

Self-reflective terrain-aware robot adaptation for consistent off-road ground navigation

The International Journal of
Robotics Research
2024, Vol. 0(0) 1–21
© The Author(s) 2024
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649231225243
journals.sagepub.com/home/ijr



Sriram Siva^{1,2} , Maggie Wigness², John G. Rogers², Long Quang² and Hao Zhang³ 

Abstract

Ground robots require the crucial capability of traversing unstructured and unprepared terrains and avoiding obstacles to complete tasks in real-world robotics applications such as disaster response. When a robot operates in off-road field environments such as forests, the robot's actual behaviors often do not match its expected or planned behaviors, due to changes in the characteristics of terrains and the robot itself. Therefore, the capability of robot adaptation for consistent behavior generation is essential for maneuverability on unstructured off-road terrains. In order to address the challenge, we propose a novel method of self-reflective terrain-aware adaptation for ground robots to generate consistent controls to navigate over unstructured off-road terrains, which enables robots to more accurately execute the expected behaviors through robot self-reflection while adapting to varying unstructured terrains. To evaluate our method's performance, we conduct extensive experiments using real ground robots with various functionality changes over diverse unstructured off-road terrains. The comprehensive experimental results have shown that our self-reflective terrain-aware adaptation method enables ground robots to generate consistent navigational behaviors and outperforms the compared previous and baseline techniques.

Keywords

Terrain-aware navigation, self-reflective adaptation, robot learning

Received 18 October 2022; Revised 16 November 2023; Accepted 12 December 2023

1. Introduction

Autonomous ground robots require the crucial capability of navigating over unstructured and unprepared terrains in off-road environments and avoiding obstacles to complete tasks in real-world applications such as search and rescue, disaster response, and reconnaissance (Kawatsuma et al., 2012; Park et al., 2017; Schwarz et al., 2017; Kuntze et al., 2012). When operating in field environments, ground robots need to navigate over a wide variety of unstructured off-road terrains with changing types, slope, friction, and other characteristics that cannot be fully modeled beforehand. In addition, ground robots operating over a long-period of time often experience changes in their own functionalities, including damages (e.g., failed robot joints and flat tires), natural wear and tear (e.g., reduced tire traction), and varying robot configurations (e.g., varying payload). As illustrated in Figure 1, such challenges make robot navigation in unstructured off-road environments a challenging problem. Thus, robot adaptation to changes in terrains and robot functionalities is essential to the success of off-road ground navigation.

The importance of robot adaptation to unstructured off-road terrains has been driving extensive research over recent

decades. Numerous approaches were designed based on the theories of learning and control, which has shown promising performance. However, the expected navigational behaviors generated by previous approaches are not always executed accurately by robots when navigating over unstructured off-road terrains. In other words, a robot's actual navigational behaviors typically do not align with the expected behaviors. This inconsistency primarily stems from two reasons. First, as ground robots traverse real-world off-road environments, they encounter terrains with diverse characteristics that cannot be pre-modeled, such as tall grass terrain with hidden rocks as illustrated in Figure 1. Second, ground robots may face negative effects or setbacks (Borges et al., 2019; Knight et al., 2001), which refer to adverse

¹Colorado School of Mines, Golden, CO, USA

²DEVCOM Army Research Laboratory (ARL), Adelphi, MD, USA

³University of Massachusetts Amherst, Amherst, MA, USA

Corresponding author:

Hao Zhang, Human-Centered Robotics Laboratory in the Manning College of Information and Computer Sciences, University of Massachusetts Amherst, 740 N Pleasant St, Amherst, MA 01003, USA.
Email: hao.zhang@cs.umass.edu

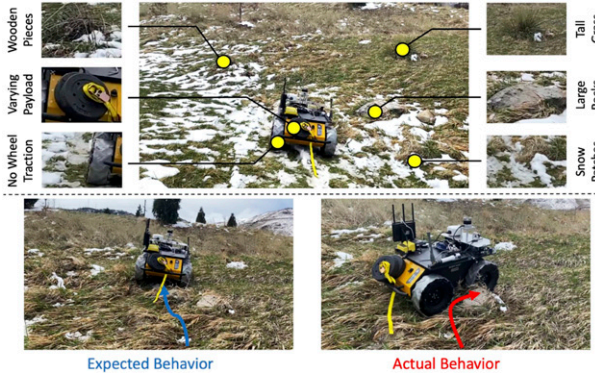


Figure 1. A motivating scenario of self-reflective terrain-aware robot adaptation for generating consistent robot navigational behaviors. When robots operate in unstructured off-road environments, the actual behaviors often do not match their expected behaviors, due to changes in the characteristics of terrains and the robots themselves. Therefore, the capability of robot adaptation for consistent behavior generation is essential for maneuverability over unstructured off-road terrains.

changes in a robot’s functionalities that increase the difficulty of achieving its expected behaviors. Examples of possible setbacks include reduced wheel traction and heavy payload. The previous approaches generally struggled to simultaneously adapt to changes in both terrains and setbacks.

To address these difficulties, we develop a novel method of self-reflective terrain-aware robot adaptation for consistent navigational behavior generation, which enables robots to accurately execute the expected behaviors through robot self-reflection while adapting to varying unstructured off-road terrains. In psychology, self-reflection is considered as the humans’ ability to modulate our behaviors by being aware of ourselves (Marcovitch et al., 2008; Smith, 2002; Ardel and Grunwald, 2018). In robotics, particularly in the context of robot learning, we use the term self-reflection to refer to the robot’s capability of adopting self-awareness (e.g., the robot knows that it moves slower than expected) to modulate navigational behavior generation. To enable robot self-reflection, our approach monitors the difference between the expected and actual behaviors, and then adapts the robot’s navigational behaviors accordingly to minimize this difference for consistent navigation. In contrast to adaptation that focuses on adjusting the robot’s navigational behaviors in response to external environmental changes only, self-reflective adaptation enables the robot to adapt to changes in both the environment and in itself without the need for explicit modeling. Moreover, our method learns representations of unstructured terrains from the robot’s multi-sensory observations, which are used to recognize unstructured terrains and generate navigational behaviors. Our approach also fuses the historical robot observations and robot behaviors that affect the robot’s current behavior generation. All the above components are integrated into a unified mathematical

framework of constrained regularized optimization with a theoretical convergence guarantee.

The contribution of this paper focuses on the introduction of the first method of self-reflective terrain-aware adaptation for ground robots to generate consistent navigational controls to traverse unstructured off-road environments. The specific novelties of this paper include:

- We introduce a novel method for terrain-aware robot adaptation to unstructured off-road terrains, which is able to simultaneously recognize discriminative terrain features and adaptively generate corresponding ground navigation behaviors.
- We introduce a novel idea of robot self-reflection and implement one of the first learning-based approaches that enable the ground robot to learn an offset control to compensate for the behavior differences in order to enhance consistent ground maneuverability.
- We implement an optimization algorithm to effectively address the constrained regularized optimization problem that is formulated to enable self-reflective terrain-aware robot adaptation, which holds a theoretical guarantee to converge to the global optimal solution.

Furthermore, as an experimental contribution, we perform extensive experiments and provide a comprehensive performance evaluation of terrain adaptation methods. We design a set of scenarios for ground robots with various functionality changes to traverse diverse individual and complex unstructured off-road terrains in natural field environments.

The remainder of this paper is structured as follows. We provide a review of related work in Section 2. Our proposed methods for self-reflective terrain-aware robot adaptation are discussed in Section 3. We derive the optimization algorithm in Section 4. After discussing experimental results in Section 5, we provide concluding thoughts in Section 6.

2. Related work

In this section, we provide a review of related work in the field of robot adaptation to unstructured terrains, including control- and learning-based methods. We also offer a review of recent work related to robot self-reflection.

2.1. Control methods for ground navigation

Methods based on classical control theories typically use predefined models to generate ground navigation behaviors in order to reach a desired goal position. Many early methods used a fuzzy logic-based implementation (Masmoudi et al., 2016; Seraji and Howard, 2002) without considering prior knowledge of robot dynamics to perform terrain navigation. System identification models were developed that use robot trajectory data to learn the robot’s dynamics and accordingly perform ground navigation (Nehmzow et al., 2007; Iqbal et al., 2021; Nazari et al., 2013).

Robot adaptation based upon control theory is typically performed by online tuning of control model's parameters using feedback computed from the model's observations. For example, trajectory optimization methods were designed for robot navigation through differential dynamics programming (DDP) (Aoyama et al., 2021, Rahman and Waslander, 2021) or iterative linear quadratic regulators (iLQR) (Fridovich-Keil et al., 2020, Truong et al., 2021). Both learn non-linear robot dynamics models from the robot's interactions with terrains. Closed-loop feedback control, such as the Model Predictive Control (MPC) (Ostafew et al., 2016; Zhu et al., 2018) and Model Predictive Path Integral (MPPI) (Williams et al., 2016), were also designed to improve control robustness to terrain noise and robot model error.

However, these methods generally lack the capability to integrate multi-modal sensory perception into their models. Additionally, they utilize a simplified representation of the world to create an occupancy map and perform path planning based solely on 3D geometric information (Li et al., 2019), without explicitly capturing terrain characteristics, such as terrain type. Several approaches also consider simple features such as slope angle and roughness to encode the terrain (Ostafew et al., 2016). Consequently, these methods fail to acquire a comprehensive representation of real-world terrain, particularly in unstructured environments. Furthermore, these methods lack self-awareness, further impeding their ability to adapt to setbacks encountered by the robot.

2.2. Learning-based methods for navigation

Learning-based techniques for navigational behavior generation and terrain adaptation have gained significant attention over the past years because of their effectiveness and flexibility (Thrun, 1998).

2.2.1. Pure data-driven learning. Methods based on pure data-driven learning train a machine learning model from the robot's observations or past experience only to generate navigational behaviors. Methods were designed to learn a robot behavior model using high-dimensional observations (Proctor et al., 2018; Williams et al., 2015). Learning from demonstration (LfD) techniques (Argall et al., 2009; Rana et al., 2018) were widely studied to transfer expert knowledge to mobile robots for ground navigation (Wigness et al., 2018; Wang et al., 2021). Terrain-aware navigation was also implemented by combining representation learning for terrain recognition along with apprenticeship learning to perform terrain adaptation (Siva et al., 2019). Navigational affordances were learned from experts over different terrains for navigation (Kahn et al., 2021). External disturbances were also considered to enable robust ground navigation in unstructured environments (Pereida and Schoellig, 2018; Jeong and Chwa, 2017; Altan and Hacıoğlu, 2020). Reinforcement learning is applied to enable robot ground navigation, which learns from the robot's own experience in a trial-and-error fashion when

navigating in the environment (Fathinezhad et al., 2016; Han et al., 2018; Kahn et al., 2018b). Rapid terrain-aware adaptation can be achieved by updating learned policies via inferring key terrain parameters Kumar et al. (2021). Lifelong reinforcement learning can improve the ground navigation performance by continuously optimizing the learned models (Kahn et al., 2018a; Wang et al., 2021; Sofman et al. 2006). Although pure data-driven learning methods offer greater flexibility in adapting to different environments, they often struggle with generalization to new and unseen environments. In addition, such methods lack the safety guarantees to be deployed in unstructured environments.

2.2.2. Machine-learning-based control. Methods based on machine-learning-based control assume a control model and learn the parameter values of the control model from the robot's observations or past experience (Wu et al., 2019; Yang et al., 2020; Duriez et al., 2017; Brunton and Kutz, 2019). Several approaches were implemented based upon dynamics mode decomposition (DMD) (Schmid, 2010; Tu, 2013) and sparse identification of non-linear dynamics (SINDy) (Mamakoukas et al., 2019; L'Erario et al., 2020) to learn control models for system identification and ground navigation (Wang and Noguchi, 2021; Kutz et al., 2016). Evolutionary algorithms were applied to estimate parameter values of the robot's control model in an online fashion for ground navigation (Cáceres et al., 2017; Ramirez et al., 1999). For robots with a high degree of freedom, methods were implemented to integrate the iterative linear quadratic regulators (iLQRs) with machine learning to explore control model parameters for adaptive navigation (Gillespie et al., 2018; Della Santina et al., 2017). Similarly, learning-based control methods were also designed that use a neural network as a functional approximator to represent the robot's dynamics model whose parameters can be updated in an online fashion (Nagariya and Saripalli, 2020). While machine-learning-based control methods are more computationally efficient, they do require knowledge of the robot's dynamics to perform effectively in unstructured environments.

Learning-based methods, in general, face challenges related to generalization, sample inefficiency, and the lack of safety guarantees, which are all essential for successful operation in unstructured environments. Additionally, these methods often fail to address inconsistencies caused by setbacks in both the terrain and the robot itself, such as reduced traction or increased payload. Overcoming these challenges is crucial to improving the performance and reliability of learning-based methods in unstructured environments.

2.3. Self-reflective robot adaptation

Self-reflective adaptation methods enable a robot to be aware of its own functionalities and capabilities to adapt the robot's behaviors (Rubilar et al., 2014; Zhang et al., 2020; Ishida, 2015). Notably, these methods do not require a pre-defined model of the robot, differentiating them from control theory-

based approaches. Early self-reflection methods were used for robots to perform self-supervised improvements while operating in outdoor environments (Zagal and Lipson, 2009). Methods were also developed to enable a robot to reflect on the past navigational experience to prioritize a set of policies in order to improve robot navigation (Altahhan, 2016). To adaptively respond to changes in the environment, a self-reflective risk-aware artificial cognitive model was developed (Zhang et al., 2016). Payload changes were considered in a self-reflection approach to update the robot's dynamics model for adaptive navigation planning (Demir and Sezer, 2019). Robot self-reflection was used in (Jin et al., 2020) to learn navigational behaviors in cluttered environment by reflecting on the perceptual data from 2D LiDAR sensors. Recently, a self-reflection method was also designed to iteratively update the kinematics model of a soft-robot based on its interaction with the environment (Hawkes et al., 2021). A deep learning method was implemented to perform reflection on the robot's trajectory from perceptual data in order to plan the robot's path toward the goal (Liu et al., 2021).

Most existing self-reflective robot adaptation approaches focused on adapting the robot's behaviors only to changes in the robot's functionalities. But they are generally not able to simultaneously adapt to external environments. Although our preliminary work (Siva et al., 2021) partially addresses this issue, it models terrain and robot functionality changes in a single loss function, and thus cannot distinguish the effects of terrains and robot functionalities on ground navigation. In this paper, we propose a new robot navigation method that explicitly and jointly characterize the terrain for navigation (i.e., terrain-awareness) and adapt to changes in the robot's functionalities (i.e., self-reflection).

3. Approach

In this section, we discuss our novel principled robot learning approach to enable ground robots to adapt their navigational behaviors to unstructured off-road terrains and perform self-reflection to generate consistent navigational behaviors. An overview of our approach is shown in Figure 2.

Notation: We denote scalars using lowercase italic letters (e.g., $m \in \mathbb{R}$), vectors using boldface lowercase letters (e.g., $\mathbf{m} \in \mathbb{R}^p$), matrices using boldface capital letters, for

example, $\mathbf{M} = \{\mathbf{m}_j^i\} \in \mathbb{R}^{p \times q}$ with its i -th row and j -th column denoted as $\mathbf{m}_j^i$ and $\mathbf{m}_j$, respectively. The Frobenius norm of matrix $\mathbf{M} \in \mathbb{R}^{p \times q}$ is computed as $\|\mathbf{M}\|_F = \sqrt{\sum_{i=1}^p \sum_{j=1}^q (\mathbf{m}_j^i)^2}$. We use boldface capital Euler script letters to denote tensors (i.e., 3D matrices), for example, $\mathcal{M} = \{\mathbf{m}_j^{i(k)}\} \in \mathbb{R}^{p \times q \times r}$. Unstacking tensor $\mathcal{M}$ along its height (p), width (q), and depth (r) provides slices of matrices $\mathbf{M}^i$, $\mathbf{M}_j$, and $\mathbf{M}^{(k)}$, respectively (Rabanser et al., 2017). The key variables used in our formulation are defined and briefly explained in Table 1.

3.1. Terrain-aware ground navigation

We use multiple sensors installed on a mobile ground robot to collect observations when the robot traverses unstructured terrains, including RGBD images, LiDAR scans, and IMU readings. At each time step t , we extract m types of features (such as visual features and local elevations) from the multi-sensory data. These features are concatenated into a feature vector denoted as $\mathbf{x}^{(t)} \in \mathbb{R}^d$, where $d = \sum_{i=1}^m d_i$ with d_i denoting the dimensionality of the i -th feature type. Features extracted from a sequence of c past time steps are stacked into a matrix as a terrain feature instance denoted as $\mathbf{X} = [\mathbf{x}^{(t)}; \dots; \mathbf{x}^{(t-c+1)}] \in \mathbb{R}^{d \times c}$. To train our approach, we collect a set of n feature instances that are obtained when the robot traverses over various terrains and denote this training set as a terrain feature tensor $\mathcal{X} = [\mathbf{X}_1, \dots, \mathbf{X}_n] \in \mathbb{R}^{d \times n \times c}$.

We denote the terrain types that are associated with $\mathcal{X}$ as $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_n] \in \mathbb{Z}^{l \times n}$, where $\mathbf{z}_i \in \mathbb{Z}^l$ denotes the vector of terrain types and l represents the number of terrain types. Each element $z_j^i \in \{0, 1\}$ indicates whether $\mathbf{X}_i$ has the j -th terrain type. During training, ground truth of the terrain types is provided by human experts. Given $\mathcal{X}$ and $\mathbf{Z}$, terrain recognition can be formulated as an optimization problem:

$$\begin{aligned} \min_{\mathcal{W}} \quad & \mathcal{L}(\mathcal{W} \otimes \mathcal{X} - \mathbf{Z}) \\ \text{s.t.} \quad & \mathcal{W} \otimes \mathcal{W}^\top = \mathcal{I} \\ & \mathcal{W} \otimes \mathcal{W} = \mathcal{W}^\top \otimes (\mathcal{W}^\top)^\top \end{aligned} \quad (1)$$

where $\mathcal{W} \in \mathbb{R}^{l \times d \times c}$ is a weight tensor used to encode the importance of each element in $\mathcal{X}$ toward estimating terrain

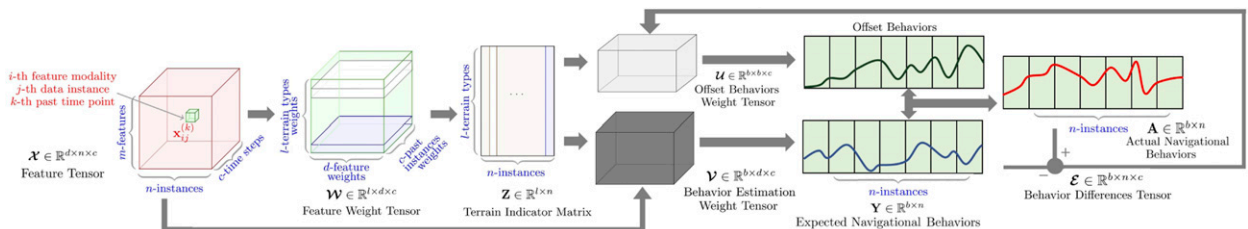


Figure 2. Overview of our proposed method for self-reflective terrain-aware robot adaptation to enable ground robots to adapt their navigational behaviors to unstructured off-road terrains and perform self-reflection to generate consistent navigational behaviors in the unified constrained regularized optimization framework.

types. Each tensor element $w_j^{i(k)} \in \mathcal{W}$ denotes the weight of the j -th terrain feature from the k -th past time step to recognize the i -th terrain type. $\mathbf{W}^i$, $\mathbf{W}_j$, and $\mathbf{W}^{(k)}$ are the slices of matrices that are obtained by unstacking $\mathcal{W}$ along its height, width, and depth, respectively. $\mathcal{I} \in \mathbb{R}^{l \times d \times c}$ is the identity tensor (Qi, 2017), which is used to mathematically define the orthogonality in tensors. The operator $\otimes$ denotes the vector Kronecker tensor product that performs the matrix-wise multiplication of the two tensors (Neudecker, 1969). In equation (1), the tensor product $\otimes$ takes each terrain feature instance $\mathbf{X}_i \in \mathcal{X}$, and multiplies it with the weight tensor $\mathcal{W}$.

The Log-Cosh loss, that is, $\mathcal{L}(\cdot) = \log(\cosh(\cdot))$ (Xu et al., 2020; Yu et al., 2023), is used in the objective function in equation (1) to encode the error of using terrain features $\mathcal{X}$ to recognize terrain types $\mathbf{Z}$, through the learning model that is parameterized by $\mathcal{W}$. The advantage of using the Log-Cosh loss (e.g., over the ℓ_2 -loss) is twofold. First, the Log-Cosh loss is robust to outliers, because a linear error becomes a much smaller value using the logarithmic scale; thus, it can reduce the learning bias over outliers. Because observations acquired by a robot when navigating over unstructured off-road terrain are usually noisy, the use of the Log-Cosh loss is desirable. Second, $\mathcal{L}(\cdot)$ provides a better optimization of $\mathcal{W}$ near the optimality. This is because, near the optimal value of $\mathcal{W}$, the objective value computed by $\mathcal{L}(\cdot)$ in equation (1) can change significantly when $\mathcal{W}$ shows small changes (Yu et al., 2023). The two constraints together in equation (1) are the necessary conditions for orthogonality. Orthogonality in tensors makes each slice of matrices that stack up a tensor to be full rank, which means that each slice of matrices is independent and orthogonal (or perpendicular) to each other and thus sharing the least similarity (Li et al., 2018). In our formulation, the

orthogonality makes $\mathbf{W}^i$, $i = 1, \dots, l$ that corresponds to all the terrain types to be independent of each other.

The first main novelty of this paper is that we propose a new method for terrain-aware navigation adaptation by joint terrain classification and behavior learning under a unified constrained optimization framework. This is achieved by projecting an observation's feature space into a terrain type space and then further projecting the terrain type space into a robot navigational behavior space.

Formally, we denote the robot's navigational behaviors as $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n] \in \mathbb{R}^{b \times n}$, where $\mathbf{y}_i \in \mathbb{R}^b$ represents the navigational behaviors that the robot is expected to execute after observing terrain instance $\mathbf{X}_i$, and b denotes the number of distinctive robot controls (e.g., linear and angular velocities). The expected behaviors can be obtained through demonstrations recorded from human experts as the experts control (i.e., teleoperate) ground robots to traverse various terrains. Similar to terrain classification, navigational behaviors are learned using a history of observations from the past c time steps, which allows our formulation to implicitly consider the dynamics of a ground robot. Then, the problem of terrain-aware navigation adaptation through joint terrain recognition and behavior generation can be formulated as:

$$\begin{aligned} \min_{\mathcal{W}, \mathcal{V}} \quad & \mathcal{L}(\mathcal{W} \otimes \mathcal{X} - \mathbf{Z}) + \mathcal{L}(\mathcal{V} \otimes \mathcal{W} \otimes \mathcal{X} - \mathbf{Y}) \\ \text{s.t.} \quad & \mathcal{W} \otimes \mathcal{W}^\top = \mathcal{I} \\ & \mathcal{W} \otimes \mathcal{W} = \mathcal{W}^\top \otimes (\mathcal{W}^\top)^\top \end{aligned} \quad (2)$$

where $\mathcal{V} \in \mathbb{R}^{b \times d \times c}$ is a weight tensor that indicates the importance of the input features $\mathcal{X}$ to generate navigational behaviors $\mathbf{Y}$, and each element $v_j^{i(k)}$ represents the weight of using j -th terrain feature from k -th past time step toward generating the i -th navigational behavior. We further adopt $\mathbf{V}^i$, $\mathbf{V}_j$ and $\mathbf{V}^{(k)}$ to denote the slices of matrices obtained by unstacking $\mathcal{V}$ along its height, width and depth, respectively.

The second term in the objective function in equation (2) is a loss that encodes the difference between the learning model and the demonstrated robot's navigational behaviors. This loss aims to learn a projection from a history of observations to terrain types, and then use the terrain types to generate the robot's behaviors, thus enabling terrain-awareness for our navigational behavior generation method. Furthermore, this loss function encodes the non-linear nature of the robot navigational behavior generation as a linear function of terrain features. This can be achieved due to two reasons. First, for short periods of time (e.g., $c \in [1, 20]$), under slower speeds (less than 2 m/s), the dynamics of the robot do not dramatically change (Tassa et al., 2012; Alothman and Gu, 2016) and thus the non-linearity in robot motion is not severe. Second, learning to solve non-linear tasks can be lifted to a linear space due to the high-dimension of the input, for example, as supported by the Koopman operator theory (Proctor et al., 2018).

Different terrain features capture different characteristics of unstructured terrains, for example, color, texture, and slope. These features typically have different contributions

Table 1. Definition of Key Variables Used in Our Learning-Based Self-Reflective Adaptation Approach.

Variables	Definition
$\mathbf{x}^{(t)} \in \mathbb{R}^d$	Terrain feature vector extracted at time t
$\mathbf{X} \in \mathbb{R}^{d \times c}$	Terrain feature instance matrix
$\mathcal{X} \in \mathbb{R}^{d \times n \times c}$	Feature tensor from n -different instances
$\mathbf{Z} \in \mathbb{Z}^{l \times n}$	Terrain indicator matrix corresponding to $\mathcal{X}$
$\mathbf{Y} \in \mathbb{R}^{b \times n}$	Expected robot navigational behaviors for $\mathcal{X}$
$\mathbf{A} \in \mathbb{R}^{b \times n}$	Actual robot navigational behaviors for $\mathcal{X}$
$\mathcal{E} \in \mathbb{R}^{b \times n \times c}$	Behavior difference tensor
$\mathcal{W} \in \mathbb{R}^{l \times d \times c}$	Weight tensor to encode terrain features
$\mathcal{V} \in \mathbb{R}^{b \times d \times c}$	Weight tensor to learn estimated behaviors
$\mathcal{U} \in \mathbb{R}^{b \times b \times c}$	Weight tensor to learn offset behaviors
d	Dimensionality of the feature vector
c	Past time steps used to acquire observations
n	Number of training instances
l	Number of terrain types
b	Number of robot behavior controls

towards terrain recognition and behavior generation. Similarly, features extracted in specific time steps can be more important than others. Thus, it is essential to identify the most discriminative features when multiple types of features from a history of time steps are used. To achieve this capability, we design a new regularization term called the behavior norm, which is mathematically defined as:

$$\|\mathcal{V} \otimes \mathcal{W}\|_B = \sum_{j=1}^m \sum_{k=1}^c \left\| \mathbf{V}_j^{(k)} \mathbf{W}_j^{(k)} \right\|_F \quad (3)$$

where $\mathbf{V}_i^{(k)}$ and $\mathbf{W}_i^{(k)}$ denote the slices of matrices of $\mathcal{V}$ and $\mathcal{W}$ obtained at the k -th past time step for the i -th type of features, respectively. This norm enforces sparsity between feature modalities obtained over the past c time steps in order to identify the most discriminative features and time steps during training.

Finally, the problem of terrain-aware behavior generation for terrain adaptation is formulated as:

$$\begin{aligned} \min_{\mathcal{W}, \mathcal{V}} & \mathcal{L}(\mathcal{W} \otimes \mathcal{X} - \mathbf{Z}) + \mathcal{L}(\mathcal{V} \otimes \mathcal{W} \otimes \mathcal{X} - \mathbf{Y}) + \lambda_1 \|\mathcal{V} \otimes \mathcal{W}\|_B \\ \text{s.t. } & \mathcal{W} \otimes \mathcal{W}^\top = \mathcal{I} \\ & \mathcal{W} \otimes \mathcal{W} = \mathcal{W}^\top \otimes (\mathcal{W}^\top)^\top \end{aligned} \quad (4)$$

where λ_1 is a trade-off hyperparameter to control the amount of regularization. The problem formulation in equation (4) allows for terrain-aware navigational behavior generation. However, ground robots operating in an unstructured environment over a long-period of time often experience changes in their own functionalities, such as damages (e.g., failed motors and flat tires), natural wear and tear (e.g., reduced tire traction), and changed robot configurations (e.g., varying payload). These negative effects, also called *setbacks*, reduce the effectiveness of robot navigation and cause the robot's actual navigational behaviors to deviate from the planned or expected behaviors.

3.2. Self-reflection for consistent navigational behavior generation

The second main novelty of this paper is to propose the idea of self-reflection for a ground robot to generate consistent navigational behaviors by adapting to changes in the robot's functionalities. In psychology, self-reflection is considered as the humans' capability of modulating our behaviors by being aware of ourselves (Marcovitch et al., 2008; Smith, 2002; Ardelt and Grunwald, 2018). We use the term self-reflection in robotics to refer to the robot's capability of adopting self-awareness (e.g., knowing it has slower speeds than expected) to modulate the terrain-aware behavior generation. Through self-reflection, we enable a ground robot to generate terrain-aware navigational behaviors that are also conditioned on its self-awareness. The robot can be self-aware in the way that it continuously monitors the difference between its actual and expected navigational behaviors, where this difference is caused by setbacks or negative effects. Then, our method adapts the parameters of

terrain-aware behavior generation according to the robot's self-awareness and computes an offset control to generate consistent navigational behaviors without explicitly modeling the robot setbacks.

Mathematically, we denote the robot's actual navigational behaviors as $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n] \in \mathbb{R}^{b \times n}$, where $\mathbf{a}_i \in \mathbb{R}^b$ is the actual navigational behaviors corresponding to the i -th instance $\mathbf{X}_i$. The actual behaviors can be estimated when the ground robot navigates over unstructured terrains using pose estimation methods such as SLAM or visual odometry (Shan and Englot, 2018; Shan et al., 2016). Due of setbacks, the robot's actual behaviors usually do not match its expected or planned navigational behaviors. At time t , we compute the difference in navigational behaviors over the past c time steps as $\mathbf{E} = [(\mathbf{a}^{(t-1)} - \mathbf{y}^{(t-1)}), \dots, (\mathbf{a}^{(t-c)} - \mathbf{y}^{(t-c)})] \in \mathbb{R}^{b \times c}$ and further denote the behavior differences for all instances $\mathcal{X}$ as a tensor $\mathcal{E} = [\mathbf{E}_1, \dots, \mathbf{E}_n] \in \mathbb{R}^{b \times n \times c}$.

Then, we introduce a novel loss function to encode self-reflection for achieving consistent terrain-aware navigational behavior generation as follows:

$$\mathcal{L}(\mathcal{U} \otimes \mathcal{W} \otimes \mathcal{E} - (\mathbf{A} - \mathbf{Y})) \quad (5)$$

where $\mathcal{U} = [\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(c)}] \in \mathbb{R}^{b \times b \times c}$ is the weight tensor with $\mathbf{U}^{(k)} \in \mathbb{R}^{b \times c}$ indicating the importance of the k -th past behavior difference $\mathbf{a}^{(k)} - \mathbf{y}^{(k)}$ to generate offset behaviors $\mathcal{U} \otimes \mathcal{W} \otimes \mathcal{E}$. Because the loss function in equation (5) encodes the error between the behavior differences and the generated offset behaviors, minimizing this loss function is equivalent to minimizing the behavior differences using the generated offset behaviors, or maximizing the consistency of the actual and expected behaviors. This allows our proposed approach to estimate the influence of past behavior differences and terrain types on the current behavior difference. By monitoring the differences over the past c time steps to improve navigational behavior at the current time step, this loss function allows our approach to be self-reflective and act as a closed-loop controller for consistent behavior generation while being terrain aware.

During robot navigation, historical data from the past time steps often contributes differently towards generating offset behaviors because of inertia. For example, a heavier payload increases the inertia of the robot; thus a longer time history needs to be considered for generating offset behaviors. Thus, we introduce an additional regularization term to learn the most informative behavior differences in the past time steps for efficient self-reflection, which is defined as:

$$\|\mathcal{U}\|_R = \sum_{k=1}^c \left\| \mathbf{U}^{(k)} \right\|_F \quad (6)$$

where $\mathbf{U}^{(k)}$ denotes the slice of matrices obtained from $\mathcal{U}$ at the k -th past time step.

Finally, integrating all components together, we formulate the problem of self-reflective terrain-aware robot

adaptation for generating consistent robot navigational behaviors as a regularized optimization problem in a unified mathematical framework:

$$\begin{aligned} \min_{\mathcal{W}, \mathcal{V}, \mathcal{U}} \quad & \mathcal{L}(\mathcal{W} \otimes \mathcal{X} - \mathcal{Z}) + \mathcal{L}(\mathcal{V} \otimes \mathcal{W} \otimes \mathcal{X} - \mathcal{Y}) \\ & + \mathcal{L}(\mathcal{U} \otimes \mathcal{W} \otimes \mathcal{E} - (\mathbf{A} - \mathbf{Y})) \\ & + \lambda_1 \|\mathcal{V} \otimes \mathcal{W}\|_B + \lambda_2 \|\mathcal{U}\|_R \\ \text{s.t.} \quad & \mathcal{W} \otimes \mathcal{W}^\top = \mathcal{I} \\ & \mathcal{W} \otimes \mathcal{W} = \mathcal{W}^\top \otimes (\mathcal{W}^\top)^\top \end{aligned} \quad (7)$$

where λ_2 , similar to λ_1 , is a hyperparameter to control the trade-off between the loss functions and regularization terms. An illustration of our complete approach for self-reflective terrain-aware robot adaptation is presented in Figure 2.

After computing the optimal values of the weight tensors $\mathcal{W}$, $\mathcal{U}$, and $\mathcal{V}$ according to Algorithm 1 in the training phase, a robot can apply our self-reflective terrain-aware adaptation method to generate consistent navigational behaviors during execution. At each time step t in the execution phase, the robot computes multi-model features $\mathbf{X}^{(t)}$ from observations obtained by its onboard sensors over the past c time steps. The robot also estimates the corresponding actual behaviors using pose estimation techniques (such as SLAM or visual odometry (Shan and Englot, 2018; Shan et al., 2016) and computes the matrix of the behavior differences $\mathbf{E}^{(t)}$. Then our approach can be used by the robot to generate self-reflective terrain-aware navigational behaviors as:

$$\mathbf{y}^{(t)} = \mathcal{V} \otimes \mathcal{W} \otimes \mathbf{X}^{(t)} + \mathcal{U} \otimes \mathcal{W} \otimes \mathbf{E}^{(t)} \quad (8)$$

The first term in equation (8) uses the learned model parameterized by $\mathcal{W}$ and $\mathcal{V}$ to project the input $\mathbf{X}^{(t)}$ in the feature space to the space of terrain types, which is then further projected to the behavior space. It generates the navigational behaviors that are aware of terrain types, which allows a ground robot to adapt its navigational behaviors to unstructured terrains. The second term uses the learned model parameterized by $\mathcal{W}$ and $\mathcal{U}$ to project a sequence of past behavior differences $\mathbf{E}^{(t)}$ to the navigational behavior space while considering the terrain types. This term provides offsets based on the monitoring of the differences between the expected and actual behaviors in order to compensate for the setbacks and improve consistency in navigational behaviors. Overall, the generation of self-reflective navigational behaviors at time step t relies on the terrain observations obtained up to time t and the calculated behavior differences, considering actual and expected behaviors up to time $t - 1$.

4. Optimization algorithm

In this section, we implement a new iterative algorithm, as shown in Algorithm 1, to compute the optimal solution to

the formulated constrained regularized optimization problem in equation (7). The optimization problem in equation (7) is not easy to solve, as it poses challenges due to the constraints imposed on the objective function, dependent variables, and the presence of the two non-smooth structured regularization terms. Since the two regularization terms cannot be differentiated at non-smooth points during optimization, second-order optimization algorithms (such as the Newton's or Secant's method) are not applicable.

In order to address the non-smooth terms and the dependent variables, we introduce a new optimization algorithm based on the alternating minimization scheme, which can be considered as a special version of gradient descent. While gradient descent provides a fundamental mathematical tool, deriving a closed-form solution with a convergence guarantee is not always feasible. Our algorithm offers such a closed-form solution and is guaranteed to converge to the global optima of the minimization problem in equation (7).

To implement the optimization algorithm, we begin with converting the constraints in equation (7) into a matrix form as:

$$\begin{aligned} \min_{\mathcal{W}, \mathcal{V}, \mathcal{U}} \quad & \mathcal{L}(\mathcal{W} \otimes \mathcal{X} - \mathcal{Z}) + \mathcal{L}(\mathcal{V} \otimes \mathcal{W} \otimes \mathcal{X} - \mathcal{Y}) \\ & + \mathcal{L}(\mathcal{U} \otimes \mathcal{W} \otimes \mathcal{E} - (\mathbf{A} - \mathbf{Y})) \\ & + \lambda_1 \|\mathcal{V} \otimes \mathcal{W}\|_B + \lambda_2 \|\mathcal{U}\|_R \\ \text{s.t.} \quad & \mathbf{W}^i (\mathbf{W}^i)^\top = \mathbf{I}^l; \forall i = 1, \dots, l \\ & \mathbf{W}_j (\mathbf{W}_j)^\top = \mathbf{I}_d; \forall j = 1, \dots, d \\ & \mathbf{W}^{(k)} (\mathbf{W}^{(k)})^\top = \mathbf{I}^{(c)}; \forall k = 1, \dots, c \end{aligned} \quad (9)$$

where $\mathbf{I}^l$, $\mathbf{I}_d$, and $\mathbf{I}^{(c)}$ are identity matrices of size l , d , and c . In equation (9), the constraints are written using a matrix form. This form still enforces orthogonality, the same as the tensor form in equation (7), as the matrix-form constraints make each column in each slice of matrices $\mathbf{W}^i$, $\mathbf{W}_j$, and $\mathbf{W}^{(k)}$ to be full rank. Thus, each slice of matrix in tensor $\mathcal{W}$ is also full rank and $\mathcal{W}$ is orthogonal.

Then we can rewrite the objective function in equation (9) using Lagrangian multipliers λ^l , λ_d and $\lambda^{(c)}$ as:

$$\begin{aligned} \min_{\mathcal{W}, \mathcal{V}, \mathcal{U}} \quad & \mathcal{L}(\mathcal{W} \otimes \mathcal{X} - \mathcal{Z}) + \mathcal{L}(\mathcal{V} \otimes \mathcal{W} \otimes \mathcal{X} - \mathcal{Y}) \\ & + \mathcal{L}(\mathcal{U} \otimes \mathcal{W} \otimes \mathcal{E} - (\mathbf{A} - \mathbf{Y})) + \lambda_1 \|\mathcal{V} \otimes \mathcal{W}\|_B \\ & + \lambda_2 \|\mathcal{U}\|_R + \sum_{i=1}^l \lambda^l \|\mathbf{W}^i (\mathbf{W}^i)^\top - \mathbf{I}^l\|_F \\ & + \sum_{j=1}^d \lambda_d \|\mathbf{W}_j (\mathbf{W}_j)^\top - \mathbf{I}_d\|_F \\ & + \sum_{k=1}^c \lambda^{(c)} \|\mathbf{W}^{(k)} (\mathbf{W}^{(k)})^\top - \mathbf{I}^{(c)}\|_F \end{aligned} \quad (10)$$

Each of the Lagrangian multipliers is designed to have a high value (i.e., $\lambda^l, \lambda_d, \lambda^{(c)} \gg 0$), such that even a

small variation in orthogonality in each slice of matrix from weight tensor $\mathcal{W}$ can result in a high cost. To compute the optimal weight tensor $\mathcal{W}$, we minimize equation (10) with respect to $\mathbf{W}^i$, $i = 1, \dots, l$, resulting in:

$$\begin{aligned} & \tanh(\mathcal{W} \otimes \mathcal{X} - \mathbf{Z}) \left((\mathbf{X}^i)^\top \mathbf{W}^i \right) \\ & + \tanh(\mathcal{V} \otimes \mathcal{W} \otimes \mathcal{X} - \mathbf{Y}) \left((\mathbf{X}^i \mathbf{V}^i)^\top \mathbf{W}^i \right) \\ & + \tanh(\mathcal{U} \otimes \mathcal{W} \otimes \mathcal{E} - (\mathbf{A} - \mathbf{Y})) \left((\mathbf{E}^i \mathbf{V}^i)^\top \mathbf{W}^i \right) \\ & + \left(\lambda_1 \mathbf{Q}^B + \sum_{i=1}^l \lambda^l \mathbf{Q}^l + \sum_{j=1}^d \lambda_d \mathbf{Q}_d + \sum_{k=1}^c \lambda^{(c)} \mathbf{Q}^{(c)} \right) \mathbf{W}^i = 0 \end{aligned} \quad (11)$$

where $\mathbf{Q}^B$, $\mathbf{Q}^l$, $\mathbf{Q}_d$ and $\mathbf{Q}^{(c)}$ denote block diagonal matrices that are dependent on the weight tensor $\mathcal{W}$. Mathematically, we express each element in the j -th column and k -th row in $\mathbf{Q}^B$ as $1 / \sum_{i=j}^m \sum_{k=1}^c \left\| \mathbf{V}_j^{(k)} \mathbf{W}_j^{(k)} \right\|_F$. Each of the i -th blocks in $\mathbf{Q}^l$ is given by $\mathbf{I}^i / \left\| \mathbf{W}^i (\mathbf{W}^i)^\top - \mathbf{I}^i \right\|_F$. The j -th diagonal block in $\mathbf{Q}_d$ is given as $\mathbf{I}_d / \left\| \mathbf{W}_j (\mathbf{W}_j)^\top - \mathbf{I}_d \right\|_F$ and each block diagonal element in $\mathbf{Q}^{(c)}$ is given by $\mathbf{I}^{(c)} / \left\| \mathbf{W}^{(k)} (\mathbf{W}^{(k)})^\top - \mathbf{I}^{(c)} \right\|_F$. Because each slice of matrix $\mathbf{W}^i$ and the block diagonal matrices $\mathbf{Q}^B$, $\mathbf{Q}^l$, $\mathbf{Q}_d$, $\mathbf{Q}^{(c)}$ are interdependent, we need an iterative algorithm to compute them.

We use the optimal slices of weight matrices $\mathbf{W}^i$ to solve weight tensors $\mathcal{V}$ and $\mathcal{U}$. Accordingly, we differentiate equation (10) with respect to $\mathbf{V}^i$ and $\mathbf{U}^i$, $i = 1, \dots, b$, resulting in:

$$\begin{aligned} & \tanh(\mathcal{V} \otimes \mathcal{W} \otimes \mathcal{X} - \mathbf{Y}) \left((\mathbf{W}^i \mathbf{X}^i)^\top \mathbf{V}^i - \mathbf{W}^i \mathbf{X}^i \mathbf{y}^i \right) \\ & + \lambda_1 \mathbf{O}^B = 0 \end{aligned} \quad (12)$$

and

$$\begin{aligned} & \tanh(\mathcal{U} \otimes \mathcal{W} \otimes \mathcal{E} - (\mathbf{A} - \mathbf{Y})) \left((\mathbf{W}^i \mathbf{E}^i)^\top \mathbf{U}^i - \mathbf{E}^i \right) \\ & + \lambda_2 \mathbf{P}^{(R)} = 0 \end{aligned} \quad (13)$$

where $\mathbf{O}^{(B)}$ and $\mathbf{P}^{(R)}$ represent block diagonal matrices that are dependent on $\mathcal{V}$ and $\mathcal{U}$, respectively. Mathematically, we calculate each element in the j -th column and k -th row in $\mathbf{O}^B$ as $1 / \sum_{i=j}^m \sum_{k=1}^c \left\| \mathbf{V}_j^{(k)} \mathbf{W}_j^{(k)} \right\|_F$. The k -th block diagonal element in $\mathbf{P}^{(R)}$ is calculated by $\mathbf{I}_{(k)} / \sum_{k=1}^c \left\| \mathbf{U}^{(k)} \right\|_F$. The optimal values of $\mathbf{V}^i$ and $\mathbf{U}^i$ are employed to update $\mathbf{W}^i$ in the next iteration. Similarly, $\mathbf{U}^i$ and $\mathbf{P}$ are interdependent, and thus an iterative algorithm is needed to update their values.

Algorithm 1: The implemented algorithm to solve the formulated constrained regularized optimization problem in Eq. (7).

Input : $\mathcal{X} \in \mathbb{R}^{d \times n \times c}$, $\mathbf{Y} \in \mathbb{R}^{b \times n}$, $\mathbf{A} \in \mathbb{R}^{b \times n}$, and $\mathcal{E} \in \mathbb{R}^{b \times n \times c}$

Output : The weight tensors $\mathcal{W} \in \mathbb{R}^{l \times d \times c}$, $\mathcal{V} \in \mathbb{R}^{b \times d \times c}$ and $\mathcal{U} \in \mathbb{R}^{b \times b \times c}$

- 1 Initialize $\mathcal{W}$, $\mathcal{V}$ and $\mathcal{U}$;
- 2 **while** not converge **do**
- 3 Calculate the block diagonal matrix $\mathbf{Q}^B$ with j -th column and k -th row element given as $\frac{\mathbf{I}_1}{\sum_{i=j}^m \sum_{k=1}^c \left\| \mathbf{V}_j^{(k)} \mathbf{W}_j^{(k)} \right\|_F}$;
- 4 Calculate the block diagonal matrix $\mathbf{Q}^l$ with i -th diagonal block given as $\frac{\mathbf{I}^l}{\left\| \mathbf{W}^i (\mathbf{W}^i)^\top - \mathbf{I}^i \right\|_F}$;
- 5 Calculate the block diagonal matrix $\mathbf{Q}_d$ with j -th diagonal block given as $\frac{\mathbf{I}_d}{\left\| \mathbf{W}_j (\mathbf{W}_j)^\top - \mathbf{I}_d \right\|_F}$;
- 6 Calculate the block diagonal matrix $\mathbf{Q}^{(c)}$ with k -th diagonal block given as $\frac{\mathbf{I}^{(c)}}{\left\| \mathbf{W}^{(k)} (\mathbf{W}^{(k)})^\top - \mathbf{I}^{(c)} \right\|_F}$;
- 7 Compute each slice of matrices $\mathbf{W}^i$ from Eq. (11);
- 8 Calculate the block diagonal matrix $\mathbf{O}^B$ with j -th column and k -th row element given as $\frac{\mathbf{I}_1}{\sum_{i=j}^m \sum_{k=1}^c \left\| \mathbf{V}_j^{(k)} \mathbf{W}_j^{(k)} \right\|_F}$;
- 9 Compute the matrix $\mathbf{V}^i$ from Eq. (12);
- 10 Calculate the block diagonal matrix $\mathbf{P}^{(R)}$ with k -th diagonal block given as $\frac{\mathbf{I}_2}{\sum_{k=t}^{t-c} \left\| \mathbf{U}^{(k)} \right\|_F}$;
- 11 Compute the matrix $\mathbf{U}^i$ from Eq. (13);
- 12 **return:** $\mathcal{W}$, $\mathcal{V}$ and $\mathcal{U}$

In the following, we prove that Algorithm 1 decreases the value of the objective function in equation (7) with each iteration and converges to the global optimal solution. But first, we present a lemma from Nie et al., 2010:

Lemma 1. Given any two matrices $\mathbf{A}$ and $\mathbf{B}$, the following inequality relation holds:

$$\left\| \mathbf{B} \right\|_F - \frac{\left\| \mathbf{B} \right\|_F^2}{2 \left\| \mathbf{A} \right\|_F} \leq \left\| \mathbf{A} \right\|_F - \frac{\left\| \mathbf{A} \right\|_F^2}{2 \left\| \mathbf{A} \right\|_F} \quad (14)$$

Theorem 1. Algorithm 1 iteratively converges to the global optimal solution, that is, the optimal values of weight tensors $\mathcal{W}$, $\mathcal{V}$ and $\mathcal{U}$ that minimizes the objective of the constrained regularized optimization problem in Eq. (7).

Proof. According to Step 7 in Algorithm 1, the value of $\mathbf{W}^i(s+1)$ is computed from $\mathcal{W}(s)$ in the s -th iteration by:

$$\begin{aligned}
\mathbf{W}^i(s+1) = & \mathcal{L}(\mathcal{W} \otimes \mathcal{X} - \mathbf{Z}) + \lambda_1 \text{Tr}(\mathbf{V}^i \mathbf{W}^{i\top}) \mathbf{Q}^B(s+1) (\mathbf{V}^i \mathbf{W}^i) \\
& + \lambda^l \text{Tr}(\mathbf{W}^i \mathbf{W}^{i\top} - \mathbf{I}^l) \mathbf{Q}^l(s+1) (\mathbf{W}^i \mathbf{W}^{i\top} - \mathbf{I}^l) \\
& + \lambda_d \text{Tr}(\mathbf{W}_j \mathbf{W}_j^\top - \mathbf{I}_d)^\top \mathbf{Q}_d(s+1) (\mathbf{W}_j \mathbf{W}_j^\top - \mathbf{I}^d) \\
& + \lambda^{(c)} \text{Tr}(\mathbf{W}^{(k)} \mathbf{W}^{(k)\top} - \mathbf{I}^{(c)}) \mathbf{Q}^{(c)}(s+1) \\
& (\mathbf{W}^{(k)} \mathbf{W}^{(k)\top} - \mathbf{I}^{(c)})
\end{aligned} \tag{15}$$

where the operator Tr denotes the trace of a matrix. From Step 9 in Algorithm 1, we obtain:

$$\begin{aligned}
\mathbf{V}^i(s+1) = & \mathcal{L}(\mathcal{V} \otimes \mathcal{W} \otimes \mathcal{X} - \mathbf{Y}) \\
& + \lambda_1 \text{Tr}(\mathbf{V}^i \mathbf{W}^i)^\top \mathbf{O}^B(s+1) (\mathbf{V}^i \mathbf{W}^i)
\end{aligned} \tag{16}$$

From Step 11 in Algorithm 1 we also obtain:

$$\mathbf{U}^i(s+1) = \mathcal{L}(\mathcal{U} \otimes \mathcal{W} \otimes \mathcal{E} - (\mathbf{A} - \mathbf{Y})) + \lambda_2 \text{Tr} \mathbf{U}^i \mathbf{P}(s+1) \mathbf{U}^i \tag{17}$$

Then, we derive that:

$$\begin{aligned}
& \mathcal{F}_1(s+1) + \mathcal{F}_2(s+1) + \mathcal{F}_3(s+1) + \lambda_1 \text{Tr}(\mathbf{V}^i \mathbf{W}^i)^\top \\
& \mathbf{Q}^B(s+1) (\mathbf{V}^i \mathbf{W}^i) \\
& + \lambda^l \text{Tr}(\mathbf{W}^i \mathbf{W}^{i\top} - \mathbf{I}^l)^\top \mathbf{Q}^l(s+1) (\mathbf{W}^i \mathbf{W}^{i\top} - \mathbf{I}^l) \\
& + \lambda_d \text{Tr}(\mathbf{W}_j \mathbf{W}_j^\top - \mathbf{I}_d)^\top \mathbf{Q}_d(s+1) (\mathbf{W}_j \mathbf{W}_j^\top - \mathbf{I}^d) \\
& + \lambda^{(c)} \text{Tr}(\mathbf{W}^{(k)} \mathbf{W}^{(k)\top} - \mathbf{I}^{(c)})^\top \mathbf{Q}^{(c)}(s+1) (\mathbf{W}^{(k)} \mathbf{W}^{(k)\top} - \mathbf{I}^{(c)}) \\
& + \lambda_1 \text{Tr}(\mathbf{V}^i \mathbf{W}^i)^\top \mathbf{O}^B(s+1) (\mathbf{V}^i \mathbf{W}^i) + \lambda_2 \text{Tr} \mathbf{U}^i \mathbf{P}(s+1) \mathbf{U}^i \\
& \leq \mathcal{F}_1(s) + \mathcal{F}_2(s) + \mathcal{F}_3(s) + \lambda_1 \text{Tr}(\mathbf{V}^i \mathbf{W}^i)^\top \mathbf{Q}^B(s) (\mathbf{V}^i \mathbf{W}^i) \\
& + \lambda^l \text{Tr}(\mathbf{W}^i \mathbf{W}^{i\top} - \mathbf{I}^l)^\top \mathbf{Q}^l(s) (\mathbf{W}^i \mathbf{W}^{i\top} - \mathbf{I}^l) \\
& + \lambda_d \text{Tr}(\mathbf{W}_j \mathbf{W}_j^\top - \mathbf{I}_d)^\top \mathbf{Q}_d(s) (\mathbf{W}_j \mathbf{W}_j^\top - \mathbf{I}^d) \\
& + \lambda^{(c)} \text{Tr}(\mathbf{W}^{(k)} \mathbf{W}^{(k)\top} - \mathbf{I}^{(c)})^\top \mathbf{Q}^{(c)}(s) (\mathbf{W}^{(k)} \mathbf{W}^{(k)\top} - \mathbf{I}^{(c)}) \\
& + \lambda_1 \text{Tr}(\mathbf{V}^i \mathbf{W}^i)^\top \mathbf{O}^B(s) (\mathbf{V}^i \mathbf{W}^i) + \lambda_2 \text{Tr} \mathbf{U}^i \mathbf{P}(s) \mathbf{U}^i
\end{aligned} \tag{18}$$

where:

$$\begin{aligned}
\mathcal{F}_1(s) &= \mathcal{L}(\mathcal{W}(s) \otimes \mathcal{X} - \mathbf{Z}) \\
\mathcal{F}_2(s) &= \mathcal{L}(\mathcal{V}(s) \otimes \mathcal{W}(s) \otimes \mathcal{X} - \mathbf{Y}) \\
\mathcal{F}_3(s) &= \mathcal{L}(\mathcal{U}(s) \otimes \mathcal{W}(s) \otimes \mathcal{E} - (\mathbf{A} - \mathbf{Y}))
\end{aligned}$$

After substituting $\mathbf{Q}^B$, $\mathbf{Q}^l$, $\mathbf{Q}_d$, $\mathbf{Q}^{(c)}$, $\mathbf{O}^B$ and $\mathbf{P}^R$ in equation (18), we obtain:

$$\begin{aligned}
& \mathcal{F}_1(s+1) + \mathcal{F}_2(s+1) + \mathcal{F}_3(s+1) \\
& + \lambda_2 \sum_{k=1}^c \frac{\|\mathbf{U}^{(k)}(s+1)\|_F^2}{2\|\mathbf{U}^{(k)}(s)\|_F} \\
& + \lambda_1 \sum_{i=j}^m \sum_{k=1}^c \frac{\|\mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s+1)\|_F^2}{2\|\mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s)\|_F} \\
& + \lambda^l \sum_{i=1}^l \frac{\|\mathbf{W}^i(s+1) (\mathbf{W}^i(s+1))^\top - \mathbf{I}^l\|_F^2}{\|2\mathbf{W}^i(s) (\mathbf{W}^i(s))^\top - \mathbf{I}^l\|_F} \\
& + \lambda_d \sum_{j=1}^d \frac{\|\mathbf{W}_j(s+1) (\mathbf{W}_j(s+1))^\top - \mathbf{I}_d\|_F^2}{\|2\mathbf{W}_j(s) (\mathbf{W}_j(s))^\top - \mathbf{I}_d\|_F} \\
& + \lambda^{(c)} \sum_{k=1}^c \frac{\|\mathbf{W}^{(k)}(s+1) (\mathbf{W}^{(k)}(s+1))^\top - \mathbf{I}^{(c)}\|_F^2}{\|2\mathbf{W}^{(k)}(s) (\mathbf{W}^{(k)}(s))^\top - \mathbf{I}^{(c)}\|_F} \\
& \leq \mathcal{F}_1(s) + \mathcal{F}_2(s) + \mathcal{F}_3(s) + \lambda_2 \sum_{k=1}^c \frac{\|\mathbf{U}^{(k)}(s)\|_F^2}{2\|\mathbf{U}^{(k)}(s)\|_F}
\end{aligned} \tag{19}$$

$$\begin{aligned}
& + \lambda_1 \sum_{i=j}^m \sum_{k=1}^c \frac{\|\mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s)\|_F^2}{2\|\mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s)\|_F} \\
& + \lambda^l \sum_{i=1}^l \frac{\|\mathbf{W}^i(s) (\mathbf{W}^i(s))^\top - \mathbf{I}^l\|_F^2}{\|2\mathbf{W}^i(s) (\mathbf{W}^i(s))^\top - \mathbf{I}^l\|_F} \\
& + \lambda_d \sum_{j=1}^d \frac{\|\mathbf{W}_j(s) (\mathbf{W}_j(s))^\top - \mathbf{I}_d\|_F^2}{\|2\mathbf{W}_j(s) (\mathbf{W}_j(s))^\top - \mathbf{I}_d\|_F} \\
& + \lambda^{(c)} \sum_{k=1}^c \frac{\|\mathbf{W}^{(k)}(s) (\mathbf{W}^{(k)}(s))^\top - \mathbf{I}^{(c)}\|_F^2}{\|2\mathbf{W}^{(k)}(s) (\mathbf{W}^{(k)}(s))^\top - \mathbf{I}^{(c)}\|_F}
\end{aligned}$$

Using Lemma 1, we obtain the following inequalities in equations (20)–(24). From $\mathbf{Q}^B$ and $\mathbf{O}^B$, we obtain:

$$\begin{aligned}
& \sum_{i=1}^m \sum_{k=1}^c \left(\left\| \mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s+1) \right\|_F - \frac{\left\| \mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s+1) \right\|_F^2}{2\left\| \mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s) \right\|_F} \right) \\
& \leq \sum_{i=1}^m \sum_{k=1}^c \left(\left\| \mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s) \right\|_F - \frac{\left\| \mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s) \right\|_F^2}{2\left\| \mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s) \right\|_F} \right)
\end{aligned} \tag{20}$$

From $\mathbf{Q}^l$, we obtain:

$$\begin{aligned}
& \sum_{i=1}^m \left(\left\| \mathbf{W}^i(s+1) \mathbf{W}^{i\top}(s+1) - \mathbf{I}^l \right\|_F - \frac{\left\| \mathbf{W}^i(s+1) \mathbf{W}^{i\top}(s+1) - \mathbf{I}^l \right\|_F^2}{2 \left\| \mathbf{W}^i(s) \mathbf{W}^{i\top}(s) - \mathbf{I}^l \right\|_F} \right) \\
& \leq \sum_{i=1}^m \left(\left\| \mathbf{W}^i(s) \mathbf{W}^{i\top}(s) - \mathbf{I}^l \right\|_F - \frac{\left\| \mathbf{W}^i(s) \mathbf{W}^{i\top}(s) - \mathbf{I}^l \right\|_F^2}{2 \left\| \mathbf{W}^i(s) (\mathbf{W}^{i\top}(s)) - \mathbf{I}^l \right\|_F} \right)
\end{aligned} \tag{21}$$

From $\mathbf{Q}_d$, we obtain:

$$\begin{aligned}
& \sum_{j=1}^d \left(\left\| \mathbf{W}_j(s+1) (\mathbf{W}_j(s+1))^\top - \mathbf{I}_d \right\|_F - \frac{\left\| \mathbf{W}_j(s+1) (\mathbf{W}_j(s+1))^\top - \mathbf{I}_d \right\|_F^2}{2 \left\| \mathbf{W}_j(s) (\mathbf{W}_j(s))^\top - \mathbf{I}_d \right\|_F} \right) \\
& \leq \sum_{j=1}^d \left(\left\| \mathbf{W}_j(s) (\mathbf{W}_j(s))^\top - \mathbf{I}_d \right\|_F - \frac{\left\| \mathbf{W}_j(s) (\mathbf{W}_j(s))^\top - \mathbf{I}_d \right\|_F^2}{2 \left\| \mathbf{W}_j(s) (\mathbf{W}_j(s))^\top - \mathbf{I}_d \right\|_F} \right)
\end{aligned} \tag{22}$$

From $\mathbf{Q}^{(c)}$, we obtain:

$$\begin{aligned}
& \sum_{k=1}^c \left(\left\| \mathbf{W}^{(k)}(s+1) (\mathbf{W}^{(k)}(s+1))^\top - \mathbf{I}^{(c)} \right\|_F - \frac{\left\| \mathbf{W}^{(k)}(s+1) (\mathbf{W}^{(k)}(s+1))^\top - \mathbf{I}^{(c)} \right\|_F^2}{2 \left\| \mathbf{W}^{(k)}(s) (\mathbf{W}^{(k)}(s))^\top - \mathbf{I}^{(c)} \right\|_F} \right) \\
& \leq \sum_{k=1}^c \left(\left\| \mathbf{W}^{(k)}(s) (\mathbf{W}^{(k)}(s))^\top - \mathbf{I}^{(c)} \right\|_F - \frac{\left\| \mathbf{W}^{(k)}(s) (\mathbf{W}^{(k)}(s))^\top - \mathbf{I}^{(c)} \right\|_F^2}{2 \left\| \mathbf{W}^{(k)}(s) (\mathbf{W}^{(k)}(s))^\top - \mathbf{I}^{(c)} \right\|_F} \right)
\end{aligned} \tag{23}$$

From $\mathbf{P}^{(R)}$, we obtain:

$$\begin{aligned}
& \sum_{k=1}^c \left(\left\| \mathbf{U}^{(k)}(s+1) \right\|_F - \frac{\left\| \mathbf{U}^{(k)}(s+1) \right\|_F^2}{2 \left\| \mathbf{U}^{(k)}(s) \right\|_F} \right) \\
& \leq \sum_{k=1}^c \left(\left\| \mathbf{U}^{(k)}(s) \right\|_F - \frac{\left\| \mathbf{U}^{(k)}(s) \right\|_F^2}{2 \left\| \mathbf{U}^{(k)}(s) \right\|_F} \right)
\end{aligned} \tag{24}$$

Then, adding equations (20)–(24) to equation (19) on both sides results in the following inequality:

$$\begin{aligned}
& \mathcal{F}_1(s+1) + \mathcal{F}_2(s+1) + \mathcal{F}_3(s+1) \\
& + \lambda_2 \sum_{k=1}^c \left\| \mathbf{U}^{(k)}(s+1) \right\|_F \\
& + \lambda_1 \sum_{i=j}^m \sum_{k=1}^c \left\| \mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s+1) \right\|_F \\
& + \lambda^l \sum_{i=1}^l \left\| \mathbf{W}^i(s+1) \mathbf{W}^{i\top}(s+1) - \mathbf{I}^l \right\|_F \\
& + \lambda_d \sum_{j=1}^d \left\| \mathbf{W}_j(s+1) \mathbf{W}_j^\top(s+1) - \mathbf{I}_d \right\|_F \\
& + \lambda^{(c)} \sum_{k=1}^c \left\| \mathbf{W}^{(k)}(s+1) \mathbf{W}^{(k)\top}(s+1) - \mathbf{I}^{(c)} \right\|_F \\
& \leq \mathcal{F}_1(s) + \mathcal{F}_2(s) + \mathcal{F}_3(s) + \lambda_2 \sum_{k=1}^c \left\| \mathbf{U}^{(k)}(s) \right\|_F \\
& + \lambda_1 \sum_{i=j}^m \sum_{k=1}^c \left\| \mathbf{V}_j^{(k)}(s) \mathbf{W}_j^{(k)}(s) \right\|_F \\
& + \lambda^l \sum_{i=1}^l \left\| \mathbf{W}^i(s) \mathbf{W}^{i\top}(s) - \mathbf{I}^l \right\|_F + \lambda_d \sum_{j=1}^d \left\| \mathbf{W}_j(s) \mathbf{W}_j^\top(s) - \mathbf{I}_d \right\|_F \\
& + \lambda^{(c)} \sum_{k=1}^c \left\| \mathbf{W}^{(k)}(s) \mathbf{W}^{(k)\top}(s) - \mathbf{I}^{(c)} \right\|_F
\end{aligned} \tag{25}$$

This inequality in equation (25) proves that the objective value is decreased in each iteration. The Log-Cosh losses in equation (9) are convex (Gangal et al., 2007) and the Frobenius norms in equation (9) are also convex (Negahban and Wainwright, 2012). Thus, as a sum of convex functions, the objective function in equation (9) is also convex. Therefore, Algorithm 1 is guaranteed to converge to the global optimal values of weight tensors $\mathcal{W}$, $\mathcal{V}$, and $\mathcal{U}$ that solve the constrained regularized optimization problem in equation (7).

Time Complexity. As the formulated optimization problem in equation (7) is convex, Algorithm 1 converges fast (e.g., within tens of iterations only). In each iteration of our algorithm, computing Steps 3, 4, 5, 6, 8, and 10 is trivial. Steps 7, 9, and 11 can be computed by solving a system of linear equations with quadratic complexity.

5. Experiments

In this section, we first discuss the implementation details of our approach and the experimental setup used. Then we analyze the experimental results obtained by our method and compare them with the previous state-of-the-art methods in multiple real-world off-road ground navigation scenarios.

5.1. Experimental setups

We evaluate the proposed approach using Clearpath Husky ground robots, which navigate on various unstructured off-road terrains. The robots operate on an Intel 4.3 GHz i7 processor and run Ubuntu 18.04 and Robot Operating System (ROS) Melodic. Furthermore, these robots are equipped with multiple exteroceptive and proprioceptive sensors. The exteroceptive sensors consist of an Intel RealSense D435 color-depth camera and an Ouster OS1-64 LiDAR, which are used to observe the terrain. The proprioceptive sensors include a Microstrain 3DM-GX5-25 Internal Measurement Unit (IMU) and wheel encoders, providing measurements of the robot’s states during navigation on these terrains. To ensure consistent observations, the data is linearly interpolated to a frequency of 20 Hz.

5.1.1. Algorithmic implementation. The optimization algorithm introduced in Section 4 is carried out during the offline training phase, and no online optimization is used during the execution phase. We provide the terrain feature tensor $\mathcal{X}$, the terrain indicator matrix $\mathcal{Z}$, the expected behaviors $\mathcal{Y}$, the actual behaviors $\mathcal{A}$, and the behavior difference tensor $\mathcal{E}$ as input variables to our optimization algorithm. Our algorithm then alternatively optimizes weight tensors $\mathcal{W}$, $\mathcal{U}$, and $\mathcal{V}$, over various iterations and converges to the global optimal solution according to equation (25). We use C++ in ROS Melodic on Ubuntu 18.04 OS to implement the optimization algorithm based on the mathematical derivation in equations (9)–(25).

Our optimization solver is derived based on the mathematical optimization scheme called alternating minimization. To solve an optimization problem with a convex objective function, this solver holds a theoretical guarantee to converge to the global optima solution (Boyd and Vandenberghe, 2004). For practical considerations, to ensure convergence to obtain the global optimal solution, our proposed algorithm incorporates several key elements of convex optimization during the training process, including initialization, hyperparameter selection, and termination condition. We choose random initialization weights to prevent the problems associated with vanishing or exploding gradients. Regularization plays a crucial role to train learning models in order to reduce overfitting, facilitate convergence, and generally encode our prior knowledge of the structure of the data. Therefore, the hyperparameter values (e.g., λ_1 and λ_2 in our approach) must be carefully selected to balance between the losses and the regularization terms. If the hyperparameters have too large values, the optimization algorithm may oscillate around the global minimum. On the other hand, setting the hyperparameter values too low can undermine the effectiveness of regularization, thus potentially causing overfitting. In our implementation, cross-validation is used to determine the best hyperparameter values (Figure 8(b)). Finally, we carefully design termination conditions to ensure convergence to a global solution. During the training process, the algorithm monitors the continuous decrease in the cost of the objective function. Once the cost reaches a pre-defined threshold value and remains unchanged for several consecutive steps, it is assumed to be a convergence point.

5.1.2. Training and execution procedures. We train our approach on a dataset recorded as a human demonstrator drives the robot over different types of terrain. This recorded data includes the robot’s observations from IMU, RGBD camera, and LiDAR sensors. Specifically, we extract multiple visual features from color images to represent unstructured terrains, which includes Histogram of Oriented Gradients (HOG) to represent the shape of the terrains (Dalal and Triggs, 2005) and local binary patterns (LBPs) to represent the texture of the terrain (Ahonen et al., 2006). We also compute a robot centric elevation map (Fankhauser et al., 2014) of the terrain around the robot using LiDAR data. This elevation is used to represent the terrain characteristics such as the slope, step height, and terrain normals and is local to the robot. Moreover, we concatenate the IMU readings and robot wheel odometry data into a feature. The various features used in our experiments are demonstrated in Figure 3. All of these extracted features from the robot’s different sensors are concatenated to form the feature vector and is used as input to our approach. Furthermore, the linear and angular velocities commanded by the human demonstrator are vectorized to form the expected behaviors. The actual navigational behaviors are estimated using the LiDAR-based SLAM technique (Shan and Englot, 2018),

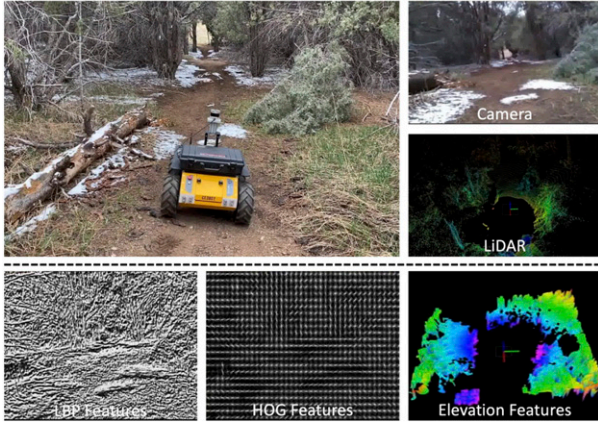


Figure 3. Multi-modal features extracted from the robot's onboard camera and LiDAR to characterize unstructured off-road terrains in our experiments.

that is, the vectorized linear and angular velocities obtained from SLAM.

The training data includes approximately 30,000 instances from nearly 8 h of driving. Each instance of the dataset includes the inputs from past c -time steps. Using this dataset, the optimization algorithm takes roughly 4 h to converge to the optimal solution when $c = 6$. However, increasing c exponentially increases the time of convergence. For example, using $c = 16$ increases the runtime of the algorithm to nearly 26 h. During training, we observe that the best results for consistent off-road ground navigation are obtained for the following value of hyperparameters $\lambda_1 = 1$ and $\lambda_2 = 0.1$ and $c = 6$ (see Section 5.5 for more details). Accordingly, these hyperparameter values are then used for all the experiments. In the execution phase, our approach uses the optimal weight tensors learned during the training phase without additional online learning/optimization.

In the experiments, we quantitatively evaluate our approach using four evaluation metrics, with a lower value indicating a better performance:

- *Failure Rate* is defined as the number of times that the robot fails to complete the navigation task within a set of experimental trails. A failure happens when the robot is stuck on the terrain or by an obstacle, or when the robot flips over.
- *Traversal Time* is defined as the average time for the robot to complete the navigation task over the given terrain.
- *Inconsistency* is defined as the cumulative average of the errors between the actual and expected behaviors of the robot during navigation. It generally evaluates the extent to which the robot deviates from its planned navigational behaviors. Specifically, the inconsistency metric combines the error in the robot's position and the error in the robot's orientation. Since inconsistency is defined as a composite metric, it does not have a standard unit.

- *Jerkiness* is defined as the average summation of the acceleration derivatives along all the axes. It indicates how smooth a robot can traverse a terrain. Reducing jerkiness is important to improve state estimation and SLAM that may assume smooth robot motions.

We compare the proposed approach with three previous robot navigation methods, including Model Predictive Path Integral (MPPI) (Williams et al., 2016), Non-Linear Control Optimization tool (NLOPT) (Johnson, 2011), and Terrain Representation and Apprenticeship Learning (TRAL) (Siva et al., 2019). MPPI and NLOPT are control-based methods, whereas TRAL is a learning-based method.

5.1.3. System implementation. An overview of the ROS-based software architecture implemented on our physical robots to perform consistent off-road ground navigational is illustrated in Figure 4. Our approach, implemented as a ROS package in C++, collaboratively functions with various other packages within the ROS framework and runs in real-time at 20 Hz. Acting as a local controller (i.e., generating navigational behaviors for the present time step), our approach generates navigational behaviors and works in conjunction with both global and local planners.

The global planner employs a cost map to generate a global path, providing intermediate waypoints to the local planner, which guides the robot towards its destination. Utilizing point cloud data from the perception package, the local planner identifies obstacles in close proximity to the robot, creating a local path that avoids these obstacles between the waypoints. The local path and associated angular velocities produced by the local planner are passed to our method, which adapts the velocity commands to generate consistent navigational behaviors ensuring the robot accurately follows the path. Specifically, our approach computes expected linear navigational velocities based on terrain feature vectors extracted by the feature extraction package. Our approach continuously monitors the difference between the robot's expected linear velocities and actual linear velocities achieved based on proprioception. Then, our approach calculates offset linear velocity commands based on a sequence of past differences between expected and achieved linear velocities. These offset behaviors are then applied to compensate for the behavior differences during off-road ground navigation scenarios, ensuring consistent navigation.

5.2. Robot navigation over individual unstructured off-road terrains

In this set of experiments, the robot navigates over individual types of off-road terrains, including concrete, grass, gravel, large rock, and snow. These terrains are illustrated in Figure 5. Each terrain track is around 10 m long and has different characteristics. We train our method using the data collected when a human operator manually controls the robot to traverse over each of the terrains. The learned

model is then deployed on the robot to generate terrain-aware controls as the ground robot autonomously navigates on the terrains. Evaluation metrics for each method are computed across ten trials on each type of individual terrain track.

The quantitative results achieved by our approach and its comparison with other methods are provided in Table 2. We observe that all methods enable the robot to successfully navigate over these individual types of off-road terrains with the exception of a single failure from NLOPT occurring on the large-rock terrain due to high centering. We compute the traversal time by averaging the time used by the robot to traverse over each terrain for all successful runs. We observe that our approach has the lowest traversal time over the concrete, grass, and snow terrains. For the gravel and large-rock terrains, our approach uses more time than MPPI. For all individual terrains, TRAL has a higher traversal time than MPPI and our approach, but NLOPT achieves the longest traversal time of all methods. Additionally, note that the model-based methods, MPPI and NLOPT, display the smallest standard deviations in traversal time across the set of terrains. This indicates that these approaches are not adapting their navigation approach to different terrain types. On the other hand, our approach is able to adaptively generate terrain-aware navigational controls. For example, our method generates a higher velocity control and obtains a lower traversal time over smoother terrains (e.g., concrete, grass, and snow).

Table 2 also shows the results of the inconsistency and jerkiness metrics. We observe that our approach achieves the lowest inconsistency for most individual terrains except the grass terrain, followed by the MPPI and TRAL

methods. MPPI has the lowest inconsistency over grass terrain. TRAL obtains lower inconsistency than MPPI over the large-rock terrain, but performs worse than MPPI over other terrains. NLOPT performs the worst and has the highest inconsistency over all individual terrains. In general, we observe that the learning-based methods obtain a better performance than the model-based methods in terms of inconsistency. For the metric of jerkiness, we observe that our approach obtains the lowest jerkiness over most terrains except snow, where MPPI marginally outperforms our approach. For almost all cases, both NLOPT and TRAL obtain higher jerkiness values than MPPI and our approach across each terrain type.

5.3. Robot adaptation to terrain transitions

In this set of experiments, we evaluate our proposed method when the ground robot navigates over unstructured terrain transitions. We leveraged an off-road ground navigation circuit, which is an experimental testing facility at the U.S. DEVCOM Army Research Laboratory Robotics Research Collaboration Campus (R2C2). An aerial view of the circuit and its terrain is shown in Figure 6.

We consider the scenario of a ground robot navigating the transitions between any combination of two terrains: grass, gravel, large rocks, mud, and sand. The aim is to evaluate how different approaches adapt with the changing terrains, for example, robots need to slow down when transitioning from grass to rocks. In this experiment, no additional training is performed and the previously trained model from navigating over individual types of

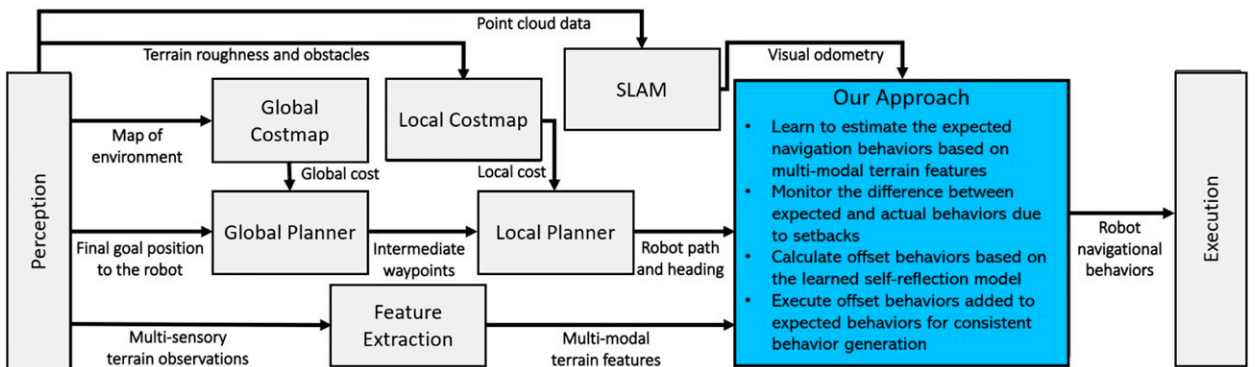


Figure 4. An overview of the software architecture implemented in ROS on our physical robots to achieve consistent off-road ground navigation. Our approach is represented by the blue box, while the other modules that collaborate with our approach in the software architecture are depicted by gray boxes.



Figure 5. Individual types of terrains used in the experiments. From left to right: concrete, grass, large rock, gravel, and snow.

unstructured terrains is used. The evaluation metrics are computed from ten trial runs.

The quantitative results obtained by our approach and comparisons with other methods are provided in Table 3. In terms of failure rate, we observe that NLOPT has the worst performance. Comparatively, the MPPI and TRAL methods demonstrate a lower failure rate, with our approach achieving the lowest failure rate, thereby performing the best. Navigation failures for all methods are observed when the robot traverses over terrain combinations including sand. Failures while navigating over sand occur either when the robot gets high-centered or when the terrain slope is greater than 25° . Our approach, with the ability to generate consistent behaviors, is able to compensate for wheel slip while traversing the sand terrain and obtains a lower failure rate.

In terms of the traversal time metric, we observe that both NLOPT and TRAL obtain the highest traversal time values, with NLOPT obtaining the highest value. Both MPPI and our approach obtain lower traversal time values. The traversal time values from our approach are higher when traversing terrains consisting of rock, sand, and mud, and are relatively lower when traversing terrains consisting of grass. This indicates the ability of our approach to adapt its behaviors to different types of terrains. The traversal time value is higher when navigating over terrains that include sand due to a higher rate of wheel slips.

In terms of the inconsistency metric, we observe that the NLOPT and TRAL methods obtain the highest values. Both MPPI and our approach have comparatively lower values of inconsistency, with our approach obtaining the smallest inconsistency values across terrain combinations. For all methods, the inconsistency value is high for terrain

transitions including sand due to an increased rate of wheel slip. We observe a similar trend in terms of the jerkiness metric, where both the NLOPT and TRAL approaches obtain higher jerkiness values, with NLOPT having the highest value and performing the poorest. However, both MPPI and our approach obtain the least jerkiness value, with our approach performing slightly better than MPPI. For all methods, the inconsistency value is higher when traversing the terrain transitions including rock and gravel due to their rugged nature.

From this set of experiments, we observe that our approach generally performs the best over all the terrain combinations and outperforms previous methods as it achieves the fewest failures, while also performing best in terms of inconsistency and jerkiness.

5.4. Robot self-reflective adaptation over unstructured terrains

In this set of experiments, we evaluate our approach's use of self-reflection to enhance robot navigation. Specifically, we perform experiments in scenarios where robots experience unexpected setbacks in their functionality. Each of these setbacks increases the difficulty of robot navigation over unstructured terrain and requires the robot to adapt via self-reflection. These setbacks are detailed below:

- *Over-Inflated Tires:* In this scenario, we over-inflate each tire on the wheeled ground robot, which causes reduced traction with the terrain. With this setback, the robot has a higher center of mass and increased jerkiness when traversing, as there is less damping from the over-

Table 2. Quantitative results for scenarios when the robot traverses over the individual types of unstructured terrain shown in Figure 5. Successful runs (with no failures) are used to calculate the metrics of traversal time, inconsistency, and jerkiness. Our approach is compared against classical control based approaches, MPPI (Williams et al., 2016) and NLOPT (Johnson, 2011), and a learning-based method, TRAL (Siva et al., 2019).

Terrain	Failure rate (/10)				Traversal time (s)				Inconsistency				Jerkiness (m/s^3)			
	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours
Concrete	0	0	0	0	12.82	17.82	16.04	12.49	0.65	0.79	0.69	0.56	5.13	4.98	5.19	4.35
Grass	0	0	0	0	13.45	18.02	16.39	13.32	0.61	0.90	0.88	0.62	7.82	8.52	8.32	6.98
Gravel	0	0	0	0	13.96	16.39	15.32	14.84	1.21	1.67	1.72	1.03	16.11	16.66	16.43	15.09
Large rock	0	1	0	0	13.45	17.87	16.71	16.31	3.97	5.64	3.21	2.91	13.75	17.61	14.85	12.32
Snow	0	0	0	0	14.22	18.09	17.49	14.08	7.18	9.93	8.12	6.64	5.12	5.76	5.49	5.42



Figure 6. The off-road ground navigation circuit consists of different types of terrains (such as grass, gravel, large rock, mud, and sand) as well as diverse terrain transitions. The size of the circuit is 10×21 m with terrain slopes varying between 0 and 30° .

Table 3. Quantitative results based on ten runs for scenarios when the robot navigates over the various terrain transitions shown in Figure 6. Successful runs (with no failures) are used to calculate the metrics of traversal time, inconsistency and jerkiness. Our approach is compared against classical control based approaches, MPPI (Williams et al., 2016) and NLOPT (Johnson, 2011), and a learning-based method, TRAL (Siva et al., 2019).

Terrain	Failure rate (/10)				Traversal time (s)				Inconsistency				Jerkiness (m/s ³)			
	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours
Mud-sand	2	3	2	1	25.27	31.54	22.19	21.82	7.86	8.13	7.06	6.45	9.13	11.31	10.73	8.95
Sand-grass	2	3	2	1	24.01	31.99	23.04	22.57	7.19	7.65	6.99	6.25	8.58	10.61	10.09	8.41
Grass-mud	0	0	0	0	18.67	20.69	21.98	20.09	3.85	3.98	3.45	3.16	4.29	5.31	5.04	4.20
Mud-rock	0	0	0	0	19.76	21.86	22.07	20.87	6.50	6.73	5.84	5.34	11.74	14.54	13.80	11.59
Rock-sand	2	3	1	1	22.21	22.67	23.17	22.03	10.88	11.14	10.07	9.53	11.12	13.78	13.06	10.91
Sand-gravel	2	3	1	1	23.09	23.98	23.65	22.98	11.81	12.28	10.61	9.70	15.57	19.28	18.30	15.26
Gravel-grass	0	0	0	0	22.13	26.04	22.73	22.62	7.25	7.55	7.51	6.65	15.15	18.97	18.09	15.03
Grass-rock	0	0	0	0	20.01	26.55	21.82	19.13	4.45	4.90	4.01	3.91	6.38	7.91	7.50	6.26
Rock-gravel	0	0	0	0	17.89	27.03	22.41	20.17	3.78	3.91	4.31	3.18	13.34	16.53	15.68	13.09

inflated tires. We tested our approach with over-inflated tires on three different terrains. The first terrain, mixed terrain I (MT-I), is a mixture of concrete, large rocks, grass, and snow which the robot can experience them at once or individually. The second terrain, mixed terrain II (MT-II), is a mixture of grass, gravel, and bushes. Finally, the third terrain, forest terrain, is a cluttered environment with logs and rocks hidden by fallen leaves and tall grass. None of these three terrains have any significant slope.

- *Reduced Tire Traction:* In this scenario, we completely deflate the robot’s tires and apply duct tape around the deflated tires, significantly reducing tire traction. This setback causes the robot to have a lower center of mass. We then evaluate our approach with reduced tire traction on four different terrains. The first terrain, snow-grass (referred to as RT-Snow-Grass), is a grass terrain covered by snow. The second terrain, snow-mud (referred to as RT-Snow-Mud), is a mud terrain covered with snow. The third terrain, hill (referred to as RT-Hill), is a hill terrain consisting of rocks, snow, and logs covered by tall grass. Finally, the fourth terrain, snow-forest (referred to as RT-Snow-Forest), is a forest terrain covered with snow. Each of these terrains, with the exception of RT-Snow-Forest terrain, have varying slopes of 0°–45°, which increases the difficulty of robot traversal.
- *Reduced Tire Traction with Increased Payload:* In this scenario, in addition to the deflated and duct-taped robot tires, we introduce 25 lbs of payload to the back of the robot. With this setback, we evaluate our approach on two different terrains. The first terrain, snow-grass (referred to as RT-P-Snow-Grass), is a grass terrain covered with snow. The second terrain, hill (referred to as RT-P-Hill), consists of rocks and snow, covered by tall grass. Both these terrains have slopes ranging from 0° to 45°.

These nine different testing locations are demonstrated in Figure 7, which represent the challenges that a ground robot can experience in real-world unstructured environments. The evaluation metrics for each method are calculated based

on ten trials of robot runs on each of the terrains. No additional training is performed, and we employ the same model from experiments on individual off-road terrains (from Section 5.2), with all parameters kept the same. The quantitative results obtained by our approach and its comparison with other approaches are presented in Table 3. We discuss the results of the three setback experiments next.

5.4.1. Over-inflated tires results. For the scenario of over-inflated tires, we can observe that NLOPT obtains the highest failure rate. Both MPPI and TRAL perform better than NLOPT, achieving fewer failures. Comparatively, our approach achieves the lowest failure rate across all three terrains. In general, we observe that the failure rate is lower on the mixed terrain tracks, and higher on the cluttered forest terrain.

In terms of traversal time, we observe varying trends across the methods depending on the terrain type. Notice that NLOPT has the lowest traversal time on the forest terrain, but the highest traversal time on both the mixed terrains. On the other hand, our approach displays the inverse, with some of the lowest traversal times on the mixed terrain and the highest traversal time on the forest terrain. We see this increase in the traversal time on the forest terrain as an illustration of our approach’s ability to adapt its navigational behaviors with cluttered terrains. That is, speed is reduced to ensure the robot can successfully navigate the environment. This is further supported by the fact that our approach achieves the fewest failure cases in the forest terrain. Also note that although NLOPT has the fastest traversal time for the forest terrain, it fails to reach the goal in half of the experimental trials. In general, MPPI and TRAL exhibit slower traversal rates than our approach, and in cases with faster traversal time we also observe that they result in higher failure rates.

In terms of the inconsistency metric, our approach, with its ability to generate consistent behaviors, obtains the lowest values and outperforms previous methods. Specifically, both TRAL and NLOPT have high inconsistency values over all terrains. In terms of the jerkiness metric, we

observe that all approaches have high values in the forest terrain due to its cluttered nature. Our approach has the lowest jerkiness value in the mixed terrains but performs poorly in forest terrain. Across all three terrains, MPPI, NLOPT, and TRAL perform similarly with a slightly better performance exhibited from MPPI on two out of three terrain types (Table 4).

5.4.2. Reduced tire traction results. For the scenario of reduced tire traction, generally we observe an increased failure rate compared to the scenario of over-inflated tires. Specifically, in the terrains with significant slopes, that is, RT-Snow-Mud, RT-Hill, and RT-Snow-Grass, the failure rate is higher when compared to RT-Snow-Forest. Throughout all the terrains, we observe that the NLOPT approach obtains the highest failure rate and performs the poorest. Both MPPI and TRAL perform better, with our approach performing the best and obtaining the lowest failure rate.

In terms of the traversal time metric, we observe that the MPPI approach performs better than most approaches across all the terrains except for the RT-Snow-Mud terrain, where our approach has the lowest traversal time. This indicates the ability of our approach to exhibit terrain-aware behaviors as our approach is able to adapt its traversal speed with different terrains. Both TRAL and NLOPT have higher traversal times and exhibit poor performance over all terrains.

For the inconsistency metric, our approach obtains the lowest value over all terrains, followed by both MPPI and TRAL, with MPPI performing better on the RT-Snow-Grass and RT-Snow-Forest terrain, and TRAL performing better on the remaining terrains. The NLOPT approach obtains the highest inconsistency value, indicating a poor performance when compared to other approaches. When observing the jerkiness metrics obtained by the different approaches, we see a similar trend of our approach performing the best across different terrains, followed by both MPPI and TRAL. The NLOPT approach has some of the highest jerkiness values on all the terrains, indicating a poor performance overall. Overall, we see a smaller value of jerkiness from all methods on the RT-Snow-Forest terrain compared to the forest terrain discussed in the previous experiments with over-inflated tires. This indicates that the rugged nature of the forest terrain is suppressed when the terrain is covered with snow.

5.4.3. Reduced tire traction with increased payload results. Finally, we evaluate our approach in the scenario of reduced tire traction with increased payload. These setbacks, especially on terrains with varying slopes, make it very challenging for mobile robots to traverse the terrain [Grand et al., 2004](#). Accordingly, we observe that the failure rates are high in this scenario compared to the previous scenarios. Specifically, in the RT-P-Hill terrain, all methods fail at least 40% of the time. Overall, our approach achieves



Figure 7. Various terrain tracks used to evaluate the self-reflection capability of the robot. From left to right: mixed terrain I (MT-I), mixed terrain II (MT-II), forest, reduced-traction snow-grass (RT-Snow-Grass), reduced-traction mud-snow (RT-Snow-Mud), reduced-traction hill (RT-Hill), reduced-traction forest (RT-Snow-Forest), reduced-traction payload snow-grass (RT-P-Snow-Grass), and reduced-traction payload hill terrain (RT-P-Hill).

Table 4. Quantitative results based on ten runs for the terrain scenarios shown in Figure 7 used to evaluate the self-reflection capability of the robot. Successful runs (with no failures) are used to calculate the metrics of traversal time, inconsistency and jerkiness. Our approach is compared against classical control-based approaches, MPPI ([Williams et al., 2016](#)) and NLOPT ([Johnson, 2011](#)), and a learning-based method, TRAL ([Siva et al., 2019](#)).

Terrain	Failure rate (/10)				Traversal time (s)				Inconsistency				Jerkiness (m/s ³)			
	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours	MPPI	NLOPT	TRAL	Ours
MT-I	0	0	0	0	14.51	17.68	16.84	13.74	0.65	0.79	0.69	0.56	5.68	5.57	5.13	4.95
MT-II	1	2	1	0	13.44	17.48	16.74	13.64	0.78	0.98	0.88	0.71	7.66	8.75	8.35	6.52
Forest	2	5	3	1	14.67	13.67	15.29	15.96	1.21	1.67	1.72	1.11	16.25	16.62	16.55	16.78
RT-Snow-Grass	3	4	4	2	13.24	17.66	16.18	13.72	4.61	5.90	5.88	4.11	9.12	9.52	9.32	8.98
RT-Snow-Mud	5	7	4	2	13.28	17.84	16.10	12.03	3.97	5.64	3.21	3.01	13.14	17.15	14.12	12.35
RT-Hill	3	6	3	2	13.07	15.85	16.42	13.91	14.98	15.23	14.91	13.07	13.75	17.61	14.85	12.32
RT-Snow-Forest	0	0	0	0	13.55	16.63	14.39	14.34	7.18	9.93	8.12	6.62	6.72	6.88	6.92	6.47
RT-P-Snow-Grass	4	7	5	2	13.96	16.39	15.32	13.84	9.12	9.96	9.57	8.54	16.11	16.66	16.43	16.09
RT-P-Hill	5	8	6	4	14.59	15.87	14.38	15.77	11.18	15.93	14.12	10.62	13.12	17.56	14.86	12.51

the best performance, followed by the MPPI, TRAL, and NLOPT approaches, respectively.

In terms of traversal time, we observe mixed results as our approach has the lowest traversal time on the RT-P-Snow-Grass terrain. However, it has the highest traversal time on the RT-P-Hill terrain, whereas the TRAL approach has the lowest traversal time. The MPPI approach has the second-best performance throughout, and the NLOPT approach has the highest traversal time on both terrains indicating a poor performance.

Similar to the previous scenarios, we observe that our approach has the lowest inconsistency value on both the terrains, followed by the MPPI and TRAL approaches. Again, we observe that the inconsistency values obtained by the NLOPT approach are higher than the other approaches. A similar trend is seen in terms of the jerkiness metric as well. Our approach has the best performance, followed by the MPPI approach on all terrains. Both the TRAL and NLOPT approaches have a higher value of jerkiness, indicating a poor performance.

Across all three categories of setbacks, it is observed that with the generation of consistent navigational behaviors, our approach can reduce the number of failures, with the trade-off of increased traversal time. Again, this increased traversal time can be attributed to the decrease in speed to ensure safety, which ultimately also helps produce lower inconsistency and jerkiness metrics.

5.5. Discussion

We further discuss the experimental results to investigate the characteristics of our approach. We analyze how our approach performs in the Mixed-Terrain I (MT-I) scenario, considering different parameters. This analysis forms the foundation for selecting particular parameter values to be used consistently in all our experimental runs. We include performance on various frame sequences, hyperparameter analysis, relative weights of each time step, and the discriminative feature modalities.

5.5.1. Dependence on frame sequences. Our approach uses a sequence of historical frames with length c to generate self-reflective consistent behaviors. Figure 8(a) illustrates the inconsistency value under different values of c . We use the best inconsistency value obtained from different runs of $\lambda_1 \in$

$[0.001, 10]$ and $\lambda_2 \in [0.001, 10]$ for each c in this comparison. Figure 8(a) uses a box plot to display the distribution of results based on a five number summary (i.e., minimum, first quartile, medium, thirds quartile, and maximum). Although, the performance of our approach barely differs over various frame sequence length c , the best performance is observed when $c = 6$, that is, box plot with lowest minimum. This value of frame sequences is then used for all of our experiments. These values of c are affected by the robot's speed and can differ between robotic platforms.

5.5.2. Hyperparameter analysis. Our approach utilizes two hyperparameters λ_1 and λ_2 that balance the amount of loss from our objective function and the regularization terms. Figure 8(b) depicts how the inconsistency metric changes with each of these hyperparameter values for six historical frame sequences, that is, $c = 6$. In general, it is seen that the inconsistency metric changes significantly with variations in λ_2 as compared to λ_1 , signifying the importance of modeling historical time steps. Altogether, it is observed that $\lambda_1 \in (0.01, 1)$ and $\lambda_2 \in (0.01, 10)$ result in a good performance. The best results are observed with $\lambda_1 = 0.1$ and $\lambda_2 = 1$. These are the fixed hyperparameters used for carrying out all the experiments.

5.5.3. Relative importance of time steps. In general, the best performance of our approach is observed when $c = 6$, meaning we leverage features from the current time step and the previous 5 time steps. Figure 8(c) presents the importance of each of these time steps by displaying the relative weights of each. It is observed that the most recent time steps are more important when compared to earlier time steps. Specifically, the recent four time steps are equally important, with each weighting 20%. Whereas, the oldest time steps (i.e., fifth and sixth of $c = 6$) contribute smaller importance of 10%. This breakdown further shows that features from previous time steps can provide equally important information and should be considered for generating self-reflective navigational behaviors. Although these observations indicate that our approach takes into account a shorter time window for generating navigational behaviors, this is primarily attributed to the use of a slower ground robot in our experiments. In the scenario of a faster ground robot, we posit that a more extended time window would be necessary for the generation of self-reflective navigational behaviors.

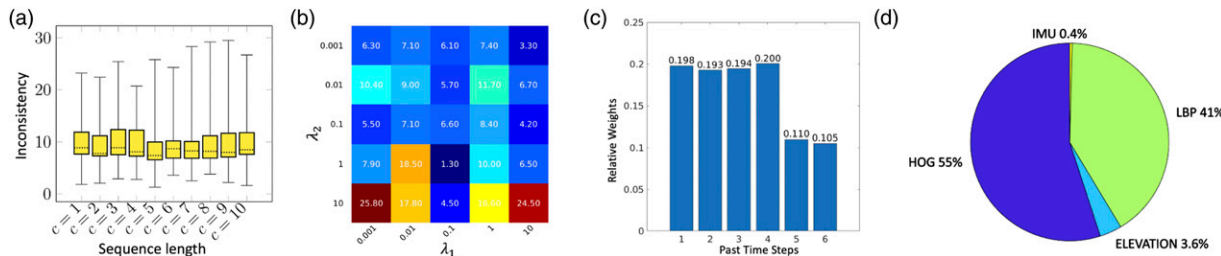


Figure 8. Analysis of our approach based on its various parameters. (a) Frame sequences, (b) hyperparameter analysis, (c) relative importance of time steps, and (d) discriminative feature modalities.

5.5.4. Discriminative feature modalities. Our approach can automatically estimate the importance of the various feature modalities in generating navigational behaviors. The results from evaluating the modality importance are shown in Figure 8(d). It is observed that two feature modalities, HOG and LBP, are the most discriminative in performing self-reflective terrain adaptation. These two features, both extracted from camera images, combined account for 96% of the relative importance. The IMU features and elevation features from LiDAR are comparatively much less important. However, to perform self-reflection, we rely on the LiDAR data to perform SLAM (Shan and Englot, 2018) and provide the actual behaviors so the behavior differences can be computed. In this sense, LiDAR is also an important data modality for our approach.

6. Conclusion

In this paper, we introduce a novel method for self-reflective terrain-aware adaptation that enables an autonomous robot to generate consistent behaviors to navigate on unstructured off-road terrains. Formulated under a unified constrained regularized optimization framework, our approach monitors the inconsistency between expected and actual behaviors that are caused by robot setbacks and then adapts the robot's navigational behaviors accordingly to enable consistent navigation. In addition, our approach identifies discriminative terrain features and fuses them to perform effective adaptive navigational behaviors to changing terrain. We develop an algorithm to solve the formulated optimization problem, which holds a theoretical guarantee to converge to the global optimal solution of the optimization problem. To evaluate our approach, we conduct extensive experiments using real ground mobile robots with varying functionality changes over diverse unstructured off-road terrains. Experimental results have shown that our self-reflective terrain-aware adaptation method outperforms previous and baseline techniques and enables ground robots to generate consistent behaviors when navigating in off-road environments.

While our method has enabled self-reflective terrain-aware robot adaptation for consistent navigation, it serves as a local controller under an external local planner and a pure learning-based method. Accordingly we consider two key future directions to further this work. First, developing a joint local planner and local controller may improve robot's navigation performance by simultaneously generating a local path for obstacle avoidance and navigational behaviors for the robot to consistently follow the path while adapting to the terrain on the path. Second, integrating learning-based models with control theories or control-based methods can serve as a great future direction. Learning-based methods have the ability to estimate model parameters through learning from data and the ability to fuse high-dimensional observations obtained from heterogeneous sensors. Control-based methods have the ability to model the behavior of dynamical systems over time as well as have good model predictability. Combining both learning- and control-based methods into a unified approach

has the potential to provide new opportunities to further improve robot navigation.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially supported by NSF CAREER Award IIS-2308492, DARPA Young Faculty Award (YFA) D21AP10114-00, and DEVCOM ARL SARA Program W911NF-20-2-0107.

ORCID iDs

Sriram Siva  <https://orcid.org/0000-0003-3457-2085>

Hao Zhang  <https://orcid.org/0000-0001-8043-9184>

References

- Ahonen T, Hadid A and Pietikainen M (2006) Face description with local binary patterns: application to face recognition. *Transactions on Pattern Analysis and Machine Intelligence* 28: 2037
- Allothman Y and Gu D (2016) Quadrotor transporting cable-suspended load using iterative linear quadratic regulator (iLQR) optimal control. In: Computer science and electronic engineering, Colchester, UK, 28–30 September 2016
- Altahhan A (2016) Self-reflective deep reinforcement learning. In: International joint conference on neural networks, Vancouver, BC, Canada, 24–29 July 2016
- Altan A and Hacıoğlu R (2020) Model predictive control of three-axis gimbal system mounted on uav for real-time target tracking under external disturbances. *Mechanical Systems and Signal Processing* 138: 106548.
- Aoyama Y, Boutselis G, Patel A, et al. (2021) Constrained differential dynamic programming revisited. In: International conference on robotics and automation, Xi'an, China, IEEE
- Ardelt M and Grunwald S (2018) The importance of self-reflection and awareness for human development in hard times. *Research in Human Development* 15: 1–13
- Argall BD, Chernova S, Veloso M, et al. (2009) A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57: 469
- Borges CDB, Almeida AMA, Júnior ICP, et al. (2019) A strategy and evaluation method for ground global path planning based on aerial images. *Expert Systems with Applications* 137: 232
- Boyd SP and Vandenberghe L (2004) *Convex Optimization*. Cambridge University Press.
- Brunton SL and Kutz JN (2019) *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press.
- Cáceres C, Rosário JM and Amaya D (2017) Approach of kinematic control for a non-holonomic wheeled robot using artificial neural networks and genetic algorithms. In: International conference and workshop on bio-inspired intelligence, Funchal, Portugal, 10–13 July 2017

- Dalal N and Triggs B (2005) Histograms of oriented gradients for human detection. In: Conference on computer vision and pattern recognition, San Diego, CA, USA, 20–25 June 2005
- Della Santina C, Bianchi M, Grioli G, et al. (2017) Controlling soft robots: balancing feedback and feedforward elements. *Robotics Automation Magazine* 24: 1
- Demir A and Sezer V (2019) Motion planning and control with randomized payloads using deep reinforcement learning. In: International conference on robotic computing, Naples, Italy, 25–27 Feb. 2019
- Duriez T, Brunton SL and Noack BR (2017) *Machine Learning Control-Taming Nonlinear Dynamics and Turbulence*. Springer.
- Fankhauser P, Bloesch M, Gehring C, et al. (2014) Robot-centric elevation mapping with uncertainty estimates. In: *Mobile Service Robotics*. World Scientific
- Fathinezhad F, Derhami V and Rezaeian M (2016) Supervised fuzzy reinforcement learning for robot navigation. *Applied Soft Computing* 40: 33–41.
- Fridovich-Keil D, Ratner E, Peters L, et al. (2020) Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In: International Conference on Robotics and Automation, Paris, France, IEEE
- Gangal AS, Kalra P and Chauhan D (2007) *Performance Evaluation of Complex Valued Neural Networks Using Various Error Functions*. Enformatika.
- Gillespie MT, Best CM, Townsend EC, et al. (2018) Learning nonlinear dynamic models of soft robots for model predictive control with neural networks. In International conference on soft robotics, Livorno, Italy, 24–28 April 2018
- Grand C, Benamar F, Plumet F, et al. (2004) Stability and traction optimization of a reconfigurable wheel-legged robot. *The International Journal of Robotics Research* 23: 1041–1058.
- Han SH, Choi HJ, Benz P, et al. (2018) Sensor-based mobile robot navigation via deep reinforcement learning. In: International conference on big data and smart computing, Shanghai, China, 15–17 Jan 2018
- Hawkes EW, Majidi C and Tolley MT (2021) Hard questions for soft robotics. *Science Robotics* 6: eabg6049.
- Iqbal U, Nouredin A, Georgy J, et al. (2021) Application of system identification techniques for integrated navigation. In: International conference on communications, signal processing, and their applications, IEEE
- Ishida Y (2015) A note on continuous self-identification as self-awareness: an example of robot navigation. *Procedia Computer Science* 60: 1865–1874.
- Jeong S and Chwa D (2017) Coupled multiple sliding-mode control for robust trajectory tracking of hovercraft with external disturbances. In: Transactions on industrial electronics, IEEE
- Jin J, Nguyen NM, Sakib N, et al. (2020) Mapless navigation among dynamics with social-safety-awareness: a reinforcement learning approach from 2D laser scans. In International conference on robotics and automation, Xi'an, China, IEEE
- Johnson SG (2011) *The NLOpt Nonlinear-Optimization Package*. NLOpt
- Kahn G, Villafior A, Abbeel P, et al. (2018a) Composable action-conditioned predictors: flexible off-policy learning for robot navigation. In: Conference on robot learning, Atlanta, Georgia, 6–9 Nov 2023
- Kahn G, Villafior A, Ding B, et al. (2018b) Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In: International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018
- Kahn G, Abbeel P and Levine S (2021) Badgr: an autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters* 6: 1312–1319.
- Kawatsuma S, Fukushima M and Okada T (2012) Emergency response by robots to fukushima-daiichi accident: summary and lessons learned. *Industrial Robot: International Journal* 39: 428–435.
- Knight R, Fisher F, Estlin T, et al. (2001) Balancing deliberation and reaction, planning and execution for space robotic applications. In: International conference on intelligent robots and systems. IEEE
- Kumar A, Fu Z, Pathak D, et al. (2021) RMA: rapid motor adaptation for legged robots. Robotics: Science and systems, Delft, Netherlands, 15–19 Jul 2024
- Kuntze HB, Frey CW, Tchouchenkov I, et al. (2012) Seneka-sensor network with mobile robots for disaster management. In Conference on Technologies for Homeland Security, Waltham, MA, USA, 13–15 Nov 2012
- Kutz JN, Brunton SL, Brunton BW, et al. (2016) *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM.
- Li Z, Nakatsukasa Y, Soma T, et al. (2018) On orthogonal tensors and best rank-one approximation ratio. *SIAM Journal on Matrix Analysis and Applications* 39
- Li X, Tang B, Ball J, et al. (2019) Rollover-free path planning for off-road autonomous driving. *Electronics* 8(6): 614.
- Liu X, Cao Z, Yu Y, et al. (2021) Robot navigation based on situational awareness. In: Transactions on cognitive and developmental systems, IEEE
- L'Erario G, Fiorio L, Nava G, et al. (2020) Modeling, identification and control of model jet engines for jet powered robotics. *IEEE Robotics and Automation Letters* 5: 2070–2077.
- Mamakoukas G, Castano M, Tan X, et al. (2019) Local koopman operators for data-driven control of robotic systems. In: Robotics: Science and Systems, Germany, 22–26 June 2019
- Marcovitch S, Jacques S, Boseovski JJ, et al. (2008) Self-reflection and the cognitive control of behavior: implications for learning. *Mind, Brain, and Education* 2(3): 136
- Masmoudi MS, Krichen N, Masmoudi M, et al. (2016) Fuzzy logic controllers design for omnidirectional mobile robot navigation. *Applied Soft Computing* 49: 901
- Nagariya A and Saripalli S (2020) An iterative LQR controller for off-road and on-road vehicles using a neural network dynamics model. In: Intelligent vehicles symposium, Las Vegas, NV, USA, IEEE
- Nazari M, Amiryani J and Nazemi E (2013) Improvement of robot navigation using fuzzy method. In: Joint Conference of AI & Robotics and 5th RoboCup Iran Open International Symposium, Tehran, Iran, 8 April 2013

- Negahban S and Wainwright MJ (2012) Restricted strong convexity and weighted matrix completion: optimal bounds with noise. *The Journal of Machine Learning Research* 13: 1665
- Nehmzow U, Akanyeti O, Weinrich C, et al. (2007) Robot programming by demonstration through system identification. In: International Conference on Intelligent Robots and Systems, San Diego, CA, USA, IEEE
- Neudecker H (1969) A note on Kronecker matrix products and matrix equation systems. *SIAM Journal on Applied Mathematics* 17(3): 603–606.
- Nie F, Huang H, Cai X, et al. (2010) Efficient and robust feature selection via joint ℓ_2 , ℓ_1 -norms minimization. In: *Advances in Neural Information Processing Systems*. Morgan Kaufmann Publishers
- Ostafew CJ, Schoellig AP, Barfoot TD, et al. (2016) Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking. *Journal of Field Robotics* 33: 133–152.
- Park S, Oh Y and Hong D (2017) Disaster response and recovery from the perspective of robotics. *International Journal of Precision Engineering and Manufacturing* 18: 1475–1482.
- Pereida K and Schoellig AP (2018) Adaptive model predictive control for high-accuracy trajectory tracking in changing conditions. In: International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018
- Proctory JL, Bruntonz SL and Kutz JN (2018) Generalizing Koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems* 17: 909–930.
- Qi L (2017) *Transposes, L-Eigenvalues and Invariants of Third Order Tensors*. arXiv
- Rabanser S, Shchur O and Günnemann S (2017) *Introduction to Tensor Decompositions and Their Applications in Machine Learning*. arXiv
- Rahman S and Waslander SL (2021) Uncertainty-constrained differential dynamic programming in belief space for vision based robots. *IEEE Robotics and Automation Letters* 6: 3112–3119.
- Ramírez DR, Limón D, Gomez-Ortega J, et al. (1999) Nonlinear MBPC for mobile robot navigation using genetic algorithms. In: International Conference on Robotics and Automation, Detroit, MI, USA, 10–15 May 1999
- Rana M, Mukadam M, Ahmadzadeh SR, et al. (2018) Towards robust skill generalization: unifying learning from demonstration and motion planning. In: Intelligent Robots and Systems, Madrid, Spain, 1–5 Oct 2018
- Rubilar F, Escobar MJ and Arredondo T (2014) Bio-inspired architecture for a reactive-deliberative robot controller. In: International Joint Conference on Neural Networks, Beijing, China, 6–11 July 2014
- Schmid PJ (2010) Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics* 656: 5–28.
- Schwarz M, Rodehutsors T, Droschel D, et al. (2017) NimRo rescue: solving disaster-response tasks with the mobile manipulation robot Momaro. *Journal of Field Robotics* 34: 400–425.
- Seraji H and Howard A (2002) Behavior-based robot navigation on challenging terrain: a fuzzy logic approach. *IEEE Transactions on Robotics and Automation* 18: 308–321.
- Shan T and Englot B (2018) LeGO-LOAM: lightweight and ground-optimized Lidar Odometry and mapping on variable terrain. In: International conference on intelligent robots and systems, Madrid, Spain, 1–5 October 2018
- Shan M, Bi Y, Qin H, et al. (2016) A brief survey of visual odometry for micro aerial vehicles. In: Annual conference of the industrial electronics society. IEEE
- Siva S, Wigness M, Rogers J, et al. (2019) Robot adaptation to unstructured terrains by joint representation and apprenticeship learning. In: Robotics: Science and systems, Germany, 22–26 June 2016
- Siva S, Wigness M, Rogers J, et al. (2021) Enhancing consistent ground maneuverability by robot adaptation to complex off-road terrains. In: Conference on robot learning. IEEE
- Smith R (2002) Self-reflection and the self. In: *Rewriting the Self*, Routledge
- Sofman B, Lin E, Bagnell JA, et al. (2006) Improving robot navigation through self-supervised online learning. *Journal of Field Robotics* 23: 1059–1075.
- Tassa Y, Erez T and Todorov E (2012) Synthesis and stabilization of complex behaviors through online trajectory optimization. In: International conference on intelligent robots and systems. IEEE
- Thrun S (1998) Lifelong learning algorithms. In: *Learning to Learn*. Boston, MA, USA: Kluwer Academic Publishers
- Truong J, Chernova S and Batra D (2021) Bi-directional domain adaptation for sim2real transfer of embodied navigation agents. *IEEE Robotics and Automation Letters* 6: 2634–2641.
- Tu JH (2013) *Dynamic Mode Decomposition: Theory and Applications*. PhD Thesis. Princeton University.
- Wang H and Noguchi N (2021) *Real-Time States Estimation of a Farm Tractor Using Dynamic Mode Decomposition*. GPS Solutions.
- Wang Z, Xiao X, Liu B, et al. (2021) APPLI: adaptive planner parameter learning from interventions. In: International conference on robotics and automation, Xi'an, China, IEEE
- Wigness M, Rogers JG and Navarro-Serment LE (2018) Robot navigation from human demonstration: learning control behaviors. In: International conference on robotics and automation, Brisbane, QLD, Australia, 21–25 May 2018
- Williams MO, Kevrekidis IG and Rowley CW (2015) A data-driven approximation of the koopman operator: extending dynamic mode decomposition. *Journal of Nonlinear Science* 25: 1307–1346.
- Williams G, Drews P, Goldfain B, et al. (2016) Aggressive driving with model predictive path integral control. In: International conference on robotics and automation, Stockholm, Sweden, 16–21 May 2016
- Wu Z, Tran A, Rincon D, et al. (2019) Machine learning-based predictive control of nonlinear processes. Part I: theory. *AI-ChE Journal* 65: e16729
- Xu X, Li J, Yang Y, et al. (2020) Towards effective intrusion detection using log-cosh conditional variational AutoEncoder. In: Internet of things journal. IEEE
- Yang S, Wan MP, Chen W, et al. (2020) Model predictive control with adaptive machine-learning-based model for building energy efficiency and comfort optimization. *Applied Energy* 271: 115147

- Yu L, Xu L and Jiang X (2023) An Effective Method for Detecting Unknown Types of Attacks Based on Log-Cosh Variational Autoencoder. *Applied Sciences* 13(22): 12492.
- Zagal JC and Lipson H (2009) Self-reflection in evolutionary robotics: resilient adaptation with a minimum of physical exploration. In: Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. ACM
- Zhang H, Reardon C, Han F, et al. (2016) SRAC: self-reflective risk-aware artificial cognitive models for robot response to human activities. In: International conference on robotics and automation, Stockholm, Sweden, 16–21 May 2016
- Zhang S, Pöhlmann R, Wiedemann T, et al. (2020) Self-aware swarm navigation in autonomous exploration missions. *Proceedings of the IEEE* 108: 1168–1195.
- Zhu Q, Han Y, Cai C, et al. (2018) Robust optimal navigation using nonlinear model predictive control method combined with recurrent fuzzy neural network. *Mathematical Problems in Engineering* 2018: 1–19