The NATURE Autonomy Stack – An open-source stack for off-road navigation

Christopher Goodin[a], Marc N. Moore[a], Daniel W. Carruth[a], Christopher R. Hudson[a], Lucas D. Cagle[a], Stefan Wapnick[b], and Paramsothy Jayakumar[c]

[a]Center for Advanced Vehicular Systems, Mississippi State University, Starkville, MS
[b]McGill University, Montreal, Quebec, Canada
[c]US Army DEVCOM Ground Vehicle Systems Center, Warren, MI

ABSTRACT

Off-road autonomous navigation remains an ongoing challenge for autonomous ground vehicles (AGV). The challenges of navigating in an unstructured environment include identifying and detecting both positive and negative obstacles, distinguishing navigable from non-navigable vegetation, identifying soft soil, and negotiating rough or sloping terrain. While many recent works have dealt with various aspects of the off-road navigation problem, up to now there has not been a free and open-source autonomy stack for off-road that included integrated modules for perception, planning, and control. Therefore, we have recently developed the NATURE (Navigating All Terrains Using Robotic Exploration) autonomy stack as a publicly available resource to facilitate the advancement of off-road navigation research. The NATURE stack is implemented using the Robotic Operating System (ROS) and can be built to work with both ROS-1 and ROS-2. The modular nature of the NATURE stack makes it an ideal resource for researchers who want to evaluate a particular algorithm for perception, planning, or control without developing an entire navigation stack from scratch. NATURE features several options for both global and local path planning including A*, artificial potential field, and spline-based planning, as well as multiple options for perception including a simple geometrically based obstacle finder and more advanced custom traversability algorithm derived from 3D lidar. In this presentation we give an overview of the NATURE stack and show some past uses of the stack in both simulated and field experiments.

Keywords: Autonomy, Path-Planning, Perception

1. INTRODUCTION

Autonomous navigation in off-road environments presents unique challenges that have no counterpart in on-road autonomy systems.[1] Off-road autonomous ground vehicles (AGV) must operate without the benefit of infrastructure like roads, marked lanes, or traffic signs and signals. This leads to challenges in the three main software functions of autonomous software - perception, planning, and control. Perception challenges arise because of the inconsistent nature of off-road terrain,[2] planning challenges occur because of the wide range of factors affecting mobility in off-road terrain,[3] and control challenges occur due to interaction with terrain features like soft-soil[4] or vegetation.[5] Perhaps due to these challenges, prior to the work presented in this paper there was no publicly available open-source autonomous navigation stack available for passenger-sized off-road vehicles.

Over the past several years, we have developed an open source autonomy stack for passenger-sized off-road vehicles, the NATURE (Navigating All Terrains Using Robotic Exploration) stack, in order to meet the ongoing need for a software stack that can be used by the research community in the development and testing of perception, planning, and control algorithms for AGV. We have demonstrated the viability of the stack in simulated[6,7] and physical[8] testing. In this work, we present the first in-depth overview of the technology and software in the NATURE stack, and present several recent applications that have used the stack to support ongoing AGV research, development, and testing.

Further author information: (Send correspondence to C.G.)
C.G.: E-mail: cgoodin@cavs.msstate.edu

2. BACKGROUND AND MOTIVATION

The most prominent and popular open source software for autonomous navigation is the Robotic Operating System (ROS).[9] ROS has extensive software modules, known as packages, and an associated software and messaging environment, that allow the fast integration of robotic components. One fundamental component of ROS is the navigation stack,[10] a set of interoperable functions for perception, planning, and control of AGV. While the ROS navigation stack has been used extensively, it is optimized for flat terrain and has an underlying assumption of 2D terrain. Subsequently, it has been used primarily for skid-steered, indoor robots.[11]

Because ROS is open and extensible, many third-party developers have published open-source code that can be integrated into the ROS framework as a package. Most developers focus on a single capability or new algorithm, but there are a few integrated "full-stack" software packages availabe through third parties for ROS. Of these, Autoware and Apollo are the most popular and widely used.[12] Autoware is an open source, full stack autonomy software kit that has been available for nearly a decade[13] and Apollo is a similar but more recent competitor. However, both Autoware and Apollo are optimized for on-road autonomous driving and driver assist functions.[12]

More recently, ROS2 updates to the navigation stack (NAV2)[14] have improvements that make it more suited for use in off-road terrain with larger passenger sized vehicles, such as including planners like the Dynamic Window Approach[15] that can be adapted for Ackermann-steered vehicles. However, this does not meet the need of the many existing research platforms that rely on ROS-1 due to hardware or software constraints. Because of the limitations of the existing tools discussed above, we developed the NATURE stack to be an open-source, extensible toolkit for off-road autonomous navigation that is compatible with both ROS-1 and ROS-2. To our knowledge, NATURE is the only software stack meeting this criteria.

3. NATURE STACK

The NATURE stack is a ROS package, available on Github. One unique feature of the NATURE stack is the capability to build the stack in either ROS-1 or ROS-2. This is enabled by encapsulating all ROS or ROS-2 software functions in NATURE-specific naming conventions. The encapsulating functions refer to the appropriate ROS-1 or ROS-2 features by using a C preprocessor directive at compile time. The compile-time flags in the CMake file included with the repository, and the package repository contains CMake files and package XML descriptions for both ROS and ROS-2.

The package includes all software and data necessary to control an autonomous vehicle, including modules for perception, vehicle control, global planning, and local planning. These follow closely with the basic functions listed in a recent study of testing methods for unmanned vehicles, which list "environment perception", "motion execution", "mission planning", and "motion planning"[16] as key functions.

Following generally accepted conventions for off-road navigation,[17] in the NATURE stack global planning involves finding a path to the goal over a spatially large scale while local planning involves obstacle avoidance and navigation toward intermediate points provided by the global planner. In terms of the time horizon, the global planner may forecast on the scale of minutes while the local planner forecasts on the scale of seconds or fractions of a second.

In the following subsections, the software modules for each of the main functions in NATURE are described in more detail. In some cases, there are multiple options provided for a function. In this case, the relative strengths and weaknesses of these modules are discussed.

We note that most of the individual modules described below are implementations of algorithms that have been used in robotics and autonomy for decades. This approach was chosen for the NATURE stack in order to focus on integration, reliability, and ease of installation for the software stack. Although there are number of other cutting edge algorithms available for perception, planning, and control, the ones presented here were chosen based on several requirements including

- Software / algorithm availability and documentation
- Software implementation does not require 3rd party libraries or packages to be installed

- Fast computational performance that does not require GPU or other special computing
- Does not require specialized sensors beyond those commonly used in autonomous navigation such as cameras, lidar, and GPS

Following these guidelines, the algorithms and modules presented below serve as a starting point for any researchers who wish to integrate their own more advanced methods into the software stack.

3.1 Perception

In autonomous navigation perception is the process or set of processes that converts raw sensor data (which may be digital or analog) into actionable information for use by the other autonomous functions like global and local path planning and vehicle control. In abstract, the perception function must accept data from one or more sensors as input and output some format of world model, which may be a map, a list of features, or some other world model. In the NATURE stack implementation of this concept, the input sensor data is a ROS PointCloud2 message, and the output is an occupancy grid.

Point clouds are a convenient format for sensor data representation because they can be created by a variety of sensor types including lidar, stereo cameras, depth cameras, and automotive radar. Therefore, in keeping with the principles outlined above, the choice of the point cloud format for input into the perception module does not require a specific type or model of sensor. In the following discussion, the registration of the point cloud to the map coordinates is assumed to have already taken place prior to input into the perception module.

For most camera, lidar, and radar sensors that produce a point cloud, the points are initially in the sensor coordinate frame, with the 3D coordinates of each point being relative to the location and orientation of the sensor. In order to use successive point cloud messages to generate a continuous map, the points must be registered to the map coordinate system, which may or may not be stationary. At any rate, the point cloud registration process requires a high degree of accuracy in the position and orientation estimate (localization) of the vehicle. This localization functionality is not included with the NATURE stack because it is highly dependent on the type vehicle and existing sensors in use.

Similarly, occupancy grids have been used as a representation of the world in autonomous world-modelling and path-planning for many decades.[18] Although the occupancy grid is a 2D structure, in the NATURE stack the full 3D information of the point cloud is used to generate the prediction of the occupancy of a given cell, as will be discussed in the next subsections on the two perception modules available in the stack.

3.1.1 Slope-based Segmentation

The slope-based obstacle detector follows the method outlined in[19] to measure the local terrain slope directly from a registered 3D point cloud. Each time a new point cloud message is received, the points are checked one by one to see which cell it occupies in the existing occupancy grid. The highest, 2^{nd} highest, and lowest point measured in each grid cell are maintained in software memory, and new points in the cell are checked against the currently tracked points, replacing existing points if they are higher or lower. The "slope" of each cell is calculated as the vertical distance between the lowest and 2^{nd} highest point in each cell, divided by the length of the cell side. The cell size is defined in the launch parameters of the perception node. All cells that are above a user defined slope threshold are classified as obstacles in the occupancy grid.

The advantages of this approach are the simplicity, computational efficiency, and versatility. The primary disadvantage of this approach is that while it is effective at detecting obstacles, it makes not distinction between obstacles like a single stem of tall grass or a brick wall of equal height - the only discriminator is height. Therefore, this method tends to overestimate the severity of many obstacles in off-road driving.

3.1.2 FTTE

In contrast to the simplistic approach of the slope algorithm, the Fast Terrain Traversability Estimate (FTTE)[3] perception algorithm uses the lidar to estimate three mobility factors - vegetation density, surface slope, and surface roughness. Surface slope and roughness are calculated by fitting planes to each local region. The severity of each obstacle is estimated by dividing the world map into 3D voxel cubes and calculating the *porosity* of each voxel, which is the fraction of scans that pass through the cell versus those that return. Solid obstacles like walls or tree trunks have a low porosity, while extended objects like bushes and grass have high porosity. Porosity is therefore used as a proxy for obstacle severity / resistance by the FTTE method. More details on the FTTE model are available in our previous publication on the FTTE method.[3]

3.2 Global Planner

The global planning module takes the output of the perception module (the occupancy grid), along with user defined mission waypoints that are input in the launch file. The A* algorithm[20] has been used as a global path planner for AGV for decades and has been demonstrated to give fast, optimal solutions for large-scale planning problems.[21] Therefore, in keeping with the goals outlined above, A* is chosen as the default global path planner for NATURE.

The global planner in NATURE also publishes the current vehicle mode, which can be either *startup*, *active*, or *shutdown*. There are three options for shutdown behavior which include 1) bring the vehicle to a smooth stop and return to idle, 2) bring the vehicle to a smooth stop and shut down the autonomy system, or 3), bring the vehicle to an immediate stop and shut down. The desired behavior at shutdown is set by the user, and the global planner automatically initiates the desired shutdown behavior when the last global waypoint is reached.

The NATURE global planner also subscribes to a waypoints topic so that the global mission waypoints can be updated in real-time by an external mission-level controller, if desired. The mission level controller is not provided by the NATURE stack, but this feature allows NATURE to be integrated into a larger framework. If the global planner receives revised waypoints, it will update the global plan to follow the new waypoints.

3.3 Local Planner

There are two built-in options for the local planner in the NATURE stack; a cubic-spline planner and a potential field planner.

3.3.1 Spline Planner

The cubic spline planner is based on recent work on safe trajectory planning for AGV.[22] The planner generates a set of splines in the coordinate frame of the desired global plan and calculates the relative cost of each spline depending on several factors including deviation from the desired path (output from the global planner), the change from the current trajectory, the rollover risk, and the passenger comfort (based on lateral jerk). The relative importance of these factors can be weighted with user defined inputs. The advantages of the spline planner are that it fast and yields smooth, driveable local plans in most cases. The disadvantage is that it has an underlying assumption that the global path can also be represented as a cubic spline. If this assumption is violated then the cubic spline planner may give very high curvature local plans.

3.3.2 Potential Field Planner

An alternative to the cubic spline planner is the artificial potential field (APF) method. A recent review of motion planning algorithms[23] listed the APF method[24] as one of the state-of-the-art methods at that time. The APF planner takes the obstacle map that is created by the perception module and calculates an artificial potential field based on the distance and direction to the goal point and the repulsive potential of the obstacles. The APF planner tends to create smooth paths but does not explicitly consider vehicle kinematics when creating the desired local plan. The advantage of the APF planner is that it works for a variety of terrain types, and when used in conjunction with a global planner like A* that can avoid horseshoe shaped obstacles that are known to be unsolvable by the APF planner, can give smooth results for the desired local path.

3.4 Vehicle Controller

Once the local path has been generated, the vehicle controller node in NATURE calculates the desired throttle, braking and steering commands to follow the local path and maintain the desired speed. Steering control is implemented using the pure pursuit controller (PPC),[25] which has been adopted for Ackermann steered vehicles for several decades. In fact, a recent review[26] identified pure pursuit as one of the most popular algorithms for steering control, along with the Stanley algorithm,[27] and PID control.[28]

The PPC works by comparing the vehicles current location and heading to the location of a point along the desired path at some distance ahead of the current location, called the lookahead distance. The steering is adjusted to meet a radius of curvature required to intersect the lookahead point. The lookahead distance is an adjustable parameter in the PPC. Smaller lookaheads will follow the path more closely but also give more jerk in the steering, while longer lookahead values will give smoother steering but may round off edges on sharp corners in the local path.

The steering controller also supports skid-steered vehicles such as the Clearpath Warthog using the skid-steering control algorithm proposed by Kanayama *et al.*,[29] which uses the offset of the current heading from the desired heading with a proportional controller.

Throttle control in the NATURE vehicle control node is achieved using a PID controller[30] with an optional feed-forward term. The PID parameters can be set in the launch file, and optionally a polynomial equation that represents throttle setting versus speed can be input as a feed-forward term in the PID control if the vehicle has been adequately characterized. The desired speed is a user-defined constant value in the NATURE stack.

4. USE CASES

The NATURE stack has been deployed on a number of different vehicles and has been used for a variety of applications. The simplicity, flexibility, and versatility of the stack allow it to be easily integrated into existing environments or extended with new capabilities. Table 1 lists the different vehicles on which the NATURE has been deployed, along with some of the relevant hardware details which will be discussed in more detail in the following sections.

Table 1: Autonomous vehicles using the NATURE stack and their hardware

Vehicle	**Primary Sensor**	**Localization**	**Computing**
MRZRD4	Ouster OS1 Lidar	Corrected GPS + IMU	NVIDIA AGX Pegasus
Warthog	Ouster OS1 Lidar	Wheel Odometry	NVIDIA AGX Pegasus
RC Car	L515 Camera	Wheel Odometry	Jetson AGX Orin

4.1 Deployment in Simulation

The NATURE stack has been used to support simulated experiments with AGV for a number of different projects. Many of these make use of the MSU Autonomous Vehicle Simulator (MAVS), an open-source simulation library for AGV that includes simulations of the vehicle, terrain, sensors, and environment.[31] MAVS features interfaces for ROS and ROS2 that allow it to quickly and easily be connected with the NATURE stack.

The MAVS-NATURE combination was used to study the influence of environmental conditions like rain and dust on autonomous navigation.[6] More recently, it was used to study autonomous navigation by small vehicles over rough terrain.[32] Furthermore, the NATURE stack was adapted and extended by a NATO working group to study the requirements for autonomous vehicle simulators.[33]

The MAVS-NATURE combination is also used to develop and test new capability, as shown in Figure 1, which shows the NATURE stack being visualized in RVIZ and tested with a new module for a rapidly-exploring random tree (RRT) planner.[34] By using the MAVS-NATURE combination, it is possible to develop and test ROS nodes with new features or algorithms, allowing users to easily expand the default NATURE capabilities.

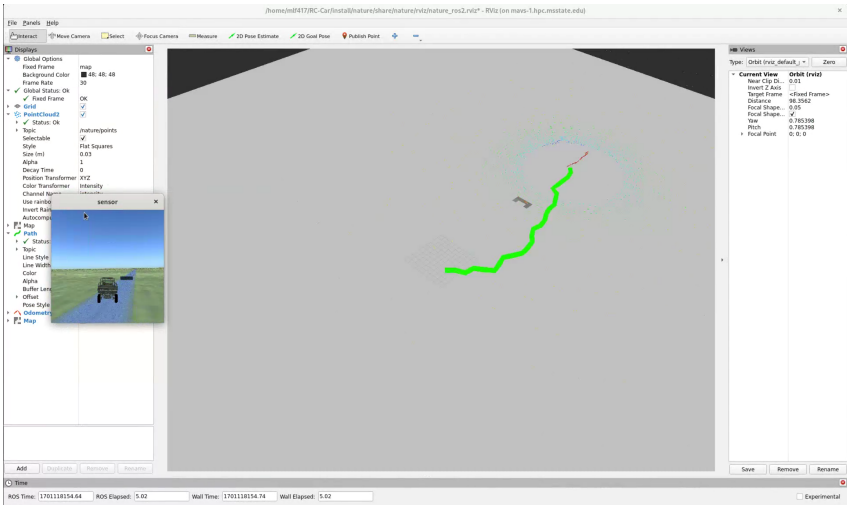


Figure 1: Using NATURE-MAVS to develop new capability in simulation

The expansion of the NATURE stack to include the RRT planner was accomplished within the scope of an undergraduate robotics class over a single semester. This project is noteworthy because the students involved had no prior experience with the NATURE stack or ROS framework. Despite this, they were able to not only understand the architecture of the stack, but also effectively integrate the RRT planner into it. Their success underscores the NATURE stacks' user-friendly design and its capability for straightforward integration of new functionalities, making it an ideal stack for educational purposes.

4.2 Deployment on MRZR D4 and Clearpath Warthog

The NATURE stack has been deployed on several vehicles for the purpose of development, testing, and study of AGV processes. Figure 2 shows an example of recent experiments performed with the MRZR to evaluate the capabilities of the NATURE stack and compare the results to those measured in simulation.

Figure 2: Experiments performed using the NATURE stack deployed on the MRZR-D4 (left) and the Clearpath Warthog (right)

The NATURE stack was deployed on an MRZRD4 and tested in a variety of conditions including dusty gravel roads and extreme soft soil conditions (Figure 2 (left)). Our findings in these tests are summarized in a recent publication,[8] where it was found that soft soil in particular had a drastic impact on the steering and speed control of the autonomous MRZR.

The NATURE stack was also implemented on the Clearpath Warthog vehicle. The vehicle comes equipped with ROS and drive-by-wire capability, and we added the sensing and compute package to the vehicle, as shown

in Figure 2 (right). With the NATURE stack implemented on these two vehicles, we evaluated a novel method for tuning the PID controller of the NATURE stack by using an empirical model of the vehicle longitudinal speed.[35] This allowed us to tune the PID controllers for both vehicles and demonstrate these in field experiments.

4.3 Deployment on Remote Control Car

Because the NATURE stack is lightweight and can use any sensor that can generate point clouds, it has also been implemented on a $1/10^{th}$ scale remote control (RC) car, as shown in Figure 3 (left). The autonomous RC car is used by students and researchers at our small-scale autonomy test track (Figure 3 (right)), which is used to test a variety of different autonomy algorithms and advanced maneuvers without the cost and risk associated with testing larger vehicles.

Figure 3: Using the NATURE stack on $1/10^{th}$ scale car

Due to the small size and inherent limitations in the power supply of the RC car, a depth camera was leveraged for point cloud generation. This strategic adaptation caused by a physical limitation in the vehicle showcases the NATURE stack's ability to seamlessly incorporate a broad array of sensor technologies while remaining effective. Output from the depth camera is shown in Figure 4, which demonstrates not only the detailed environment mapping achievable with a low power, compact sensor but also underscores the potential of the NATURE stack to facilitate autonomy at multiple scales.

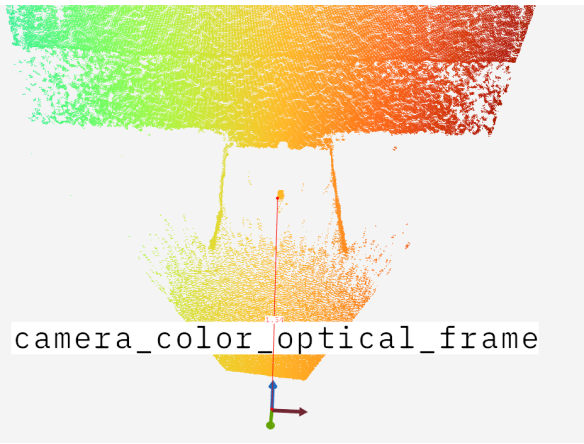
Figure 4 shows the result of testing the detection range of the L515 depth camera on a $1/10^{th}$ scaled version based on a prior study using the NATURE stack and the MRZRD4 in Goodin *et al.*[7] This test showed that NATURE was able to leverage the depth camera to detect the obstacle in a lane bounded by two walls at 15 feet (the original study conducted with a MRZRD4 using a Ouster v1 LiDAR had a detection range of 150ft).

5. SUMMARY AND CONCLUSION

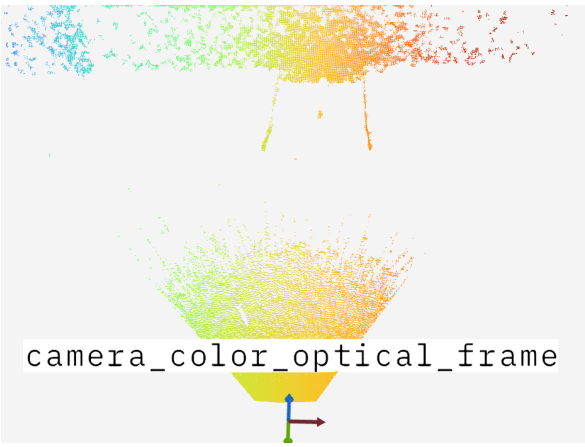
The NATURE stack is an open-source autonomy stack for off-road navigation that is implemented for both ROS and ROS2. Designed to be lightweight, reliable, and versatile, the NATURE stack has been implemented on a variety of different vehicle platforms and simulations using different sensors and computing hardware. Several applications of the NATURE stack were discussed in this work, including developing and testing new autonomous navigation algorithms, comparing real and simulated testing, and studying autonomy safely using $1/10^{th}$ scale vehicles. Future work on the NATURE stack will the development and incorporation of additional modules for path planning, perception, and control.

ACKNOWLEDGMENTS

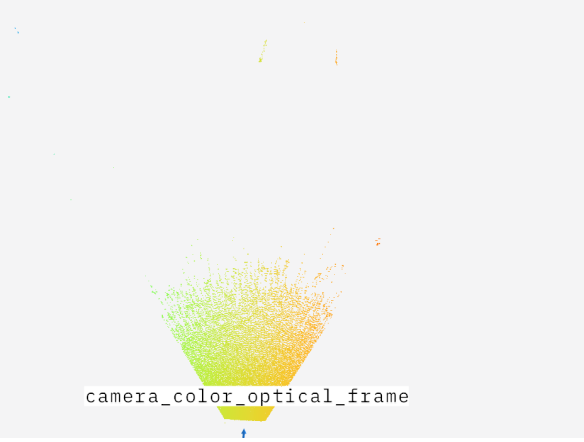
Portions of the NATURE stack were developed with the financial support of the Automotive Research Center (ARC) in accordance with Cooperative Agreement W56HZV-14-2-0001 U.S. Army Tank Automotive Research, Development and Engineering Center (TARDEC) Warren, MI.



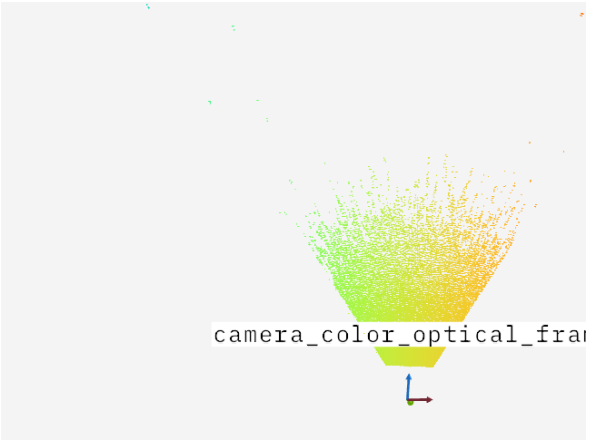
(a) 5ft L515 Test



(b) 10ft L515 Test



(c) 15ft L515 Test



(d) 20ft L515 Test

Figure 4: Example point clouds generated by the depth camera on the RC-car implementation of the NATURE stack. The obstacle is detectable to 15 feet distance.

REFERENCES

[1] Kelly, A., Stentz, A., Amidi, O., Bode, M., Bradley, D., Diaz-Calderon, A., Happold, M., Herman, H., Mandelbaum, R., Pilarski, T., et al., "Toward reliable off road autonomous vehicles operating in challenging environments," *The International Journal of Robotics Research* **25**(5-6), 449–483 (2006).

[2] Sharma, S., Ball, J. E., Tang, B., Carruth, D. W., Doude, M., and Islam, M. A., "Semantic segmentation with transfer learning for off-road autonomous driving," *Sensors* **19**(11), 2577 (2019).

[3] Goodin, C., Dabbiru, L., Hudson, C., Mason, G., Carruth, D., and Doude, M., "Fast terrain traversability estimation with terrestrial lidar in off-road autonomous navigation," in [*Unmanned Systems Technology XXIII*], **11758**, 117580O, International Society for Optics and Photonics (2021).

[4] Priddy, J. D., Berney IV, E. S., and Peters, J. F., "Effect of near-surface hydrology on soil strength and mobility," *Geological Society, London, Special Publications* **362**(1), 301–320 (2012).

[5] Rybansky, M., "Determination the ability of military vehicles to override vegetation," *Journal of Terramechanics* **91**, 129–138 (2020).

[6] Carruth, D., Goodin, C., Dabbiru, L., Scherer, N., and Jayaku-mar, P., "Predicting error propagation in autonomous ground vehicle subsystems," in [*Proc. Ground Veh. Syst. Eng. and Technol. Symp*], 11–13 (2020).

[7] Goodin, C., Carruth, D. W., Dabbiru, L., Hudson, C. H., Cagle, L. D., Scherrer, N., Moore, M. N., and Jayakumar, P., "Simulation-based testing of autonomous ground vehicles," in [*Autonomous Systems: Sensors, Processing and Security for Ground, Air, Sea and Space Vehicles and Infrastructure 2022*], **12115**, 167–174, SPIE (2022).

[8] Carruth, D. W., Goodin, C., Dabbiru, L., Scherrer, N., Moore, M. N., Hudson, C. H., Cagle, L. D., and Jayakumar, P., "Comparing real and simulated performance for an off-road autonomous ground vehicle in obstacle avoidance," *Journal of Field Robotics* **41** (2024).

[9] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y., "Ros: an open-source robot operating system," in [*ICRA workshop on open source software*], **3**, 5, Kobe, Japan (2009).

[10] Guimarães, R. L., de Oliveira, A. S., Fabro, J. A., Becker, T., and Brenner, V. A., "Ros navigation: Concepts and tutorial," *Robot Operating System (ROS) The Complete Reference (Volume 1)* **1**, 121–160 (2016).

[11] Kangutkar, R., Lauzon, J., Synesael, A., Jenis, N., Simha, K., and Ptucha, R., "Ros navigation stack for smart indoor agents," in [*2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*], 1–10, IEEE (2017).

[12] Raju, V. M., Gupta, V., and Lomate, S., "Performance of open autonomous vehicle platforms: Autoware and apollo," in [*2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*], 1–5, IEEE (2019).

[13] Kato, S., Takeuchi, E., Ishiguro, Y., Ninomiya, Y., Takeda, K., and Hamada, T., "An open approach to autonomous vehicles," *IEEE Micro* **35**(6), 60–68 (2015).

[14] Zheng, K., "Ros navigation tuning guide," *Robot Operating System (ROS) The Complete Reference (Volume 6)* **1**, 197–226 (2021).

[15] Fox, D., Burgard, W., and Thrun, S., "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine* **4**(1), 23–33 (1997).

[16] Huang, W., Wen, D., Geng, J., and Zheng, N.-N., "Task-specific performance evaluation of ugvs: Case studies at the ivfc," *IEEE transactions on intelligent transportation systems* **15**(5), 1969–1979 (2014).

[17] Otte, M. W., Richardson, S. G., Mulligan, J., and Grudic, G., "Path planning in image space for autonomous robot navigation in unstructured environments," *Journal of Field Robotics* **26**(2), 212–240 (2009).

[18] Elfes, A., "Using occupancy grids for mobile robot perception and navigation," *Computer* **22**(6), 46–57 (1989).

[19] Goodin, C., Carrillo, J., Monroe, J. G., Carruth, D. W., and Hudson, C. R., "An analytic model for negative obstacle detection with lidar and numerical validation using physics-based simulation," *Sensors* **21**(9), 3211 (2021).

[20] Hart, P. E., Nilsson, N. J., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics* **4**(2), 100–107 (1968).

[21] Sanchez-Ibanez, J. R., Perez-del Pulgar, C. J., and García-Cerezo, A., "Path planning for autonomous mobile robots: A review," *Sensors* **21**(23), 7898 (2021).

[22] Hu, X., Chen, L., Tang, B., Cao, D., and He, H., "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mechanical Systems and Signal Processing* **100**, 482–500 (2018).

[23] Oroko, J. A. and Nyakoe, G., "Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: a review," *Proceedings of Sustainable Research and Innovation Conference* **1** (2012).

[24] Barraquand, J., Langlois, B., and Latombe, J.-C., "Numerical potential field techniques for robot path planning," *IEEE transactions on systems, man, and cybernetics* **22**(2), 224–241 (1992).

[25] Coulter, R. C., "Implementation of the pure pursuit path tracking algorithm," tech. rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST (1992).

[26] Yao, Q., Tian, Y., Wang, Q., and Wang, S., "Control strategies on path tracking for autonomous vehicle: State of the art and future challenges," *IEEE Access* **8**, 161211–161222 (2020).

[27] Amer, N. H., Hudha, K., Zamzuri, H., Aparow, V. R., Abidin, A. F. Z., Abd Kadir, Z., and Murrad, M., "Adaptive modified stanley controller with fuzzy supervisory system for trajectory tracking of an autonomous armoured vehicle," *Robotics and Autonomous Systems* **105**, 94–111 (2018).

[28] Al-Mayyahi, A., Wang, W., and Birch, P., "Path tracking of autonomous ground vehicle based on fractional order pid controller optimized by pso," in [*2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*], 109–114, IEEE (2015).

[29] Kanayama, Y., Kimura, Y., Miyazaki, F., and Noguchi, T., "A stable tracking control method for a non-holonomic mobile robot.," in [*IROS*], 1236–1241 (1991).

[30] Islam, F., Nabi, M., Farhad, M. M., Peranich, P., Ball, J. E., and Goodin, C., "Evaluating performance of extended kalman filter based adaptive cruise control using pid controller," in [*Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2021*], **11748**, 1174807, International Society for Optics and Photonics (2021).

[31] Hudson, C., Goodin, C., Miller, Z., Wheeler, W., and Carruth, D., "Mississippi state university autonomous vehicle simulation library," in [*Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium*], 11–13 (2020).

[32] Johnson, W. P., [*Assessment of Simulated and Real-World Autonomy Performance with Small-Scale Unmanned Ground Vehicles*], Mississippi State University (2022).

[33] Paramsothy Jayakumar, "Off-road mobility assessment methods and tools for autonomous military ground vehicles." URL: https://apps.dtic.mil/sti/trecms/pdf/AD1201490.pdf (5 2023).

[34] LaValle, S. M., Kuffner, J. J., Donald, B., et al., "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and computational robotics: new directions* **5**, 293–308 (2001).

[35] Goodin, C., Moore, M. N., Carruth, D. W., Hudson, C. R., Cagle, L. D., and Jayakumar, P., "An empirical vehicle speed model for tuning throttle controller parameters," *International Journal of Vehicle Performance* **10**, 196–214 (2024).