GAME-GUIDED SKILL DISCOVERY THROUGH SELF-PLAY FOR PLAYABLE AGENT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

We present **Game-Guided Skill Discovery (GGSD)**, a framework that uses self-play in games to discover motor skills that are directly playable by humans. Playable skills provide a compact abstraction for controlling embodied agents through a small set of learned behaviors rather than low-level actions. To be effective, these skills should be semantically distinct, interpretable, and expressive—properties that existing unsupervised skill-discovery methods often fail to achieve simultaneously. GGSD achieves these desiderata by grounding skill discovery in competitive gameplay. A hierarchical agent competes against its past selves, with a high-level policy selecting from a small discrete skill set and a skill-conditioned low-level policy learning the corresponding behaviors. After training, a human can replace the high-level policy and directly control the agent through the same discrete skills. Despite the small number of high-level actions, skill transitions give rise to emergent *combo behaviors*, expanding expressivity beyond individual primitives. Across Ant, Franka-arm, and Unitree G1 environments, we show that GGSD produces human-playable skills that humans can compose to solve unseen tasks, such as `Maze` and `CubePush`, without additional training. An interactive demo is available at `https://ggsd-demo.github.io`.

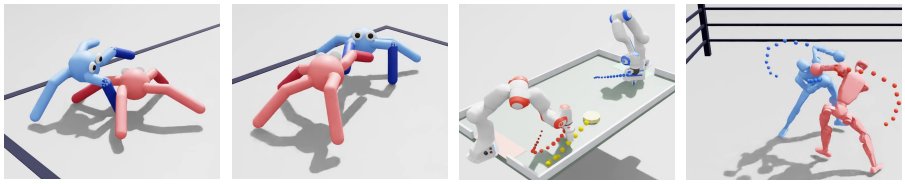


Figure 1: Different game rules guide agents toward different playable skills. Shown are gameplay snapshots of agents learned with GGSD.

1 INTRODUCTION

Learning a compact repertoire of reusable motor skills is a long-standing goal in skill discovery. Such skills can provide useful abstractions not only for downstream policy learning, but also for direct human control: if complex behaviors are organized into a small set of intuitive skills, a human can select and compose them without specifying low-level actions. For this interface to be effective, the discovered skills should be semantically distinct, interpretable, scalable to high-degree-of-freedom agents, and sufficiently expressive despite a compact action space.

Existing unsupervised skill-discovery methods do not naturally guarantee these properties. Most approaches optimize objectives based on mutual information (MI) (Gregor et al., 2016; Sharma et al., 2019; Eysenbach et al., 2019; Kwon, 2020; Laskin et al., 2022) or Wasserstein dependency measures (WDM) (Park et al., 2021; 2024), encouraging different skills to induce distinguishable state distributions or trajectories. While this can produce interpretable behaviors in simple environments, increasing agent complexity introduces many ways for skills to differ without being behaviorally meaningful. As a result, distinctiveness alone can yield skills that are easy to distinguish but difficult to interpret or reuse.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We argue that *competitive games* provide a natural source of structure for discovering such skills, as illustrated across diverse embodiments in Figure 1. A simple game rule specifies what constitutes success while leaving open how success should be achieved. Through self-play, evolving opponents continually expose the agent to new strategic and physical situations, encouraging the emergence of useful behaviors without requiring users to specify individual skills a priori. Self-play has long been shown to induce emergent behaviors, from superhuman strategies in competitive games Silver et al. (2016; 2018); Vinyals et al. (2019); Baker et al. (2020); Oh et al. (2021) to structured motor behaviors in physically embodied agents Bansal et al. (2018); Haarnoja et al. (2024); Jansonnie et al. (2024). We ask whether this same mechanism can be used to discover a compact repertoire of skills that is directly playable by humans.

Based on this idea, we introduce **Game-Guided Skill Discovery (GGSD)**, a framework that uses self-play in simple 1v1 competitive games to discover playable motor skills. Each agent is controlled by a hierarchical policy. A high-level policy selects from a small set of discrete skills, while a skill-conditioned low-level policy maps the selected skill to motor actions. The game objective encourages behaviors that are useful for winning, while a mutual-information objective encourages different skill codes to acquire distinct behavioral semantics.

After training, we remove the high-level policy and map each skill to a human input, such as a keyboard button, allowing users to directly control the agent without training a new task-specific controller. We intentionally keep the number of skills small (only five or six) in all environments to maintain a simple interface for human play. Despite this compact action vocabulary, the controller gains additional expressivity through *skill transitions*: executing one skill can place the agent in a state from which another skill produces a qualitatively different behavior, giving rise to emergent *combo behaviors*. Similar to button combinations in commercial games, these transitions allow a small set of discrete skills to support behaviors richer than the individual primitives alone.

As a result, GGSD discovers semantically diverse and human-interpretable skills even for high-degree-of-freedom embodiments such as humanoids. These skills are also directly reusable beyond the games in which they are learned. Without any additional policy training, humans can compose them to solve previously unseen downstream tasks, including complex locomotion in `Maze` and object interaction in `CubePush`.

The core contributions of our work are as follows:

- We introduce GGSD, a skill-discovery framework that uses self-play in games as lightweight guidance for learning human-playable motor skills.
- We show that GGSD discovers semantically distinct and human-interpretable skills that scale to high-degree-of-freedom agents. Despite using only a small discrete skill set, transitions between skills give rise to emergent *combo behaviors* that substantially expand the controller's expressivity.
- We demonstrate GGSD across Ant, Franka Arm, and Unitree G1 environments, and show that humans can directly compose the learned skills to solve previously unseen locomotion and object-interaction tasks without additional policy training.

2 Related Works

2.1 Self-Play for Motor Skill Learning

Fictitious play provides a classical mechanism for stabilizing self-play by repeatedly learning best responses to the empirical average of opponents' historical strategies (Brown, 1951; Robinson, 1951). Fictitious self-play extends this idea to extensive-form games (Heinrich et al., 2015), while Neural Fictitious Self-Play (NFSP) approximates best responses with deep reinforcement learning and historical average strategies with supervised learning (Heinrich & Silver, 2016). These methods primarily aim to learn equilibrium strategies. Similarly, our method trains against a distribution of historical policies. Competing against a diverse pool of past selves encourages the agent to develop reusable skills that remain useful across a wide range of opponents rather than specializing to a particular strategy.

Consistent with this intuition, self-play and competitive interaction have been shown to induce complex motor behaviors in physically simulated agents. Bansal et al. (2018) demonstrated the emergence of behaviors such as running, blocking, tackling, and kicking through multi-agent competition, while later works extended competitive learning to bipedal soccer (Haarnoja et al., 2024), robotic manipulation (Jansonnie et al., 2024), and hierarchical multi-drone volleyball (Zhang et al., 2025). Won et al. (Won et al., 2021) studied high-DoF humanoids in boxing and fencing, but learned the underlying motor skills from reference motions before training competitive strategies.

In contrast, we use self-play itself as guidance for *discovering* the low-level skill repertoire. Rather than learning a monolithic game-playing policy or relying on predefined/reference-based motor skills, our method explicitly organizes behaviors induced by competition into discrete, diverse, and reusable skills that can be directly controlled by humans.

2.2 Guidance in Skill Discovery

Several works introduce external guidance to address a key limitation of unsupervised skill discovery, as optimizing solely for distinctiveness or state coverage can produce behaviors that are diverse but semantically meaningless. Language-Guided Skill Discovery (Rho et al., 2025b) uses LLM-generated state descriptions to encourage semantically distinct skills. However, obtaining language descriptions throughout the explored state space becomes increasingly impractical as agent dimensionality grows. DoDont (Kim et al., 2024) uses desirable and undesirable demonstrations, while Reference Grounded Skill Discovery (Rho et al., 2026) uses reference motions to guide the learned skill repertoire. While effective for high-DoF agents, the discovered repertoire is largely shaped by the provided motion dataset, and preparing sufficiently diverse reference motions can itself be costly. In contrast, guidance from simple game rules is lightweight. A game's rules can be specified in only a few lines of code, while self-play autonomously discovers how to succeed under those rules. This allows the agent to discover novel behaviors beyond explicitly provided examples, while remaining scalable to high-DoF systems.

3 Preliminaries

Markov Decision Process. We first consider a standard Markov decision process (MDP), defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s' \mid s, a)$ is the transition probability, $r(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi(a \mid s)$ is trained to maximize the expected discounted return

$$J(\pi) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{1}$$

Two-player games. We consider a two-player game between an agent and an opponent. The game state is given by $s = (s^{\text{me}}, s^{\text{foe}})$, where s^{me} denotes the agent's own state, and s^{foe} denotes the state of the opponent. Similarly, the two players take actions $a^{\text{me}} \in \mathcal{A}^{\text{me}}$ and $a^{\text{foe}} \in \mathcal{A}^{\text{foe}}$. The game dynamics are described by

$$P_{\text{game}}\left(s' \mid s, a^{\text{me}}, a^{\text{foe}}\right), \tag{2}$$

and the controlled agent receives a game reward $r^{\text{me}}(s, a^{\text{me}}, a^{\text{foe}})$.

When the opponent follows a fixed policy $\pi_{\text{foe}}(a^{\text{foe}} \mid s)$ over continuous actions, its action can be marginalized into the environment dynamics. In particular, we define the induced transition probability

$$\tilde{P}_{\pi_{\text{foe}}}(s' \mid s, a^{\text{me}}) = \int_{\mathcal{A}^{\text{foe}}} \pi_{\text{foe}}(a^{\text{foe}} \mid s) P_{\text{game}}\left(s' \mid s, a^{\text{me}}, a^{\text{foe}}\right)\, da^{\text{foe}}, \tag{3}$$

with the corresponding expected reward

$$\tilde{r}_{\pi_{\text{foe}}}(s, a^{\text{me}}) = \int_{\mathcal{A}^{\text{foe}}} \pi_{\text{foe}}(a^{\text{foe}} \mid s) r^{\text{me}}(s, a^{\text{me}}, a^{\text{foe}})\, da^{\text{foe}}. \tag{4}$$

Therefore, for a fixed opponent policy, the two-player game reduces to an ordinary single-agent MDP

$$\mathcal{M}_{\pi_{\text{foe}}} = \left(\mathcal{S}, \mathcal{A}^{\text{me}}, \tilde{P}_{\pi_{\text{foe}}}, \tilde{r}_{\pi_{\text{foe}}}, \gamma\right). \tag{5}$$

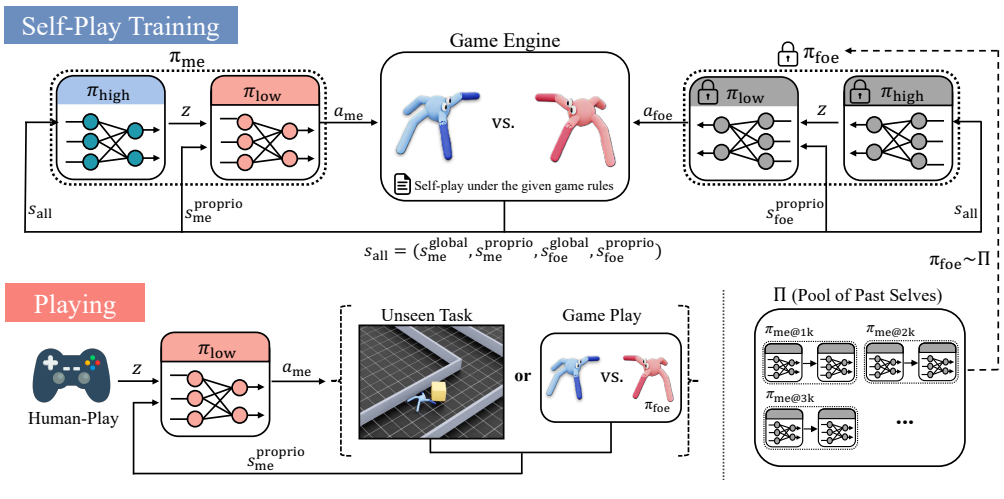


Figure 2: Overview of GGSD. **Top:** We jointly learn a hierarchical policy through self-play against opponents sampled from a pool of past checkpoints. **Bottom:** After training, a human can replace the high-level policy and directly control the agent across diverse scenarios.

From the perspective of the controlled agent, training against a fixed opponent is thus equivalent to standard reinforcement learning in an environment whose dynamics implicitly include the opponent's behavior. Self-play can be viewed as repeatedly changing this MDP as the opponent policy evolves over the course of training.

4 Game-Guided Skill Discovery

GGSD converts game rules into human-playable skills through self-play. We first describe our self-play procedure, then introduce hierarchical skill learning, and finally explain how the learned skills are made directly playable by humans.

4.1 Self-Play with a Pool of Past Selves

A straightforward form of self-play trains against a copy of the current policy. However, as the policy is updated, the opponent changes simultaneously, resulting in a continuously moving training objective. Instead, as illustrated in Figure 2, we maintain a pool of historical policy checkpoints and sample a fixed opponent at the beginning of each episode.

Recall from Sec. 3 that a fixed opponent policy π_{foe} induces a single-agent MDP $\mathcal{M}_{\pi_{\text{foe}}}$. We maintain an opponent pool

$$\Pi_K = \{\pi^{(1)}, \ldots, \pi^{(K)}\}, \tag{6}$$

where $\pi^{(K)}$ denotes the most recent checkpoint. When the pool size K is greater than 1, we sample an opponent according to

$$\rho_K(k) = \begin{cases} p, & k = K, \\ \dfrac{1-p}{K-1}, & 1 \le k < K, \end{cases} \tag{7}$$

and keep it fixed throughout the episode. Thus, the latest checkpoint is sampled with probability p, while the remaining probability is distributed uniformly across earlier checkpoints.

This results in an opponent distribution that is piecewise stationary rather than continuously changing, while still maintaining competitive pressure from recent opponents. The parameter p controls the balance between these two effects: larger values place greater emphasis on the latest opponent, while smaller values increase diversity from past strategies. We use $p \in [0.6, 0.8]$ in all experiments.

Our procedure is inspired by fictitious play (Brown, 1951; Robinson, 1951) and its self-play variants (Heinrich et al., 2015; Heinrich & Silver, 2016). While we do not implement exact fictitious play,

we follow its central principle of training against a distribution of past strategies, with additional emphasis on recent opponents.

4.2 Hierarchical Skill Learning

Jointly learning strategies and motor skills. GGSD jointly learns the high- and low-level policies during self-play. The high-level policy determines *which* skill to execute as part of the current game strategy, while the low-level policy determines *how* that skill is physically realized. Joint training allows the two levels to co-evolve: improved strategies expose the agent to new situations that call for new low-level skills, while an expanding skill repertoire enables more sophisticated strategies.

This joint training is implemented through a temporal hierarchy. Let $s_t^{\text{all}} = (s_t^{\text{me}}, s_t^{\text{foe}})$ denote the full game state, and let $\tau_j = jk$ denote the j-th high-level decision time.

The high-level policy selects a discrete skill

$$z_j \sim \pi_{\text{high}}\left(z \mid s_{\tau_j}^{\text{all}}\right), \tag{8}$$

which is held fixed for the following k environment steps. Within this interval, the low-level policy produces a motor action at every step,

$$a_t \sim \pi_{\text{low}}\left(a \mid s_t^{\text{proprio}}, z_j\right), \qquad \tau_j \le t < \tau_j + k, \tag{9}$$

where s_t^{proprio} contains only the controlled agent's proprioceptive observations.

Hierarchical policy optimization. During training, we retain the sampled skill variables and optimize the joint distribution over the augmented trajectory. For one skill interval, its policy-dependent probability factorizes as

$$p_\theta\left(z_j, a_{\tau_j:\tau_j+k-1} \mid s_{\tau_j:\tau_j+k-1}\right) = \pi_{\text{high}}\left(z_j \mid s_{\tau_j}^{\text{all}}\right) \prod_{t=\tau_j}^{\tau_j+k-1} \pi_{\text{low}}\left(a_t \mid s_t^{\text{proprio}}, z_j\right). \tag{10}$$

Thus, the joint log likelihood decomposes into high- and low-level terms, allowing the two policies to be optimized at their respective temporal resolutions. We use separate PPO objectives for both levels, following prior work on joint hierarchical policy optimization (Li et al., 2020).

Each level maintains its own value function and reward. The high-level policy is optimized only for the game objective, since it is responsible for strategic skill selection. The low-level policy additionally receives the mutual-information reward r_t^{MI} defined in Eq. 15, which encourages different skill codes to acquire distinct motor behaviors:

$$r_t^{\text{high}} = r_t^{\text{game}}, \qquad r_t^{\text{low}} = r_t^{\text{game}} + \lambda_{\text{MI}} r_t^{\text{MI}}. \tag{11}$$

We view the game objective as providing the behavioral guidance that shapes which skills emerge, while MI primarily associates these behaviors with distinct skill codes; see Appendix B for further discussion.

The low-level critic $V_{\text{low}}(s_t^{\text{all}}, z_j)$ operates at every environment step. The high-level critic operates only at skill-selection steps, and is trained using target values calculated according to:

$$y_j^{\text{high}} = \sum_{i=0}^{k-1} \gamma^i r_{\tau_j+i}^{\text{game}} + \gamma^k V_{\text{high}}\left(s_{\tau_{j+1}}^{\text{all}}\right). \tag{12}$$

Thus, the high-level policy uses an effective discount factor of γ^k between consecutive skill decisions, while the low-level policy uses the original discount factor γ at every environment step. We use $k = 10$ in all experiments.

4.3 Playable Learned Skills

Having described how the two levels are jointly trained, we now introduce the design choices that allow the learned low-level skills directly playable by humans.

Discrete and diverse skills. We use a discrete skill variable $z \in 1, \ldots, M$, represented as a one-hot vector, with $M = 5$ or 6 depending on the environment. This compact skill set provides a simple interface for human control, while state-dependent skill transitions enable emergent *combo behaviors* (Sec. 5.3).

To prevent the high-level policy from collapsing to a small subset of skills, we regularize it with categorical entropy:

$$\mathcal{H}_{\text{high}} = \mathbb{E}_{s^{\text{all}}} \left[\mathcal{H} \left(\pi_{\text{high}}(\cdot \mid s^{\text{all}}) \right) \right]. \tag{13}$$

This encourages the policy to leverage the full discrete skill set during gameplay.

Mutual-information maximization. High-level entropy encourages diverse skill *selection*, but does not ensure that different codes correspond to distinct behaviors. We therefore maximize the mutual information between the selected skill Z and the resulting state S. A discriminator $q_\phi(z \mid s)$ gives the variational lower bound

$$I(Z; S) \geq \mathcal{H}(Z) + \mathbb{E}_{p(z,s)} \left[\log q_\phi(z \mid s) \right]. \tag{14}$$

Although the skill distribution is induced by the high-level policy, it is fixed with respect to θ_{low} during the low-level update. The discriminator therefore provides the intrinsic reward

$$r_t^{\text{MI}} = \log q_\phi(z_j \mid s_t). \tag{15}$$

Together, high-level entropy encourages the use of multiple skills, while the MI objective encourages different skill codes to induce distinguishable behaviors.

Opponent-agnostic low-level control. For direct human control, each skill should retain consistent motor semantics across different game situations. To encourage this, we restrict both the low-level policy and the discriminator to proprioceptive observations, while allowing the high-level policy to observe the full game state:

$$\pi_{\text{high}} \left(z \mid s^{\text{all}} \right), \qquad \pi_{\text{low}} \left(a \mid s^{\text{proprio}}, z \right), \qquad q_\phi \left(z \mid s^{\text{proprio}} \right). \tag{16}$$

This places opponent-dependent strategic reasoning in the high-level controller and prevents the discriminator from distinguishing skills based on opponent states, encouraging low-level skills to retain consistent motor semantics that are easier for humans to control.

Putting these components together, the two policies optimize the following objective:

$$\begin{aligned} \theta_{\text{high}}^* &= \arg\min_{\theta_{\text{high}}} \mathcal{L}_{\text{high}}^{\text{PPO}} \left(r^{\text{game}} \right) - \lambda_{\text{H}} \mathcal{H}_{\text{high}}, \\ \theta_{\text{low}}^* &= \arg\min_{\theta_{\text{low}}} \mathcal{L}_{\text{low}}^{\text{PPO}} \left(r^{\text{game}} + \lambda_{\text{MI}} r^{\text{MI}} \right). \end{aligned} \tag{17}$$

Pseudocode for GGSD is provided in Algorithm 1, and the full hyperparameters in Appendix D.

5 Experiments

5.1 Implementation Details

Game Rules. GGSD transforms a given game rule into a set of playable motor skills. We consider four games with distinct objectives and embodiments. (1) In `AntSumo`, an agent wins by pushing its opponent out of the arena, while falling results in a loss. (2) In `AntFencing`, an agent wins by touching the opponent's root body with one of its front legs; falling or leaving the arena results in a loss. (3) In `FrankaAirHockey`, two Franka robot arms compete to strike a puck into the opponent's goal. (4) In `G1Boxing`, two humanoid robots compete by striking the opponent's head or torso, with the objective of inflicting more damage than the opponent. Detailed reward terms, observations, and environment configurations for each game are provided in the Appendix C.

Self-Play. Sampling a separate opponent policy for each of the 4,096 environments is prohibitively expensive in GPU memory, so we use grouped opponent sampling. We divide the 4,096 parallel environments into 10 groups, each of which samples one opponent policy shared by all environments in the group. Thus, training requires only the current policy and 10 frozen opponent policies to reside on the GPU simultaneously. We add the current policy to the opponent pool every 2,000 updates and resample each group's opponent every 200 updates.

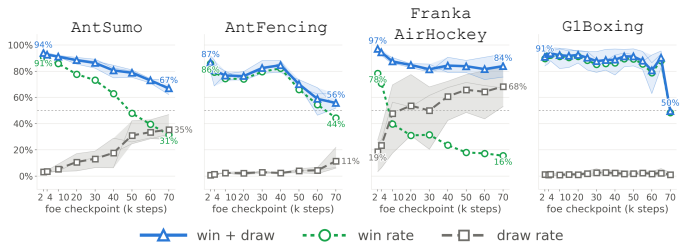


Figure 3: Win and draw rates of the final policy against itself and its previous checkpoints.

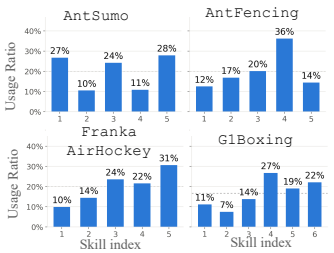


Figure 4: Skill usage ratios, showing no collapse.

5.2 GGSD Learns to Play Games Well

We first examine whether our self-play procedure is stable and whether it produces agents that can effectively play the given games. Figure 3 reports the performance of the final policy, trained for 70k updates, against checkpoints from 2k to 70k updates. For each of three independent training runs, we evaluate each checkpoint pair over 1,000 games and average the results.

Self-play produces increasingly competitive agents. The final policy achieves high win rates against policies from early stages of training. As the opponent checkpoint becomes more recent, the win rate decreases while the draw rate increases, indicating that the policies become increasingly competitive. Importantly, the combined win-and-draw rate remains above 50% against every evaluated checkpoint. For `G1Boxing`, the final policy achieves over 80% win rate against every past checkpoint except itself at 70k, suggesting that training has not yet fully saturated. As training approaches a plateau, stronger recent opponents should lead to a more gradual decline in win rate.

The learned agent also performs strongly against human players. Four users each played five games against the final 70k checkpoint on `AntSumo`, achieving only two human wins (10% human win rate). Overall, these results suggest that training improves the policy against the opponent pool as a whole, rather than overfitting to a particular opponent checkpoint.

Figure 4 further shows how frequently each learned skill is selected during gameplay. Across all games, the high-level policy does not collapse to a single skill; instead, it consistently makes use of multiple skills. This indicates that the discovered skill set remains behaviorally relevant during competitive play.

5.3 Qualitative Analysis of Learned Skills

We next examine what motor behaviors emerge from GGSD. Figure 5 visualizes all skills learned in each game. For visualization, we fix a single skill at the beginning of an episode, execute it continuously for five seconds, and overlay snapshots of the resulting motion.

GGSD discovers highly interpretable skills. The learned skills exhibit clear and readily interpretable behavioral semantics. For `AntSumo`, the agents discover a variety of turning, locomotion, and pushing behaviors. In `G1Boxing`, the humanoid learns basic locomotion primitives as well as task-specific behaviors such as punching and guarding. For example, one skill raises the arms in front of the upper body, resembling a defensive guard against incoming punches.

These qualitative patterns are consistent across random seeds. Although different seeds do not necessarily produce identical skill sets, they repeatedly recover behaviors that are important for successful gameplay, such as locomotion and striking. This suggests that the game objective provides a consistent behavioral structure while still allowing multiple solutions to emerge.

Skill transitions produce emergent combo behaviors. A particularly interesting phenomenon is that meaningful behaviors can emerge not only from individual skills but also from transitions between them. This is especially apparent in `FrankaAirHockey`. Individual Franka skills are relatively static: when executed in isolation, each skill moves the end effector toward a particular re-

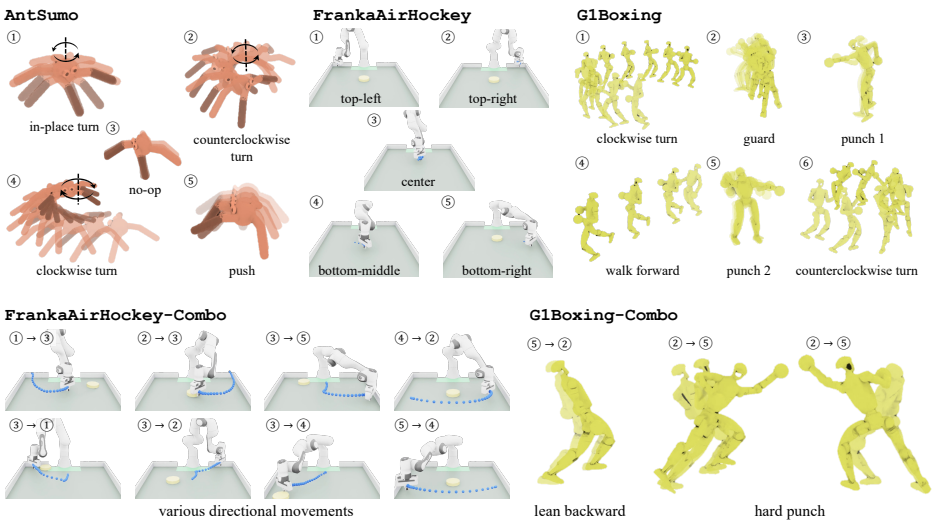


Figure 5: Learned motor skills (top) and combo behaviors through skill transitions (bottom).

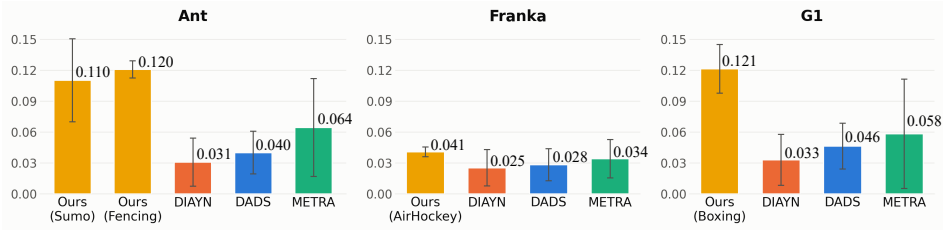


Figure 6: Semantic diversity of learned skills across different embodiments. Higher is better.

gion of the table and then remains there. However, transitions between these discrete skills generate rapid and purposeful movements that are used to strike the puck.

A similar effect appears in `G1Boxing`. For example, transitioning from skill 5 to skill 2 produces a backward lean used to evade or absorb punches, whereas the reverse transition produces a strong forward punch by rapidly shifting momentum. These examples show that discrete skills act as compositional building blocks whose transitions produce richer behaviors than the individual skills alone.

State-dependent skill semantics. Emergent combos reveal an interesting tension between semantic consistency and compositional expressivity: the effect of a skill can depend on the state induced by preceding skills. Nevertheless, the resulting controllers remain playable, suggesting that consistency need not hold globally. Instead, each skill can retain predictable semantics within a small number of state regimes, while transitions between regimes enable qualitatively different behaviors. This expands the expressivity of a compact skill set without increasing the number of human inputs, although too many regimes could make the interface difficult to interpret. We discuss this further in Appendix A.

5.4 Quantitative Analysis Against Baselines

We quantitatively evaluate how effectively game-based guidance promotes *semantically distinct* skills. Distinctiveness alone is insufficient. Two skills may be distinct while both lack clear semantic meaning.

We therefore adapt the *language distance* metric of Rho et al. (2025b) to a vision-language setting. For each skill, we sample rollout videos at 5 fps, use `Gemini-3.6-Flash` to generate a natural-language behavior description, and compute pairwise distances between skill descriptions

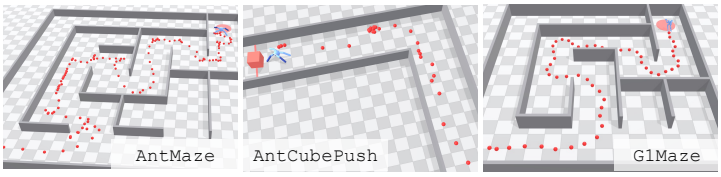


Figure 7: Successful human-play trajectories on unseen tasks. Red dots indicate cube in `AntCubePush`, and robot in `Maze`.

Table 1: Human play results.

Task	Succ.	Time(s)
AntMaze	100%	82.7
AntCube	100%	62.2
G1Maze	84%	22.0

using `Gemini-Embedding-2`. For each training run, we average these pairwise distances to obtain a semantic-diversity score, and report the mean across three runs. Results are averaged over three seeds, with prompting and evaluation details provided in the Appendix E. We compare against DIAYN, DADS (Sharma et al., 2019), and METRA (Park et al., 2024). Since METRA uses a continuous skill space, we train it with a 5-dimensional skill and evaluate five skills corresponding to the one-hot basis vectors.

GGSD learns semantically distinct skills. As shown in Figure 6, GGSD achieves the highest semantic diversity across all embodiments. It is approximately $2\times$ higher than the strongest baseline on Ant and G1, and 28% higher on Franka. These results indicate that game-based guidance promotes not only diverse motions, but semantically distinct behaviors.

5.5 Human Play on Unseen Downstream Tasks

Finally, we evaluate whether skills learned through gameplay can be directly reused by humans on unseen tasks. We keep the learned low-level policies fixed and replace the high-level policy with human input, without additional training.

Using the policy learned from `AntSumo`, we evaluate `AntMaze` and `AntCubePush`; we also evaluate the `G1Boxing` policy on `G1Maze`. In all tasks, users observe a top-down view and select among the learned skills in real time.

GGSD skills are directly playable by humans. We conducted a human evaluation with five participants, including two authors, each performing five trials per task.[1] Before evaluation, participants read brief control tips and practiced for at most five minutes. Table 1 reports the success rate and average completion time across all participants and trials. The average success rate is at least 84% on all three tasks. Figure 7 shows representative trajectories from the user study. `AntCubePush` requires object manipulation and the maze tasks introduce wall contacts, neither of which is present in the training games. Moreover, all policies are trained in Isaac Lab (Mittal et al., 2025) but evaluated in MuJoCo (Todorov et al., 2012) web. Despite these task and simulator shifts, humans can compose a small set of GGSD skills to reliably navigate and manipulate objects without additional policy optimization.

6 Conclusion

We investigated whether self-play in 1v1 competitive games can produce reusable motor skills that are directly playable by humans. The learned skills support interactive gameplay and can be reused by humans without additional training to solve unseen downstream tasks. Our results suggest that games provide a simple form of guidance for discovering diverse, reusable, and human-playable skills.

Limitations and future work. A central limitation of GGSD is that the learned skill repertoire is bounded by the skills required by the game. Simple games may demand only a narrow set of behaviors, limiting transfer to downstream tasks. For example, a policy trained on `G1Boxing` is unlikely to acquire dexterous manipulation skills. Scaling GGSD to games that require broader and more diverse motor capabilities may enable more generic skill repertoires that transfer across a wider range of downstream tasks, which we view as an important direction for future work.

[1] This human evaluation was reviewed and approved by an Institutional Review Board (IRB).

AI USE STATEMENT

In this work, we used generative AI tools to assist with experimental code development, language polishing, and literature search. We did not use generative AI tools to develop the core research ideas, decide which experiments to conduct, determine how the results should be interpreted or presented, write the initial manuscript draft, or produce and polish the visualizations of the experimental results.

All AI-assisted code was reviewed and tested by the authors, literature identified with AI assistance was manually verified against the original sources, and the AI-assisted mathematical derivation was independently checked by the authors for correctness. All AI-assisted text was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made extensive efforts to facilitate the reproducibility of our work across multiple dimensions:

- A complete pseudocode description of our algorithm is provided in Algorithm 1.
- The full set of observations and rewards used in our experiments is detailed in Appendix C.
- Full hyperparameters used in our experiments are provided in Appendix D.
- An interactive web demo is available at `https://ggsd-demo.github.io`, allowing readers to directly interact with and evaluate our trained agents.
- The full codebase will be publicly released upon acceptance.

REFERENCES

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2020.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*, 2018.

George W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans (ed.), *Activity Analysis of Production and Allocation*. Wiley, New York, 1951.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H. Huang, Dhruva Tirumala, Jan Humplik, Markus Wulfmeier, Saran Tunyasuvunakool, Noah Y. Siegel, Roland Hafner, Michael Bloesch, Kristian Hartikainen, Arunkumar Byravan, Leonard Hasenclever, Yuval Tassa, Fereshteh Sadeghi, Nathan Batchelor, Federico Casarini, Stefano Saliceti, Charles Game, Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber, Nicole Hurley, Francesco Nori, Raia Hadsell, and Nicolas Heess. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022, 2024. doi: 10.1126/scirobotics.adi8022.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 805–813. PMLR, 2015.

Paul Jansonnie, Bingbing Wu, Julien Perez, and Jan Peters. Unsupervised skill discovery for robotic manipulation through automatic task generation. In *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, pp. 926–933. IEEE, 2024. doi: 10.1109/Humanoids58906.2024.10769879.

Hyunseung Kim, Byungkun Lee, Hojoon Lee, Dongyoon Hwang, Donghu Kim, and Jaegul Choo. Do's and don'ts: Learning desirable skills with instruction videos. *Advances in Neural Information Processing Systems*, 37:47741–47766, 2024.

Taehwan Kwon. Variational intrinsic control revisited. *arXiv preprint arXiv:2010.03281*, 2020.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Unsupervised reinforcement learning with contrastive intrinsic control. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=9HBbWAsZxFt.

Alexander Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2020.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

Inseok Oh, Seungeun Rho, Sangbin Moon, Seongho Son, Hyoil Lee, and Jinyun Chung. Creating pro-level ai for a real-time fighting game using deep reinforcement learning. *IEEE Transactions on Games*, 14(2):212–220, 2021.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations*, 2021.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. In *International Conference on Learning Representations*, 2024.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics*, 41(4), 2022.

Seungeun Rho, Kartik Garg, Morgan Byrd, and Sehoon Ha. Unsupervised skill discovery as exploration for learning agile locomotion. In *Conference on Robot Learning*, pp. 2678–2694. PMLR, 2025a.

Seungeun Rho, Laura Smith, Tianyu Li, Sergey Levine, Xue Bin Peng, and Sehoon Ha. Language guided skill discovery. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Seungeun Rho, Aaron Trinh, Danfei Xu, and Sehoon Ha. Reference grounded skill discovery. In *The Fourteenth International Conference on Learning Representations*, 2026.

Julia Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54(2):296–301, 1951. doi: 10.2307/1969530.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. doi: 10.1038/s41586-019-1724-z.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics (TOG)*, 40 (4):1–11, 2021.

Ruize Zhang, Sirui Xiang, Zelai Xu, Feng Gao, Shilong Ji, Wenhao Tang, Wenbo Ding, Chao Yu, and Yu Wang. Mastering multi-drone volleyball through hierarchical co-self-play reinforcement learning. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 5278–5300. PMLR, 2025.

Algorithm 1 Game-Guided Skill Discovery (GGSD)

Require: π_{high}, π_{low}, discriminator q_ϕ, opponent pool Π_K, latest-opponent probability p, skill duration k, λ_{MI}, λ_{H}

for each training iteration **do**
  Construct the opponent distribution ρ_K as in Sec. 4.1
  Initialize rollout buffers $\mathcal{D}_{\text{high}}$ and $\mathcal{D}_{\text{low}}$
  for each rollout environment or environment group **do**
    Sample opponent index $i \sim \rho_K$ and fix $\pi_{\text{foe}} \leftarrow \pi^{(i)}$
    for each high-level decision step t **do**
      Sample $z \sim \pi_{\text{high}}(\cdot \mid s_t^{\text{all}})$
      Execute z for up to k environment steps using $\pi_{\text{low}}(\cdot \mid s^{\text{proprio}}, z)$
      Compute per-step MI and low-level rewards according to Eqs. 15 and 17
      Store low-level transitions in $\mathcal{D}_{\text{low}}$
      Accumulate the discounted game reward over the executed skill interval and store the resulting high-level transition in $\mathcal{D}_{\text{high}}$
    end for
  end for
  Update q_ϕ on $\mathcal{D}_{\text{low}}$ by maximizing $\mathbb{E}[\log q_\phi(z \mid s^{\text{proprio}})]$
  Update π_{low} and V_{low} with PPO using the low-level reward in Eq. 17
  Update π_{high} and V_{high} with PPO using the high-level target in Eq. 12 and entropy regularization
  if checkpoint interval is reached **then**
    Add the current hierarchical policy to the opponent pool and increment K
  end if
end for

A State-Dependent Skill Semantics and Emergent Combos

Our experiments reveal a tension between *semantic consistency* and *compositional expressivity*. For direct human control, each discrete skill should have a predictable meaning. Pressing the same button should generally produce the same type of behavior. At the same time, we intentionally keep the skill set small to preserve a simple interface, which limits the behaviors that individual skills can represent.

Skill transitions provide a natural way to increase expressivity without adding more buttons. Because the low-level policy is state-conditioned, $\pi_{\text{low}}(a \mid s, z)$, the behavior induced by a skill z can depend on the state from which it is executed. One skill may move the agent into a particular region of the state space, from which another skill produces a qualitatively different behavior. This gives rise to *combo behaviors* that do not appear when either skill is executed in isolation.

This does not necessarily require abandoning semantic consistency. Instead, the state space can be viewed as containing a small number of behaviorally distinct regimes,

$$\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_R, \tag{18}$$

within which each skill remains relatively predictable. Skill semantics can therefore be locally consistent within a regime while changing after the agent transitions to another regime. This resembles a compact finite-state controller, where transitions between regimes allow the same small set of skill inputs to realize a broader behavioral repertoire.

`G1Boxing` provides a concrete example. The backward-leaning pose forms a distinctive regime relative to ordinary locomotion and boxing states. From regular standing states, the learned skills produce behaviors such as walking, guarding, and punching. After entering the backward-leaning state, however, a subsequent skill can exploit the stored posture and momentum to produce a substantially stronger forward punch.

This interpretation is compatible with our mutual-information objective. The discriminator encourages different skill codes to induce distinguishable behaviors, but does not require a skill to produce exactly the same motion from every state. Thus, state-dependent effects can coexist with distinguishable semantics as long as the resulting behaviors remain sufficiently predictable and separable.

There is nevertheless a limit to this mechanism. If the state space were fragmented into too many context-dependent regimes, the same button could acquire too many meanings, making the controller difficult to understand. Our results instead suggest a useful middle ground: a small number of state-dependent regimes can increase compositional expressivity while retaining a compact and interpretable interface. Understanding this trade-off between semantic consistency, compositionality, and human playability is an interesting direction for future work.

B An Alternative View of Mutual Information in Skill Discovery

Mutual information (MI) is often treated as a central objective for skill discovery. Here, we offer a complementary interpretation: rather than viewing MI as the primary mechanism that *creates* meaningful behaviors, we view it mainly as an *association mechanism* that assigns distinct behaviors to latent skill codes.

Once different skills induce behaviors that are sufficiently distinguishable for Z to be inferred from the resulting behavior, the MI objective is largely satisfied. It does not by itself require those behaviors to be semantically meaningful, useful, natural, or even substantially different. Small but easily detectable behavioral differences may be sufficient. From this perspective, MI more naturally answers *'which behavior should belong to which latent code?"* than *'which behaviors should be discovered?"* The latter requires an additional source of behavioral structure. Below, we examine five skill-discovery approaches that use MI as a central component and ask what additional source of behavioral structure each method provides. As we will see, the nature of this additional guidance largely determines the kinds of skills that ultimately emerge.

DIAYN: task-agnostic exploration. DIAYN (Eysenbach et al., 2019) combines MI maximization with maximum-entropy RL. Entropy encourages broad, task-agnostic exploration, while MI partitions the resulting behavioral variation across latent codes. In simple agents, this may yield recognizable skills such as different locomotion directions. In high-dimensional agents, however, many easily distinguishable behaviors need not correspond to coherent or useful motor skills. Under this view, DIAYN relies primarily on unguided exploration to generate behavioral variation and MI to organize it.

ASE: imitation guidance. ASE (Peng et al., 2022) combines an MI-based skill objective with adversarial imitation from motion data. The imitation objective constrains learning toward natural human motion, while MI organizes this behavioral space into a controllable latent representation. Thus, imitation largely determines learned skills, whereas MI helps make them controllable through z.

RGSD: structured latent guidance. Reference-Grounded Skill Discovery (RGSD) (Rho et al., 2026) provides behavioral structure by grounding the latent representation in reference motions. Once this representation is structured, a DIAYN-style reward can encourage the policy to visit states associated with a particular reference, effectively turning an MI-form objective into an imitation-like signal. This illustrates how the behavioral effect of MI depends strongly on the representation surrounding it.

SDAX: locomotion-task guidance. SDAX (Rho et al., 2025a) combines skill discovery with explicit task rewards for challenging locomotion behaviors such as leaping, climbing, and obstacle traversal. Task rewards direct exploration toward useful regions of behavior space, while the skill objective encourages different latent codes to capture distinct solutions. Skill discovery therefore acts as strategic exploration rather than diversity for its own sake.

GGSD: competitive-game guidance. GGSD follows the same broad perspective but obtains behavioral structure from competitive self-play. To defeat evolving opponents, the agent must discover effective ways to move, evade, strike, block, reposition, and interact with the environment. Because these behaviors emerge in service of winning, the resulting distinctions are naturally tied to strategically meaningful actions rather than arbitrary behavioral variation. Skills such as punching,

guarding, turning, or striking a puck emerge because they are useful for winning, not because they are explicitly specified.

MI then organizes these behaviors into a compact discrete skill set. The game objective determines *what kinds of behaviors are useful to discover*, while MI encourages different latent codes to represent distinguishable motor behaviors. Without the game objective, MI has no preference for semantically meaningful behaviors over arbitrary distinguishable motion; without MI, self-play need not organize effective behaviors into individually controllable skills.

This perspective suggests a common decomposition across these methods:

$$\underbrace{\text{behavioral guidance}}_{\text{what behaviors emerge}} \quad + \quad \underbrace{\text{MI-based association}}_{\text{which } z \text{ represents them}}. \tag{19}$$

Under this view, the central contribution of GGSD is not a new mechanism for maximizing mutual information, but a new source of *guidance for skill discovery*: competitive games provide a lightweight objective from which strategically useful and semantically meaningful motor behaviors can emerge, while MI organizes them into the discrete skill vocabulary required for direct human control.

C Environment Details

All environments are symmetric two-player games between "me" and "foe". Both agents receive observations with identical layouts under swapped roles, allowing a single policy to control either side. Following the notation in the main text, the high-level policy and critics receive the full game state s^{all}, while the low-level actor receives only the controlled agent's proprioceptive observation s^{proprio}. Thus, s^{proprio} forms a subset of s^{all}.

Components marked with ✓ are included in s^{proprio} and are therefore available to the low-level actor. The remaining components are available only through s^{all}. Unless noted otherwise, positions are expressed relative to the per-environment origin and velocities in the agent's body frame.

The discriminator receives the same proprioceptive information as the low-level actor, except that the previous-action dimensions are removed in `AntSumo` and `G1Boxing`.

Table 2: Observation-space dimensions.

Environment	s^{proprio}	s^{all}	# Skills
Ant Sumo	35	91	5
Ant Fencing	35	91	5
Franka Hockey	23	61	5
G1 Boxing	86	194	6

C.1 ANT SUMO

Two quadruped ants compete in an $8\,\mathrm{m} \times 8\,\mathrm{m}$ square arena; a player loses when it is pushed out of the arena. Observation and reward are presented in Table 3 and 4, respectively.

Table 3: Ant Sumo observation space. Components marked with ✓ are included in s^{proprio}; all components together form s^{all}.

Group	Component	Dim.	s^{proprio}
Me	Base height z	1	✓
	Base linear velocity (body frame)	3	✓
	Base angular velocity (body frame)	3	✓
	Base yaw	1	✓
	Projected gravity (body frame)	3	✓
	Joint positions (normalized)	8	✓
	Joint velocities ($\times 0.2$)	8	✓
	Previous action	8	✓
Me (global)	Global position (x, y)	2	
	Base roll	1	
Relative	Foe position relative to me (world frame, xyz)	3	
Foe	Global position (x, y)	2	
	Base height z	1	
	Base linear velocity (body frame)	3	
	Base angular velocity (body frame)	3	
	Base yaw and roll	2	
	Projected gravity (body frame)	3	
	Joint positions (normalized)	8	
	Joint velocities ($\times 0.2$)	8	
	Previous action	8	
Strategic	Planar distance to foe	1	
	Foe relative position (x, y) in me's body frame	2	
	Foe relative velocity (x, y) in me's body frame	2	
	Normalized distances to the four arena walls	4	
	Relative heading $(\sin \Delta\psi, \cos \Delta\psi)$	2	
	Normalized episode progress t/T	1	
	Total s^{all}	**91**	
	Total s^{proprio}	**35**	

Table 4: Reward terms for Ant Sumo.

Term	**Description**	**Expression**	**Weight**
Approach	Decrease in distance to the opponent	$d_{t-1} - d_t$	2.0
Push	Decrease in the opponent's distance to the nearest edge	$w_{t-1} - w_t$	2.0
Win	Opponent leaves the arena or falls (once, terminal)	$\mathbf{1}[\text{win}]$	$+20$
Lose	Opponent wins	$\mathbf{1}[\text{lose}]$	-20
Timeout	Draw at 10 s (once, terminal)	$\mathbf{1}[\text{draw}]$	-3
Alive, escape, upright, action, joint-velocity			0

C.2 ANT FENCING

Ant Fencing uses the same observation space as Ant Sumo (Table 3). The game uses the same arena and agent configuration, but a player additionally loses when the opponent's front foot touches its torso with sufficient contact force. The observation space is identical to that of Ant Sumo. Reward terms are presented in Table 5.

Table 5: Reward terms for Ant Fencing. The reward is identical to Ant Sumo except that the push term is disabled. Arena 8×8 m, 10 s episodes, random initial heading.

Term	**Description**	**Expression**	**Weight**
Approach	Decrease in distance to the opponent	$d_{t-1} - d_t$	2.0
Win	Front foot hits the opponent's torso ($\|F\| > 1$ N), or opponent leaves the arena / falls	$\mathbf{1}[\text{win}]$	$+20$
Lose	Opponent's front foot hits the agent's torso, or agent leaves the arena / falls	$\mathbf{1}[\text{lose}]$	-20
Timeout	Draw at 10 s	$\mathbf{1}[\text{timeout}]$	-3

C.3 FRANKA AIR HOCKEY

Two 7-DoF Franka arms face each other across an air-hockey table and attempt to shoot a puck into the opponent's goal; the first player to score two goals wins the match. All positions and velocities are expressed in the agent's own table frame, whose $+x$ axis points toward the opponent's goal. For the second player, the world x and y axes are negated so that both players observe the game from the same viewpoint. Joint velocities are scaled by 0.1. The end-effector (EE) yaw is measured relative to the agent's facing direction and encoded as $(\sin\psi, \cos\psi)$. The low-level actor observes only the controlled arm state, while the puck, opponent, and match state are included only in s^{all}.

Table 6: Franka Air Hockey observation space. Components marked with ✓ are included in s^{proprio}; all components together form s^{all}.

Group	Component	Dim.	s^{proprio}
Me	Arm joint positions	7	✓
	Arm joint velocities ($\times 0.1$)	7	✓
	EE position (table frame)	3	✓
	EE linear velocity (table frame)	3	✓
	EE yaw $(\sin\psi, \cos\psi)$	2	✓
	EE yaw rate	1	✓
Puck	Puck position (table frame)	3	
	Puck linear velocity (table frame)	3	
	Puck angular velocity (table frame)	3	
	Puck position relative to me's EE	3	
	Foe goal center relative to puck (x, y)	2	
Foe	Foe EE position (table frame)	3	
	Foe EE linear velocity (table frame)	3	
	Foe arm joint positions	7	
	Foe arm joint velocities ($\times 0.1$)	7	
Match	Me score (normalized by score-to-win)	1	
	Foe score (normalized by score-to-win)	1	
	Round progress	1	
	Normalized match progress t/T	1	
	Total s^{all}	**61**	
	Total s^{proprio}	**23**	

Table 7: Reward terms for Franka Air Hockey. A match lasts 12 s and is won by the first player to score two goals; if the puck rests for 3 s inside (0.5 s outside) the workspace, the match ends with the current score. x^p is the puck position along the axis toward the opponent's goal.

Term	**Description**	**Expression**	**Weight**
Puck progress	Forward puck displacement toward the opponent's goal (backward motion ignored)	$\max(0,\, x_t^p - x_{t-1}^p)$	10.0
Goal	Agent scores (once per goal)	$\mathbf{1}[\text{goal}]$	$+40$
Concede	Opponent scores (once per goal)	$\mathbf{1}[\text{concede}]$	-40
Match win	Two goals first, or leading at timeout / idle-puck termination	$\mathbf{1}[\text{match win}]$	$+80$
Match lose	Mirror of match win	$\mathbf{1}[\text{match lose}]$	-80
Draw	Tied at termination	$\mathbf{1}[\text{draw}]$	0
Action, workspace penalties			0

C.4 G1 BOXING

Two Unitree G1 humanoids with 23 actuated joints compete in a $10\,\mathrm{m} \times 10\,\mathrm{m}$ arena. Successful strikes to the opponent's torso or head reduce its hit points (HP), and the player with more remaining HP at timeout wins.

Table 8: G1 Boxing observation space. Components marked with ✓ are included in s^{proprio}; all components together form s^{all}.

Group	Component	Dim.	s^{proprio}
Me	Base yaw	1	✓
	Base height z	1	✓
	Base linear velocity (body frame)	3	✓
	Base angular velocity (body frame)	3	✓
	Projected gravity (body frame)	3	✓
	Glove positions (heading frame, 2×3)	6	✓
	Joint positions	23	✓
	Joint velocities ($\times 0.2$)	23	✓
	Previous action	23	✓
Me (global)	Global position (x, y)	2	
Relative	Foe position relative to me (heading frame, xyz)	3	
Foe gloves	Foe glove positions (me's torso frame, 2×3)	6	
	Foe glove velocities (me's torso frame, 2×3)	6	
Foe	Global position (x, y)	2	
	Base yaw	1	
	Base height z	1	
	Base linear velocity (body frame)	3	
	Base angular velocity (body frame)	3	
	Projected gravity (body frame)	3	
	Glove positions (heading frame, 2×3)	6	
	Joint positions	23	
	Joint velocities ($\times 0.2$)	23	
	Previous action	23	
Match	Me HP (normalized by initial HP)	1	
	Foe HP (normalized by initial HP)	1	
	Normalized time remaining $1 - t/T$	1	
	Total s^{all}	**194**	
	Total s^{proprio}	**86**	

Table 9: Reward terms for G1 Boxing, adapted from Won et al. Won et al. (2021). Arena 10×10 m, 15 s episodes, initial HP $H_0 = 10^5$ per player. Punch damage is $150\,\bar{v}_\perp^2$, where $\bar{v}_\perp$ is the glove's closing velocity normal to the struck face clipped at 12 m/s (head hits $\times 1.5$); it drains the victim's HP. The outcome is decided by HP alone; a fall or arena exit voids the round. $\hat{\mathbf{u}}$ is the unit vector toward the opponent, $\mathbf{f}_{\text{torso}}, \mathbf{f}_{\text{pelvis}}$ are the forward axes of the torso and pelvis, $\mathbf{v}_\perp$ is the root XY velocity rotated by 90° (speed clipped at 1 m/s), $\tilde{q}$ is the joint position normalised to $[-1, 1]$ within its soft limits, and F_{ij} are self-contact forces.

Term	**Description**	**Expression**	**Weight**
Punch	HP drained from the opponent minus HP lost	$(\Delta H_t^{\text{opp}} - \Delta H_t^{\text{me}})/H_0$	125
Facing	Torso and pelvis facing the opponent, in $[-2, 2]$	$\mathbf{f}_{\text{torso}} \cdot \hat{\mathbf{u}} + \mathbf{f}_{\text{pelvis}} \cdot \hat{\mathbf{u}}$	0.03
Facing velocity	Penalises moving sideways relative to the body's facing	$-\big(\lvert\mathbf{f}_{\text{torso}} \cdot \mathbf{v}_\perp\rvert + \lvert\mathbf{f}_{\text{pelvis}} \cdot \mathbf{v}_\perp\rvert\big)$	0.15
Approach	Decrease in distance to the opponent; zero when $d_t \le 1$ m	$d_{t-1} - d_t$	3.0
Joint velocity	Squared joint velocity, clipped at ± 30 rad/s	$-\sum_j \text{clip}(\dot{q}_j)^2$	3×10^{-5}
Joint limit	Excess beyond 80 % of the soft joint range, clipped at 0.3	$-\sum_j \text{clip}\big(\max(\lvert\tilde{q}_j\rvert - 0.8,\, 0)\big)^2$	0.5
Action rate	Squared change of the action	$-\lVert\mathbf{a}_t - \mathbf{a}_{t-1}\rVert^2$	0.005
Self-contact	Sum of self-collision forces, clipped at 50 N per pair	$-\sum_{(i,j)} \text{clip}(\lVert F_{ij}\rVert)$	3×10^{-4}
Win / Lose	Opponent's HP reaches zero first, or higher HP at timeout (mirror for lose)	$\mathbf{1}[\text{win}]$ / $\mathbf{1}[\text{lose}]$	$+30$ / -30
Draw	Simultaneous HP-out or equal HP at timeout	$\mathbf{1}[\text{draw}]$	-5
Arena out	Agent falls (torso below 0.65 m) or leaves the arena	$\mathbf{1}[\text{arena out}]$	-3

D HYPERAPARAMETERS

We provide the hyperparameters used in our experiments.

Table 10: Training hyper-parameters of the hierarchical self-play policies.

Hyper-parameter	**Ant Sumo**	**Ant Fencing**	**Franka Hockey**	**G1 Boxing**
PPO				
Parallel environments	4096			
Rollout length (steps / env)	32	32	24	32
Training iterations	70 000			
Learning epochs / mini-batches	5 / 4			
Learning rate	1×10^{-4}			
Discount γ	0.99	0.99	0.995	0.99
GAE λ	0.95			
Clip ratio ϵ	0.2			
Value loss coefficient (clipped)	1.0			
Max gradient norm	1.0			
Entropy coefficient, low-level	5×10^{-4}	5×10^{-4}	1×10^{-3}	5×10^{-4}
Entropy coefficient, high-level	0.02	0.02	0.02	0.05
Initial action std, low-level	1.0	1.0	1.0	0.2
Action clipping	10	10	10	1
Control frequency (Hz)	60	60	60	30
Seed	42			
Hierarchy and skill discovery				
Number of skills K	5	5	5	6
Skill duration	10			
MI reward scale β	0.02	0.02	0.005	0.05
Discriminator learning rate	1×10^{-4}			
Self-play opponent pool				
Warm-up iterations (mirror self-play)	0	0	2000	2000
Opponent snapshot interval (iterations)	2000	2000	2000	1000
Opponent resampling interval (iterations)	200			
Opponent groups	10	10	4	10
Fraction of groups using the latest policy	0.6	0.6	0.6	0.3

Network architecture. All networks are multilayer perceptrons with ELU activations. For the Ant Sumo, Ant Fencing, and G1 Boxing tasks, the high-level policy has three hidden layers of [512, 512, 256] units and outputs a categorical distribution over the K skills; the low-level policy and both critics (one for each level) have four hidden layers of [512, 512, 256, 128] units, and the skill discriminator has two hidden layers of 256 units. For Franka Hockey, the high-level policy, the low-level policy, the critics, and the discriminator all use four hidden layers of [512, 512, 512, 256] units. The low-level policy receives only the agent's own proprioceptive state together with the one-hot skill and outputs the mean of a Gaussian over joint actions with a learned state-independent standard deviation; for G1 Boxing the mean is additionally bounded with a $\tanh$.

E VLM-BASED SKILL DIVERSITY METRIC

Pipeline. For every trained policy we record one clip per skill (Ant: 5, Franka: 5, G1: 6 skills; for METRA a skill is the one-hot unit vector along each latent axis). Each clip is a single-environment rollout that starts from a reset state and runs for 5 s, rendered at 1920×1080. The clip is captioned by a vision-language model (VLM), the caption is embedded with a text-embedding model, and the diversity score of a policy is the mean pairwise cosine distance

$$D = \frac{1}{\binom{K}{2}} \sum_{i<j} \big(1 - \cos(\mathbf{e}_i, \mathbf{e}_j)\big)$$

over the K skill embeddings $\mathbf{e}_i$ (L2-normalized). The prompt asks the VLM to answer *no semantic meaning* when a clip shows no interpretable behavior; every pair containing such a clip is assigned

distance 0, so uninterpretable skills do not add diversity. Scores are averaged over 3 training seeds (mean $\pm$ standard deviation).

Models and settings. Captioning uses `gemini-3.6-flash` (temperature 0.1, at most 1024 output tokens). The video is sampled by the API at 5 fps over a fixed segment of the clip: Ant 1–4 s, Franka 2–4 s, G1 0–5 s. The segment skips the initial settling phase after the reset where the robot has not yet started to move. Captions are embedded with `gemini-embedding-001` (task type `SEMANTIC_SIMILARITY`, 3072 dimensions). The marker *no semantic meaning* is matched case-insensitively as a substring, since the model occasionally wraps it in quotes or a full sentence.

Prompts. One prompt is used per robot; the same prompt is applied to every method on that robot.

- **Ant.** *"This is a rendered clip of a simulated robot. Describe the semantic meaning of the robot's behavior in one sentence. If you cannot think of a meaningful downstream task for which this behavior could be useful, say 'no semantic meaning'."*
- **Franka.** *"This is a rendered clip of a simulated robot arm. Describe which region of the table surface the robot arm moves over. If the robot arm moves too far away from the table, or would otherwise be unlikely to meaningfully interact with objects on the table, say 'no semantic meaning'."*
- **G1.** *"This is a rendered clip of a humanoid robot with a sphere attached to each hand. Describe the semantic meaning of the robot's behavior in one sentence. If the behavior has no clear semantic meaning, respond with exactly 'no semantic meaning'."*

Example captions. Ant (ours): *"The quadruped robot performs a self-righting maneuver to flip itself over and stand back up on its feet."*, *"The robot is rotating in place."*, *"The robot is crawling forward."* Franka (ours): *"The robot arm remains stationary over the bottom-right region of the table surface."* G1 (ours): *"The humanoid robot is walking backwards."*