

A Memory Update Module

The agent memory update module primarily consists of an agent detection mechanism and a memory encoding structure. In our main approach, this involves an MLP-based agent detector and a graph-based memory module. This section elaborates on the graph structure and the pre-training of the detector.

A.1 Graph-based Memory Representation

The graph-based memory module offers a structured approach to model inter-agent states and their relationships. For each observing agent, a local graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ is dynamically constructed or updated at each timestep t .

- **Nodes (\mathcal{V}_t):** Each node $v_i \in \mathcal{V}_t$ corresponds to an inferred latent state representation of a specific agent i (either the self-agent or a detected peer). These node features, denoted $\mathbf{z}_i \in \mathbb{R}^{D_{node}}$, are generated by encoding the raw observation associated with agent i .
- **Edges (\mathcal{E}_t):** Edges $e_{ij} \in \mathcal{E}_t$ link pairs of nodes (v_i, v_j) , representing the inferred relationship between agent i and agent j . In our work, these are spatial relationships, with edge features $\mathbf{f}_{ij} \in \mathbb{R}^{D_{edge}}$ derived from their relative positions.

This dynamic graph serves as input to a Graph Neural Network (GNN), which processes the relational information to produce a contextualized memory representation for the observing agent, thereby informing its subsequent actions.

A.2 Pre-training Process

Key components of the agent memory module are pre-trained to establish meaningful initial representations, facilitating more effective subsequent MARL training.

MLP Agent Detector. This module processes an agent’s observation to output probabilities $\mathbf{p} = [p_1, \dots, p_N]^\top$, where p_k denotes the likelihood of observing an agent of type- k . These probabilities are later used to gate memory updates during MARL training via a threshold τ_{det} . Pre-training is supervised using ground-truth presence labels $y_k \in \{0, 1\}$, which are set to 1 if agent- k appears within the field of view or one body length of the observer, and 0 otherwise. To facilitate reliable detection, agents in our environment are either designed with or inherently exhibit distinct visual appearances, and each agent type is represented by a dedicated channel in the observation tensor, enabling effective training of an MLP-based detector.

$$\mathcal{L}_{det} = - \sum_{k=1}^N [y_k \log(p_k) + (1 - y_k) \log(1 - p_k)]. \quad (14)$$

Graph Memory Encoders. The node and edge encoders within the graph memory are pre-trained as follows: (i) *Node Encoder*: for an observing agent k , its encoder produces a feature $\mathbf{z}_{k \rightarrow j}$ representing its understanding of another agent j ’s state. This is trained to predict agent j ’s true latent state \mathbf{s}_j . The MSE loss is weighted by a detection mask, which is derived from the predicted transition probabilities $p_{k \rightarrow j}$ produced by a pre-trained detector, using a fixed threshold to determine valid entries; (ii) *Edge Encoder*: this encoder generates 2D edge features \mathbf{f}_{ij} from inter-agent cues, trained to directly predict the true relative position vector $(\Delta x_{ij}, \Delta y_{ij})$ between agents i and j using an MSE loss.

$$\begin{cases} \mathcal{L}_{node}^{(k)} = \sum_{j \neq k} p_{k \rightarrow j} \cdot \mathbb{E} [||\mathbf{z}_{k \rightarrow j} - \mathbf{s}_j||^2] . \\ \mathcal{L}_{edge} = \mathbb{E} [||\mathbf{f}_{ij} - (\Delta x_{ij}, \Delta y_{ij})||^2] . \end{cases} \quad (15)$$

These pre-training steps provide the memory module with an initial capacity for relevant information extraction, potentially accelerating overall learning.

B Derivation with Cantelli Inequality

In this section, we detail the transformation of the distributionally robust chance constraint, Eq.(6) of the main paper. This transformation leverages Cantelli’s inequality to arrive at a tractable loss

under second-order cone program (SOCP) constraint. In this section, we set $\bar{c} = 1$ for illustration. We begin by defining auxiliary variables based on the quantities in the chance constraint:

$$\tilde{s} = \Psi_t - \mu_\Psi(f_n^o) \quad (16a)$$

$$b = 1 - \mu_\Psi(f_n^o) \quad (16b)$$

We also define the set S describing the ambiguity in the first and second moments of \tilde{s} , and the set $\mathcal{D}_{\tilde{s}}$ of distributions consistent with these moments:

$$S = \left\{ (\mu_1, \sigma_1) : |\mu_1| \leq \sqrt{\gamma_1 \Sigma_\Psi}, \mu_1^2 + \sigma_1^2 \leq \gamma_2 \Sigma_\Psi \right\} \quad (17a)$$

$$\mathcal{D}_{\tilde{s}} = \left\{ \mathbb{P}_r \in \mathcal{P}(\mathbb{R}) : \begin{array}{l} |\mathbb{E}_{\mathbb{P}_r}[\tilde{s}]| \leq \sqrt{\gamma_1 \Sigma_\Psi} \\ \mathbb{E}_{\mathbb{P}_r}[\tilde{s}^2] \leq \gamma_2 \Sigma_\Psi \end{array} \right\} \quad (17b)$$

where $\mathcal{P}(\mathbb{R})$ is the set of all probability distributions on \mathbb{R} . Then, the original chance constraint $\inf_{\mathbb{P}_r \in \mathcal{P}_{\text{cal}}} \mathbb{P}_r(\Psi_t \leq 1) \geq 1 - \epsilon$ can be rewritten using \tilde{s} and b . The infimum term is:

$$\begin{aligned} \inf_{\mathbb{P}_r \in \mathcal{P}_{\text{cal}}} \mathbb{P}_r(\Psi_t \leq 1) &= \inf_{\mathbb{P}_r \in \mathcal{D}_{\tilde{s}}} \mathbb{P}_r(\tilde{s} \leq b) \\ &= \inf_{(\mu_1, \sigma_1) \in S} \inf_{\mathbb{P}_r \in \mathcal{P}(\mu_1, \sigma_1^2)} \mathbb{P}_r(\tilde{s} \leq b). \end{aligned} \quad (18)$$

where \mathcal{P}_{cal} refers to the set of distributions under multi-agent contextual intention uncertainty in the main paper. To this end, Eq. (18) formulates the problem as a bi-level optimization. The outer layer finds the worst-case mean μ_1 and variance σ_1^2 within the ambiguity set S . The inner layer finds the worst-case probability $\mathbb{P}_r(\tilde{s} \leq b)$ for a given mean and variance.

To solve the inner optimization problem, we employ the one-sided Cantelli's inequality. For a random variable X with mean $\mathbb{E}[X]$ and variance $\text{Var}[X] = \sigma^2$, Cantelli's inequality provides bounds on tail probabilities. Specifically, for the lower tail, the bound is defined as:

$$\mathbb{P}_r(X - \mathbb{E}(X) \geq -\lambda) \leq \frac{\sigma^2}{\sigma^2 + \lambda^2}$$

Applying this to our inner problem $\inf_{\mathbb{P}_r \in \mathcal{P}(\mu_1, \sigma_1^2)} \mathbb{P}_r(\tilde{s} \leq b)$, we get:

$$\inf_{\mathbb{P}_r \in \mathcal{P}(\mu_1, \sigma_1^2)} \mathbb{P}_r(\tilde{s} \leq b) = \begin{cases} \frac{(b - \mu_1)^2}{\sigma_1^2 + (b - \mu_1)^2}, & \text{if } b \geq \mu_1 \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

In practical multi-agent reinforcement learning scenarios, the desired confidence level $1 - \epsilon$ is positive. For the chance constraint $\inf_{\mathbb{P}_r} \mathbb{P}_r(\tilde{s} \leq b) \geq 1 - \epsilon$ to hold with $1 - \epsilon > 0$, we must be in the regime where $b \geq \mu_1$. If $b < \mu_1$, the infimum probability would be 0, violating the constraint. Therefore, for any relevant $(\mu_1, \sigma_1) \in S$, the condition $b \geq \mu_1$ must be satisfied. This implies $b \geq \sup_{(\mu_1, \sigma_1) \in S} \mu_1 = \sqrt{\gamma_1 \Sigma_h}$ (assuming $\Sigma_h > 0$). Consequently, Eq. (18) simplifies to:

$$\inf_{\mathbb{P}_r \in \mathcal{P}_{\text{cal}}} \mathbb{P}_r(\Psi_t \leq 1) = \inf_{(\mu_1, \sigma_1) \in S} \frac{(b - \mu_1)^2}{\sigma_1^2 + (b - \mu_1)^2} \quad (20)$$

Solving the optimization problem in Eq. (20) over the set S (defined by moment bounds $|\mu_1| \leq \sqrt{\gamma_1 \Sigma_h}$ and $\mu_1^2 + \sigma_1^2 \leq \gamma_2 \Sigma_h$) yields the following worst-case probability:

$$\inf_{\mathbb{P}_r \in \mathcal{P}_{\text{cal}}} \mathbb{P}_r(\Psi_t \leq 1) = \begin{cases} \frac{1}{\left(\frac{b/\sqrt{\Sigma_h} - \sqrt{\gamma_1}}{\sqrt{\gamma_2 - \gamma_1}} \right)^2 + 1}, & \text{if } \sqrt{\gamma_1} \leq \frac{b}{\sqrt{\Sigma_\Psi}} \leq \frac{\gamma_2}{\sqrt{\gamma_1}} \\ \frac{(b/\sqrt{\Sigma_h})^2 - \gamma_2}{(b/\sqrt{\Sigma_h})^2}, & \text{if } \frac{b}{\sqrt{\Sigma_\Psi}} > \frac{\gamma_2}{\sqrt{\gamma_1}} \\ \text{infeasible,} & \text{if } \frac{b}{\sqrt{\Sigma_\Psi}} < \sqrt{\gamma_1}. \end{cases}$$

The chance constraint requires this infimum probability to be at least $1 - \epsilon$: $\inf_{\mathbb{P}_r \in \mathcal{P}_{\text{cal}}} \mathbb{P}_r(\Psi_t \leq 1) \geq 1 - \epsilon$. By substituting the expressions above and performing algebraic manipulations, we arrive at the tractable second-order cone constraint. Substituting $b = 1 - \mu_\Psi(f_n^o)$, the condition becomes:

$$\mu_\Psi(f_n^o) + \beta \sqrt{\Sigma_\Psi} \leq 1 \quad (21a)$$

$$l_n = \begin{cases} \sqrt{\gamma_1} + \sqrt{\frac{1-\epsilon}{\epsilon}(\gamma_2 - \gamma_1)}, & \text{if } \gamma_1/\gamma_2 \leq \epsilon \leq 1 \\ \sqrt{\frac{\gamma_2}{\epsilon}}, & \text{if } 0 < \epsilon < \gamma_1/\gamma_2. \end{cases} \quad (21b)$$

534 This provides the final tractable form for the first inequality in Eq. (6) of the main paper. For
 535 this second-order cone constraint, we wrap it with a ReLU function to construct the following loss
 536 $\mathcal{L}_{\text{exploit}}^{\text{UB}}$, which forms an upper bound of the exploit loss. We substitute \bar{c} to recover the Eq. (7)
 537 presented in the main text.

538 To ensure stability and prevent variables Ψ_t from shrinking indefinitely, we also impose a lower-bound
 539 constraint $\mathbb{P}_r(\Psi_t \geq \underline{c}) \geq 1 - \epsilon$, which can be converted into loss $\mathcal{L}_{\text{exploit}}^{\text{LB}}$ in a similar manner.

540 C InfoNCE Bounds on Mutual Information

541 This appendix details how the InfoNCE objective provides lower and upper bounds for the mutual
 542 information $I([r_{t-1}^e, f_{n-1}^o]; f_n^o)$. We begin our derivation by establishing the lower bound relationship
 543 $I_{\text{nice}}(c) \leq I([r_{t-1}^e, f_{n-1}^o]; f_n^o)$. For notational simplicity, we denote the context $[r_{t-1}^e, f_{n-1}^o]$ as z_n .

544 In our work, the InfoNCE objective is defined using a critic function $c([r_{t-1}^e, f_{n-1}^o]; f_n^o)$, which
 545 measures the compatibility between the context $[r_{t-1}^e, f_{n-1}^o]$ and the current inferred intention f_n^o .
 546 In practice, $c(\cdot, \cdot)$ is implemented as a bilinear layer followed by a softmax, yielding scores in the
 547 range $[0, 1]$. Given a positive pair (f_n^o, z_n) drawn from the joint distribution $p(z_n, f_n^o)$, and a set of
 548 N negative samples $\mathcal{F}^- = \{f_n^{-o,j}\}_{j=1}^N$ drawn from the complement of f_n^o in its state space \mathcal{F} , the
 549 InfoNCE objective is defined as:

$$I_{\text{nice}}^c \triangleq \mathbb{E}_{p(z_n, f_n^o)} \mathbb{E}_{\mathcal{F}^-} \left[\log \frac{\exp(c(z_n, f_n^o))}{\sum_{j=1}^N \exp(c(z_n, f_n^{-o,j}))} \right]. \quad (22)$$

550 where the positive pairs (z_n, f_n^o) are obtained from the current prediction pair. Negative samples
 551 are generated by independently sampling N times from the state space of f_n^o . Let \mathbb{I} be an indicator
 552 variable, with $\mathbb{I} = 1$ indicating a positive sample randomly drawn from the state space, and $\mathbb{I} = 0$
 553 indicating a negative one (i.e., f_n^o is replaced by f_n^{-o}). The corresponding conditional probabilities
 554 have the following properties:

$$\begin{cases} p(z_n, f_n^o | \mathbb{I} = 1) = p(z_n, f_n^o). \\ p(z_n, f_n^{-o} | \mathbb{I} = 0) = p(z_n) p(f_n^o). \end{cases} \quad (23)$$

555 We aim to maximize the InfoNCE objective I_{nice}^c . We assume that the optimal critic c^* can identify
 556 the log-posterior probability of a randomly sampled pair being the positive one. Thus, we have:

$$\begin{aligned} I_{\text{nice}}^c &\leq I_{\text{nice}}^{c^*} \\ &\leq \mathbb{E}_{p(z_n, f_n^o)} \mathbb{E}_{\mathcal{F}^-} [c^*(z_n, f_n^o)] \\ &= \mathbb{E}_{p(z_n, f_n^o)} \mathbb{E}_{\mathcal{F}^-} [\log p(\mathbb{I} = 1 | z_n, f_n^o, \mathcal{F}^-)]. \end{aligned} \quad (24)$$

557 Next, we expand $\log p(\mathbb{I} = 1 | z_n, f_n^o, \mathcal{F}^-)$ using Bayes' theorem. Considering one positive sample
 558 and N negative samples, the prior probabilities are $p(\mathbb{I} = 1) = 1/(N+1)$ and $p(\mathbb{I} = 0) = N/(N+1)$
 559 for the collection of negative samples.

$$\begin{aligned} \log p(\mathbb{I} = 1 | z_n, f_n^o, \mathcal{F}^-) &= \log \frac{p(z_n, f_n^o | \mathbb{I} = 1) p(\mathbb{I} = 1)}{p(z_n, f_n^o | \mathbb{I} = 1) p(\mathbb{I} = 1) + p(z_n, f_n^{-o} | \mathbb{I} = 0) p(\mathbb{I} = 0)} \\ &= \log \frac{p(z_n, f_n^o)}{p(z_n, f_n^o) + N p(z_n) p(f_n^o)}. \end{aligned} \quad (25)$$

560 This expression for $\log p(\mathbb{I} = 1 | z_n, f_n^o, \mathcal{F}^-)$ represents the log-probability of the given f_n^o being the
 561 true positive, relative to N distractors drawn from $p(z_n) p(f_n^o)$. Continuing from this expression:

$$\begin{aligned} \log \frac{p(z_n, f_n^o)}{p(z_n, f_n^o) + N p(z_n) p(f_n^o)} &= \log \left(\frac{p(z_n, f_n^o)}{p(z_n) p(f_n^o)} \cdot \frac{p(z_n) p(f_n^o)}{p(z_n, f_n^o) + N p(z_n) p(f_n^o)} \right) \\ &\leq \log \frac{p(z_n, f_n^o)}{p(z_n) p(f_n^o)} - \log N. \end{aligned} \quad (26)$$

562 Taking the expectation $\mathbb{E}_{p(z_n, f_n^o)} \mathbb{E}_{\mathcal{F}^-}$ on both sides of the result from Eq. (24) combined with the
 563 inequality above:

$$\mathbb{E}_{p(z_n, f_n^o)} \mathbb{E}_{\mathcal{F}^-} [\log p(\mathbb{I} = 1 | z_n, f_n^o, \mathcal{F}^-)] \leq \mathbb{E}_{p(z_n, f_n^o)} \left[\log \frac{p(z_n, f_n^o)}{p(z_n) p(f_n^o)} \right] - \log N. \quad (27)$$

564 The term $\mathbb{E}_{p(z_n, f_n^o)} \left[\log \frac{p(z_n, f_n^o)}{p(z_n)p(f_n^o)} \right]$ is the definition of mutual information $I(z_n; f_n^o)$. Therefore, by
 565 substituting $z_n = [r_{t-1}^e, f_{n-1}^o]$ back into Eq. (24) and rewriting $I(z_n; f_n^o)$ with the original notation,
 566 we have:

$$I([r_{t-1}^e, f_{n-1}^o]; f_n^o) \geq I_{\text{ncc}}^c + \log N. \quad (28)$$

567 This demonstrates that the mutual information is lower-bounded by the InfoNCE objective I_{ncc}^c plus
 568 a term $\log N$. This justifies using I_{ncc}^c as a tractable surrogate to maximize a lower bound on the
 569 mutual information.

570 Similarly, the upper bound inequality can be derived using the same method.

$$\begin{cases} I([r_{t-1}^e, f_{n-1}^o]; f_n^o) \leq I_{\text{ncc}}^{1-c} \\ I_{\text{ncc}}^{1-c} \triangleq \log N - \mathbb{E}_{p(z_n, f_n^o)} \mathbb{E}_{\mathcal{F}^-} \left[\log \frac{\exp(1-c(z_n, f_n^o))}{\sum_{j=1}^N \exp(1-c(z_n, f_n^{o,j}))} \right] \end{cases} \quad (29)$$

571 D Intrinsic Reward in Linear MDPs

572 This appendix provides a theoretical justification for our proposed intrinsic reward by connecting it to
 573 the exploration bonus in LSVI-UCB within the context of linear MDPs.

574 D.1 Preliminary

575 Least-Squares Value Iteration with Upper Confidence Bounds (LSVI-UCB) is an algorithm designed
 576 for efficient exploration and learning in linear MDPs. In a linear MDP, the transition kernel and
 577 reward function are assumed to be linear with respect to a d -dimensional feature map $\eta(s, a)$ of state-
 578 action pairs. Consequently, for any policy π , the action-value function $Q^\pi(s, a)$ can be expressed as
 579 $\chi^\top \eta(s, a)$ for some parameter vector $\chi \in \mathbb{R}^d$.

580 LSVI-UCB iteratively collects data and updates the Q -function parameters. In each episode, the
 581 agent acts according to the current optimistic Q -function. The parameter χ_t is then updated via
 582 regularized least-squares:

$$\chi_t \leftarrow \arg \min_{\chi \in \mathbb{R}^d} \sum_{i=0}^m \left[r_t(s_t^i, a_t^i) + \gamma \max_{a'} Q_t^{\text{target}}(s_{t+1}^i, a') - \chi^\top \eta(s_t^i, a_t^i) \right]^2 + \lambda \|\chi\|_2^2,$$

583 where m is the number of collected transitions (indexed by i), $\lambda > 0$ is a regularization parameter, and
 584 Q_t^{target} is typically a previous estimate or a slowly updating target. The closed-form solution involves
 585 the Gram matrix $\Lambda_t = \sum_{i=0}^m \eta(s_t^i, a_t^i) \eta(s_t^i, a_t^i)^\top + \lambda I$. Crucially, LSVI-UCB employs a UCB-style
 586 exploration bonus to construct an optimistic Q -function: $Q_t(s, a) = \chi_t^\top \eta(s, a) + r^{\text{ucb}}(s, a)$, where
 587 the bonus is $r^{\text{ucb}}(s, a) = \zeta [\eta(s, a)^\top \Lambda_t^{-1} \eta(s, a)]^{1/2}$. This bonus quantifies the epistemic uncertainty
 588 associated with the value estimate of (s, a) .

589 D.2 Connection to LSVI-UCB Bonus

590 We establish a connection between our intrinsic reward and the LSVI-UCB exploration bonus. In
 591 linear MDPs, CERMIC's parameters Θ is rewritten as ω_t and our curiosity representation x_t can
 592 be represented as $x_t = \omega_t \eta(s_t, a_t) \in \mathbb{R}^c$. Here, $\eta(s_t, a_t) \in \mathbb{R}^d$ is the state-action encoding, and
 593 $\omega_t \in \mathbb{R}^{c \times d}$ is a parameter matrix. This representation is learned by predicting the next state s_{t+1} via
 594 the regularized least-squares problem:

$$\omega_t \leftarrow \arg \min_W \sum_{i=0}^m \left\| s_{t+1}^i - \omega \eta(s_t^i, a_t^i) \right\|_F^2 + \lambda \|\omega\|_F^2, \quad (30)$$

595 The proof proceeds by vectorizing the matrix ω_t and defining an expanded feature matrix $\tilde{\eta}$. Specif-
 596 ically, $\text{vec}(\omega_t) \in \mathbb{R}^{cd}$ is the column-wise vectorization of ω_t , and $\tilde{\eta}(s_t, a_t) \in \mathbb{R}^{cd \times c}$ is a block-
 597 diagonal matrix with $\eta(s_t, a_t)$ repeated c times along its diagonal. This construction satisfies
 598 $\text{vec}(\omega_t)^\top \tilde{\eta}(s_t, a_t) = (\omega_t \eta(s_t, a_t))^\top$. For clarity, the definitions are:

$$\begin{cases} \text{vec}(\omega_t) = [w_{11}, \dots, w_{1d}, w_{21}, \dots, w_{cd}]^\top \in \mathbb{R}^{cd} \\ \tilde{\eta}(s_t, a_t) = \mathbf{I}_c \otimes \eta(s_t, a_t) \in \mathbb{R}^{cd \times c} \end{cases} \quad (31)$$

599 where \mathbf{I}_c denotes the identity matrix (with c dimensions), and \otimes is the Kronecker product.

Assumption 1 (Gaussian Prior) We consider a linear model for predicting the c -dimensional next state s_{t+1} from d -dimensional state-action features $\eta(s_t, a_t)$, formulated as $s_{t+1} = W\eta(s_t, a_t) + \xi_t$. The parameter matrix $W \in \mathbb{R}^{c \times d}$ is assumed to follow a zero-mean Gaussian prior distribution $W \sim \mathcal{N}(0, \lambda^{-1}I)$, and the model noise $\xi_t \in \mathbb{R}^c$ is assumed to follow a standard multivariate Gaussian distribution $\xi_t \sim \mathcal{N}(0, I_c)$, independent of W and $\eta(s_t, a_t)$.

To analyze the intrinsic reward, we adopt a Bayesian linear regression perspective for Eq. (30). Our goal of the analysis is to obtain the posterior distribution $p(\omega_t | \mathcal{D}_m)$ to compute the Bayesian Surprise in the intrinsic reward. Firstly, under Assumption 1, we have:

$$s_{t+1} | (s_t, a_t), \omega_t \sim \mathcal{N}(\omega_t \eta(s_t, a_t), \mathbf{I}) \equiv \mathcal{N}(\tilde{\eta}(s_t, a_t)^\top \text{vec}(\omega_t), \mathbf{I}). \quad (32)$$

Given the Gaussian prior $\text{vec}(W) \sim \mathcal{N}(0, \lambda^{-1}I)$, Bayes' rule states:

$$\log p(\text{vec}(\omega_t) | \mathcal{D}_m) = \log p(\text{vec}(\omega_t)) + \sum_{i=0}^m \log p(s_{t+1}^i | s_t^i, a_t^i, \text{vec}(\omega_t)) + C. \quad (33)$$

where C is a constant. Then, substituting the Gaussian PDF for Eq. (32) into Eq. (33) yields the log-posterior:

$$\begin{aligned} \log p(\text{vec}(\omega_t) | \mathcal{D}_m) &= -\frac{\lambda}{2} \|\text{vec}(\omega_t)\|_2^2 - \frac{1}{2} \sum_{i=0}^m \|\tilde{\eta}(s_t^i, a_t^i)^\top \text{vec}(\omega_t) - s_{t+1}^i\|_2^2 + C \\ &= -\frac{1}{2} (\text{vec}(\omega_t) - \tilde{\mu}_t)^\top \tilde{\Lambda}_t (\text{vec}(\omega_t) - \tilde{\mu}_t) + C', \end{aligned} \quad (34)$$

where C' is a constant and $\tilde{\mu}_t = \tilde{\Lambda}_t^{-1} \sum_{i=0}^m \tilde{\eta}(s_t^i, a_t^i) s_{t+1}^i$, $\tilde{\Lambda}_t = \sum_{i=0}^m \tilde{\eta}(s_t^i, a_t^i) \tilde{\eta}(s_t^i, a_t^i)^\top + \lambda \mathbf{I}$. Thus, the posterior distribution is $\text{vec}(\omega_t) | \mathcal{D}_m \sim \mathcal{N}(\tilde{\mu}_t, \tilde{\Lambda}_t^{-1})$. The covariance matrix of $\text{vec}(\omega_t)$ is $\tilde{\Lambda}_t^{-1}$. The structure of $\tilde{\Lambda}_t$ is:

$$\tilde{\Lambda}_t = \mathbf{I}_n \otimes \left(\sum_{i=0}^m \eta(s_t^i, a_t^i) \eta(s_t^i, a_t^i)^\top + \lambda \mathbf{I} \right) = \mathbf{I}_n \otimes \Lambda_t, \quad (35)$$

Notably, $\Lambda_t = \sum_{i=0}^m \eta(s_t^i, a_t^i) \eta(s_t^i, a_t^i)^\top + \lambda \mathbf{I}$ is the Gram matrix from LSVI-UCB. To this end, the intrinsic reward, related to Bayesian Surprise, can be simplified to a change in differential entropy, where $\text{Cov}(\cdot)$ denotes the covariance:

$$\begin{aligned} &\mathcal{D}_{KL}(p(\omega_t | (s_t, a_t, s_{t+1}) \cup \mathcal{D}_m) \| p(\omega_t | \mathcal{D}_m)) \\ &= \mathcal{D}_{KL}(p(\text{vec}(\omega_t) | (s_t, a_t, s_{t+1}) \cup \mathcal{D}_m) \| p(\text{vec}(\omega_t) | \mathcal{D}_m)) \\ &= \mathcal{H}(\text{vec}(\omega_t) | \mathcal{D}_m) - \mathcal{H}(\text{vec}(\omega_t) | (s_t, a_t, s_{t+1}) \cup \mathcal{D}_m) \\ &= [\log \det(\text{Cov}(\text{vec}(\omega_t) | \mathcal{D}_m)) - \log \det(\text{Cov}(\text{vec}(\omega_t) | (s_t, a_t, s_{t+1}) \cup \mathcal{D}_m))] / 2. \end{aligned} \quad (36)$$

Substituting the covariance $\tilde{\Lambda}_t^{-1}$ and the updated covariance after observing (s_t, a_t, s_{t+1}) :

$$\begin{aligned} &\mathcal{D}_{KL}(p(\omega_t | (s_t, a_t, s_{t+1}) \cup \mathcal{D}_m) \| p(\omega_t | \mathcal{D}_m)) \\ &= [\log \det(\tilde{\Lambda}_t + \tilde{\eta}(s_t, a_t) \tilde{\eta}(s_t, a_t)^\top) - \log \det(\tilde{\Lambda}_t)] / 2 \\ &= [\log \det(\mathbf{I} + \tilde{\eta}(s_t, a_t)^\top \tilde{\Lambda}_t^{-1} \tilde{\eta}(s_t, a_t))] / 2, \end{aligned} \quad (37)$$

where the last equality follows from the Matrix Determinant Lemma. Given the block-diagonal structure of $\tilde{\eta}(s_t, a_t)$ and $\tilde{\Lambda}_t^{-1}$, the term $\tilde{\eta}(s_t, a_t)^\top \tilde{\Lambda}_t^{-1} \tilde{\eta}(s_t, a_t)$ is a diagonal matrix with c identical scalar entries $\eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t)$. Thus, Eq. (37) simplifies to:

$$\begin{aligned} &\mathcal{D}_{KL}(p(\omega_t | (s_t, a_t, s_{t+1}) \cup \mathcal{D}_m) \| p(\omega_t | \mathcal{D}_m)) \\ &= [\log \det(\mathbf{I}_c \otimes \eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t) + \mathbf{I}_c)] / 2 \\ &= (c/2) \cdot \log(1 + \eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t)). \end{aligned} \quad (38)$$

By leveraging the inequality $\frac{x}{2} \leq \log(1+x) \leq x$, we obtain:

$$\frac{\eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t)}{2} \leq \log(1 + \eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t)) \leq \eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t). \quad (39)$$

Combining these results, we obtain the bounds for the square root of the mutual information:

$$\sqrt{\frac{c}{4}} [\eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t)]^{1/2} \leq r_t^i \leq \sqrt{\frac{c}{2}} [\eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t)]^{1/2}. \quad (40)$$

Recalling that the LSVI-UCB bonus is $r^{\text{ucb}} = \zeta [\eta(s_t, a_t)^\top \Lambda_t^{-1} \eta(s_t, a_t)]^{1/2}$, we can express the relationship as:

$$\frac{1}{\zeta} \sqrt{\frac{c}{4}} \cdot r^{\text{ucb}} \leq r_t^i \leq \frac{1}{\zeta} \sqrt{\frac{c}{2}} \cdot r^{\text{ucb}}. \quad (41)$$

Defining $\rho = \frac{1}{\zeta} \sqrt{\frac{c}{4}}$, this can be written as $\rho \cdot r^{\text{ucb}} \leq r_t^i \leq \sqrt{2} \rho \cdot r^{\text{ucb}}$. This proof establishes a connection between our proposed intrinsic reward and the UCB bonus, providing theoretical grounding for the stability of our reward mechanism.

D.3 Empirical Implementation

The intrinsic reward r^i relies on a Bayesian view of model parameters Θ . As our CERMIC model uses non-Bayesian neural networks, this section outlines a practical approximation for r^i .

While directly computing r^i is challenging, we address this by deriving a computable lower bound using the Data Processing Inequality (DPI), which states that post-processing cannot increase information. Applying a learned representation function $\mathcal{R}(s_t, a_t)$ (a part of CERMIC) to the transition (s_t, a_t, s_{t+1}) , the DPI yields:

$$\begin{aligned} r^i(s_t, a_t) &= \left[\mathcal{H}((s_t, a_t, s_{t+1}) | \mathcal{D}_m) - \mathcal{H}((s_t, a_t, s_{t+1}) | \Theta, \mathcal{D}_m) \right]^{1/2} \\ &\geq \left[\mathcal{H}(\mathcal{R}(s_t, a_t, s_{t+1}) | \mathcal{D}_m) - \mathcal{H}(\mathcal{R}(s_t, a_t, s_{t+1}) | \Theta, \mathcal{D}_m) \right]^{1/2} \triangleq r_{\text{approx}}^i. \end{aligned} \quad (42)$$

The approximated reward r_{approx}^i thus becomes the KL divergence between the conditional representation $\mathcal{R}(x_t | s_t, a_t)$ and its marginal $p_{\text{margin}}(x_t | \mathcal{D}_m)$:

$$r_{\text{approx}}^i(s_t, a_t) = \mathbb{E}_\Theta [D_{\text{KL}}(\mathcal{R}_\phi(x_t | s_t, a_t) \| p_{\text{margin}}(x_t | \mathcal{D}_m))]^{1/2}. \quad (43)$$

The marginal $p_{\text{margin}}(x_t | \mathcal{D}_m)$ over model parameters is intractable. Following common practice in information-theoretic exploration with non-Bayesian models, we approximate $p_{\text{margin}}(x_t | \mathcal{D}_m)$ with a fixed standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$. This renders Eq. (43) empirically computable.

E Benchmark Adaptation to Sparse-Reward Settings

Our CERMIC model and all compared algorithms were trained in environments with sparse reward signals to specifically assess exploration capabilities. However, for a comprehensive evaluation against baselines, particularly those not designed for extreme sparsity, final performance was measured using the original dense extrinsic rewards provided by the benchmarks. To facilitate the sparse-reward training, we adapted standard MARL benchmarks, VMAS and SMACv2, into sparse-reward configurations as detailed below. This transformation creates challenging scenarios where intrinsic motivation is paramount for effective learning.

VMAS Benchmarks. For VMAS tasks that originally featured dense rewards (e.g., based on distance-to-target or control efforts), we induced sparsity by identifying a clear success condition for each task and restructuring the reward mechanism accordingly. Specifically, original dense or intermediate reward components were removed or nullified, except for a significant positive reward granted *only* upon the achievement of the predefined success condition (e.g., reaching a target zone, achieving a formation). Episode termination conditions remained largely unchanged, but rewards became predominantly contingent on successful terminal states. For instance, in a Balance* task, agents receive a large positive reward only upon moving the object to the goal region, with zero or minimal rewards during transit. This approach ensures that learning signals are tied to meaningful task completion rather than continuous incremental progress.

SMACv2 Benchmarks. Similarly, for SMACv2 scenarios, while many already possess somewhat sparse rewards, we further amplified sparsity to stringently test exploration. The primary reward

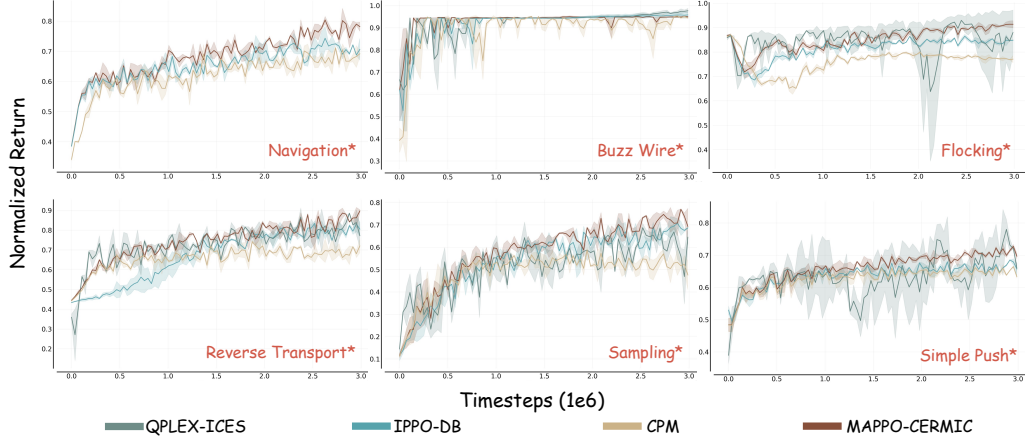


Figure 6: Comparative performance on VMAS.

signal was strictly tied to the ultimate battle outcome: a large positive reward for winning, a negative or zero reward for losing, and a neutral or slightly negative reward for a draw. All intermediate battlefield rewards, such as bonuses for damaging enemy units or penalties for sustaining damage, were removed. This concentrates the learning signal at the conclusion of an engagement, compelling agents to learn long-term coordination and strategy based purely on the sparse win/loss feedback, especially in scenarios where incremental damage yields no immediate reward.

These modifications ensure that agents in both benchmarks must explore extensively to discover successful action sequences, as intermediate feedback is largely absent, thus providing a robust testbed for curiosity-driven mechanisms like CERMIC.

F Supplementary Experimental Results

Performance Curve Comparison. We present further comparative experiments showcasing the performance of CERMIC against other methods in Fig. 6 below. The results demonstrate that CERMIC consistently surpasses prior SoTA models and other curiosity-driven approaches, achieving superior task performance. Furthermore, it is observable that our method exhibits more stable learning curves, characterized by narrower error bands. We attribute this enhanced stability and performance to CERMIC’s ability to effectively filter noisy signals while concurrently encouraging robust exploration.

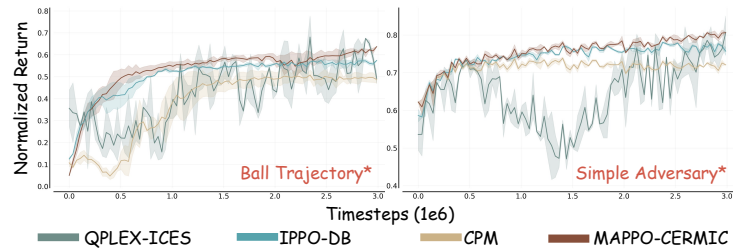


Figure 7: Performance comparison under environmental noise on VMAS task.

Robustness to Noise. To further validate the noise-filtering capabilities of CERMIC, we conducted comparative experiments in environments augmented with “random box” noise, as depicted in Fig. 7. The results indicate that, in these noisy conditions, CERMIC maintains more stable learning curves and narrower error bands compared to other algorithms. Notably, this advantage in stability is more pronounced than in the noise-free task counterparts (Fig. 6). Furthermore, we observe that the error bands in the Simple Adversary task are smaller than those in the Ball Trajectory task. This difference can be attributed to the richer multi-agent contextual information available in the former, which includes signals from both adversaries and teammates, whereas the latter primarily involves

information from a single partner. Consequently, the increased contextual feedback translates to a stronger noise-filtering effect, as reflected in the learning curves. This underscores CERMIC’s ability to leverage multi-agent information to enhance task understanding and mitigate the impact of environmental stochasticity.

Role Diversity and Task Adaptation. While CERMIC consistently improves performance, we observe a more substantial performance gain over traditional baselines in tasks involving diverse agent roles (e.g., both cooperation and competition, as in *Simple Adversary**) compared to purely cooperative/competitive settings (e.g., *Ball Trajectory**); see Fig. 7. We attribute this to the challenge faced by traditional methods in interpreting heterogeneous agent behaviors under partial observability, where inferring intent and role from observations is inherently difficult. In contrast, CERMIC’s task-adaptive weighting mechanism (γ) enables agents to better infer and adapt to others’ intentions and plans, thereby enhancing performance in complex social interactions.

Incremental Enhancement. We conducted an ablation study to analyze the incremental impact of CERMIC’s components on task performance. This involved comparing MAPPO, MAPPO-CERMIC, and MAPPO-CERMIC with a pre-trained memory module. The results, presented in Fig. 8, intuitively illustrate the contribution of each component to the baseline algorithm’s performance. It can be observed that the core CERMIC module primarily serves to elevate overall performance, while also providing a modest improvement in learning stability. The subsequent integration of a pre-trained memory module further enhances this stability, leading to more consistent learning trajectories.

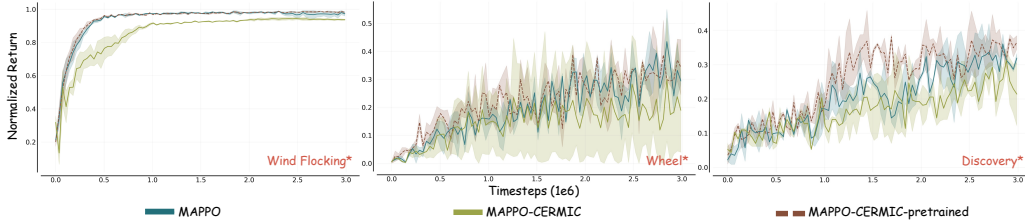


Figure 8: Incremental impact of CERMIC components on VMAS tasks.

Testing Agent Generalization. We further investigated the generalization capabilities of agents by evaluating their performance when trained under one reward density (dense or sparse) and tested under a sparse-reward setting. The results are presented in Table 3. A consistent trend observed across all algorithms is a performance degradation when agents trained on dense rewards are deployed in sparse-reward scenarios. However, CERMIC demonstrates a notable ability to mitigate this performance drop. We attribute this to CERMIC’s capability to enable agents to understand the task by observing other agents, rather than solely relying on trial-and-error learning. These findings underscore the importance of research in sparse-reward reinforcement learning and highlight the potential for CERMIC’s broad applicability.

Table 3: **Agent generalization performance on VMAS tasks.** Δ denotes the performance drop from dense-trained to sparse-trained.

Method	Balance*			Give Way*			Passage*		
	Sparse \uparrow	Dense \uparrow	Δ \downarrow	Sparse \uparrow	Dense \uparrow	Δ	Sparse \uparrow	Dense \uparrow	Δ \downarrow
CPM	61.0	58.2	-2.8	3.73	3.58	-0.15	162	153	-9
QPLEX-ICES	63.2	60.7	-2.5	3.96	3.85	-0.11	164	159	-5
MAPPO-CERMIC	67.3	65.6	-1.7	3.94	3.82	-0.12	172	168	-4

Visualization of Intrinsic Reward. To illustrate the temporal behavior of CERMIC’s intrinsic reward, we visualized its values throughout a complete episode of the *Balance** task, as shown in Fig. 9. A clear trend is the gradual decrease of this intrinsic reward over time, indicating a diminishing role of curiosity-driven exploration as the agent gains familiarity with the task environment. Furthermore, peaks in the intrinsic reward consistently coincide with an agent encountering novel situations, such as initial environmental entry or first-time interactions with other agents. This observation aligns with our design objective for the intrinsic reward, which is to incentivize exploration of unfamiliar states and interactions.

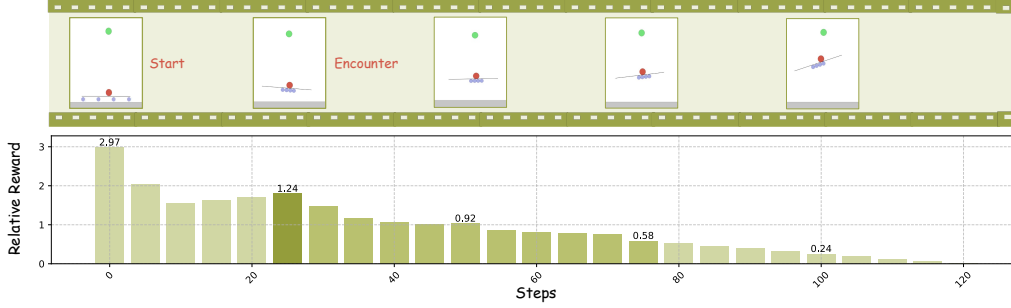


Figure 9: Visualization of CERMIC’s intrinsic reward during one episode of the Balance* task.

G Alternative Memory Architectures

While graph-based memory offers structured relational reasoning, alternative architectures provide different trade-offs. Table 4 compares key architectural parameters with typical example values for these memory types.

MLP-based Memory. An MLP processes the agent’s current (potentially encoded) observation to output a fixed-size memory vector. This is a stateless, feed-

GRU-based Memory. A Gated Recurrent Unit (GRU) maintains a hidden state that is updated at each timestep based on the current encoded observation and its previous state, capturing temporal dependencies.

Table 4: Architectural parameters for different memory modules.

Parameter Type	Graph-based	MLP-based	GRU-based
Core Encoding			
Node Embedding Dim (D_{node})	256	-	-
Edge Embedding Dim (D_{edge})	16	-	-
Obs. Encoder Output Dim (D_{obs_enc})	64 (for detection/encoding)	64	64
Memory/State Output Dim	512 (D_{GNN_out})	256 (D_M)	256 (D_H)
Network Structure			
Obs. Encoder (CNN) Layers/Depth	3 conv layers	3 conv layers	3 conv layers
GNN Layers (L_{GNN})	2	-	-
GNN Hidden Dim (D_{GNN_hid})	256	-	-
GNN Heads (N_{heads})	4	-	-
MLP Hidden Layers (L_{MLP})	(Node/Edge Encoders: 2)	2 (post obs-encoder)	-
MLP Hidden Units/Layer (U_{MLP})	(Node/Edge Encoders: 128)	128	-
GRU Layers (L_{GRU})	-	-	2
Est. Total Parameters	High	Low	Medium

The Graph-based module is generally the most expressive due to its explicit relational structure but also tends to be the most parameter-heavy. MLP-based memory is the simplest, while GRU-based memory offers a balance for capturing temporal context. The choice depends on the specific requirements of the MARL task, including the complexity of inter-agent interactions and available computational resources.

H Implementation Details

For brevity, we previously use abbreviated task names in Table 1. Their full names are as follows: Disper = Disperse, Naviga = Navigation, Sampli = Sampling, Passag = Passage, Transp = Transport, Balanc = Balance, GiveWa = GiveWay, Wheel = WheelGathering, Flocki = Flocking (*: sparse-reward version), StaHun = StagHuntRepeated, CleaUp = Cleanup, ChiGam = ChickenGameRepeated, PriDil = PrisonersDilemmaRepeated, pro5v5 = Protoss5v5, and zer5v5 = Zerg5v5.

Table 5: Key training hyperparameters.

Parameter Category	Value / Setting
<i>General Optimization</i>	
Discount Factor (γ)	0.99
Learning Rate (Adam)	1e-4
Adam Optimizer ϵ	1e-6
<i>Target Network Updates</i>	
Update Type	Soft (Polyak averaging)
Polyak Tau (τ)	0.005
<i>Exploration Strategy</i>	
Initial Epsilon (ϵ_{init})	0.8
Final Epsilon (ϵ_{end})	0.01
<i>Training Duration</i>	
Max Frames	3000000
<i>On-Policy Collection & Training</i>	
Collected Frames per Batch	36000
Environments per Worker	60
Minibatch Iterations	30
Minibatch Size	2400
<i>Off-Policy Collection & Training</i>	
Optimizer Steps per Collection	1800
Train Batch Size	1024
Replay Buffer Size	1500000

741 This paragraph outlines the core hyperparameters used for training our models and baselines, unless
 742 specified otherwise for particular tasks or algorithms. Specific parameters related to intrinsic rewards
 743 (e.g., exploitation weight factor α) are detailed separately in the experimental setup as they vary
 744 across tasks and model configurations. Training/evaluation was conducted via BenchMARL suite on
 745 NVIDIA Quadro RTX 8000 GPUs.