

# Accelerated Multi-Modal Motion Planning Using Context-Conditioned Diffusion Models

Edward Sandra<sup>1,3</sup> Lander Vanroye<sup>1,3</sup> Dries Dirckx<sup>1,3</sup> Ruben Cartuyvels<sup>2</sup> Jan Swevers<sup>1,3</sup> Wilm Decré<sup>1,3</sup>

**Abstract**—Classical methods in robot motion planning, such as sampling-based and optimization-based methods, often struggle with scalability towards higher-dimensional state spaces and complex environments. Diffusion models, known for their capability to learn complex, high-dimensional and multi-modal data distributions, provide a promising alternative when applied to motion planning problems and have already shown interesting results. However, most of the current approaches train their model for a single environment, limiting their generalization to environments not seen during training. The techniques that do train a model for multiple environments rely on a specific camera to provide the model with the necessary environmental information and therefore always require that sensor. To effectively adapt to diverse scenarios without the need for retraining, this research proposes Context-Aware Motion Planning Diffusion (CAMPD). CAMPD leverages a classifier-free denoising probabilistic diffusion model, conditioned on sensor-agnostic contextual information. An attention mechanism, integrated in the well-known U-Net architecture, conditions the model on an arbitrary number of contextual parameters. CAMPD is evaluated on a 7-DoF robot manipulator and benchmarked against state-of-the-art approaches on real-world tasks, showing its ability to generalize to unseen environments and generate high-quality, multi-modal trajectories, at a fraction of the time required by existing methods.

## I. INTRODUCTION

Robot motion planning is a fundamental problem in robotics, involving the task of finding a collision-free trajectory for a robot between a start and goal configuration. Solving this problem can be challenging, particularly in high-dimensional and complex environments. High-dimensional robot motion planning problems naturally arise from robots with many degrees of freedom (DoF), as well as when multiple robots share the same environment and a motion plan must be planned for all robots simultaneously. Complex environments emerge because robots often need to operate in cluttered spaces. Moreover, motion planners are frequently required to consider constraints beyond avoiding collisions, such as dynamic constraints to ensure the feasibility of the motion on the robot. Additionally, the motion planner should be highly efficient to respond effectively to disturbances in the environment within a limited time frame during on-line deployment. Classical approaches frequently encounter limitations in addressing these challenges. They can be split up into two families: sampling-based methods and

optimization-based methods. Sampling-based methods [1]–[3], while asymptotically complete, produce non-smooth trajectories and often struggle to scale efficiently as problem dimensions increase. Optimization-based methods [4]–[8] on the other hand, can produce solutions that adhere to arbitrary constraints and optimize specific objectives, but are prone to becoming trapped in local minimizers and are sensitive to the initializations.

Recently, learning-based techniques have emerged as a promising alternative to classical motion planning algorithms in robotics, with methods broadly categorized into learning from demonstration on the one hand and reinforcement learning (RL) [9] on the other hand. Learning from demonstration trains neural networks on expert trajectories, generated either by humans or classical algorithms, and can replace components or entire motion planners. Standalone approaches [10]–[12] generate collision-free trajectories directly, while hybrid methods integrate deep learning with classical planners to enhance performance, such as guiding sampling processes [13] or providing initial guesses for optimization-based planners [14]–[16]. In contrast, RL uses trial-and-error interactions with the environment to optimize a reward function, eliminating the need for expert datasets. Policies learned via RL can function as standalone planners or complement existing methods [17]. Learning-based methods hold promise for addressing high-dimensional planning problems, but challenges remain, including limited generalization to unseen environments during training, substantial data requirements for learning from demonstrations, and the complexities of reward shaping and inefficient learning in reinforcement learning.

Several of these promising, emerging methods use diffusion models [12], [15], [16], [18]–[21]. This generative model family has shown success in a variety of applications, including image generation, video generation and speech generation [22]. In robotics, diffusion models have been applied to both imitation [23]–[25] and policy learning [26]–[30]. However, important research challenges remain [21], particularly in enhancing the generalization and robustness of these deep learning-based methods. Existing diffusion models are typically not environment-aware, which limits their adaptability to new environments [12], [26]. While environment information can be incorporated via gradient guidance [15], [31], this approach increases computational cost. Techniques that train models to be environment-aware rely on a camera to provide the necessary environmental information, requiring an additional processing step and increasing the algorithm’s complexity [16].

<sup>1</sup>Department of Mechanical Engineering, KU Leuven, 3001 Heverlee, Belgium

<sup>2</sup>Department of Computer Science, KU Leuven, 3001 Heverlee, Belgium

<sup>3</sup>Flanders Make@KU Leuven, Belgium

Correspondence to: Edward Sandra [edward.sandra@kuleuven.be](mailto:edward.sandra@kuleuven.be)

The implementation is publicly available at <https://meco-group.github.io/campd>.

This work addresses the generalization challenge by introducing Context-Aware Motion Planning Diffusion (CAMPD), a classifier-free diffusion probabilistic model (DPM) for robot motion planning, capable of rapidly generating multi-modal solutions and adapting to unseen environments and tasks. CAMPD is a planning-as-inference approach [32], where collision-free, near-optimal plans are generated by sampling from the model’s learned conditional distribution. An attention mechanism [33] is integrated within the U-Net [34] architecture to condition the model on encoded environmental parameters. After training with occasional context dropout, classifier-free guidance is employed during inference to steer the sampled trajectories towards collision-free regions. The proposed method is evaluated in simulation on a 7-DoF Franka Emika Panda robot in sphere-based and real-world environments, both with varying number of obstacles. Results are compared against a classical planner and state-of-the-art learning-based methods.

The key contributions of this work are as follows:

- 1) **Context-aware diffusion-based motion planner:** A diffusion-based motion planner is introduced that integrates sensor-agnostic contextual information, such as obstacle locations, directly into the model.
- 2) **Improved generalization:** Extensive experiments provide evidence that CAMPD achieves significantly better generalization to novel environments compared to state-of-the-art planners.
- 3) **Real-time, executable trajectories:** CAMPD generates high-quality, dynamically feasible trajectories in real time, enabling rapid planning and replanning in dynamic settings without the need for post-processing.

The proposed approach is evaluated on a diverse set of challenging motion planning problems, demonstrating strong generalization to previously unseen environments and effectively capturing the multi-modal nature of locally optimal solutions. CAMPD directly outputs robot-executable trajectories using an efficient post-processing step. Its computational efficiency makes it particularly well-suited for real-time, online applications, enabling faster planning and replanning in dynamic environments at a fraction of the computational cost required by existing methods.

## II. RELATED WORK

Leveraging denoising diffusion probabilistic models [18], [19] has shown promise in generating multi-modal plans for high-dimensional problems. Janner et al. [12] introduced a method in which sampling from the model and planning become nearly identical. However, the model needs to be trained per environment, limiting its adaptability to different environments. MPD [15] and EDMP [31] integrate the environment into their methods by exploiting cost functions to guide the diffusion process, thereby generating collision-free trajectories. While their method allows for generalization to unseen environments during inference, the learned network lacks direct knowledge of the environments and only relies on these cost functions to achieve collision avoidance. Furthermore, hand-crafted cost functions require tuning and can

lead to increased computation time. SceneDiffuser [35] and DiffusionSeeder [16] condition their model on environmental information. SceneDiffuser utilizes point clouds as environmental input, while DiffusionSeeder employs depth images encoded by a Vision Transformer [36]. Moreover, DiffusionSeeder leverages the trajectories sampled from the diffusion model as priors for cuRobo [7], an optimization-based planner, effectively blending learning-based and optimization-based approaches while maintaining low computation times. While these methods yield promising results, they rely on environmental representations such as point clouds or camera data, which may introduce unnecessary complexity when a simpler description of the environment (e.g. precise object dimensions and locations) is available. Furthermore, this abstraction of the environment representation allows flexibility in how environmental information is obtained. Decision Diffuser [37] similarly conditions on contextual information, but it does not support an arbitrary number of contextual elements, such as a variable number of obstacles, during inference. In contrast to these methods, CAMPD leverages a classifier-free DPM, conditioned on sensor-agnostic contextual information, supporting an arbitrary number of contextual elements.

## III. METHODOLOGY

This work studies the robot motion planning problem, where the goal is to rapidly generate a fast, smooth, collision-free trajectory to bring a robot manipulator from an initial configuration to a goal configuration.

### A. Terminology

In this work, a trajectory  $\tau \triangleq (\mathbf{q}_0, \dots, \mathbf{q}_{H-1}) \in \mathbb{R}^{d_q \times H}$  is defined as a sequence of robot configurations in discrete-time with horizon  $H$  and configuration space dimension  $d_q$ . The context  $\mathcal{C}$  of a trajectory  $\tau$  is defined as a structured collection of relevant environmental information and other parameters that provide useful insights for the task. The context  $\mathcal{C} = \{\mathbf{c}_{k,l}\}$  can be categorized into  $K$  distinct types (e.g. obstacle types), where each type  $k = 0, \dots, K-1$  can have multiple instances, each represented as a vector  $\mathbf{c}_{k,l}$  with specific parameters (e.g. context type 1 are  $L$  spheres with parameters  $\mathbf{c}_{1,l} = [x_l, y_l, z_l, r_l]$  for  $l = 0, \dots, L-1$ ). The start and goal configurations  $\mathbf{q}_0$  and  $\mathbf{q}_{H-1}$  are not considered part of the context.

### B. Neural Network Architecture

As in MPD [15], CAMPD employs planning-as-inference via a diffusion probabilistic model (DPM) [18], [19]. A DPM consists of two processes: a forward and reverse diffusion process. The noise  $\epsilon$  added in the forward process is learned using the mean squared error (MSE) loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{t, \epsilon, \tau_0, \mathcal{C}} [\|\epsilon - \epsilon_\theta(\tau_t, \mathcal{C}, t)\|^2], \quad (1)$$

with  $\theta$  the network parameters and  $\epsilon_\theta$  the learned estimate of  $\epsilon$ . The estimated mean  $\mu_\theta$  of the reverse process is then given by [19]

$$\mu_\theta(\tau_t, \mathcal{C}, t) = \frac{1}{\sqrt{\alpha_t}} \left( \tau_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \epsilon_\theta(\tau_t, \mathcal{C}, t) \right). \quad (2)$$

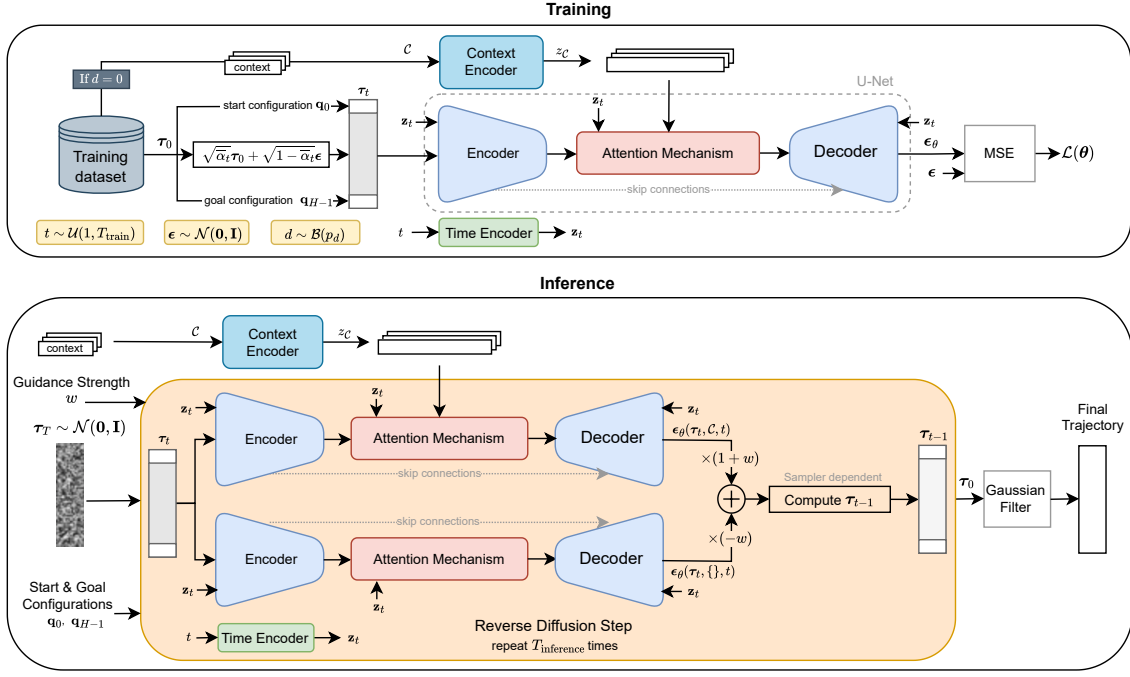


Fig. 1: Overview of CAMPD: During training, trajectories are perturbed with Gaussian noise at a random diffusion step, while start and goal states remain fixed for conditioning. Context is encoded and randomly dropped (Bernoulli  $p_d$ ) for conditional/unconditional training. A U-Net with attention predicts the noise using MSE loss. During inference, noise is iteratively denoised via reverse diffusion with classifier-free guidance ( $w$ ), keeping start and goal fixed. Finally, a Gaussian filter is applied to the resulting trajectory to reduce jerk.

In CAMPD, the diffusion model estimates the added noise  $\epsilon$  in both conditional and unconditional modes, such that classifier-free guidance [38] can be employed:

$$\epsilon_{\theta}'(\tau_t, \mathcal{C}, t) = (1 + w)\epsilon_{\theta}(\tau_t, \mathcal{C}, t) - w\epsilon_{\theta}(\tau_t, \{\}, t) \quad (3)$$

with  $w$  the guidance strength. The network architecture consists of a time encoder, context encoder and a U-Net, as illustrated in Figure 1.

*a) Time Encoder:* The time encoder applies sinusoidal positional encoding [33] followed by a single-hidden-layer multi-layer perceptron (MLP), yielding a latent representation  $\mathbf{z}_t = \text{MLP}_t(\text{PE}(t)) \in \mathbb{R}^{d_z}$  to condition the U-Net on the diffusion step.

*b) Context Encoder:* The context encoder maps the context information  $\mathcal{C}$  to a set of latent representations, denoted as  $\mathbf{z}_{\mathcal{C}} = \{\mathbf{z}_{\mathbf{c}_{k,l}} \mid \mathbf{z}_{\mathbf{c}_{k,l}} \in \mathbb{R}^{d_z}\}$ . Each instance  $\mathbf{c}_{k,l}$  of context type  $k$  is processed through a single-hidden-layer MLP $_k$ , yielding the latent representation  $\mathbf{z}_{\mathbf{c}_{k,l}} = \text{MLP}_k(\mathbf{c}_{k,l})$ . A schematic overview of the context encoder is provided in Figure 2. These latent representations are subsequently utilized in the attention mechanism of the U-Net.

*c) U-Net:* The U-Net, denoted as  $\epsilon_{\theta, \text{U}}(\tau_t, \mathbf{z}_{\mathcal{C}}, \mathbf{z}_t)$ , estimates the added noise  $\epsilon$ , given the noisy trajectory  $\tau_t$  and the latent time and context representations  $\mathbf{z}_t$  and  $\mathbf{z}_{\mathbf{c}_{k,l}}$ . The convolutional encoder and decoder networks are similar to the one used by [12] and are conditioned on the latent time step representation  $\mathbf{z}_t$ . Between the encoder and decoder, an

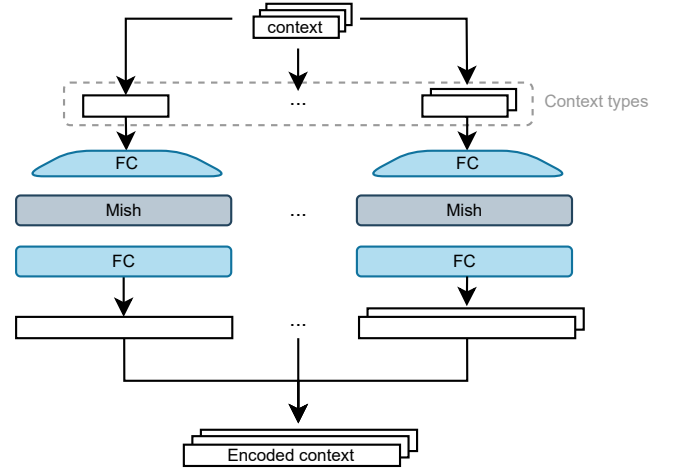


Fig. 2: Context Encoder. Each instance  $\mathbf{c}_{k,l}$  of context type  $k$  is processed by a dedicated multi-layer perceptron (MLP)

attention mechanism is integrated to condition the model on the context information. This allows the model to incorporate an arbitrary number of context instances  $\mathbf{c}_{k,l}$  (e.g. environments with varying numbers of obstacles), rather than conditioning the model on a fixed context encoding. The attention mechanism is detailed in Figure 3. The attention mechanism is inspired by the Transformer architecture [33]. It consists of a multi-head self-attention mechanism and a multi-head attention mechanism with the latent representations  $\mathbf{z}_t$

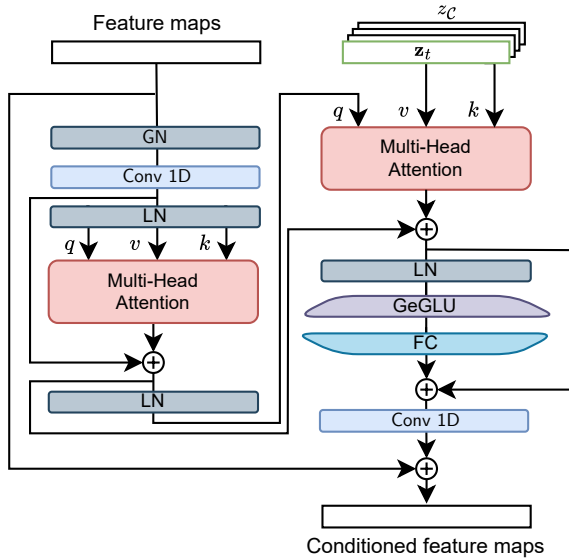


Fig. 3: Attention Mechanism. The encoded context vectors  $\mathbf{z}_{C_{k,l}}$  and the latent time step representation  $\mathbf{z}_t$  are fused into the U-Net through a residual architecture of self- and cross-attention layers.

$\mathbf{z}_{C_{k,l}}$  as both keys and values, followed by a feed-forward network.

### C. Training

Given a noisy trajectory  $\tau_t$ , a time step  $t$  and context  $\mathcal{C}$ , the model learns to predict the added noise  $\epsilon$  via the mean squared error (MSE) loss given in Equation (1). To enable classifier-free guidance during inference, the model is trained in both conditional and unconditional modes by setting the context information to an empty set following a Bernoulli distribution  $\mathcal{B}(p_d)$  with probability  $p_d$ . To make the model conditioned on the start and goal configurations, these configurations are not perturbed during training. The training algorithm is detailed in Algorithm 1 and illustrated in Figure 1.

### D. Inference

Trajectories are generated by iteratively denoising white Gaussian noise through a reverse diffusion process. The form of this process depends on the chosen sampler: it can be a discrete-time stochastic process (e.g., DDPM [19]), a deterministic non-Markovian process (e.g., DDIM [39]), or the numerical solution of a differential equation using high-order solvers (e.g., DPM-Solver++ [40]). To strengthen the effect of contextual information during inference, classifier-free guidance [38] is applied (Equation (3)). To ensure the trajectory begins and ends at the desired configurations, the first and last states of the noisy sequence  $\tau_t$  are fixed to the start and goal states, respectively. Finally, a Gaussian filter is applied to the resulting trajectory to reduce jerk. The complete inference procedure is described in Algorithm 2 and illustrated in Figure 1.

### Algorithm 1: Training

---

**Input:** Training set  $\mathcal{D}$ , Diffusion model  $\epsilon_\theta$ , learning rate  $\alpha$ , noise schedule parameters  $\{\bar{\alpha}_t\}$ , probability of unconditional training  $p_d$

- 1 **while** not converged **do**
  - // Sample batch of trajectories and related contexts
  - 2  $\{(\tau_0^i, \mathcal{C}^i)\}_{i=1}^N \sim \mathcal{D}$ ;
  - // Randomly discard contexts to train conditionally or unconditionally
  - 3  $b^i \sim \mathcal{B}(p_d), \forall i$ ;
  - 4  $\mathcal{C}^i \leftarrow \{\}$  **if**  $b^i = 1, \forall i$ ;
  - 5  $t^i \sim \mathcal{U}(1, T_{\text{train}}), \epsilon^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \forall i$ ;
  - // Evaluate noised trajectories
  - 6  $\tau_t^i \leftarrow \sqrt{\bar{\alpha}_t} \tau_0^i + \sqrt{1 - \bar{\alpha}_t} \epsilon^i, \forall i$ ;
  - // Set start and goal configurations
  - 7  $\tau_{t,0}^i \leftarrow \tau_{0,0}^i, \tau_{t,H-1}^i \leftarrow \tau_{0,H-1}^i \forall i$ ;
  - 8  $\epsilon_0^i \leftarrow 0, \epsilon_{H-1}^i \leftarrow 0 \forall i$ ;
  - // Compute batch loss
  - 9  $\mathcal{L}(\theta) \leftarrow \frac{1}{N} \sum_{i=1}^N \|\epsilon^i - \epsilon_\theta(\tau_t^i, \mathcal{C}^i, t^i)\|^2$ ;
  - // Update model parameters
  - 10  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$ ;

---

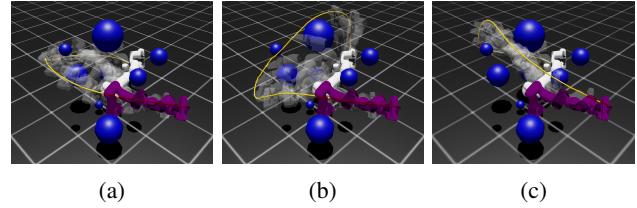


Fig. 4: Example of multi-modal trajectories generated by CAMPD in an unseen environment with spherical obstacles (blue). The white and purple arms indicate start and goal configurations, respectively. The yellow line shows the end-effector trajectory, with (c) highlighting the best trajectory of a batch of 100 samples.

## IV. EXPERIMENTAL RESULTS

To evaluate CAMPD, experiments were conducted using a 7-DoF Franka Emika Panda robot. CAMPD is compared against a classical motion planner (RRT-Connect [2] + Fatrop [8]) that combines sampling-based and optimization-based techniques, as well as MPD [15]. Additional evaluations were performed on the test set of the M $\pi$ Net [41] dataset, which features simulated real-world robotic tasks, comparing CAMPD to cuRobo [7] and DiffusionSeeder [16]. The same architecture and set of hyperparameters is used for all experiments, unless otherwise specified. See Appendix V-A for more details.

### A. Evaluation in Simulated Sphere-Based Environments

The robot is tasked with navigating from a start configuration to a goal configuration through cluttered environments

---

**Algorithm 2:** Inference

---

**Input:** Diffusion model  $\epsilon_\theta$ , start configuration  $\mathbf{q}_0$ , goal configuration  $\mathbf{q}_{H-1}$ , context  $\mathcal{C}$ , guidance strength  $w$ , sampling algorithm  $f_{\text{denoise}}$

// Initialization

- 1  $\tau_{T_{\text{inf}}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  ;
- // Set start and goal configurations
- 2  $\tau_{T_{\text{inf}},0} \leftarrow \mathbf{q}_0, \tau_{T,H-1} \leftarrow \mathbf{q}_{H-1}$  ;
- // Encode context
- 3  $z_{\mathcal{C}} \leftarrow \{\text{MLP}_k(\mathbf{c}_{k,l}) \mid \mathbf{c}_{k,l} \in \mathcal{C}\}$  ;
- 4 **for**  $t = T_{\text{inf}}, \dots, 1$  **do**
- // Encode time step
- 5  $\mathbf{z}_t \leftarrow \text{MLP}_t(\text{PE}(t))$  ;
- // Classifier-free guidance
- 6  $\epsilon'_t \leftarrow (1+w)\epsilon_{\theta,\mathbf{U}}(\tau_t, z_{\mathcal{C}}, \mathbf{z}_t) - w\epsilon_{\theta,\mathbf{U}}(\tau_t, \{\}, \mathbf{z}_t)$  ;
- // Denoise via the sampling algorithm
- 7  $\tau_{t-1} \leftarrow f_{\text{denoise}}(\tau_t, \epsilon'_t, t)$  ;
- // Set start and goal configurations
- 8  $\tau_{t,0} \leftarrow \mathbf{q}_0, \tau_{t,H-1} \leftarrow \mathbf{q}_{H-1}$  ;
- // Apply Gaussian filter
- 9  $\tau_0 \leftarrow \text{GaussianFilter}(\tau_0)$  ;

**Output:** Trajectory  $\tau_0$

---

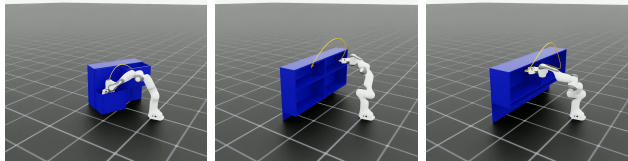


Fig. 5: Feasible trajectories generated by CAMPD for three previously unseen problems from the M $\pi$ Net [41] test set.

containing a varying number of spheres between one and ten. The randomly sized and placed spheres are spaced such that there is sufficient free space between them for the robot to pass through. The context consists of sphere instances, each represented by a vector  $\mathbf{c}_{\text{spheres},l} = [x_l, y_l, z_l, r_l]$  containing the position and radius. The training set was generated using a hybrid planner of RRT-Connect [2] and Fatrop [8], resulting in a total of 113 469 locally optimal and smooth trajectories across 2400 distinct environments. The reverse sampling process was performed using DDPM [19].

Figure 4 shows an example of multi-modal solutions generated by CAMPD in an unseen environment. The results are summarized in Table I. On a first test set of 1000 environments not seen during training, CAMPD outperforms the hybrid planner in both success rate and variance. The positive smoothness difference suggests that, on average, the trajectories generated by CAMPD are less smooth than those from the hybrid planner. Nonetheless, there are still cases where CAMPD finds more optimal solutions by identifying modes that the hybrid planner missed. Surprisingly, CAMPD surpasses the method used to generate its own dataset in these

metrics. Intuitively, this may occur because the diffusion model learns a global distribution over feasible trajectories from many planning instances, allowing it to sample solutions in regions of the solution space that the hybrid planner may fail to find a feasible trajectory.

In a second test set of 600 unseen environments (Table I, second row), involving partially fixed obstacles, CAMPD outperforms MPD in terms of success rate, feasible trajectory rate, and best smoothness difference, indicating that it is more successful in finding feasible and optimal solutions. This is particularly surprising, as part of the environment is already seen during MPD training, while it was not present in the CAMPD dataset. However, because MPD’s optimization costs balance each other, MPD finds a more diverse set of solutions, but they tend to be less smooth.

In both experiments, the guidance strength  $w$  has a significant impact on the performance of CAMPD. A higher guidance strength pushes the trajectories further away from those generated by the unconditional model (a straight line in joint space in this case, as that is the minimal mean squared error if the trajectories in the training set are equally distributed over the robot’s joint space). This can result in a higher success rate and feasible trajectory rate. However, a higher guidance strength often leads to less smooth solutions, as indicated by the best smoothness difference.

CAMPD is computationally efficient and deterministic, taking only 0.066 s per batch of 100 trajectories, compared to  $(16.49 \pm 14.73)$  s per batch of 100 trajectories for the hybrid planner, whose slowness stems from low RRT-Connect success rates, and  $(3.165 \pm 0.818)$  s per batch of 100 trajectories for MPD, whose slowness arises from cost-gradient computations.

### B. Evaluation of Simulated Real-World Tasks

To evaluate CAMPD in real-world scenarios, a model was trained to solve tasks from the M $\pi$ Net simulation test set [41]. Each task specifies an initial configuration and a target end-effector pose, and includes realistic scenes with cuboids and cylinders forming furniture-like setups. Since CAMPD requires a goal configuration, the inverse kinematics (IK) solver from cuRobo [7] is used during inference to generate a batch of feasible joint-space configurations from the target pose. The task context consists of cuboids and spheres, each represented by feature vectors:  $\mathbf{c}_{\text{cuboid},l} = [x_l, y_l, z_l, w_l, h_l, d_l, q_{w,l}, q_{x,l}, q_{y,l}, q_{z,l}]$  and  $\mathbf{c}_{\text{sphere},l} = [x_l, y_l, z_l, r_l, h_l, q_{w,l}, q_{x,l}, q_{y,l}, q_{z,l}]$ . Here,  $(x_l, y_l, z_l)$  denotes the object position,  $(w_l, h_l, d_l)$  the cuboid dimensions,  $r_l$  the sphere radius, and  $(q_{w,l}, q_{x,l}, q_{y,l}, q_{z,l})$  the orientation quaternion. This context is explicitly provided as part of the task description. The training dataset, generated using cuRobo [7], contains 1 million trajectories across 200 000 distinct environments, drawn from the same distribution as the M $\pi$ Net test set.

CAMPD is evaluated against two baselines: cuRobo [7] and DiffusionSeeder [16]. Table II reports results on the M $\pi$ Net test set, with baseline metrics for DiffusionSeeder taken from [16]. DiffusionSeeder detects obstacles using

TABLE I: Evaluation on sphere-based environments with  $N_{\text{batch}} = 100$ . Results are reported as mean  $\pm$  standard deviation. Time (T) denotes the duration to generate one batch of trajectories. Success (S) indicates whether at least one feasible trajectory exists in the batch. Feasible Trajectories Rate (FTR) is the percentage of collision-free trajectories with all joints within limits. Best Smoothness Difference (BSD) reflects the relative smoothness gap between the best CAMPD trajectory and that of the baseline. Variance (Var) quantifies the sum of distance variances across corresponding joint configurations over the trajectory dimension.

Experiment	Method	T (s) ↓	S (%) ↑	FTR (%) ↑	BSD (%) ↓	Var ↑	
Random Spheres	RRTC + Fatrop	16.49 $\pm$ 14.70*	86.6 $\pm$ 34.1	–	–	0.5 $\pm$ 1.4	
	CAMPD	$w = 1$	<b>0.066 <math>\pm</math> 0.0</b>	<b>97.5 <math>\pm</math> 15.6</b>	79.4 $\pm$ 31.0	<b>2.2 <math>\pm</math> 7.0</b>	1.7 $\pm$ 3.3
		$w = 1.5$	0.066 $\pm$ 0.0	97.4 $\pm$ 15.9	82.3 $\pm$ 30.4	3.9 $\pm$ 10.7	1.9 $\pm$ 3.7
		$w = 2$	0.066 $\pm$ 0.0	97.0 $\pm$ 17.1	<b>82.8 <math>\pm</math> 30.7</b>	5.0 $\pm$ 12.4	2.2 $\pm$ 4.5
		$w = 5$	0.066 $\pm$ 0.0	97.0 $\pm$ 17.1	67.9 $\pm$ 31.5	9.7 $\pm$ 45.2	<b>3.7 <math>\pm</math> 8.3</b>
Partially Random Spheres	MPD	3.165 $\pm$ 0.8	80.7 $\pm$ 39.5	59.6 $\pm$ 43.4	–	<b>4.8 <math>\pm</math> 6.8</b>	
	CAMPD $w = 1.5$	<b>0.066 <math>\pm</math> 0.0</b>	<b>90.7 <math>\pm</math> 29.1</b>	<b>73.3 <math>\pm</math> 37.5</b>	<b>-36.0 <math>\pm</math> 14.5</b>	1.7 $\pm$ 3.1	

\*Under ideal parallel CPU execution.

TABLE II: Evaluation on the M $\pi$ Net [41] test set. Results for DiffusionSeeder are taken from [16].

	$N_{\text{batch}} = 12$								$N_{\text{batch}} = 64$				
	cuRobo	DiffusionSeeder		IK cuRobo + CAMPD				cuRobo	IK cuRobo + CAMPD				
		$N_{\text{iters}} = 25$	50	DDPM $w = 1$	DPMSolver++ 1	2	5		DDPM $w = 1$	1	2	5	
Plan Time (ms)	26	15	17	7+20	<b>8+5</b>	8+5	8+5	111	8+52	<b>8+7</b>	8+7	8+7	
Success Rate	<b>92.0%</b>	85.1%	85.8%	90.7%	86.9%	87.6%	78.9%	93.8%	<b>98.3%</b>	96.2%	<b>97.2%</b>	93.3%	
Jerk (rad/s <sup>3</sup> )	36.5	108.8	103.6	26.1	<b>25.9</b>	29.1	41.6	62.2	<b>37.1</b>	37.2	42.0	75.6	
Motion Time (s)	<b>1.12</b>	1.26	1.26	1.38	1.36	1.37	1.47	<b>1.01</b>	<b>1.13</b>	1.14	1.14	1.22	
Translation Err (mm)	0.60	0.98	0.95	0.004	<b>0.003</b>	0.003	0.003	0.87	<b>0.003</b>	0.003	0.003	0.003	
Quaternion Err (°)	0.13	0.93	1.44	0.005	<b>0.004</b>	0.004	0.004	0.16	<b>0.004</b>	0.004	0.004	0.004	

depth camera data rather than exact object poses. To ensure fairness, Euclidean signed distance field computation time is excluded from the results. As DiffusionSeeder inherently relies on camera input, it is not possible to provide it with exact obstacle information. This setup therefore represents the fairest possible comparison between the methods. CAMPD achieves faster planning times than both cuRobo and DiffusionSeeder. Moreover, at higher batch sizes, CAMPD attains a higher success rate than cuRobo, as sampling more trajectories increases the likelihood of finding a better one. This improvement comes at only a minimal additional computational cost relative to cuRobo. However, cuRobo consistently produces trajectories with the shortest motion time. Figure 5 shows examples of tasks from the test set with a trajectory generated by CAMPD.

## V. CONCLUSIONS

This work proposes Context-Aware Motion Planning Diffusion (CAMPD), a diffusion-based motion planning method capable of incorporating contextual information. CAMPD utilizes a diffusion model conditioned on contextual factors to estimate the added noise in trajectories. Experiments conducted in simulation demonstrate that CAMPD excels in generalizing to unseen environments, outperforming the existing method MPD in terms of success rate and solution optimality. Additionally, CAMPD showcases exceptional computational efficiency, making it suitable for online applications. Its ability to generate high-quality, executable trajectories directly on the robot further highlights its po-

tential for practical deployment in complex motion planning tasks. While CAMPD shows promising results, several challenges remain. Training the model requires vast amounts of high-quality data, and the effectiveness of environment sampling strategies strongly influences performance. Moreover, CAMPD relies on structured representations such as spheres and cuboids, which can be difficult to replicate in real-world settings. Developing strategies to approximate generic objects with sets of geometric primitives from sensor data will be essential for real-world applicability. Finally, despite CAMPD’s fast runtime, its dependence on high-performance GPU hardware limits practicality in resource-constrained environments. Future work will therefore focus on reducing the dependency on large amounts of data, improving the model’s generalization capabilities, reducing its computational complexity, and expanding its applicability to sensor-based environments.

## APPENDIX

### A. Model Parameters

Diffusion steps were  $T_{\text{train}} = 25$  with latent dimension  $d_z = 64$ . The U-Net had depth 4, 4 attention heads, and input dimension 64 per head. Training used batch size 128, learning rate  $1 \times 10^{-4}$ , and unconditional probability  $p_d = 0.33$ . Models ran on an NVIDIA<sup>®</sup> GeForce RTX<sup>™</sup> 4090 GPU. For DDPM,  $T_{\text{inf}} = T_{\text{train}}$ ; for DPMSolver++,  $T_{\text{inf}} = 3$ . Final trajectories were smoothed using a Gaussian filter with standard deviation  $\sigma = 2$  and window size 7.

## ACKNOWLEDGMENTS

This work was supported by KU Leuven’s Special Research Fund (BOF), grant STG/25/004 (Advanced programming and control of smart and high-performance machines, vehicles, and robotic systems) and by Flanders Make through SBO project LearnOpTra (Learning meets optimization for robust and multimodal trajectory planning and control). Flanders Make is the Flemish strategic research center for the manufacturing industry.

## REFERENCES

- [1] S. M. LaValle *et al.*, “Rapidly-exploring random trees: A new tool for path planning,” 1998.
- [2] J. Kuffner and S. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, pp. 995–1001 vol.2.
- [3] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmap for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [4] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, “CHOMP: Covariant hamiltonian optimization for motion planning,” *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.
- [5] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, “Motion planning with sequential convex optimization and convex collision checking,” *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.
- [6] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, “Continuous-time gaussian process motion planning via probabilistic inference,” *CoRR*, vol. abs/1707.07383, 2017.
- [7] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox, “Curobo: Parallelized collision-free robot motion generation,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8112–8119.
- [8] L. Vanroye, A. Sathya, J. De Schutter, and W. Decré, “Fatrop : A fast constrained optimal control problem solver for robot trajectory optimization and control,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 10 036–10 043.
- [9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.
- [10] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, “Motion planning networks: Bridging the gap between learning-based and classical motion planners,” *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2021.
- [11] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, “From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1527–1533.
- [12] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, “Planning with diffusion for flexible behavior synthesis,” 2022. [Online]. Available: <https://arxiv.org/abs/2205.09991>
- [13] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, “Neural rrt\*: Learning-based optimal path planning,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.
- [14] T. Barbié, T. Nishio, and T. Nishida, “Trajectory prediction with a conditional variational autoencoder,” *Journal of Robotics and Mechatronics*, vol. 31, no. 3, pp. 493–499, 2019.
- [15] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, “Motion planning diffusion: Learning and planning of robot motions with diffusion models,” *arXiv.org*, 2023.
- [16] H. Huang, B. Sundaralingam, A. Mousavian, A. Murali, K. Goldberg, and D. Fox, “Diffusionseeder: Seeding motion optimization with diffusion for rapid motion planning,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.16727>
- [17] S. Aradi, “Survey of deep reinforcement learning for motion planning of autonomous vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740–759, 2022.
- [18] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2256–2265.
- [19] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851.
- [20] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” 2021. [Online]. Available: <https://arxiv.org/abs/2011.13456>
- [21] T. Ubukata, J. Li, and K. Tei, “Diffusion model for planning: A systematic literature review,” 2024. [Online]. Available: <https://arxiv.org/abs/2408.10266>
- [22] H. Cao, C. Tan, Z. Gao, Y. Xu, G. Chen, P.-A. Heng, and S. Z. Li, “A survey on generative diffusion models,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 2814–2830, 2024.
- [23] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, and S. Devlin, “Imitating human behaviour with diffusion models,” 2023. [Online]. Available: <https://arxiv.org/abs/2301.10677>
- [24] M. Reuss, M. Li, X. Jia, and R. Lioutikov, “Goal-conditioned imitation learning using score-based diffusion policies,” 2023. [Online]. Available: <https://arxiv.org/abs/2304.02532>
- [25] I. Kapelyukh, V. Vosylius, and E. Johns, “Dall-e-bot: Introducing web-scale diffusion models to robotics,” *IEEE Robotics and Automation Letters*, vol. 8, no. 7, p. 3956–3963, Jul. 2023.
- [26] 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*, 2024.
- [27] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, “3d diffuser actor: Policy diffusion with 3d scene representations,” 2024. [Online]. Available: <https://arxiv.org/abs/2402.10885>
- [28] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, “Consistency policy: Accelerated visuomotor policies via consistency distillation,” 2024. [Online]. Available: <https://arxiv.org/abs/2405.07503>
- [29] M. T. Jackson, M. T. Matthews, C. Lu, B. Ellis, S. Whiteson, and J. Foerster, “Policy-guided diffusion,” 2024. [Online]. Available: <https://arxiv.org/abs/2404.06356>
- [30] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz, “Diffusion policy optimization,” 2024. [Online]. Available: <https://arxiv.org/abs/2409.00588>
- [31] K. Saha, V. Mandadi, J. Reddy, A. Srikanth, A. Agarwal, B. Sen, A. Singh, and M. Krishna, “Edmp: Ensemble-of-costs-guided diffusion for motion planning,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2024, p. 10351–10358.
- [32] B. Matthew and T. Marc, “Planning as inference,” *Trends in Cognitive Sciences*, vol. 16, no. 10, pp. 485–488, 2012.
- [33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017.
- [34] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015.
- [35] S. Huang, Z. Wang, P. Li, B. Jia, T. Liu, Y. Zhu, W. Liang, and S.-C. Zhu, “Diffusion-based generation, optimization, and planning in 3d scenes,” 2023. [Online]. Available: <https://arxiv.org/abs/2301.06015>
- [36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” 2021. [Online]. Available: <https://arxiv.org/abs/2010.11929>
- [37] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, “Is conditional generative modeling all you need for decision-making?” 2023. [Online]. Available: <https://arxiv.org/abs/2211.15657>
- [38] J. Ho and T. Salimans, “Classifier-free diffusion guidance,” 2022. [Online]. Available: <https://arxiv.org/abs/2207.12598>

- [39] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2022. [Online]. Available: <https://arxiv.org/abs/2010.02502>
- [40] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models," 2023. [Online]. Available: <https://arxiv.org/abs/2211.01095>
- [41] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," 2022. [Online]. Available: <https://arxiv.org/abs/2210.12209>