Application of Diffusion Models for Robotic Path Planning

Beste Aydemir
Ludwig Maximilian University of Munich (LMU)
Munich, Germany
beste.aydemir@campus.lmu.de

Christophe Schmit
Technical University of Munich (TUM)
Munich, Germany
christophe.schmit@tum.de

***Abstract*—This project explores the generation of trajectories for path planning in 2D & 3D environments using diffusion models. By applying diffusion models to traditional path planning, we demonstrate their generative power for creating diverse and feasible solutions in both 2D (Maze2D Medium) and 3D (Kuka Lwr 3) path planning scenarios. We showcase the diffusion model's capability to extend beyond its training datasets after learning the trajectory data distribution. The diffusion model does not require an environment state action model as the states and actions are learned in a joint manner. Additionally, this approach can serve as an initial guess or warm start for non-learning-based methods, offering enhanced trajectory exploration.**

I. Introduction

This project seeks to expand upon the work of [1], which uses a diffusion model to generate trajectories $\boldsymbol{\tau}$ for the whole planning horizon. To get meaningful trajectories, the iterative denoising process is guided by $h(\boldsymbol{\tau})$, a function that contains information about prior evidence, preferred outcomes, or rewards. This guides the model generation towards trajectories that are closer to optimal. Many trajectory planning techniques require an understanding of the environment's behavior and they often rely on approximate models. In [1], an alternative approach is proposed to trajectory generation, where the trajectory optimization is combined into the modeling problem. This approach is reported to handle sparse rewards, as well as having the ability to plan new rewards without retraining.

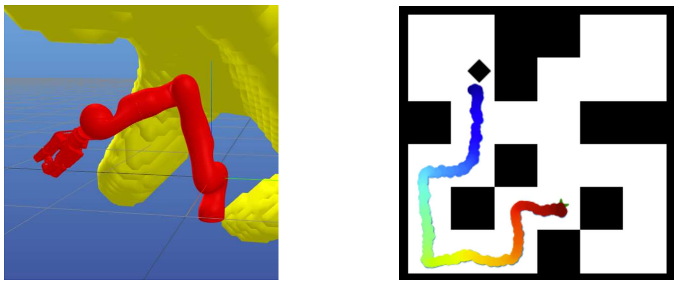
(a) Kuka Lwr3 (b) PointMaze Medium

Fig. 1: Overview: 2D & 3D Environments

A. Related Work

1) Diffusion Models: Diffusion models represent a class of probabilistic generative models that function by gradually degrading data through the introduction of noise. These models are trained to learn the reverse process, enabling them to generate new samples. A well-known variant within this category is the denoising diffusion probabilistic models (DDPMs). [2]

In this work, we aim to apply this diffusion principle to model the distribution of the state and action trajectories over a horizon $\boldsymbol{\tau}$. The generation of $\boldsymbol{\tau}$ is achieved through an iterative denoising procedure by diffusion probabilistic models. Specifically, the forward diffusion process is defined by $q(\boldsymbol{\tau}^i|\boldsymbol{\tau}^{i-1})$, where noise is incrementally added at each step i. To sample a new $\boldsymbol{\tau}$ from this distribution, the model learns how to iteratively denoise using the parameters θ, represented by $p_\theta(\boldsymbol{\tau}^{i-1}|\boldsymbol{\tau}^i)$.

Janner et al. [1] introduces a novel approach that integrates trajectory optimization, traditionally left to classical optimizers, into the modeling process, utilizing a diffusion probabilistic model for planning. The planning mechanism uses an iterative process to denoise two-dimensional arrays, encapsulating various state-action pairs. During this process, the model is constrained by a small receptive field that ensures local consistency within each denoising step as shown in 2. By combining multiple denoising iterations, this localized consistency contributes to the overall global coherence of the resulting plan. The model outputs the state action pairs for the whole planning horizon non-autoregressively.

Similar strategies involves training trajectory diffusion models with flexible constraints guided by rewards. [3] presents conditional generative models that integrate additional constraints beyond reward conditioning. However, the effectiveness of these models is limited by the quality of available offline expert data, highlighting the need for improvement in adapting to new tasks or environments.

II. Methodology

We build on the work of [1] on diffusion-based trajectory planning by applying it to a 2D and 3D environment.

A. Network Architecture

In both environments, we employed the Diffuser network architecture designed by [1] which incorporates a U-Net structure with residual blocks that include temporal convolutions, group normalization, and Mish nonlinearities. This design

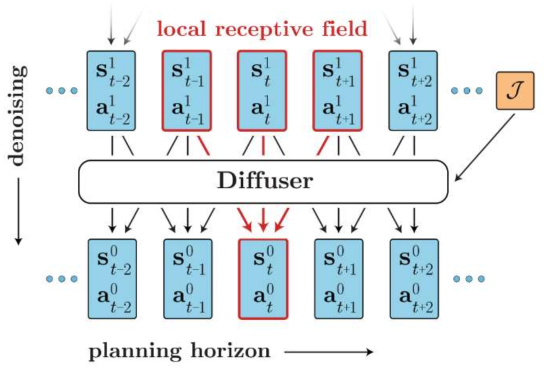

Fig. 2: Diffused Q-Values [1]

is analogous to U-Nets, known for their success in image-based diffusion models. In contrast, the diffuser architecture substitutes the two-dimensional spatial convolutions with one-dimensional temporal convolutions. Since the model is fully convolutional, the prediction horizon is dictated by the input dimensionality, allowing it to adjust dynamically during planning if needed. The network consists of 3.68 million parameters. The diffusion process is guided through conditioning, where in the 2D case, the start and target positions are fixed, and in the 3D case, the start and target q-values are fixed.

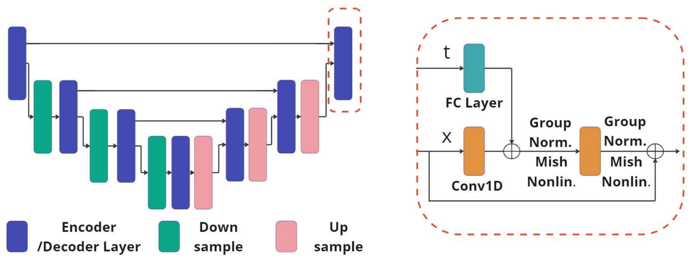

Fig. 3: U-Net Architecture

B. Dataset

Two distinct datasets were used for the path planning process, encompassing both 2D and 3D environments. The PointMaze Medium dataset represents trajectories of states and actions within a 2D maze, guided by a PD controller and QIteration [4]. Each episode features the same maze configuration, but with new start-target positions. The states are modelled as the position and velocity in both (x,y) directions. The dataset was originally generated by D4RL [5] under the Maze2D domain.

For the Kuka Lwr3, a robot with 7 degrees of freedom (DOF), the states are represented by the relative angular positions of its joints. Actions are modeled as changes in the q-values of each joint over time. Additionally, the temporal horizon of the dataset is reduced to 20 steps, in contrast to the longer horizon in the 2D case. Moreover, with 1.5 million trajectories the dataset is substantially larger compared to the 2D dataset. It is noteworthy that the trajectories show an uneven distribution in Cartesian space, with disparities on the order of magnitude of three. This imbalance results in certain start and end positions being underrepresented within the dataset. Detailed specifications of the dataset, including variations in size, are outlined in Table I.

Environment	Dim.	(State, Action) Dim.	Steps per Episode	Episodes
PointMaze	2D	(4, 2)	209	4778
Kuka Lwr3	3D	(7, 7)	20	1.5 Mio

TABLE I: Dataset descriptions for the two environments.

C. 2D PointMaze Medium

We train the Diffuser network on the specified maze dataset using the training specifications outlined in Table II. After training, the model can generate trajectories between the maze's starting and target point pair.

As we sample from the learned distribution using the model, the process begins with a random vector. Initially, the generated trajectories appear incoherent or noisy. However, as training progresses, the model becomes increasingly familiar with the trajectory data distribution, shown in Figure 5. This improved understanding allows the model to generate more coherent and plausible trajectories that better align with the maze's structure.

Figure 4a shows all the trajectories within the maze configuration. Although the dataset offers complete coverage of the maze in terms of trajectories, it lacks significant variation between two specific points. This limitation is exemplified in Figure 4b, which displays all the trajectories in the dataset between the bottom left square and the top right square. These trajectories generally follow a similar direction, leaving other potential paths unexplored.

One notable outcome of our approach is that the model can produce novel trajectories that were not explicitly present in the training data. This ability to generate previously unseen paths is a significant advantage, allowing for greater flexibility and exploration in solving the maze. Novel trajectories are visualized in Figure 5c.

The desired starting and target positions of the trajectory are specified to the model as conditioning inputs. In this context, conditioning guides the diffusion model's generation process by providing specific information about the trajectory's initial and final states. This conditioning helps the model focus on producing relevant trajectories that connect these points. Without retraining, the model can generate trajectories between any specified two points.

D. 3D Kuka Robot Lwr3

After working on the 2D PointMaze environment, we applied the Diffuser architecture to a 3D environment on the Kuka Lwr3 robot.

Figure 6 presents the outputs of the diffusion model applied to fixed start and end conditions on the validation dataset. It corresponds to the q-values of the seven individual joints and

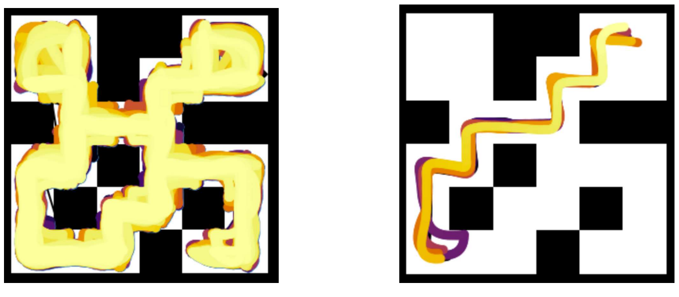
(a) Full Dataset (b) Fixed Start & Target

Fig. 4: Trajectories from the Training Dataset

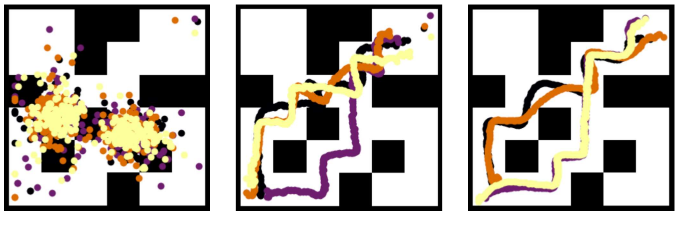
(a) Step 0 (b) Step 50k (c) Step 100k

Fig. 5: PointMaze Medium Training Evolution

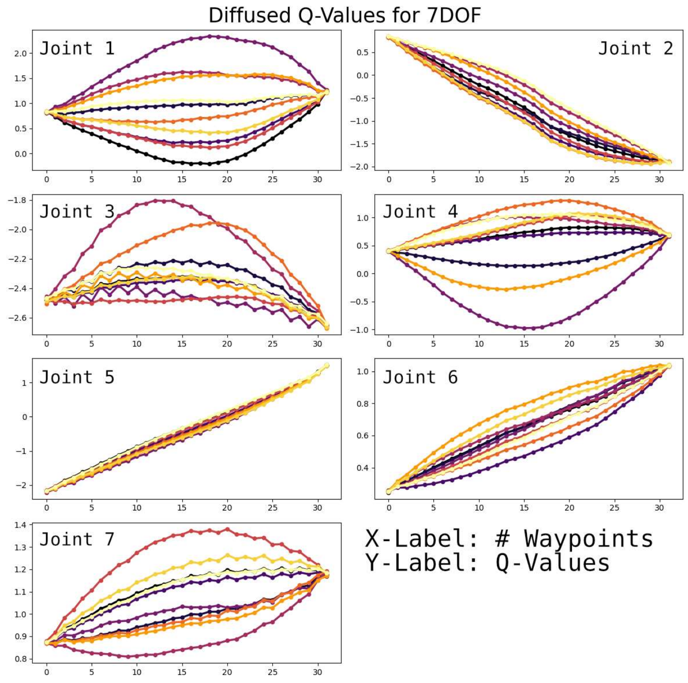

Fig. 6: Diffused Q-Values

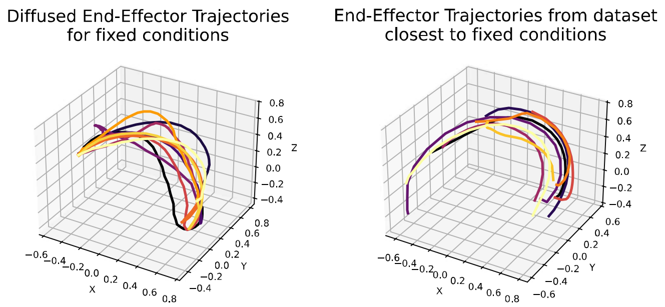

Fig. 7: Comparison: End-Effector Trajectory

Parameter	**2D Pointmaze Medium**	**3D Kuka Lwr3**
Epochs	100	25
Steps per Epoch	1,000	10,000
Training Batch Size	32	32
Learning Rate	10^{-4}	10^{-4}
Diffusion Steps	100	100
Horizon	256	32

TABLE II: Training specifications

their evolution over the planning horizon. Eight individual trajectories were sampled for the experiment, demonstrating the model's ability to generate diverse configurations for identical initial and final states. Figure 7 illustrates the corresponding end-effector trajectories, showcasing the model's capacity to produce varied paths compared to the nearest training dataset samples with similar conditions. The experiment shows that the diffusion model effectively generalizes to create various viable trajectories beyond those explicitly taught during training.

We define a safety metric to evaluate trajectory collisions within the robot environment, using an aggregation of violations across the seven joints via a signed-distance field that considers each joint's violation depth. Future fine-tuning will include extended training epochs, exploration of various hyperparameters, and adjustments to the diffusion timesteps.

E. Training Specification

Table II provides an overview of model training in both environments, highlighting changes in the number of epochs and horizon length based on the dataset configurations. The training and sampling for both environments were performed on Google Cloud Engine using a NVIDIA T4 GPU.

III. Conclusion

We successfully applied the diffusion process to 2D and 3D path planning problems in this project. Our approach demonstrated the generative power of diffusion models by creating diverse trajectory solutions that extended beyond the datasets used for training. This underscores the potential benefit of employing diffusion models to provide warm starts for traditional solvers, enhancing trajectory exploration and optimization. By integrating diffusion models with traditional path-planning methods, we have shown their utility for decision-making.

The diffusion model offers a unique approach to trajectory generation in maze environments but comes with some challenges. One major downside is that the current model for Maze2D requires retraining if the maze configuration changes. This means the model is not maze-agnostic, limiting its flexibility. To overcome this, additional guidance from the environment would be necessary. Another area for improvement is that the model has a fixed horizon length during training and inference, which could be problematic if we need trajectories of different lengths. Additionally, the sampling time on an NVIDIA T4 was between in the degree of seconds (1 and 10 seconds), which could be slow and unsuitable for real-time applications. Lastly, as we discussed before, the model does not provide any safety guarantees, which could be a concern in environments where safety is crucial.

We further note some engineering challenges with the implementation. The original implementation required many software packages and libraries with limited or no maintained support, even though the implementation was only a few years ago. Migrating all the conventionally used reinforcement learning environments from unmaintained OpenAI Gym to maintained Farama Foundation Gymnasium and reformatting the current D4RL datasets required significant effort and time. Even though our model utilized offline data in training, we observed that the ever-changing environments provide a bottleneck to every reinforcement learning approach, both in implementation effort and simulation power.

On the other hand, the diffusion model has several advantages. It presents an exciting alternative to the conventional autoregressive approach and offers new ways to tackle trajectory generation problems. There are some uses in multi-agent settings [6], as it can generate trajectories for multiple agents without requiring separate models for each one. This version of planning can be useful in different degrees as well, from high-level strategic planning to low-level motion control. Furthermore, it can be trained on something other than explicit environment dynamics, which is especially useful when these dynamics are non-deterministic or challenging to model. Another key benefit is the model's ability to generate novel trajectories, as it can combine various algorithms from the dataset in unique ways, providing more diverse solutions.

In conclusion, while the diffusion model has some limitations, its flexibility, creativity, and applicability strengths make it a valuable tool for trajectory generation, especially in complex or uncertain environments.

Future research will explore different diffusion architectures, moving beyond the current U-Net architecture to potential Vision Transformer (ViT) models as the backbone as employed in [7]. Currently, the model operates without knowledge of the training environment, limiting its effectiveness in collision avoidance. The following steps involve training ingesting environment data to evaluate collision metrics, guiding it toward generating diverse and safe trajectories. This will enable the model to create more practical and feasible path-planning solutions in complex and dynamic environments.

References

[1] 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

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[3] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision-making?" 2023.

[4] T. D. Science, "Fundamental iterative methods of reinforcement learning," 2020. [Online]. Available: https://towardsdatascience.com/fundamental-iterative-methods-of-reinforcement-learning-df8ff078652a

[5] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," 2020.

[6] 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," 2024. [Online]. Available: https://arxiv.org/abs/2305.17330

[7] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, and J. Zhu, "All are worth words: A vit backbone for diffusion models," 2023. [Online]. Available: https://arxiv.org/abs/2209.12152