# Fast, Safe, and Proactive Runtime Planning and Control of Autonomous Ground Vehicles in Changing Environments

Grace Glaubit†
*Computer Science*
*University of Virginia*
Charlottesville, USA
gsg6bu@virginia.edu

Katie Kleeman†
*Engineering Systems and Env.*
*University of Virginia*
Charlottesville, USA
knk9ab@virginia.edu

Noelle Law†
*Electrical and Comp. Eng.*
*University of Virginia*
Charlottesville, USA
nl9sb@virginia.edu

Jeremiah Thomas†
*Engineering Systems and Env.*
*University of Virginia*
Charlottesville, USA
jt6fz@virginia.edu

Shijie Gao
*Electrical and Comp. Eng.*
*University of Virginia*
Charlottesville, USA
sg9dn@virginia.edu

Rahul Peddi
*Engineering Systems and Env.*
*University of Virginia*
Charlottesville, USA
rp3cy@virginia.edu

Esen Yel
*Engineering Systems and Env.*
*University of Virginia*
Charlottesville, USA
ey3un@virginia.edu

Nicola Bezzo
*Engineering Systems and Env.*
*University of Virginia*
Charlottesville, USA
nb6be@virginia.edu

2021 Systems and Information Engineering Design Symposium (SIEDS) | 978-1-6654-1250-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/SIEDS52267.2021.9483719

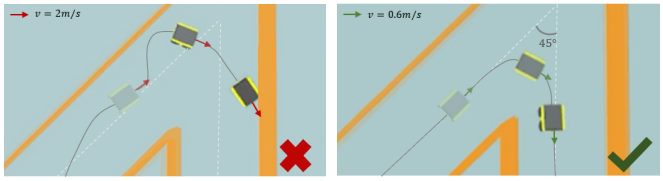
***Abstract*—Autonomous ground vehicles (UGVs) traversing paths in complex environments may have to adapt to changing terrain characteristics, including different friction, inclines, and obstacle configurations. In order to maintain safety, vehicles must make adjustments guided by runtime predictions of future velocities. To this end, we present a neural network-based framework for the proactive planning and control of an autonomous mobile robot navigating through different terrains. Using our approach, the mobile robot continually monitors the environment and the planned path ahead to accurately adjust its speed for successful navigation toward a desired goal. The target speed is selected by optimizing two criteria: (1) minimizing the rate of change between predicted and current vehicle speed and (2) maximizing the speed while staying within a safe distance from the desired path. Additionally, we introduce random noise into the network to model sensor uncertainty and reduce the risk of predicting unsafe speeds. We extensively tested and validated our framework on realistic simulations in Gazebo/ROS with a UGV navigating cluttered environments with different terrain frictions and slopes.**

***Index Terms*—motion planning, terrain traversability, neural network, unmanned ground vehicles**

## I. Introduction

Unmanned ground vehicles (UGVs) are increasingly popular for a wide range of use cases. The autonomous car industry is booming, with a market set to reach $42 billion by 2025[1]. Advances in data collection capabilities make UGVs an ideal choice for Intelligence, Surveillance, and Reconnaissance (ISR) missions and other dangerous tasks such as collapsed cave exploration and detection of hazards.

An important part of these tasks is balancing safety and efficiency, which is especially hard to do in unknown terrains. Complex environments that combine difficult maneuvers, tight turns, hills, and slippery or rough surfaces can cause issues with existing control algorithms (Fig. 1). For example, an autonomous truck can maintain a constant speed as it maneuvers its route under normal conditions, but it must slow down to safely execute turns under icy road conditions. If a UGV moves too fast, it could reach an unsafe state like crashing into an obstacle (Fig. 1(a)). Driving too slowly may be safer, but being too conservative is inefficient. Therefore, it is important for a UGV to maximize its speed while maintaining safety.




(a) (b)

Fig. 1: Example of UGV traversing over an icy surface. The UGV in (a) moves at its maximum speed, consequently crashing while (b) limits its speed based on surface conditions and the upcoming turn, and successfully maneuvers the path.

In this work, we propose a neural network (NN)-based framework for proactive control of an autonomous UGV navigating complex terrains. We consider the planned path and the friction, obstacles, and ramps in the environment to find the maximum speed that the UGV can operate at and still complete its mission. Our framework proactively adjusts the UGV's speed based on the environment's friction to meet future safe speed targets. It also maintains a safe deviation threshold to avoid obstacles near the planned trajectory. This framework could be especially useful for time-critical missions.

The paper is organized as follows: Sec. II reviews the state of the art for UGV navigation in complex environments, Sec. III covers preliminaries, Sec. IV provides our problem statement, Sec. V outlines our approach, Sec. VI details simulations, and Sec. VII discusses conclusions and future work.

† First four authors contributed equally to the paper

[1]https://www.digitalistmag.com/improving-lives/2019/04/02/rise-of-autonomous-vehicles-why-ethics-matter-06197534/

978-1-6654-1250-6/21/$31.00 ©2021 IEEE

## II. Literature Review

The predominant approach for UGV navigation in challenging environments is learning-based methods. In [1], a temporal-based NN architecture for UGV navigation is used to generate a steering command. The approach in [2] presents navigation through complex environments using an unsupervised learning algorithm that adapts driving control based on surface classification. However, this work does not consider turning radius in maximum velocity determination.

In [3], a general approach to long-range path planning in challenging terrains is proposed, in which a learned model uses partial knowledge to predict local motions of a robot. Meanwhile, [4] proposes a risk-averse path-planning algorithm that identifies potential traversable paths, ranked in terms of safety and distance. Similarly, [5] provides a reinforcement learning-based approach for safe local planning to navigate a vehicle through unknown terrain.

Many NN-based approaches for autonomous navigation do not consider model uncertainty. The work in [6] proposes a method to handle model uncertainty in autonomous driving scenarios by using dropout networks to predict vehicle crashes in advance. Uncertainty and error are modelled in [7] through formulating traversabilty as a probabilistic feature, in which locally spaced elevation differences are verified or discarded.

Our work considers similar challenges to the aforementioned work, and builds a NN-based navigation system that allows a UGV to successfully traverse complex environments by generating a safe and efficient forward velocity command while also considering model uncertainty.

## III. Preliminaries

### A. UGV System Dynamics

In this work, we model a UGV as a differential drive robot, which is steered by independently controlling the speed of the wheels on either side of the robot. The left-hand wheels of the UGV move at a speed of $v_L$ while the right-hand wheels move at a speed of $v_R$. The wheels are separated by distance $W$. The equations of motion are as follows:

$$\dot{x} = \frac{(v_R + v_L)\cos\theta}{2} \quad \dot{y} = \frac{(v_R + v_L)\sin\theta}{2} \quad \dot{\theta} = \frac{v_R - v_L}{W}$$

We chose this model because we use a simulated Clearpath Robotics Jackal UGV for training and testing, but our framework applies to any type of UGV.

### B. Environment and Obstacles

The objective of this work is to enable a vehicle to safely and efficiently navigate any challenging environment with different terrain properties.

The environment we use for training, testing, and implementation is comprised of a few key components. The first is the *frictional coefficient* of the surface, $\mu$, which determines how slippery the path is. We use a range of $\mu$ values from 0.05 to 1, which are typical of common surfaces. We limit our $\mu$ values at 1 as the UGV behavior does not change past 1. Since the focus of this paper is on proactive motion planning and control, we assume that $\mu$ for each surface is measured by onboard sensors: for example, use of lidar and cameras enable detection of roughness and type of surface ahead, similar to what is presented in [2].

The next component is the *angle* of the path on the x-y plane. Speed is dependent on this turning angle, as the robot must slow down to safely make tight turns. We further consider the angle of the surface along the z-axis to account for any ramps on the desired path of the UGV. We must distinguish between the three variables we use to refer to angles in this work. We represent values describing the turning angle of the path $\boldsymbol{\tau}$ on a flat surface with $\psi_\tau$. The angle input into our NN (explained in more detail in Section V-B2) is represented by $\theta$, and the orientation of the robot is represented with $\phi$. We distinguish between the orientation of the robot, the orientation of the robot's motion, and the desired orientation with the subscripts $x$, $m$, and $r$, respectively.

*Obstacle configuration* is the last environmental component considered. This determines the deviation threshold ($\delta_\tau$) depending on UGV placement, which is the distance the UGV is allowed to deviate from the given path before being classified as unsafe. For example, if the UGV is driving through an open field, the $\delta_\tau$ would be large, but in a forest it would be smaller so it could safely navigate between trees.

These components encompass the main aspects of a typical environment a UGV would encounter, and most environments can be modeled by changing them.

## IV. Problem Formulation

The goal of this work is to adapt control policy so that a vehicle is able to maximize its speed while ensuring safety. We aim to find the fastest safe speed the robot can go depending on the friction of the surface, the turning angle, and the presence of ramps. This requires the identification of both $\psi_\tau$ and the angle of the ramp. Once these are identified, a maximum safe speed is predicted. Stated more formally:

**Problem I: Speed Adaptation in Uncertain Terrains** *Consider a UGV with dynamics as discussed in Section III-A traversing uncertain environments, as explained in Section III-B following a predefined obstacle-free path $\boldsymbol{\tau}$. Develop a policy that dynamically adjusts speed to stay at all times within a safe deviation from the path, $\delta_\tau$, to avoid collision with any obstacle, and successfully reach the final goal $\boldsymbol{x}_g$: Mathematically, the UGV needs to compute a speed $\boldsymbol{v}$ such that:*

$$\|\boldsymbol{x}(t) - \boldsymbol{x}_\tau(t)\|_2 \leq \delta_\tau, \ \ \forall t > 0$$
$$\lim_{t\to\infty} \|\boldsymbol{x}(t) - \boldsymbol{x}_g\|_2 = 0$$

*where $\boldsymbol{x}(t)$ is the current position of the robot while $\boldsymbol{x}_\tau(t)$ is the robot's closest point to the trajectory $\boldsymbol{\tau}$.*

## V. Approach

Our proposed architecture uses a cascading system of NNs to allow a UGV to safely and efficiently navigate through complex environments. Fig. 2 provides a high-level overview of the approach followed in this work.

The vehicle navigates along a desired path by analyzing both features related to the planar path and any ramp surfaces in its environment. Planar characteristics, such as friction of the environment, upcoming turn angles, and the deviation

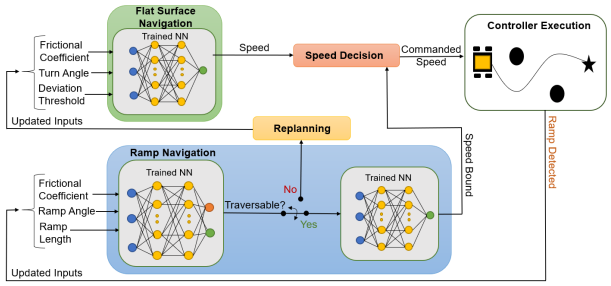



Fig. 2: High level system architecture.

threshold, are fed into the flat surface navigation NN, which outputs the maximum safe speed.

When a ramp is detected in the environment, the ramp navigation system will first use a NN to detect if it's traversable at any speed given ramp length, ramp angle, and friction of the surface. If the ramp is traversable, then ramp and environment characteristics are inputted into another NN that outputs the minimum or maximum speed the UGV can successfully traverse the ramp. This upper or lower bound is compared with the speed output of the flat surface navigation NN, and if the speed is within safety bounds, it is sent to the controller. Otherwise, the commanded speed is set based on the speed bound. If it is found that the vehicle is unable to traverse the ramp at any speed, the path is replanned to avoid the ramp.

In the following sections, we will discuss the details of the proposed framework starting with the controller, followed by the design of the flat surface and ramp navigation NNs, and finally how input uncertainty is modelled in the approach.

### A. Pure-Pursuit Control

The control algorithm is based on a pure-pursuit tracking algorithm [8] that controls steering so that the robot can move from its current location, $\boldsymbol{x}(t) = (x(t), y(t))$, to an intermediate goal along a predefined path, $\bar{\boldsymbol{x}}_g = (\bar{x}_g, \bar{y}_g)$. The goal location is updated as the robot approaches it and the path is revised to a new intermediate goal. The goal remains at a fixed distance from the vehicle, so the UGV will only reach the goal at the end of the trajectory. Pure pursuit is computationally fast and contains information about the future path and desired future robot state [8]. This information is leveraged to proactively send a control input to the robot in the present that will allow it to succeed in the future. Additionally, it allows for smoother behavior compared to go-to-goal algorithms as the UGV can take continuous turns on a smooth arc, rather than following harsh angles [8].

First, the path is discretized into $n$ points, $\boldsymbol{\tau} = \{\boldsymbol{x}_\tau^1, \boldsymbol{x}_\tau^2, \ldots, \boldsymbol{x}_\tau^n\}$, that serve as waypoints from the robot's starting position to the final goal. The intermediate goal $\bar{\boldsymbol{x}}_g$ used for the pure pursuit control is calculated by first finding the point in the path closest to the robot's current location, $\boldsymbol{x}_\tau^i \in \boldsymbol{\tau}$, and then adding a constant horizon distance $\bar{n} \leq n$ as follows:

$$\bar{\boldsymbol{x}}_g = \boldsymbol{x}_\tau^{i+\bar{n}}, \quad i \in \{1, \ldots, n\} \tag{1}$$

Once the intermediate goal is determined, the angular velocity, $\omega_z$, is calculated as follows:

$$\omega_z = \phi_r - \phi_x \quad \text{with} \quad \phi_r = \arctan \frac{y_g - y(t)}{x_g - x(t)} \tag{2}$$

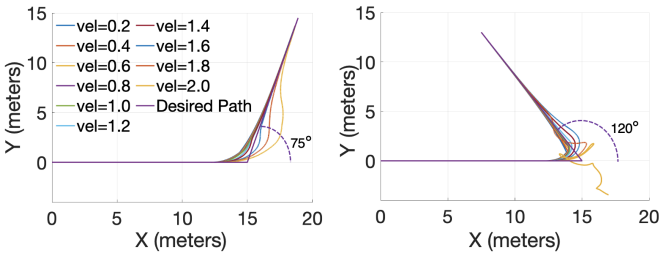
where $\phi_x$ is the orientation of the robot and $\phi_r$ is the desired reference orientation.

In ideal conditions, the UGV will travel along its desired path at its maximum speed. However, changes in various terrain properties make it necessary to proactively adapt its linear speed, as presented in the next section.

### B. Flat Surface Neural Network

The flat surface NN predicts the maximum safe speed of a UGV over a given flat terrain. The NN uses the friction coefficient of the environment ($\mu$), the deviation threshold ($\delta_\tau$), and the turn angle ($\theta$) in the upcoming path as inputs.

*1) Speed Data Collection and Processing:* To create this NN, speed training data were collected in environments of varying surface friction coefficients as the UGV executes trajectories of various $\psi_\tau$ and $\delta_\tau$ values. The UGV was instructed to travel at a constant speed for 15 meters, turn at a specified angle $\psi_\tau$, and then travel 15 meters to a goal point. Training angles ranged from 0° through 165° with increments of 15°. Frictional coefficients of $\mu$ = 1.0, 0.5, 0.09, 0.05, and 0.009 were used to simulate environments with varying levels of friction. The maximum speed for the simulated Jackal UGV is 2.0 m/s. The commanded speeds ranged from 0.2 to 2.0 m/s with a step of 0.2 m/s. To minimize the number of training runs, experiments were conducted with a $\delta_\tau$ of 4 m. Data for the remaining thresholds were then computed during post-processing. All combinations of these variables resulted in data for 600 trial runs, which were then processed for use in training the NN. Two examples of these training trajectories are shown in Fig. 3.

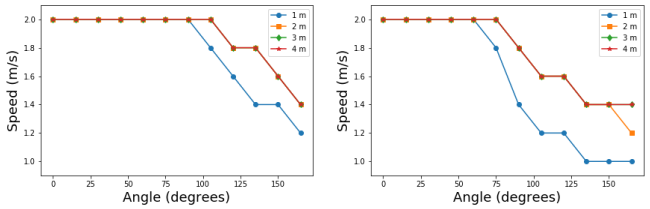



(a) $\mu = 0.05$, $\psi_\tau = 75°$ (b) $\mu = 0.09$, $\psi_\tau = 120°$

Fig. 3: Paths for various friction coefficients and angles.

The speed training data were first analyzed to determine if the UGV successfully reached the goal position. A run was classified as a success if the UGV reached the goal point without deviating from the path by more than the given threshold, and was classified as a failure otherwise. For example, in Fig. 3(b), the UGV is unable to make it to the goal position at a speed of 2.0 m/s. Data were then processed taking into account $\delta_\tau$. This resulted in 3600 training runs that indicate success or failure for each combination of speed, angle, $\mu$, and $\delta_\tau$. Once these data were computed, the maximum successful speed was found for each combination of angle, $\mu$, and $\delta_\tau$. An example of this output is shown in Fig. 4 for two values of $\mu$.

*2) Flat Surface Speed Prediction and Adaptation:* The flat surface NN depicted in Fig. 2 predicts safe speeds. The robot is always able to reach the goal at the minimum tested speed of 0.2 m/s, regardless of the surface friction. We trained a




(a) $\mu$ = 0.09 (b) $\mu$ = 0.05

Fig. 4: Maximum safe speeds for different environmental friction coefficients and deviation threshold.

network with 10 neurons in one hidden layer to predict the maximum safe speed given a $\mu$, $\theta$, and $\delta_\tau$ as inputs. The turn angle $\theta$ fed into the NN is calculated as follows:

$$\theta = \psi_\tau - \phi_x \tag{3}$$

where $\psi_\tau$ and $\phi_x$ are the angle of the path and the angle of the robot's motion with respect to the global coordinate system.

It is critical that the robot calculates both the angle at the look-ahead distance and the angles between that point and its current position, as is shown in Fig. 5. This ensures that the robot does not accelerate if there is a more imminent turn at a greater angle. The algorithm loops through the points on the path directly in front of the robot to the determined look-ahead distance, then calculates the angle at each point in between the two. Once $\theta$ at each point is calculated, the maximum angle is found and input into the NN.

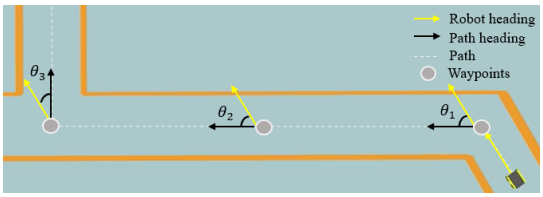



Fig. 5: Turn angle ($\theta$) calculation.

Entering a turn with too high of a speed can cause the robot to lose control and fail the maneuver. Therefore, the NN uses an asymmetrical mean square error (MSE) to make conservative predictions that are more likely to succeed. MSE is calculated normally during training but is biased to favor slower speeds.

Furthermore, to prevent over-correction from slippage, an alignment check is implemented in the algorithm. Once the desired speed is determined, the alignment check ensures the orientation of the robot, $\phi_x$, is aligned within a certain threshold, $\epsilon$, with the angle of the path, $\psi_\tau$.

$$|\phi_x - \psi_\tau| < \epsilon \tag{4}$$

The alignment condition is checked over $n_a \in \mathbb{N}$ consecutive iterations to ensure the robot is not only briefly aligned with the path. If this condition is met, the speed predicted by the NN is sent to the robot. Otherwise, the current speed is held until the condition is met unless the predicted speed is less than the current speed.

Once the robot determines a safe speed to maintain along the look-ahead distance, it can change its speed immediately or wait until it is closer to the turn to make the change. Inertia can cause a delay for the robot's actual speed to reach the commanded linear speed. To ensure that the robot can slow down in time to make the upcoming turn safely, we define a reaction distance based on $\mu$. For environments with a smaller value of $\mu$, the robot must start slowing down earlier to reach the commanded linear speed before approaching the turn. Therefore, slippery environments require a larger look-ahead distance than rough environments. Based on data collected on the stopping distance for each $\mu$ at maximum speed, different values of $\mu$ have been grouped into bins that determine the look-ahead distance as shown in Table I.

TABLE I: Categorization of look-ahead distance based on $\mu$.

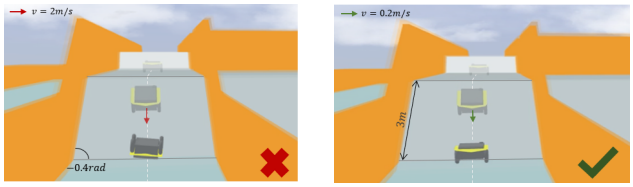
| Frictional Coefficient | Look-Ahead Distance |
|---|---|
| $\mu \geq 0.2$ | 2.5 m |
| $0.2 > \mu \geq 0.04$ | 5 m |
| $0.04 > \mu \geq 0.009$ | 10 m |
| $\mu < 0.009$ | 15 m |

### C. Ramp Neural Network

A similar procedure is deployed for ramp traversal.

*1) Ramp Data Collection and Processing:* Complex environments often contain sloped surfaces (e.g., when driving inside a parking lot, down a hill, etc.). Therefore, data were collected for vehicle traversal of ramps of varying lengths, slopes, and $\mu$. The ramps ranged from 0.5 m to 3 m in length, with an increment of 0.5 m. For each ramp length, angles of $\pm$0.4, $\pm$0.5, $\pm$0.55, and $\pm$0.6 radians (= $\pm$22.9°, $\pm$28.6°, $\pm$31.5°, $\pm$37.8°) were used, in which the sign indicates whether the UGV is to traverse the ramp upwards or downwards. Frictional coefficients, $\mu$, of 0.09, 0.5, and 1 were used for each length-angle combination. It was seen in simulation that $\mu$ values below 0.09 produced identical results, and so these experiments were not taken into consideration while training the NN to prevent over-weighting of the results.

Each ramp combination was tested with commanded speeds ranging from 0.1 m/s to 2.0 m/s, incrementing by 0.1 m/s. This produced 2880 trials which were analyzed to find the range of speeds that can be commanded to a UGV to safely traverse up or down a ramp of given parameters. Fig. 6(a) demonstrates an unsuccessful ramp traversal because the UGV executes the path at maximum speed. Fig. 6(b) shows the vehicle safely maneuvering the same environment by taking environment characteristics into consideration and limiting its speed.

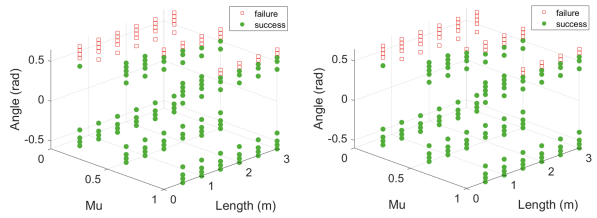



(a) (b)

Fig. 6: Example of UGV traversing a downwards ramp. The UGV in (a) moves at maximum speed, consequently flipping while (b) limits speed based on surface and ramp conditions.

*2) Ramp Traversability Prediction:* When a UGV discovers there is a ramp along its current trajectory, it must first determine if it can successfully traverse the ramp at any speed. The ramp traversal NN has 5 neurons in one hidden layer and takes ramp length, ramp angle, and the surface

friction coefficient as inputs. The NN outputs a probability as to whether or not the ramp can be successfully traversed, and any output less than 50% certainty is classified as not traversable. Fig. 7(a) depicts training data classification results for ramp traversability and Fig. 7(b) displays the associated NN prediction results. Our network produced one false positive and one false negative over 161 training points.




(a) Ramp training success (b) Ramp prediction success

Fig. 7: Training traversability classification vs. NN predictions.

*3) Ramp Speed Prediction, Adaptation, and Replanning:* If a ramp is traversable, the vehicle must determine the range of safe speeds to execute its desired path. Therefore, two NNs are designed to predict the speed bounds for ramp traversals. For downward ramps, the maximum speed bound is predicted given $\mu$, ramp angle, and length with an asymmetric MSE penalizing overpredictions. For upward ramps, the minimum speed bound is predicted given $\mu$, ramp angle, and length with an asymmetric MSE penalizing underpredictions. Both networks use 10 neurons in one hidden layer.

The speed decision block, seen in Fig. 2, compares the upper speed bound for downwards ramps and the lower speed bound for upwards ramps to the speed output of the flat surface navigation NN. If the speed is within safety bounds, it is sent to the controller. Otherwise, the commanded speed is set based on the upper or lower speed bound. This system ensures that the robot only changes speed when necessary.

However, if the planned path is found to be non-traversable, a new path must be calculated. There are a variety of existing path planning techniques that could work here such as greedy search [9], Dijkstra's algorithm [10], and A* search [11]. For the sake of computational simplicity and efficiency of our code, we used a modified version of A*. A* is a best-first search algorithm, however it has a very high space complexity because it stores all potential nodes. In our case study, discussed in section VI, we use a track with multiple paths, rather than a graph with many different path options. Therefore we modified the A* algorithm by keeping the focus on minimizing the cost function, but with predefined paths, rather than nodes and edges. The path options are ranked according to our cost function, distance, giving the path with the shortest distance top priority. If at some point the current path is determined to be non-traversable, the control will switch to the path with the next highest priority.

### D. Modeling Input Uncertainties

Environmental and sensor uncertainty were modeled by applying Gaussian noise to the NN inputs during training and prediction. This resulted in noisy outputs, with more noise signifying less certainty. Risk was reduced by using the variance in output values to adjust the speed given to the control algorithm. For a given set of inputs, we run the NN 10 times and calculated the mean and standard deviation of the outputs. For situations where overpredictions were less safe, 2 standard deviations were subtracted from the mean. Likewise, for situations where underpredictions were less safe, 2 standard deviations were added. Fig. 8 compares unadjusted predictions (purple) with adjusted predictions (green). Some of the unadjusted predictions fall into the unsafe speed range above the training data, but after adjustment, they are all within the successful speed range. This method also produces more conservative estimates where the network is more sensitive to uncertainty in the inputs. For example, we observe that adjusted speed predictions are more conservative for large-angle turns and in more slippery environments.

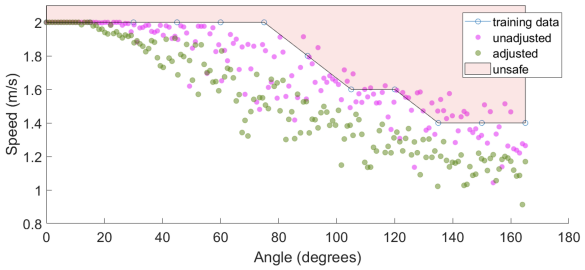



Fig. 8: Noisy speed predictions at $\mu = 0.05$ and $\delta_\tau = 4$ meters, adjusted for variance.

## VI. Case Study

The proposed framework was validated in simulation using Gazebo and ROS, on the complex environment depicted in Fig. 9 with different turning angles, walls, and ramps. The robot is tasked to traverse through such complex environment under different surface friction coefficients, $\mu$. Simulations were run with walls placed at a distance of 1 m surrounding the desired trajectory and with no walls to distinguish the behavior of the UGV when it is not constrained by a tight boundary (i.e., a large $\delta_\tau$). For the latter, $\delta_\tau$ = 3 m was provided to guarantee that the system doesn't deviate too much from the desired path. Path characteristics are highlighted in Fig. 9, where the

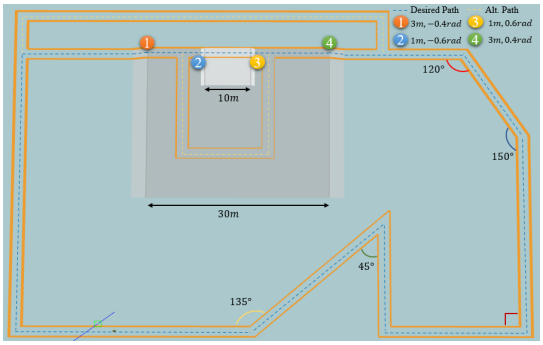
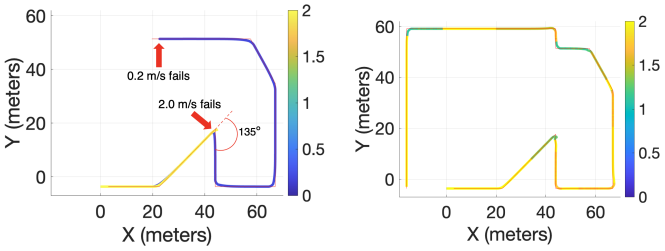



Fig. 9: Bird's eye view of Gazebo environment used for validation.

Jackal UGV must maneuver through turns of 30°, 45°, 60°, 90°, and 135°. The shortest possible segment the robot must navigate is 7.5 m. The desired trajectory features ramps of 3 m and 1 m, with angles of $\pm 0.4$ and $\pm 0.6$ radians (22.9°, 37.8°), respectively. The figures in the following section show the commanded velocity (m/s) using a colorbar.

### A. Simulation Results

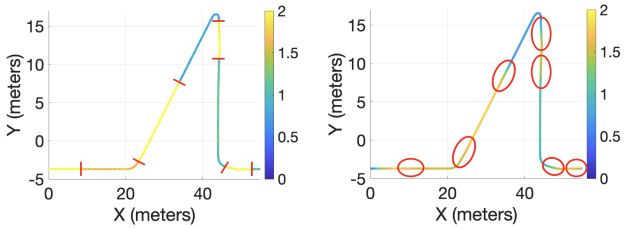
Our approach is first compared to two trials using constant speeds of 0.2 m/s and 2.0 m/s to demonstrate the need for adaptation. The UGV is often unsuccessful when travelling at the constant speeds due to obstacles such as ramps and sharp turns. In contrast, by using our NN-based speed adaptation and replanning, the UGV successfully reaches the goal for each $\mu$ value. Fig. 10 shows an example of UGV traversals using the constant and dynamic speeds. The color-gradient side bar on the figure indicates the commanded speed. In Fig. 10(a), the UGV cannot traverse the first ramp while going 0.2 m/s due to insufficient speed. Whereas at 2.0 m/s, the UGV is too fast to stay within $\delta_\tau$ while executing a 135° turn. In Fig. 10(b), the UGV leverages our framework and successfully reaches the final goal by adapting its speed and executing a replanning because it recognizes that it cannot successfully traverse the ramp at $\mu$ = 0.09. Due to higher slippage on $\mu$ = 0.09, the commanded speed here decreases 10 m before turns – the specified look-ahead distance for $\mu$ = 0.09 in Table 1 – to ensure the UGV is able to slow enough to traverse the turn while staying within $\delta_\tau$.




(a) Constant speed (b) Dynamic speed

Fig. 10: Constant vs. dynamic UGV speed on $\mu = 0.09$.

Fig. 11 shows the effect of look-ahead distance on speed. The commanded and actual speed of the UGV are compared prior to turn traversal on $\mu$ = 0.05. Fig. 11(b) shows the delay in reaching commanded speeds in Fig. 11(a) due to a low $\mu$.




(a) Commanded speed (b) Actual speed

Fig. 11: Commanded vs. actual speed on $\mu$ = 0.05.

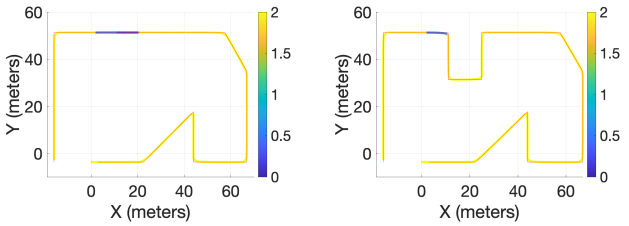
Fig. 12 shows another example of replanning. The robot is able to traverse a 1m long ramp at $\mu$ = 0.5 (Fig. 12(a)), but because a 2m long ramp is predicted "not safely traversable", it must take another path as shown in Fig. 12(b).

Finally, Fig. 13 shows the effect of deviation threshold. With a $\delta_\tau$ of 1 m, the NN predicts a safe speed of 1.2 m/s. With a $\delta_\tau$ of 3 m, the NN predicts a speed of 1.7 m/s, which would be an unsafe speed at $\delta_\tau$ = 1 m, as demonstrated in Fig. 13(b).

## VII. Conclusion and Future Work

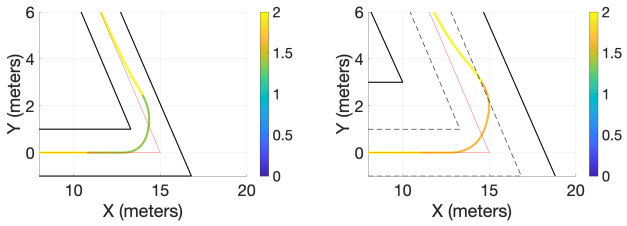
In this work we have presented a NN-based framework to predict the maximum safe speed to traverse terrains characterized by different friction, turns, ramps, and obstacles. A pure-pursuit algorithm is considered and an A*-based replanning procedure is deployed to guarantee that the system can navigate safely through the considered path. Uncertainties in the NN prediction are also taken into account during training to provide safe speeds. Our framework was validated with extensive simulations in Gazebo.

Future work will extend this framework to consider unknown environments and uncertainties in terrain detection.




(a) Ramp length = 1 m (b) Ramp length = 2 m

Fig. 12: Commanded speed on trials demonstrating path replanning on different ramp lengths on $\mu$ = 0.5.




(a) $\delta_\tau$ = 1 m (b) $\delta_\tau$ = 3 m

Fig. 13: Comparison of UGV speeds executing a 120° turn with $\mu$ = 0.09 and different $\delta_\tau$ values.

## Acknowledgments

This work is based on research sponsored by DARPA under Contract No. FA8750-18-C-0090.

## References

[1] H. U. Unlu, N. Patel, P. Krishnamurthy, and F. Khorrami, "Sliding-window temporal attention based deep learning system for robust sensor modality fusion for ugv navigation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4216–4223, 2019.

[2] R. Rosenfeld, M. Restrepo, W. Gerard, W. Bruce, A. Branch, G. Lewin, and N. Bezzo, "Unsupervised surface classification to enhance the control performance of a ugv," 04 2018, pp. 225–230.

[3] J. Guzzi, R. O. Chavez-Garcia, M. Nava, L. M. Gambardella, and A. Giusti, "Path planning with local motion estimations," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2586–2593, 2020.

[4] M. Ono, T. J. Fuchs, A. Steffy, M. Maimone, and J. Yen, "Risk-aware planetary rover operation: Autonomous terrain classification and path planning," in *2015 IEEE Aerospace Conference*, 2015, pp. 1–10.

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

[6] R. Michelmore, M. Kwiatkowska, and Y. Gal, "Evaluating uncertainty quantification in end-to-end autonomous driving control," 2018.

[7] S. Thrun, M. Montemerlo, and A. Aron, "Probabilistic terrain analysis for high-speed desert driving," in *Robotics: Science and Systems*, 2006.

[8] R. C. Coulter, 1992.

[9] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, p. 1098–1101, 1952.

[10] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[11] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.