2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

MetricNet: Recovering Metric Scale in Generative Navigation Policies

Abhijeet Nayak*[1], Débora Oliveira Makowski*[1], Samiran Gode*[1], Cordelia Schmid[2] and Wolfram Burgard[1]

***Abstract*— Generative navigation policies have made rapid progress in improving end-to-end learned navigation. Despite their promising results, this paradigm has two structural problems. First, the sampled trajectories exist in an abstract, unscaled space without metric grounding. Second, the control strategy discards the full path, instead moving directly towards a single waypoint. This leads to short-sighted and unsafe actions, moving the robot towards obstacles that a complete and correctly scaled path would circumvent. To address these issues, we propose MetricNet, an effective add-on for generative navigation that predicts the metric distance between waypoints, grounding policy outputs in metric coordinates. We evaluate our method in simulation with a new benchmarking framework and show that executing MetricNet-scaled waypoints significantly improves both navigation and exploration performance. Beyond simulation, we further validate our approach in real-world experiments. Finally, we propose MetricNav, which integrates MetricNet into a navigation policy to guide the robot away from obstacles while still moving towards the goal.**

I. Introduction

Classical navigation methods [1], [2], [3] have a long history of success in complex scenarios [4], [5], [6]. These methods reliably find paths to a goal by either generating or using pre-built maps of the environment [7], [8]. However, their primary drawback is the extensive calibration required to generalize to new environments or robot embodiments [9]. To bridge the domain gap, generative methods train visual-navigation foundation models using large-scale datasets [10], [11], [12], [13]. These methods often frame navigation as an end-to-end problem, where a diffusion-based policy [14] directly predicts a sequence of 2D waypoints using the current visual observation.

Such navigation policies operate on abstract 2D waypoints. This abstraction enables generality, allowing for cross-embodiment and scene invariance as long as the predicted direction is correct. The only tuning required to deploy this policy on a new robot is the distance the agent moves towards the chosen waypoint per time step, which is generally set as v_{max}/f, where v_{max} is the maximum linear velocity and f is the control frequency. For successful deployment, the policy must generate directionally accurate waypoints at a high frequency to enable reactive control.

Despite their significant progress, the applied v_{max}/f scale does not ground the trajectory in metric coordinates. Additionally, even though navigation policies predict a full trajectory, the agent ultimately executes only one short-term action based on a single waypoint. This approach contrasts with the original use of diffusion policies in manipulation [14], where the full metric trajectory is executed sequentially and policies are often scene- or embodiment-specific. This leads to unsafe behaviors, such as moving directly towards an obstacle that the full predicted path would have navigated around, as shown in Fig. 2. By grounding the trajectory in real-world coordinates, the metric representation enables explicit safety verification and goal-oriented guidance for downstream motion planners.

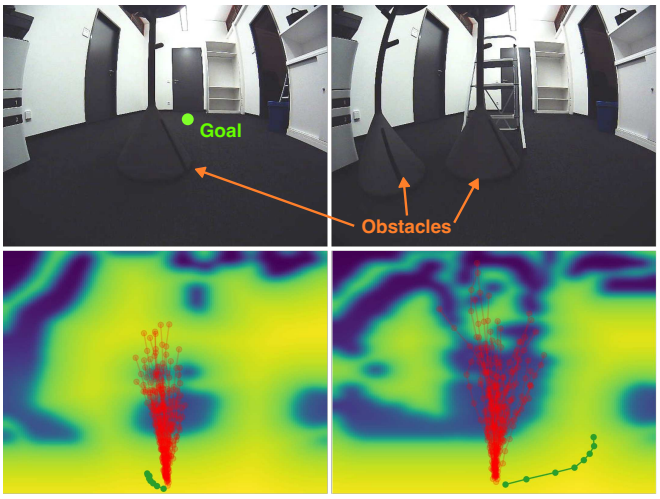


Fig. 1. Two examples of MetricNet transforming waypoints sampled from NoMaD [13] (red) into a collision-free plan (green) around an obstacle to reach the goal. First, our novel MetricNet module grounds the waypoints in the 3D world by predicting a metric scale. Second, MetricNav leverages these scaled points for goal following and collision avoidance around objects. The trajectories are plotted on a Truncated Signed Distance Function (TSDF).

Our method MetricNet addresses these problems by predicting the real distance between waypoints, enabling the robot to act directly on metric-grounded trajectories. MetricNet encodes the waypoints predicted from a diffusion policy and the current observation into a predicted real-world scale. We show that acting directly on the metric waypoints outperforms the state-of-the-art control approach in both simulation and real-world deployment. We also show that using the grounded waypoints helps guide the trajectories away from obstacles and towards the goal, as in Fig. 1.

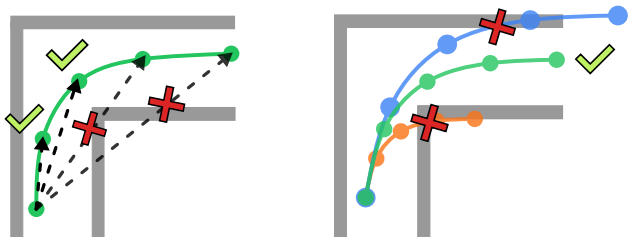
The main contributions of our work are:

- MetricNet, a novel network that learns to predict a

1 Artificial Intelligence and Robotics Lab, Department of Computer Science and Artificial Intelligence, University of Technology Nuremberg, Germany. {firstname.lastname}@utn.de

2 Inria, Ecole Normale Supérieure, CNRS, PSL Research University, France. {firstname.lastname}@inria.fr

* These authors contributed equally to the work. Listing order is random. Available at https://utn-air.github.io/metricnet



(a) Using different waypoints as goal for the velocity controller.

(b) Scaling the same trajectory with different scales.

Fig. 2. In (a), we illustrate that when using a velocity controller, selecting only the goal waypoint can still lead to collisions despite the predicted trajectory in metric scale being free of obstacles. Note this waypoint only points towards the direction. In (b), we show that depending on the scale factor ϕ, the resulting path may collide with walls if the scale is too small or too large for the environment. This highlights the need not only for the correct scale but also for a controller that executes the whole trajectory in metric space.

metric scale factor, grounding the outputs of generative policies into executable, metric grounded trajectories.

- A new benchmarking framework to evaluate generative visual-goal navigation and exploration policies, featuring challenging scenarios.
- MetricNav, a guidance framework that uses the metric trajectory to enforce collision avoidance and goal following simultaneously.
- An open-source release of our codebase, trained models, and real-world deployment videos.

II. Related Work

A. Learning-based Navigation

Recent machine learning methods have tackled the domain generalization problem in navigation using large-scale datasets. Approaches like RECON [15] and GNM [16] showed success with only RGB input, with the latter achieving zero-shot generalization by normalizing the action space. Other approaches have improved accuracy by incorporating additional sensing beyond RGB input, such as LiDAR [17], [18] and monocular depth priors [11], [19], [20].

A concurrent line of research explores language-conditioned navigation, which uses natural language to specify goals. For example, Poliformer [21], [22] proposes a zero-shot discrete navigation policy through reinforcement learning. Parallel work StreamVLN [23] introduced a dual system in which the navigation model runs faster than the language interpreter, enabling rapid reactions. While these advances in language instruction are promising, our primary focus is on visual navigation, where we predict actions directly from images captured by a front-facing camera.

B. Generative Navigation Models

Generative models have shown remarkable generalization capabilities in multimodal tasks such as navigation. ViNT [10] pioneered the use of diffusion [24] to generate intermediate goal images, while NoMaD [13] applied diffusion policy [14] to directly infer actions from visual observations. Subsequent work extended NoMaD's diffusion framework by sampling initial noise from a learned distribution [25], [26] or adopting conditional flow matching [11] for greater efficiency. More recently, diffusion has also been used in world models for navigation [27], where future visual observations are predicted from past inputs to evaluate whether a planned trajectory will successfully achieve the goal.

Although these methods demonstrate effective trajectory generation through imitation learning, they neglect the fact that training datasets typically lack collision examples. To address this limitation, NaviDiffusor [20] incorporates 3D-based obstacle avoidance into diffusion policies via gradient guidance, while NavDP [12] employs a critic to estimate the best collision-free path. Similarly, GND [28] encodes LiDAR scans into a traversability map to improve safety.

However, one must note that all approaches cited above rely on a large-scale dataset and therefore require input normalization for training. At deployment, unscaling this normalization needs environment- and robot-specific fine-tuning, which may lead to collisions. In this work, we propose directly learning a network to project the trajectory back into metric units, thereby eliminating the need for handcrafted adjustments and enabling robust navigation.

III. Overview

A. Problem Definition

Generative visual-navigation models address goal-conditioned navigation by learning a policy π which, given an observation horizon $\mathcal{O}$ and a goal image $\mathbf{g}_i$, predicts an action $A_i = \pi(\mathbf{o}_i, \mathbf{g}_i)$ that steers the robot towards the goal $\mathbf{g}_i$. In the absence of a goal, π should explore the environment without collisions. The observation horizon $\mathcal{O}_i = [\mathbf{o}_i : \mathbf{o}_{i-T_O}]$ contains the images from the camera $\mathbf{o}_i$ at the timestamp i and in the past horizon of length T_O. The policy π samples from the distribution $P(A_i \mid \mathcal{O}_i)$ where A_i is the set of predicted future actions in ego frame with a future horizon length T_A, such that $A_i = [\mathbf{a}_i : \mathbf{a}_{i+T_A}]$, where $\mathbf{a}_i$ is a single 2D waypoint. The trajectory is only executed until a chosen waypoint $T_W \leq T_A$. The predicted action A_i is normalized by the average waypoint distance of the segment. To scale it back to the metric 3D world, an unknown scale ϕ is needed. In this work, we predict this scale using MetricNet, enabling easier deployment without the need for manual tuning.

B. Action Generation for Navigation Policies

A generic navigation policy takes as input the observation horizon $\mathcal{O}_i$, along with an optional goal image $\mathbf{g}_i$. These images are first encoded into tokens, which are then processed by a transformer to create a context vector $\mathbf{c}_i$. This context is then passed as a condition to the generative policy to sample waypoints A_i in a normalized space..

Previous works used different generative models to sample trajectories. NoMaD [13] uses Denoising Diffusion Probabilistic Models (DDPM) [14] for action generation, while FlowNav [11] employs Conditional Flow Matching (CFM) [29]. NaviDiffusor [20] is built on top of NoMaD

by guiding the diffusion policy to the collision-free space. Diffusion policies [14] sample from a Gaussian distribution and iteratively denoise the sample using a learned noise-prediction network $\varepsilon_\theta(A_i^k, \mathbf{c}_i, k)$. The time k indicates the current denoising iteration, while A_i^k denotes the action after k rounds of noise addition. Starting from A_i^1, drawn from the Gaussian prior, the denoising update is performed as

$$A_i^{k-1} = \eta(A_i^k - \gamma\varepsilon_\theta\big(A_i^k, \mathbf{c}_i, k) + \mathcal{N}(0, \sigma^2 I)\big). \quad (1)$$

During training, k is randomly selected to get the noise ε_k which is then added to the groundtruth sample A_i^0 to obtain A_i^k. The model is trained by reducing

$$\mathcal{L} = \text{MSE}\Big(\varepsilon_k, \varepsilon_\theta\big(A_i^k, \mathbf{c}_i, k\big)\Big). \quad (2)$$

C. Action Normalization

To ensure stable training, the trajectories of length T_A are normalized in two steps before applying the loss in Eq. 2. The first step compensates for length differences across datasets, since outdoor vehicles typically move faster than indoor robots, as in

$$\phi_{\text{gt}} = \frac{\sum_{k=0}^{T_A-1} \|\mathbf{a}_{i+(k+1)} - \mathbf{a}_{i+k}\|_2}{T_A - 1}. \quad (3)$$

where ϕ_{gt} is the average distance between consecutive waypoints, and $\tilde{A}_i = A_i/\phi_{\text{gt}}$. We define ϕ_{gt} as the ground truth waypoint distance. As in diffusion policy [14], $\tilde{A}$ is converted into a sequence of deltas $\Delta\tilde{A}_i$ that represent the displacement between consecutive waypoints, with $\Delta\tilde{\mathbf{a}}_t = \tilde{\mathbf{a}}_t - \tilde{\mathbf{a}}_{t-1}$. The trajectory is then normalized between $[-1, 1]$ using

$$\Delta\hat{\mathbf{a}} = \frac{2\big(\Delta\tilde{\mathbf{a}} - [x_{\text{min}}, y_{\text{min}}]\big)}{[x_{\text{max}}, y_{\text{max}}] - [x_{\text{min}}, y_{\text{min}}]} - 1, \quad (4)$$

The minima and maxima are calculated across all training datasets. During inference, $\Delta\tilde{\mathbf{a}}$ is reversed as

$$\Delta\tilde{\mathbf{a}} = \frac{[x_{\text{max}}, y_{\text{max}}] - [x_{\text{min}}, y_{\text{min}}]}{2\big(\Delta\hat{\mathbf{a}} - [x_{\text{min}}, y_{\text{min}}]\big)} + 1. \quad (5)$$

We retrieve the full waypoint trajectory by integrating $\tilde{A}_i$ with $\tilde{\mathbf{a}}_t = \tilde{\mathbf{a}}_{t-1} + \Delta\tilde{\mathbf{a}}_t$ and $\tilde{\mathbf{a}}_i = [0, 0]$.

Note that, during inference, the first normalization step cannot be reversed directly because the metric scale to the real-world ϕ_{gt} is unknown. To recover the metric trajectory, previous works apply a handcrafted scaling factor ϕ_{tuned} as

$$A_i = \phi_{\text{tuned}}\tilde{A}_i. \quad (6)$$

D. Velocity Control

Earlier approaches deploy the trajectory A_i moving directly to the target waypoint T_W. We call this approach velocity control. The linear velocity v_i and angular velocity ω_i are defined as

$$\begin{aligned} v_i &= \frac{\|\mathbf{a}_{i+T_W}\|_2}{\delta t} \text{ and} \\ \omega_i &= \frac{\arctan(\tilde{y}_{i+T_W}/\tilde{x}_{i+T_W})}{\delta t}, \end{aligned} \quad (7)$$

where $\tilde{\mathbf{a}}_{i+T_W} = [\tilde{x}_{i+T_W}, \tilde{y}_{i+T_W}]$, timestep $\delta t = 1/f$, and f is the control frequency. This operation assumes that the robot moves to the goal waypoint at T_W in one time unit. However, in practice, the robot's velocity is clipped at $(v_{\text{max}}, \omega_{\text{max}})$. Since the waypoint $\tilde{\mathbf{a}}_{i+T_W}$ is still normalized by the average waypoint distance, $v_i \gg v_{max}$. For instance, in the case of NoMaD [13] where $v_{\text{max}} = 0.4$m/s and $f = 15$Hz, if $\tilde{\mathbf{a}} \approx 1$, $v_i = 15$m/s for $\phi_{\text{tuned}} = 1$. Therefore, the state-of-the-art uses $\phi_{\text{tuned}} = v_{\text{max}}/f$ in Eq. 6, which is not necessarily metric scaling. As a result, ϕ_{tuned} does not represent the distance between each waypoint in metric space, ϕ_{gt}. Also, as shown in Fig. 2a, velocity control targets a single waypoint T_W without executing the rest of the planned path, causing the robot to cut corners and collide with obstacles. The novel addition in our work is learning ϕ_{gt}, allowing trajectories to be projected to the metric space without tuning a scale.

IV. Methodology

In this section, we first describe our novel network that grounds sampled trajectories from generative policies in metric coordinates. We then define how to act upon a sequence of these grounded waypoints. Finally, we show how metric grounded waypoints can be used to improve the obstacle-avoidance and goal-following capability of diffusion policies.

A. MetricNet

Our architecture is shown in Fig. 3. We encode the unscaled trajectory $\tilde{A}$ using a one-dimensional convolutional network, where $w_\theta(A_i) \in \mathbb{R}^D$ denotes the waypoint encoding with $D = 384$. We split the current observation $\mathbf{o}_i$ into P_f patches of 7×7 pixels and encode them using an EfficientNet-B0 [30] denoted by $\rho_\theta(\mathbf{o}_i) \in \mathbb{R}^{P_f \times F}$ with $F = 1280$. These patch embeddings are projected into a common latent dimension D using a residual network.

Like [11], we add depth information to our tokens using the pre-trained DINOv2 [32] ViTS encoder from Depth-Anything-V2 [31] to generate patch tokens $d_\theta(\mathbf{o}_i) \in \mathbb{R}^{P_d \times D}$. The patch tokens of size 16×16 pixels are then processed by a single-block residual network, which adds flexibility to the tokens since the DINOv2 encoder remains frozen.

The next block is a transformer encoder F that concatenates the observation, depth, and waypoint tokens using self-attention. We prepend a classifier token $\text{CLS}_{\text{in}} \in \mathbb{R}^D$ to the beginning of the sequence to capture contextual information [33]. We use a small transformer of six heads with three attention layers each to create the CLS_{out} token

$$\text{CLS}_{\text{out}} = F\big(\text{CLS}_{\text{in}}, w_\theta(A_i), \rho_\theta(\mathbf{o}_i), d_\theta(\mathbf{o}_i)\big). \quad (8)$$

CLS_{out} is then passed through an MLP $g_\theta(\text{CLS}_{\text{out}})$ to predict a scale ϕ_{pred} that grounds the waypoints to the metric space.

The objective used to train MetricNet is defined as

$$\mathcal{L}(\alpha) = \lambda \cdot \text{MSE}\Big(\phi_{\text{pred}}, \phi_{\text{gt}}\Big), \quad (9)$$

where λ is a loss scaling factor. To avoid gradient collapse, we scale the loss to millimeters ($\lambda =$ 1e-3).

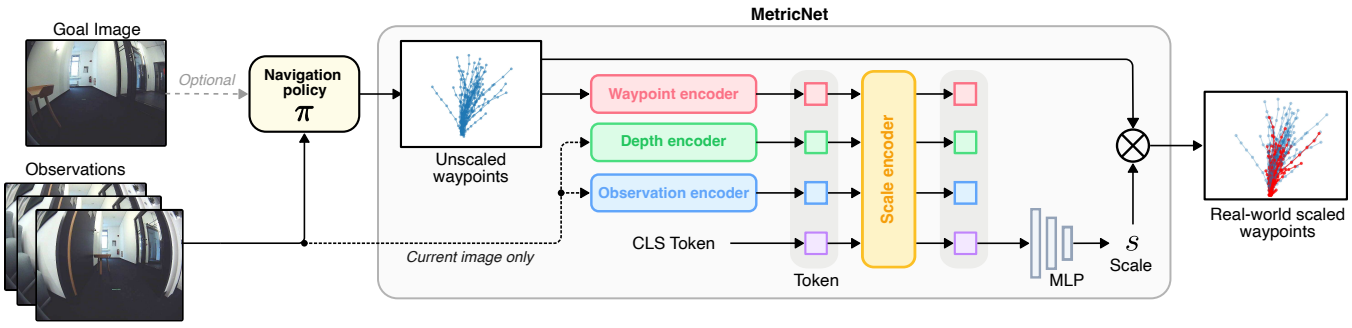


Fig. 3. MetricNet architecture. Our network estimates a factor that converts the unscaled output of generative visual-goal navigation policies into metric scale. MetricNet first tokenizes patches of the current observation using an image encoder. In parallel, the observation is processed with the pre-trained encoder from the Depth-Anything-V2 [31] to produce depth patch tokens. A transformer then processes the combined token sequence, prepending a CLS token to summarize an aggregated representation. Finally, the output CLS token is passed through an MLP to predict the estimated scale. This scale is multiplied by the original unscaled waypoints (blue), generating grounded trajectories in metric coordinates (red).

B. Position Control

With MetricNet, the trajectory is scaled to metric space, allowing us to move across the entire sequence of waypoints. We refer to this method as position control. Since the robot's view may be occluded (e.g., in curves) or obstacles may be too distant to recognize, we follow only part of the trajectory horizon T_A before recomputing the trajectory [34]. As in velocity control, we set waypoint T_W as the trajectory limit. However, unlike velocity control, a position controller executes the full segment from 0 to T_W, preventing cutting corners and reducing collisions.

C. MetricNav

To give the robot additional collision avoidance and goal guidance capabilities, we draw inspiration from previous ideas [20], [35] to create MetricNav, which uses MetricNet to guide any navigation diffusion policy away from obstacles and towards the goal as shown in Fig. 4. We define the collision and goal costs which are used for the guiding the diffusion policy and the post diffusion goal selection below.

1) Collision Cost: Given the metric waypoints, we can now project the rescaled trajectories in a metric pointcloud. To estimate this pointcloud, we use a metric depth model for monocular images. Specifically, we use the ViTB variant of Depth-Anything-V2 [31], which is a state-of-the-art model for metric depth estimation. With approximate intrinsic calibration parameters, the depth image can be converted into an ego-centric pointcloud. To make a clear distinction between obstacle and free points, we perform RANSAC-based ground plane segmentation on the pointcloud. The ground plane points are marked as free points, while the rest of the points are considered obstacle points.

We use these points to create a local Truncated Signed Distance Function (TSDF), where the positions of the obstacles correspond to their actual positions in the 3D space. In contrast to the TSDF representation previously used [20], [35], where non-zero values exist only in the obstacle space, our TSDF representation has non-zero values in both the obstacle and free space. This ensures that gradients can still be accumulated to guide trajectories away from the obstacles, even though the sampled paths lie entirely in free space. The collision cost is defined as $\mathcal{F}_{\text{coll}}$

$$\mathcal{F}_{\text{coll}}(A_i^k, \mathbf{o}_i) = \sum_{m=i}^{i+T_A} \big[\mathcal{C}(\mathbf{a}_m) + \mathcal{C}(\mathbf{a}_m + \sigma_{\text{L}}) + \mathcal{C}(\mathbf{a}_m + \sigma_{\text{R}})\big], \tag{10}$$

where $\mathcal{C}(\mathbf{a}_m)$ is the cost of each waypoint, and $\mathcal{C}(\mathbf{a}_m + \sigma_{\text{L}})$ and $\mathcal{C}(\mathbf{a}_m + \sigma_{\text{R}})$ are the costs at the points that represent the boundaries of the robot.

2) Goal Cost: Different from NaviDiffusor [20], our cost is explicitly based on the most likely goal direction. Since diffusion-based navigation policies can generate multiple trajectories that move towards a goal image, we initially sample $\mathcal{K}$ trajectories from the policy. For an ideal navigation policy, the most likely action would point the robot to the image goal. Assuming this to be the case for our base policy, we apply spherical k-means [36] to cluster the direction of the N-th waypoint of the sampled $\mathcal{K}$ actions. The action closest to the cluster center with the largest number of associated actions is chosen as the goal direction.

After selecting a goal, we run a second loop of action generation to define the goal cost function for each scaled action as

$$\mathcal{F}_{\text{goal}}(A_i) = 1 - \frac{\mathbf{v}_{\text{goal}} \cdot \mathbf{v}_{\mathbf{a}_{i+N}}}{\|\mathbf{v}_{\text{goal}}\|_2 \|\mathbf{v}_{\mathbf{a}_{i+N}}\|_2}, \tag{11}$$

where $\mathbf{v}_{\text{goal}}$ is defined as the direction that points to the goal and $\mathbf{v}_{\mathbf{a}_i}$ is defined as the direction of each action based on its N-th waypoint.

3) Guidance: Obstacle avoidance is carried out by moving the waypoints towards the free space. The goal guidance is carried out by maximizing the cosine-similarity between the goal position and the N-th waypoints of the generated actions. We define the total cost as

$$\mathcal{F}_{\text{total}}(A_i^k, \mathbf{o}_i) = \alpha \mathcal{F}_{\text{goal}}(A_i^k) + \beta \mathcal{F}_{\text{coll}}(A_i^k, \mathbf{o}_i). \tag{12}$$

We accumulate gradients for each waypoint in a trajectory by performing a backward pass on this loss function. The

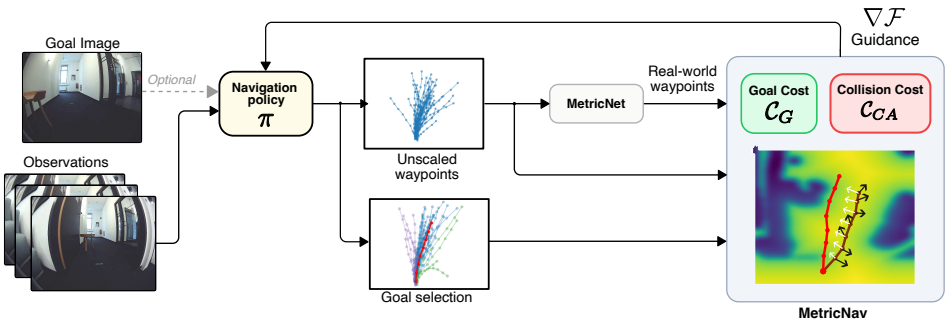


Fig. 4. MetricNav architecture uses a navigation policy to sample $\mathcal{K}$ trajectories (blue). Next, spherical k-means clustering is used to estimate the most common goal trajectory (red). This trajectory is then used to derive a cost composed of a goal following and a collision avoidance term using an estimated TSDF from monocular depth. This cost is then fed into the diffusion policy to guide the sampled trajectory into the obstacle-free space, while moving towards the goal. The white arrows show the goal guiding gradients whereas the brown arrows show the collision avoiding gradients.

cost $\mathcal{F}(A_i^k, \mathbf{o}_i)$ is then used to guide each denoising step

$$A_i^{k-1} = \eta(A_i^k - \gamma\varepsilon_\theta(A_i^k, \mathbf{c}_i, k) + \mathcal{N}(0, \sigma^2 I)) - \mu\nabla_{A_i^k}\mathcal{F}_{\text{total}}(A_i^k, \mathbf{o}_i). \quad (13)$$

By performing a gradient descent on the unnormalized waypoints with step size μ, the sampled trajectories are guided away from collision spaces, while also keeping them pointed towards the general goal direction. We run the guidance steps in diffusion only for $k < 2$. In our experiments, we use α = 0.5, β = 0.01 and μ = 0.1, which were tuned to provide the best goal-guided and collision-free actions.

4) Final Action Selection: With the use of goal and collision guidance, the actions are safer while also being sufficiently goal-directed. Given $\mathcal{K}$ sampled actions, we perform an action selection step which includes computing the goal similarity and collision cost for each guided trajectory. The goal factor $\mathcal{C}_G$ is defined as the cosine similarity between an action and the chosen goal. Collision factor $\mathcal{C}_{CA}$ is defined as the negative of the collision cost in Eq. 10. The total cost is defined as

$$\mathcal{C}_{\text{total}} = \gamma\mathcal{C}_G + (1-\gamma)\mathcal{C}_{CA}. \quad (14)$$

The parameter γ pushes trajectories either towards goal guidance or collision avoidance. The action with the best $\mathcal{C}_{\text{total}}$ is executed.

V. Experiments

A. Training

We trained three baseline models: NoMaD [13], FlowNav [11] and NaviDiffusor [20]. Similar to previous policies, we use a prediction horizon of $T_A = 8$ and a goal waypoint $T_W = 3$ for both position and velocity control. We benchmark against the baselines in both real-world and simulation. To ensure fairness, all baseline models are trained on the same data split. Policies are trained on synthetic data for benchmarking and real-world data for deployment to maximize navigation performance and reduce the sim-to-real gap. All navigation baselines were trained using the original authors' official implementations and training strategies. MetricNet was trained for 15 epochs on an NVIDIA H200 GPU with a batch size of 1024. We use AdamW [37] optimizer with an initial learning rate of 1e-4 with the cosine scheduler [38]. We train MetricNet on both real-world and synthetic data, totaling around 1.5 million data points.

B. Datasets

For the real-world data, we combine GoStanford [39], RECON [15], Tartan Drive [40], SACSoN [41], and SCAND [42], which contain indoor and outdoor trajectories featuring a diverse range of embodiments and camera parameters. Similar to NoMaD [13], we randomly select trajectory segments with a past observation context of length $T_O = 4$. The segments are then normalized following Eq. 3 and Eq. 4 to allow for training on cross-embodied datasets.

We generate synthetic training data by using the Habitat simulator [43] to collect 16k trajectories across 800 Matterport3D indoor scenes [44]. We uniformly select camera parameters such as the fisheye angle between 0° and 180° and the camera height between 25cm and 1m. Trajectories involving stairs are excluded, since our robot is incapable of reproducing this behavior.

For MetricNet, the ground truth scale ϕ_{gt} is collected by averaging the distance between the waypoints in each selected segment. To increase diversity of ϕ_{gt}, we subsample trajectories by randomly skipping between 1 and 5 waypoints in indoor datasets, and up to 2 waypoints in outdoor datasets. We subsample synthetic trajectories by spacing them every 1 to 10 waypoints. While MetricNet is trained on synthetic datasets with one observation every 10 cm, the synthetic dataset for policy training uses a coarser resolution, with one observation every 25 cm, as smaller waypoint spacing tended to bias models toward straight-line predictions.

C. Simulation Benchmarking

For benchmarking in simulation, we compare the baselines over 20 topological maps in 20 different environments. A topological map is an ordered collection of images as in [13], [16], [45]. The trajectories are randomly sampled and have a minimum length of 7.5m. To make a fair comparison, the agents are spawned on the first position of each topological map [15] and run with the same seed across all baselines.

We test over 5 different seeds, resulting in an evaluation set of 2k repetitions per baseline. For collision checking, we use the dimensions of the TurtleBot4.

D. Real-world Deployment

For real-world experiments, we deploy all models on a TurtleBot4 using ROS2 Humble on Discovery Server mode. We use an NVIDIA RTX A500 laptop GPU for model inference. Following the baselines [13], [11], we use a fisheye camera capturing images at 15Hz. To follow the grounded metric waypoints, we use a rotate–translate strategy: the robot first rotates at angular velocity $\omega_{\max}$ for $\arctan(\tilde{y}_{i+T_W})/\omega_{\max}$ seconds, then translates at linear velocity $v_{\max}$ for $\|\tilde{\mathbf{a}}_i\|/v_{\max}$ seconds. We avoid using reactive controllers such as the Dynamic Window Approach [46] to prevent giving the models unfair advantages. The test environment is a long corridor of length approximately 20m with multiple turns and obstacles positioned both at the corners and along the straight segments. Each model is evaluated over 5 runs. We report the average success rate and collision count. An evaluation is terminated if the robot experiences a frontal collision or a situation where the robot cannot recover within 10 seconds.

E. Experimental Results

For navigation, we collect a topological map and measure performance as the maximum percentage of the topomap that the robot can reach before collision. We follow a similar strategy in the real-world experiments for goal-conditioned navigation. Exploration experiments are only run in simulation, where we score the distance traveled before collision.

1) Comparing different controller paradigms: Fig. 5 compares models moving across all waypoints until T_W (position control) or targeting a single waypoint (velocity control). In all cases, position control achieves higher success rates and greater explored distance when using MetricNet compared to velocity control. We attribute this improvement to MetricNet, which grounds waypoint predictions in metric space, enabling the robot to follow waypoints directly rather than taking incremental steps. Additionally, we notice that using a fixed scale of $v_{\max}/f$ degrades performance across all models compared to MetricNet, highlighting the effectiveness of learning a waypoint scaling factor for better control. While MetricNet improved all baselines, the performance differences between the base policies are smaller than what is reported in other papers. This is a direct result of our benchmark, which we designed to be more challenging to test the limits of current models. Our environments include tight curves and topological maps created with shortest paths that force the robot to navigate near walls. By providing this difficult evaluation, we aim to encourage the development of more robust navigation policies.

2) Real world deployment: Real-world navigation results are reported in Table I, with each model tested using both velocity and position controllers. Note that position controller is scaled with MetricNet, while velocity control uses $v_{\max}/f$. Since inference runs on a laptop, MetricNav is slower and less suited to velocity control. Note that MetricNav achieves the highest goal-reaching rate with fewer collisions per run. Moreover, all models show improved goal-reaching when paired with MetricNet and the position controller compared to velocity control.

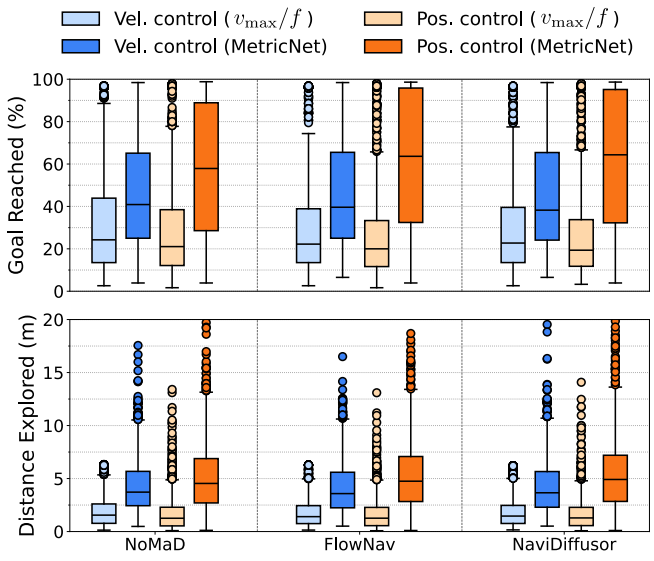


Fig. 5. Box plot for navigation and exploration in simulation across velocity and position control using a constant scale and prediction by MetricNet. Using MetricNet outperforms using the constant scale proposed by previous work across all base navigation policies. Moreover, using metric waypoints predicted by MetricNet with position controller improves the result compared to the velocity control. Note that in all navigation experiments, at least one seed fails and one reaches the goal across all baseline policies. The key difference lies in the median of the distribution and in how many average seeds succeed.

TABLE I: SUCCESS RATE AND NO. OF COLLISIONS PER RUN FOR NAVIGATION IN REAL-WORLD DEPLOYMENT

Method	**Velocity control**		**Position control**	
	SR ↑	#Coll ↓	SR ↑	#Coll ↓
NoMaD [13]	0.76	1.0	0.87	1.8
FlowNav [11]	0.85	1.0	0.93	1.2
NaviDiffusor [20]	0.89	0.8	0.93	0.8
MetricNav (ours)	-	-	**0.96**	**0.6**

3) Using goal-guidance and metric TSDF: Table II reports the navigation results for all baselines and MetricNav using metric grounded waypoints produced by MetricNet. Since we do not have the real-world constraints on time synchronization, we benchmark MetricNav also in velocity control. We use $T_W = 1$ as a goal waypoint to create a fair comparison to the baselines that do not have active collision avoidance. Note that MetricNav has higher scores in velocity control and comparable results in position control. MetricNav performs better than other baselines in velocity control because it avoids obstacles while still moving towards the goal.

4) Qualitative Results: In Fig. 6 we show how the values chosen for α and β for the cost function affect the

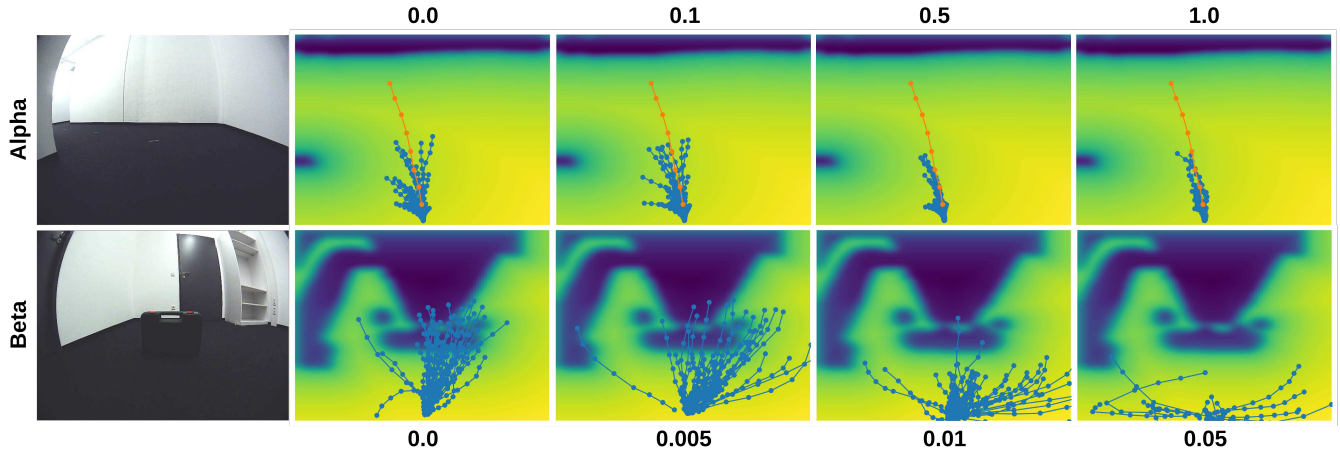


Fig. 6. Example of trajectories generated with different values of α and β. The larger the value of α, the stronger the guidance of the denoised trajectories to the goal. The larger the value of β, the stronger the guidance away from obstacles.

TABLE II: AVERAGE FRACTION OF THE TOPOMAP COMPLETED USING POSITION AND VELOCITY CONTROL IN THE SIMULATION BENCHMARK

Method	**Goal reached** ↑			
	NoMaD	FlowNav	NaviDiffusor	MetricNav
Vel. control	0.46	0.45	0.46	**0.51**
Pos. control	0.59	**0.61**	**0.61**	**0.61**

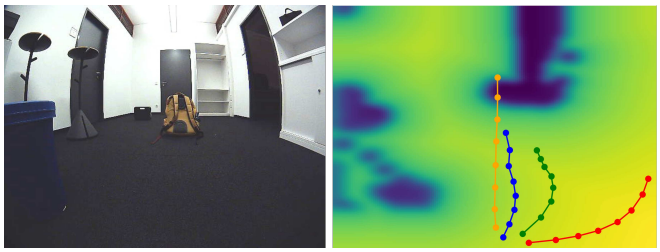


Fig. 7. Example of trajectories produced with different γ. The original selected goal trajectory (orange) collides with the object, whereas all guided and scaled trajectories by MetricNav lie in the free space. The trajectory with $\gamma = 1$ (blue) considers only goal cost and stays close to the goal. The trajectory with $\gamma = 0$ (red) considers only collision avoidance and prefers driving itself away to a safe state. A balance can be found with $\gamma = 0.5$ (green).

distribution of guided trajectories that are sampled from the generative policy. The orange trajectory represents the chosen goal based on the spherical k-means clustering carried out on the $\mathcal{K}$ sampled trajectories. As α controls the goal guidance, we observe that increasing α values guide the sampled trajectories towards the goal with increasing strength. As β controls the collision avoidance strength, we observe that increasing β values moves trajectories away from the obstacles. In Fig. 7 we also show how γ affects the chosen action that is executed on the robot. We fix $\alpha = 0.5$ and $\beta = 0.01$ and plot actions for $\gamma \in \{0, 0.5, 1\}$. The orange trajectory represents the chosen goal trajectory, the blue trajectory ($\gamma = 1.0$) closely follows it, the red trajectory ($\gamma = 0.0$) prioritizes collision avoidance, and the green trajectory ($\gamma = 0.5$) balances both. This shows that unguided trajectories often cause collisions, whereas guided ones help the robot progress towards the goal while avoiding obstacles.

VI. CONCLUSIONS

In this paper, we present MetricNet, a network that predicts the scale factor to project unnormalized trajectories from navigation diffusion policies into metric space. Grounding trajectories in metric space avoids hand-tuning calibration and allows multiple waypoints to be executed, significantly improving performance in both navigation and exploration. We demonstrate that using metric waypoints from MetricNet in our guidance framework MetricNav helps diffusion navigation policies avoid obstacles and move towards predefined image goals. We validate our method in simulation and real-world experiments. As future work, we propose using a controller that allows for reactiveness to dynamic obstacles while executing the metric waypoints, such as the Dynamic Window Approach.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the BayernKI project v106be. BayernKI funding is provided by Bavarian state authorities. We also acknowledge the support by the Körber European Science Prize. This work has been partially supported by the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG).

REFERENCES

[1] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[2] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, K. L.E., and S. Thrun, *Principles of Robot Motion Planing*. MIT-Press, 2005.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[4] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, 2000.

[5] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, 2006.

[6] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, *et al.*, "Cerberus in the darpa subterranean challenge," *Science Robotics*, 2022.

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[8] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[9] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, 2016.

[10] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "Vint: A large-scale, multi-task visual navigation backbone with cross-robot generalization," in *Proc. of the Conf. on Robot Learning (CoRL)*, 2023.

[11] S. Gode, A. Nayak, D. N. Oliveira, M. Krawez, C. Schmid, and W. Burgard, "Flownav: Combining flow matching and depth priors for efficient navigation," *arXiv:2411.09524*, 2024.

[12] W. Cai, J. Peng, Y. Yang, Y. Zhang, M. Wei, H. Wang, Y. Chen, T. Wang, and J. Pang, "Navdp: Learning sim-to-real navigation diffusion policy with privileged information guidance," *arXiv:2505.08712*, 2025.

[13] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "Nomad: Goal masked diffusion policies for navigation and exploration," in *Proc. of the IEEE Int. Conference on Robotics & Automation (ICRA)*, 2024.

[14] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *Int. Journal of Robotics Research (IJRR)*, 2024.

[15] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine, "Rapid exploration for open-world navigation with latent goal models," in *Proc. of the Conf. on Robot Learning (CoRL)*, 2021.

[16] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "Gnm: A general navigation model to drive any robot," in *Proc. of the IEEE Int. Conference on Robotics & Automation (ICRA)*, 2023.

[17] J. Liang, A. Payandeh, D. Song, X. Xiao, and D. Manocha, "Dtg: Diffusion-based trajectory generation for mapless global navigation," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

[18] J. Liang, P. Gao, X. Xiao, A. J. Sathyamoorthy, M. Elnoor, M. C. Lin, and D. Manocha, "Mtg: Mapless trajectory generator with traversability coverage for outdoor navigation," in *Proc. of the IEEE Int. Conference on Robotics & Automation (ICRA)*, 2024.

[19] J. Kim, J. Sim, W. Kim, K. Sycara, and C. Nam, "Care: Enhancing safety of foundation models for visual navigation through collision avoidance via repulsive estimation," *arXiv:2506.03834*, 2025.

[20] Y. Zeng, H. Ren, S. Wang, J. Huang, and H. Cheng, "Navidiffusor: Cost-guided diffusion model for visual navigation," in *Proc. of the IEEE Int. Conference on Robotics & Automation (ICRA)*, 2025.

[21] K.-H. Zeng, Z. Zhang, K. Ehsani, R. Hendrix, J. Salvador, A. Herrasti, R. Girshick, A. Kembhavi, and L. Weihs, "Poliformer: Scaling on-policy rl with transformers results in masterful navigators," in *Proc. of the Conf. on Robot Learning (CoRL)*, 2025.

[22] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martín-Martín, P. Stone, K.-H. Zeng, and K. Ehsani, "Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning," in *Proc. of the Conf. on Robot Learning (CoRL)*, 2024.

[23] M. Wei, C. Wan, X. Yu, T. Wang, Y. Yang, X. Mao, C. Zhu, W. Cai, H. Wang, Y. Chen, *et al.*, "Streamvln: Streaming vision-and-language navigation via slowfast context modeling," *arXiv:2507.05240*, 2025.

[24] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, 2020.

[25] H. Ren, Y. Zeng, Z. Bi, Z. Wan, J. Huang, and H. Cheng, "Prior does matter: Visual navigation via denoising diffusion bridge models," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[26] L. Zhou, A. Lou, S. Khanna, and S. Ermon, "Denoising diffusion bridge models," in *ICLR*, 2024.

[27] A. Bar, G. Zhou, D. Tran, T. Darrell, and Y. LeCun, "Navigation world models," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[28] J. Liang, D. Das, D. Song, M. N. H. Shuvo, M. Durrani, K. Taranath, I. Penskiy, D. Manocha, and X. Xiao, "Gnd: Global navigation dataset with multi-modal perception and multi-category traversability in outdoor campus environments," in *Proc. of the IEEE Int. Conference on Robotics & Automation (ICRA)*, 2025.

[29] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2023.

[30] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. of the Int. Conference on Machine Learning (ICML)*, 2019.

[31] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," in *Advances in Neural Information Processing Systems*, 2025.

[32] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, *et al.*, "DINOv2: Learning robust visual features without supervision," *Transactions on Machine Learning Research*, 2024.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of the Conference of the North American chapter of the Association for Computational Linguistics (NAACL)*, 2019.

[34] K. Black, M. Y. Galliker, and S. Levine, "Real-time execution of action chunking flow policies," *arXiv:2506.07339*, 2025.

[35] P. Roth, J. Nubert, F. Yang, M. Mittal, and M. Hutter, "Viplanner: Visual semantic imperative learning for local navigation," in *Proc. of the IEEE Int. Conference on Robotics & Automation (ICRA)*, 2024.

[36] K. Hornik, I. Feinerer, M. Kober, and C. Buchta, "Spherical k-means clustering," *Journal of statistical software*, 2012.

[37] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2019.

[38] ——, "Sgdr: Stochastic gradient descent with warm restarts," in *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2022.

[39] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, and S. Savarese, "Gonet: A semi-supervised deep learning approach for traversability estimation," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[40] S. Triest, M. Sivaprakasam, S. J. Wang, W. Wang, A. M. Johnson, and S. Scherer, "TartanDrive: A large-scale dataset for learning off-road dynamics models," in *Proc. of the IEEE Int. Conference on Robotics & Automation (ICRA)*, 2022.

[41] N. Hirose, D. Shah, A. Sridhar, and S. Levine, "SACSON: Scalable autonomous control for social navigation," *IEEE Robotics and Automation Letters*, 2023.

[42] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas, and P. Stone, "Socially compliant navigation dataset (SCAND): A large-scale dataset of demonstrations for social navigation," *IEEE Robotics and Automation Letters*, 2022.

[43] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, *et al.*, "Habitat: A platform for embodied ai research," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[44] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *Int. Conf. on 3D Vision (3DV)*, 2017.

[45] D. Shah and S. Levine, "ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints," in *Proc. of Robotics: Science and Systems (RSS)*, 2022.

[46] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, 2002.