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

Scalable Multi Agent Diffusion Policies for Coverage Control

Frederic Vatnsdal, Romina Garcia Camargo, Saurav Agarwal, Alejandro Ribeiro

***Abstract*—We propose MADP, a novel diffusion-model-based approach for collaboration in decentralized robot swarms. MADP leverages diffusion models to generate samples from complex and high-dimensional action distributions that capture the interdependencies between agents' actions. Each robot conditions policy sampling on a fused representation of its own observations and perceptual embeddings received from peers. To evaluate this approach, we task a team of holonomic robots piloted by MADP to address coverage control—a canonical multi agent navigation problem. The policy is trained via imitation learning from a clairvoyant expert on the coverage control problem, with the diffusion process parameterized by a spatial transformer architecture to enable decentralized inference. We evaluate the system under varying numbers, locations, and variances of importance density functions, capturing the robustness demands of real-world coverage tasks. Experiments demonstrate that our model inherits valuable properties from diffusion models, generalizing across agent densities and environments, and consistently outperforming state-of-the-art baselines.**

I. Introduction

In large-scale environments, the use of multi robot teams becomes essential. Prior work has proposed strategies for decentralized coordination [1], [3], [4], yet existing policies often fail to scale or adapt effectively as team size increases. A central difficulty lies in achieving diversity: rather than designing a single navigation strategy, agents must be able to adapt their behavior to the specific demands of the current situation.

Generative Diffusion Models (GDMs) have recently emerged as a powerful framework for high-dimensional tasks [5]. By capturing complex, multi-modal distributions with stable training dynamics, they are a natural candidate for multi agent control in continuous environments. Recent studies have applied GDMs to trajectory generation in both single and multi agent settings [6]–[15], demonstrating strong expressivity over short- and long-horizon behaviors, robustness to out-of-distribution scenarios, and promising scalability. However, these works have not yet exploited the full scalability potential of GDMs, with some evaluating for large robot swarms but generating trajectories in a centralized manner.

We propose a collaborative control architecture in which each robot runs a diffusion-policy controller parameterized by a spatial transformer (Sections II and III). Following the framework of a learned Perception–Action–Communication (LPAC) loop [1], an agent encodes its local sensor data into a compact feature vector, broadcasts this vector to nearby teammates, fuses incoming messages, and conditions the diffusion process on the combined representation to sample its next control command. The policy is trained centrally by imitating a clairvoyant expert with full state access, but it can be executed in a fully decentralized manner. We evaluate our approach on the coverage control problem [2] (Section IV), where a swarm seeks to minimize a coverage cost defined over an importance density function (IDF). As in [1], [16], we assume restricted sensing ranges and limited communication radii, which make the problem more challenging while also reflecting realistic operating conditions.

Our model leverages the stochastic and multi-modal properties of diffusion models to learn policies that adapt to diverse coverage demands (Section V and illustrated in Figure 1). In-distribution evaluations show that our Multi Agent Diffusion Policy (MADP) consistently outperforms state-of-the-art baselines. Beyond training conditions, MADP demonstrates strong generalization: it adapts to variations in the number, size, and location of importance features, and maintains performance when the number of robots or features is scaled beyond the training configuration. These results highlight both the adaptability and scalability of diffusion-based policies for decentralized multi robot control.

A. Related work

In robotics, GDMs have been applied to single-agent tasks such as manipulation and motion planning [6], [7], where actions are generated by iteratively denoising from Gaussian noise. The denoising process is typically conditioned on observations, obstacle maps, or task goals to guide trajectory generation [8], [9]. Recent work has explored diffusion models for multi agent motion generation. MotionDiffuser [10] and MADiff [11] use diffusion models with self- and cross-attention for motion prediction and trajectory generation, conditioning on scene context or inter-agent interactions. Both approaches are intrinsically centralized and do not address or analyze scalability. The authors in [12] propose a data-efficient framework that incorporates constraints to produce collision-free trajectories, achieving scalability to tens of robots. However, the method remains centralized and emphasizes coordination rather than deeper collaboration between agents. SwarmDiff [14] generates swarm trajectories centrally, which are then refined at the robot level with Model Predictive Control (MPC). Although it employs a diffusion transformer, it does not model agents or spatial structure directly. Liang et al. [15] integrate constrained optimization into diffusion sampling, but the combination of nonconvexity

FV, RGC, and AR are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA. SA is with the Department of Mechanical Engineering, Indian Institute of Technology Bombay (IITB), India.

Emails: {vatnsdal,rominag,sauravag,aribeiro}@seas.upenn.edu.

This work was supported by ARL DCIST CRA W911NF-17-2-0181.

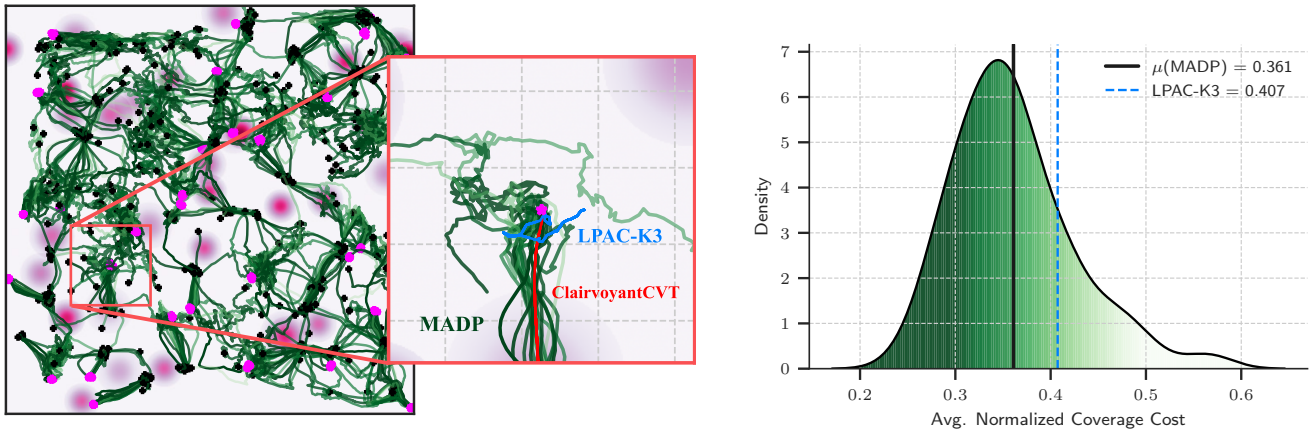

Fig. 1. Demonstration of the Multi Agent Diffusion Policy (MADP) for multi robot collaboration in the coverage control task. The inherent stochasticity of MADP allows the generation of a set of finite-horizon trajectories for each robot from their respective positions. **Left:** The coverage control environment with features of interests modeled as purple bivariate Gaussians with randomly sampled means, variances, and positions. For each robot, 20 trajectories (shown in green) are generated corresponding to each run of the MADP for 150 time steps. **Middle:** The enhanced view captures the trajectories generated by a single robot starting at the magenta dot. MADP generates trajectories shown in green (darker shades correspond to a better coverage cost), while the trajectories of LPAC-K3 (blue) [1] and Clairvoyant CVT [2] (red) are shown for comparison. **Right:** MADP outperforms the state-of-the-art baseline for coverage control on average and occasionally finds solutions with a similar cost to the clairvoyant expert. This is especially true when the importance density function (IDF) has much smaller regions of interest than what the models are trained on.

and high dimensionality limits scalability. Their follow-up work [13] introduces discrete guided diffusion, which decomposes the global planning task into local subproblems, each solved with a diffusion model, improving scalability.

II. Diffusion Models for Decentralized Multi Robot Policies

We apply our GDM to a system of N robots with observations $\mathbf{O}(t) = [\mathbf{o}_1(t), \cdots, \mathbf{o}_N(t)] \in \mathbb{R}^{d_o \times N}$ and actions $\mathbf{U}(t) = [\mathbf{u}_1(t), \cdots, \mathbf{u}_N(t)] \in \mathbb{R}^{d_u \times N}$. The GDM comprises two processes: forward corruption and backward inference. The forward process is the iterative corruption of expert actions $\mathbf{U}_0$ over k steps with standard white Gaussian noise until convergence to pure noise at $\mathbf{U}_K \sim \mathcal{N}(0, \mathbf{I})$. This process is described by the Markov chain of conditional Gaussian distributions $q(\mathbf{U}_k|\mathbf{U}_{k-1})$ and is governed by a sequence of decreasing coefficients $\{\alpha_k\}_{k=1}^K$ called the noise schedule,

$$q(\mathbf{U}_k|\mathbf{U}_{k-1}) = \mathcal{N}\left(\sqrt{\alpha_k}\mathbf{U}_{k-1}, (1-\alpha_k)\mathbf{I}\right). \tag{1}$$

The chain in (1) can be expressed as the square-root-weighted combination of $\mathbf{U}_{k-1}$ and white Gaussian noise $\epsilon_k \sim \mathcal{N}(0, \mathbf{I})$,

$$\mathbf{U}_k(\mathbf{U}_{k-1}, \epsilon_k) = \sqrt{\alpha_k}\mathbf{U}_{k-1} + \sqrt{1-\alpha_k}\epsilon_k. \tag{2}$$

Equations (1) and (2) require iterating along the chain to obtain every $\mathbf{U}_k$, which makes them cumbersome. In practice, we define $\bar{\alpha}_k := \prod_{j=1}^k \alpha_j$ [5] to obtain $\mathbf{U}_k$ directly from $\mathbf{U}_0$ and rewrite (1) as $q(\mathbf{U}_k|\mathbf{U}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_k}\mathbf{U}_0, (1-\bar{\alpha}_k)\mathbf{I})$ and (2) as $\mathbf{U}_k(\mathbf{U}_0, \epsilon_k) = \sqrt{\bar{\alpha}_k}\mathbf{U}_0 + \sqrt{1-\bar{\alpha}_k}\epsilon_k$.

GDM inference takes the form of the reverse process in which a fixed prior $\mathbf{U}_K \sim \mathcal{N}(0, \mathbf{I})$ is propelled back down the chain,

$$p(\mathbf{U}_{k-1} \mid \mathbf{U}_k, \mathbf{O}; \mathbf{H}) \sim \mathcal{N}\left(\mu(\cdot\,; \mathbf{H}), \Sigma(\cdot\,; \mathbf{H})\right), \tag{3}$$

conditioned on observations $\mathbf{O}$ and parameterized by $\mathbf{H}$, until $\mathbf{U}_0$ is reconstructed. For expedited sampling with fewer than K backward steps, we use the denoising diffusion implicit model (DDIM) reverse process [17] which can be made deterministic when $\eta = 0$, $(\eta \in [0, 1])$ (see (4)). Inference invokes the learned noise predictor $\hat{\varepsilon}(\mathbf{U}_k, \mathbf{O}; \mathbf{H})$ which we use to define the mean $\mu(\cdot\,; \mathbf{H})$ and we select a fixed covariance $\Sigma(\cdot\,; \mathbf{H}) = \sigma_k^2\mathbf{I}$, as in [17],

$$\mu(\mathbf{U}_k, \mathbf{O}; \mathbf{H}) = \sqrt{\frac{\bar{\alpha}_{k-1}}{\bar{\alpha}_k}}\Big(\mathbf{U}_k - \frac{1-\bar{\alpha}_k/\bar{\alpha}_{k-1}}{\sqrt{1-\bar{\alpha}_k}}\hat{\varepsilon}\left(\mathbf{U}_k, \mathbf{O}; \mathbf{H}\right)\Big). \tag{4}$$

To draw a sample $\mathbf{U}_{k-1} \sim p\left(\mathbf{U}_{k-1} \mid \mathbf{U}_k, \mathbf{O}; \mathbf{H}\right)$, DDIM provides us with a deterministic mapping that preserves the forward marginals $q(\mathbf{U}_k \mid \mathbf{U}_0)$,

$$\mathbf{U}_{k-1} = \rho_1\mathbf{U}_k + \rho_2\hat{\varepsilon}(\mathbf{U}_k, \mathbf{O}; \mathbf{H}) + \sigma_k\epsilon_k, \tag{5}$$

where $\rho_1 = \sqrt{\frac{\bar{\alpha}_{k-1}}{\bar{\alpha}_k}}$ and $\rho_2 = \sqrt{1-\bar{\alpha}_{k-1}-\sigma_k^2} - \sqrt{\frac{\bar{\alpha}_{k-1}(1-\bar{\alpha}_k)}{\bar{\alpha}_k}}$ and $\sigma_k = \eta\sqrt{\frac{(1-\bar{\alpha}_{k-1})}{(1-\bar{\alpha}_k)}}\sqrt{1-\frac{\bar{\alpha}_k}{\bar{\alpha}_{k-1}}}$.

In the context of imitation learning, we are furnished with a dataset of expert actions and observations $\mathcal{D} := \{\mathbf{U}_0^{(m)}, \mathbf{O}^{(m)}\}_{m=1}^M$ where M is the total number of training examples. The inputs to the denoising function are obtained by sampling expert actions and observations from the dataset, $\{\mathbf{U}_0, \mathbf{O}\}_m \sim \mathcal{D}$, the noise schedule step from a uniform distribution $k \sim \mathcal{U}(1, K)$ and white Gaussian noise, $\epsilon_k \sim \mathcal{N}(0, \mathbf{I})$. We introduce the forward posterior to demonstrate how we can simplify learning this posterior with $p(\cdot\,; \mathbf{H})$,

$$q(\mathbf{U}_{k-1} \mid \mathbf{U}_k, \mathbf{U}_0) = \mathcal{N}(\tilde{\mu}(\mathbf{U}_k, \mathbf{U}_0), \sigma_k^2\mathbf{I}) \tag{6}$$

where $\tilde{\mu}(\mathbf{U}_k, \mathbf{U}_0)$ is the forward posterior mean. We measure the similarity of distributions q and p with the Kullback-Leibler (KL) divergence. Because q and p are both Gaussians

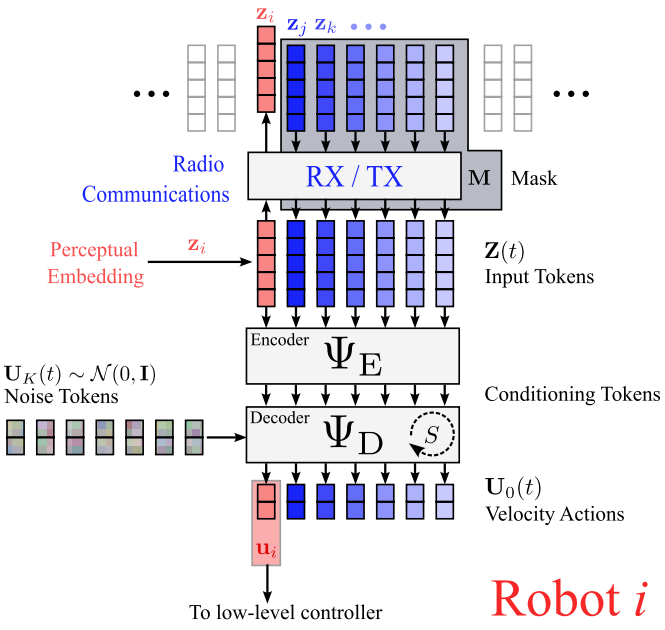

Fig. 2. Decentralized sampling of outputs from MADP: Each robot in the system runs MADP *locally* and shares perceptual token $\mathbf{z}_i$ with other robots in communication range. A token buffer stores the received embeddings from other robots, which are then processed by the encoder Ψ_{E} once per sample generation. The denoiser Ψ_{D} drives $\mathbf{U}_K(t)$ to $\mathbf{U}_0(t)$ through $S = 50$ denoising steps (DDIM [17]). Finally, the output velocity action $\mathbf{u}_i(t)$ is extracted and sent to the low-level controller.

with fixed isotropic covariances, the KL divergence is proportional to the squared difference of means under optimization,

$$\mathbb{E}_q \left[D_{\mathrm{KL}} \left(q \parallel p \right) \right] \propto \mathbb{E}_q \left[\frac{1}{2\sigma_k^2} \| \tilde{\mu}_k - \mu(\cdot\,; \mathbf{H}) \|_2^2 \right]. \tag{7}$$

Expressing the means in terms of the noise ϵ that produced $\mathbf{U}_k$ enables us to simplify to the standard DDPM loss [5],

$$\mathcal{L}_{\mathrm{GDM}}(\mathbf{H}) = \mathbb{E}_{\mathbf{U}_0, \mathbf{O}, k, \epsilon} \| \epsilon - \hat{\varepsilon}(\mathbf{U}_k, \mathbf{O}; \mathbf{H}) \|_2^2. \tag{8}$$

In this paper, we train a GDM parameterized by spatial transformers (Section III) to solve (8) for a coverage control problem (Section IV).

III. The MADP Architecture

We propose a Multi Agent Diffusion Policy (MADP) architecture for the purpose of generating diverse solutions to decentralized and occluded coverage control problems. Our architecture is constructed as a diffusion model and parameterized by a pair of spatial transformers (ST), a variant of the general transformer architecture in which we impart a rotary encoding scheme [18] onto the query and key projections. Spatial transformers have the key characteristic that they can be trained in a centralized environment and deployed in a decentralized communication and computation setting. The architecture (Figure 2) contains three main components: the Perception Module, the Encoder and the Decoder. It takes the local maps observed in the environment as input and produces velocity actions for each robot in the system. We generate environments of size $1024\,\mathrm{m} \times 1024\,\mathrm{m}$ for training and evaluation of MADP policies for coverage control.

A. Perception

Each robot uncovers the occluded portion of the environment using its sensing capabilities defined by a $64\,\mathrm{m} \times 64\,\mathrm{m}$ square centered on the robot. Robots never explicitly share raw information about the environment with each other, i.e., each robot has only its own local view of the environment determined by what it has sensed so far. We define the *local maps* that each robot processes as the square $(256\,\mathrm{m} \times 256\,\mathrm{m})$ centered on the robot ($1\,\mathrm{m}$ per pixel). Therefore, each robot has access to its historic sensor readings, wider than the sensor view but limited to the local map dimensions.

Conditioning starts with the observation $\mathbf{o}_i \in \mathbb{R}^{4\times32\times32}$ made by each robot. This observation is a set of four local maps downsized from (256×256) to (32×32) using bilinear interpolation. The local maps are: the density function, boundaries and obstacles, neighbors' x, and neighbors' y. A three-layer convolutional neural network (CNN) produces an embedding $\mathbf{z}_i \in \mathbb{R}^{32}$ for each observation of each robot. We collect these embeddings and concatenate the position $\mathbf{x}_i$ to produce the input to the encoder $\Psi_{\mathrm{E}}(\mathbf{Z}_0; \mathbf{H}_{\mathrm{E}})$, where $\mathbf{Z}_0 = \left[[\mathbf{z}_1, \mathbf{x}_1]^\top, \cdots, [\mathbf{z}_N, \mathbf{x}_N]^\top \right] \in \mathbb{R}^{34\times N}$ and $\mathbf{H}_{\mathrm{E}}$ are the parameters of the encoder.

B. Spatial Transformer

Both the encoder and decoder are realized as spatial transformers [19]. A spatial transformer is a set processing architecture composed of attention layers and a relative positional encoding scheme. The positional encoding must be relative to preserve permutation and shift equivariance, which is necessary because it is impossible to guarantee a fixed permutation during policy inference. Specifically, we incorporate Rotary Positional Embedding (RoPE) [18].

The dot-product attention mechanism [20] is constructed by projecting the input $\mathbf{Z} \in \mathbb{R}^{d_z \times N}$ into the query space: $\mathbf{QZ}$, into the key space: $\mathbf{KZ}$ and into the value space: $\mathbf{VZ}$ for $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{d_A \times d_z}$, where we have chosen the same dimensions for all spaces. We compute the attention matrix $\mathbf{A} \in \mathbb{R}^{d_A \times d_A}$ which captures the compatibility between queries and keys,

$$\mathbf{A} = \mathbf{sm}\langle \mathbf{QZ}, \mathbf{KZ} \rangle. \tag{9}$$

Dot-product attention is finalized by computing the weighted sum of the values, producing the output $\mathbf{Y} \in \mathbb{R}^{d_A \times N}$,

$$\mathbf{Y} = \mathbf{AVZ}. \tag{10}$$

We combine (9) and (10) to express the contextual representation, $\mathbf{Y} = \mathbf{sm}\langle \mathbf{QZ}, \mathbf{KZ} \rangle \mathbf{VZ}$.

In practice, transformers use multiple layers L and heads H, i.e., channels. Then, we can rewrite the self-attention in (9) as a multi-headed layer where the output of the previous layer is the input to the next,

$$\mathbf{Y}_\ell^h = \mathbf{sm}\langle \mathbf{Q}_\ell^h \mathbf{Z}_{\ell-1}, \mathbf{K}_\ell^h \mathbf{Z}_{\ell-1} \rangle \mathbf{V}_\ell^h \mathbf{Z}_{\ell-1}. \tag{11}$$

The outputs of (11) are subsequently concatenated across heads and passed through the multi-layer perceptron $\mathbf{W}_\ell$ with a skip connection,

$$\mathbf{Z}_\ell = \sigma\big(\left[\mathbf{Y}_\ell^1, \ldots, \mathbf{Y}_\ell^H \right] \mathbf{W}_\ell + \mathbf{Z}_{\ell-1} \big). \tag{12}$$

The learnable parameters at each layer and head are $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, $\mathbf{W}$, where we omit the layer and head notation. We implement the non-linearity σ as a Leaky ReLU.

To encode positions, we implement RoPE with a sinusoidal basis established by creating a vector of angular frequencies $\mathbf{w} = [\omega_1, \cdots, \omega_{D/4}]$ where D is the embedding dimension of the key, query and value spaces. Pairing adjacent real elements yields $D/2$ complex channels; splitting again between x and y channels yields $D/4$ frequencies per axis. Frequencies in $\mathbf{w}$ are produced by the geometric series $\omega_i = 2\pi\tau^{-4i/D}$ where τ is the encoding period which we select to be proportional to the environment size. Let $\mathbf{X} \in \mathbb{R}^{2\times N}$ contain the 2D positions of N agents. Then, indexing agents with n, we can construct the complex matrix $\tilde{\mathbf{X}} \in \mathbb{C}^{D/2\times N}$ to house the sinusoids as follows,

$$[\tilde{\mathbf{X}}]_{i,n} = \begin{cases} \exp(j\mathbf{w}_i[\mathbf{X}]_{x,n}) & \text{if } i \leq D/4 \\ \exp(j\mathbf{w}_{i-D/4}[\mathbf{X}]_{y,n}) & \text{if } D/4 < i \leq D/2 \end{cases} \tag{13}$$

where $j = \sqrt{-1}$. We introduce the rotary positional encoding to each head in only the first layer of the transformer,

$$\mathbf{A}_1^h = \mathbf{sm}\left(\mathrm{Re}\left((\tilde{\mathbf{X}} \odot (\tilde{\mathbf{Q}}_1^h \mathbf{Z}_0))^\dagger (\tilde{\mathbf{X}} \odot (\tilde{\mathbf{K}}_1^h \mathbf{Z}_0))\right)\right), \tag{14}$$

where $\tilde{\mathbf{Q}}_1^h, \tilde{\mathbf{K}}_1^h \in \mathbb{C}^{D/2\times D}$ are complex renderings of the real-valued query and key matrices, created by pairing adjacent row elements. They are multiplied with the rotary encoding element-wise ($\odot$) and the real part is extracted. This has the powerful effect of directly embedding relative positions encoded by our Fourier basis into the attention matrix $\mathbf{A}$. Because the conjugate transpose ($\dagger$) is invoked by the inner product, the complex exponentials defined in (13) produce terms of the kind $\exp(j\mathbf{w}_i([\mathbf{X}]_{x,n} - [\mathbf{X}]_{x,m}))$ $\forall n, m \in [1, N]$, and therefore the positional contribution to a_{nm} depends on the relative difference rather than the absolute positions.

A vital characteristic of the transformer is that it can operate on any number N of received embeddings from other robots. Therefore, we can train the transformer in a centralized setting and deploy it locally on each robot for decentralized operation. We have found that adding an attention mask during training aids stability when the density or scale of the scenario changes. The attention mask is the element-wise *AND* between a windowing mask and a graph component mask $\mathbf{M}_{att} = \mathbf{M}_W \wedge \mathbf{M}_C$. Defining the attention radius in meters as r_{att}, we have,

$$[\mathbf{M}_W]_{ij} = \begin{cases} 1 & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\|_2 < r_{att} \\ 0 & \text{otherwise} \end{cases}$$
$$[\mathbf{M}_C]_{ij} = \begin{cases} 1 & \text{if } \mathcal{C}(i) = \mathcal{C}(j) \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $\mathcal{C}(i) \in \mathbb{N}$ is a mapping from a node to the ID of the graph component the node is currently on.

C. Encoder and Decoder

The encoder Ψ_E and decoder Ψ_D have similar spatial transformer parameterizations underpinning them. The primary difference is that the decoder contains cross-attention layers to incorporate the conditioning from the encoder with the input $\mathbf{U}_k$ undergoing denoising. Specifically, each decoder layer is comprised of self-attention followed by cross-attention. Both components have $L_{ST} = 8$ layers, $H = 8$ heads of dimension $d_h = 32$, a geometrically spaced set of encoding frequencies with period $\tau = 1024\,\mathrm{m}$ and an attention radius of $256\,\mathrm{m}$. In the transformers, we use a pre-norm scheme where the layer norm is applied before attention. During training, we use $T = 1000$ steps of noise over a linearly spaced, decreasing range of $\alpha \in [0.9999, \ldots, 0.98]$ and $\alpha_k < \alpha_{k-1}$. Algorithm 1 details how we can sample from MADP using DDIM.

Algorithm 1: MADP Sampling (DDIM)

Input : model $\hat{\varepsilon}(\cdot\,; \mathbf{H})$, observations $\mathbf{O}$, DDPM steps K, DDIM steps S, schedule $\{\alpha_k\}_{k=1}^K$, noise scale $\eta \in [0, 1]$

Output: $\mathbf{U}_0$

subset $\mathcal{K} = (k_S{=}K > \cdots > k_1 > k_0{=}0)$,
$\bar{\alpha}_0 \leftarrow 1, \quad \bar{\alpha}_k \leftarrow \prod_{j=1}^k \alpha_j, \quad \forall k = \{1, \ldots, K\}$

$\mathbf{U}_{k_S} \sim \mathcal{N}(0, \mathbf{I})$

Pre-compute constants and conditioning

$\sigma_i \leftarrow \eta\sqrt{\frac{1-\bar{\alpha}_{k_{i-1}}}{1-\bar{\alpha}_{k_i}}}\sqrt{1 - \frac{\bar{\alpha}_{k_i}}{\bar{\alpha}_{k_{i-1}}}}, \quad i = 1, \ldots, S$

$c_{0i} \leftarrow \sqrt{\bar{\alpha}_{k_{i-1}}/\bar{\alpha}_{k_i}}, \quad i = 1, \ldots, S$

$c_{1i} \leftarrow \sqrt{1 - \bar{\alpha}_{k_i}}, \quad i = 1, \ldots, S$

$c_{2i} \leftarrow \sqrt{1 - \bar{\alpha}_{k_{i-1}} - \sigma_i^2}, \quad i = 1, \ldots, S$

$\mathbf{M}_{att} \leftarrow$ compute_mask($\mathbf{X}$)

$\mathbf{C} \leftarrow \Psi_\mathrm{E}(\mathrm{CNN}(\mathbf{O}), \mathbf{X}, \mathbf{M}_{att})$

for $i = S$ **down to** 1 **do**

  $\hat{\epsilon} \leftarrow \Psi_\mathrm{D}(\mathbf{U}_{k_i}, \mathbf{X}, \mathbf{C}, \mathbf{M}_{att})$

  sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$

  $\mathbf{U}_{k_{i-1}} \leftarrow c_{0i}\mathbf{U}_{k_i} + (c_{2i} - c_{0i}\,c_{1i})\,\hat{\epsilon} + \sigma_i\,\epsilon$

IV. Decentralized Coverage Amidst Occlusions

Coverage control is the problem of coordinating a swarm of robots to provide effective sensor coverage to areas of variable importance. We consider the coverage problem on a planar environment, $\mathcal{X} \subset \mathbb{R}^2$. A scalar field called the *importance density function* (IDF) maps points in the environment to values of importance, $\Phi\colon \mathcal{X} \to \mathbb{R}_+$. The IDF is occluded and therefore unknown to each robot at t_0, save for the small sensing region surrounding each robot. Robots are modeled with single integrator dynamics. The robot actions are bounded in norm, such that $\|\mathbf{u}_i(t)\|_2 \leq u_{\max}$. The system's dynamics are then,

$$\mathbf{X}(t + \Delta t) = \mathbf{X}(t) + \Delta t\mathbf{U}(t). \tag{16}$$

The coverage cost is measured by accumulating the distance between the closest agent and a given point $\mathbf{v}$ in the environment, weighted by the IDF at that point,

$$\mathcal{J}(\mathbf{X}(t)) = \int_{\mathbf{v}\in\mathcal{X}} \min_{i\in\{1,\ldots,N\}} f(\|\mathbf{x}_i(t) - \mathbf{v}\|_2)\Phi(\mathbf{v})d\mathbf{v}. \tag{17}$$

Computing the integral in (17) is impractical due to the minimum that needs to be found for each point. Instead, we use Voronoi tessellation, which decomposes the environment into cells V_i, each assigned to a robot i, such that the robot i is closest to any point in its corresponding cell V_i. The positions of the robots induce the Voronoi tessellation $\mathcal{T} := \{V_i\}_{1:N}$ of $\mathcal{X}$ and the cells emerge from,

$$V_i = \{\mathbf{v} \in \mathcal{X} \mid \|\mathbf{x}_i - \mathbf{v}\|_2 \leq \|\mathbf{x}_j - \mathbf{v}\|_2, \forall\, j \neq i\}. \quad (18)$$

The objective function (17) is made computationally tractable by the tessellation $\mathcal{T}$ and we also select $f(x) = x^2$,

$$\mathcal{J}(\mathbf{X}(t)) = \sum_{i=1}^{N} \int_{\mathbf{v} \in V_i} \|\mathbf{x}_i(t) - \mathbf{v}\|_2^2 \Phi(\mathbf{v}) d\mathbf{v}. \quad (19)$$

For our experimental setup, we consider holonomic robots that have limited observability, i.e., limited knowledge about the state of the environment $\mathcal{X}$ and its discovery, and the position of other robots $\mathbf{X}(t)$. The sensor field of view (FOV) is defined by the square $r_s \times r_s$ centered around each robot. A communication radius r_c is defined to determine which other robots in the swarm are within robot i's neighborhood. The nature of our architecture, however, allows attention to be applied between agents that are more than one hop away. This introduces a new notion of neighborhood as agents within the attention mask defined in Section III-B. We denote the set of agents within (masked) attention of agent i, as $\mathcal{M}_i$. Robot i will have access to information obtained from the local environment, as well as positions $\mathbf{x}_j$ and embeddings $\mathbf{z}_j$ from neighboring robots, $j \in \mathcal{M}_i$. Our setting has ideal wireless conditions with neither delays nor errors in transmission.

A. Imitation Learning

We create a dataset $\mathcal{D} = \{\mathbf{U}_0^{(m)}, \mathbf{X}^{(m)}, \mathbf{O}^{(m)}\}$ of 100×10^3 examples by sampling observations and actions from rollouts of the clairvoyant Centroidal Voronoi Tessellation (CVT) expert (see Appendix). The dataset is generated in environments of size $1024\,\mathrm{m} \times 1024\,\mathrm{m}$ containing $N = 32$ robots and $F = 32$ Gaussian features. The coverage control simulator and data generation code are taken from the Coverage Control Library [1]. We divide this dataset into 70 000 training examples, 20 000 validation examples and 10 000 test examples. The learning hyperparameters, including the architecture parameters, were selected by performing a Bayesian sweep over the learning rate and weight decay. MADP was trained with a learning rate of $\mathrm{lr} = 8.5 \times 10^{-5}$ and a weight decay of $\mathrm{wd} = 2.1 \times 10^{-12}$. The model was trained for 1000 epochs and a batch size of 196 on an Nvidia RTX 3090 GPU; the validation loss patience was set to 500 with a minimum error of 1×10^{-4}.

V. Results

We present our simulated experimental results to quantify MADP's capabilities. Unless otherwise stated, we consider an environment $\mathcal{X}$ of size $(1024\,\mathrm{m} \times 1024\,\mathrm{m})$. Each scenario has $N = 32$ holonomic robots, each with a communication radius of $r_c = 256\,\mathrm{m}$, and sensor aperture of $(64\,\mathrm{m} \times 64\,\mathrm{m})$. A total of $F = 32$ bivariate Gaussian features $\phi_i = \mathcal{N}(\mu_i, \sigma_i^2 \mathbf{I}) \quad \forall i \in \{1, \ldots, F\}$ are placed randomly in the environment, such that $\mu_i^{(x)}, \mu_i^{(y)} \sim \mathcal{U}(0, 1024)$, with variable variance, $\sigma_i \sim \mathcal{U}(40, 60)$. We restrict our Gaussian features so that their peak density values lie in the range $[0.6, 1.0]$ and the distribution is truncated at $2\sigma_i$. For an example of an environment, see Figure 5. The primary evaluation metric is the coverage cost defined in (19), normalized by the initial coverage cost at $t = 0$. For a description of the baselines, see Appendix. For $N = 32$, MADP inference on an RTX 3090 with FP32 is $0.36 \pm 0.006\,\mathrm{s}$ per step.

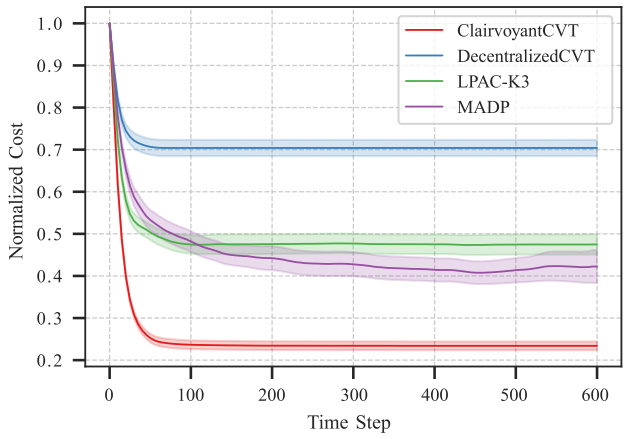

Fig. 3. Normalized coverage cost as each policy is rolled out over 50 randomly generated environments with $N = 32$ robots, $F = 32$ Gaussian features in the IDF over 600 time steps. The dark line of each trace is the mean cost, while the band shows the 95% confidence interval. Lower values indicate better performance. Both LPAC-K3 [1] and the proposed MADP perform substantially better than the baseline Decentralized CVT algorithm, with MADP performing better than LPAC-K3.

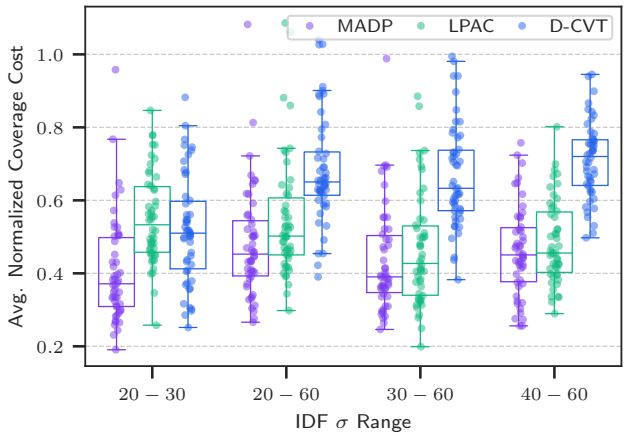

Fig. 4. Normalized cost distribution obtained from evaluating MADP, LPAC-K3, and DCVT on 50 unseen environments for varying standard deviations (σ) of importance of IDF features. For training, the σ for each feature was randomly sampled from $[40, 60]$, and the policies are tested on different ranges of σ, i.e., out-of-distribution IDFs. MADP consistently performs better than the baselines and demonstrates greater adaptability when the size of the Gaussian features is aggressively constrained.

Figure 3 shows the normalized coverage cost over 600 rollout steps, averaged across 50 unseen, randomly gener-

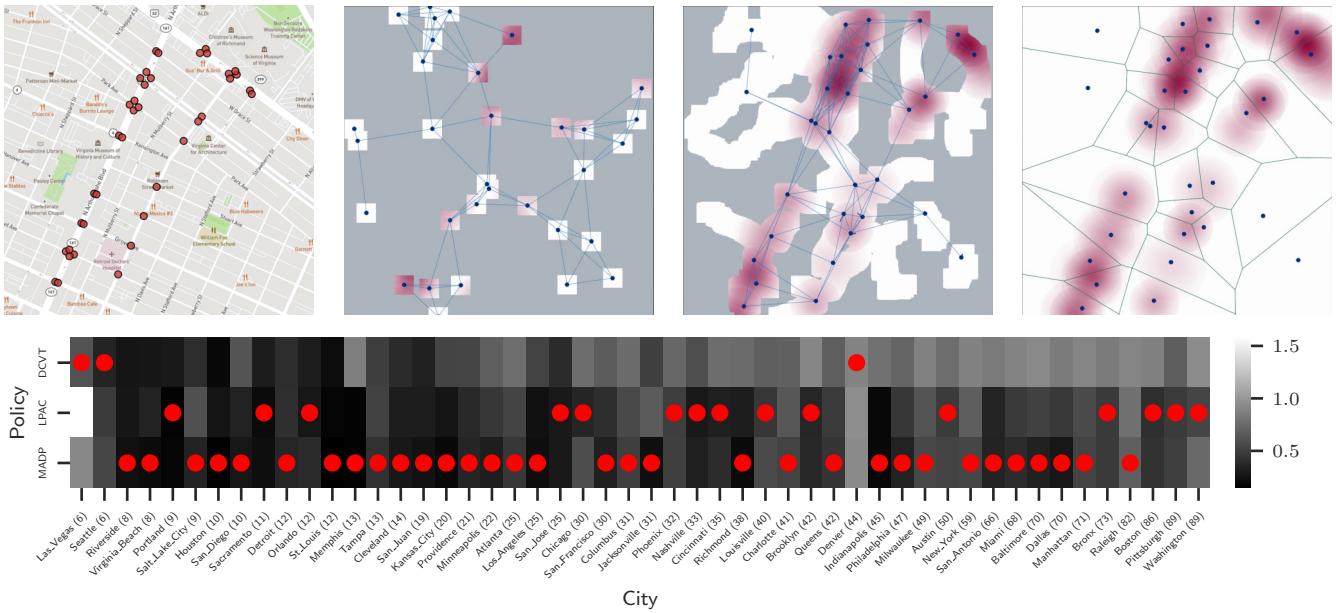

Fig. 5. **Top:** We use the locations of traffic lights from a city (left) to inform the generation of the importance density function (fully visible; right). Shown here: a rollout of MADP on the Richmond map, running the policy over 600 timesteps. From the second image on the left to right, the timesteps are 0, 300, and 600. In the last frame, we reveal the complete IDF and overlay the Voronoi tessellation (18). **Bottom:** The heatmap shows the normalized coverage cost of policy rollouts over 600 time steps for each city. Darker values indicate better performance. A red dot is placed on the cell of the best-performing policy for each city. The initial positions of the $N = 32$ robots are sampled from a uniform distribution in the environment. The parenthetical next to the city name indicates the number of Gaussian Features F. The scenario configuration is the same across controllers. Map data is from OpenStreetMap [21].

ated environments. The model was trained with $N = 32$ robots and $F = 32$ Gaussian features. MADP significantly outperforms DCVT. MADP also outperforms LPAC after about 100 steps. This experiment serves as an in-distribution evaluation of the proposed policy. Subsequent subsections examine out-of-distribution settings with varying numbers of robots, Gaussian features, and environment configurations.

A. Adaptability to Small Areas of Interest

We evaluate MADP in environments where the Gaussian features are reduced in size, controlled by adjusting the range of values for σ_i. This scenario is relevant for aerial robots, where altitude changes can alter the scale of observed Gaussian features (i.e., features appear smaller at higher elevation). Results are shown in Figure 4 for different ranges of σ. Only the range $\sigma \in [40, 60]$ is in-distribution for both MADP and LPAC; the remaining ranges contain sizes of features not seen during training. MADP achieves the lowest normalized coverage costs, demonstrating effective generalization to environments with varying distributions. Performance remains consistently strong across all tested values, with only a few outliers.

B. Real-World Scenarios

Thus far, we have evaluated the policies on synthetic importance density functions. We now test performance on IDFs informed by real-world traffic light locations. These maps are constructed by capturing $(1024\,\mathrm{m} \times 1024\,\mathrm{m})$ neighborhoods from 50 U.S. cities and placing a Gaussian at each traffic light.

TABLE I
MEAN NORMALIZED COVERAGE COST OF REAL-WORLD CONFIGURATIONS

Scenario	DCVT	LPAC-K3	MADP
50 Cities	0.63 ± 0.03	0.45 ± 0.03	$\mathbf{0.39} \pm 0.03$
Uniform	0.72 ± 0.01	0.48 ± 0.01	$\mathbf{0.46} \pm 0.01$
Square	0.71 ± 0.02	0.46 ± 0.02	$\mathbf{0.44} \pm 0.02$
Line	0.76 ± 0.02	0.20 ± 0.01	$\mathbf{0.17} \pm 0.01$

We roll out MADP, LPAC-K3, and DCVT on each city for 600 timesteps and record the mean normalized coverage cost (Figure 5). MADP achieves the lowest cost in 32 out of 50 cities, indicated by the red markers denoting the best policy in each case. Table I reports the mean cost $\pm$ standard error across all 50 cities, reaffirming MADP's superior performance in out-of-distribution conditions.

To further assess adaptability to real-world deployments, we evaluate three initialization scenarios for robot positions (Table I): (i) Uniform, (ii) Square, and (iii) Line; Figure 6. In all cases, the positions of each robot i, $(x, y)_i$, are sampled from the uniform distributions $x \sim \mathcal{U}(x_{\min}, x_{\max})$ and $y \sim \mathcal{U}(y_{\min}, y_{\max})$. The difference is in the range of values x and y can take during the initialization of the robots in the environment. In (i), robots can be initialized anywhere in the environment, $x, y \in [0, 1024]\mathrm{m}$. This is the same configuration used when generating the dataset. We implement (ii) by uniformly sampling within a smaller square $(102\,\mathrm{m} \times 102\,\mathrm{m})$ centered at coordinate $(166.25\,\mathrm{m}, 166.25\,\mathrm{m})$ in the lower left corner, simulating, e.g., the deployment

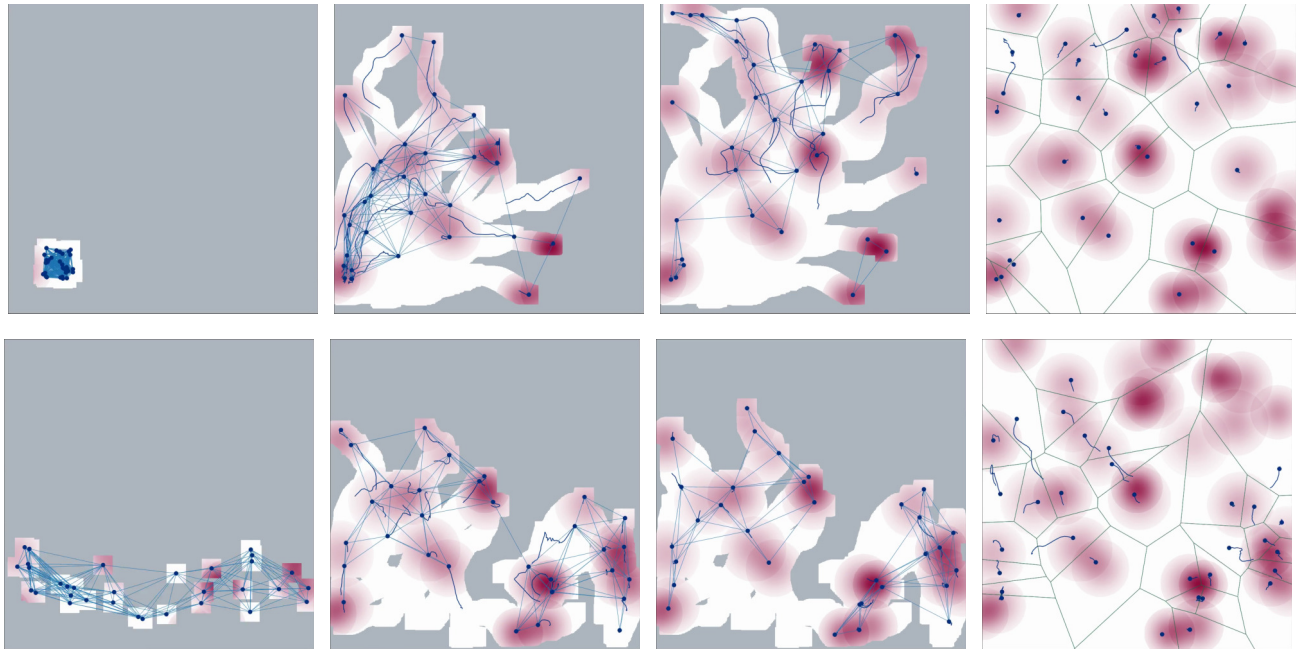

Fig. 6. We experiment with different rules for generating the initial positions of the robots (blue dots). In each case, we define a subset of the environment over which we uniformly sample the positions. We explore restricting the launch positions to a small square placed on the outskirts of the environment (**Top**) and a band located near the bottom edge (**Bottom**). The areas of the map that have not been seen by any robot are colored gray and communication between robots is represented by a blue edge between nodes. From left to right, snapshots of the experiments are at 0, 200, 400 and 600 timesteps. The complete IDF and induced Voronoi tessellation (18) are shown at step 600.

of quadrotors from a launch pad. Thus for the (ii), we have, $x, y \in [115.25, 217.25]\mathrm{m}$. For (iii), we define a tighter bound on y only, creating a band of width $256\,\mathrm{m}$ for the initial positions. In this case, we have that $x \in [0, 1024]\mathrm{m}$ and $y \in [96, 352]\mathrm{m}$. For each starting configuration, we consider 50 unseen environments and 600 timesteps. In all cases, MADP outperforms the baselines, underscoring the robustness of the policy to initial conditions and its capacity to effectively explore the environment.

C. Scalability and Transferability

As a final evaluation, we study the scalability and transferability of the learned policy, originally trained with $N = 32$ robots and $F = 32$ features. Figure 7 reports the relative improvement of MADP over baselines DCVT and LPAC-K3, measured as $(\overline{\text{Baseline}} - \overline{\text{MADP}})/\overline{\text{Baseline}}$, where $\bar{\cdot}$ denotes the average over trials.

For each setting we test the algorithms in 50 unseen environments. While performance decreases when the number of robots is reduced, MADP shows clear gains as the swarm size scales, outperforming the baseline in larger teams. Sensitivity to the number of features F is minimal, indicating strong out-of-distribution generalization. Overall, these results highlight the scalability and transferability of MADP to settings with larger robot populations.

VI. Conclusion

We propose MADP, a scalable, diffusion-based control policy for decentralized multi agent systems. The novelty of MADP is in the combination of generative diffusion models with spatial transformers to enable sampling from the distribution of navigation policies for large-scale teams. We demonstrate MADP on the coverage control task, showing that it exceeds the performance of the state-of-the-art LPAC-K3 across several variations of the problem. MADP excels when areas of interest are smaller, which makes the problem more challenging. We posit that this could be the result of the stochasticity of GDMs enhancing exploration through perturbation. The diversity of trajectories produced by MADP motivates further investigation; we plan to explore ways to exploit this property in future work. Guidance control and model predictive path integral (MPPI) control are promising approaches that leverage the use of these diverse stochastic trajectories to further improve performance.

Appendix

The coverage control problem can be reduced to finding centroidal Voronoi tessellations (CVT) of the environment. A cost function is minimized by assigning each robot to a region of the space and performing gradient descent toward a local minimum. A well-known algorithm for this problem is that proposed by Lloyd [22]. This original version assumes full access to the state of the environment. In decentralized systems, robots exchange each other's positions and local sensing information, as proposed in [2].

We consider several baselines based on finding Voronoi partitions, with different levels of observability. The relevant benchmarks are summarized as follows:

- **Clairvoyant:** Centralized algorithm with full access to the robot positions and the entire IDF.

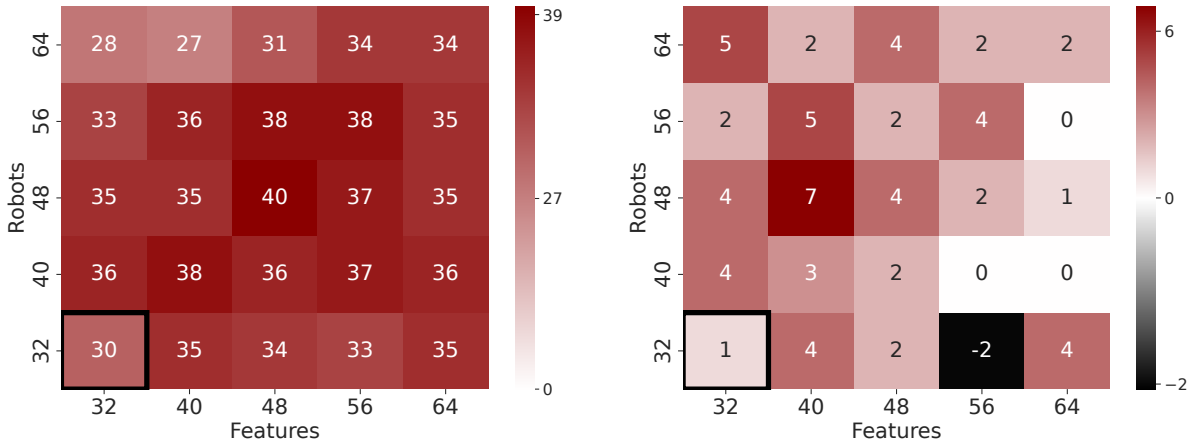

Fig. 7. Each heatmap is a comparison against a baseline: against DCVT (**Left**) and against LPAC-K3 (**Right**). Each cell in the heatmap shows the percentage difference for evaluation of MADP against the respective baseline on environments with varying numbers of robots (N) and features (F). A positive value indicates an improvement over the baseline. MADP and LPAC-K3 are both trained on $N = F = 32$.

- **Decentralized CVT (DCVT):** Decentralized algorithm with information exchange limited by the communication range. Each robot has access to the positions of its neighbors. Knowledge about the IDF is limited to local, historical sensory data.
- **LPAC-K3 [1]:** State-of-the-art, learning-based benchmark. The algorithm is decentralized with information exchange limited by the communication range, each robot having access to the positions of the neighbors, and the sensing is local.

References

[1] S. Agarwal, R. Muthukrishnan, W. Gosrich, V. Kumar, and A. Ribeiro, "LPAC: Learnable perception-action-communication loops with applications to coverage control," *IEEE Trans. Robot.*, vol. 41, pp. 5986–6005, 2025.

[2] J. Cortés, S. Martìnez, T. Karataş, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, 2004.

[3] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6252–6259.

[4] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11 785–11 792.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[6] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *Int. J. Robot. Res.*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[7] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Proc. Int. Conf. Machine Learning*. PMLR, Jun. 2022, pp. 9902–9915, iSSN: 2640-3498.

[8] Y. Luo, C. Sun, J. B. Tenenbaum, and Y. Du, "Potential based diffusion motion planning," in *Proc. Int. Conf. Machine Learning*, 2024.

[9] J. Carvalho, A. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: learning and planning of robot motions with diffusion models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023.

[10] C. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, and D. Anguelov, "MotionDiffuser: Controllable multi-agent motion prediction using diffusion," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2023, pp. 9644–9653.

[11] Z. Zhu *et al.*, "MADiff: offline multi-agent learning with diffusion models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024.

[12] Y. Shaoul, I. Mishani, S. Vats, J. Li, and M. Likhachev, "Multi-robot motion planning with diffusion models," in *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[13] J. Liang, S. Koenig, and F. Fioretto, "Discrete-guided diffusion for scalable and safe multi-robot motion planning," *The 40th Annual AAAI Conference on Artificial Intelligence*, 2026.

[14] K. Ding *et al.*, "Swarmdiff: Swarm robotic trajectory planning in cluttered environments via diffusion transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2025, pp. 4203–4212.

[15] J. Liang, J. K. Christopher, S. Koenig, and F. Fioretto, "Simultaneous multi-robot motion planning with projected diffusion models," in *Proceedings of the 42nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Singh *et al.*, Eds., vol. 267. PMLR, 13–19 Jul 2025, pp. 37 162–37 180. [Online]. Available: https://proceedings.mlr.press/v267/liang25e.html

[16] W. Gosrich *et al.*, "Coverage control in multi-robot systems via graph neural networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8787–8793.

[17] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proc. Int. Conf. Learn. Representations*, 2021.

[18] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "RoFormer: Enhanced transformer with rotary position embedding," *Neurocomput.*, vol. 568, no. C, Feb. 2024. [Online]. Available: https://doi.org/10.1016/j.neucom.2023.127063

[19] D. Owerko, F. Vatnsdal, S. Agarwal, V. Kumar, and A. Ribeiro, "MAST: Multi-agent spatial transformer for learning to collaborate," 2025. [Online]. Available: https://arxiv.org/abs/2509.17195

[20] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017.

[21] OpenStreetMap contributors, "Openstreetmap," 2025, oDbL 1.0. See https://www.openstreetmap.org/copyright. [Online]. Available: https://www.openstreetmap.org

[22] S. Lloyd, "Least squares quantization in pcm," in *Transactions on Information Theory*. IEEE, 1982, pp. 129–137.