FlowNav: Learning Efficient Navigation Policies via Conditional Flow Matching

Samiran Gode*
Department of Engineering
University of Technology Nuremberg

Abhijeet Nayak*
Department of Engineering
University of Technology Nuremberg

Wolfram Burgard
Department of Engineering
University of Technology Nuremberg
*These authors contributed equally to this work.

Abstract: Effective robot navigation in dynamic environments is a challenging task that depends on generating precise control actions at high frequencies. Recent advancements have framed navigation as a goal-conditioned control problem. Current state-of-the-art methods for goal-based navigation, such as diffusion policies, either generate sub-goal images or robot control actions to guide robots. However, despite their high accuracy, these methods incur substantial computational costs, which limits their practicality for real-time applications. Recently, Conditional Flow Matching (CFM) framework has emerged as a more efficient and robust generalization of diffusion. In this work, we explore using CFM to learn action policies that help the robot navigate its environment. Our results demonstrate that CFM can generate highly accurate robot actions. CFM not only matches the accuracy of diffusion policies but also significantly improves runtime performance. This makes it particularly advantageous for real-time robot navigation, where swift, reliable action generation is vital for collision avoidance and smooth operation. By leveraging CFM, we provide a pathway to more scalable, responsive robot navigation systems capable of handling the demands of dynamic and unpredictable environments.

Keywords: Navigation, Diffusion, Continuous Normalizing Flows, Flow Matching

1 Introduction

Mobile robots need fast action control outputs especially when dealing with dynamic obstacles and for improved utility. While diffusion-based policies [1] for navigation [2] are effective for generating multi-modal data they struggle when actions need to be quick. Recent work in training Continuous Normalizing Flows (CNF) [3] using flow matching [4, 5, 6, 7] has proven to be quite successful in other modalities and provides much faster inference times [7]. In our work, we show that using the CFM framework instead of a diffusion-based policy for navigation improves the inference times significantly (8x) while providing the same level of accuracy.

Navigation for mobile robots is an important and well-studied problem that requires a robot to move from it's current state to a goal state while avoiding obstacles and following a close-to-optimal path. Robots first perceive the environment to find areas of interest as well as to map and localize within their environment [8]. Then, based on optimizing some predetermined objective, the robot plans a path to its target destination on the map and acts optimally to move along the planned path. This technique has demonstrated success in various challenging scenarios [9] and has been deployed in

numerous applications. Despite these successes, mobile robots haven't been as widely deployed as one would have thought because of issues related to reliability, cost, safety, etc.

Recent work [2, 10, 11] tries to solve this problem by training models across different embodiments and environments with large amounts of data. The aim of such an approach is to replicate the successes seen in different modalities such as language and be able to train a visual navigation foundation model [11]. Further work [2] tries to improve on this by using diffusion policies [1] for action generation. Diffusion policy is a great choice because it provides the ability to train multimodal action distributions and provides stable training[7].

Diffusion models have proven to be effective across multiple modalities [7, 12, 13, 14, 15] for data generation. They approximate a stochastic differential equation that transforms the Gaussian distribution into the desired distribution. Recent work [4, 5, 6, 7] has shown that CNF trained with a flow matching framework leads to high-quality samples and the source distribution can be arbitrary and does not necessarily have to be Gaussian. CNF can be trained with a regression over the drift of the ODE and provides a stable objective similar to diffusion. CFM can use straighter flows which can be integrated in fewer steps to get the output.

In the rest of the paper, we first introduce related work and talk about classical and learning-based navigation. Next, we give some background on Denoising Diffusion Probabilistic Models (DDPM) [12] and how it is used in diffusion policy [1] and NoMaD [2]. We then present the CFM framework and explain how we use it for training our navigation policy. Next, we explain our experimental evaluations and their results and show that FlowNav is as effective as the state of the art while being considerably faster.

2 Related Work

Robot navigation has been a central research area in robotics for decades, with various approaches proposed to address the challenge of guiding robots through complex environments. Traditional methods for navigation often relied on geometry-based algorithms such as Simultaneous Localization and Mapping (SLAM) [8, 16] and path planning techniques like Dijkstra's and the A* algorithm [17, 18]. These methods typically involved constructing explicit maps of the environment and planning paths based on the geometry of obstacles. However, these classical approaches struggle to generalize across environments and become computationally expensive when dealing with dynamic obstacles or unstructured terrains.

In recent years, there has been a shift toward learning-based methods [19] for robot navigation, driven by advances in machine learning and the availability of large-scale data. One of the most influential directions in this space is end-to-end learning, where deep neural networks are trained to directly map sensory inputs, such as images, to actions that control the robot. A significant body of work [2, 10, 11, 20] in learning-based methods focuses on general goal-conditioned vision-based navigation. For instance, GNM [10] introduced the idea of training a single model on diverse data collected across various robots. This model used a normalized action space, which allowed zero-shot generalization to new robot embodiments.

Parallelly, advancements in generative modeling, particularly diffusion models [12, 21], have influenced robot navigation. For example, ViNT [11], which builds upon GNM by leveraging transformers to process visual input, uses a diffusion policy to generate sub-goal images. The idea of using sub-goal images enables kilometer-scale navigation. However, sub-goal image generation can be computationally expensive.

To do away with sub-goal image generation, diffusion models have been adapted to generate action policies instead [1]. NoMaD [2] addresses this by using a diffusion policy to directly learn actions, rather than generating sub-goal images. NoMaD also introduces a novel goal-masking idea to train a single network for both goal-directed and exploratory navigation.

While diffusion-based approaches have proven to be accurate in generating actions for navigation, by virtue of being stochastic differential equations they require many passes through the network [7]. This can lead to significant computational overhead, making them less suitable for real-time applications where efficiency is critical. In contrast, the emergence of CFM [4, 5, 6, 7], which generalizes diffusion, offers a promising alternative. CFM with simpler flows [7] eliminates the need for many iterative denoising steps by learning a continuous flow that matches the target distribution. This makes CFM a potentially more robust and computationally efficient way to generate action policies, particularly in dynamic and time-sensitive environments.

3 Background

We provide an introduction to DDPM [12] which is the backbone of Diffusion Policy [1]. While diffusion policies were used for object manipulation, NoMaD [2] extended the use of the diffusion policy for action generation for robot navigation. We also provide a brief overview of NoMaD in this section. In the later sections, we explain how we use CNF trained with CFM to generate actions for robot navigation. Additionally, we draw parallels between CFM and DDPM and postulate why using CFM is better.

3.1 Denoising Diffusion Probabilistic Models

Assume we have a set of n examples $X = \{x_1, x_2, \ldots, x_n\}$ which are drawn from a probability distribution $p(x)$. We want to be able to sample from $p(x)$. We will later also extend this problem to sampling from $p(x \mid z)$ for some latent variable z, but for now let us limit it to $p(x)$. One way to sample from this distribution is by using a denoising process [12], where noise is gradually removed from an initial random sample to recover a clean sample that follows the target distribution. This denoising process is also called Stochastic Langevin Dynamics [22].

DDPM operates by initially sampling from a Gaussian distribution and then progressively denoising the sample using a learned noise prediction model, until it reaches a sample from the desired distribution. Assuming we need to denoise for k iterations, we first sample some x_j^k from a Gaussian distribution $q(x)$. Given a trained model $\varepsilon_\theta(x_j^k, k)$, with trainable parameters θ, which predicts the noise added in the k-th step, we denoise the sample according to the equation 1.

$$x^{(k-1)} = \alpha \left(x^k - \gamma \varepsilon_\theta(x_j^k, k) + N(0, \sigma^2 I) \right) \tag{1}$$

where the predicted noise is removed while also adding a small amount of Gaussian noise, which helps maintain the stochastic nature of the diffusion process.

To train a model for this denoising process, for each sample $x_i^0 \in X$, an iteration k is randomly selected along with the corresponding noise ε^k. We then add this noise to the sample x_i^0 to obtain a noisy sample x_i^k. Next, we need to predict the noise at the k-th step, which can be achieved using equation 2.

$$L = \text{MSE}(\varepsilon^k, \varepsilon_\theta(x_j^k, k)). \tag{2}$$

This loss function has been shown to minimize the KL-Divergence between the generated data distribution and the sample data distribution [12].

3.2 Diffusion Policy

The process of sampling from $p(x)$ can also help us sample from $p(x \mid z)$, where z is a latent variable. This latent variable could be CLIP [23] embeddings which are used for text to image generation or in the case of the diffusion policy [1], embeddings from the robot observations. More specifically, the distribution from which we want to sample is $P(A_t \mid O_t)$, where A_t and O_t are the action and observation representations at time t. The policy takes a observation horizon of T_o steps and has a prediction horizon of T_p steps and an action execution horizon of T_a steps. The

only change required in equation 2 is that the noise prediction network now also processes the observations as shown in equation 3.

$$L = \text{MSE}(\varepsilon^k, \varepsilon_\theta(O_t, A_t^0 + \varepsilon^k, k)) \tag{3}$$

3.3 Navigation with Goal Masked Diffusion (NoMaD)

NoMaD [2] uses the diffusion policy to model task-agnostic exploratory actions as well as goal-directed actions. The novelty in NoMaD was that a single model was trained in an end-to-end manner to handle both, goal-directed and exploratory navigation. This is made possible by updating the observation representation that is passed to the diffusion policy by using the provided goal image, while also including an observation horizon for exploratory navigation. NoMaD also makes training on datasets from different robots easier by following the action normalization idea presented in [10]. This way, the model is trained on diverse datasets, with the only requirements that the dataset contains frontal RGB images along with trajectories of the robot in its environment.

4 Methodology

We use the CFM framework [7] to train our action generation model. Conditional flow matching is a simulation-free training-objective to train continuous normalizing flows [3].

4.1 ODEs and Probability Flows

We define a smooth time varying vector field $u : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ defines an ordinary differential equation 4:

$$dx = u_t(x)\, dt \tag{4}$$

The solution for this ODE is represented as ϕ. We define $\phi(0)$ as x, which means x_0 moves based on velocity u_t, and $\phi(t)$ is the point it is moved to at time t. We want to be able to learn the velocity $u_t(x)$ that moves from an initial distribution $q_0(x)$ to the desired distribution $q_1(x)$. As a result, this will allow us to have a mapping that enables us to sample from $q_1(x)$.

4.2 Flow Matching Objective

At training time, we sample t, x_t, and u_t given some sample $x_1 \sim q_1(x)$. The time t is sampled from $\mathcal{N}(0, 1)$, while x_t and u_t can be sampled in multiple ways. We use the strategy mentioned in [6, 7] and is shown in equations 5 and 6:

$$p_t(x \mid z) = \mathcal{N}(x \mid tx_1 + (1 - t)x_0, \sigma^2) \tag{5}$$

$$u_t(x \mid z) = x_1 - x_0 \tag{6}$$

We however use $\sigma = 0$ and thus our x_t becomes

$$x_t = x_1 + (1 - t)x_0 \tag{7}$$

Given trainable parameters θ, we define a time dependent vector field $v_\theta(x, t)$ that regresses to u using equation 8 as given in [7]:

$$\mathbb{E}_{t,q(z),p_t(x|z)} \left\| v_\theta(x, t) - u_t(x|z) \right\|^2 \tag{8}$$

4.3 FlowNav Architecture

Our architecture is based on the architecture used in [2]. We use the current observation, ${}^i o$, with $T_o = 5$ past observation images, that is ${}^{(i-T_o)}o : {}^{(i-1)}o$, and process them through an image encoder to generate image tokens. Additionally, tokens are generated from the goal image o_g depending on whether the model is being trained for the goal-directed or exploratory case. These tokens are then

passed to a transformer which creates the observation context embeddings ${}^{i}c$. We now use two prediction heads: the temporal distance head f_d which predicts the temporal distance between ${}^{i}o$ and o_g and the flow prediction network that predicts v_θ. Both the prediction heads are trained jointly in a supervised manner. The overall architecture is shown in 1

Our main objective is to sample actions ${}^{i}a_t$, where t is the time in $[0, 1]$ that is used for sampling in the CFM framework and i denotes the action at the ith robot step. We need these actions to also be conditioned on the observation context embeddings ${}^{i}c$. As a result, the flow matching objective in equation 8 is transformed into the loss function shown in equation 9 that we use to train our network.

$$\mathcal{L} = \mathbb{E}_{t,{}^{i}a_0,{}^{i}a_1,{}^{i}c} \left\| v_\theta({}^{i}a_t, {}^{i}c, t) - u_t \right\|^2 + \lambda \cdot \text{MSE}(d({}^{i}o, o_g), f_d({}^{i}c)) \tag{9}$$

where ${}^{i}a_0$ is from a tractable distribution and ${}^{i}a_1$ is from the set of real robot motions.

During inference, we first sample $v_0 \sim \mathcal{N}(0, 1)$. An ODE solver is employed to update v_0 based on the learned $v_\theta({}^{i}a_t, {}^{i}c, t)$. Although various solvers, such as RK-4, Dopri-5, etc are available, we find that the first-order Euler update yields the best performance for our application. For the action prediction, we use a prediction horizon of $T_p = 8$ which means that we have a set of T_p normalized waypoints with respect to the ego at $(0, 0)$. A robot-specific controller can be used to un-normalize these waypoints to obtain embodiment-specific controls out of which T_a controls can be executed.

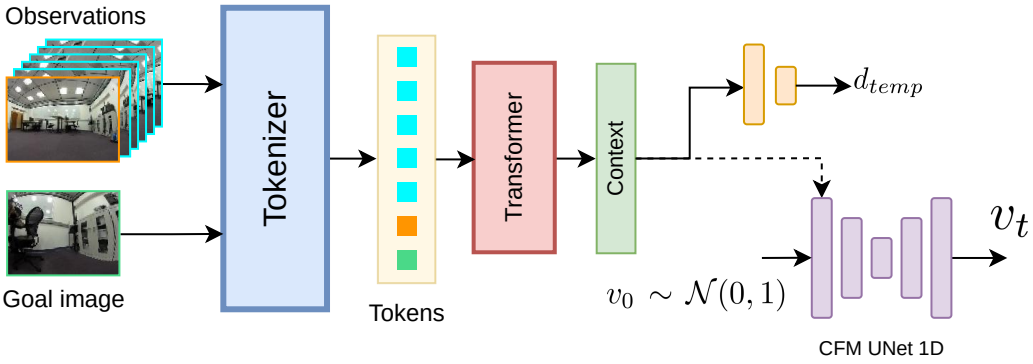

Figure 1: Architecture for our FlowNav approach. We follow the architecture used in [2]. The current and past observations, along with the goal image are tokenized using image encoders. These tokens are processed by a transformer to create the observation context embedding c_t. The context is then used as an input to the temporal distance prediction network f_d and the flow prediction network v_θ.

5 Experimental Evaluation

5.1 Training Strategy

We use the NoMaD [2] repository as our base software. We update the training and evaluation pipelines to use CFM instead of diffusion policies. We implement this by using the TorchCFM [7, 24] library. Our model is trained end-to-end in a supervised manner using the loss function shown in equation 9.

As we only modify the action generation part of NoMaD, we do not report results for the evaluation of the temporal distance network. We train the model for 20 epochs with a batch size of 256 and a learning rate of $1e^{-4}$. We use the Go Stanford (GS) [25] and RECON [26] datasets to create the train and test splits. These datasets contain a set of trajectories of the robot in an environment and the captured frontal RGB images. A part of a trajectory is selected as the observation horizon, while a way-point in the future is chosen as the goal location.

During the evaluation, we use the NoMaD [2] diffusion policy as our baseline. To compare inference times, we pass a batch of inputs through the network and measure the time taken to generate actions for the entire batch. Additionally, we evaluate the cosine similarity between the ground-truth actions and the actions generated by each algorithm.

Table 1: Comparing action accuracies on the GS and RECON evaluation splits. Accuracies can range between 0 and 1. We observe that CFM reaches comparable accuracies in a fewer number of steps when compared to diffusion policies.

	k-steps	1	2	4	8
GS	NoMaD	**0.70**	0.73	0.87	**0.95**
	Ours	0.41	**0.92**	**0.92**	0.92
RECON	NoMaD	**0.68**	0.72	0.84	0.94
	Ours	0.42	**0.95**	**0.94**	0.94

Table 1 shows the results for action accuracies for the GS and RECON evaluation splits. For the diffusion policy, we observe that action accuracy steadily improves with an increasing number of diffusion steps, reaching accuracies of 0.95 and 0.94 on the two datasets after eight diffusion steps. This is expected since diffusion policies are designed to progressively de-noise the initial input, leading to a cleaner, noise-free output over time. When using CFM for action generation, we observe that it achieves accuracies of 0.92 and 0.95 on the two datasets within just two steps. This indicates that CFM reaches comparable action accuracy in significantly fewer steps compared to the diffusion policy.

Table 2: Comparing inference times (in ms) on a batch of inputs. We observe that CFM is faster than diffusion, regardless of the number of evaluation steps for each algorithm.

k-steps	1	2	4	8
NoMaD	5.4	10.6	20.9	41
Ours	**0.4**	**5.6**	**15.6**	**36**

Table 2 compares the inference times (in ms) of both algorithms. We observe that CFM is faster than diffusion, regardless of the number of steps each algorithm is evaluated for. Consequently, when we combine our findings, we find that CFM reaches similar action accuracy in fewer steps compared to the diffusion policy. This indicates that CFM reaches comparable action accuracy in significantly fewer steps than the diffusion policy, leading to an 8x improvement in inference times. This reduction in run-time makes CFM potentially more suitable for real-world deployment.

Figure 8 presents the frontal RGB observations and goal images for two episodes, alongside plots comparing the ground truth actions with the predicted actions for each of these episodes. The initial position of the robot (assumed to be at (0, 0)) is marked by the green "x" while the goal location is indicated by the red "x". The ground truth action is shown in green. We compare the ground truth action with the action generated by the diffusion policy run for eight steps (represented by the dotted blue line) and the action generated using CFM that uses two euler steps (shown as the solid magenta line). In the plots for both episodes, we observe that CFM generates actions that bring the robot close to the goal location. Additionally, the actions generated by CFM outperform those produced by the diffusion policy. When combined with our results on inference times, this suggests that CFM offers a more efficient way to generate actions and control the robot effectively.

6 Conclusion and Future Work

In this paper, we demonstrated that Conditional Flow Matching provides advantages in the context of learning action policies for robot navigation. We also show that CFM achieves a comparable

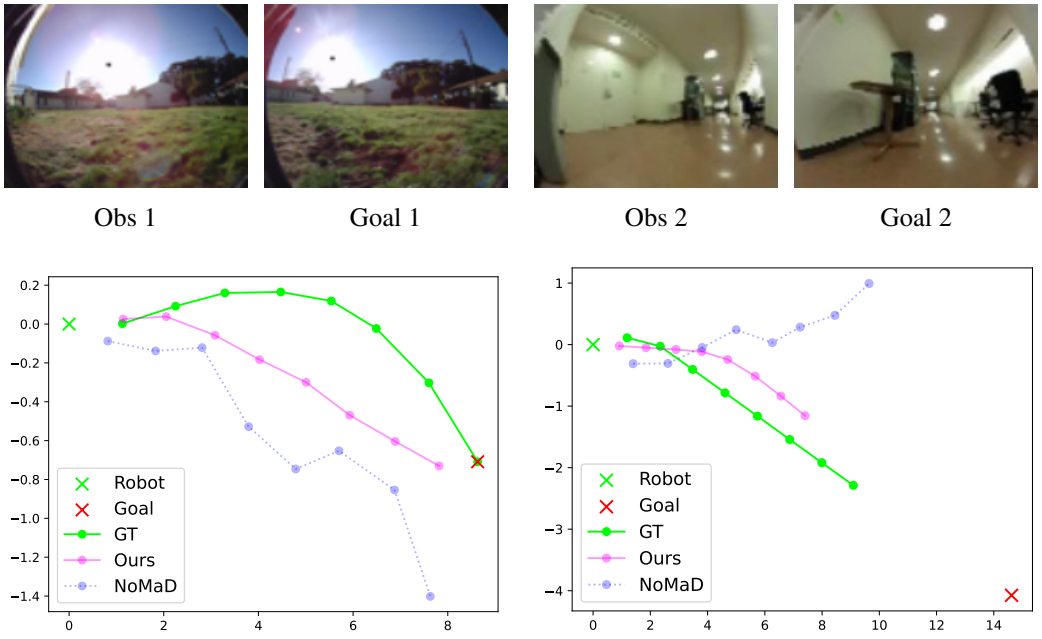

Figure 8: The top row shows the frontal RGB observation and goal images of two episodes from the RECON and the GS datasets. The bottom row shows the plots that compare the ground truth action and the generated actions. We compare our method (using CFM) with NoMaD, which uses a diffusion policy for action generation. We observe that in both episodes, our method learns to generate better actions that take the robot closer to the goal location.

action accuracy when compared to diffusion policies. Additionally, we show that diffusion policies and CFM achieve similar action accuracies in eight and two steps respectively, leading to an eight times faster inference when using CFM. The faster inference time translates to faster sensory input processing. This is especially advantageous in navigation where there is a high likelihood of dynamic objects in the environment. In future work, we would like to deploy our approach on a real-world robot to analyze the real-time advantages ensured by our approach. Another direction we want to focus on is the use of off-policy reinforcement learning to learn an optimal trajectory for goal-based exploration that the robot learns over multiple explorations of a region of the map.

References

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[2] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[3] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[4] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *International Conference on Learning Representations (ICLR)*, 2023.

[5] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *International Conference on Learning Representations (ICLR)*, 2023.

[6] Q. Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint 2209.14577*, 2022.

[7] A. Tong, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, K. Fatras, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

[8] S. Thrun, W. Burgard, and D. Fox. Probabilistic robotics cambridge, 2005.

[9] R. Behringer, S. Sundareswaran, B. Gregory, R. Elsley, B. Addison, W. Guthmiller, R. Daily, and D. Bevly. The darpa grand challenge-development of an autonomous vehicle. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 226–231. IEEE, 2004.

[10] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.

[11] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023.

[12] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[13] P. Dhariwal and A. Nichol. Diffusion models beat GANs on image synthesis. *Neural Information Processing Systems (NeurIPS)*, 2021.

[14] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. Structured denoising diffusion models in discrete state-spaces. *Neural Information Processing Systems (NeurIPS)*, 2021.

[15] G. Corso, H. Stärk, B. Jing, R. Barzilay, and T. Jaakkola. DiffDock: Diffusion steps, twists, and turns for molecular docking. *International Conference on Learning Representations (ICLR)*, 2023.

[16] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *Aaai/iaai*, 1999(343-349):2–2, 1999.

[17] J.-C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[18] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[19] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE, 2018.

[20] D. Shah and S. Levine. Viking: Vision-based kilometer-scale navigation with geographic hints. *arXiv preprint arXiv:2202.11271*, 2022.

[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[22] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[23] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[24] A. Tong, N. Malkin, K. Fatras, L. Atanackovic, Y. Zhang, G. Huguet, G. Wolf, and Y. Bengio. Simulation-free schrödinger bridges via score and flow matching. *arXiv preprint 2307.03672*, 2023.

[25] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, and S. Savarese. Gonet: A semi-supervised deep learning approach for traversability estimation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3044–3051. IEEE, 2018.

[26] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine. Rapid Exploration for Open-World Navigation with Latent Goal Models. In *5th Annual Conference on Robot Learning*, 2021.