three tasks considered in this work. For a fair comparison, the shaped rewards are kept identical to prior work in PointNav [58], FurnMove [29], and 3 vs. 1 with Keeper [38]. For terminal rewards, as per the definitions in Sec. 3.1, there exists no \( r^\perp \) component (hence, ‘\( \bigtimes \)’). The results for these reward structures have been reported in the main paper (Tab. 1 and Tab. 2).

Common across all tasks, \( I[\text{success}] \) denotes the indicator function conditioned on success of the episode. ‘Step penalty’ is a small negative reward to encourage completion of the episode in fewer steps. In PointNav, the progress reward is based on the shortest-path geodesic distance to the goal from the current location of the agent (\( d_{t-1}^{\text{geodesic}} \)). In FurnMove, particularly in [29], the authors use a variant of the PointNav progress reward. Particularly, Manhattan distances are used (\( d_{t-1}^{\text{Man.}} \)) and a positive reward is received only if the agents get the furniture item closer to the goal compared to the minimum distance in previous steps (note the use of ‘\( \max(min) \)’). For 3 vs. 1 with Keeper, the progress reward is called the ‘checkpoint reward’. It is received if the agents are able to move the ball to a zone closer to the goal.

| Task        | Reward structure | \( r^\text{success}_t \) | \( r^\text{progress}_t \) | \( r^\perp_t \)            |
|-------------|------------------|--------------------------|-----------------------------|---------------------------|
| PointNav    | Shaped           | 10 \( \cdot I[\text{success}] \) | \( d_{t-1}^{\text{geodesic}} - d_{t-1}^{\text{geodesic}} \) | \( \bigtimes \)          |
| PointNav    | Terminal         | 10 \( \cdot (1 - 0.9 \cdot \frac{d_t}{d_0}) \cdot I[\text{success}] \) | \( \bigtimes \)          | Step penalty (−0.01)      |
| FurnMove    | Shaped           | 1 \( \cdot I[\text{success}] \) | \( \max(0, \min_{k=0,\ldots,t-1} d_{t-1}^{\text{Man.}} - d_{k}^{\text{Man.}}) \) | Failed action penalty (−0.02) |
| FurnMove    | Terminal         | 1 \( \cdot I[\text{success}] \) | \( \bigtimes \)          | Failed coordination (−0.01) |
| 3 vs. 1 with Keeper | Shaped           | 1 \( \cdot I[\text{success}] \) | \( \bigtimes \)          | Checkpoint reward (0.1)   |
| 3 vs. 1 with Keeper | Terminal       | 1 \( \cdot I[\text{success}] \) | \( \bigtimes \)          | 0                         |

Table A.1. Reward structures. For each task (PointNav, FurnMove, and 3 vs. 1 with Keeper), we list the three components of shaped and terminal reward structures. This includes a positive reward conditioned on success \( r^\text{success}_t \), a goal-dependent progress reward \( r^\text{progress}_t \), and a goal-independent reward \( r^\perp_t \). Terminal rewards, that we focus on in this work, do not include progress reward (see Sec. 3.1 for definitions). Hence, progress reward is marked with a ‘\( \bigtimes \)’ for terminal settings.

A.2. Alternate Terminal Reward Structures

As shown in Tab. A.1, the terminal reward structure for FurnMove is equal to the shaped reward structure [29] except that it does not include the progress reward component \( r^\text{progress}_t \). All results in the main paper (FurnMove column of Tab. 1 and Tab. 2) and Tab. A.3 and Tab. A.4 correspond to this terminal reward setting.

We, additionally, study an alternative terminal reward formulation where we drop the failed action penalty from the reward structure, i.e., only step penalty and failed coordination constitute \( r^\perp_t \). We found the metrics to improve with this reward structure, with only 250k episode of training. These results are summarized in Tab. A.2. By further dropping the failed coordination...
penalty, the training became sample-inefficient. This is coherent with the findings of Jain et al. [29]. Particularly, given the tightly-coupled nature of the agents in collaboratively moving furniture, the failed coordination reward is critical for efficient learning.

A.3. Additional FurnMove Metrics

In the main paper, we report the primary metrics of % successful episodes (Success) and a Manhattan distance based SPL (MD-SPL). For a fair comparison, we report three additional metrics that were included in [29]. This includes number of actions taken per agent (Ep Length), probability of uncoordinated actions (Invalid Prob Mass), and meters from goal at the episode’s end (Final Distance). Tab. A.3 and Tab. A.4 supplement the results reported in Tab. 1 and Tab. 2, respectively.

| Training Routine                  | MD-SPL↑ | Success↑ | Ep Length↓ | Invalid Prob Mass↓ | Final Distance↓ |
|-----------------------------------|---------|----------|------------|--------------------|-----------------|
| DirectPix                         | 0.0     | 0.8      | 249.3      | 0.150              | 3.42            |
| GRIDToPix                         | **4.0** | **24.6** | **210.7**  | **0.110**          | **2.848**       |
| GRIDToPix → DirectPix             | 3.1     | 14.5     | 224.2      | 0.116              | 3.215           |
| Gridworld expert (upper bound)    | 19.2    | 56       | 139.0      | 0.077              | 1.943           |

Table A.3. Additional quantitative rewards (terminal reward structure). This table supplements results reported in Tab. 1. In addition to metrics of MD-SPL and success rate, we include other relevant metrics. Vertical arrows, i.e., ↑ and ↓ denotes whether larger or smaller metric values are preferred, respectively.

| Training Routine                  | MD-SPL↑ | Success↑ | Ep Length↓ | Invalid Prob Mass↓ | Final Distance↓ |
|-----------------------------------|---------|----------|------------|--------------------|-----------------|
| DirectPix                         | 11.2    | 58.4     | 155.7      | 0.311              | 1.154           |
| GRIDToPix                         | 9.7     | 62.0     | 154.6      | 0.264              | 1.17            |
| GRIDToPix → DirectPix             | **15.3**| **68.6** | **133.6**  | **0.213**          | **0.826**       |
| Gridworld expert (upper bound)    | 22.2    | 76.3     | 109.7      | 0.275              | 0.722           |

Table A.4. Additional quantitative rewards (shaped reward structure). This table supplements results reported in Tab. 2.

A.4. Gridworld Encoder and Implementation Details

The models utilized for visual and gridworld agents are alike. Particularly, we make minimal edits to adapt the observation encoder to be able to process gridworld tensors instead of RGB tensors. This is briefly summarized below. PointNav. For the PointNav, the visual encoder transforms a (3, 256, 256) RGB tensor into a feature of length 512 via three convolutional blocks and a linear layer. The grid encoder transforms a (1, 100, 100) top-down tensor into a feature of the same length as the visual counterpart (512) via four convolutional blocks and a linear layer. For both visual and gridworld agents, the policy is represented via a GRU of hidden size 512 followed by linear layers to serve as actor and critic heads. FurnMove. Similarly as for PointNav, for FurnMove the visual encoder transforms a (3, 84, 84) RGB tensor into a feature of length 512 via five convolutional blocks (no linear layers). The grid encoder transforms a (9, 15, 15) top-down tensor into a feature of length 512 via four convolutional blocks (for consistency, no linear layers). For both visual and gridworld agents in FurnMove task, the policy is represented via a LSTM of hidden size 512 followed by linear layers to serve as actor and critic heads. Note that we use the (best-performing) mixture-of-marginals actor head introduced in [29] for all FurnMove experiments.

3 vs. 1 with Keeper. For the 3 vs. 1 with Keeper task, the (3, 1280, 960) RGB tensor is scaled to a (1, 96, 72) gray-scale image. The visual encoder transforms this (1, 96, 72) tensor into a feature of length 512 via three convolution layers and a linear layer. The gridworld model observes the state as a vector of length 572 and transforms it to a feature of length 64 using a three linear layers. For both visual and gridworld agents, linear layers on top of the extracted feature serve as actor and critic heads.

A.5. Training Routines

In the main paper we compare our GRIDToPix and GRIDToPix→DIRECTPix routines with DirectPix (Sec. 5.1). Here, we include different exploration policies for imitation learning. We also include additional details of teacher forcing [76, 7] and IL → RL transition (GRIDToPix→DIRECTPix).

Exploratory policies $\mu$. Recall, from Sec. 3.3, the GRIDToPix loss is

$$L_{\text{GRIDToPix}} = E[\mathbb{E}_{\alpha \sim \pi^V(\cdot | O_\mu, H_\mu)} [-\log \pi^V(\alpha | O_\mu, H_\mu)]] .$$ (5)
The choice of exploratory policy \( \mu \) leads to three variants, each widely adopted in IL tasks:

- **Student forcing (SF):** This is an on-policy method, i.e., the target policy \( \pi^\nu \) to be learnt is the exploration policy \( \mu \).

- **Teacher forcing (TF):** In this case we use \( \mu = \pi^\nu \), i.e., the visual agent takes the expert’s actions. This helps the visual agent frequently observe states closer to the goal, which it would only see late in training if following SF. The downside of TF: the visual agent \( \mathcal{V} \) observes only states which meaningfully lead to the goal. Hence, TF is susceptible to covariate shift, i.e., the visual agent \( \mathcal{V} \) exhibits low resilience to recover if it ventures ‘off-track.’

- **Annealed teacher forcing or DAgger (DA):** Agents take actions by combining SF and TF via \( a_t^\nu = (1 - \beta) a_t^{SF} + \beta a_t^{TF} \) where \( \beta \sim \text{Bernoulli}(p) \). A decay of \( p \) from 1 to 0 is adopted during training to transition smoothly from TF to SF. After such annealing, the visual agent’s policy \( \pi^\nu \) is generally trained with SF until convergence.

**Teacher forcing probability.** Details of teacher forcing for our methods are summarized in Tab. A.5. We employ annealed teacher forcing (see Sec. 3.3) for a part of the training budget. We anneal (decay) the teacher forcing probability linearly for PointNav, FurnMove, and exponentially for the 3 vs. 1 with Keeper.

The GRIDToPIX routine is purely IL and is denoted in Tab. A.5 with an IL arrow (\( \leftarrow \rightarrow \)) spanning from 0% to 100%. In contrast, GRIDToPIX→DIRECTPix includes a warm-start with IL followed by reward-based learning. Hence, in Tab. A.5, we have a shorter IL arrow followed by an RL arrow (\( \leftarrow \rightarrow \)) along with the algorithm used to maximize rewards.

| Task                | GRIDToPIX | GRIDToPIX→DIRECTPix |
|---------------------|-----------|---------------------|
| PointNav            | ![Diagram](image1) | ![Diagram](image2) |
| FurnMove            | ![Diagram](image3) | ![Diagram](image4) |
| 3 vs. 1 with Keeper | ![Diagram](image5) | ![Diagram](image6) |

Table A.5. *Training routines.* The figures show how teacher forcing probability varies over training and the transition from IL to RL.

### A.6. Visual Predator-Prey – Task Where Reward Shaping Is Not Enough

Predator-prey is a multi-agent task defined within the OpenAI multiple-particle environments (MPE) [46]. It entails controlling a team of predators while a competing team of prey is controlled by the game engine. Specifically, \( n \) predators work together to chase a team of \( n/3 \) prey that move faster than the predators. The objective is to optimize rewards by controlling the policies of the predators. Prior work [46, 43, 33] assumes an agent observes a 1D vector summarizing the positions and velocities of all agents in a neighborhood. For consistency, we consider this 1D vector to be our gridworld observation. In addition, akin to [41], we define an analogous visual setting where agents only process a top-down map in pixel space.

Note, for visual tasks like PointNav, FurnMove, and 3 vs. 1 with Keeper, we created (or leveraged existing) gridworlds. As standard visual tasks are mostly navigational, reward shaping is typically tractable. Hence, to build a testbed where reward shaping isn’t tractable, we are effectively creating the visual world for the complex, multi-agent predator-prey task.

Below, we include experimental details and corresponding results. Note, despite basic reward shaping, DirectPix agents
cannot learn whereas GridToPix comes close to the upper bound of gridworld experts.

**Gridworld observation.** The predator observes its location and velocity, the relative location of the neighboring landmarks and fellow predators, and the relative location and velocity of the three preys.

**Shaped rewards.** Due to the complexity of the task, it’s intractable to perfectly shape the reward. However, a positive reward for bumping into a prey and a negative reward based on the distance to the prey is provided.

**Model architecture.** We use a standard CNN model [48]. Our model has four hidden layers. The first three layers are convolutional layers with 32, 64, and 64 filters. All convolutional layers have filters of size $4 \times 4$ and stride 2. The fourth layer is a fully connected layer with 256 hidden units. Following the last hidden layer are linear layers that predict the value and actor policy.

**Evaluation.** Agents are evaluated using the average rewards obtained in test episodes. We train the agents for 3M environment steps. Learning curves are reported on test episodes.

**Results.** We experiment with number of predators $n = \{3, 6\}$ visual predator-prey tasks. The learning curves for $n = 3$ and setup for $n = 6$ are illustrated in Fig. A.1. The average rewards for DirectPix, GridToPix, GridToPix→DirectPix, and gridworld experts are included in Tab. A.6. Despite basic reward shaping, joint optimization of perception and planning leads to a DirectPix policy demonstrating no meaningful behaviour. With the help of self-supervision via the gridworld expert, GridToPix and GridToPix→DirectPix perform significantly better with 48.7 and 37.9 average rewards over the training budget of 3M steps. Results on the $n = 6$ setting show a similar trend: DirectPix and GridToPix obtain average rewards of $-70.2$ and $237.1$ (the gridworld expert earns $281.1$).

### A.7 Qualitative Results of DirectPix Trained with Terminal Rewards

As we report in Sec. 5, DirectPix doesn’t learn a meaningful policy in any of the tasks when given only terminal rewards. Closer inspection reveals that the DirectPix agent for the PointNav task learns a degenerate probability distribution with almost all probability mass allocated to the ‘Stop’ action. Similarly, many of the strategies learned by DirectPix agents for 3 vs. 1 with Keeper are also myopic. Particularly, the agents cannot effectively pass to each other, pushing the ball outside the field lines (see Fig. A.2). Another common failure mode is shooting at the goal from too far off (see Fig. A.3).
Figure A.2. **Failure modes – Misdirected passes.** Two episodes (top strip and bottom strip) that highlight a common failure mode of DirectPix training with terminal rewards. Particularly, player 1 attempts to pass the ball to player 2 but misdirects it to hit outside the field lines. For readability, the relevant players are marked as P1 and P2. Also, ball is highlighted using a yellow arrow.

Figure A.3. **Failure modes – Long distance shooting.** Two episodes (top strip and bottom strip) that highlight a common failure mode of DirectPix training with terminal rewards. Particularly, in the top strip, player 1 attempts to score by shooting too ambitiously from the starting point. In the bottom strip, player 2 does the same, after receiving the ball from player 1. In both episodes the goal keeper can easily intercept the ball.