DiPPeR: Diffusion-based 2D Path Planner applied on Legged Robots

Jianwei Liu*, Maria Stamatopoulou* and Dimitrios Kanoulas

***Abstract*— In this work, we present DiPPeR, a novel and fast 2D path planning framework for quadrupedal locomotion, leveraging diffusion-driven techniques. Our contributions include a scalable dataset of map images and corresponding end-to-end trajectories, an image-conditioned diffusion planner for mobile robots, and a training/inference pipeline employing CNNs. We validate our approach in several mazes, as well as in real-world deployment scenarios on Boston Dynamic's Spot and Unitree's Go1 robots. DiPPeR performs on average 70 times faster for trajectory generation against both search based and data driven path planning algorithms with an average of 80% consistancy in producing feasible paths of various length in maps of variable size, and obstacle structure.**

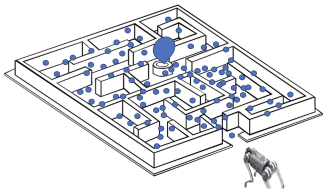
I. INTRODUCTION

Mobile robots, and especially legged ones, have the capacity to evolve into multi-purpose machines, useful in many application scenarios, such as production sites, household services, remote inspection, and disaster search-and-rescue [1], [2], [3]. Path planning is crucial in enabling legged robots to navigate autonomously and effectively complete the attributed tasks in various complex environments. Several studies, e.g. [4], [5], were dedicated towards the development of safe and efficient path planning algorithms, many of which utilize traditional methods such as Rapidly-exploring Random Trees (RRT) and A^*-based methods. However, such approaches often struggle to effectively handle the complexities and uncertainties associated with real-time sensor inputs [6]. Efficient and reliable path planning for quadrupeds is an ongoing challenge, with data driven approaches, such as Neural A* [7], showing promising efforts into overcoming the shortcomings of traditional approaches. Learning from demonstration methods using image conditioned Diffusion, have also shown promising results in path planning, applied mainly to manipulators [8], [9], [10], however, with minimal literature on their application on quadrupeds. Diffusion policies iteratively infer the action-score gradient, conditioned on visual observations. This allows for expression of multi-modal action distributions and scalability to higher-dimensional output space (allowing the generation of sequence of future actions), and training stability while maintaining distributional expressivity [8].

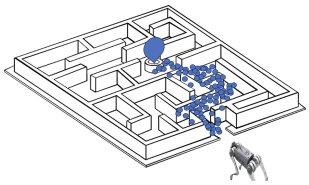
The authors are with the Department of Computer Science, University College London, Gower Street, WC1E 6BT, London, UK. {jianwei.liu.21, maria.stamatopoulou.21, d.kanoulas}@ucl.ac.uk

*equal contribution

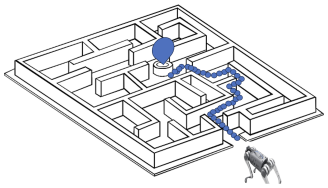
This work was supported by the UKRI Future Leaders Fellowship [MR/V025333/1] (RoboHike) and the CDT for Foundational Artificial Intelligence [EP/S021566/1].



(a) Random Noise initialization



(b) Half way through denoising



(c) Final denoised trajectory plan

Fig. 1: Illustration of DiPPeR global path generation process.

In this paper, we leverage the progress made in diffusion driven path planning, and develop an end-to-end 2D path planning framework for quadrupedal locomotion. We develop a training and inference pipeline using a Convolutional Neural Network (CNN) based architecture. The main contributions of our method is the introduction of:

1) a scalable dataset comprising of randomly generated mazes and corresponding end-to-end trajectories,
2) an image-conditioned diffusion planner for mobile robots,
3) a training/inference pipeline using CNN architecture,
4) trajectory generation significantly faster than both sample based and data driven SOTA path planners and
5) a real-world deployment stack with a platform-invariant framework validation.

The remaining of the paper is structured as follows. In Sec. II, we briefly introduce literature in path planning relevant to our proposed method. In Sec. III, we provide the necessary background knowledge. In Sec. IV, we define our proposed method, with our experimental results presented in Sec. V. Finally, in Sec. VI, we summarize the results and we conclude with some future work.

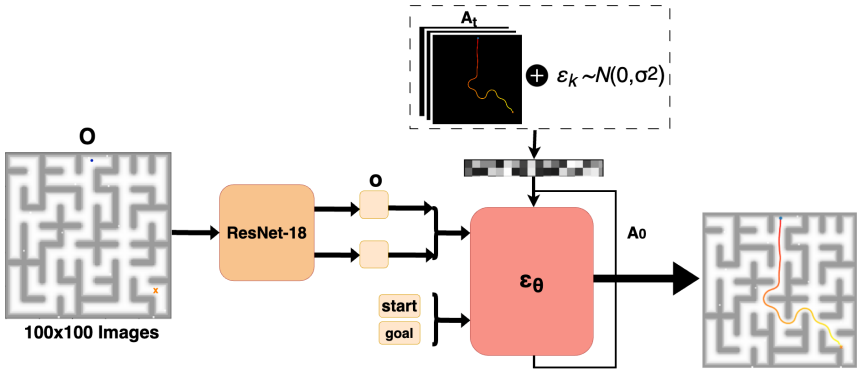


Fig. 2: DiPPeR - Image Conditioned Diffusion Training Pipeline: A Map Image Observations sample O is fed to the ResNet-18 Visual Encoder and converted to latent embeddings o. The x and y of the start and goal positions are also added as part of O. Noise e_k sampled from the prior Gaussian Distribution is added to the trajectory instance A_t. The noisy sample is passed as an input to the diffusion network ϵ_θ and is conditioned by O. The network ϵ_θ takes the form of a CNN and it outputs the denoised action A_0.

II. RELATED WORK

Path planning algorithms, have a long history in robotics and are primarily split between classical and data-driven.

A. Classical Path Planning

Classical approaches in path planning rely on search-based and sampling-based methods. Search-based path planning provides mathematical guarantees of converging to a solution if it exists. A^* and its variations, offer simplicity in implementation and effectiveness to find valid paths. For instance, in very recent works [11], the authors introduced an extension of A^* to drive a mobile platform to sanitize rooms. In [12], an A^* algorithm was used to find the collision-free path for a legged robot to achieve autonomous navigation. Traditional methods heavily rely on fixed heuristic functions, such as the Euclidean distance, which lacks adaptability to varying robot configurations and environments and are usually computational heavy. In our work, such heuristics are not required as the path is learned through demonstration of multiple optimal trajectories.

Sampling-based planners efficiently create paths in high-dimensional spaces, by sampling points in the state space. Relevant literature in the field of quadrupedal robots include [4], where an extension of an RRT-based algorithm is used for controlling a quadruped robot during the DARPA Robotics Challenge in 2015. More recently in [5], a novel sampling-based planner was introduce to shorten the computational time for finding a new path for quadrupedal robots. While these approaches demonstrate satisfactory performance and probabilistic convergence, their limitations lie in the increasing planning time as the complexity of the environment increases, due to the iterative nature of the algorithms. Our proposed method, utilizes diffusion process to parallelize the generation of the entirety of the trajectory, overcoming this limitation.

B. Data-Driven Path Planning

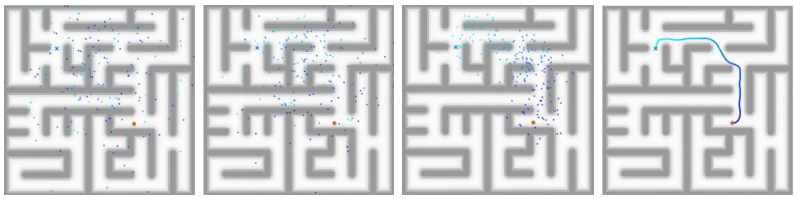
State-of-the-art research in the field has shifted towards incorporating machine learning techniques, which directly learn the behavior of path finding. These methods employ approaches such as expert demonstration [13] or imitation learning [6] to learn how to plan paths. Recent works directly address the issue of lack of semantically labeled maps in classical search-based methods by using data-driven approaches directly on raw image [14], [6], [15]. Specifically, Yonetani et al. [16] introduced Neural A^* (N-A^*) – a differentiable variant of the canonical A^*, coupled with a neural network trained end-to-end. The method works by encoding natural image inputs into guidance maps and searching for path-planning solutions on them, resulting in significant performance improvements over previous approaches in terms of efficiency. However, the use of Convolutional Neural Networks (CNNs), introduce limitations in the processing of larger maps due to computational complexity and reduction of performance. Our method, avoids this limitation allowing fast training and inference for input maps of any size.

C. Diffusion for Path Planning

Diffusion methods have gain popularity in the domain of path planing with many works presenting promising results. Hong et al. [17] developed diffusion maps applied to find a local path for reaching a goal and avoiding collisions with dynamic obstacles simultaneously, by computing transition probabilities between grid points. Janner et al. [10] and Chi et al. [8] developed impressive path planners, applied to robotic manipulators, by providing demonstration data and learning the trajectories through diffusion. We aim to leverage the promising results and develop a diffusion planing pipeline applied to quadrupedal locomotion.

III. PRELIMINARIES

Path planning is essential for robot autonomous navigation and involves calculating a trajectory for a robot to follow in a map, between a start and end point. The solution of



(a) $k = 0$ (b) $k = 300$ (c) $k = 700$ (d) $k = 1000$

Fig. 3: Denoising diffusion steps ($k = 1000$, $\text{path}_l = 200$) to generate a path from noisy samples.

path planing refers to the generation of an optimal path, that full-fills the properties of finding the collision free, shortest, and smooth route between start and goal positions [18]. As elaborated in Sec. II, there are plethora of approaches to solve path planing. For relevance to our method we expand on Probabilistic Diffusion-based path planing, leveraging the learning capabilities of CNNs.

Diffusion

Image-guided diffusion models [19] have emerged as a powerful generative model with impressive performance when dealing with image datasets, among others. They provide the ability to transform a latent encoded representation into a more meaningful description of the image data. A popular variation is the Denoising Diffusion Probabilistic Model (DDPM), a generative model defined through parameterized Markov chains trained using variations inference. A forward chain converts input data into noise and a reverse chain converts the noisy data back to its original form. In particular, the noisy data is generated by transforming the data distribution into a Gaussian distribution. Then, the denoising occurs by learning transition kernels parameterized using deep neural networks, for reversing the noisy data back to the input [20]. The learned denoising kernel $p_\theta(x_{t-1} \mid x_t)$ is parameterized by a prior Gaussian distribution $p(x_T) = \mathcal{N}(x_T :, 0, I)$. Thus, the kernel can be defined by

$$\mathrm{p}_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right) \qquad (1)$$

where, θ represents the model parameters, $\mu_\theta(x_t, t)$ the mean and $\Sigma_\theta(x_t, t)$ the variance, parameterized by the deep neural network. A popular deep neural network choice for image conditioned diffusion are CNNs due to their benefits in dealing with image datasets.

Convolutional Neural Networks (CNNs)

CNNs comprise a class of deep artificial neural networks, dominant in computer vision tasks. CNNs are designed to learn spacial hierarchies of features, from low- to high-level patterns. The main building blocks of CNNs are the convolution, pooling (for feature extraction of the input images), and fully connected layers (which map the extracted features into the final output), commonly using classification [21].

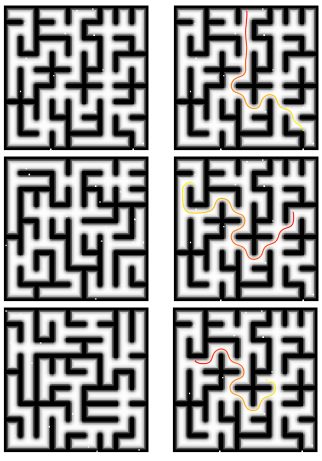
IV. METHOD

Our method is inspired by the baseline papers [8], [10]. We adapt the image conditioned diffusion pipeline [8] to solve the problem of mobile robot path planing, while also conditioning for the starting and goal position of the trajectory based on [10]. Initially the training dataset is generated, comprising of the 100x100 sized random and solvable maps and a number of end-to-end trajectories for each map (Sec. IV-A). This is tenfold larger and more complicated dataset than the one provided in the baseline paper [10]. The dataset is fed into the training pipeline (Sec. IV-B), where the optimal trajectories are learned through demonstration by preserving local consistencies. The inference pipeline (Sec. V-A) allows the generation of the optimal trajectory given start and goal positions and the relevant map.

A. Data generation

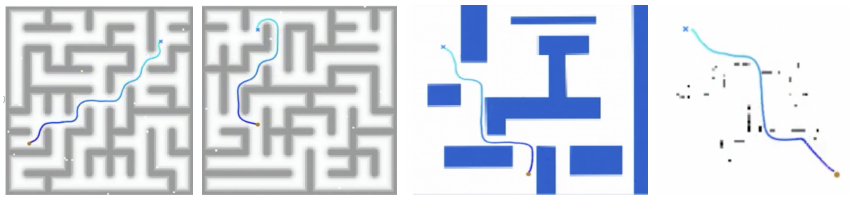
Creating the training dataset includes generating random map images and feasible end-to-end 2D trajectories. Examples of the randomly generated maps and end-to-end trajectories are depicted in Fig. 4.

1) Map generation: Map generation is done through Kruskal's Minimum Spanning Tree (MST) Algorithm [22], with the edges representing potential wall locations and the nodes representing cells. The algorithm works by initially considering all edges of a randomly weighted graph and sorting them by their weights. Then, it iteratively adds edges to the MST, starting with the smallest weight, while ensuring that the graph remains acyclic. This process continues until all vertices are in the MST or the desired number of edges



(a) Maps (b) Trajectories

Fig. 4: Generated samples from the dataset: 4a) examples of 100×100 random solvable maps and 4b) examples of end-to-end trajectories, generated through A^*.



(a) (b) (c) (d)

Fig. 5: Validating DiPPeR's performance and generalization. A random point is selected for the start and goal position on the provided map. Maps a) and b) are part of the validation dataset to validate the performance of the network in connecting the starting and end points while also avoiding the obstacles. Maps c) and d) are used to test the ability of the network to generalize to different out-of-distribution environments of varying scale, color and obstacle structure.

is reached. Kruskal's algorithm employs disjoint-set data structures to efficiently detect and avoid creating cycles during edge selection, resulting in a tree that spans all vertices with the minimum possible total edge weight.

2) Path generation: To provide trajectories in the training framework, the generate paths need to be *feasible* - avoid all obstacles. To generate feasible paths we use the A^* path finding algorithm that uses a combination of heuristic estimates and cost information to efficiently find the shortest path between the start and end nodes. We randomize the position of the start and goal node to generate a variety of trajectories per map.

B. Training Framework

We formulate DiPPeR as a vision-guided mobile robot path planner generated through DDPMs, as presented in Sec. III. Our observation space O comprises of the 100×100 pixel images, representing the randomly generated maps. Our action space A_t comprises of 2D trajectories for each map. The subscript t refers to a sequence of timesteps, with $t = T$ representing the training horizon. We empirically conclude that a dataset of $10,000$ maps with 100 trajectories each, is sufficiently large to achieve generalization and adaptability to unseen maps. Fig. 2 provides a graphical representation of the proposed training framework.

DDPM Training: DDPM in our method, is used to approximate the conditional distribution $p(A_t \mid O)$ of the action vector A_t, given the map image observation O. This formulation speeds up the diffusion process and improves the generated actions by predicting an action conditioned to an observation, which translates to predicting trajectories given the specific map image.

DDPM takes as input A_t^k with added noise e^k, sampled from the prior Gaussian distribution and performs K denoising iterations $(A_t^{k-1}, A_t^{k-2}, ..., A_t^0)$ through gradient descent, following Eq. 2. The output is the noise free representation of the input vector A_t^0:

$$A_t^{k-1} = \alpha(A_t^k - \gamma\epsilon_\theta(O, A_t^k, k) + \mathcal{N}(0, \sigma^2 I)), \quad (2)$$

where ϵ_θ represents the noise prediction network. The variables α, γ, σ and e^k, when expressed as functions of k compose the noise schedule that drives the learning process. The hyperparameters α, γ, σ determine the scheduling learning rate which controls the extent to which the diffusion policy captures high and low-frequency characteristics of action signals.

Training ϵ_θ, involves predicting the noise added to a random sample A_t^0, through Eq. 2. For each A_t^0 a denoising iteration k is selected with an added corresponding noise value e^k and variance. The mean squared error between the e^k and the predicted noise value from ϵ_θ is then calculated based on Eq. 3, with the aim to be minimized along the gradient descent.

$$\mathcal{L} = MSE(\epsilon_k, \epsilon_\theta(O, A_t^0 + \epsilon_k, k)) \quad (3)$$

Unlike [8], [10], we do not perform receding predictive horizon and do not implement a controller to translate the policy's output action to robot action. DiPPeR actions can be directly implemented to the real robot to find the optimal end-to-end trajectory given the map of the environment. We observe that varying the horizon length during training has a significant impact in the performance of the model. The generated trajectories in our dataset have variable length from 10 to 400. The horizon length should be long enough to capture the whole range of the dataset, hence we set it equal to 180 which is the estimated average trajectory length.

An important design choice is selecting the architecture of ϵ_θ. We chose a CNN due to it's capabilities of dealing with image datasets.

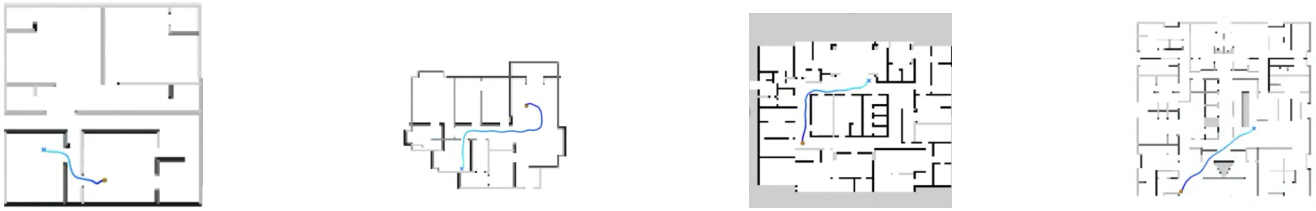
DiPPeR: We develop two variations of DiPPeR$_{cnn}$. The first version has observation space O as defined in Sec IV-B. The second version adds two extra terms to O, the trajectory start and end points, each expressed as a 2D vector corresponding to the x and y pixel coordinates as proposed by [10]. A 1D temporal CNN is used with conditioning the actions generation on the observations by $p(A_t \mid O)$. The conditioning occurs through Feature-wise Linear Modulation (FiLM).

Visual Encoder: A ResNet-18 visual encoder with spacial softmax pooling is trained end-to-end to convert the observation image O to a latent embedding o while preserving spatial information. The ResNet is trained alongside ϵ_θ.

V. RESULTS

A. Inference Pipeline

After training, the inference pipeline is used to validate DiPPeRs' performance. A start and goal position are ran-



(a) 280 × 280 (b) 360 × 360 (c) 600 × 600 (d) 760 × 760

Fig. 6: Maps used for comparing different planning methods with their respective sizes. The blue depicted line corresponds to the DiPPeR generated trajectory.

domly sampled from a uniform distribution and are then passed into ϵ_θ, alongside O. A $path_l$ variable defines the number of noisy samples used during the denoising diffusion process. The value of path$_l$ is defined as function of the approximate length of the estimated trajectory, i.e. for start and goal positions being further apart, the $path_l$ will be larger and vice-versa. A vector of noise sampled from the prior Gaussian distribution with length equal to path$_l$ and 2 dimensions (x and y pixel coordinates), is fed in the DiPPeR. The reverse chain of the DDPM model is used to iteratively denoise the input vector. The output is the final trajectory A_0, connecting the start and goal points, while aiming to follow a feasible path. The progress of denoising during inference is depicted in Fig. 3.

B. Simulation Results

The evaluation of DiPPeR's performance is completed in two stages. *Performance* evaluation is performed by sampling map images from the validation dataset and *Out-of-distribution* evaluation which is performed by selecting unseen map images of varying scale, color and obstacle structure to test DiPPeR's generalization capabilities. In both cases a random start and goal position is selected along the map and inference is performed following Sec. V-A. For all experiments the number of diffusion iterations is empirically chosen as $k = 1000$, to ensure both full convergence and minimum inference time. In most cases convergence is achieved with a smaller k however we decide to keep it constant to preserve uniformity across experiments. The path length path$_l$ parameter has a significant impact on the diffusion performance and needs to be varied according to the desired trajectory length. Given the start and goal position and the structure of the map, the number of feasible pixels can be measured to create an approximate estimation of path$_l$. Significantly larger path$_l$ will results in trajectories that loops locally and significantly smaller path$_l$ result in trajectories going through obstacles to connect the start and end goal.

The chosen evaluation metric is the success rate. Success is achieved when the output trajectory A_0 connects the start and goal points while following a feasible path. Some examples of achieved successful trajectories are shown in Fig. 5. We evaluate DiPPeR in 10 images from the validation set and 10 out-of-distribution maps. For each map we generate 10 random start and goal positions and repeat inference 3 times, to ensure consistency of the output. The success rate is calculated by dividing the number of successful experiments by the total number experiments and is then converted to a percent value. The start and goal position conditioned version of the DiPPeR outperformed the non-conditioned one in all tests and we set it as our default DiPPeR version.

The percentage success rate $\%sr$ for DiPPeR is presented in Table I.

Dataset	$\%sr$
Validation	79
Out-of-Distribution	81
Average	80

TABLE I: Inference success rate for validation and out-of-distribution datasets.

DiPPeR is on average 80% successful in providing feasible paths. It performs best on out-of-distribution maps as they contain instances of maps much simpler than the training dataset. To understand the failure cases, we plot the success rate against the trajectory length (Fig. 7). The success rate drops for both extremes of smaller and larger trajectories and performs the best for trajectories of length close to 180. This is expected as the choice of horizon length is set equal to 180 during training, however it presents a limitation that we aim to address in future work.

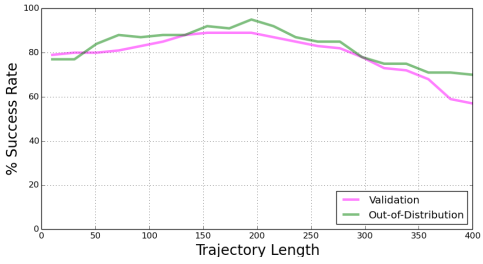


Fig. 7: Plot of the average percent success rate against the trajectory length for maps sampled from both the Validation and the Out-of-Distribution Dataset

To assess the performance of DiPPeR against SOTA path planing frameworks, we evaluate its convergence speed against a sample-based planner A^* and a data driven planner N-A^*. We compare the algorithms by using maps of variable size and obstacle structure depicted on Fig. 6. For each map

Fig. 8: Real World Deployment: Spot (top) and Go1 (bottom) navigating around a maze environment using DiPPER in combination with the developed navigation stack (Fig. 9).

10 trajectories with random start and end points are generated by the 3 algorithms. The time taken for trajectory generation is measured for all experiments and the 10 values for each map are averaged. The results are summarized in Table II.

maps	DiPPeR	N-A^*	A^*
(a)	**0.4**	4.70	6.03
(b)	**0.4**	14.73	17.51
(c)	**0.4**	5.17	15.59
(d)	**0.4**	75.20	84.84

TABLE II: Average time in seconds taken for trajectory generation by DiPPeR, N-A^*, A^*. Maps a)-d) are presented in Fig. 6.

The maps for comparison are sampled from the MRPB benchmark dataset [23].

DiPPeR is on average 70 faster against the compared SOTAs algorithms, with feasible trajectory generation taking only $0.4s$ regardless of maze size or trajectory length. This is due to the properties of diffusion and due to the design choice to normalize images within the pipeline.

C. Real-World Deployment

A schematic representation of the real work deployment of DiPPeR is depicted in Fig. 9. We validate the performance of DiPPeR in the real world and its platform agnostic property through deployment on Unitree Go1[1] and Boston Dynamics Spot[2].

In order to leverage the existing robot navigation frameworks, DiPPeR is integrated with the 2D ROS navigation Stack[3], to acts as a global path planner. Given the occupancy map, DiPPeR generates a global path which is further refined by the local planner - to avoid violation of the robots kinodyanmics constraints and an external tracking system - to mitigate for state estimation inaccuracies. The local planner used is the Timed-Elastic-Band (TEB) [24], [25] and the external tracker of choice is Phasespace tracking cameras[4]. Phasespace cameras allow for 960 Hz robot real-time localization. Examples of successful deployment of the pipeline can be seen in Fig. 8.

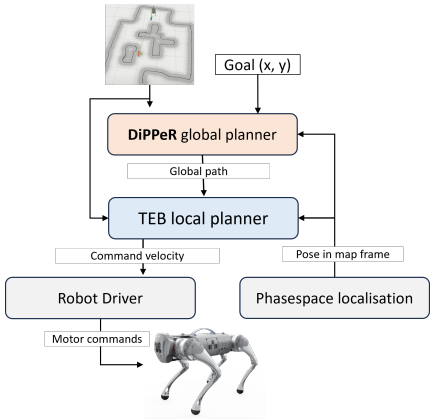


Fig. 9: Schematic structure of the navigation stack for DiPPeR real robot deployment.

VI. CONCLUSION

In this paper we present DiPPeR, an image guided diffusion based 2D-path planner. The planner is successful in generating feasible paths of variable length on average 80% of the time, for maps of various size and obstacle structure. DiPPeR outperformed in speed against both search based and data driven planner by a factor of 70. We validate the transfer of the planner in the real world and showcase its platform agnostic capabilities by successfully testing it on two different robots. We identified that DiPPeR tends to performs less optimally for trajectories significantly longer than the training horizon. We are aiming to address this in future work by also experimenting with different network architectures, such as transformers, that show promising results in the field of diffusion.

VII. ACKNOWLEDGMENTS

We would like to thank Davide Paglieri for sharing his insights regrading the diffusion pipeline training.

[1]https://www.unitree.com/en/go1/
[2]https://www.bostondynamics.com/products/spot
[3]http://wiki.ros.org/navigation
[4]https://www.phasespace.com/

REFERENCES

[1] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[2] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," in *Conference on Robot Learning*, 2021.

[3] Z. Xie, X. Da, B. Babich, A. Garg, and M. van de Panne, "GLiDE: Generalizable Quadrupedal Locomotion in Diverse Environments with a Centroidal Model," in *Algorithmic Foundations of Robotics XV*. Springer International Publishing, 2023, pp. 523–539.

[4] C. Lau and K. Byl, "Smooth RRT-Connect: An Extension of RRT-connect for Practical Use in Robots," in *IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, 2015, pp. 1–7.

[5] Y. Zhang, H. Jiang, X. Zhong, X. Zhong, and J. Zhao, "MI-RRT-Connect Algorithm for Quadruped Robotics Navigation with Efficiently Path Planning," *Journal of Physics: Conference Series*, vol. 2402, no. 1, p. 012014, 2022.

[6] S. Choudhury, M. Bhardwaj, S. Arora, A. Kapoor, G. Ranade, S. Scherer, and D. Dey, "Data-Driven Planning via Imitation Learning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1632–1672, 2018.

[7] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki, "Path planning using neural a* search," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 12 029–12 039. [Online]. Available: http://proceedings.mlr.press/v139/yonetani21a.html

[8] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[9] 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 preprint arXiv:2308.01557*, 2023.

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

[11] H. Huang, Y. Li, and Q. Bai, "An improved A* Algorithm for Wheeled Robots Path Planning with Jump Points Search and Pruning Method," *Complex Eng Syst*, vol. 2, p. 11, 2022.

[12] M. Kusuma, C. Machbub, *et al.*, "Humanoid Robot Path Planning and Rerouting Using A-Star Search Algorithm," in *2019 IEEE International Conference on Signals and Systems (ICSigSys)*. IEEE, 2019, pp. 110–115.

[13] 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)*. IEEE, 2017, pp. 1527–1533.

[14] B. Ichter and M. Pavone, "Robot Motion Planning in Learned Latent Spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

[15] L. Lee, E. Parisotto, D. S. Chaplot, E. Xing, and R. Salakhutdinov, "Gated Path Planning Networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2947–2955.

[16] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki, "Path planning Using Neural a* Search," in *International conference on machine learning*. PMLR, 2021, pp. 12 029–12 039.

[17] S. Hong, J. Lu, and D. P. Filev, "Dynamic diffusion maps-based path planning for real-time collision avoidance of mobile robots," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 2224–2229.

[18] S. G. Tzafestas, "11 - mobile robot path, motion, and task planning," in *Introduction to Mobile Robot Control*, S. G. Tzafestas, Ed. Elsevier, 2014, pp. 429–478. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124170490000110

[19] H. Ali, S. Murad, and Z. Shah, "Spot the fake lungs: Generating synthetic medical images using neural diffusion models," in *Artificial Intelligence and Cognitive Science*, L. Longo and R. O'Reilly, Eds. Cham: Springer Nature Switzerland, 2023, pp. 32–39.

[20] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion Models: A Comprehensive Survey of Methods and Applications," *arXiv e-prints*, p. arXiv:2209.00796, Sept. 2022.

[21] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, pp. 611–629, 2018.

[22] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[23] J. Wen, X. Zhang, Q. Bi, Z. Pan, Y. Feng, J. Yuan, and Y. Fang, "MRPB 1.0: A unified benchmark for the evaluation of mobile robot local planning approaches," in *IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 8238–8244.

[24] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; German Conference on Robotics*. VDE, 2012, pp. 1–6.

[25] ——, "Efficient Trajectory Optimization Using a Sparse Model," in *European Conference on Mobile Robots*. IEEE, 2013, pp. 138–143.