TERP: Reliable Planning in Uneven Outdoor Environments using Deep Reinforcement Learning

Kasun Weerakoon[1], Adarsh Jagan Sathyamoorthy[1], Utsav Patel[2], and Dinesh Manocha[2]

***Abstract*— We present a novel method for reliable robot navigation in uneven outdoor terrains. Our approach employs a fully-trained Deep Reinforcement Learning (DRL) network that uses elevation maps of the environment, robot pose, and goal as inputs to compute an attention mask of the environment. The attention mask is used to identify reduced stability regions in the elevation map and is computed using channel and spatial attention modules and a novel reward function. We continuously compute and update a navigation cost-map that encodes the elevation information or the level-of-flatness of the terrain using the attention mask. We then generate locally least-cost waypoints on the cost-map and compute the final dynamically feasible trajectory using another DRL-based method. Our approach guarantees safe, locally least-cost paths and dynamically feasible robot velocities in uneven terrains. We observe an increase of 35.18% in terms of success rate and, a decrease of 26.14% in the cumulative elevation gradient of the robot's trajectory compared to prior navigation methods in high-elevation regions. We evaluate our method on a Husky robot in real-world uneven terrains ($\sim 4m$ of elevation gain) and demonstrate its benefits.**

I. Introduction

Autonomous mobile robots have increasingly been used for many real-world field applications such as indoor and outdoor surveillance, search and rescue, planetary/space exploration, large agricultural surveys, etc. Each of these applications requires the robot to operate in different kinds of terrains, which can be characterized by visual features such as color and texture, and geometric features such as elevation changes, slope, etc.

A robot's stability, which includes its pitch and roll angles being within certain limits, is predominantly dictated by the elevation changes/unevenness and slope of the terrain [1]. For reliable navigation, robots need to identify unsafe elevation changes and plan trajectories along planar regions to a large extent. However, sensing and navigating in uneven unstructured environments can be challenging because we do not have a complete model of the terrain with all the elevation information [2], [3]. Rather this information is gathered using camera or LiDAR sensors as the robot navigates. In addition, elevation changes cannot be adequately inferred only from visual features of the environment [4]. Prior works have addressed this by using grid-based data structures such as Octomaps [5] and elevation maps [6], which are 2D grids that contain the maximum elevation (in meters) at each grid.

Many other techniques have been proposed in the outdoor domain including semantic segmentation-based perception methods [7], [8], [9], [10], and Deep Reinforcement Learning-based navigation methods [11], [12]. Semantic segmentation methods in this domain typically classify terrains in images based on whether they are traversable for a robot [7], [8], [9] by learning the visual features corresponding to the color and texture of different objects. However, they are trained using human annotations which may not be suitable for all types of robots with different dynamic constraints. Therefore, a terrain that is traversable for a robot could be misclassified as non-traversable. As a result, navigation methods that use these segmentation methods for perception could be overly conservative for robot navigation and may result in sub-optimal trajectories [13].

DRL-based navigation approaches [14], [15], have been used for robot navigation in crowded scenes with dynamic obstacles. However, they are mostly limited to navigation on flat surfaces and do not account for varying elevation in outdoor terrains. Moreover, these DRL methods are typically trained using simulations and may face performance degradation in real-world environments [14]. Such methods also cannot provide any optimality guarantees on the robot's trajectories.

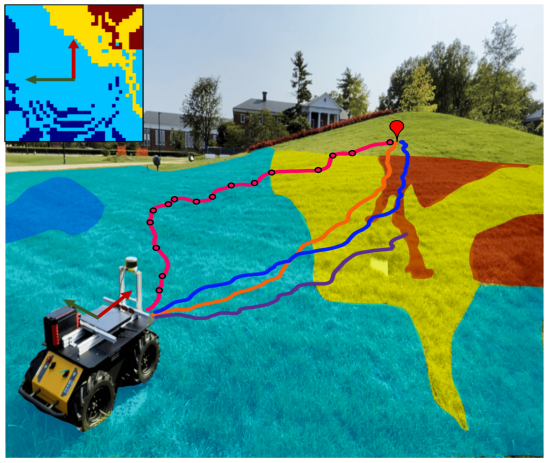

Fig. 1: Robot trajectories while navigating in an uneven terrain (elevation gain $\geq 3m$) using four different methods: our method TERP (pink), Ego-graph (orange), DWA (Blue) and velocities from the end-to-end Attn-DRL method (violet). TERP computes an attention mask to highlight reduced-stability regions for navigation, which are then used to compute and update navigation cost-maps continuously (e.g., cost-map computed at the robot's start location is shown in the top left) , with the unsafe regions where the robot's pitch and roll angle could exceed the stability limit are highlighted in yellow and red (superimposed on the terrain). TERP then generates waypoints (pink points) that are dynamically feasible and reachable by locally least-cost paths. Other methods navigate through unsafe regions with high elevation gradients which could lead to unstable robot orientations. TERP leads to trajectories with low elevation gradients (26.14% lower than prior navigation methods), leading to higher rate of reaching the goal (35.18% increase).

This work was supported in part by ARO Grants W911NF1910069, W911NF2110026 and U.S. Army Grant No. W911NF2120076. We acknowledge the support of the Maryland Robotics Center.

[1] Authors are with Dept. of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA. kasunw@umd.edu, asathyam@umd.edu

[2] Authors are with Dept. of Computer Science, University of Maryland, College Park, MD, USA. upatel22@umd.edu, dm@cs.umd.edu

Main contributions: We present TERP (Terrain Elevation-based Reliable Path Planning), a novel method for navigating a robot in a reliable and stable manner in uneven outdoor terrains. That is, our approach identifies unsafe regions (with high elevation gradients) and computes least-cost waypoints to avoid those regions. Our overall approach uses a continuous stream of local elevation maps, robot pose, and goal-related vectors as inputs. The novel components of our approach include:

- A DRL network (Attn-DRL) that uses a Convolutional Block Attention Module (CBAM) [16] and a reward function to learn appropriate terrain features in the local elevation map that indicate *reduced-stability regions* in the environment. The Attn-DRL network is primarily used for perception. After training, for any given input frame, we extract an attention mask from an intermediate feature vector in the Attn-DRL network. Our attention mask is not biased by human annotations and leads to an increase of 31.9% in the success rate for reaching the goal in high-elevation environments compared to not using attention, and a decrease of up to 75% in cumulative elevation gradient of the robot's trajectory.
- A 2D navigation cost-map computed using the extracted attention mask and a normalized elevation map of the environment. Our cost-map encodes a certain region's degree of planarity i.e., a measure of whether the robot's pitch and roll angles will be within safe limits, without the risk of flipping-over. Using the attention-based navigation cost-map leads to a 35.18% higher rate of reaching the goal in high elevation terrains when compared to prior methods.
- A method to compute waypoints on the cost-map that are dynamically feasible (i.e., satisfying the kinodynamic constraints of the robot) and reachable by locally least-cost trajectories. These trajectories are used as inputs to DWA-RL [17], a navigation scheme that computes dynamically feasible collision-free robot velocities. Our formulation results in a decrease in the Cumulative Elevation Gradient (CEG) of 26.14%. Decoupling perception to Attn-DRL network, and navigation to DWA-RL leads to improved overall performance, as compared to using one end-to-end network approach. (see Fig. 1 and 5c and d).

Our overall formulation has significantly lower execution times (takes one-fourth the time) than a state-of-the-art segmentation method [7] and, guarantees that the robot's trajectories are locally least-cost and dynamically feasible. We point the reader to [18] for a detailed version of our work.

II. Related Work

In this section, we discuss existing work on identifying terrain features and robot navigation on uneven terrains.

A. Identifying Terrain Features

Early works that addressed terrain identification utilized classification and modeling techniques [19], [20], [21]. These include Gaussian mixture models [22], Markov random fields [23], terrain mapping [24], and 3D environment reconstruction [25] with perception data from laser range finders [26] or from LiDARs, stereo cameras, and infrared cameras.

Recent methods have also constructed 3D maps, identified dynamic obstacles for unstructured scene navigation [27] and in walkways [28]. Due to the advent of deep neural networks (DNN), classification techniques have been used for terrain understanding [29], [30], [31], with some methods using robot-ground interaction data to train DNNs [32]. Most of the visual features extracted by DNNs and external geometric features can vary significantly from one environment to another. Hence, elevation estimation methods were introduced to generalize the terrain understanding problem.

Recent studies such as [12] highlight the importance of terrain elevation data to perform robust outdoor navigation tasks. [33] presents a method to perform terrain semantic segmentation and mapping using point clouds and RGB images for an autonomous excavator. In our work, we utilize an elevation map of the environment computed using point clouds from a 3D LiDAR.

B. Robot Navigation on Uneven Terrains

Early works in uneven terrain navigation addressed it using binary classification (obstacle vs. free space) [34], a continuous obstacle space [35], or potential fields [36] for high-speed rough terrain navigation.

Recent developments in deep learning have led to motion planners with high-dimensional input processing capabilities [11]. For instance, [37], [38] have incorporated on-board sensory inputs and prior knowledge of the environment to estimate navigation cost functions or cost-maps. In [39], a nonlinear geometric cost function has been trained to imitate human expert's behavior in uneven terrains. Similarly, an energy-cost model was proposed in [15] to generate energy-efficient robot paths on rough terrains.

Numerous end-to-end DRL methods have also been proposed for uneven terrain navigation [12], [40], [41]. These methods have used perception inputs such as elevation maps, the robot's orientation, and depth images to train an A3C-based network [40] or have used RGB images and point clouds to train a multi-modal fusion network [41]. In [4], the effectiveness of zero to local-range sensing was compared using a rainbow DRL-based local planner. All the aforementioned DRL methods were trained and tested only on simulated uneven terrains. Hence, transferring such methods into a real robot while maintaining comparable navigation performance can be challenging [14]. In our method, we decouple perception (to Attn-DRL network) and navigation to two separate modules (see Fig.2). This ensures that our method's real-world performance is comparable to its simulation performance.

III. Background

A. Notations, and Definitions

We highlight the symbols and notation we use in Table I. For our formulation, we consider a differential drive robot mounted with a 3D LiDAR with a range of r_{sense}, and a 360° field of view. We assume that the robot's start and goal locations are in stable regions i.e., the robot's roll and pitch angles are within a safe limit. We use 2D elevation maps ($\mathbf{E^t}$) (from processed point cloud data) obtained from a 3D LiDAR. $E^t(i,j)$ denotes elevation value at the (i,j) grid coordinate (see Fig. 4). All 2D data structures in our work ($\mathbf{E^t}, \mathbf{E_N^t}, \mathbf{A^t}, \mathbf{C^t}$) are regarded as $n \times n$ matrices or grids with the robot located at the center. We transform the indices (i,j) such that the robot's position corresponds to

Symbols	Definitions
$\mathbf{E^t}$	An $n \times n$ 2D local elevation map of the sensed environment at time instant t
c	Ground clearance of the robot
$\mathbf{E_N^t}$	Elevation map normalized based on ground clearance at time instant t
$\mathbf{A^t}$	Attention mask obtained from an intermediate feature vector at time instant t
$\mathbf{C^t}$	Navigation cost-map at time instant t
res	Resolution to convert index locations to real-world locations
r_{sense}	Radius (in meters) of the robot's sensing region
e_{max}, e_{min}	Max and min elevation values in $E^t(x, y)$ (in meters)
d_{goal}	Distance between the robot and its goal
α_{goal}	Angle between the robot's heading direction and the direction to the goal
$\alpha_{relative}$	Angle between the robot's current location and the goal w.r.t. the robot's start location
$\mathbf{g}$	Goal location relative to the robot
ϕ, θ	Roll and pitch angles of the robot

TABLE I: List of symbols used in our approach.

$(i, j) = (0, 0)$. We obtain real-world positions (x, y) from indices (i, j) as

$$(x, y) = res * (i, j). \tag{1}$$

B. Deep Reinforcement Learning

Our Attn-DRL network is based on the Deep Deterministic Policy Gradient (DDPG) algorithm [42] combined with Convolutional Neural Network layers and CBAM [16] (see Section III-C). We choose DDPG because it uses an actor-critic network, which leads to stable convergence to the optimal policy compared to policy-based or value-based methods. We briefly describe the input and action spaces of our DRL network here and our network architecture and reward function in Section IV. The perception input consists of a 2D normalized elevation map $\mathbf{E_N^t}$ (see Equation 3). The robot pose and goal related inputs include d_{goal}, α_{goal}, $\alpha_{relative}$, $|\phi|$, $|\theta|$, and the elevation gradient vector $\mathbf{h}$ in the robot's heading direction, defined as follows:

$$\mathbf{G} = ||\nabla \mathbf{E^t}||_2, \quad \mathbf{h} = G(n/2 : 1, 0). \tag{2}$$

where ∇ denotes the gradient operation resulting in an $n \times n$ $\mathbf{G}$ matrix and $G(n/2 : 1, 0)$ denotes all the values in rows $n/2$ to 1 in the 0^{th} column (according to our indexing conventions). Our Attn-DRL network is end-to-end with its action space being a linear and angular robot velocity pair.

Although DRL methods are useful for training certain robot behaviors in simulations, their performance degrades when transferred to real-world scenarios. In addition, the velocities computed by the end-to-end DRL network cannot guarantee dynamic feasibility or optimality (in terms of any metric). Therefore, we choose not to use the output velocities from Attn-DRL, and instead decouple perception and navigation. We use Attn-DRL purely for perception by extracting an attention mask from an intermediate feature vector F_{ref} in the network (see Section IV-B.1 and Fig. 3), and then computing a navigation cost-map. Computing waypoints and robot velocities for navigation is performed in a separate module (see Fig. 2).

C. Convolutional Block Attention Module

To highlight *reduced-stability regions* in the environment's elevation map, we use CBAM, a light-weight attention module that can be integrated with any CNN architecture [16]. Given a feature vector (say an output of a CNN) with a certain number of channels, CBAM sequentially applies attention modules along the channels and then along the spatial axis (different parts on the feature vector) to obtain a refined feature vector (see Fig. 3). This leads to our Attn-DRL network learning the features in the elevation map that are relevant for reliable navigation.

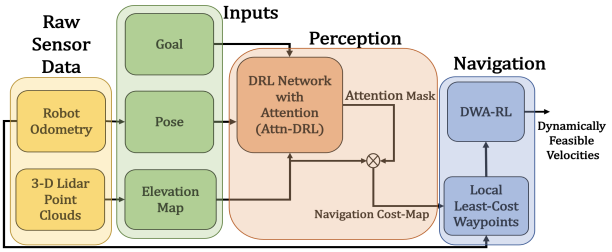

Fig. 2: TERP's overall system architecture. We extract an attention mask from the Attn-DRL network continuously for each frame of the input and compute the navigation cost-map. Our architecture decouples perception and navigation to different modules instead of using an end-to-end DRL network, resulting in improved performance. It also ensures that our method performs comparably in both simulated and real-world environments. The advantages of this formulation are highlighted in Section V-D.

IV. TERP: Terrain Elevation-based Robot Path Planning

We explain the three major stages of our method: 1. training the Attn-DRL network and extracting the attention mask, 2. computing a navigation cost-map and 3. generating locally least-cost trajectories.

A. Normalized Elevation Map

We use existing software packages to process raw point cloud data from a 3D LiDAR and obtain an $n \times n$ local elevation map $\mathbf{E^t}$ around the robot. We employ nearest neighbor interpolation to interpolate values for missing points based on nearby points.

A robot with higher ground clearance c can navigate through higher degrees of unevenness than a robot with lower ground clearance. We account for this difference by computing a normalized local elevation map as, $\mathbf{E_N^t} =$

$$\begin{cases} (E^t(i,j) - c) + 0.1(e_{max} - e_{min}) & \text{if } E^t(i,j) - c > 0, \\ (E^t(i,j) - c) & \text{Otherwise.} \end{cases} \tag{3}$$

B. Computing Attention Masks

We use $\mathbf{E_N^t}$ and the other input vectors (Section III-B) to train the Attn-DRL network. We explain its novel network architecture and the reward function used for training in simulated environments.

1) Network Architecture: Our network architecture (Fig. 3) has two branches. On the first branch, we use a 2D convolutional (conv-) layer of dimensions $40 \times 40 \times 8$ to process the input normalized elevation map (with $n = 40$). This outputs a feature vector with eight channels, which are passed on to the channel and spatial attention modules sequentially. Then, the channels are passed through another conv-layer. The channel and spatial attention modules weigh the features in the different channels and locations in each channel to identify *reduced-stability regions* relevant for reliable navigation. The resultant output is a refined feature vector (F_{ref}) with dimensions $40 \times 40 \times 8$, which is finally passed through a dimension reduction conv-layer.

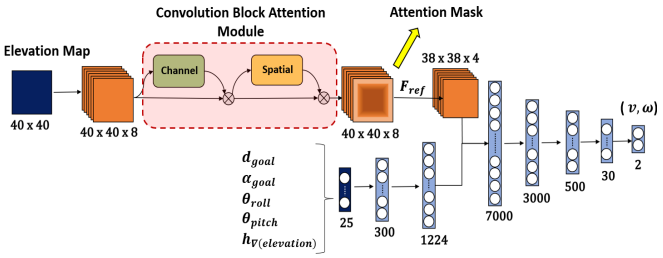

Fig. 3: The network architecture of our Attn-DRL network. The dark blue layers represent our normalized elevation map and the vector containing the robot's pose, goal, and elevation gradient-related vectors. The orange layers represent convolutional layers, and the light blue layers are fully-connected layers. The green and yellow blocks are the channel and spatial attention modules, respectively. The dimensions of each layer are mentioned next to it. Our network run in real-time with an execution time of ~ 0.08 seconds.

On the second branch, we concatenate the other inputs (d_{goal}, α_{goal}, $\alpha_{relative}$, $|\phi|$, $|\theta|$, and $\mathbf{h}$ vector) to obtain a 1D vector of size $n/2 + 5$ (25 in this case). This is processed using a fully connected layers of dimensions shown in Fig. 3. The outputs from the two branches are concatenated through several fully connected layers to finally obtain the robot's linear and angular velocities (which are used only for training and comparison). We use $ReLU$ activation in the hidden layers and $tanh$ activation at the output layer.

2) Reward Functions: The reward function is used to shape the velocities of the end-to-end attn-DRL network during training, i.e., to reward desirable and penalize undesirable robot actions/velocities. This in turn trains the intermediate feature vectors (such as F_{ref}) in the network to learn features that are relevant for reliable navigation (leading to high rewards). The total reward collected by the robot for performing an action at any instant is given as,

$$R_{tot} = \beta_1 R_{dist} + \beta_2 R_{head} + \beta_3 R_{stable} + \beta_4 R_{grad}. \quad (4)$$

Here, the different β are the weighing factors for the different reward components. R_{dist} and R_{head}, are the penalties for the robot moving away from its goal. They are defined as,

$$R_{dist} = -d_{goal}, \quad R_{head} = -|\alpha_{goal}|. \quad (5)$$

The novel terms of our reward function are R_{stable} and R_{grad} which encourage the network to learn features to maintain the robot's stability and avoid regions with high elevation gradients. R_{stable} is the reward for the robot maintaining stability (having low roll and pitch angles), and R_{grad} is the penalty for heading in a direction with a high elevation gradient. They are defined as,

$$R_{stable} = \cos^2\phi + \cos^2\theta, \quad R_{grad} = -\mathbf{w}\cdot\mathbf{h}, \quad (6)$$

where $\mathbf{w}$ is a weight vector with weights in ascending order, and $\mathbf{h}$ is the heading gradient vector defined in Equation 2. The $\mathbf{w}$ vector weighs elevation changes that are closer to the robot higher than the ones farther away.

3) Attention Mask Extraction: As mentioned in Section III-B, we extract an attention mask $\mathbf{A}$ from the refined feature vector F_{ref}. Since F_{ref}'s different channels are already weighted by the channel and spatial attention modules, the attention mask can be obtained by an unweighted summation along the channels. This is,

$$\mathbf{A}^t(i,j) = \sum_{i=1}^{8} F_{ref}(x,y,i). \quad (7)$$

The attention mask weighs elevation changes in the direction the robot is moving or turning higher than all other directions. This is depicted in Fig.4(b), where the attention mask corresponds to the normalized elevation map presented in Fig.4(a). The robot turns to its right to head towards its goal, and the high elevation changes on the right are highlighted in the attention mask.

C. Computing Navigation Cost-map

Since the normalized elevation map and the attention mask represent the environment's elevation values and the weights for the different regions at a time instant t, we utilize them to compute a navigation cost-map as,

$$\mathbf{C^t} = \mathbf{E_N^t} \odot \mathbf{A^t}. \quad (8)$$

Here $\odot$ represents element-wise multiplication. The computed costmap is shown in Fig.4(c). To completely avoid certain high-cost regions while navigating (costs higher than a certain threshold C_{max}), we perform the following operation.

$$C^t(i,j) = \infty \qquad if \qquad C^t(i,j) > C_{max}. \quad (9)$$

C_{max} is set not only based on the elevation in a region, but also on the robot's maximum torque capabilities. Therefore, all regions not reachable by the robot are assigned infinite cost.

D. Computing Least-cost Waypoints

Consider a local cost-map $\mathbf{C^t}$ and a goal location $\mathbf{g}$ that lies outside it. Choosing least-cost waypoints (some index (i, j)) inside $C^t(i,j)$ that lead the robot towards its goal is non-trivial. This is because the robot operates without any global knowledge of the environment. In addition, the chosen waypoint must be safely reachable by the robot. To address this, we formulate the following method.

Given a robot with its goal at $\mathbf{g}$, we consider an approximate circle $\mathcal{C}_{explore}$ in $\mathbf{C^t}$ centered at the robot with a radius $r_{explore} < r_{sense}$. Next, we consider an arc $\mathcal{A}_1^t$ on the circumference of $\mathcal{C}_{explore}$ that makes an angle of $\gamma_{explore}$ at the center with the goal vector $\mathbf{g}$ bisecting it (see Fig.4(d)). We use all locations $(i,j) \in \mathcal{A}_1^t$ as candidate waypoints. We then choose the set of *least-cost* waypoints as,

$$L_{min}^t = \{\underset{(i,j)\in\mathcal{A}_1^t}{\operatorname{argmin}} (cost((i,j)))\}. \quad (10)$$

Here, for $cost((i,j))$ is the cumulative cost of navigating to (i,j) from the robot's location $(0,0)$ in the cost-map by trajectories computed by Dijkstra's algorithm [43]. If multiple index locations exist in L_{min}^t, we choose the waypoint by transforming it to real-world coordinates (see Equation 1) as,

$$x^*, y^* = \underset{x,y\in(res\cdot L_{min}^t)}{\operatorname{argmin}} (dist((x,y),\mathbf{g})). \quad (11)$$

In the highly unlikely event that $L_{min}^t = \emptyset$ (implying all trajectories have infinite cost), we expand the arc $\mathcal{A}_1^t$ to $\mathcal{A}_2^t$, which makes an angle of $2\cdot\gamma_{explore}$ at $\mathcal{C}_{explore}$'s center. We then consider $(i,j) \in \mathcal{A}_2^t \setminus \mathcal{A}_1^t$ as candidate waypoints and repeat the procedure.

When the terrain is planar, the robot can safely have a larger exploration circle and choose waypoints on its circumference. This significantly reduces the number of waypoints that need to be computed before reaching the goal. Therefore, $r_{explore}$ is formulated as a function of the costs within the circle as,

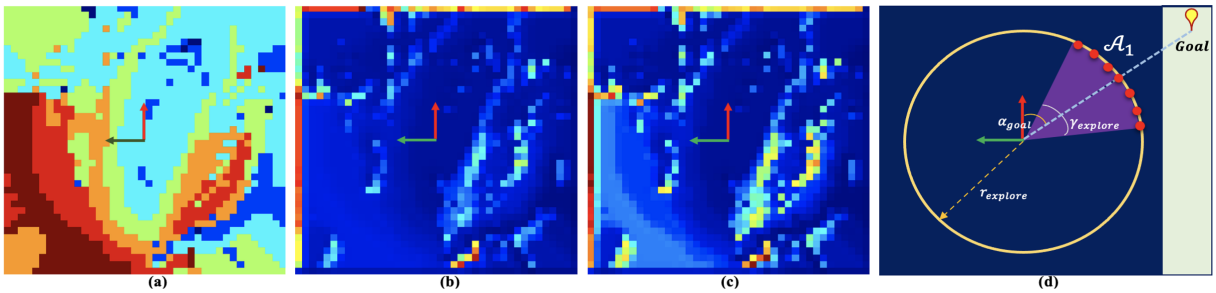

Fig. 4: (a) The normalized input elevation map $\mathbf{E_N^t}$. The colors range from light shades of blue (low elevation) to dark shades of red (high elevation) indicating elevation relative to the robot; (b) Attention mask extracted from Attn-DRL network. The light blue regions highlight the *reduced-stability regions* in the direction of the goal (see (d)). The attention mask $\mathbf{A^t}$ does not focus on high elevation regions in the left bottom since they are not relevant to the robot's current goal direction (top right); (c) Cost-map computed from $\mathbf{E_N^t}$ and $\mathbf{A^t}$. The lighter regions have high cost.; (d) Least-cost waypoint computation with the other parameters represented. The red dots represent the candidate waypoints on the arc $\mathcal{A}_1$.

$$r_{explore} = k_1 + k_2 \cdot \frac{1}{mean(C^t(i,j))}; \quad \forall i,j \in \mathcal{C}_{explore}. \tag{12}$$

where k_1 and k_2 are constants. If the costs within $\mathcal{C}_{explore}$ are low (implying a planar surface), the formulation expands the radius.

Proposition IV.1. *Our method always navigates the robot in a dynamically feasible trajectory with a low cumulative elevation gradient locally.*

Proof. This result follows from the fact that the waypoints computed using Equation 11 are reachable with the least trajectory cost locally. In addition, since our cost-map has infinite costs for regions that are unsafe and unreachable based on the robot's torque requirements, such regions are guaranteed to be avoided while computing trajectories. ■

E. Dynamic Feasibility

We use DWA-RL [17] to compute robot velocities to follow the least-cost trajectory computed by Dijkshtra's algorithm. DWA-RL guarantees that the velocities are collision-free and obey the robot's kinodynamic constraints. Therefore, the robot can also avoid collisions with dynamic obstacles in the environment. Fig. 2 shows how the components in our method are connected. Our approach decouples perception to our DRL network with the attention modules and the navigation to our local waypoint computation and DWA-RL. This architecture leads to superior performance and guarantees least-cost waypoints and the dynamic feasibility of the final velocities.

V. Results and Analysis

We explain our method's implementation in simulations and on a real-world robot. We then explain our evaluation metrics and discuss our inferences.

A. Implementation

Our Attn-DRL network is implemented in Pytorch. We use simulated uneven terrains with a Clearpath Husky robot created using ROS Melodic and Unity game engine to train the Attn-DRL network. The simulated Husky robot is mounted with a Velodyne VLP16 3D LiDAR. The network is trained in a workstation with an Intel Xeon 3.6 GHz processor and an Nvidia Titan GPU.

We replicate the simulated robot's setup on a real Husky robot. In addition to the LiDAR, the robot is mounted with a laptop with an Intel i9 CPU and an Nvidia RTX 2080 GPU. We use the Octomap [5] and Grid-map [44] ROS packages to obtain the elevation map $\mathbf{E^t}$ of size 40×40 using the point clouds from the Velodyne ROS package. Our network is computationally light enough to run real-time on the laptop along with the aforementioned ROS packages.

B. Evaluation Metrics

We use the following metrics to compare our method's navigation performance with the Dynamic Window Approach (DWA) [45], the ego-graph [46] method used in [4], and our end-to-end Attn-DRL network (without decoupling perception and navigation). The ego-graph method uses a cost function based on α_{goal} and the terrain's cumulative elevation gradient along a set of local trajectories to find the minimum cost trajectory. We add a distance-to-goal cost to this cost function and use it for additional comparison (ego-graph+).

Success Rate - The number of times the robot reached its goal while avoiding *reduced-stability regions* and collisions over the total number of attempts.

Cumulative Elevation Gradient (CEG) - The summation of the elevation gradients experienced by the robot along a trajectory (i.e., $||\nabla z_{rob}||$, where ∇z_{rob} represents the robot's elevation at any time instant).

Normalized Trajectory length - The robot's trajectory length normalized using the straight-line distance to the goal for both successful and unsuccessful trajectories.

Goal Heading Deviation - The cumulative angle difference between the robot's heading and the goal along a trajectory (i.e., $\sum_{i=1}^{t_{goal}} \alpha^i_{goal}$, where t_{goal} is the time to reach the goal).

C. Testing Scenarios

We evaluate our elevation based navigation framework and compare it with prior methods in four scenarios(see Fig.5).

- **Low-Elevation** - Maximum elevation gain $\leq 1m$.
- **Medium-Elevation** - Elevation gains between $1-2m$.
- **High-Elevation** - Maximum elevation gain $\geq 3m$.
- **City-curb** - A city environment with curbs higher than the robot's ground clearance that it must avoid.

D. Analysis and Comparison

The results are shown in Table II. We observe that all the methods perform well under low and medium elevation conditions in terms of success rate. However, TERP displays significant improvement in successfully reaching goals in highly-elevated environments. This is due to TERP's ability to navigate through the locally least-cost waypoints even

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

Fig. 5: Robot trajectories when navigating in different simulated and real-world uneven terrains using TERP (pink), TERP w/o attention (yellow), end-to-end Attn-DRL network (violet), ego-graph (orange), Ego-graph+ (green) and DWA (blue). (a) High-elevation; (b) City-curb; (c) real-world medium-elevation; (d) real-world curb. We observe that TERP computes trajectories with low elevation gradients in uneven terrains and is also able to handle challenging curb scenarios well. See [18] for detailed evaluations and results.

Metrics	Method	Low Elevation	Medium Elevation	High Elevation	City Curb
Success Rate (%)	DWA [45]	71	83	61	39
	Ego-graph [46]	69	79	62	29
	Ego-graph+	67	81	54	31
	End-to-end Attn-DRL	70	79	60	45
	TERP without Attention	76	83	67	47
	TERP with Attention	**82**	**85**	**73**	**62**
CEG	DWA [45]	0.68	1.34	2.49	0.298
	Ego-graph [46]	0.73	1.25	2.83	0.407
	Ego-graph+	0.72	1.23	2.64	0.380
	End-to-end Attn-DRL	0.65	1.24	2.15	0.084
	TERP without Attention	0.70	1.47	2.13	0.032
	TERP with Attention	**0.61**	**1.22**	**2.09**	**0.008**
Norm. Traj. Length	DWA [45]	1.15	**1.05**	1.17	1.22
	Ego-graph [46]	1.06	1.19	1.10	1.01
	Ego-graph+	**1.03**	1.20	**1.01**	1.04
	End-to-end Attn-DRL	1.42	1.18	0.98	1.12
	TERP without Attention	1.16	1.23	1.33	**0.91**
	TERP with Attention	1.39	1.21	1.24	1.55
Goal Heading Deviation	DWA [45]	49.24	**10.34**	**7.19**	13.18
	Ego-graph [46]	**29.56**	43.02	8.43	**12.14**
	Ego-graph+	33.26	69.23	43.19	24.16
	End-to-end Attn-DRL	83.51	65.42	71.86	46.58
	TERP without Attention	45.93	67.51	75.78	23.08
	TERP with Attention	74.46	53.98	89.11	161.91

TABLE II: Relative performance of our method versus other methods on various metrics. TERP consistently outperforms other methods in terms of success rate, and CEG even in challenging high-elevation and city-curb environments. The computed stable trajectories possess longer trajectory lengths and higher deviations from the goal depending on the environment. Our quantitative results also highlight the advantages of the included attention modules in our Attn-DRL network.

under high elevation conditions. Our end-to-end Attn-DRL method tries to plan trajectories along low-cost regions even though the minimum cost cannot be guaranteed. Hence, the success rate is lower and the CEG is higher than our TERP method. DWA sometimes avoids the high-elevations in front of the robot by treating them as obstacles. However, when the robot reaches an elevated terrain, DWA cannot recognize the elevations to avoid during navigation. In contrast, ego-graph-based methods select the optimal navigation velocities based on the front elevation gradients. These methods cannot maneuver the robot along the minimum-gradient trajectory because they have a discretized action space.

The CEG values in Table II imply that our method attempts to reach the goal by following the least-cost waypoints, which eventually result in minimum elevation gradients along the trajectory. However, this could lead to longer trajectories and higher deviations from the goal direction than other methods. The results in Table II validate that TERP can achieve better success rates while maintaining comparable trajectory lengths.

We observe that navigating in a city environment with curbs higher than the robot's ground clearance is a challenging task for all the methods. Comparatively small elevation gains of the curb regions could possibly lead to erroneous trajectories. However, TERP with attention outperforms all the other methods in terms of success rate (113% improvement w.r.t ego-graph) by attending to the critical elevation gradients near the curb (see Fig. 5b and d).

Ablation Study for the Attention Module: We compare navigation performance of TERP with and without the attention module. TERP without attention utilizes input elevation maps as the cost-map to find least-cost waypoints. Hence, trajectories generated by this approach select waypoints based on the locally minimum elevations. In contrast, TERP with attention follows the least-cost waypoints by attending to spatially critical elevation changes along the robot's trajectory (see Fig. 4c). Hence, the attention module improves the robot's spatial awareness to perform safe navigation tasks (See CEG in Table II, yellow and pink paths in Fig. 5).

Computational Complexity: We compare TERP's execution times with GANav [7], a state-of-the-art semantic segmentation method. TERP is computationally light enough to perform in real-time with execution times between 0.08-0.1 seconds (10-12.5Hz), while GANav takes 0.32-0.39 seconds on the same processing hardware.

Decoupled vs. end-to-end: Based on the end-to-end Attn-DRL network's trajectories in simulation (Fig. 5a and b) and in the real-world (Fig. 1 and 5c and d), we observe a significant degradation in performance. In contrast, TERP exhibits similar performance in both cases. Additionally, TERP also guarantees least-cost dynamic feasible waypoints, while the end-to-end network cannot.

VI. Conclusions, Limitations and Future Work

We present a novel formulation to reliably navigate a ground robot in uneven outdoor environments. Our hybrid architecture combines intermediate output from a DRL network with attention with input elevation data to obtain a cost-map to perform local navigation tasks. We generate locally least-cost waypoints on the obtained cost-map and integrate our approach with an existing DRL method that computes dynamically feasible robot velocities. We validate our approach in simulation and on real-world uneven terrains and compare it with the other navigation techniques on various navigation metrics.

Our framework has a few limitations which we expect to address in the future. The effective sensing point cloud percentage can be low especially near steep ditches due to the low reflectance of the LiDAR's rays. In such cases, additional depth sensors need to be integrated to achieve better perception. In addition, boundaries between different surfaces may not be sensed by the elevation map. Using a fusion of RGB image data and point clouds could help address this problem.

References

[1] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

[2] D. Silver, J. A. D. Bagnell, and A. T. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *International Journal of Robotics Research*, vol. 29, no. 12, pp. 1565 – 1592, October 2010.

[3] S. Siva, M. Wigness, J. G. Rogers, and H. Zhang, "Robot adaptation for generating consistent navigational behaviors over unstructured off-road terrain," 2021.

[4] S. Josef and A. Degani, "Deep reinforcement learning for safe local planning of a ground vehicle in unknown rough terrain," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6748–6755, 2020.

[5] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[6] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.

[7] T. Guan, D. Kothandaraman, R. Chandra, A. J. Sathyamoorthy, and D. Manocha, "Ganav: Group-wise attention network for classifying navigable regions in unstructured outdoor environments," 2021.

[8] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, and S. Savarese, "Gonet: A semi-supervised deep learning approach for traversability estimation," 2018.

[9] K. Viswanath, K. Singh, P. Jiang, S. P. B., and S. Saripalli, "Offseg: A semantic segmentation framework for off-road driving," 2021.

[10] A. Singh, K. Singh, and P. B. Sujit, "Offroadtranseg: Semi-supervised segmentation using transformers on offroad environments," 2021.

[11] F. G. Oliveira, A. A. Neto, D. Howard, P. Borges, M. F. Campos, and D. G. Macharet, "Three-dimensional mapping with augmented navigation cost through deep learning," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, pp. 1–21, 2021.

[12] D. C. Guastella and G. Muscato, "Learning-based methods of perception and navigation for ground vehicles in unstructured environments: a review," *Sensors*, vol. 21, no. 1, p. 73, 2021.

[13] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," 2020.

[14] H. Hu, K. Zhang, A. H. Tan, M. Ruan, C. Agia, and G. Nejat, "A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6569–6576, 2021.

[15] N. Ganganath, C.-T. Cheng, and K. T. Chi, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *IEEE transactions on industrial informatics*, vol. 11, no. 3, pp. 601–611, 2015.

[16] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[17] U. Patel, N. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dynamically feasible deep reinforcement learning policy for robot navigation in dense mobile crowds," 2020.

[18] K. Weerakoon, A. J. Sathyamoorthy, U. Patel, and D. Manocha, "Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning," 2021.

[19] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick, "Traversability classification using unsupervised on-line visual learning for outdoor robot navigation," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 518–525.

[20] B. Liu, M. Adams, and J. Ibanez-Guzman, "Multi-aided inertial navigation for ground vehicles in outdoor uneven environments," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 4703–4708.

[21] N. K. Govindaraju, M. C. Lin, and D. Manocha, "Quick-cullide: Fast inter-and intra-object collision culling using graphics hardware," in *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* IEEE, 2005, pp. 59–66.

[22] L. Nardi and C. Stachniss, "Actively improving robot navigation on different terrains using gaussian process mixture models," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4104–4110.

[23] C. Wellington, A. Courville, and A. Stentz, "A generative model of terrain for autonomous navigation in vegetation," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1287–1304, 2006.

[24] M. Agrawal, K. Konolige, and R. C. Bolles, "Localization and mapping for autonomous navigation in outdoor terrains : A stereo vision approach," in *2007 IEEE Workshop on Applications of Computer Vision (WACV '07)*, 2007, pp. 7–7.

[25] Y. Zhuang, W. Wang, H. Chen, and K. Zheng, "3d scene reconstruction and motion planning for an autonomous mobile robot in complex outdoor scenes," in *Proceedings of the 2010 International Conference on Modelling, Identification and Control*, 2010, pp. 692–697.

[26] K. M. Wurm, R. Kümmerle, C. Stachniss, and W. Burgard, "Improving robot navigation in structured outdoor environments by identifying vegetation from laser data," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1217–1222.

[27] Q. Yang, D. Qu, and F. Xu, "Mobile robot autonomous navigation and dynamic environmental adaptation in large-scale outdoor scenes," in *Intelligent Robotics and Applications*, H. Yu, J. Liu, L. Liu, Z. Ju, Y. Liu, and D. Zhou, Eds. Cham: Springer International Publishing, 2019, pp. 314–325.

[28] Y. Morales, A. Carballo, E. Takeuchi, A. Aburadani, and T. Tsubouchi, "Autonomous robot navigation in outdoor pedestrian walkways," *J. Field Robotics*, vol. 26, pp. 609–635, 08 2009.

[29] C. Bai, J. Guo, and H. Zheng, "Three-dimensional vibration-based terrain classification for mobile robots," *IEEE Access*, vol. 7, pp. 63 485–63 492, 2019.

[30] J. Zürn, W. Burgard, and A. Valada, "Self-supervised visual terrain classification from unsupervised acoustic feature learning," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 466–481, 2021.

[31] W. Wang, B. Zhang, K. Wu, S. A. Chepinskiy, A. A. Zhilenkov, S. Chernyi, and A. Y. Krasnov, "A visual terrain classification method for mobile robots' navigation based on convolutional neural network and support vector machine," *Transactions of the Institute of Measurement and Control*, p. 0142331220987917, 2021.

[32] F. Vulpi, A. Milella, R. Marani, and G. Reina, "Recurrent and convolutional neural networks for deep terrain classification by autonomous robots," *Journal of Terramechanics*, 2021.

[33] T. Guan, Z. He, D. Manocha, and L. Zhang, "TTM: Terrain traversability mapping for autonomous excavator navigation in unstructured environments," in *arXiv:2109.06250*, 2021.

[34] S. Laubach, J. Burdick, and L. Matthies, "An autonomous path planner implemented on the rocky 7 prototype microrover," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 1, 1998, pp. 292–297 vol.1.

[35] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[36] S. Shimoda, Y. Kuroda, and K. Iagnemma, "Potential field navigation of high speed unmanned ground vehicles on uneven terrain," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 2828–2833.

[37] D. Silver, J. A. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1565–1592, 2010.

[38] K. Zakharov, A. Saveliev, and O. Sivchenko, "Energy-efficient path planning algorithm on three-dimensional large-scale terrain maps for mobile robots," in *International Conference on Interactive Collaborative Robotics*. Springer, 2020, pp. 319–330.

[39] R. Valencia-Murillo, N. Arana-Daniel, C. López-Franco, and A. Y. Alanís, "Rough terrain perception through geometric entities for robot navigation," in *2nd International Conference on Advances in Computer Science and Engineering (CSE 2013)*. Atlantis Press, 2013, pp. 1–2.

[40] K. Zhang, F. Niroui, M. Ficocelli, and G. Nejat, "Robot navigation of environments with unknown rough terrain using deep reinforcement learning," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2018, pp. 1–7.

[41] A. Nguyen, N. Nguyen, K. Tran, E. Tjiputra, and Q. D. Tran, "Autonomous navigation in complex environments with deep multimodal fusion network," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5824–5830.

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[43] J.-C. Chen, "Dijkstra's shortest path algorithm," *Journal of formalized mathematics*, vol. 15, no. 9, pp. 237–247, 2003.

[44] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5. [Online]. Available: http://www.springer.com/de/book/9783319260525

[45] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, March 1997.

[46] A. Lacaze, Y. Moscovitz, N. DeClaris, and K. Murphy, "Path planning for autonomous vehicles driving over rough terrain," in *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intell*. IEEE, 1998, pp. 50–55.