Conditioned Score-Based Models for Learning Collision-Free Trajectory Generation

João Carvalho, Mark Baeirl, Julen Urain, Jan Peters
Intelligent Autonomous Systems, Technische Universität Darmstadt
joao@robot-learning.de

Abstract

Planning a motion in a cluttered environment is a recurring task autonomous agents need to solve. This paper presents a first attempt to learn generative models for collision-free trajectory generation based on conditioned score-based models. Given multiple navigation tasks, environment maps and collision-free trajectories pre-computed with a sample-based planner, using a signed distance function loss we learn a vision encoder of the map and use its embedding to learn a conditioned score-based model for trajectory generation. A novelty of our method is to integrate in a temporal U-net architecture conditioning variables such as the latent representation of the environment and task features, using a cross-attention mechanism. We validate our approach in a simulated 2D planar navigation toy task, where a robot needs to plan a path that avoids obstacles in a scene.

1 Introduction

Recent advances in new architectures and training methods of diffusion and score-based models (SBMs) have shown impressive results in image and text-to-image generation [24, 4, 23, 10, 20]. One field where these models are still not fully explored is robotics [7, 27]. Due to the multimodality and dimensionality of sensory data, e.g. inputs from vision or robot trajectories, it can be worth to study diffusion models as components of intelligent robots.

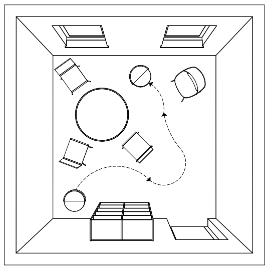
Figure 1: A planar robot task includes moving from a start to a goal position, while avoiding obstacles.

Figure 1 pictures an autonomous robot cleaning a house floor. One crucial task is to plan collision-free motions (paths) between two locations. With access to an environment map, sampling-based methods, such as variants of Probabilistic Road Maps (PRM) [9] and Rapidly exploring Random Trees (RRT) [13], are commonly used. By replanning over multiple scenarios, the robot observes data that is often unstructured, such as point clouds of tables, chairs or shelves. It is therefore natural to reuse this collected information if it needs to replan in a new situation. To speed up planning, several methods have proposed neural motion planners that encode environment and motion information using neural networks, which are afterwards used as priors for planners [17, 19].

In this work, we leverage recent developments in SBMs and Signed Distance Fields (SDFs) [16] to build a generative model for collision-free trajectory generation, conditioned on an environment map and task relevant features. We summarize our contributions as: (1) learn an environment encoder based on an SDF loss to encode geometrical properties; (2) introduce a conditioning model for planning - the Conditional Temporal U-Net - extending the work of Janner et al. [7] with ideas from Rombach et al. [23]; (3) learn a generative SBM of collision-free trajectories given a new environment and task; (4) show first results in a simulated 2D navigation task.

NeurIPS 2022 Workshop on Score-Based Methods.

Score-Based Models for robotics Few works have explored using score-based and diffusion models in robotics. Janner et al. [7] presented the *Diffuser*, a diffusion model for trajectory planning based on temporal convolutions, which enforces temporal ordering and locality. We use their Temporal U-Net architecture as a component in our work. Wang et al. [31] used diffusion as regularizers and policies for step-based offline reinforcement learning (RL). In robotics, Urain et al. [27] used diffusion to learn cost functions for jointly optimizing motion and grasping poses.

Motion planning We consider two methods for motion planning: sampled-based and optimization-based. Both strategies can be augmented with better sampling distributions learned from previous data to guide and speed up planning. **Sampling-based motion planning** includes classical algorithms such as PRM [9], RRT [13] and RRT* [12]. Several works have proposed learning conditional sampling distributions using the environment and task information as context variables, e.g. in [6] the environment is given as an occupancy map. Similar to our work, Deep Sampling-based Motion Planner (DeepSMP) [17] learns an autoencoder given a point cloud. Instead, we learn an encoder trained on an SDF loss, which has two advantages - there is no need to learn a decoder for reconstruction; and we get access to the environment's SDF and its gradient, which can be used as an extra cost for planning. Other works [17, 19, 18, 30] learn a conditioned one-step neural planner, and to sample different solutions they add dropout to ensure stochasticity. Contrarily, we learn a trajectory distribution model to easily introduce the notion of smoothness, which is important in robot motions. **Optimization-based motion planning** includes methods that optimize a trajectory either via gradient optimizers or stochastic optimization, e.g. Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [21], Stochastic Trajectory Optimization for Motion Planning (STOMP) [8] or Gaussian Process Motion Planner (GPMP) [15]. These methods often use an uninformed initial distribution at the start of optimization. In CHOMP a straight trajectory connects the initial and final points, STOMP uses a distribution, whose mean is a straight line, but has high entropy in the middle of the trajectory, and decreasing entropy towards the initial and final points, GPMP uses a Gaussian Process Prior. Learning better initial distributions can thus speed up these methods, as shown in [28]. Our conditioned SBM can be used a prior distribution for both sampling and optimization-based planners.

2 Background

2.1 Motion Planning as Inference

Let $\boldsymbol{s} \in \mathcal{S} \subseteq \mathbb{R}^d$ encode the state of a robot (agent) and its environment. In motion planning, a trajectory is represented in discrete-time with horizon H as a sequence of states $\boldsymbol{\tau} \triangleq (\boldsymbol{s}_1, \dots, \boldsymbol{s}_H) \in \mathbb{R}^{H \times d}$. It is common to optimize $\boldsymbol{\tau}$ given a *context* $\mathcal{C}$, which can include an occupancy map, obstacle locations, start and final positions, etc. Optimization-based motion planning formulates the problem as trajectory optimization $\boldsymbol{\tau}^* = \arg\min_{\boldsymbol{\tau}} \sum_i c_i(\boldsymbol{\tau}, \mathcal{C})$, where c_i are different costs related to trajectory smoothness, obstacle avoidance or goal reaching [28]. The connection between trajectory optimization and probabilistic inference is well established [1, 26, 14]. Following the notation in [7], the probability of a trajectory (a random variable) factorizes (up to a normalizing constant) as $\tilde{p}(\boldsymbol{\tau}|\mathcal{C}) \propto p(\boldsymbol{\tau}|\mathcal{C})h(\boldsymbol{\tau}|\mathcal{C})$, where $p(\boldsymbol{\tau}|\mathcal{C})$ is a prior distribution (that will be will be learned from data) and $h(\boldsymbol{\tau}|\mathcal{C})$ is a task-specific distribution, e.g. a delta distribution for the starting state of the trajectory. Trajectory optimization computes the maximum-a-posteriori solution $\boldsymbol{\tau}^* = \arg\max_{\boldsymbol{\tau}} \log \tilde{p}(\boldsymbol{\tau}|\mathcal{C})$. On the other hand, in inference we sample from $\tilde{p}(\boldsymbol{\tau}|\mathcal{C})$, which allows to get multiple solutions. Langevin dynamics [32] is a common approach to sample from $\tilde{p}$, for which we need $\nabla_{\boldsymbol{\tau}} \log \tilde{p}(\boldsymbol{\tau}|\mathcal{C}) = \nabla_{\boldsymbol{\tau}} \log p(\boldsymbol{\tau}|\mathcal{C}) + \nabla_{\boldsymbol{\tau}} h(\boldsymbol{\tau}|\mathcal{C})$. We propose to approximate the gradient of the log-prior distribution on trajectories with a *conditioned* SBM $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{\tau}, \boldsymbol{c}) \approx \nabla_{\boldsymbol{\tau}} \log p(\boldsymbol{\tau}|\mathcal{C})$.

2.2 Score-Based Models as Trajectory Generative Models

Several generative modelling techniques, such as Generative Adversarial Networks (GANs) [3], Variational Auto Encoders (VAEs) [11] and Normalizing Flows (NFs) [22], are trained to maximize the data log-likelihood. Sampling is done by applying deterministic transformations to a random variate from a simple distribution. Instead, SBMs are implicit models [5, 29, 24], which perturb the data with diffusion and learn to reconstruct it by denoising using the score-function - the gradient of the log-probability w.r.t. the input. Sampling is done by iteratively transforming a sample from a simple distribution following the (parametrized) score-function $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_t, t) \approx \nabla_{\boldsymbol{\tau}_t} \log p(\boldsymbol{\tau}_t, t)$, where

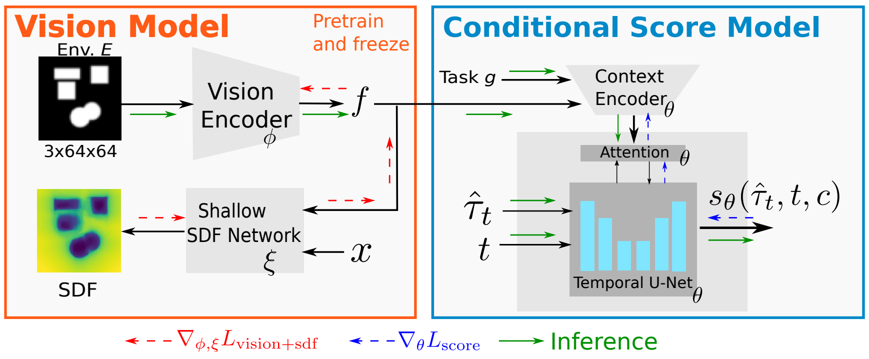

Figure 2: Architecture of our approach. A vision encoder is learned using an SDF loss. The SDF network is shallow to force the encoder to learn the geometric properties of the environment. The task and vision embeddings are concatenated as inputs to a context encoder, used to condition the SBM. The score-model is implemented as a temporal U-Net conditioned with a cross-attention mechanism.

$t \in [0, 1]$ is the time of the denoising step. The goal is to move an initial noisy sample $\boldsymbol{\tau}_1$ to a sample from the data distribution $\boldsymbol{\tau}_0$. SBMs can be trained with Denoising Score Matching (DSM) by minimizing

$$\mathcal{L}_{\text{score}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}), t\sim\mathcal{U}(0,T)} \left[\lambda(t)\mathbb{E}_{\boldsymbol{\tau}_t\sim p(\boldsymbol{\tau}_t|\boldsymbol{\tau},t)} \left[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_t, t) - \nabla_{\boldsymbol{\tau}_t} \log p(\boldsymbol{\tau}_t|\boldsymbol{\tau}, t)\|^2\right]\right], \quad (1)$$

where $p(\boldsymbol{\tau}_t|\boldsymbol{\tau}, t) = \mathcal{N}\left(\boldsymbol{\tau}_t; \boldsymbol{\tau}, \sigma^2(t)\mathbf{I}\right)$ is the density of the perturbed trajectory at time t, $\lambda(t) = (a^{2t} - 1)/(2\log a)$ and $\sigma(t) = \sqrt{\lambda(t)}$. Due to space constraints, for more details we refer the reader to [25]. In [7] a discrete-time denoising diffusion probabilistic model is used instead [4].

3 Conditioned Score-Based Generator for Collision-Free Trajectories

Given a dataset of environments E, task features $\boldsymbol{g}$ (e.g. initial and final positions) and trajectories $\mathcal{D} = \{(\boldsymbol{\tau}^i, E^i, \boldsymbol{g}^i)\}_{i=1}^N$, we model a conditional distribution $p(\boldsymbol{\tau}|\boldsymbol{c})$. The environment E is given as an occupancy map, which is transformed into latent features with a vision encoder $\boldsymbol{f}_{\boldsymbol{\phi}}(E)$ parametrized by $\boldsymbol{\phi}$. We create a context embedding $\boldsymbol{c} = \boldsymbol{c}_{\boldsymbol{\theta}}(\boldsymbol{f}_{\boldsymbol{\phi}}(E), \boldsymbol{g})$, which is used as a conditioning variable for the SBM. We propose to learn an implicit representation of $p(\boldsymbol{\tau}|\boldsymbol{c})$ by modelling a conditioned score function $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_t, t, \boldsymbol{c}) \approx \nabla_{\boldsymbol{\tau}_t} \log p(\boldsymbol{\tau}_t, t|\boldsymbol{c})$, implemented as a ***conditional temporal U-Net***. The network architecture uses the Diffuser from [7] as an unconditional model, and we introduce conditioning by using a cross-attention mechanism at the end of each temporal residual block of the U-Net, similar to [23]. We experimented with other types of conditioning, e.g. concatenating the context embedding, trajectory and time, but found that cross-attention produced better results.

Learning the vision encoder and score models The overall architecture is depicted in Fig. 2. Extra details can be found in App. B. The **vision model** takes as input an image E (black-and-white occupancy map), and produces latent features $\boldsymbol{f}_{\boldsymbol{\phi}}(E)$. With access to the distribution of environments and ground-truth signed distance values $\text{sdf}_{\text{gt}}(E, \boldsymbol{x})$ for a point $\boldsymbol{x} \in \mathbb{R}^2$, we pretrain the vision encoder, a CNN with parameters $\boldsymbol{\phi}$, by minimizing an SDF L_1-loss

$$\mathcal{L}_{\text{vision + sdf}}(\boldsymbol{\phi}, \boldsymbol{\xi}) = \mathbb{E}_{E\sim p(E), \boldsymbol{x}\sim p(\boldsymbol{x}|E)} \left[|\text{SDF}_{\text{gt}}(E, \boldsymbol{x}) - \hat{\text{SDF}}_{\boldsymbol{\xi}}(\boldsymbol{f}_{\boldsymbol{\phi}}(E), \boldsymbol{x})|\right], \quad (2)$$

where $\hat{\text{sdf}}_{\boldsymbol{\xi}}$ is the learned SDF. We use a shallow SDF network, to enforce the vision encoder to represent as much of the environment features. The latent representation $\boldsymbol{f}$ could also be learned with a VAE, but this model would be larger (due to the decoder) and would not give access to the SDF, which we use for ranking samples. With a pretrained vision model, the **conditioned SBM** is trained by sampling a batch of environments, tasks and trajectories, and minimizing the score loss with DSM

$$\mathcal{L}_{\text{score}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau}, E, \boldsymbol{g}\sim\mathcal{D}}\mathbb{E}_t \left[\lambda(t)\mathbb{E}_{\boldsymbol{\tau}_t} \left[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_t, t, \boldsymbol{c}_{\boldsymbol{\theta}}(\boldsymbol{f}_{\boldsymbol{\phi}}(E), \boldsymbol{g})) - \nabla_{\boldsymbol{\tau}_t} \log p(\boldsymbol{\tau}_t|\boldsymbol{\tau}, t)\|^2\right]\right]. \quad (3)$$

Following [25, 23], the context encoder $\boldsymbol{c}_{\boldsymbol{\theta}}$ and the conditioned SBM $\boldsymbol{s}_{\boldsymbol{\theta}}$ are jointly trained. We experimented with other types of architectures, such as learning jointly the vision features and the score-model, but found it to be more difficult to train. Also, our solution is modular, allowing to use other types of pre-trained modules needed for robotics without retraining everything, e.g. one could use dense object features to create movements conditioned on a particular object.

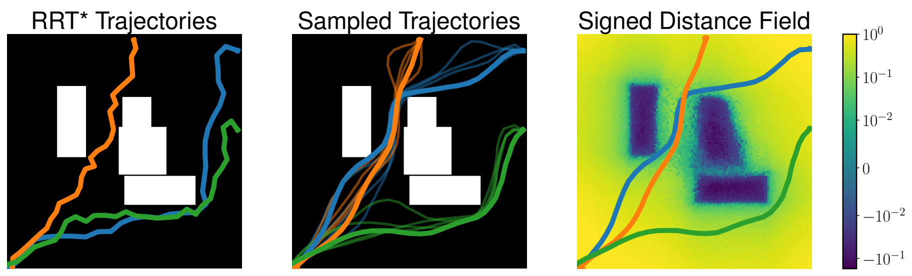

Figure 3: This figure shows a sample from the validation set of the RECTANGLES AND CIRCLES environments. The environment is 64×64 image, where white is where an obstacle is present. The left figure shows trajectories generated with RRT* for different goal positions (top right area). Thinner trajectories in the center are sampled with the conditioned SBM. Thicker trajectories are the ones considered after ranking. The right plot shows the learned SDF and the selected trajectories.

Trajectory generation and ranking Given a new environment and task we compute the context vector $\boldsymbol{c}$. To generate trajectories we solve a reserve SDE by using the learned SBM and the Euler-Maruyama solver. In practice, we sample a noisy trajectory $\boldsymbol{\tau}_1 \sim \mathcal{N}\left(\mathbf{0}, 0.5(a^2-1)\mathbf{I}\right)$ and iteratively run $\boldsymbol{\tau}_{t-\Delta t} = \boldsymbol{\tau}_t + a^{2t}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_t, t, \boldsymbol{c})\Delta t + a^t\sqrt{\Delta t}\boldsymbol{z}$, where Δt is a time discretization (in our case $\Delta t = 1/500$), $a = 1.1$ and $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In a real world scenario we are able to execute only one trajectory. Therefore, we propose a simple heuristic metric to select and rank trajectories – discard the ones if any point along them has a negative SDF, which means collision with an obstacle. Afterwards, we select the trajectory that has the shortest path. Even though this does not guarantee the optimal trajectory is found, it can be a good first heuristic prior for motion optimization methods.

4 2D Planar Navigation Experiments

Our simulated setup includes an image of an environment and a robot located at an initial position (lower left) that needs to move to a final position (upper right) while avoiding obstacles (Fig. 3). The datasets consist of an image that describes a 2D environment, different tasks for each environment where tasks are different start and end points, and RRT* generated collision-free trajectories. The training set has 50k environments, each having 10 random tasks and one RRT* demonstration. To train the trajectory generator, we used an horizon of 32 steps, hence $\boldsymbol{\tau} \in \mathbb{R}^{32\times 2}$. The validation set corresponds to 5% of the total data. For each environment an SDF for the obstacles is created as well.

Fig. 3 shows the results in the validation set of the RECTANGLES AND CIRCLES dataset. The left plot shows the trajectories generated with RRT*, for different goal positions. Thinner trajectories in the center plot are sampled with our learned conditioned SBM, and thicker trajectories are the ones considered after ranking. The right plot shows the learned SDF. Notice how these trajectories traverse regions with positive SDF values. In the center plot some of the green trajectories cross the obstacle, which can be partially explained by errors in the learned SDF. Nevertheless, these trajectories are discarded by our heuristic. As a quantitative measure, we evaluated our method across all validation environments and obtained around 90% of collision free trajectories. More experiments can be seen in Figures A.1 and A.2, where we observe that the model produces diverse (multimodal) trajectories.

5 Conclusion and Future Work

In this work we presented an architecture for a conditional generative score-based model, with applications to collision-free trajectory generation for planar navigation. To encode a conditional distribution, we augmented a temporal U-Net model with a conditioning mechanism based on cross-attention. As contextual variables we use task features and a latent encoding of an environment map, which is trained with a signed distance function loss. First experiments of our method in a simulated 2D planar navigation task show promising results to encode collision-free trajectory generation. In future work, we will incorporate our learned model as a prior to speed up sample-and optimization-based planners and extend our approach to robots arms with more degrees of freedom. We believe that due to the modelling power of score-based methods, they can be used as learned components of intelligent robotic systems.

Acknowledgments and Disclosure of Funding

This work was funded by the German Federal Ministry of Education and Research (project IKIDA, 01IS20045). Authors are with the Institute for Intelligent Autonomous Systems (IAS) of the Technische Universität Darmstadt (TU Darmstadt). Jan Peters is affiliated with the German Research Center for AI (DFKI), Research Department: Systems AI for Robot Learning, Hessian.AI and the Centre for Cognitive Science of TU Darmstadt. Emails: `{joao,julen,jan}@robot-learning.de`, `mark.baierl@stud.tu-darmstadt.de`

References

[1] Hagai Attias. Planning by probabilistic inference. In Christopher M. Bishop and Brendan J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, volume R4 of *Proceedings of Machine Learning Research*, pages 9–16. PMLR, 03–06 Jan 2003. URL `https://proceedings.mlr.press/r4/attias03a.html`. Reissued by PMLR on 01 April 2021.

[2] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 303–312, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917464. doi: 10.1145/237170.237269. URL `https://doi.org/10.1145/237170.237269`.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[5] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL `http://jmlr.org/papers/v6/hyvarinen05a.html`.

[6] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning, 2017. URL `https://arxiv.org/abs/1709.05448`.

[7] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[8] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13, 2011. URL `http://www-clmc.usc.edu/publications/K/kalakrishnan-ICRA2011.pdf`. clmc.

[9] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi: 10.1109/70.508439.

[10] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2426–2435, June 2022.

[11] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

[12] J.J. Kuffner and S.M. 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)*, volume 2, pages 995–1001 vol.2, 2000. doi: 10.1109/ROBOT.2000.844730.

[13] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning, 1998.

[14] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018. URL `https://arxiv.org/abs/1805.00909`.

[15] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, sep 2018. doi: 10.1177/0278364918790369. URL `https://doi.org/10.1177%2F0278364918790369`.

[16] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[17] Ahmed Hussain Qureshi and Michael C. Yip. Deeply informed neural sampling for robot motion planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pages 6582–6588. IEEE, 2018. doi: 10.1109/IROS.2018.8593772. URL `https://doi.org/10.1109/IROS.2018.8593772`.

[18] Ahmed Hussain Qureshi, Anthony Simeonov, Mayur J. Bency, and Michael C. Yip. Motion planning networks. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 2118–2124. IEEE, 2019. doi: 10.1109/ICRA.2019.8793889. URL `https://doi.org/10.1109/ICRA.2019.8793889`.

[19] Ahmed Hussain Qureshi, Jiangeng Dong, Austin Choe, and Michael C. Yip. Neural manipulation planning on constraint manifolds. *IEEE Robotics Autom. Lett.*, 5(4):6089–6096, 2020. doi: 10.1109/LRA.2020.3010220. URL `https://doi.org/10.1109/LRA.2020.3010220`.

[20] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL `https://arxiv.org/abs/2204.06125`.

[21] Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494, 2009. doi: 10.1109/ROBOT.2009.5152817.

[22] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org, 2015. URL `http://proceedings.mlr.press/v37/rezende15.html`.

[23] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[24] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11895–11907, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html`.

[25] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

[26] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1049–1056, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553508. URL `https://doi.org/10.1145/1553374.1553508`.

[27] Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. Se(3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion, 2022. URL `https://arxiv.org/abs/2209.03855`.

[28] Julen Urain, An T. Le, Alexander Lambert, Georgia Chalvatzaki, Byron Boots, and Jan Peters. Learning implicit priors for motion optimization, 2022. URL `https://arxiv.org/abs/2204.05369`.

[29] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_a_00142.

[30] Jiankun Wang, Wenzheng Chi, Chenming Li, Chaoqun Wang, and Max Q.-H. Meng. Neural rrt*: Learning-based optimal path planning. *IEEE Transactions on Automation Science and Engineering*, 17(4):1748–1758, 2020. doi: 10.1109/TASE.2020.2976560.

[31] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning, 2022. URL `https://arxiv.org/abs/2208.06193`.

[32] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

A Additional Results

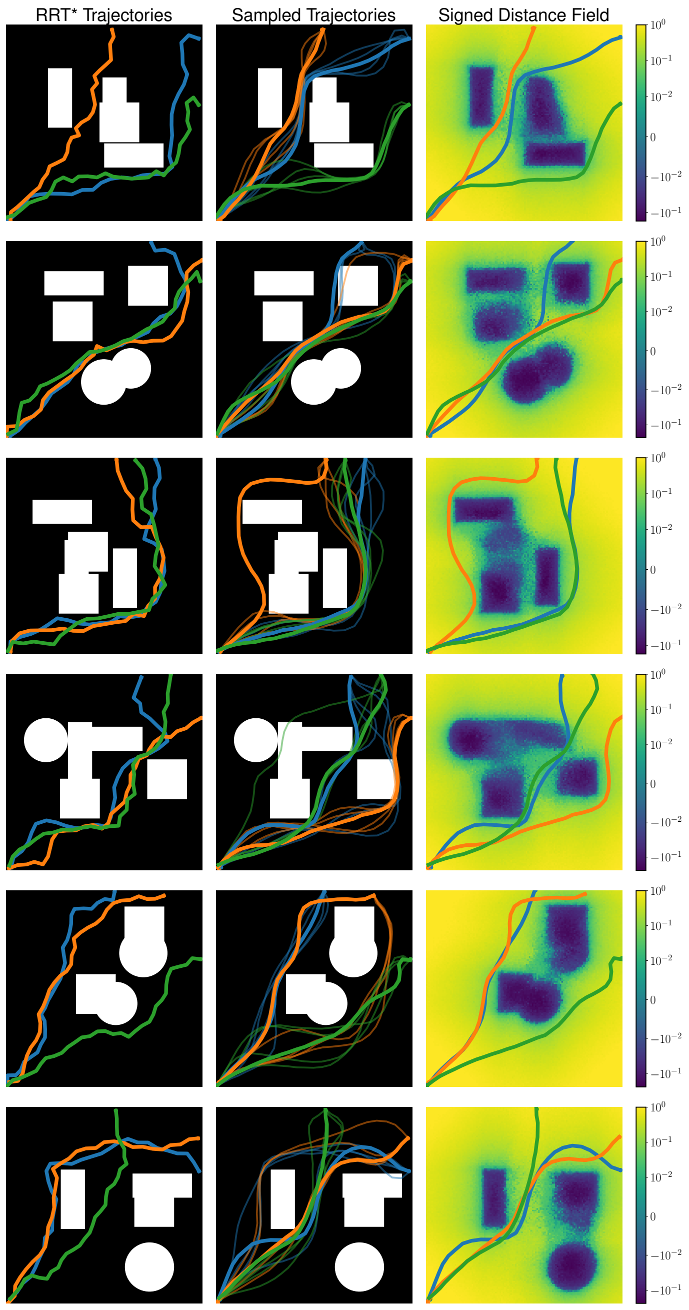

Figure A.1: This figure shows a sample from the validation set of the RECTANGLES AND CIRCLES environments. The environment is depicted as black-and-white 64×64 image, where white is where an obstacle is present. The leftmost figure shows the trajectories generated with RRT*, for different goal positions. Thinner trajectories in the center plot are sampled with our learned conditioned SBM. Thicker trajectories are the ones considered after ranking. The rightmost plot shows the learned SDF and the selected trajectories.

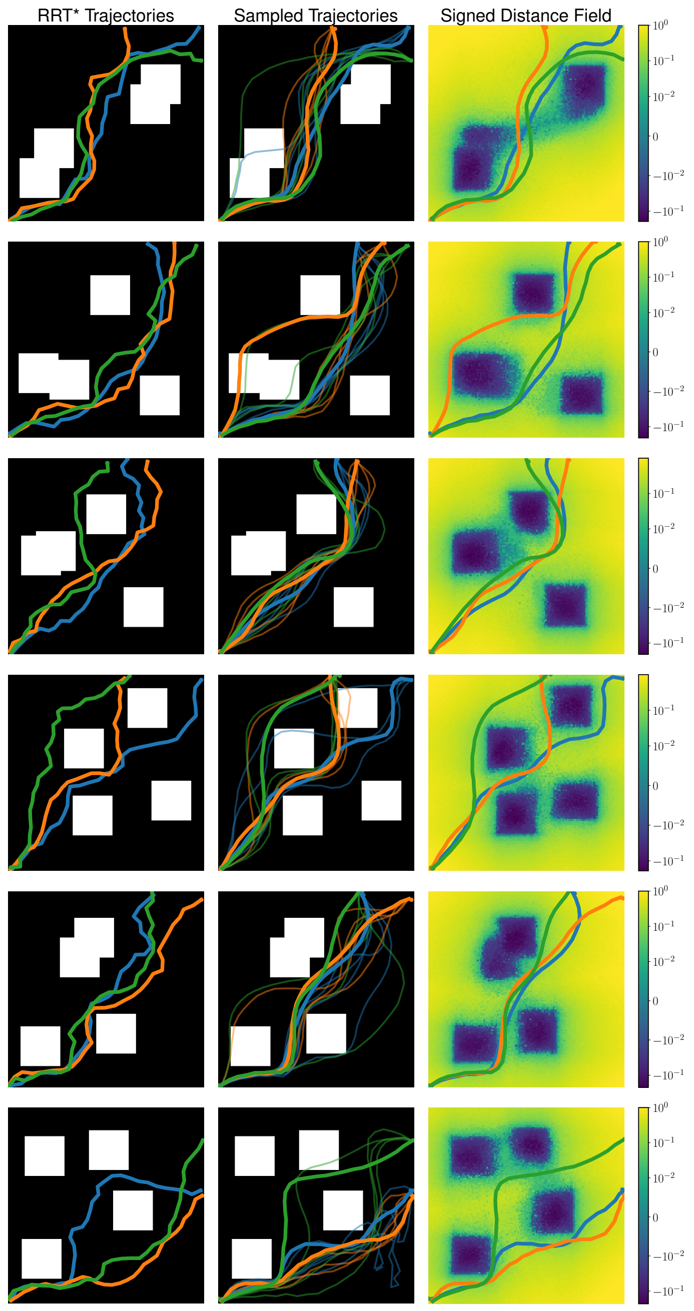

Figure A.2: This figure shows a sample from the validation set of the SQUARES environments. The environment is depicted as black-and-white 64×64 image, where white is where an obstacle is present. The leftmost figure shows the trajectories generated with RRT*, for different goal positions. Thinner trajectories in the center plot are sampled with our learned conditioned SBM. Thicker trajectories are the ones considered after ranking. The rightmost plot shows the learned SDF and the selected trajectories.

B Details on the Model Architecture

The temporal U-Net architecture and dimensions are the same as in the original work [7]. Table 1 details the other components of the model. More details on the architecture, training, hyperparameters, etc., can be found in the accompanying code upon publication.

Table 1: Vision Encoder. Conv2d:=(out channels, kernel size, stride, padding)

Vision Encoder	
Layer	Dimensions
Input	3x64x64
Conv2d	(32, 5, 2, 2), ReLU
Conv2d	(32, 5, 2, 2), ReLU
Conv2d	(64 , 5, 2, 2), ReLU
Conv2d	(128, 5, 2, 2), ReLU
Conv2d	(256, 5, 2, 2), ReLU
Flatten	-
Linear	256, ReLU
Linear	128, ReLU
Output	128

Shallow SDF	
Layer	Dimensions
Input	128 (vision encoder) + 2 (point in 2D space)
Linear	128, ReLU
Linear	128, ReLU
Output	128

Context Encoder	
Layer	Dimensions
Input	128 (vision encoder) + 4 (task features)
Linear	128, ReLU
Output	128

Cross Attention	
Parameter	Dimensions
attention heads	2
attention dim	64

C Background on Deep Signed Distance Functions

Given an environment consisting of free-space and obstacles, Signed Distance Functions (SDFs) are continuous mappings that compute the distance from a point in space to the closest obstacle surface $\text{SDF}(\boldsymbol{x}) \in \mathbb{R}$ [2]. In our current approach $\boldsymbol{x} \in \mathbb{R}^2$, but in general it is a point in 3D. If $\text{SDF}(\boldsymbol{x}) < 0$, the point lies inside of the obstacle, $\text{SDF}(\boldsymbol{x}) > 0$ it lies outside, and if $\text{SDF}(\boldsymbol{x}) = 0$ it lies on the surface. To generalize to multiple shapes, deep SDFs [16] use deep neural networks to learn both the latent representation $\boldsymbol{z}_S$ of a shape S and encode the distance function, $\hat{\text{SDF}}_{\boldsymbol{\xi}}(\boldsymbol{z}_S, \boldsymbol{x}) \approx \text{SDF}_S(\boldsymbol{x})$. They use an auto-decoder network to learn the latent shape embedding, and jointly train this network and the SDF network to learn the geometric details of an object. Note that the SDF network should be responsible for learning only a metric, and the latent embedding should encode the geometrical properties of the object. We leverage this idea to learn a geometry encoding of our 2D environments.