Supplementary Material for "Hidden organization of trajectories: from interacting particles to reinforcement-learning agents to humans"

Hongkun Li[1], Sergei Gepshtein[2,3], Yunyun Li[1,*], Fabio Marchesoni[1], Sergey Savel'ev[4]
[1]MOE Key Laboratory of Advanced Micro-Structured Materials, School of Physics Science and Engineering, Tongji University, Shanghai 200092, China.
[2]Department of Cognitive Science, University of California San Diego, La Jolla, CA 92093, USA.
[3]Salk Institute for Biological Studies, La Jolla, CA 92037, USA.
[4]Department of Physics, Loughborough University, LE11 3TU, UK.
[*]Corresponding author: yunyunli@tongji.edu.cn

Supplementary Note 1 Definitions of the three indices

Coverage (C_v) quantifies the fraction of the designated region that has been visited by at least one agent. The region is discretized into a uniform grid [the cell size is chosen as 0.1 times the system length (0.1L)], and coverage is computed as $C_v = \frac{N_{\rm occ}}{N_{\rm tot}}$, where $N_{\rm occ}$ and $N_{\rm tot}$ represent the number of occupied and total grid cells, respectively, averaged across all episodes.

Overlap (O_v) quantifies the spatial redundancy between the historical trajectories of the two agents, independent of time. It measures how much the regions explored by the two agents overlap throughout the evaluation period. Let P_1 and P_2 denote the sets of trajectory points for agent 1 and 2, containing N_1 and N_2 points, respectively. The equation is shown below:

$$O_v = \frac{1}{2}\left(\frac{1}{N_1}\sum_{p_i\in P_1}\mathbb{I}\big[d(p_i,P_2)<r_{th}\big]+\frac{1}{N_2}\sum_{q_j\in P_2}\mathbb{I}\big[d(q_j,P_1)<r_{th}\big]\right). \tag{S1}$$

where $\mathbb{I}[\cdot]$ is the indicator function, equal to 1 if the condition is true and 0 otherwise, and $d(p,P)=\min_{q\in P}\|p-q\|$ is the distance from point p to the nearest point in set P, computed over the entire trajectory and averaged across all test episodes. Using a K-nearest-neighbor [1]–based approach, the overlap is computed as the fraction of points from one agent lying within a small threshold distance r_{th} (0.1L) of the other's trajectory.

Spreading (S_p) describes the mean spatial separation between the two agents over time, capturing the degree of coordination. It is computed as the time-averaged

pairwise distance:

$$S_p = \frac{1}{T}\sum_{t=0}^{T}\sqrt{(x_1^t - x_2^t)^2 + (y_1^t - y_2^t)^2}. \tag{S2}$$

where (x_1^t, y_1^t) and (x_2^t, y_2^t) denote the agent positions at time t, averaged across all episodes.

Supplementary Note 2 Definitions of the clustering evaluation indices

The clustering evaluation indices used in our study are summarized as follows:

Silhouette coefficient (S): It evaluates both intra-cluster compactness and inter-cluster separation, defined as

$$S = \frac{b - a}{\max(a, b)}. \tag{S3}$$

where a is the average distance between a sample and all other points in the same cluster, and b is the minimum average distance between that sample and points in any other cluster. $S \in [-1, 1]$, with higher values indicating well-separated, cohesive clusters.

Davies–Bouldin (DB) index quantifies the average similarity between clusters, defined as

$$DB = \frac{1}{K}\sum_{i=1}^{K}\max_{j\neq i}\frac{\sigma_i + \sigma_j}{d_{ij}}. \tag{S4}$$

where $\sigma_i = \frac{1}{n_i}\sum_{x\in C_i}\|x - c_i\|$ is the mean intra-cluster distance for cluster i, $d_{ij} = \|c_i - c_j\|$ is the Euclidean distance between cluster centers, K is the total number of clusters, and n_i is the number of samples in cluster C_i. A lower DB value indicates more compact and better-separated clusters, with $DB < 1$ typically reflecting good clustering quality.

Adjusted Rand Index (ARI): Measures the agreement between two cluster label assignments (e.g., across bootstrap resampling trials), adjusted for chance:

$$ARI = \frac{\mathrm{RI} - \mathbb{E}[\mathrm{RI}]}{\max(\mathrm{RI}) - \mathbb{E}[\mathrm{RI}]}. \tag{S5}$$

where RI (Rand Index) is the fraction of sample pairs that are either assigned to the same cluster in both clustering results or to different clusters in both clustering results, and $\mathbb{E}[\mathrm{RI}]$ is its expected value under random labeling.

Bootstrap resampling repeatedly selects random subsets (typically 80% of the dataset) to assess clustering stability. ARI ranges from 0 to 1, with values near 1 indicating highly consistent and reproducible clustering.

Normalized Mutual Information (NMI): It evaluates the mutual dependence between two clustering results:

$$NMI = \frac{2I(X;Y)}{H(X) + H(Y)}. \tag{S6}$$

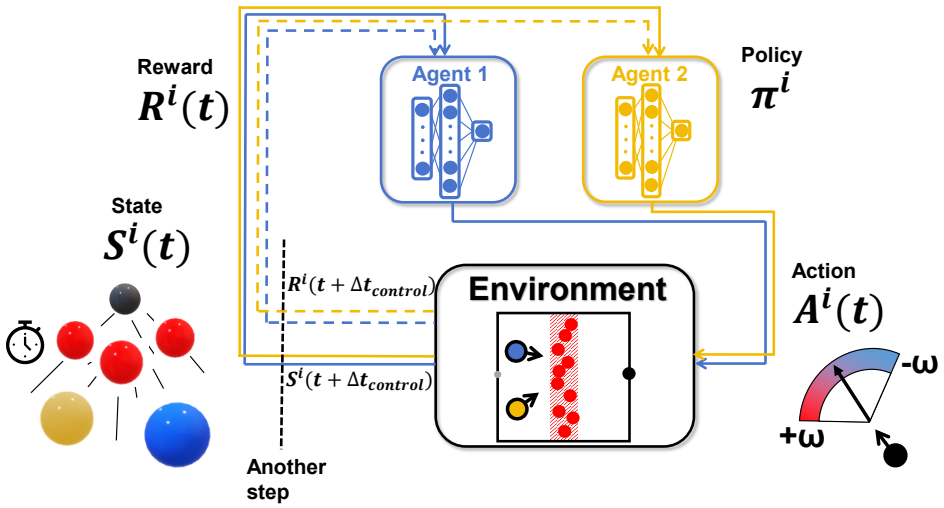

Fig. S1 Reinforcement learning framework. Each agent is controlled by an individual policy network (policy π^i, $i = 1, 2$) that maps environmental observations [state $S^i(t)$] to angular velocity [action $A^i(t)$].

where $I(X;Y)$ is the mutual information and $H(X), H(Y)$ are the entropies of the two clusterings. $NMI \in [0, 1]$, and higher values indicate stronger agreement and reproducibility.

Overall, higher Silhouette, ARI, and NMI scores, together with a lower DB index, correspond to more robust and well-defined clustering structures.

Supplementary Note 3 Learning framework for self-propelled agents

Agent control policies are trained using the Proximal Policy Optimization (PPO) algorithm. Our implementation is adapted from the work of [2]. Each agent is governed by a neural network that maps states to actions, aiming to maximize cumulative expected rewards derived from target capture and successful exit conditions [see Fig. S1]. The state space provides sufficient information for both agents to develop strategies and includes: $(x_{\text{exit}} - x,\ y_{\text{exit}} - y,\ x_{\text{target1}} - x,\ y_{\text{target1}} - y,\ x_{\text{target2}} - x,\ y_{\text{target2}} - y,\ x_{\text{target3}} - x,\ y_{\text{target3}} - y,\ x_{\text{other}} - x,\ y_{\text{other}} - y,\ x,\ y,\ \theta,\ t)$, where $x_{\text{exit}} - x$ and $y_{\text{exit}} - y$ denote relative coordinates of the exit, $x_{\text{target1,2,3}} - x$ and $y_{\text{target1,2,3}} - y$ denote relative coordinates of the three nearest targets, x, y is agent position, θ is its orientation, $(x_{\text{other}}, y_{\text{other}})$ is the position of the other agent, and t denotes the normalized remaining time, defined as $t = \big[\text{episode length (total control step number)} - \text{elapsed control step number}\big]/\text{episode length}$.

At each control step, agents receive an immediate reward, $R^i(t + \Delta t_{\text{control}})$, which quantifies their task performance—such as rewards for collecting targets or penalties for time consumption. By iteratively refining policies via PPO, the system effectively closes the perception–action–learning loop. Through extensive training over numerous

simulated episodes, agents progressively evolve cooperative or competitive behaviors, dictated by the underlying reward structure.

The total reward for PPO consists of three components::

$$R_{\text{target}} = \begin{cases} A/N_{target}, & \text{if a target is captured.} \\ 0, & \text{otherwise.} \end{cases} \tag{S7}$$

$$R_{\text{exit}} = \begin{cases} -B, & \text{if no agent exits by episode end.} \\ 0, & \text{otherwise.} \end{cases} \tag{S8}$$

$$R_{\text{time}} = -\frac{\exp\left(k \cdot \frac{\text{elapsed control steps}}{\text{total control steps}}\right) - 1}{[\exp(k) - 1]/k - 1} \cdot \frac{1}{\text{total time-steps}}. \tag{S9}$$

Here, R_{target} is normalized such that capturing all ten targets yields reward A. The term R_{time} implements a time-dependent penalty whose instantaneous magnitude grows exponentially (for $k > 0$) as the episode advances. The sum of R_{time} in one episode is approximately -1 (derivation details in the next section). For different episode lengths, we set $k = 50\tau/[(\text{episode length}) \cdot \Delta t_{\text{control}}]$ where $\tau = L/v_0 = 1$ to avoid sparse rewards in the initial phase for cases with high episode lengths. Overall, the total episode reward lies approximately within $[-B - 1, A)$. We prioritize exit reaching over target collection by setting the reward weights to (A = 1) and (B = 9). The large penalty in R_{exit} enforces the requirement that at least one agent must escape.

The reward allocation is as follows:

Cooperative: Agents share R_{target} (≥ 0) and R_{time} (< 0). If neither exits, both incur R_{exit} (≤ 0).

Competitive: R_{target} is not shared. The episode terminates when one agent exits, and the other incurs R_{exit}. Thus, agents compete for both targets and the exit.

Mixed: R_{target} is not shared, but no penalty is assigned if one agent exits. Agents compete for targets but cooperate for exit completion.

In our PPO training, each run is initialized with an independent random seed uniformly sampled from $[0, 1 \times 10^6)$. Training proceeds for 50000 episodes, during which the cumulative reward is continuously recorded to track learning progress, using a moving average over a 1000-episode window. In the competitive setting where two agents compete for the exit, we employ sequential training [3] to ensure both agents gain sufficient experience. Specifically, every 100 episodes, we alternate training between the two agents, updating one while keeping the other's neural network weights fixed. Only runs in which the smoothed reward curve converges to a stable plateau are retained for analysis, as they indicate that learning has approached a steady behavioral regime. After training, we evaluate the trained neural networks by conducting a certain number of test episodes.

Supplementary Note 4 Derivation details of R_{time}

Denote the total number of control steps in a full episode by N, index discrete control steps by $m = 1, \ldots, N$ and total number of time-steps by $N_{timestep}$. It is convenient to view the penalty in terms of a normalized continuous weight function on $x \in [0, 1]$:

$$w(x) \;=\; \frac{e^{kx} - 1}{\displaystyle\int_0^1 \left(e^{ks} - 1\right) ds} \;=\; \frac{e^{kx} - 1}{\dfrac{e^k - 1}{k} - 1}, \qquad x \in [0, 1]. \tag{S10}$$

With this notation the discrete per-time-step penalty can be written as

$$R_{\text{time}}(m) \;=\; -\, w\Big(\frac{m}{N}\Big) \cdot \frac{1}{N_{time-step}}. \tag{S11}$$

By construction the denominator normalizes w so that its integral over $[0, 1]$ equals unity. Consequently

$$\sum_{m=1}^{N} R_{\text{time}}(m) \approx -\int_0^1 w(x)\, dx = -1. \tag{S12}$$

i.e. when an episode runs to completion the cumulative time penalty is approximately -1. The explicit factor $1/N_{time-step}$ distributes this unit mass across discrete timesteps and therefore makes the total time penalty comparable across different episode lengths.

Supplementary Note 5 Details of training PPO

Proximal Policy Optimization (PPO)

Our experiments employ Proximal Policy Optimization (PPO) within an Actor–Critic paradigm [4]. Concretely, the architecture separates decision-making and value estimation:

- The *Actor* (policy network π_θ) produces actions conditioned on the current observation.
- The *Critic* (value network V_ϕ) provides an estimate of expected return from a given state and serves as a baseline to reduce variance in the policy updates.

The Critic thus supplies low-variance advantage signals used by the Actor to drive policy improvement, enabling more efficient and stable learning.

PPO builds on the canonical policy-gradient objective:

$$L^{\text{PG}}(\theta) = \mathbb{E}_t \left[r_t(\theta)\, \hat{A}_t \right]. \tag{S13}$$

where $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the importance sampling ratio between the updated policy and the behavior policy, and $\hat{A}_t$ denotes an estimator of the advantage (we compute this via GAE [5] using the Critic). Although (S13) is an unbiased objective, unconstrained changes in $r_t(\theta)$ may cause excessively large policy steps and destabilize training.

To limit such abrupt updates, PPO uses a clipped surrogate objective that effectively enforces a trust region around the current policy:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min\Big(r_t(\theta)\hat{A}_t,\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t \Big) \right]. \tag{S14}$$

The scalar $\epsilon > 0$ (we use values in the range 0.1–0.3) sets the clipping bounds. For positive advantages the clipping prevents $r_t(\theta)$ from exceeding $1+\epsilon$, and for negative advantages it prevents $r_t(\theta)$ from falling below $1-\epsilon$. The minimum operator picks the more conservative term, mitigating overly optimistic parameter updates.

This clipped surrogate provides a simple, first-order alternative to second-order trust-region approaches [e.g., TRPO [6]], allowing repeated mini-batch passes without destabilizing policy learning. In parallel, the Critic is trained by regression to bootstrapped value targets, improving the advantage estimates used by the Actor.

In this study, PPO serves as the principal learning algorithm for agents operating in multi-agent pursuit tasks across a variety of spatial geometries; its robustness to noisy gradients and balance between exploration and exploitation make it well-suited to these coordination-sensitive environments.

Network architectures and action parameterization

Both Actor and Critic are implemented as fully connected networks with hidden-layer width fixed to 64 units. For continuous control we adopt a Gaussian policy: the Actor outputs the mean of a diagonal Gaussian and the policy standard deviation is modeled as a trainable log-standard-deviation vector.

Actor (Gaussian policy). The policy network maps state vectors $s \in \mathbb{R}^{\text{state\_dim}}$ through two tanh-activated hidden layers (each of size 64). The network head produces a mean vector $\mu(s)$ and we enforce action bounds by applying a scaled hyperbolic tangent:

$$\mu = \text{max\_action} \cdot \tanh\big(\text{Linear}(\cdot)\big).$$

A learnable parameter $\log\sigma$ (broadcast to match μ's shape) defines $\sigma = \exp(\log\sigma) > 0$, yielding the diagonal Gaussian $a \sim \mathcal{N}(\mu, \sigma^2 I)$.

Critic. The value estimator $V_\phi(s)$ mirrors the Actor's hidden structure: two fully connected layers (64 units each, with tanh activations), followed by a linear output producing a scalar value estimate.

Training procedure

Data collection proceeds by interacting with the environment until a buffer of $B = 2048$ transitions $\{(s_t, a_t, r_t, s_{t+1}, d_t)\}$ is accumulated, where d_t flags terminal steps.

Advantages $\hat{A}_t$ are computed with Generalized Advantage Estimation (GAE) [5], which trades off bias and variance via an exponential weighting controlled by λ:

$$\delta_t = r_t + \gamma(1 - d_t)V_\phi(s_{t+1}) - V_\phi(s_t). \tag{S15}$$

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l (1 - d_{t+l})\delta_{t+l}. \tag{S16}$$

and efficiently implemented by the backward-recursion:

$$\hat{A}_t = \delta_t + \gamma\lambda(1 - d_t)\hat{A}_{t+1}, \quad \hat{A}_T = 0. \tag{S17}$$

We set the discount factor $\gamma = 0.99$ and GAE parameter $\lambda = 0.95$.

The full PPO loss used to update the policy also includes an entropy bonus:

$$\begin{aligned} L^{\mathrm{PPO}}(\theta) = \mathbb{E}_t\Big[& \min\big(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big) \\ & + \beta_{\mathrm{ent}}\, \mathcal{H}[\pi_\theta](s_t)\Big], \end{aligned} \tag{S18}$$

where $\mathcal{H}[\pi_\theta](s_t)$ denotes the policy entropy and β_{ent} weights entropy regularization to encourage exploration.

The Critic minimizes the mean-squared error to a bootstrapped target:

$$\mathcal{L}^{\mathrm{VF}}(\phi) = \mathbb{E}_t\left[\left(V_\phi(s_t) - V_t^{\mathrm{target}}\right)^2\right], \qquad V_t^{\mathrm{target}} = \hat{A}_t + V_\phi(s_t). \tag{S19}$$

Rollouts are split into mini-batches of size 64, and network parameters for both Actor and Critic are updated for $K = 10$ epochs per iteration to make efficient reuse of collected samples while preserving stability.

Implementation specifics

To stabilize optimization and improve convergence we apply several practical measures:

Orthogonal weight initialization to promote healthy signal flow. **Advantage normalization**: per-batch standardization $\hat{A}_t \leftarrow (\hat{A}_t - \mu_{\hat{A}})/(\sigma_{\hat{A}} + 10^{-8})$ to reduce variance. **Entropy regularization** with an optional exponential decay schedule to gradually shift from exploration toward exploitation. **Gradient clipping** by global norm (max norm = 0.5) to avoid exploding updates. **Learning-rate schedule**: a warm-up followed by cosine annealing.

The hyperparameters used consistently across experiments are as follows: both actor and critic learning rates are 3×10^{-4}; total training comprises 50,000 episodes; discount factor $\gamma = 0.99$; GAE parameter $\lambda = 0.95$; clipping coefficient $\epsilon = 0.2$; Adam optimizer is used for all parameter updates; and all nonlinearities employ the Tanh activation.

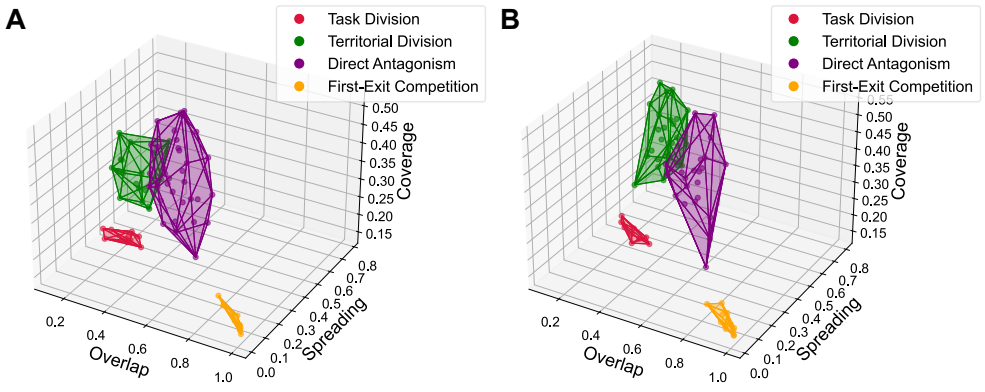

Fig. S2 Clustering results for two additional target distribution cases. (**A**) Dual uniform bands and (**B**) Diagonal band. The hulls exhibit no overlap (IoU=0). Consistent separation of behavioral clusters confirms the robustness of the classification method.

The learning-rate policy implements a linear warm-up for the initial 3,000 episodes, after which the rate decays according to a cosine schedule down to a minimum of 1×10^{-6}.

Supplementary Note 6 Three-dimensional cluster convex hulls of Dual bands and Diagonal band

See Fig. S2.

Detailed construction of trajectory overlap maps

To construct these trajectory overlapping maps, the accessible region is discretized into a 100×100 gird for Fig.2, a 200×200 grid for Fig.4 and a 30×15 grid for Fig.6 in main text. Each grid cell is indexed by an integer pair (k, l), representing its position along the x- and y-axes, respectively. For all time-steps (or control steps) of all episodes, the total number of trajectory points of particle or agent or visitor i in cell (k, l) is recorded as $N_{\mathrm{tr}}^{i}(k, l)$, where $i = 1, 2$ corresponds to the particles, agents or visitors. For each non-empty cell, we compute the normalized contribution of each particle, agent or visitor as

$$w_i(k, l) = \frac{N_{\mathrm{tr}}^{i}(k, l)}{\sum_i N_{\mathrm{tr}}^{i}(k, l)}. \tag{S20}$$

and use these weights to linearly combine the RGB colors associated with the two particles, agents or visitors:

$$\mathbf{C}(k, l) = \sum_i w_i(k, l)\, \mathbf{c}_i. \tag{S21}$$

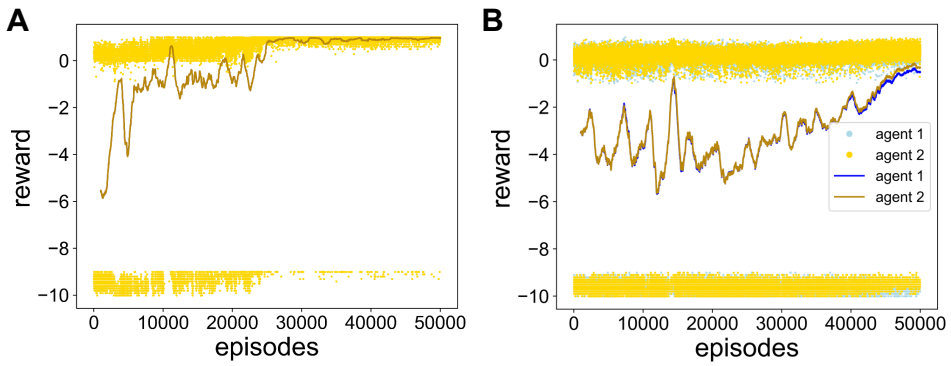

Fig. S3 Representative examples of episode reward trajectories. (**A**) Successful case under the cooperative setting. (**B**) Failed case under the mixed setting. In the cooperative scenario, both agents share a single reward signal, so only one curve is shown. In contrast, in the mixed setting, each agent receives an individual reward, and the two curves correspond to the respective agents.

where $\mathbf{c}_i$ are the RGB vectors of blue $(0, 0, 1)$ and yellow $(1, 1, 0)$. The resulting RGB color code quantifies the local overlap of the trajectories. Grid cells with no trajectory points are rendered in white.

Examples of successful and failed training cases

Figure S3 presents representative examples of both successful and unsuccessful training outcomes. Panel A corresponds to successful training under the cooperative settings, respectively, where the smoothed episode reward curves exhibit distinct plateaus, indicating convergence to stable behavioral strategies. In panel B, representing a mixed setting, the reward trajectory shows no clear plateau, suggesting that the agents failed to achieve stable coordination, and the training is thus regarded as unsuccessful.

Criterion of choosing cluster number

To determine the optimal number of clusters in agglomerative clustering, we rely on two complementary quantitative criteria. First, we evaluate the Silhouette coefficient across different cluster counts to assess the quality of cluster separation. Second, we plot the Sum of Squared Errors (SSE)—which measures the total squared distance between data points and their assigned cluster centers, reflecting cluster compactness—and apply the Kneedle algorithm [7] to identify the elbow point, where further increasing the number of clusters results in only marginal decreases in SSE. By jointly considering these metrics, we select a cluster number that balances separation and compactness while remaining interpretable in practice. As shown in Fig. S4, for the Hamiltonian system with Morse potential, both the Silhouette coefficient and the elbow point indicate an optimal cluster number of 3, which we adopt. For the Yukawa potential, although the elbow point occurs at 3, the silhouette score peaks at 4; we therefore select 4 clusters. In contrast, for human trajectories, the silhouette score decreases with increasing cluster number, while the elbow point is identified at 3;

balancing interpretability with these quantitative metrics, we select 3 as the optimal number of clusters.

For self-propelled agents, we adopt a similar criterion, selecting 4 as the cluster number for Uniform, Gaussian, and Diagonal bands. However, to further discriminate between distinct clusters, we introduce a task-specific metric termed *Capture Asymmetry*, designed to capture behaviorally differences. This metric is defined as

$$A = \frac{|N_a - N_b|}{N_a + N_b + \epsilon}. \tag{S22}$$

where N_a and N_b denote the numbers of targets captured by the two agents, and ϵ is a small constant to avoid division by zero. This normalized measure ranges from 0 (perfectly balanced capture) to 1 (maximally asymmetric behavior).

For Dual bands, both the silhouette score and the elbow method suggest selecting 3 clusters, which would effectively merge the *Task Division* and *Territorial Division* policies (shown in Fig. S6D; note that a 4-cluster solution separates these two groups). However, these policies exhibit markedly different Capture Asymmetry. Specifically, for Dual bands, the *Task Division* policy often results in one agent reaching the exit without capturing any targets, leading to a high asymmetry value (shown in Fig. S6B). To preserve this behaviorally meaningful distinction, we therefore select 4 as the optimal number of clusters for the dual-band case. In summary, this demonstrates that integrating task-specific behavioral insights alongside standard metrics is helpful. Such additional information ensures the selection of a cluster number that accurately reflects the underlying motion patterns although this remains contingent upon the specific system.

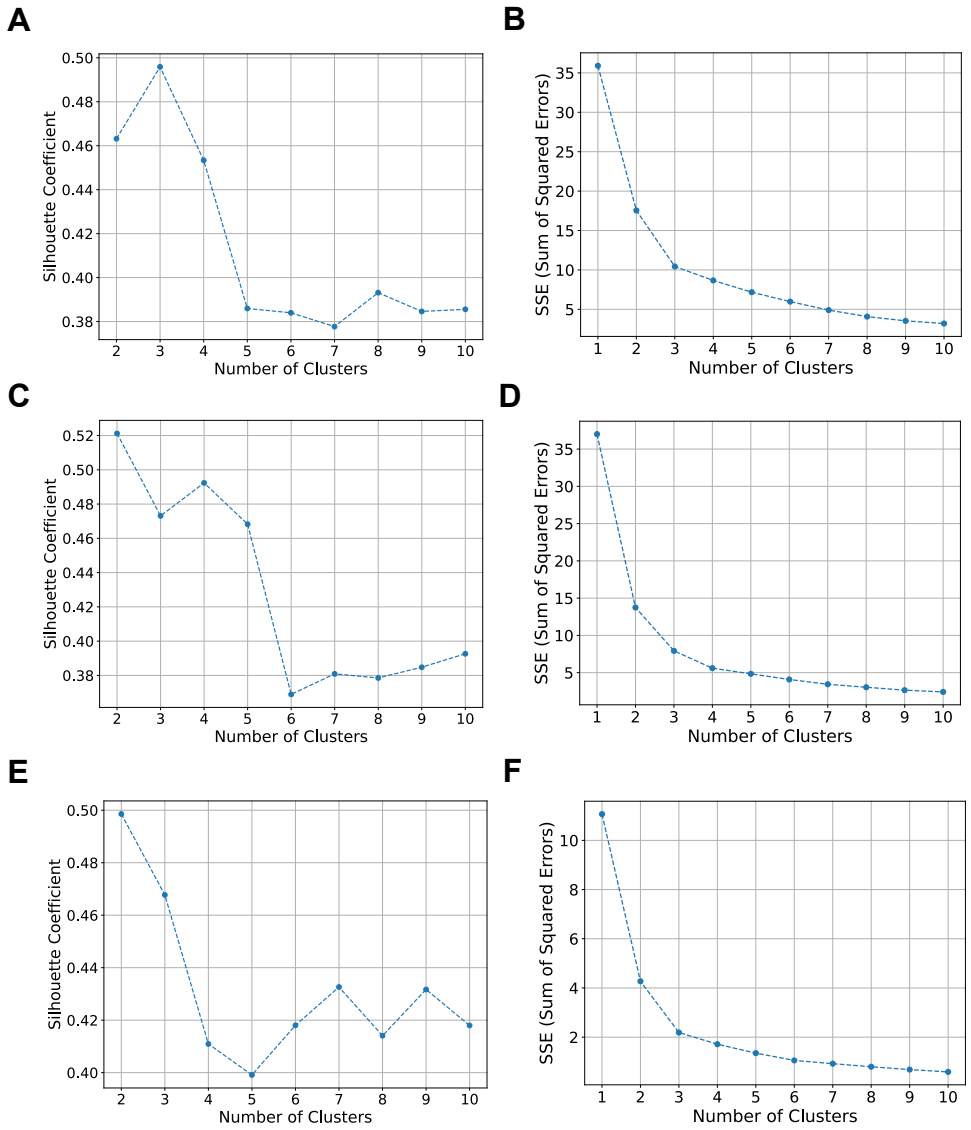

Fig. S4 Silhouette coefficients and SSE across varying numbers of clusters. Panels (**A**) and (**B**) display results for Hamiltonian particles interacting via the Morse potential, while (**C**) and (**D**) correspond to the Yukawa potential. Panels (**E**) and (**F**) depict results derived from human trajectories.

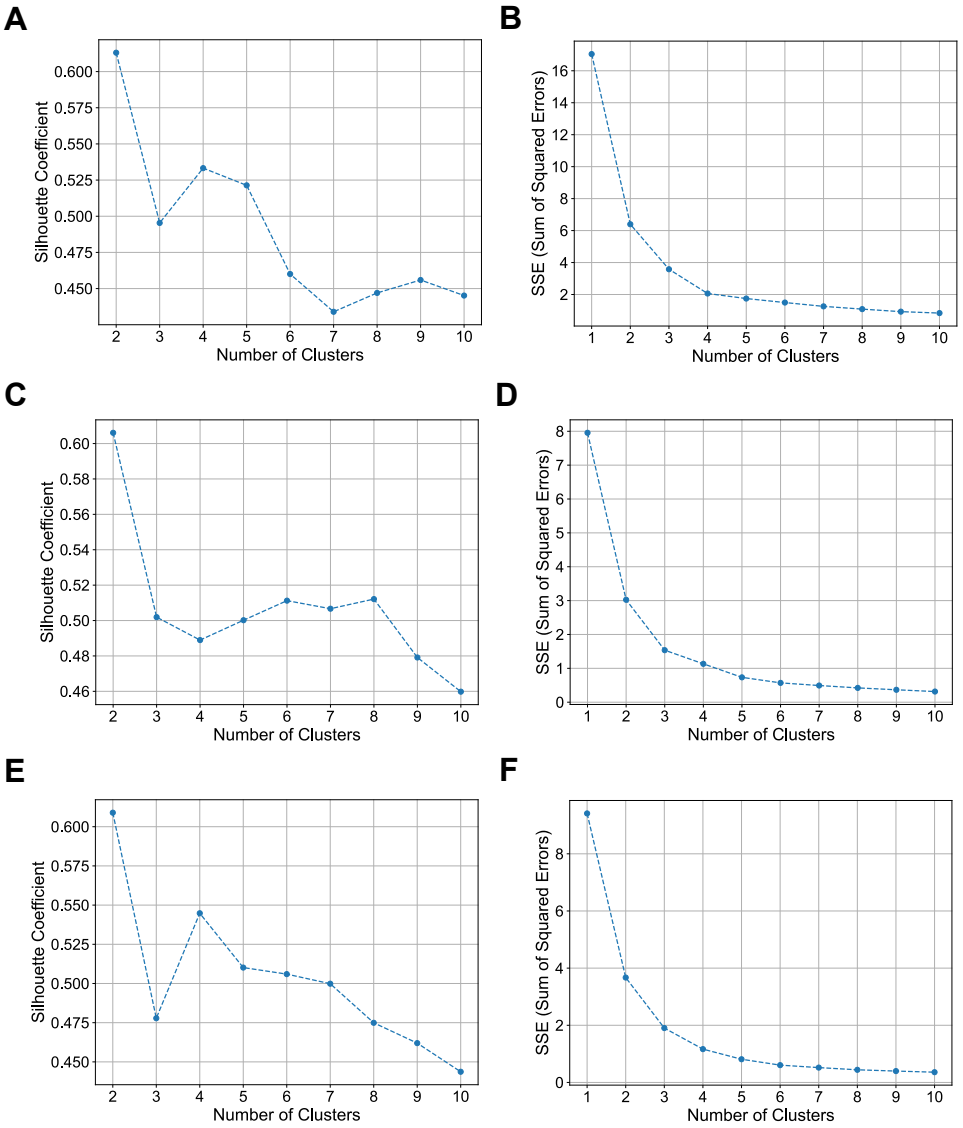

Fig. S5 Silhouette coefficients and SSE across varying numbers of clusters for self-propelled agents. Panels (**A**) and (**B**) are for Uniform and Gaussian bands; panels (**C**) and (**D**) correspond to Dual bands; and panels (**E**) and (**F**) correspond to Diagonal band.

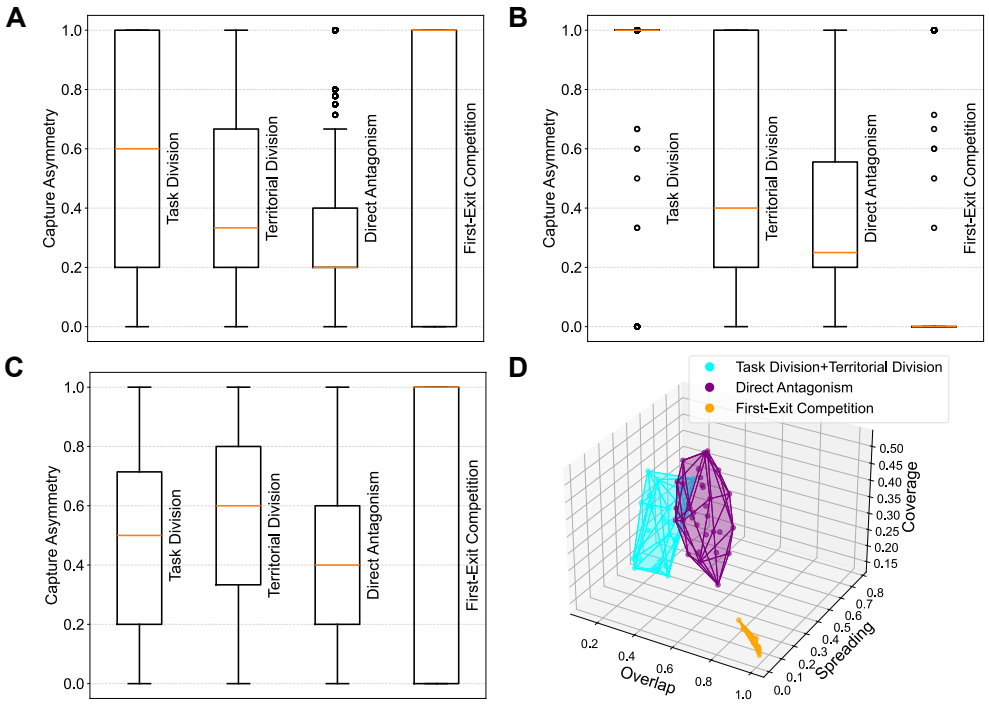

Fig. S6 Box plots of Capture Asymmetry for different target distributions. (**A**) Uniform and Gaussian bands, (**B**) Dual bands, and (**C**) Diagonal band. (**D**) Clustering results for Dual bands with three clusters. The orange line indicates the median.

References

[1] Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)

[2] Lizhi: DRL-code-pytorch: PPO continuous. MIT License (2022). https://github.com/Lizhi-sjtu/DRL-code-pytorch/tree/main/5.PPO-continuous

[3] Dai, H., Mazza, M.G., Li, Y., Marchesoni, F., Savel'ev, S.: Training strategies for competing multiagent dynamical systems. Phys. Rev. E **112**, 065310 (2025)

[4] Konda, V., Tsitsiklis, J.: Actor-critic algorithms. In: Solla, S., Leen, T., Müller, K. (eds.) Advances in Neural Information Processing Systems, vol. 12, pp. 1008–1014. MIT Press, ??? (1999)

[5] Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438 (2018)

[6] Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: Proc. Int. Conf. Mach. Learn. (ICML), pp. 1889–1897 (2015)

[7] Satopaa, V., Albrecht, J., Irwin, D., Raghavan, B.: Finding a kneedle in a haystack: detecting knee points in system behavior. In: Proc. IEEE Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW), pp. 166–171 (2011)