Enhancing Consistent Ground Maneuverability by Robot Adaptation to Complex Off-Road Terrains
Supplementary Material

Sriram Siva[1], **Maggie Wigness**[2], **John G. Rogers**[2], and **Hao Zhang**[1]
[1]Colorado School of Mines, Golden, CO 80401, USA
{sivasriram,hzhang}@mines.edu
[2]U.S. Army Research Laboratory, Adelphi, MD 20783, USA
{maggie.b.wigness.civ,john.g.rogers59.civ}@army.mil

In this supplementary material document, Section 1 presents the proof of convergence for the proposed optimization algorithm presented in the main paper. Section 2 presents the implementation details of our approach and the software used. Section 3 provides a discussion on the discriminative features as estimated by our approach. Finally, in Section 4, we provide a summary of our approach's novel contributions, advancements over the state-of-the-art, assumptions and limitations.

1 Proof of Convergence for the Proposed Optimization Algorithm

In this section, we prove that Algorithm 1 (in the main paper) decreases the value of the objective function in Eq. (5) (in the main paper) in each iteration, and the algorithm converges to the global optimal solution.

First, we present a lemma:

Lemma 1. *Given any two vectors* $\mathbf{a}$ *and* $\mathbf{b}$*, the following inequality holds:*

$$\|\mathbf{b}\|_2 - \frac{\|\mathbf{b}\|_2^2}{2\|\mathbf{a}\|_2} \leq \|\mathbf{a}\|_2 - \frac{\|\mathbf{a}\|_2^2}{2\|\mathbf{a}\|_2}$$

Proof.

$$-(\|\mathbf{b}\|_2 - \|\mathbf{a}\|_2)^2 \leq 0$$
$$-\|\mathbf{b}\|_2^2 - \|\mathbf{a}\|_2^2 + 2\|\mathbf{b}\|_2\|\mathbf{a}\|_2 \leq 0$$
$$2\|\mathbf{b}\|_2\|\mathbf{a}\|_2 - \|\mathbf{b}\|_2^2 \leq \|\mathbf{a}\|_2^2$$
$$\|\mathbf{b}\|_2 - \frac{\|\mathbf{b}\|_2^2}{2\|\mathbf{a}\|_2} \leq \|\mathbf{a}\|_2 - \frac{\|\mathbf{a}\|_2^2}{2\|\mathbf{a}\|_2}$$

□

From Lemma 1, we can derive the following corollary:

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

$$\|\mathbf{B}\|_F - \frac{\|\mathbf{B}\|_F^2}{2\|\mathbf{A}\|_F} \leq \|\mathbf{A}\|_F - \frac{\|\mathbf{A}\|_F^2}{2\|\mathbf{A}\|_F}$$

Theorem 1. *Algorithm 1 (in the main paper) converges to the global optimal solution to the formulated regularized optimization problem in Eq. (5) (in the main paper).*

Proof. According to Step 4 of Algorithm 1, in the s-th iteration, the value of $\mathbf{W}^{(k)}(s+1)$ can be calculated as:

$$\mathbf{W}^{(k)}(s+1) = \|\boldsymbol{\mathcal{W}}(s) \otimes_3 \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{U}}(s) \otimes_3 \boldsymbol{\mathcal{E}} - \mathbf{A}\|_F^2 + \lambda_1 Tr(\mathbf{W}^{(k)}(s))^\top \mathbf{Q}^{(k)}(s+1)\mathbf{W}^{(k)}(s) \quad (1)$$

where the operator Tr denotes the trace of a matrix and $\mathbf{Q}^{(k)}(s+1) = \frac{1}{2\|\mathbf{W}^{(k)}(s)\|_F}\mathbf{I}_{d^i}$.

5th Conference on Robot Learning (CoRL 2021), London, UK.

From Step 6 of Algorithm 1, we obtain:

$$\mathbf{U}^i(s+1) = \|\boldsymbol{\mathcal{W}}(s+1) \otimes_3 \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{U}}(s) \otimes_3 \boldsymbol{\mathcal{E}} - \mathbf{A}\|_F^2 + \lambda_2 Tr(\mathbf{U}^{(k)}(s))^\top \mathbf{P}(s+1)\mathbf{U}^{(k)}(s) \quad (2)$$

where $\mathbf{P}(t+1) = \frac{1}{2\|\mathbf{U}^{(k)}\|_F}\mathbf{I}_r$.

Then, we can derive that:

$$\begin{aligned} &\mathcal{J}(s+1) + \sum_{k=1}^{c} \lambda_1 Tr(\mathbf{W}^{(k)}(s+1))^\top \mathbf{Q}^{(k)}(s+1)\mathbf{W}^{(k)}(s+1) \\ &\leq \mathcal{J}(s) + \sum_{k=1}^{c} \lambda_1 Tr(\mathbf{W}^{(k)}(s))^\top \mathbf{Q}^{(k)}(s)\mathbf{W}^{(k)}(s) \end{aligned} \quad (3)$$

where $\mathcal{J}(s) = \|\boldsymbol{\mathcal{W}}(s) \otimes_3 \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{U}}(s) \otimes_3 \boldsymbol{\mathcal{E}} - \mathbf{A}\|_F^2$.

After substituting $\mathbf{Q}^{(k)}$, we obtain:

$$\mathcal{J}(s+1) + \sum_{i=1}^{m}\sum_{k=1}^{c}(\lambda_1 \frac{\|\mathbf{W}^{i(k)}(s+1)\|_F^2}{2\|\mathbf{W}^{i(k)}(s)\|_F}) \leq \mathcal{J}(s) + \sum_{i=1}^{m}\sum_{k=1}^{c}(\lambda_1 \frac{\|\mathbf{W}^{i(k)}(s)\|_F^2}{2\|\mathbf{W}^{i(k)}(s)\|_F}) \quad (4)$$

From Lemma 1 and Corollary 1, $\forall k = 1, \ldots, c$, we obtain:

$$\sum_{k=1}^{c}\left(\|\mathbf{W}^{i(k)}(s+1)\|_F - \frac{\|\mathbf{W}^{i(k)}(s+1)\|_F^2}{2\|\mathbf{W}^{i(k)}(s)\|_F}\right) \leq \sum_{k=1}^{c}\left(\|\mathbf{W}^{i(k)}(s)\|_F - \frac{\|\mathbf{W}^{i(k)}(s)\|_F^2}{2\|\mathbf{W}^{i(k)}(s)\|_F}\right) \quad (5)$$

Adding Eqs. (4) and (5) on both sides, we obtain:

$$\mathcal{J}(s+1) + \sum_{i=1}^{m}(\lambda_1\|\mathbf{W}^i(s+1)\|_F) \leq \mathcal{J}(s) + \sum_{i=1}^{m}(\lambda_1\|\mathbf{W}^i(s)\|_F) \quad (6)$$

Eq. (6) shows that, given a fixed $\boldsymbol{\mathcal{U}}$ when updating $\boldsymbol{\mathcal{W}}$, the updated $\boldsymbol{\mathcal{W}}$ decreases the value of the objective function in each iteration.

Using the updated $\boldsymbol{\mathcal{W}}$, we can derive that:

$$\begin{aligned} &\mathcal{F}(s+1) + \sum_{k=1}^{c}(\lambda_2 Tr(\mathbf{U}^{(k)}(s+1))^\top \mathbf{P}(s+1)(\mathbf{U}^{(k)}(s+1))) \\ &\leq \mathcal{F}(s) + \sum_{k=1}^{c}(\lambda_2 Tr(\mathbf{U}^{(k)}(s))^\top \mathbf{P}(s)(\mathbf{U}^{(k)}(s))) \end{aligned} \quad (7)$$

where $\mathcal{F}(s) = \|\boldsymbol{\mathcal{W}}(s+1) \otimes_3 \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{U}}(s) \otimes_3 \boldsymbol{\mathcal{E}} - \mathbf{A}\|_F^2$. After substituting $\mathbf{P}$, we obtain:

$$\mathcal{F}(s+1) + \sum_{k=1}^{c}(\lambda_2 \frac{\|\mathbf{U}^{(k)}(s+1)\|_F^2}{2\|\mathbf{Q}^{(k)}(s)\|_F}) \leq \mathcal{F}(s) + \sum_{k=1}^{c}(\lambda_2 \frac{\|\mathbf{U}^{(k)}(s)\|_F^2}{2\|\mathbf{U}^{(k)}(s)\|_F}) \quad (8)$$

Similar to Eq. (5), from Lemma 1 and Corollary 1, $\forall k = t, \ldots, t-c$, we obtain:

$$\sum_{k=1}^{c}\left(\|\mathbf{U}^{(k)}(s+1)\|_F - \frac{\|\mathbf{U}^{(k)}(s+1)\|_F^2}{2\|\mathbf{U}^{(k)}(s)\|_F}\right) \leq \sum_{k=1}^{c}\left(\|\mathbf{U}^{(k)}(s)\|_F - \frac{\|\mathbf{U}^{(k)}(s)\|_F^2}{2\|\mathbf{U}^{(k)}(s)\|_F}\right). \quad (9)$$

Adding Eqs. (8) and Eq. (9) on both sides, we obtain:

$$\mathcal{F}(s+1) + \sum_{k=1}^{c}(\lambda_2\|\mathbf{U}^{(k)}(s+1)\|_F) \leq \mathcal{F}(s) + \sum_{k=1}^{c}(\lambda_2\|\mathbf{U}^{(k)}(s)\|_F) \quad (10)$$

Eq. (10) shows that, given fixed $\boldsymbol{\mathcal{W}}$ when updating $\boldsymbol{\mathcal{U}}$, the updated $\boldsymbol{\mathcal{U}}$ decreases the value of the objective function in each iteration.

Each iteration of Algorithm 1 includes both Eq. (6) and Eq. (10). By adding these equations on both sides, we obtain:

$$\begin{aligned}&\mathcal{J}(s+1)+\mathcal{F}(s+1)+\|\boldsymbol{\mathcal{W}}(s+1)\|_M+\|\boldsymbol{\mathcal{U}}(s+1)\|_T\\&\leq \mathcal{J}(s)+\mathcal{F}(s)+\|\boldsymbol{\mathcal{W}}(s)\|_M+\|\boldsymbol{\mathcal{U}}(s)\|_T\end{aligned}\tag{11}$$

At the end of each iteration of Algorithm 1, the loss function $\mathcal{F}$ is always less than or equal to $\mathcal{J}$, as $\mathcal{J}$ reduces the loss value by optimizing over $\boldsymbol{\mathcal{W}}$, and $\mathcal{F}$ reduces the loss value by optimizing over both $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{\mathcal{U}}$. Then we can write:

$$\mathcal{F}(s+1)+\|\boldsymbol{\mathcal{W}}(s+1)\|_M+\|\boldsymbol{\mathcal{U}}(s+1)\|_T\leq \mathcal{J}(s+1)+\|\boldsymbol{\mathcal{W}}(s+1)\|_M+\|\boldsymbol{\mathcal{U}}(s+1)\|_T\tag{12}$$

Similarly, as the loss function $\mathcal{J}(s+1)$ is always less than or equal to $\mathcal{F}(s)$ as $\mathcal{J}(s+1)$ reduces the value of loss further by optimizing $\boldsymbol{\mathcal{W}}(s)$, we can write:

$$\mathcal{J}(s+1)+\|\boldsymbol{\mathcal{W}}(s+1)\|_M+\|\boldsymbol{\mathcal{U}}(s)\|_T\leq \mathcal{F}(s)+\|\boldsymbol{\mathcal{W}}(s)\|_M+\|\boldsymbol{\mathcal{U}}(s)\|_T\tag{13}$$

From Eq. (11), (12) and (13), we can write the following inequality:

$$\begin{aligned}&\mathcal{F}(s+1)+\|\boldsymbol{\mathcal{W}}(s+1)\|_M+\|\boldsymbol{\mathcal{U}}(s+1)\|_T\leq \mathcal{J}(s+1)+\|\boldsymbol{\mathcal{W}}(s+1)\|_M+\|\boldsymbol{\mathcal{U}}(s+1)\|_T\\&\leq \mathcal{J}(s+1)+\|\boldsymbol{\mathcal{W}}(s+1)\|_M+\|\boldsymbol{\mathcal{U}}(s)\|_T\leq \mathcal{F}(s)+\|\boldsymbol{\mathcal{W}}(s)\|_M+\|\boldsymbol{\mathcal{U}}(s)\|_T\end{aligned}\tag{14}$$

From Eq. (14) we have:

$$\mathcal{F}(s+1)+\|\boldsymbol{\mathcal{W}}(s+1)\|_M+\|\boldsymbol{\mathcal{U}}(s+1)\|_T\leq \mathcal{F}(s)+\|\boldsymbol{\mathcal{W}}(s)\|_M+\|\boldsymbol{\mathcal{U}}(s)\|_T\tag{15}$$

Eq. (15) shows that the value of the objective function is decreased in each iteration. Because the objective function is convex, Algorithm 1 (in the main paper) converges to the global optimal solution to the formulated regularized optimization problem in Eq. (5) (in the main paper). □

2 Implementation of the Algorithm and Robot Software System

2.1 Algorithm Implementation and Training/Execution Procedures

The optimization algorithm is executed during the offline training phase, and no online optimization is used at the test/execution phase. We provide the terrain feature tensor $\boldsymbol{\mathcal{X}}$, the expected behaviors $\mathbf{Y}$, the actual behaviors $\mathbf{A}$, and the behavior difference tensor $\boldsymbol{\mathcal{E}}$ as input variables to our optimization algorithm. For each expected behavior $\mathbf{Y}$, the robot estimates its actual behavior $\mathbf{A}$, so the dimensionality of $\mathbf{A}$ and $\mathbf{Y}$ is the same. Our algorithm then alternatively optimizes weight tensors $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{\mathcal{U}}$, over various iterations and converges to the global optimal solution.

We train our approach on a dataset recorded while an expert (a human in our case) demonstrates robot driving over individual types of terrains, including grass, sand, gravel, medium-sized rocks, and large-sized rocks. The recorded data includes the robot's observations from IMU, RGBD camera, and LiDAR sensors, the robot's actual behavior, and the expected behavior demonstrated by the expert. The training dataset includes approximately 20,000 instances from nearly 5.5 hours of driving. Each instance includes the inputs from c time steps as defined in the main paper. This dataset is used to train our algorithm and identify the optimal hyperparameter values (Figs. 5 and 6 in the main paper) in the training phase. The optimal hyper-parameter we obtain for our training data are: $\lambda_1 = 0.1$, $\lambda_2 = 0.1$ and $c = 15$. In the testing phase, our approach uses the parameters learned during the training phase without additional online learning/optimization. Although our approach is trained using data obtained from individual types of terrains, we include unseen off-road terrains (e.g., grass-medium rocks, and mixed terrains) during testing to evaluate our approach's ability to let ground robots generate consistent behaviors in unfamiliar terrains.

The computational cost increases quadratically with c in each iteration of our algorithm. The number of iterations to convergence may also increase with the increase of c (which unfortunately does not have a mathematical bound). In practice, when the algorithm is trained using the dataset on the robot's onboard computer that has a 4.3 GHz Intel i7 processor and 16GB memory with no GPU, it takes nearly 3.5 hours for the algorithm to converge when $c = 15$, and nearly 21 hours when $c = 40$. We did not test the cases when $c > 40$ because the performance has already significantly dropped when $c > 30$ (Fig. 6 of the main paper). The computational cost is linearly dependent on n.

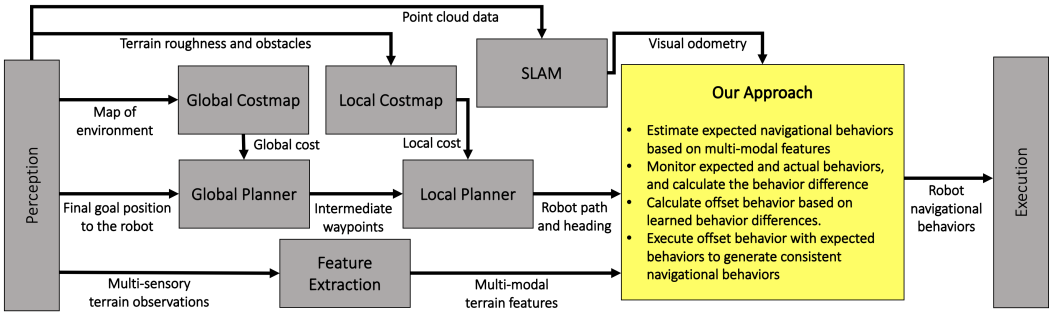

Figure 1: Overview of the ROS-based software architecture implemented on our physical robots for performing consistent behavior generation. The yellow box shows our approach and the remaining modules that work with the proposed approach in our software architecture are shown by gray boxes.

Similar to most ML methods, more training data from more diverse scenarios (e.g., diverse terrains for ground navigation in our case) often improves the results. However, training with more data also increases computational cost. It is not easy to determine the impacts of data quality on the results unfortunately. First, it is difficult to estimate data quality as it is affected by many factors such as sensor types and quality (e.g., resolution and noise), feature extraction, and human/AI-agent demos. Second, the learning model is optimized based on the current quality of data. When data quality changes (e.g., by switching to better feature extractions), the model may not be applicable anymore.

2.2 Robot Software System Implementation

We use a Clearpath Husky robot in our field experiments. The robot runs on Ubuntu 18.04 and Robot Operating System (ROS) Melodic as the operating system. Our approach is implemented using C++ as a ROS package, which runs in real time at 30 Hz on an Intel 4.3 GHz i7 processor onboard the robot without requring a GPU. The implementation code is provided as a part of the supplementary material. Instructions of executing our ROS package is also included in the submitted implementation code.

Fig. 1 provides an overview of the ROS-based software architecture implemented on our physical robots to perform consistent navigational behavior generation. Implemented as a ROS package, our approach works with many other packages within ROS. Our approach works as a local controller to generate navigational controls, and it works together with the packages of global and local planners. The global planner generates a global path using a cost map and offers the local planner intermediate way-points that lead the robot to the goal position. The local planner employs point cloud data from the perception package to detect obstacles near the robot. Then, it generates a local path for the robot to avoid the obstacles between the way-points. The local path calculated from the local planner is then passed onto our approach, which generates consistent navigational behaviors to make the robot to follow the path. Specifically, our approach computes expected navigational behaviors from terrain feature vectors extracted by the feature extraction package. Our approach continuously monitors the difference between the robot's expected behaviors and actual behaviors that are estimated using visual odometry or SLAM. Then, then our approach calculates offset behaviors based on a sequence of past behavior differences. These offset behaviors are applied to compensate for the behavior differences in order to achieve consistent ground navigation when the robot traverse over unstructured off-road terrains.

3 Results on Discriminative Feature Modalities

Our approach has the ability to automatically estimate the importance of various feature modalities during the training phase. The visualization of the multimodal features used in our experiments is provided in Fig. 2. The importance of feature modalities is estimated from the optimal weight tensor $\mathcal{W}$, which is learned on the training data from all individual types of unstructured terrains. As no

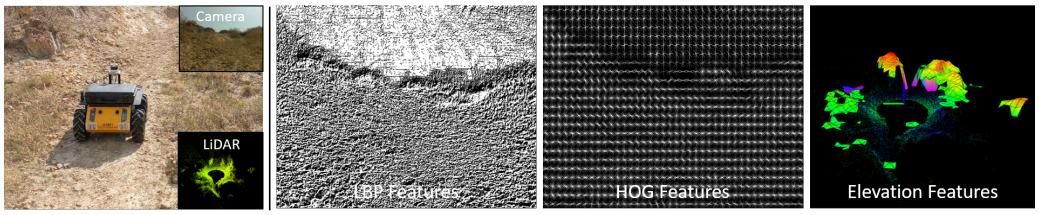

Figure 2: Multimodal features used to characterized terrain during ground navigation.

additional training is performed for the complex off-road unstructured terrains, the importance of feature modalities remains the same for both experimental scenarios.

The results on the importance of feature modalities are demonstrated in Fig. 3. It is observed that the HOG features are the most important and have a relative importance of 48.3%. The IMU features are observed to be the second most important feature modality with a relative importance of 26.7%. The grid-wise elevation features extracted from a robot's LiDAR sensor are relatively less important and have a relative importance of 13.6%. The least important feature modality is LBP and has a relative importance of 11.3%.

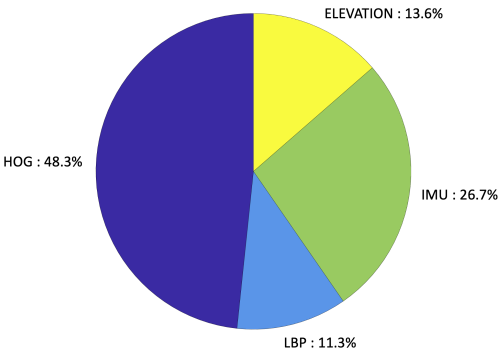

Figure 3: Relative importance of the feature modalities estimated by our approach to enable consistent off-road ground navigation.

The HOG features have relatively high importance because they capture the terrain shape information, which better characterize the terrain compared to terrain textures encoded by the LBP features. During training, the robot is controlled by a human demonstrator in a manner that reduces the jerkiness of robot navigation. This jerkiness metric is more correlated with the IMU data. Thus, we observe that the IMU features are also of high importance. The results in Fig. 3 also show that elevation maps are less important. This is probably caused by the fact that the used grid size (25 cm $\times$ 25 cm) cannot capture the essential characteristics of off-road terrains.

4 Summary of Our Approach's Novelties, Advancements, and Limitations

We further detail our approach's novelty, advancement on the state-of-the-art, and assumptions and limitations as follows.

4.1 Novel Contributions

The main novelty of our approach is the introduction of a mathematical formulation that formulates the real-world problem of consistent ground navigation over diverse off-road terrains into a mathematical problem. This formulation realizes our new idea of generating consistent navigation through learning offset behaviors that adapt a robot's navigation to various unstructured terrains. Our approach achieves this capability by continuously monitoring the difference between the robot's actual and expected behaviors. Furthermore, our formulation enables consistent navigation without explicitly modeling the setbacks that cause the behavior difference. Our formulation also fuses multi-modal features to enable terrain-aware ground navigation and automatically estimates the importance of terrain features, as seen in Fig. 3. This ability is implemented through a structured norm over parameter tensors as a regularization term under the unified mathematical formulation.

The second novelty of our paper is the implementation of a new optimization algorithm to solve the formulated regularized optimization problem, which holds a theoretical guarantee to effectively converge to the global optimal solution. The formulated optimization problem introduced in Eq. (5) (of the main paper) is challenging to solve because the regularization terms are not smooth and

because the objective function includes dependent variables. Since the regularization terms cannot be differentiated at non-smooth points during optimization, second-order optimization algorithms (such as Newton's or Secant's method [1, 2]) are not applicable. To address the non-smooth regularization terms and dependent variables, we design a new alternating minimization algorithm. Our algorithm can be viewed as a specialized version of gradient descent. Given our specific objective function, the solution must be mathematically derived. Although gradient descent is a fundamental mathematical tool, deriving a closed-form solution with the convergence guarantee is not always possible. Our algorithm is tailored and mathematically derived to solve our formulated optimization problem and provides a closed-form solution that is guaranteed to converge to the global optima monotonically and fast, as can be seen from Fig. 7 (of the main paper).

As a practical contribution, we provide a comprehensive evaluation of learning-based terrain adaptation methods by designing a set of live-robot navigation scenarios using physical ground robots over various unstructured off-road terrains.

4.2 Advancements of Our Approach

Our approach advances the state-of-the-art in both the problem domain and the solution domain. In the problem domain, we address the problem of robots' consistent ground navigational over unstructured off-road terrain. This problem has not been well studied in the field of off-road navigation and causes slower robot traversal or uncertain traversal times, further resulting in the robot failing to execute navigational tasks on time. Moreover, inconsistent navigational behaviors also result in robot localization errors and need to be addressed for successful robot off-road ground navigation.

In the solution domain, previous learning-based methods for off-road robot navigation generally ignore behavior inconsistency caused due to setbacks. Our approach advances the state-of-the-art by introducing a novel approach with a mathematical formulation and an optimization algorithm that enables consistent ground navigation with solid mathematical support and experimental evaluation.

4.3 Assumptions and Limitations

Our approach is based on two main assumptions: First, we assume the LiDAR-based SLAM method used in our autonomy stack as seen in Fig. 1, provides an accurate estimation of the robot's actual behaviors. However, this assumption may not be satisfied when robots operate in feature-sparse environments such as a hallway with white walls. With inaccurate SLAM, the generated offset behaviors can be sub-optimal. On the other hand, our focus is the unstructured off-road environment (e.g., in a forest), which is often feature rich. So our assumption is satisfied. If not, we may use SLAM methods designed for feature-sparse scenarios [3, 4] or use a Visual Inertial Navigation System (VINS) [5] in environments with visual texture. The second assumption we make is that the dynamics of the robot do not dramatically change over a shorter period of time (e.g., within one second), and the non-linearity in the robot is not severe. This assumption holds as our approach only works as a local controller and moreover, can be easily satisfied when we use robot observations from the past 0.5 seconds, i.e., a sequence length of $c = 15$ frames.

In addition, our approach also makes common assumptions such as the expert demonstrations are reasonably good and robot functions correctly without failures in actuation. Different sensor periodicities are often addressed at the robot's hardware level, e.g., using a MasterClock in our case (which is a precise timing system to synchronize all sensor measurements). In addition, our approach does not explicitly assume that all sensor measurements are always available, but it also does not explicitly address the problem of missing data (e.g., caused by sensor failure or occlusion). Implicitly, our approach is robust to missing data because it fuses measurements from multiple sensors to generate navigational behaviors. That is, if one sensor fails we still have information from the other sensors to generate behaviors. Each sensor contributes differently to navigational behavior generation. For example from Fig. 3, we observe that features obtained from the color camera (HOG and LBP) have a relative importance of $\sim 60\%$, and features from IMU measurements have a relative importance of $\sim 27\%$. Thus, missing measurements from the color camera is more severe than missing IMU data.

Our approach has two limitations: First, our approach is a local controller that works with an external local planner, as seen in Fig. 1. This local planner does not have terrain awareness, which caused most failures in our experiments, including the failure case in the video. The second limitation of

our approach is from the assumption that the LiDAR-based SLAM method in the robot's autonomy can provide good estimations of the robot's actual behavior. However, as mentioned earlier, this assumption doesn't hold in feature-sparse environments and would lead our approach to generate sub-optimal offset behaviors.

References

[1] A. Fischer. A special Newton-type optimization method. *Optimization*, 24(3-4):269–284, 1992.

[2] R. Tapia. On secant updates for use in general constrained optimization. *Mathematics of Computation*, 51(183):181–202, 1988.

[3] Z. Ren, L. Wang, and L. Bi. Robust GICP-based 3D LiDAR SLAM for underground mining environment. *Sensors*, 19(13):2915, 2019.

[4] Y. Song, M. Guan, W. P. Tay, C. L. Law, and C. Wen. UWB/LiDAR fusion for cooperative range-only SLAM. In *International Conference on Robotics and Automation*, 2019.

[5] J. Zhang, M. Ren, P. Wang, J. Meng, and Y. Mu. Indoor Localization Based on VIO System and Three-Dimensional Map Matching. *Sensors*, 20(10):2790, 2020.