2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

MIMIC-D: Multi-modal Imitation for MultI-agent Coordination with Decentralized Diffusion Policies

Dayi Dong*, Maulik Bhatt*, Seoyeon Choi, and Negar Mehr

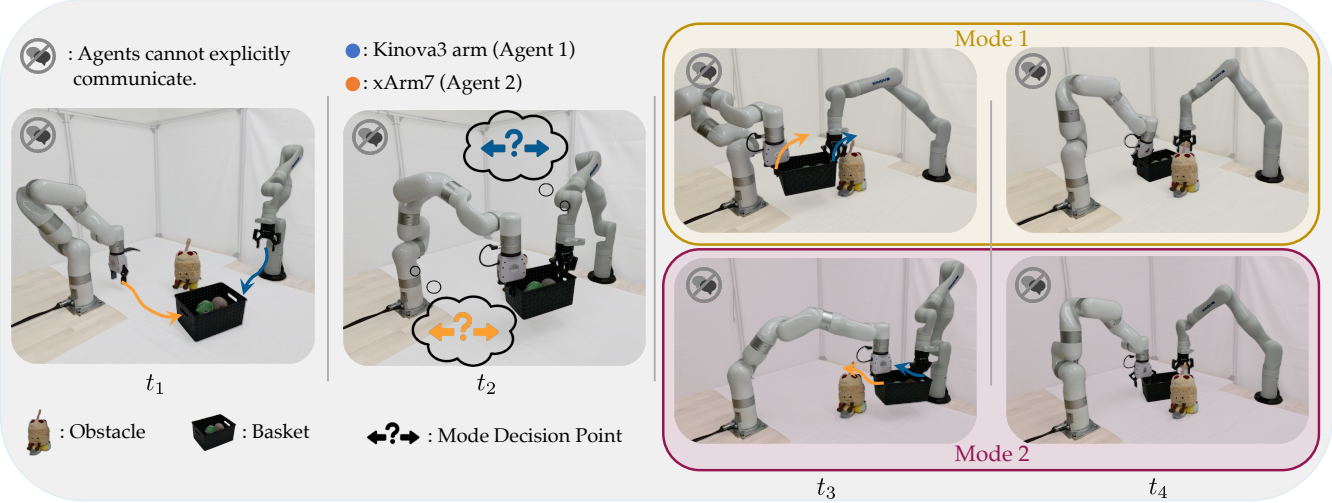


Fig. 1: **MIMIC-D deployed on a bimanual manipulation setup.** An xArm7 and a Kinova3 robotic arm collaborate to lift a basket around an obstacle. The task presents a multi-modal coordination challenge as the arms need to coordinate to either pass the obstacle on the right or on the left. Using our method, the arms achieve coordination by independently sampling their policies based on local observations without explicit communication. Our method successfully recovers both solution modes, demonstrating its ability to capture diverse coordination strategies while avoiding the freezing robot problem. The panels depict the starting configuration (t_1), mode decision (t_2), and the completion of both modes (t_3 , t_4).

***Abstract*— As robots become more integrated in society, their ability to coordinate with other robots and humans on multi-modal tasks (those with multiple valid solutions) is crucial. Such behaviors can be learned from expert demonstrations via imitation learning (IL), but when expert demonstrations are multi-modal, standard IL approaches usually average across modes or collapse to a single mode, preventing effective coordination. Being inspired by diffusion models' ability to capture complex multi-modal trajectory distributions in single-agent settings, we develop a diffusion-based framework for coordinated multi-modal behavior in multi-agent systems. However, existing multi-agent diffusion approaches typically require a centralized planner or explicit communication among agents. This assumption can fail in real-world scenarios where robots must operate independently or with agents like humans that they cannot directly communicate with. Therefore, we propose MIMIC-D, a joint training with decentralized execution paradigm for multi-modal multi-agent IL via diffusion. We jointly train all agents' policies with only local information to achieve implicit coordination. In simulation and hardware experiments, our method exhibits robust multi-modal coordination behavior in various tasks and environments, improving upon state-of-the-art baselines.**

All authors are with the Department of Mechanical Engineering, University of California Berkeley, Berkeley, CA 94709, USA {dayi.dong, maulikbhatt, seoyeon99, negar}@berkeley.edu

This work was supported by the National Science Foundation under Grants ECCS-2438314 (CAREER Award), CNS-2529645, and CCF-2423134, and by the Army Research Laboratory under Grant W911NF-26-1-0002.

*Indicates equal contribution.

I. Introduction

As society actively embraces new robotic developments in our everyday lives, these robots must learn to coordinate with humans and other agents in shared spaces where multiple coordination strategies may exist. For example, two people approaching head-on can avoid collision by both yielding right or both yielding left. Either strategy is acceptable, but the agents need to decide together to achieve the desired *coordination* because contradicting strategies lead to failure. In many scenarios, there is no reliable central planner available, so robots must implicitly coordinate multi-modal behaviors. In this paper, we propose to learn multi-modal coordination policies from expert data of multi-agent interactions, allowing robots to perform robust, decentralized execution without explicit communication.

Many approaches for solving this multi-agent coordination problem have been proposed. Learning-based methods have propelled this field forward by introducing reinforcement learning (RL) [1] and temporal-difference (TD) methods [2] when it is feasible to write an explicit reward function. Imitation learning (IL) methods like behavior cloning (BC) [3] and generative adversarial imitation learning (GAIL) [4] instead utilize expert demonstrations to learn a desired behavior. These methods, however, suffer from poor performance in complex long-horizon tasks [5], [6], [7], and struggle with multi-modal expert data, where multiple valid solution modes

exist [8], [9], [10], [11] due to mode-averaging (learning to output the average of distinct strategies) and mode collapse (perfectly imitating only one of the expert's behaviors).

Diffusion models [12] are a more recent approach in IL, originally developed for image generation because of their ability to model complex multi-modal data distributions. More recently, diffusion policy models [13], [14] have been developed to model expert behaviors for robotics. This approach is particularly compelling because it can capture multi-modality present in expert demonstrations, which is essential for successfully manipulating objects [13], [14].

We argue that *multi-modality is even more common in multi-agent systems* and more essential because if you cannot capture all of the modes, there is a greater chance of task failure. Therefore, leveraging the multi-modal capabilities of diffusion models for multi-agent systems is even more essential to ensure robust multi-agent coordination. While diffusion models have been studied in the context of multi-agent systems [14], [15], most approaches assume a centralized framework where agents' observations and actions are jointly available during the task execution [15]. However, many real-world deployments, like driving or human-robot interactions, are inherently decentralized because they lack a concrete way to communicate information, making a centralized policy unrealistic.

To address this, we propose **MIMIC-D**: **M**ulti-modal **I**mitation for **M**ult**I**-agent **C**oordination with **D**ecentralized Diffusion Policies (see Fig. 2), a joint training with decentralized execution framework for multi-modal multi-agent diffusion-based planning. Our approach ensures that agents can implicitly coordinate without explicit communication. We propose training all the agents' policies jointly so that they can build up an understanding of how to respond to one another. Then, during execution, each agent only needs its own policy and its local observations to plan out its actions. Through extensive investigations in navigation and bimanual manipulation tasks, like in Fig 1, we find that MIMIC-D achieves significantly lower collision rates and higher task completion rates due to its ability to foster coordination among agents. Additionally, we can better capture the distribution of multi-modal expert trajectories than baseline methods. In summary, our contributions are as follows:

i. A joint training with decentralized execution framework for multi-agent coordination using only local observations that effectively captures multi-modality in the expert data.
ii. On two and three-agent 2D navigation and simulated bimanual manipulation tasks, compared to other approaches, we achieve fewer collisions and better replication of the expert demonstration distribution.
iii. On a hardware bimanual manipulation task that requires coordination to move a basket, MIMIC-D achieves 95% success in 20 trials.

II. Related Work

1) Multi-agent Reinforcement Learning (MARL): In multi-agent reinforcement learning (MARL), agents learn policies through interaction. Methods like Independent Q-Learning [16] treat other agents as part of the environment, which can lead to non-stationary and instability [17]. Centralized Training with Decentralized (CTDE) methods (e.g., MADDPG [18], QMIX [19], MAPPO [20]) can stabilize the learning process by using centralized critics during training and decentralized actors during execution, but they still require manually constructable reward functions for all the agents and extensive reward shaping [21], [22].

2) Multi-agent Imitation Learning (MAIL): Multi-agent imitation learning instead leverages expert demonstrations to guide the planning of individual agents. Approaches like behavior cloning (BC) [3], [5] are simple but suffer from compounding errors in long-horizon tasks and fail to model inter-agent dependencies required for coordination. Generative adversarial IL (GAIL) [4], and by extension, multi-agent GAIL (MAGAIL) [23], can avoid significant compounding error but exhibit mode collapse when demonstrations are multi-modal.

Other than choosing an IL objective, selecting the training paradigm also impacts performance. Several works adopt the CTDE framework to address the MAIL problem (e.g., MisoDICE[24]) by leveraging joint trajectories during training. However, these methods learn policies by matching the expert's state-action occupancy measure, resulting in nearly deterministic policies at execution time and limiting recovery of multi-modal behavior. We instead seek to adopt a generative approach to learn trajectory distributions that preserves the multi-modal nature of the expert demonstrations.

3) Diffusion: Diffusion has shown great promise for policy learning [13], [14]. Denoising Diffusion Probabilistic Models (DDPM) [12] are the core framework in diffusion models, but additional works have also demonstrated smooth robot behavior and strong multi-modality (e.g., Diffusion Policy [14]). These works primarily assume a single agent and access to the full state during execution.

Recent works extend diffusion to the multi-agent setting by explicitly modeling interactions and performing centralized execution by sampling a joint multi-agent trajectory distribution (e.g., MotionDiffuser [15], DJINN [25]). MADiff [26] and LatentToM [27] are two works that adapt diffusion models towards a CTDE framework. MADiff utilizes teammate/opponent modeling and an attention-based diffusion model to handle interactions among agents, thereby achieving coordination. LatentToM introduces a consensus embedding that is based on shared observations from a third-party camera to align beliefs, acting as a bridge between the diffusion models. In contrast, we learn decentralized diffusion policies only conditioned on local observations during execution while still capturing the multi-modality inherent in interactions.

III. Problem Formulation

We study the problem of learning multi-modal policies for multi-agent coordination from expert data. Let N be the number of agents and $\mathcal{S} = \mathcal{S}^1 \times \ldots \times \mathcal{S}^N$ be the joint state space where $\mathcal{S}^i \subseteq \mathbb{R}^{n_i}$ is the state space for agent i with n_i

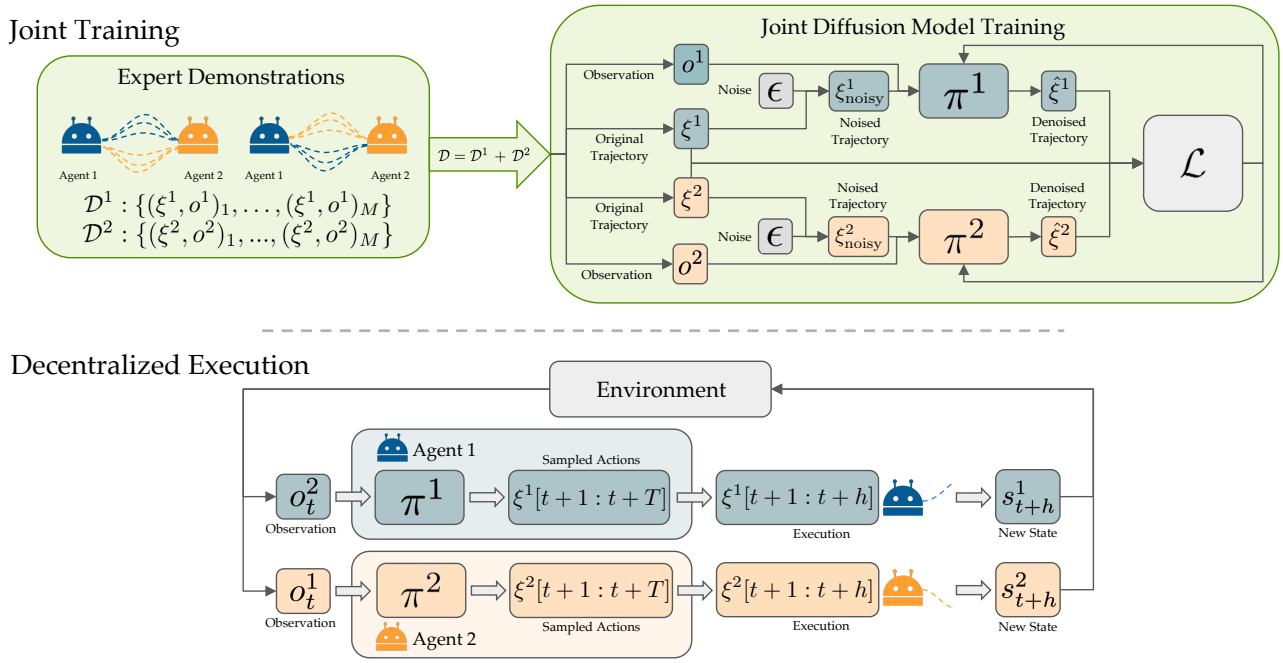


Fig. 2: **An overview of our MIMIC-D framework.** In the training process (top), we utilize a dataset of multi-agent expert demonstrations to train the robot policies jointly. During the decentralized execution process (bottom), each agent plans their trajectory independently by only making use of its local observations to sample the diffusion model.

being the dimension of the state space of agent i. Let $\mathcal{A} = \mathcal{A}^1 \times \ldots \times \mathcal{A}^N$ be the joint action space, and $\mathcal{A}^i \subseteq \mathbb{R}^{m_i}$ is the action space for agent i with m_i being the dimension of the action-space for agent i and $\mathcal{O} = \mathcal{O}^1 \times \ldots \times \mathcal{O}^N$ be the joint observation space where $\mathcal{O}^i \subseteq \mathbb{R}^{o_i}$ is the observation space for agent i with o_i being the dimension of the observation-space for agent i. We define $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ to be the state transition function, which defines the probability distribution over the next joint state given the current joint state and action, where $\mathcal{P}(\mathcal{S})$ is the set of all probability measures on $\mathcal{S}$. Let $Z^i : \mathcal{S} \to \mathcal{P}(\mathcal{O}^i)$ be the observation function where $\mathcal{P}(\mathcal{O}^i)$ is the set of all probability measures on $\mathcal{O}^i$. The observation function Z^i defines the probability of agent i receiving an observation o^i from the joint state $s = (s^1, \ldots, s^N)$.

We model the multi-agent coordination problem as a decentralized Partially Observed Markov Decision Process (dec-POMDP) denoted via $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P, Z, N \rangle$. At each time t, the system is in state $s_t \in \mathcal{S}$, and each agent i receives a local observation $o_t^i \in \mathcal{O}^i$ and chooses an action $a_t^i \in \mathcal{A}^i$ according to a decentralized policy $\pi^i(a_t^i | o_t^i)$. The joint action $a_t = (a_t^1, \ldots, a_t^N)$ of all the agents yields the next joint state $s_{t+1} \sim P(s_t, a_t)$.

A. Multi-Agent Imitation

We aim to achieve decentralized multi-modal multi-agent coordination through multi-agent imitation learning. We assume that we have access to a dataset $\mathcal{D}$ containing M expert demonstrations where each demonstration is a collection of N agents interacting with each other for a finite horizon of time T. Each demonstration in the dataset is a set of tuples $\{(\xi^i, o^i)\}_{i=1}^N$, where o^i is the observation of agent i and $\xi^i = \{a_0^i, \ldots, a_{T-1}^i\}$ is the corresponding finite-horizon (T) trajectory of actions executed by agent i associated with observation o^i. Note that the expert demonstrations may be multi-modal in nature, i.e., for a given observation at time t, o_t^i, we may have two or more different expert actions. We assume that the agents exhibit coordination through various modes in the expert demonstrations of interactions.

We parameterize each agent i's policy via parameters θ^i. Let $\theta = \{\theta^i\}_{i=1}^N$ be the set of joint policy parameters of all agents. Our goal is to learn a set of decentralized policies $\{\pi_{\theta^1}^1, \ldots, \pi_{\theta^N}^N\}$ that can collectively reproduce individual agents' behaviors and learn implicit coordination among the agents from the dataset $\mathcal{D}$. To achieve this, we operate under the joint training with decentralized execution paradigm. During the joint training, the agents share a common loss function, allowing our learning process to learn coordination among agents. During decentralized execution, we want each policy to operate independently, where each agent i only relies on its local observations at a time t, o_t^i, to select its action a_t^i.

We model each decentralized policy as a conditional diffusion model. The following section briefly reviews the continuous-time diffusion framework that forms the foundation of our approach.

B. Conditional Diffusion Models

In this section, we present the formulation of each agent i's policy as diffusion models. Given the observation o^i for each agent i, we assume that the expert action trajectories are distributed according to an unknown distribution

$\xi^i \sim p_{\text{data}}(\xi^i; o^i)$ with standard deviation σ^i_{data}. Conditional Diffusion models [28] are generative models that aim to generate samples from this unknown distribution, and achieve this by considering a family of modified distributions $p(\xi^i; \sigma^i, o^i)$ obtained by iteratively adding i.i.d. Gaussian noise of standard deviation σ^i to the data. For $\sigma^i_{\text{max}} >> \sigma^i_{\text{data}}$, $p(\xi^i; \sigma^i_{\text{max}}, o^i)$ is practically indistinguishable from pure Gaussian noise. In diffusion models, we learn a denoising process that parametrizes the conditional probability distribution $p_{\theta^i}(\xi^i; o^i)$ of trajectories via a learned denoiser $D_{\theta^i}(\xi^i; \sigma^i, o^i)$ parameterized by θ^i. The idea is to randomly sample a noisy trajectory $\xi^i_0 \sim \mathcal{N}(0, {\sigma^i_{\text{max}}}^2 \mathbf{I})$ and sequentially denoise it into trajectories ξ^i_k with noise levels $\sigma^i_0 = \sigma^i_{\text{max}} > \sigma^i_1 > \cdots > \sigma^i_K = 0$ such that ξ^i_K is distributed according to the original data distribution p_{data}.

The denoiser function, $D_{\theta^i}(\xi^i; \sigma^i, o^i)$, is usually a neural network trained to predict a clean trajectory ξ^i from a noisy version:

$$\xi^i_{\text{noisy}} = \xi^i + \epsilon, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, {\sigma^i}^2 \mathbf{I})$. The model is trained by optimizing a simple denoising objective. We sample a noise level σ^i for agent i according to $p_{\text{train}}(\sigma^i)$ defined by $\ln(\sigma^i) \sim \mathcal{N}(P_{\text{mean}}, P^2_{\text{std}})$, as done in [28], and noise $\epsilon \sim \mathcal{N}(0, {\sigma^i}^2 \mathbf{I})$ and minimize the expected error:

$$\mathcal{L}^i_{\text{diff}}(\theta^i) = \mathbb{E}_{\substack{\xi^i \sim p_{\text{data}},\ \sigma^i \sim p_{\text{train}}(\sigma^i) \\ \epsilon \sim \mathcal{N}(0, {\sigma^i}^2 \mathbf{I})}} \left[\left\| D_{\theta^i}(\xi^i + \epsilon; \sigma^i, o^i) - \xi^i \right\|_2^2 \right]. \tag{2}$$

To generate a new trajectory, we solve a probability flow ODE that transforms a sample from a simple noise distribution into a sample from the data distribution. The process starts with a sample drawn from pure Gaussian noise, $\xi^i_0 \sim \mathcal{N}(0, {\sigma^i_{\text{max}}}^2 \mathbf{I})$, and integrates backwards from σ^i_{max} to 0. We use Algorithm 1 from [28] to solve this ODE numerically and obtain the denoised trajectory ξ^i_K.

While early diffusion models commonly used U-Net or Convolutional Neural Net backbones [12], [29], [30], [31], recent work has instead leveraged a transformer backbone (DiT) [32]. The implementation of a transformer results in better scaling, improved long-horizon task performance, and flexibility in conditioning for a diffusion model. We will follow the same architecture for our diffusion policies and in the following section, detail our MIMIC-D method.

IV. MIMIC-D

In our approach, we model decentralized policies π^i as a conditional diffusion model with denoiser network $D_{\theta^i}(\xi^i; \sigma^i, o^i)$. For each agent, D_{θ^i} takes three inputs, the noisy action trajectory ξ^i_k, current noise level σ^i_k, and observation o^i, and with iterative denoising produces a sampled action trajectory ξ^i_K. The observation o^i helps give context to the policy at planning time and encodes information about the ego agent, other agents in the system, and the relevant context it can observe. In the following section, we describe the training aspects of our MIMIC-D method. An overview of the MIMIC-D framework is provided in Fig. 2.

A. Joint Training

We jointly train the policies for all the agents in the system. During the training, the agents' denoising policies share a single loss and their local observations contain information about the other agents as well. This is necessary to promote coordination and collision-avoidant behaviors by encouraging agents to learn the influence of their actions over others.

The training dataset $\mathcal{D}$ consists of M joint expert demonstrations, each of horizon T. We train a set of distinct policies $\pi_\theta = \{\pi^1_{\theta^1}, \ldots, \pi^N_{\theta^N}\}$ such that each agent has an individual denoising policy. Therefore, each agent i has its own unique parameters θ^i and denoising loss from (2). In the joint training process, we define our total shared loss to be

$$\mathcal{L}_{\text{total}}(\theta) = \sum_{i=1}^{N} \mathcal{L}^i_{\text{diff}}(\theta^i) \tag{3}$$

as the sum of the losses for each agent.

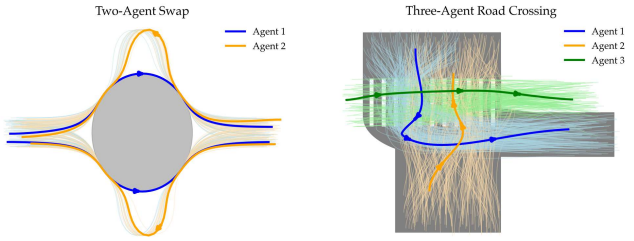
We present the training loop in Algorithm 1. At each gradient step, we loop over agents, sum the per-agent diffusion losses, then update θ by a single AdamW step using the gradient of our total loss with respect to all parameters θ. This architecture trains a specialized policy for each agent, which allows for heterogeneity among different agents in the same system. Additionally, since we train the policy parameters using the joint loss function, it encourages agents to coordinate. The shared optimization step adjusts parameters based on the entire system's performance, which encourages collaborative actions to achieve shared goals.

Algorithm 1 MIMIC-D Training Procedure

Input: Expert dataset $\mathcal{D}$
Initialize: Policy parameters $\theta = \{\theta^1, \ldots, \theta^N\}$ and a joint optimizer `Optim`
for number of training steps **do**
  $\mathcal{L}_{\text{total}} \leftarrow 0$
  for $i = 1, \ldots, N$ **do** ▷ Loop through each agent
    Sample batch $\xi^i, o^i \sim \mathcal{D}$,
    Sample noise levels $\sigma^i \sim p_{\text{train}}(\sigma^i)$
    Create noisy batch ξ^i_{noisy} using (1)
    Predict denoised batch $\hat{\xi}^i \leftarrow D_{\theta^i}(\xi^i_{\text{noisy}}; \sigma^i, o^i)$
    Compute loss $\mathcal{L}^i_{\text{diff}}(\theta^i)$ according to (2)
    $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{total}} + \mathcal{L}^i_{\text{diff}}(\theta^i)$
  end for
  Update all parameters $\theta \leftarrow \texttt{Optim}\big(\theta, \nabla_\theta \mathcal{L}_{\text{total}}\big)$
end for

B. Decentralized Online Execution

Classical diffusion policy makes use of a single centralized policy that is aware of all agents' histories and future plans at execution time. Instead, our trained policies can be executed in a decentralized fashion with each agent having access to only local observations. Additionally, we implement our approach in a receding time horizon manner to allow for replanning that considers the current positions of the other agents in the environment. This gives our method the ability



(a) Two-Agent Swap (b) Three-Agent Road Crossing

Fig. 3: Visualizations of the Two-Agent Swap and Three-Agent Road Crossing environments, along with examples of the multi-modal expert demonstrations used to train the various models. The Swap environment included 6 different solution modes, and the Road Crossing environment did not have explicit modes but rather more subtle collision avoidance behaviors.

to have agents adjust to changes in the environment and to adapt to one another's actions. At every timestep, each agent i individually acquires their local observation o^i from the environment. Then, the agents sample their own policy $\pi^i_{\theta^i}$ for ξ^i, which has horizon T. Only the first h ($h < T$) steps of each agent's actions are executed. This observing-sampling-executing process is then repeated.

V. Simulated Experiments

In this section, we detail our simulation setups, baselines, and evaluation metrics used to validate MIMIC-D's ability to coordinate multiple agents in a decentralized manner while capturing the multi-modality of demonstrations.

A. Environments

1) Two-Agent Swap: We consider an environment, which we call "Swap," that involves two agents who are trying to swap positions with one another while avoiding a central obstacle on a unitless X-Y coordinate plane. This environment involves six possible modes of achieving the swapping of positions, as shown in [33]: two trivial interaction modes where both agents pass on opposite sides of the obstacle, and four nontrivial modes where both agents go on the same side of the obstacle and one yields to the other. A visualization of the environment and expert demonstration trajectories is shown in Fig. 3a. Here, coordination is crucial to ensure that agents avoid collisions with one another, and the difficulty is elevated because of how symmetric the setup is, thus making it more difficult for agents to see a "best" mode of operation. The training dataset was evenly distributed across the six solution modes to ensure balanced coverage of the multi-modal behavior.

2) Three-Agent Road Crossing: We consider another environment called "Road Crossing" that involves three agents that are trying to avoid one another while on their way to their respective goal locations on a unitless X-Y coordinate plane. This environment demonstrates a need for agents to not only select a different mode of execution, as was the case in the "Swap" environment, but also to wait for other agents to pass before continuing to avoid collision. The state and action dimensions for each agent are both two (an x-y coordinate pair), and the agents are assumed to be holonomic point robots with single-integrator dynamics. We include this to test temporal coordination. A visualization of the environment and expert demonstration trajectories is shown in Fig. 3b. Unlike the Swap environment, this setting does not have explicitly defined discrete modes; demonstrations were collected from varied collision-avoidance interactions without enforcing a predefined distribution over behaviors.

3) Two Arm Lift: Our environment of two manipulators, based on Robosuite [34], is shown in Fig. 4a. The two Kinova3 arms collaborate to lift the pot and transfer it to the other side while avoiding the obstacle (red box). We collect expert demonstrations in two modes: passing the obstacle on the left and the right. This is a difficult task to coordinate because the two arms need to maintain their grip on the pot without dropping or pulling on it while also deciding which way to go around the obstacle. Additionally, because the two arms are not explicitly communicating and are planning independently, making this decision while adapting to the actions of the other can be very challenging. For this task, expert demonstrations were evenly collected across the two modes (passing the obstacle on the left and right) to ensure balanced training data.

B. Baselines

We compare our performance against multiple established imitation learning baselines.

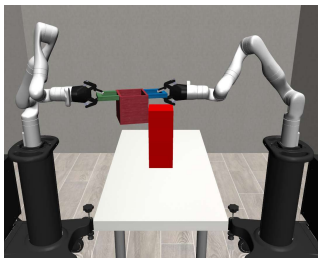
1) MLP BC: We frame this baseline as a joint training, decentralized execution BC. We train a separate BC policy for each agent from expert demonstrations using a shared loss function, but instead of a diffusion model, each policy is parameterized by a simple feedforward neural network. The learned policies are then executed in a decentralized manner.

2) MA-GAIL: MA-GAIL [23] is a GAN-based MAIL method that can be adapted to the joint training, decentralized execution paradigm. To do so, we train a single discriminator and individual generators for each agent with a shared loss.

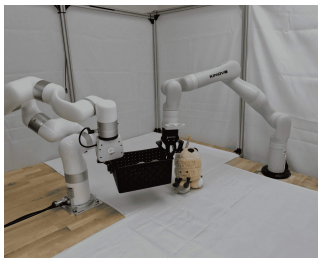
3) Vanilla Diffusion: Typically in MAIL literature, it has been shown that matching occupancy measures of each agent individually is sufficient (see Proposition 2, [23]). Therefore, we compare our approach to a simplified Diffusion variation, which we call Vanilla Diffusion that simply imitates each agent in the environment without accounting for other agents. This means that each agent's observation only encompasses their own state and the environment around them, but all agents' models share a loss according to (3) during training time. Note, we do not directly compare against LatentToM and MADiff because they offer a different level of decentralization that depends on an attention mechanism that predicts states and a consensus embedding, while MIMIC-D operates on strictly local observations that achieve implicit coordination.

C. Comparison Metrics

We choose various comparison metrics, such as the number of collisions among agents and the success rates, to compare our method against the baselines. Furthermore,



(a) Two-Arm Lift Simulation



(b) Two-Arm Lift Hardware

Fig. 4: **Two-arm lift environment visualizations.** We provide an example of what the two-arm lift task looks like for the Robosuite simulation version and the hardware demonstration. The simulation task uses two Kinova3 arms, while the hardware task uses one Kinova3 and one XArm7 arm.

since the goal of imitation learning is to accurately capture expert behavior, in our navigation scenarios, we compare the distribution of generated trajectories with that of expert trajectories. For this comparison, we employ the Wasserstein distance [35], also known as the Earth Mover's Distance (EMD), a widely used metric for measuring distances between probability distributions. Computing the EMD requires defining a distance between individual elements of the distributions. Since our elements are trajectories, we use the Fréchet distance [36] as the underlying metric to quantify the distance between two trajectories.

D. Results

1) Two-Agent Swap: In table I, we present the number of agent-agent collisions (labeled "Agent"), the number of agent-obstacle collisions (labeled "Obstacle"), and the total number of collisions (either agent-agent or agent-obstacle) for 100 sampled trajectories with randomized initial positions. Here, we set the observations o^i of each agent to be their own position and the position of the other agent. In the expert demonstrations, agents maintained a separation of 3 units and an obstacle radius of 4 units. For evaluation, we use an inter-agent distance of 2.7 units and an obstacle radius of 3.9 units, respectively, to qualify a collision. This avoids counting borderline collisions caused by discretization or noise while also increasing the ability to distinguish between methods. MIMIC-D encounters the least number of total collisions by a significant margin. It is worth noting that MLP BC encountered so few agent-agent collisions because the trajectories demonstrated mode averaging and cut through the central obstacle in nearly every instance. Vanilla Diffusion and our method both had very few agent-obstacle collisions due to diffusion's ability to effectively capture the multi-modality, but Vanilla Diffusion lacks coordination among agents, which makes it challenging for it to avoid agent-agent collisions.

As shown by the EMD in table II computed from 100 sampled trajectories, MIMIC-D and Vanilla Diffusion achieve the smallest values, best replicating the expert demonstration distribution. This is expected because we know that diffusion can effectively capture multi-modality without suffering from mode collapse or averaging. While both methods achieve similar EMD scores due to the shared diffusion foundation, our advantage lies in enabling inter-agent coordination, a failure point of the Vanilla Diffusion method.

TABLE I: Two-Agent Swap (# collisions / 100)

Method	Agent	Obstacle	Total
MLP BC	18	98	98
MAGAIL	97	99	99
Vanilla Diffusion	52	**1**	52
MIMIC-D	**12**	3	**15**

It is worth noting here that MLP BC and MAGAIL are deterministic, meaning that they would always produce a single path from a fixed initial state, unlike our method's stochastic nature, which allows it to recover the full distribution of expert demonstrations by sampling from random noise.

TABLE II: EMD between Expert Demonstrations and Sampled Trajectories for Two-Agent Swap

Method	Agent-1	Agent-2
MLP BC	1.93	1.72
MAGAIL	4.67	7.99
Vanilla Diffusion	**1.20**	1.74
MIMIC-D	1.50	**1.24**

2) Three-Agent Road Crossing: In table III, we report agent-agent collisions for a varying collision threshold. We deem there to be a collision between two agents if the minimum distance between them at any timestep is less than the threshold. Here, we set the observations o^i of each agent to be their own position and the positions of the other two agents. Expert demonstrations were collected with a clearance of 0.75 units (the agents in the demonstrations were always at least 0.75 units apart), so we sweep the collision threshold from 0.75 to more relaxed values (0.675, 0.5625, 0.375). This range serves to distinguish true collisions from near misses near the demo threshold and also reveals how well methods learn this collision avoidance behavior. While all approaches incur many threshold hits near the conservative margin of 0.75, our approach exhibits the steepest decline as the collision threshold decreases. This indicates that our method was able to learn implicit coordination.

Table IV reports the EMD between the expert demonstrations and the 100 sampled trajectories for the various methods. From here, it is clear that MIMIC-D is able to achieve the smallest value, meaning that we best replicate the expert demonstration distribution. Once again, we note that MLP BC and MAGAIL are deterministic, and for a static initial position, ended up generating the same trajectory over 100 samples, while Vanilla Diffusion and our method recovered the expert trajectory distribution.

Remark. It may be observed that there is a potential for local minima, where robots stagnate when trying to decide on a mode similar to the Freezing Robot Problem [37]. We see in experiments with our method that robots avoid this, which we believe is because diffusion is inherently stochastic. Due to the sampling process starting from noise, trajectories have inherent randomness that assists in avoiding stalling.

TABLE III: Three-Agent Road Crossing (# collisions / 100)

Method	Collision Threshold			
	0.74	0.675	0.5625	0.375
MLP BC	99	94	81	55
MAGAIL	**95**	92	68	32
Vanilla Diffusion	96	93	85	61
MIMIC-D	**95**	**14**	**1**	**0**

TABLE IV: EMD between Expert Demonstrations and Sampled Trajectories for Three-Agent Road Crossing

Method	Agent 1	Agent 2	Agent 3
MLP BC	0.4497	1.1541	0.3615
MAGAIL	1.2749	1.5166	0.5402
Vanilla Diffusion	0.4241	0.5046	0.3301
MIMIC-D	**0.3259**	**0.4394**	**0.2845**

3) Two Arm Lift: We used 50 expert demonstrations per mode to train our model. The action trajectory length is 25, and we only execute the first 10 steps of the planned trajectory. We use the end-effector pose controller provided in Robosuite to track the planned trajectory. Here, we set the observations o_t^i of each agent i at time t to be the end-effector pose of the ego agent, the end-effector pose of the other agent, and the pot's initial handle positions. Note that the poses are expressed in the respective local base frames, and it is up to the policy to interpret the other agent's pose.

After the training, we randomly spawn the pot into the environment and test our method and baselines on the lift task. For the lift task, two important sub-tasks require robot coordination to achieve the task successfully:

- The robots need to reach the pot handle and simultaneously lift the pot. If one robot tries to move fast without waiting for the other, it may fail to lift the pot, which would result in the entire task failing.
- After they successfully lift the pot, the robots need to move in coordination and choose the same mode in order to successfully transport the pot.

Therefore, we report the success rates of MIMIC-D and the baselines on these two subtasks for 20 different runs in the table V. As shown in table V, our method almost always picks up the basket even when placed randomly, and ours is the only method that is able to successfully transport the basket. MAGAIL performs the worst due to the inherent instability of GAN training, which prevents it from learning reliable policies for this complex, high-dimensional task, especially when coordination is required. MLP BC achieves partial success in lifting the pot, but the requirement of coordinated behavior between agents makes it difficult for MLP BC to complete the full task. Vanilla Diffusion benefits from the strengths of diffusion models and learns higher-quality policies than MLP BC, but still fails to capture inter-agent coordination in the presence of multi-modality. In contrast, MIMIC-D effectively leverages diffusion and enables our method to consistently lift and transport the pot with the highest success rate.

TABLE V: Two-Arm Lift simulation success rates (20 runs)

Method	Number of successful lifts	Number of overall success
MLP BC	2	0
MAGAIL	0	0
Vanilla Diffusion	5	0
MIMIC-D	**18**	**15**

VI. Hardware Experiments

To demonstrate MIMIC-D in high-dimensional real hardware, we also perform the Two-Arm Lift task in the real world. The experiment setup is shown in Fig. 4b. We use one xArm7 arm and one Kinova3 arm to lift and transfer the basket while avoiding an obstacle. This setup demonstrates the ability of our approach to handle heterogeneity among the agents. We simplify the setting by fixing the pose of the pot and removing it from the observation vector in Section V-A.3 since we already demonstrated MIMIC-D's ability to complete this task. Here, we collect two modes of expert demonstrations (left and right) with only eight demonstrations per mode. The xArm7 tracks the planned pose trajectory with an end-effector pose controller, while the Kinova3 tracks the trajectory with an end-effector velocity controller according to [38].

During execution, the two arms need to agree on which mode to execute. Despite initial attempts to move in opposite directions, after replanning, they reach a consensus. In Fig 1, a full demonstration of both modes is shown with the decision point occurring at t_2. This coordination is very difficult and demonstrates that the agents can understand both modes and adapt online only through implicit coordination. In our experiments, our method achieved **95% success rate** (19 successes out of 20 trials), demonstrating our method's reliability even when trained with only 16 total demonstrations. Although the arms show some oscillatory behavior in choosing a mode, successful attempts ultimately converge and place the basket on the opposite side of the obstacle.

We observe a higher success rate in hardware compared to simulation, primarily because in simulation, the pot is modeled as a rigid body, meaning even small oscillations could cause drops, while the hardware setup uses a slightly deformable basket that is more tolerant to oscillations. It is important to note that our method achieves this high success rate by reliably completing the task, not by exploiting the deformability. As we mentioned in Section V-D.3, the failure points in simulation for baseline methods are much more severe and occur before oscillations affect the task success. Therefore, these methods would only perform as well as they did in the simulation.

VII. Conclusion

We introduced a novel multi-agent imitation learning approach called MIMIC-D for decentralized multi-agent coordination in multi-modal tasks. By formulating the problem as a dec-POMDP and modeling agent policies as a conditional Diffusion Transformer model, we were able to recover diverse multi-model behaviors and avoid failure points like

mode collapse. Our paradigm overcomes the need for agents to use a centralized planner to coordinate among agents using only locally available information. We demonstrate the effectiveness of this approach in multiple simulated domains and on a complicated bimanual hardware setup. In all these setups, we show significant improvements over baselines in recovering expert trajectory distributions while reducing collisions and task failures. Despite these improvements, our approach exhibits limitations in highly symmetric environments, where decentralized agents may independently select incompatible modes, leading to collisions or stagnation during mutual avoidance. In the future, we aim to remove the assumption of perfect state observation by conditioning on camera inputs. Also, we will look into human-robot interactions by operating one of the robots manually and allowing others to plan autonomously. We will also explore ways to improve out-of-distribution performance, a common shortcoming of IL methods.

REFERENCES

[1] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[2] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[3] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[4] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

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

[6] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1534–1539.

[7] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu, "Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization," in *5th Annual Conference on Robot Learning*.

[8] L. Peters, D. Fridovich-Keil, C. J. Tomlin, and Z. N. Sunberg, "Inference-based strategy alignment for general-sum differential games," *arXiv preprint arXiv:2002.04354*, 2020.

[9] T. Kavuncu, A. Yaraneri, and N. Mehr, "Potential ilqr: A potential-minimizing controller for planning multi-agent interactive trajectories," *arXiv preprint arXiv:2107.04926*, 2021.

[10] N. Mehr, M. Wang, M. Bhatt, and M. Schwager, "Maximum-entropy multi-agent dynamic games: Forward and inverse solutions," *IEEE transactions on robotics*, vol. 39, no. 3, pp. 1801–1815, 2023.

[11] M. Bhatt, Y. Jia, and N. Mehr, "Strategic decision-making in multi-agent domains: A weighted constrained potential dynamic game approach," *IEEE Transactions on Robotics*, 2025.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[13] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9902–9915.

[14] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[15] C. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, D. Anguelov *et al.*, "Motiondiffuser: Controllable multi-agent motion prediction using diffusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 9644–9653.

[16] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[17] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.

[18] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[19] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[20] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in neural information processing systems*, vol. 35, pp. 24 611–24 624, 2022.

[21] K. Nagpal, D. Dong, and N. Mehr, "Leveraging large language models for effective and explainable multi-agent credit assignment," in *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, 2025, pp. 1501–1510.

[22] S. Choi, K. Ryu, J. Ock, and N. Mehr, "Craft: Coaching reinforcement learning autonomously using foundation models for multi-robot coordination tasks," *arXiv preprint arXiv:2509.14380*, 2025.

[23] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 31, 2018.

[24] T. V. Bui, T. Mai, and H. T. Nguyen, "Misodice: Multi-agent imitation from unlabeled mixed-quality demonstrations," *arXiv preprint arXiv:2505.18595*, 2025.

[25] M. Niedoba, J. Lavington, Y. Liu, V. Lioutas, J. Sefas, X. Liang, D. Green, S. Dabiri, B. Zwartsenberg, A. Scibior *et al.*, "A diffusion-model of joint interactive navigation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 55 995–56 011, 2023.

[26] Z. Zhu, M. Liu, L. Mao, B. Kang, M. Xu, Y. Yu, S. Ermon, and W. Zhang, "Madiff: Offline multi-agent learning with diffusion models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 4177–4206, 2024.

[27] C. He, G. S. Camps, X. Liu, M. Schwager, and G. Sartoretti, "Latent theory of mind: A decentralized diffusion architecture for cooperative manipulation," *arXiv preprint arXiv:2505.09144*, 2025.

[28] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *Advances in neural information processing systems*, vol. 35, pp. 26 565–26 577, 2022.

[29] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.

[30] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[31] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[32] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4195–4205.

[33] M. Bhatt, I. Askari, Y. Yu, U. Topcu, H. Fang, and N. Mehr, "Multinash-pf: A particle filtering approach for computing multiple local generalized nash equilibria in trajectory games," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 12 442–12 449.

[34] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, Y. Zhu, and K. Lin, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[35] C. Villani, "The wasserstein distances," in *Optimal transport: old and new*. Springer, 2009, pp. 93–111.

[36] T. Eiter and H. Mannila, "Computing discrete fréchet distance," 1994.

[37] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 797–803.

[38] J. Seo, N. P. S. Prakash, A. Rose, J. Choi, and R. Horowitz, "Geometric impedance control on se (3) for robotic manipulators," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 276–283, 2023.