IEEE Transactions on Field Robotics (T-FR) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-FR paper.
IEEE Transactions on Field Robotics
Special Issue on Space Robotics

Received 3 February 2025; revised 3 September 2025; accepted 7 October 2025.
Date of publication 24 November 2025; date of current version 15 January 2026.
This article was recommended by Associate Editor Rudolph Triebel.

Digital Object Identifier 10.1109/TFR.2025.3636366

Enhanced Autonomous Navigation on the Perseverance Mars Rover

OLIVIER TOUPET[1,2], MASAHIRO ONO[1], TYLER DEL SESTO[1], MARK MAIMONE[1], AND MICHAEL MCHENRY[1]

[1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA
[2]Zoox, Foster City, CA 94404 USA

CORRESPONDING AUTHOR: MASAHIRO ONO (ono@jpl.nasa.gov)

This work was supported in part by the Jet Propulsion Laboratory, California Institute of Technology, through the National Aeronautics and Space Administration under Grant 80nM0018D0004; and in part by the Mars 2020 Program.

This article has supplementary downloadable material available at https://doi.org/10.1109/TFR.2025.3636366, provided by the authors.

(Regular Article)

ABSTRACT This article presents Enhanced Autonomous Navigation (ENav), the autonomous driving algorithm of NASA's Mars rover *Perseverance*. A unique challenge for the autonomous driving of *Perseverance* is to meet strict safety and performance requirements in a highly uncertain environment with only a single-core CPU with extremely limited computing resources. ENav overcame this challenge with a novel two-stage path-selection approach that balances the path optimality and computational efficiency, combined with a unique collision checking algorithm that conservatively approximates computationally expensive kinematic settling. In addition, ENav provides robustness against slip by expanding the bounding boxes for wheels used by the collision check. These new features, together with field programmable gate array (FPGA)-accelerated vision processing, enabled *Perseverance* to autonomously drive on substantially more rock-dense terrains and increased the average daily driving distance by an order of magnitude compared to its predecessors, the *Curiosity*, *Spirit*, and *Opportunity* rovers. *Perseverance* has set several new records for autonomous driving on Mars, breaking those previously held by the *Opportunity* rover. As of the 1312th Martian day since landing, or October 28, 2024 on the Earth calendar, ∼90% of the 32.1 km of driving has used ENav to evaluate the terrain. This article provides detailed documentation of the ENav algorithm, as well as its implementation, testing, deployment, and driving results on Mars.

INDEX TERMS Autonomous navigation (AutoNav), Mars 2020, Mars rover, NASA, path planning, Perseverance, planetary rovers.

I. INTRODUCTION

A. BACKGROUND: AUTONOMOUS DRIVING ON MARS

PERSEVERANCE (Fig. 1) landed on Mars on February 18, 2021. One of the objectives of the Mars 2020 Mission[1] is to explore areas that could have supported life and to look for deposits that might preserve signs of ancient microbial life [8]. The selected landing site, Jezero Crater, was a lake some 3.5–4 billion years ago, with significant portions of an ancient river delta still evident (Fig. 2). The specific areas of scientific interest, together with constraints

[1]Mars 2020 is the name of the NASA mission while Perseverance is the name of the vehicle sent to Mars by the Mars 2020 Mission.

FIGURE 1. "Selfie" of the Perseverance rover with a deployed sample tube, taken on sol 684 (January 22, 2023).

FIGURE 2. Map of Perseverance's 32.077-km traverse as of sol 1312 (October 28, 2024). Bright white dots indicate where the rover stopped at the end of each drive sol (Martian day). The green dot shows the Ingenuity helicopter's final parking spot, and the light blue dot shows Perseverance's sol 1312 location after having successfully used ENav to climb 16°–23° sloped terrain at Serpentine Rapids and Summerland Trail. Image credit: NASA/JPL-Caltech/University of Arizona/USGS.

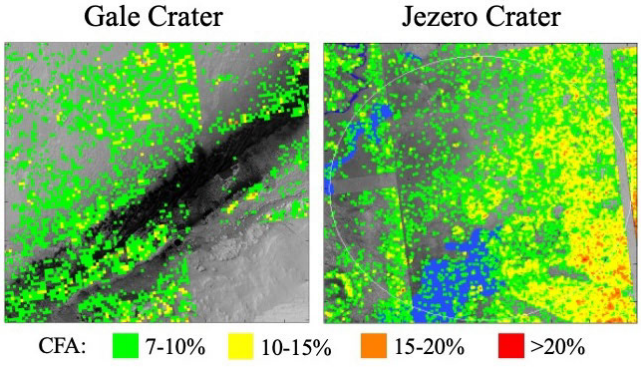


FIGURE 3. Rock density, measured in the CFA covered by rocks, in the landing sites of Mars rovers Curiosity and Perseverance, which landed in 2012 and 2021, respectively. The newer rover is tasked to drive on substantially more difficult terrain, which led to increased requirements for autonomous driving.

on the landing system, necessitated a mobility system capable of reliably driving at greater traverse rates than past Mars missions. Furthermore, compared to Gale Crater, the landing site of Mars Science Laboratory's (MSL's) Curiosity rover, Jezero Crater has substantially higher rock density (Fig. 3) and richer topography, which poses substantial challenges for driving. These factors led to the development of Enhanced Autonomous Navigation (ENav), which is the subject of this article.

Mars rovers need to be highly autonomous due to the inherent limits of communication with Mars. Typically, ground operators upload a new plan of activities once per Martian solar day, or *sol* for short, during the Martian morning. The rover executes these activities without any opportunity for real-time human monitoring or intervention. Then, after completing the execution of the sol's activities, the rover transmits critical scientific and engineering data, including imagery, to Earth via satellites orbiting Mars. This telemetry informs the subsequent plans developed by ground operators for the following sols.

Roughly speaking, Perseverance supports two primary modes of driving: "directed" and "AutoNav." In directed driving, human Rover Planners use 3-D terrain meshes derived from rover-collected imagery to assess the nearby terrain for hazards and manually select a drive path. The reliance on downlinked imagery effectively constrains Directed drives to unoccluded terrain within a distance of approximately 15–50 m, depending on terrain characteristics. Autonomous navigation (AutoNav), on the other hand, relies on onboard rover software to perform a repeating cycle of image acquisition, terrain analysis, hazard detection, and path selection, which navigates the rover autonomously to the next human-specified waypoint. Therefore, AutoNav is not restricted to nearby terrain observable from the rover's initial position, enabling it to drive far beyond the ∼15–50-meter line-of-sight distance in each driving cycle. This sol-to-sol planning, called tactical planning, is guided by long-range plans called strategic routes, which are typically provided months or years in advance. The strategic routes are planned by human strategic route planners who are also trained

in orbital image analysis and have substantial experience comparing ground-based and orbital imagery, working in concert with science team members who provide areas of interest [26].

Previous Mars rovers depended primarily on human-directed driving, typically only using AutoNav as a drive extender because it ran 4–10 times more slowly than directed driving. Specifically, during the Mars Exploration Rover (MER) mission, which landed the twin Spirit and Opportunity rovers in 2004, and the MSL mission, which landed the Curiosity rover in 2012, the typical strategy was to perform directed driving as far as visibility in the downlinked imagery allowed, and then transition to AutoNav driving to gain additional drive distance. The slower speed of AutoNav on those missions stems in part from the limited computational power of the rovers' radiation-tolerant computers (20-MHz RAD6000 for the MER rovers, and a 133-MHz RAD750 for MSL), and also from MSL's need to repeatedly pan its mast to collect imagery over a sufficiently wide field of regard. In addition, those rovers stopped to acquire images, and their AutoNav software performed lengthy computations based on those images before choosing the next path to drive. This AutoNav drive speed penalty and the resulting strategy of using AutoNav only to extend directed drives were major factors in the lower usage of AutoNav on earlier missions, as summarized in Table 1. As a result, Curiosity drove 12.6 m with AutoNav per sol on average [36].

The Mars 2020 mission identified early on that it would require substantial enhancements to AutoNav driving, both in terms of speed and also the ability to reliably navigate terrain with a greater density of mobility hazards. The baseline reference scenario (BRS), an abstract but representative mission scenario defined before the selection of a specific landing site, [16], [22], required the rover to drive 12 km in 85 sols, or approximately 150 m per sol, which substantially exceeds the realistic capability for a strategy that primarily depends on directed drives. In addition, many of the candidate landing sites, including Jezero Crater, also required the ability to navigate through areas with a higher obstacle density than past missions, as shown in Fig. 3, which the AutoNav algorithm used by MER and MSL could not support. Therefore, the mission concluded that a substantial enhancement to AutoNav would be necessary and funded the development of the new ENav algorithm.

ENav has been successfully driving the Perseverance rover since its landing, with its performance far exceeding its predecessors. As of October 28, 2024, the rover has accumulated 32.1 km of odometry, of which over 90% has been driven using AutoNav (Table 1). The rover also set new driving distance records, as we will detail in Section V-C. On Perseverance, for the first time, AutoNav is now the primary driving mode. Table 1 compares the AutoNav usage of all NASA Mars rovers to date, highlighting the benefits of ENav to Perseverance. Fig. 4 includes images showing the wheel tracks of Perseverance while driven by ENav, on drives during which it autonomously avoided obstacles.

B. COMPARISON WITH EXISTING PLANETARY ROVER AUTONOMOUS NAVIGATION

The first successful rovers on other planetary bodies were NASA's astronaut-driven Lunar Roving Vehicles (LRVs) during the Apollo missions and the teleoperated Lunokhod 1 and 2 rovers operated by the Soviet Union. The distance records for these were 39 km driven by Lunokhod 2 in 1973 and 35.74 km driven by the Apollo 17 LRV. However, neither of those vehicles provided an autonomous driving mode. The Soviet Union attempted to land small PrOP-M rovers on Mars in 1971, but was unable to operate them successfully.

The first successful Mars rover, NASA's Sojourner, which was carried to Mars as a payload of the Mars Pathfinder mission, had a simple and purely reactive autonomous driving capability called the "go to waypoint" mode. In this mode, the small six-wheeled rover went straight to the next waypoint while performing proximity hazard detection with its forward cameras, contact sensors on the bumpers, and five laser light stripes, each of which acquired four terrain point samples. If an obstacle was detected, the rover would rotate in place until the hazard was no longer detectable, move forward one-half vehicle length, and then resume its path to the waypoint with no memory of the hazard it had just avoided [20].

The first full-fledged autonomous driving was achieved by NASA's Spirit and Opportunity rovers, which landed on Mars in 2004. Their AutoNav system consisted of 2.5-D elevation mapping via stereo vision initially on 128 × 128 pixel image pairs (quickly increased to 256 × 256 pixels), visual odometry (VO)-based pose estimation, hazard avoidance using the Grid-Based Estimation of Surface Traversability Applied to Local Terrain (GESTALT) local path planner [3], [10], and later included a global path planner based on the Field D* algorithm [5].

GESTALT's hazard avoidance algorithm treats the rover as a single large disk and ensures that no obstacle is present anywhere within it. While this approximation substantially simplifies collision checking, it is overly conservative and hinders the rover's ability to navigate through a region with obstacles spaced more tightly than the rover's modeled diameter.

The next Mars rover, Curiosity, was substantially larger in size and had completely redesigned hardware, but it used largely the same AutoNav software. At every planning step, it had to take four 256 × 256 stereo pairs to provide sufficient terrain coverage. As mentioned in Section I-A, Curiosity's AutoNav system was mostly used in a drive-extending mode to accrue additional distance (typically 3–37 m) after a human-planned directed drive. The most commonly used AutoNav mode included a "VO auto" behavior, in which VO was typically not active unless some trigger dynamically caused it to be autonomously enabled (e.g., high tilt, high slip

TABLE 1. Comparison of AutoNav usage across JPL's past and present Mars rovers. The AutoNav distance includes AutoNav and, when known, mapping and guarded drive modes as well (see Table 5 and Section III-B for the definitions of drive and path-selection modes). Distances reflect the commanded idealized motion of the center of the rover, which does not include TIP motions since the center of the rover does not move during those rotations. Current Perseverance and Curiosity data are newly reported here; other data are from [36].

Rover	Years Active	Total Distance (km)	AutoNav Distance (km)	Percent Distance using AutoNav	Total Sols
Sojourner	1997	0.1	n/a	n/a	83
Spirit	2004-2010	7.7	>1.8	>24%	2,623
Opportunity	2004-2018	45.2	>2.4	>5%	5,111
Curiosity	2012-present	34.6	1.9	5%	4,347 (28 October 2024)
Perseverance	2021-present	32.1	29.1	90%	1,312 (28 October 2024)

measurement) [27]. Curiosity's average speed while driving autonomously was 17.3 m/h [27], which was substantially slower than the directed drive (∼150 m/h when driving straight) due to the highly limited onboard processing power and the need to collect and process multiple images.

China successfully landed and operated two Lunar rovers, Yutu and Yutu-2, in 2013 and 2018, respectively. While the rovers were primarily driven through teleoperation, Yutu-2 was also capable of autonomous driving on "relatively flat terrains" and successfully traversed 2.7 m autonomously on the fifth lunar day, in which autonomous planning was carried out five times, and each planning segment was about 0.5 m [7]. More recently, the Chang'e 6 mission, which achieved the first sample return from the far side of the Moon, carried a 5-kg mini rover named Jinchan. It was advertized as an "autonomous intelligent mini-robot," but its technical details are not published to the best of our knowledge. Zhurong, the first Chinese Mars rover, which successfully landed in 2021, is capable of driving approximately 20 m per sol by combining manual driving and autonomous obstacle avoidance with an A* algorithm [39]. The rover traveled 1921 m until it became inactive due to a sandstorm in 2022.

The autonomous driving capability of Perseverance is built upon the AutoNav framework developed for Spirit, Opportunity, and Curiosity, but with several major upgrades. First, although the main computer of Perseverance is the same 133-MHz RAD750 as Curiosity [known as the Rover Compute Element (RCE)], it has an additional RAD750 compute unit called the Vision Compute Element (VCE), which includes a Virtex 5 field programmable gate array (FPGA). The original purpose of the VCE was to support onboard image processing for the terrain relative navigation (TRN) needed during the entry, descent, and landing phases on Mars [15]. After the successful landing, it was reconfigured to accelerate stereo and VO image processing for driving. While the image processing algorithms are similar to those of Curiosity, the VCE substantially accelerates stereo and VO, the most computationally involved components of AutoNav [38]. This acceleration, in turn, enables the processing of 1280 × 960 imagery, with a higher angular resolution than previous rovers. In addition, Perseverance is equipped with new Navigation Cameras (NavCams) with wider field-of-view optics (approximately 90° horizontal and 70° vertical stereo FOV) and shorter exposure times [19], which eliminates the need to pan the sensor mast and enables image acquisition while in motion. These changes, together with significant software updates, enable Perseverance to drive continuously [thinking-while-driving (TWD)] instead of stopping at every planning iteration [14]. The result is a substantial increase in average driving speed from the 17.3 m/h of Curiosity to ∼92 m/h for Perseverance.

In addition, GESTALT was replaced by a more sophisticated path planner, ENav, which is the topic of this article. Unlike the disk-based hazard avoidance employed by GESTALT, ENav performs hazard assessment using knowledge of each wheel's location using the approximate clearance evaluation (ACE) algorithm [25]. This more precise assessment and the ability to run VO concurrently to cleanly register new terrain into the persistent heightmap reduce unnecessary conservatism and allow the rover to thread between more closely spaced hazards and even straddle modestly sized hazards. ENav also employs more complex path selection. GESTALT only plans one step ahead [using an arc or turn-in-place (TIP)], but ENav evaluates a series of up to four steps ahead using a parameterized tree. Other new features of ENav include robust accommodation of vehicle slip, mid-arc path reevaluation, and a facility for postdrive turn-for-communication. These features are described in detail in Section II. Although the FPGA in the VCE runs portions of the image processing pipeline, the ENav functions are all implemented on the RAD750 CPU in the RCE.

To clarify the terminology and the scope of this article, ENav specifically means the path-planning capability of Perseverance, including elevation mapping, cost evaluation, hazard assessment, and global and local path selection. Other components of AutoNav [28], such as stereo processing and VO [14], are not a part of ENav and hence not included in the scope of this article.

(a)

(b)

(c)

FIGURE 4. Examples of Perseverance's AutoNav avoiding hazards. (a) Record-setting sol 753 drive of 347.7 m in which tracks show AutoNav avoiding distant hazards. (b) Zoomed-in view showing the distant portion of sol 753 tracks where AutoNav clearly diverted around hazards. (c) Sol 884 drive in which tracks show AutoNav deviating around two right-side hazards.

C. ENav REQUIREMENTS

The aforementioned BRS of the Mars 2020 Mission specified that the rover needs to be able to drive 12 km within 85 sols. After accounting for factors such as vehicle slip, diversions from the straight-line paths to avoid hazards, and time allocated for mid-drive activities such as attitude updates involving stopping and imaging the sun, we arrived at commanded drive rate requirements of 100 m/h in benign terrains and 86 m/h in complex terrains. Benign terrain was defined as having a slope less than 15° and with rock density less than 7% cumulative fractional area (CFA). Specific distributions of slope and CFA within benign and complex terrain types were derived through an extensive analysis of potential landing sites and estimates of the paths the rover was likely to take [23].

Implicit in the drive requirements flowed down from the BRS are associated requirements involving the directness of the navigated path. Obviously, driving quickly but taking an unnecessarily circuitous route would not align with the mission objectives. Because more complex terrain necessitates more frequent and greater diversions from the straight-line path, distinct path efficiency requirements were also imposed for benign and complex terrains.

Ensuring vehicle safety is the most important requirement for any software responsible for the AutoNav of a Mars rover. A crucial element of rover safety hinges on the robustness of the planning algorithms to any inaccuracies of the perception system. For Perseverance, this necessitates that the planning software be resilient both to the inaccuracy of the stereo-derived range images (obtained from pairs of 1280×960 NavCam imagery), as well as any misregistration between subsequent range images resulting from imperfect VO calculations. Another key aspect of vehicle safety is ensuring that the motions selected for execution are safe even in the face of unpredictable vehicle slips. Note that the TWD capability, which is a critical element of Perseverance's improved drive rate, makes achieving slip robustness even more challenging. At the point of path selection, the software must account not only for the future slip that could occur during the execution of the next drive step but also for the slip that has already occurred since the last planning imagery was acquired.

A more qualitative goal that significantly impacted ENav's design was ensuring that the decisions made by the autonomous path planner were understandable by the human operators. Previously, Curiosity's GESTALT algorithm incorporated a number of interacting heuristics, making it difficult for nonexperts to understand and explain the rover's selected motions. These heuristics were partly motivated by the fact that, due to limited computation, earlier rovers typically ran either AutoNav or VO, but not both. As a result, GESTALT employed a heuristic to prefer arcs that were in the center of a long stretch of safe potential paths to avoid clipping obstacles should the rover encounter slip. But on Perseverance, we can use both AutoNav and VO to provide the best performance. Therefore, ENav's selection criteria are simpler and more concrete, based primarily on quantitative models of traverse execution time. This has fostered a robust level of trust in the software among the operations team and management.

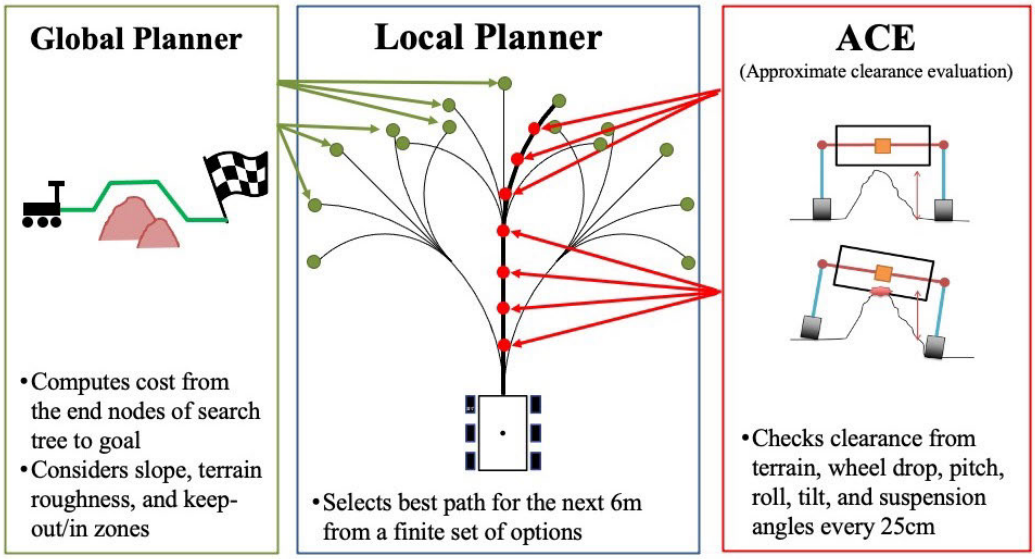


FIGURE 5. **Overall architecture of the ENav path planner.**

Moreover, the algorithm's clarity expedited the maturation of the software.

In summary, Perseverance's ENav software was designed to meet the following goals.

1) Ensuring vehicle safety despite perception and localization limitations—ENav has to be provably safe without humans in the loop. Any part of the rover except the wheels must not touch the terrain or rocks; its attitude (roll, pitch, and yaw angles) and suspension angles must always be within prescribed limits; and the wheels must not ever drop more than a specified height. Since it is on Mars, the rover is not mechanically serviceable. Thus, ENav must always select a motion that is *certain* to be safe in all circumstances.
2) Robustly ensuring vehicle safety against slips–slips were commonly experienced by past rover missions, particularly on sandy terrains. This means that the provable safety must hold under the large uncertainty in the rover's position due to slip.
3) Ensuring that the algorithm implementation is computationally efficient enough to meet average drive rate requirements despite the limitations of the 133-MHz RAD750 processor.
4) Ensuring that the resulting paths through cluttered terrains are not overly tortuous and meet the target path efficiency rate.
5) Ensuring that the objective function of the path-selection algorithm is transparent and interpretable with minimal reliance on heuristics.

Section II describes the algorithm for meeting these goals.

II. METHOD/ALGORITHMIC DESCRIPTION

This section presents the ENav algorithm. Sections II-A and II-B describe the algorithm at a high level, while Sections II-C–II-G provide additional details. Finally, Sections II-H and II-I describe two closely related capabilities of Perseverance: path reevaluation and autonomous turn for communication (TFC).

A. OVERVIEW

Fig. 5 shows the top-level architecture of ENav, which consists of three components: Local Planner, Global Planner, and ACE. The Local Planner, the central piece of the algorithm, selects a drive path reaching out to the planning horizon (typically 6 m from the current position). The available path options are made up of motion primitives organized in a tree structure. The Global Planner is a standard Dijkstra path planner on an 8- or 16-connected graph and evaluates the cost from the end points of the Local Planner's tree to the goal (cost-to-go). ACE is a kinematic collision checker that takes the rover's 2-D pose (x, y, and heading), the uncertainty envelope in the 2-D pose, and the heightmap of the terrain as inputs, and conservatively determines if the rover is safe for all possible poses within the uncertainty envelope. The rover's safety, which is the first requirement identified in Section I-C, is provided by repeatedly running ACE at a fixed interval (25 cm in Perseverance's current configuration) along a prospective path as shown in Fig. 5. The second requirement, robustness to slip, is satisfied by setting the uncertainty envelope to conservatively encompass the worst case slip at the corresponding point on the path. This feature of ENav is described below in Section II-G3. The third requirement for the ability to run with highly limited computational resources is met by the two-stage path-selection process in the Local Planner, as detailed in Section II-B. In a nutshell, the Local Planner first sorts the path options by local and global cost without running ACE on them, and then runs ACE, which is computationally expensive, only on the top-ranked paths for choosing the best path. Finally, the fourth requirement on the path efficiency and the fifth requirement on interpretability are met by the cost function structure, shown in Fig. 6 and presented in detail in Section II-F, that specifies the estimated

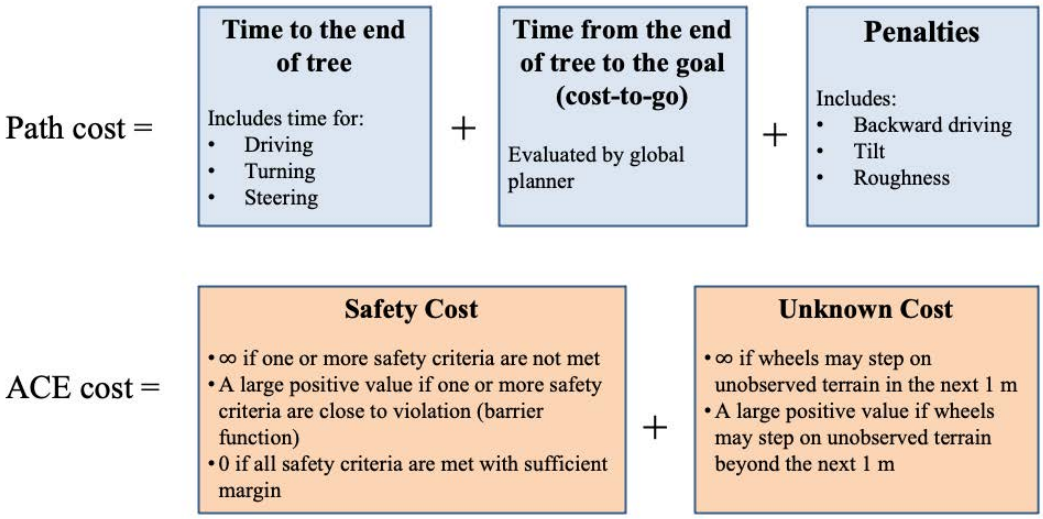


FIGURE 6. **ENav's two-stage cost function.**

time to the goal with explicit considerations of actual vehicle behaviors, including the time for steering the wheels.

ENav is a receding horizon path planner. Regardless of the distance to the final goal, the Local Planner generates a path up to a planning horizon, which is set to 6 m in our application, but can be modified by changing parameters. The rover drives only the initial portion of the selected path (1 m in our setting) until it replans a path over the next planning horizon. The cost-to-go, i.e., the cost from the planning horizon to the final goal, is estimated by the Global Planner and added to the cost function of the Local Planner (the second term in Fig. 6) in order to select paths predicted to minimize the total time required to arrive at the final goal.

B. OVERALL ENav PATH SELECTION ALGORITHM

Overall, ENav aims to minimize the time to the final goal (the first two terms in Fig. 6). The two-stage path-selection process of ENav uses two sets of cost functions, as summarized in Fig. 6: the *path cost* and *ACE cost*. In this section, we describe the overall two-stage process the Local Planner uses to select a path, while leaving the details for the following subsections. For now, the readers only need to understand that the path cost is substantially faster to compute than the ACE cost, which requires running ACE multiple times over a path. Since a single run of ACE takes 10–15 ms on the RAD750 [25], evaluating a 6-m path by running ACE at every 25 cm takes 240–360 ms.

In a nutshell, in the first stage, ENav sorts all the path options by the path cost only. This computation is quick, but the resulting path ranking is not accurate because the ACE cost is not evaluated yet. In the second stage, ENav evaluates ACE only for the top-ranked paths from the first stage. Note that ACE is evaluated many times for each path at 25-cm intervals. ENav stops the second stage once the total number of ACE runs reaches a threshold, M, and returns the best feasible path found up to this point. Since ACE dominates the computation time of ENav, this method results in approximately constant running time for the majority of cases where the rover is driving on benign terrains. M is selected such that the path-planning finishes before the rover reaches the end of the path segment planned in the previous iteration, and hence the rover can drive continuously without stops. For the minority cases where no feasible path is found when the total number of ACE runs reaches M, which typically happens when the rover drives on complex terrains, ENav keeps evaluating the paths in the ranked order until it finds a feasible path. Since this takes longer computation time, the rover needs to stop and "think."

Algorithm 1 shows the overall ENav algorithm. Here, we denote by $\mathcal{P}$ the discrete and finite set of path options considered by the Local Planner, while each path option is denoted by p_i with a one-based index i. Therefore, $\mathcal{P} = \{p_1, p_2, \cdots p_N\}$, where $N = |\mathcal{P}|$. Also, let c_i be the cost of p_i. As mentioned previously, a key parameter of the two-stage algorithm is the minimum number of required ACE runs, M.

In Line 4 of the Algorithm, the path cost is evaluated for all path options in $\mathcal{P}$, where localCost, globalCost, and penalty are defined in Section II-F. The paths are sorted by the path cost in ascending order in Line 6. Then, ENav goes down the sorted list of path options (Line 8) and runs ACE on them to evaluate the ACE cost (Line 9), where the ACE cost is infinite if the path is unsafe. The runAce function in (Line 9) also returns the number of ACE runs required for evaluating the ACE cost of the path p_i. The algorithm keeps track of the total number of ACE evaluations performed, n_{ACE} (Line 10), and the best path found so far, p_{best} (Line 12).

Line 16 describes the stop condition of the algorithm. There are two cases where the algorithm satisfies this condition and successfully returns a feasible path, p_{best}. Fig. 7 shows the two cases on a simplified example with only nine path options ($N = 9$) and $M = 100$. The first is when a feasible path is found and $c_{\min}$ turns finite before n_{ACE}

(a)

	i	Path cost	ACE cost	Total cost	n_{ACE}
	1	50.1	Inf	Inf	24
	2	50.3	5.1	55.4	48
	3	52.0	Inf	Inf	60
Selected	4	**53.0**	**0.0**	**53.0**	**84**
	5	53.2	Inf	Inf	108
	6	55.6			
	7	60.6			
	8	62.1			
	9	66.5			

(b)

	i	Path cost	ACE cost	Total cost	n_{ACE}
	1	50.1	Inf	Inf	24
	2	50.3	Inf	Inf	48
	3	52.0	Inf	Inf	60
	4	53.0	Inf	Inf	84
	5	53.2	Inf	Inf	108
	6	55.6	Inf	Inf	120
Selected	7	**60.6**	**5.4**	**66.0**	**147**
	8	62.1			
	9	66.5			

FIGURE 7. Illustrative examples of Algorithm 1 with $N = 9$ and $M = 100$. (a) Case 1: If feasible paths are found when the number of ACE runs (n_{ACE}) reaches the threshold (M), and ENav returns the best path found up to that point. (b) Case 2: If no feasible paths are found when n_{ACE} exceeds M, ENav keeps going down the list until a feasible path is found. It returns the first path that is feasible. If no feasible path is found among all the N path options, ENav returns failure.

exceeds M, as shown in Fig. 7(a). This usually happens on benign terrains. The algorithm exists as soon as n_{ACE} reaches M, and returns p_{best}, the best paths from potentially multiple feasible paths found at this point. In Fig. 7(a), Paths 2 and 4 are found feasible, and Path 4 is returned as p_{best} because it has a lower total cost. ENav does not run ACE on Paths 6–9 to reduce computation time. The second case is when n_{ACE} exceeds M before a feasible path is found and hence c_{min} remains infinite, as shown in Fig. 7(b). This usually happens on complex terrains. The algorithm keeps evaluating paths out of the ranked list $\mathcal{P}$. As soon as a single feasible path is found [Path 7 in Fig. 7(b)], it returns the path as p_{best}. ENav does not run ACE on Paths 8 and 9 in Fig. 7(b). If no feasible path is found among all the N path options, Failure is returned (Line 19).

We employed this two-stage path selection to limit the computation time of path selection when driving in more benign terrain. The parameter M is set such that the algorithm is completed during the currently executing drive step with some margin. Therefore, the rover can drive continuously in the first case where feasible paths are found before n_{ACE} exceeds M; however, in the second case, the currently executing drive step may finish before the path planning for the next drive step is complete, and the rover may need to pause for "thinking." This time spent with the drive actuators idle would result in reduced average driving speed. In other words, ENav tries to find the best path until the executing drive step completes, but after that, it greedily selects the first feasible path.

The next six subsections will provide the details of the algorithm.

C. COST MAP AND HEIGHT MAP

While HiRISE orbital images from the Mars Reconnaissance Orbiter are available and used for ground-based planning, those images are not sufficient for safe long-range autonomous driving since many hazards cannot be resolved at the available resolution (at best ∼25 cm per pixel). Furthermore, the rover does not yet have the onboard ability to localize itself against the global frame (although this is in development [37]), meaning that its position uncertainty grows as it drives until the rover is relocalized by operators. ENav addresses these unique issues through a two-layer, hierarchical grid map structure, where the upper and lower levels are called the *cost map* and *height map*, as shown in Fig. 8. Fig. 9 shows an actual cost map and height map from Mars, reconstructed from the telemetry data collected during the autonomous drive on sol 178.

1) HEIGHT MAP

The height map is a gravity-aligned digital elevation map. It covers a smaller area around the rover (30 × 30 m in our application) at a higher resolution (10 cm), and is used primarily for ACE cost evaluation. Like its predecessors, the Perseverance rover uses stereo vision for sensing the terrain topography, where the typical range limit for reliable disparity computation is about 15 m at 1280 × 960 image resolution. Once the point cloud from stereo vision is received, ENav converts it into height information in its height map by taking the average of all the points within each cell. Cells that do not have any points from the point cloud are marked "unknown."

The horizontal field of view of the NavCams, which are used for onboard stereo processing, is about 90°. Therefore, a single NavCam observation results in a 90° wedge in a heightmap. The rover takes a pair of NavCam images every 1 m. When a new NavCam observation is made, the height information in the new wedge is overwritten. As the rover moves, height information is accumulated into the height map, as shown in Fig. 9. However, since old height information is less reliable due to accumulated pose estimation error, data in a height map cell "expires" and is marked as "unknown" if the change in the rover's overall position uncertainty (between the time the data was collected until now) exceeds a preset threshold.

2) COST MAP

ENav's cost map covers a greater area (100 × 100 m in our application) than the height map at a lower resolution (1 m), and is used for local and global path planning. The goal disk, KOZs, and keep-in zones (KIZs), which are specified by human operators on Earth, are registered into the cost map at the beginning of autonomous driving. Operators typically use the orbital images to select the desired goal location, as well as keep-out and KIZs for constraining the rover's motion. If the goal is outside of the cost map, the costs along the nearest edge(s) are initialized to the time needed to drive in a straight line from that cost map cell to the goal. The cost map does not hold any height information. This is because there is no onboard height information beyond the observation horizon provided by the rover's stereo vision.

Algorithm 1 ENav Path Selection

$N \leftarrow |\mathcal{P}|$
$M \leftarrow$ Minimum number of ACE runs before returning a selected path
for $i \leftarrow 1$ to N **do**
    $c_i = \text{localCost}(p_i) + \text{globalCost}(p_i) + \text{penalty}(p_i)$ ▷ Evaluate the path cost for all paths
end for
Sort $\mathcal{P}$ by c_i in ascending order ▷ Sort by the path cost; index is reassigned after sorting
$c_{\min} = \infty$
for $i \leftarrow 1$ to N **do**
    AceCost, $n \leftarrow \text{runAce}(p_i)$ ▷ ACE returns the ACE cost and the number of ACE runs
    $n_{\text{ACE}} \leftarrow n_{\text{ACE}} + n$
    if $c_i + \text{AceCost} < c_{\min}$ **then** ▷ Evaluate the ACE cost
        $p_{\text{best}} = p_i$
        $c_{\min} = c_i + \text{AceCost}$
    end if
    if $c_{\min} < \infty$ **and** $n_{\text{ACE}} \geq M$ **then** ▷ If there is one or more safe paths and at least M ACE runs are performed
        return p_{best} ▷ Return the best path that has been found so far
    end if
end for
return Failure ▷ If all paths are unsafe, return failure

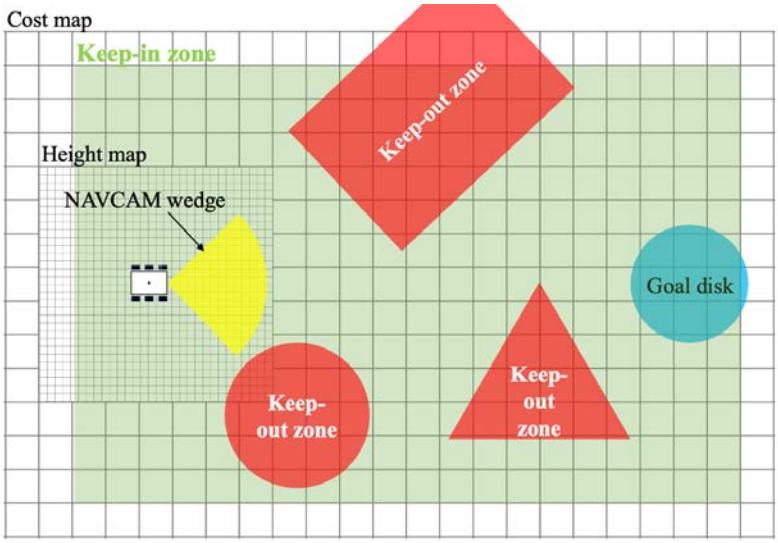


FIGURE 8. Conceptual illustration of ENav's hierarchical map structure, consisting of a larger but lower resolution cost map and a smaller but higher resolution height map.

While the cost map is agnostic about the terrain topography, it does hold the terrain tilt and roughness at those cells that have been observed by the rover, and hence, previously covered by the heightmap. For each cell in the cost map, ENav runs a plane fit algorithm over the heightmap cells within the envelope radius of the rover (2 m in our application) to compute the tilt. The roughness is either the variance or the maximum deviation in height from the fit plane, depending on parameter settings.

The cost map is updated at every planning iteration. Each cell holds the ''cost type''—an integer value representing the reason for untraversability. When ENav is initialized, a cost map cell could be untraversable only due to the keep-in/out zone violation. Once the rover observes the terrain with stereo vision, some cells might be marked untraversable due to excessive terrain tilt. As the rover traverses and runs ACE on path options, more cells might be marked untraversable for various reasons such as insufficient ground clearance or excessive wheel drop (see Section II-G for more details).

D. KEEP-IN AND -OUT ZONES

While ENav does not directly use a global map created by orbiters, human operators manually specify keep-in/out zones (KIOZs) based on their visual assessment of orbital imagery in the day-to-day operations, as shown in Fig. 8. ENav requires a path to be always inside at least one of the KIZs

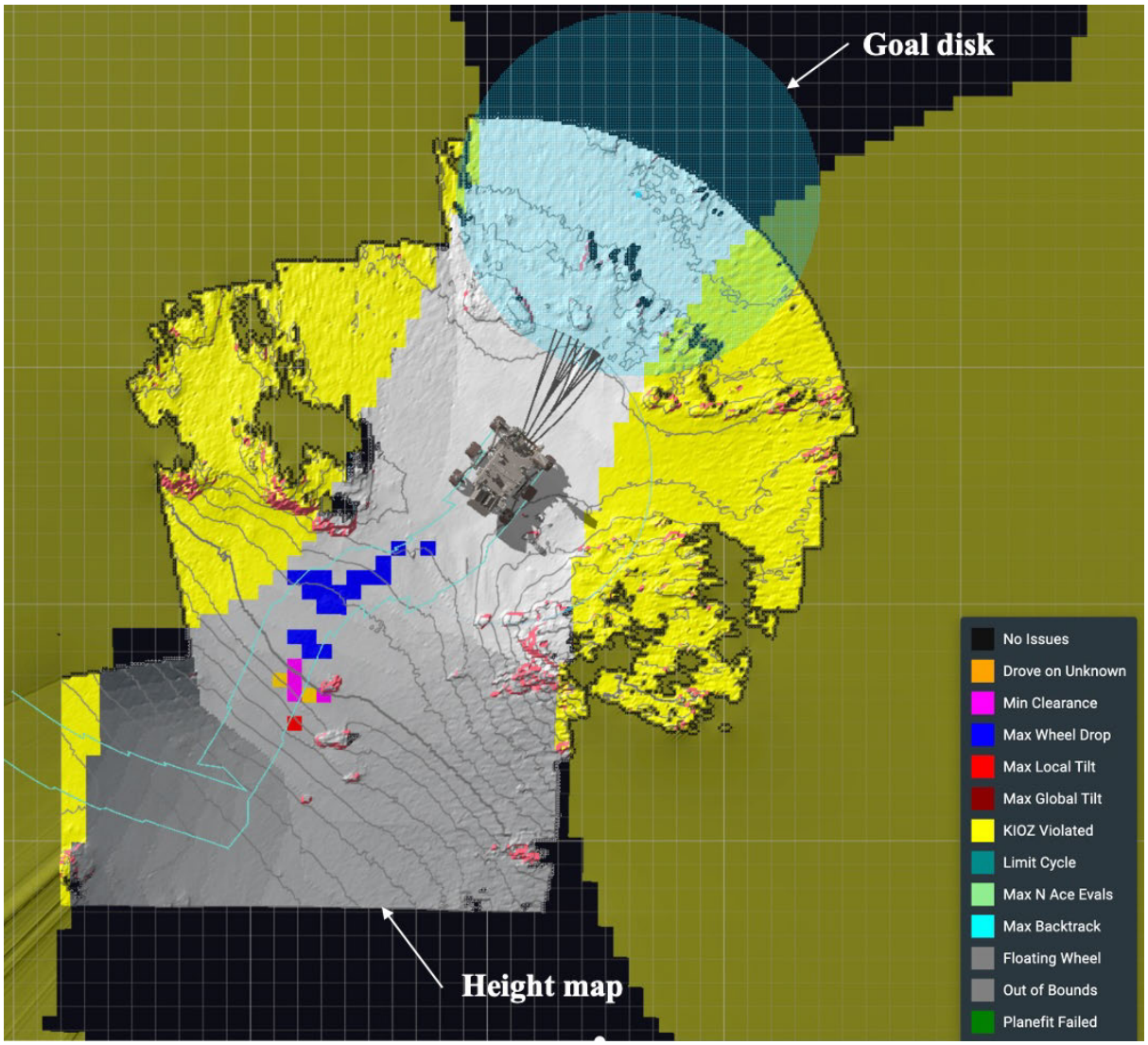


FIGURE 9. Actual cost map and height map from Perseverance on sol 178 on Mars. The cost map is at 1-m resolution and colored by the cost information in each cell. The observed surface in the height map is shown in white with 10-cm resolution. The circular areas in the cost map shown in yellow on both sides of the rover are human-specified keep-out zones (KOZs). On this sol, the rover successfully arrived at the goal autonomously by avoiding the KOZs as well as the obstacles detected by its onboard cameras.

and outside all the KOZs. The user can specify rectangular, circular, and triangular KIOZs as part of the commands sent to Mars.

KIOZs are specified relative to the global coordinate frame, known as the site frame in rover operations [1]. As previously mentioned, since ENav does not have onboard global localization, the uncertainty in the position estimate relative to the KIOZs grows monotonically until the rover is manually localized by a ground-in-the-loop process. ENav accounts for this uncertainty by shrinking KIZs and dilating KOZs by the current uncertainty as it drives. ENav employs a model in which the current position uncertainty is a weighted sum of two distances: the amount of odometry the rover has accumulated since the last manual localization *with-VO knowledge* (i.e., corrected for slip) and *without-VO knowledge* (i.e., without slip corrections). From sol 0 to 1312, we have conservatively assumed 5% uncertainty of the *with-VO* distance (allowing for worst case VO and RIMU-derived yaw error), and 50% uncertainty of the *without-VO* distance. The growth of KOZs and the shrinkage of KIZs are considered in both the Global and Local Planners. For the Global Planner, every time the cost map is updated, ENav labels each cell that violates KIOZ constraints based on the *expected* radius of the uncertainty disk, ϵ, for the current rover position (x_R, y_R) and the center location of the cell (x, y)

$$\epsilon(x, y) = \text{uncertainty}_R + C\left(\sqrt{(x - x_R)^2 + (y - y_R)^2}\right)$$

where C is a constant, and uncertainty_R is the current position uncertainty. The Local Planner also checks KIOZ satisfaction

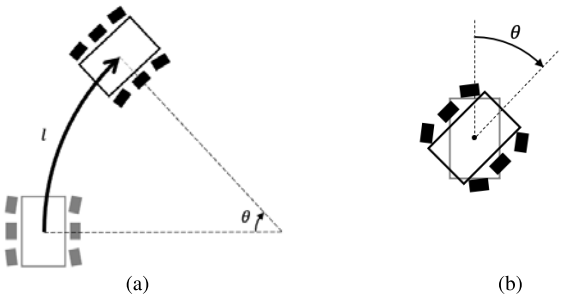


FIGURE 10. **Perseverance's motion primitives. (a) Constant-curvature arc. (b) Turn in place.**

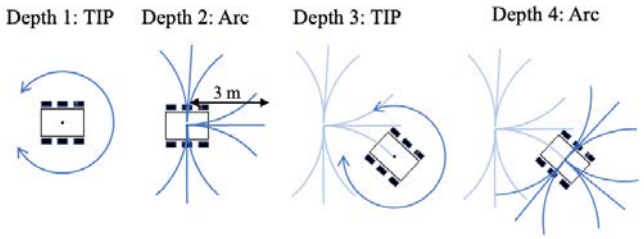


FIGURE 11. **Example of the fixed tree of ENav, which contains the majority of the path options. This illustrates four depth levels. As the name suggests, the geometry of the tree relative to the rover's pose is fixed across iterations.**

at every node of all the path options at which ACE is run (typically every 25 cm) based on the odometer distance plus the maximum estimated slip. Finally, to ensure safety, ENav checks the feasibility of the selected path once again against KIOZs before it is returned.

E. SEARCH TREE

ENav's search tree is a composite of three subtrees: *fixed tree*, *retreat tree*, and *heritage tree*, all of which are constructed with two types of motion primitives—arc and TIP. The fixed tree has a fixed configuration across all the planning steps, while the other two trees are generated dynamically at every step.

1) MOTION PRIMITIVES

There are two mechanical constraints in the motion of Perseverance.

1) Wheels cannot be steered while driving. This is because the driving and steering motors share the same motor controllers. Therefore, Perseverance cannot drive a path with continuously changing curvature.
2) Among the six wheels, only the front and rear wheels are steerable. Therefore, Perseverance cannot move sideways in a crab-like motion.

As a result, a feasible path for Perseverance is always composed of the following two motion primitives.

1) *Arc* with a constant curvature, parameterized by the arc length l and the heading change θ as in Fig. 10(a). An arc with negative length specifies backward driving.
2) TIP, i.e., a turn around the rover's geometric center without translational motion, parameterized by the heading change θ as in Fig. 10(b). Technically, TIP is modeled as an arc with $l = 0$.

2) FIXED TREE

The fixed tree has a fixed depth and includes all possible combinations of a finite set of motion primitives specified for each depth. Fig. 11 shows an example construct of the fixed tree which uses the maximum number of depths that may be specified. Each depth only involves arcs or TIPs. All arcs at the same depth have the same l and only differ in θ. For both arcs and TIPs, a finite set of θ is specified. For example, the current Perseverance parameterization uses a tree with three depth levels consisting of listed below.

1) *Depth 1 (14 Options):* TIP with $\theta = \{\pm 90°, \pm 60°, \pm 45°, \pm 30°, \pm 20°, \pm 10°, 0°, 180°\}$.
2) *Depth 2 (11 Options):* Arc with $l = 3$ m and $\theta = \{\pm 60°, \pm 45°, \pm 30°, \pm 20°, \pm 10°, 0°\}$.
3) *Depth 3 (11 Options):* Arc with $l = 3$ m and $\theta = \{\pm 60°, \pm 45°, \pm 30°, \pm 20°, \pm 10°, 0°\}$.

Exhausting the combinations in this tree configuration results in $14 \times 11 \times 11 = 1694$ path options. As the name suggests, the configuration of the fixed tree is fixed across the planning cycles.[2]

3) RETREAT TREE

As we wrote previously, ENav is a receding horizon planner where only a portion of the planned path is executed before the path is replanned in the next planning cycle.

The retreat tree includes a single path that reverses the action taken in the previous step. Fig. 12 shows an example. Imagine that the selected path consists of 0° TIP at depth 1, a 3-m straight arc at depth 2, and a 3-m curved arc at depth 3 at Iteration N. Before the next planning cycle, only the first 1 m of the 3-m straight arc at depth 2 is executed. The retreat path in this case is a 1-m-long, straight, backward driving path ($l = -1$ m, $\theta = 0$).

The retreat tree allows the rover to undo the motion in case ENav cannot find a way to make progress. It is chosen only when no feasible path is found among the fixed tree and the heritage tree. If there is no slip, the retreat path is always safe, given the successful execution of the action from the last step. In reality, it could become infeasible if the rover slips and reversing the action does not result in the same path. ENav is capable of retreating multiple steps up to a user-specified limit (currently, only three steps are stored) because it remembers the actions taken in the previous steps to avoid cycling back and forth.

[2]Directed drives also use the same path-selection algorithm, but with a different fixed tree and no terrain assessment.

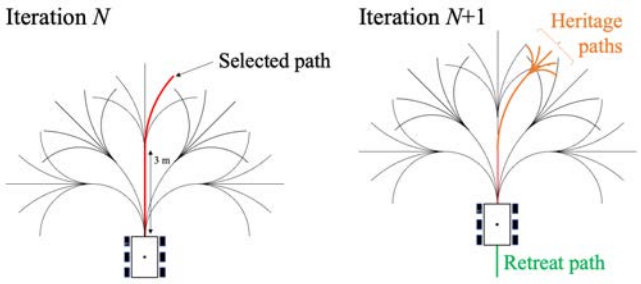


FIGURE 12. Retreat and heritage trees. Only the first 1 m of the selected path in Iteration *N* is executed. In Iteration *N* + 1, the executed portion of the path is added as the retreat tree, while the unexecuted portion is augmented with 1-m branches and added as the heritage tree.

4) HERITAGE TREE

When the driving distance between planning cycles, denoted by Δd ($\Delta d = 1$ m in our case), does not match the arc length l (3 m), the previously selected path is often not included in the fixed tree in the next step. Fig. 12 shows an example. Since the fixed tree moves with the rover, the previously selected path (the red line in the left chart) in Iteration N does not overlap with any of the path options in Iteration $N+1$ (black paths in the right chart). This inconsistency could result in a couple of practical issues. First, the rover needs to change the turning radius frequently, which in turn results in a longer driving time because the rover has to stop for steering. Second, it results in an increased chance of retreating on complex terrain because the previously selected path, which is known to be feasible, is no longer included in the next cycle.

To resolve these issues, we include the path selected in the previous planning step in the tree. Since it is shorter by Δd, it is augmented with arcs with $l = \Delta d$, as shown in Fig. 12-right (the orange paths). We call these paths ''heritage paths'' and the tree that holds the collection of heritage paths is called the heritage tree. Inclusion of the heritage tree in ENav results in more consistent behavior with a smaller number of stops for steering. The heritage tree is dynamically generated and added to the search tree at every planning cycle.

5) POSE SAMPLING

For every path in the tree, ENav samples a finite number of *poses* (x, y, and heading θ) at a fixed interval of drive distance for arcs (25 cm in our application) and at a fixed turning radius for TIPs (10°). In the later steps in the Local Planner, we evaluate the penalty and the ACE cost at every sampled pose. For example, a 6-m path without TIPs has 24 instances of pose evaluations; a path with a 30° TIP followed by a 6-m arc has 27 poses.

F. PATH COST EVALUATION

At each planning cycle, once the tree is built, the Local Planner evaluates the path cost of every path in the tree (Line 4 of Algorithm 1). As shown in Fig. 6, the path cost involves three terms: local cost, global cost, and penalty.

1) LOCAL COST

The local cost is the estimated wall clock time of executing a path up to the end of a path in the Local Planner's tree

$$\text{localCost} = \sum_{i=1}^{D} (t_i^{\text{steer}} + t_i^{\text{drive}})$$

where D is the number of motion primitives in a given path, t_i^{steer} is the time required for steering the wheels before executing the arc or turn in the ith motion primitive, and t_i^{drive} is the time required for driving or turning in the ith motion primitive. If the tilt of the rover is below a threshold, the rover will steer all four steerable wheels at the same time; if not, it will steer one wheel at a time to ensure the stability of the rover, which takes a longer time but incurs less slip. Therefore, t_i^{steer} also depends on the estimated tilt at the start of the ith arc. The estimation of t_i^{drive} is straightforward because the rover's maximum wheel ground speed is fixed at 4.2 cm/s and the maximum TIP speed at 1.5°/s. Due to the slow speed of the rover, dynamics are ignored.

2) GLOBAL COST

The global cost is the estimated wall clock time to drive from the end of the path to the final goal. For a given x, y, and θ (heading) of the end pose of a given path, it is given by

$$\text{globalCost} = t_{\text{CTG}}(x, y) + t_{\text{turn}}(\theta)$$

where t_{CTG} is the time to drive from x, y to the final goal given by the Global Planner, which is a standard Dijkstra algorithm running on an 8- or 16-connected graph, while $t_{\text{turn}}(\theta)$ is the time required for making a turn from θ at the end of a path in the Local Planner's tree to the initial heading angle of the global path given by the Global Planner.

3) PENALTY

The cost function includes penalties on terrain tilt and roughness, sampled from the cost map

$$\text{penalty} = \sum_{i=1}^{n_p} \Delta l \, (C_\tau h(\tau_i, \tau\text{th}) + C_r h(r_i, r\text{th}))$$

where n_p is the number of the sampled poses in the path (see pose sampling in Section II-E5), Δl is the drive distance between the ith and $(i+1)$th poses (in case of TIP we use the distance the wheels travel during the turn), τ_i and r_i are the tilt and roughness at the ith pose sampled from the costmap, respectively, τth and rth are fixed thresholds for tilt and roughness, respectively, C_τ and C_r are constants, and $h(x, c)$ is a hinge function given by

$$h(x, c) = \max(0, x - c).$$

Intuitively, when the tilt τ is below the threshold τth, there is no penalty for tilt; and if $\tau > \tau$th a penalty is added proportionally to $\tau - \tau$th. The same applies to roughness.

G. ACE COST EVALUATION

Next, we explain the ACE cost evaluation in Line 9 of Algorithm 1. The ACE cost of a path is computed by running ACE on all the poses sampled on the path, which are shown as red dots in Fig. 5. For example, a 6-m path without TIPs requires 24 ACE runs if the interval of pose samples is 25 cm. The ACE cost of a path is the simple sum of the ACE costs at all the evaluated poses on it. Since the paths are organized in a tree structure, the ACE results on a path are reused for other paths that share the same arcs or turns in order to save computation time. Therefore, the number of actual ACE runs for a path (n in Line 9) is not constant.

1) ACE OVERVIEW

ACE is a highly complex algorithm, and it was presented in detail in our previous publication [25]; hence, we will describe only an overview of it.

Fig. 13 provides a functional overview of ACE. It takes a height map, the 2-D pose of the rover (x, y, and θ), and a slip vector representing the predicted direction and maximum extent of slip. The output from ACE is the conservatively estimated worst case ground clearance, worst case wheel drop (which will be explained shortly), and min/max bounds of roll, pitch, and suspension angles. From the 2-D pose and the slip vector, ACE computes the *wheel boxes* for all six wheels, which represent the area in the xy plane within which each wheel could possibly be located at a given pose. In other words, wheel boxes are the uncertainty envelope for the wheels. Combined with the height map, ACE computes the maximum and minimum height within each wheel box. The maximum height difference of a wheel box is called the *wheel drop* because it is the maximum possible drop of a wheel when driving within the wheel box. From there, ACE propagates the worst case estimates of the parameters in the kinematic model. From the min/max height of all six wheels, it evaluates the min/max angles of the articulated suspension joints (called "rocker" and "bogie" angles), which are used to compute the min/max roll and pitch angles for estimating the minimum clearance between the rover body and the terrain.

If all the metrics are within the prespecified thresholds in the worst case, the rover is deemed robustly safe with the given pose and slip vector. In practice, it is not desirable to be too close to the threshold. Hence, the resulting ACE cost of a pose is given by

$$\sum_{i=1}^{n} C_i b(x_i, c) + p_U$$

where C_i is constants, and n is the number of the min/max constraints, each of which requires either $x \geq c$ or $x \leq c$. For each constraint, the barrier function b is given by

$$b(x, c) = \begin{cases} \infty, & \text{if the constraint is violated} \\ \frac{1}{|x-c|}, & \text{if the constraint is satisfied} \end{cases}$$

Practically, $b(x, c) \sim 0$ when the constraint is satisfied with an ample margin; it has a large positive value if the constraint is satisfied but with a slim margin; it is infinity if the constraint is not satisfied. The cost from the barrier function injects a preference for a path that does not come close to the safety threshold in ENav's path-selection process. Finally, p_U is a penalty for having unknown cells in wheel boxes, which will be explained next.

2) UNKNOWN TERRAIN PENALTY

The height map obtained from stereo vision often has "holes," shown as the black cells in the height map in Fig. 13, for example. The holes are created usually because the portion of terrain is behind an extrusion, such as a rock or a sand dune, and cannot be seen from the camera. We call such "holes" unknown terrain. ENav performs a dilation step over small unknown areas up to a parameterized fraction of the wheel width, currently 30%. It is not desirable for the rover wheels to step on unknown terrains. However, since unknown terrains are common, particularly in the far-field, prohibiting the rover from unknown terrain for the entire path up the planning horizon is overly conservative and often results in no feasible path. Therefore, we restrict the rover from stepping on unknowns only up to the point where the path is replanned (typically 1 m away from the current position) by setting p_U to infinity. Otherwise $p_U = 0$. The logic behind this policy is that, if a danger is hidden in a far unknown terrain, it will be avoided once it is within the distance driven at each planning cycle. In practice, an unknown patch of terrain in the far-field often disappears once the rover comes closer due to the more favorable view angle.

3) SLIP PREDICTION

ENav employs a simple approach to estimate the slip vector in Fig. 13. It assumes that the direction of slip is the gradient of the local slope. The worst case magnitude of slip is computed by

$$d_{\text{slip}} = r_{\text{slip}} \cdot \min\left(d_{\text{predict}}, d_{\max}\right)$$

where r_{slip} is the slip ratio, d_{predict} is the *prediction distance*, and $d_{\max}$ is the maximum prediction distance. The slip ratio is calculated as a function of slope called a *slip curve*, which is encoded in a parameter table using values that were empirically obtained from past drive data. The slip curve can be updated by changing the parameter table. The prediction distance is the accumulated odometry from the point where the mapping and VO stereo images in the current iteration were taken (Point 1 in Fig. 14) to the position at which ACE is run. This is because the position uncertainty against the onboard height map accumulates after the last imaging of the terrain. Due to the limitation of the onboard processing power, it takes ~10 s to process the images for localization and mapping, during which the rover moves ~50 cm. Fig. 14 visually explains this concept. Point 1 is a past position of the rover where an image pair was taken for VO and stereo vision; and Point 2 is the point at which the path currently being planned will start. Point 3 is the point at which the path is planned for the next cycle to start. The distance between

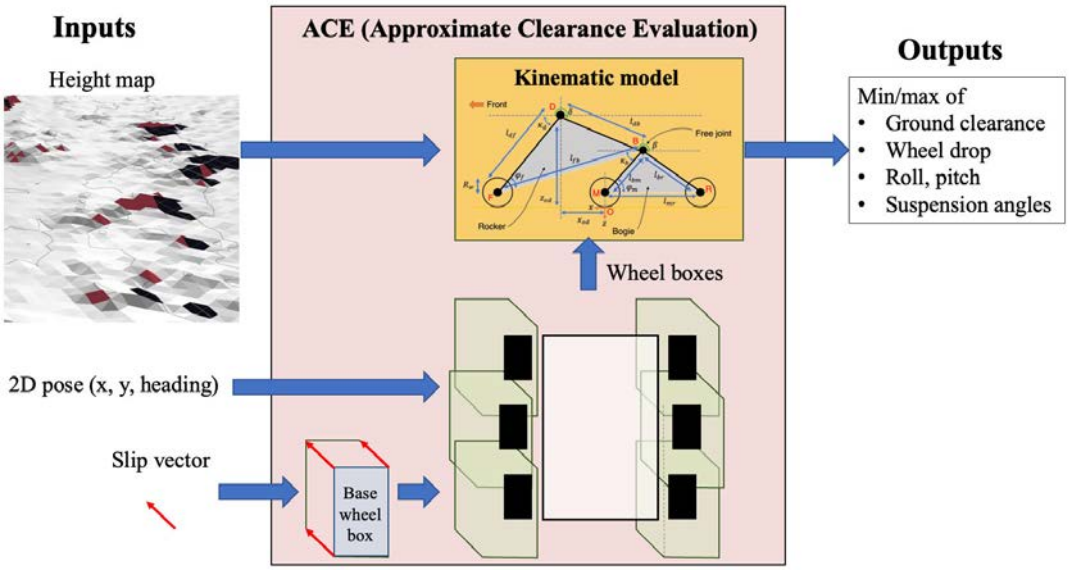


FIGURE 13. **Overview of ACE, the collision checking algorithm of ENav.**

Points 1 and 2, as well as 2 and 3, is about 1 m. Suppose the rover is driving between Points 1 and 2 and planning for a path that starts at 2. The prediction distance d_{predict} is 1 m at Point 2 and 2 m at Point 3 for this path. The predicted slip, d_{slip}, grows linearly between Points 1 and 3. Although a new pair of images is taken around Point 2, it is not used for planning the path between 2 and 3 since image processing and path-planning take ∼20-s combined.

ENav stops growing the slip prediction after Point 3 (by setting $d_{\max}$ to the distance between Points 1 and 3 in this case). This is because the planner will plan again for a new path starting at Point 3 in the next iteration; hence, the potential collisions beyond this point can be avoided by future replans. It is also a practical engineering choice because if we grow the uncertainty margin up to the planning horizon, many path options are blocked, and it typically ends up with a substantially lower drive success rate.

Other uncertainties are represented as the ‘‘base wheel box,’’ which is again specified by a set of parameters. This base wheel box is extended with the slip vector, as shown in Fig. 13, to include all the possible locations of the wheel given the uncertainties, essentially serving as the uncertainty envelope. Since ACE produces the worst case bounds for the wheel boxes, the rover is guaranteed to be safe for any deviation from the predicted pose within the uncertainty envelope if it passes the ACE check.

This concludes the detailed explanation of the path-planning algorithm. However, besides path planning, ENav is also used for two other applications: mid-arc path reevaluation and TFC, which will be briefly described in the next subsections.

H. MID-ARC PATH REEVALUATION

As explained in Section II-G, the path driven between Points 2 and 3 in Fig. 14 is planned based on images acquired at Point 1. As a result, position uncertainty accumulates up to 2 m before the next replanning cycle. Although ACE’s conservative slip margins account for uncertainties, there remains a small possibility that the rover may experience slip exceeding the modeled margin. As an additional safety measure, ENav runs *mid-arc reevaluation* to confirm the safety of the arc currently being executed. Specifically, it reruns ACE on the remainder of the currently driving arc with updated position knowledge from the latest wheel encoder, VO, and IMU observations. Mid-arc reevaluation is particularly useful for counteracting yaw slip. Yaw angle is measured by the IMU at 8 Hz, whereas translational slip is estimated by VO, which runs only at ∼0.04 Hz. If mid-arc reevaluation returns failure, the rover aborts the current arc and commands ENav to plan a new path.

ENav handles path reevaluation using the same mechanism as path planning. That is, it constructs a ‘‘tree’’ containing only a single path—the path currently being executed. The rover poses are sampled more frequently in mid-arc reevaluation (typically every 10 cm) than path planning (typically every 25 cm), meaning that ACE is run at a shorter interval. Since ACE is run only on a single path, path reevaluation can be computed quickly.

I. TURN FOR COMMUNICATION

The rover needs to point the high-gain antenna on its back to communicate directly with Earth, but the visibility can be blocked by its own structures, such as the remote sensing mast (‘‘head’’ of the rover) or the radioisotope thermoelectric generator (RTG) on its tail, particularly when it is on a slope. There are also occasions when the science team wants the rover to stop at a particular orientation after an autonomous drive to investigate a target of interest.

ENav provides a capability called autonomous TFC, to orient the rover in a direction that is favorable for

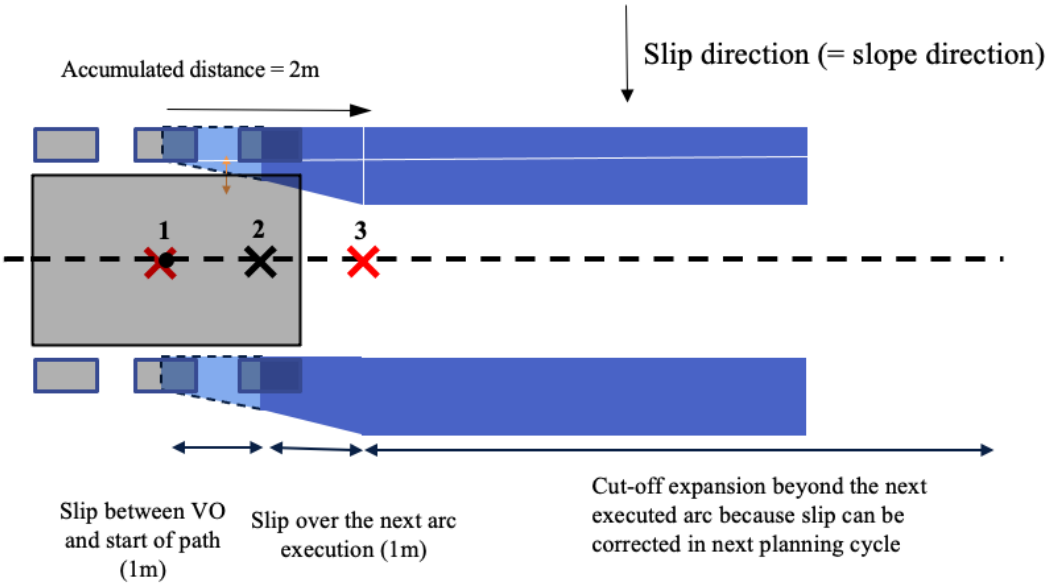


FIGURE 14. **Slip and uncertainty growth model used by ACE. Point 1 is the position of the rover at which the image pair used for planning in the current iteration was acquired, Point 2 is the location at which the path being planned in the current iteration starts, and Point 3 is the location at which the path to be planned in the next iteration starts. The conservative slip estimate used by ACE's wheel box expansion is proportional to the distance from Point 1 to the location at which ACE is evaluated.**

communication or science at the end of the drive. To use TFC, the ground operator specifies the maximum tilt that allows an unoccluded high-gain antenna (HGA) pointing direction at each of a discrete set of headings. After the regular drive is concluded, another autonomous drive is launched using a smaller disk size. The rover continues moving toward the goal until it finds a safe place to turn to the nearest heading where the maximum allowed tilt exceeds the actual terrain tilt, or it enters the smaller goal disk.

Internally, the TFC behavior is implemented by dynamically building a "TFC tree," which only contains two path options: clockwise and counterclockwise TIPs to the nearest acceptable headings. Then, ENav uses the same tree search algorithm presented in Section II-A to select the best motion.

III. FSW IMPLEMENTATION

The flight software (FSW) of the Perseverance rover consists of many modules written in the C language. All the ENav algorithms were packaged into a single FSW module named *NAVLIB*, which has ∼17 000 physical source lines of code (SLOC) comprising ENav and its interface to the rest of the rover FSW infrastructure. This section describes the overall implementations of NAVLIB.

A. IMPLEMENTATION AS A PORTABLE LIBRARY

NAVLIB was designed to be stand-alone and portable. This allowed for integrating this library both with the M2020 FSW, where it was wrapped into its own library module, as well as with a lighter and more versatile environment, such as the Robot Operating System (ROS) for quick prototyping, testing, and data analysis. We also developed a ROS-based simulator called ENav Sim (Section IV-A1a) [30]. Packaging the core algorithm as a stand-alone library also allowed researchers with no knowledge of the FSW to reuse ENav for other research projects and flight missions, as well as compare it against other path-planning approaches and possibly improve it further in the future. For example, a further enhancement using machine learning-based heuristics was developed and implemented based on NAVLIB [6].

B. ENav PATH SELECTION MODES

The rover operator invokes the autonomous path-planning capability of ENav through the "GoTo" command, which implements one leg of a drive by moving toward a particular goal location. Typically, multiple GoTo commands are issued in series to implement multiple drive legs to reach the final goal for the sol. A GoTo command takes two arguments: a *tolerance* to specify a radius around the goal, and a *path_mode* to select between four available path-selection strategies listed here. The *path_mode* is chosen from the following four options.

1) *UNGUARDED:* ENav picks the best path to the next waypoint *without* taking into account KIOZs (Section II-D) or the terrain being traversed. This essentially means a path is planned in a free space that human operators have determined has no obstacles. This drive mode is used to force the rover to drive straight between waypoints.
2) *GUARDED:* ENav picks the same uninformed path as in UNGUARDED mode, but evaluates the safety of that path by considering the terrain being traversed and the KIOZs. If the next step in the path intersects with

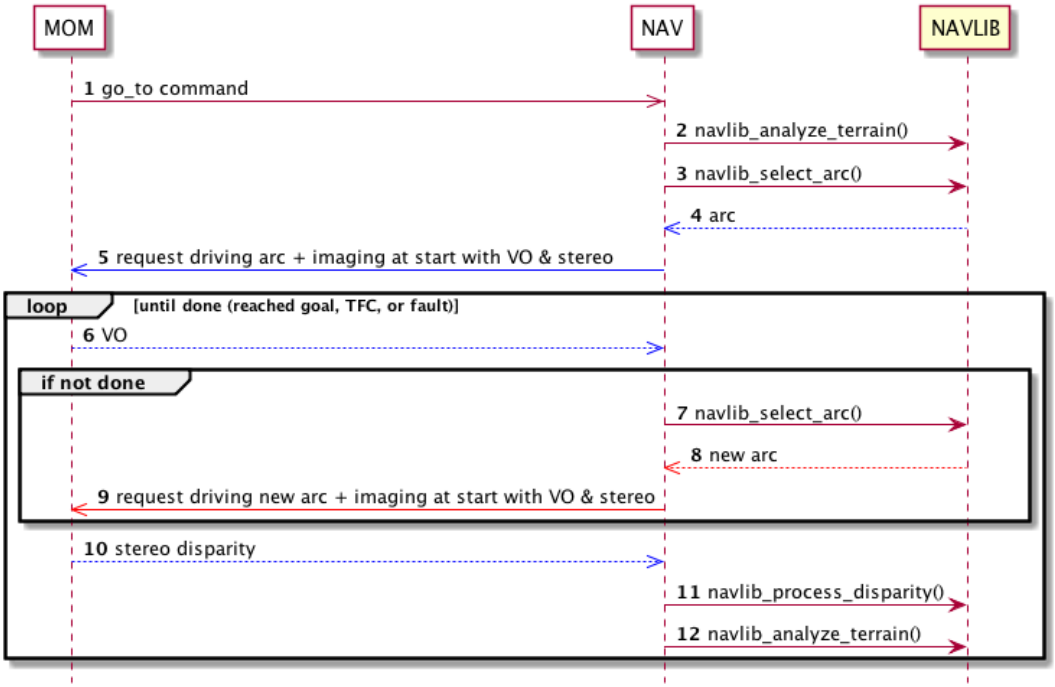


FIGURE 15. GoTo process in the nominal case (i.e., success). Number 6 indicates forwarding of VO VCE results for NAV and ENav processing.

obstacles or KOZs or leaves KIZs, the rover is stopped before that step is executed.

3) *AVOID_KOZ:* ENav picks the best path to the next waypoint by taking into account the KIOZs, but *without* considering the terrain being traversed.
4) *AVOID_ALL:* ENav picks the best path to the next waypoint by taking into account both KIOZs and the terrain being traversed. This full-fledged autonomous driving is by far the most used mode.

All four modes use the same ENav path planner. In the UNGUARDED, GUARDED, and AVOID_KOZ modes, the ACE collision checker (Section II-G1) is not run when planning a path; in the GUARDED mode, though, ACE is run on the selected path to evaluate its safety ahead of time. Likewise, the KIOZs are ignored while planning in the UNGUARDED and GUARDED modes, but in the GUARDED mode, the KIOZ constraint satisfaction is checked on the selected path. During mid-arc reevaluations in every drive mode, KOZs and KIZs are checked against the current position, and if a violation is found, the drive will be halted immediately.

Note that the driving behavior, which is referred to as "driving mode" in the rest of the article, is defined by a combination of ENav's path-selection mode and the imaging mode, which is specified by separate parameters. The FSW controls its autonomous imaging mode independently of individual drive commands. For example, if VO mode is enabled, then VO images will be requested and processed at the usual cadence, typically using 1-m steps in Mars flight operations, during whatever mobility command is currently active. Similarly, if MAP mode is enabled, then images will be processed into point clouds and added to the onboard Height map. Perseverance's nominal "AutoNav" driving mode is a combination of AVOID_ALL ENav path_mode, with both VO and MAP modes enabled. To avoid confusion, we use upper case for ENav path_modes and title case for drive modes in this article.

C. INTERACTION WITH OTHER FSW COMPONENTS

The autonomous driving capability of Perseverance is realized by intricate interactions between NAVLIB and many other FSW modules. By design, the NAVLIB module is only called by one other module: the navigation services module, named *NAV*. NAV runs as an independent VxWorks task, executing an infinite loop that listens for incoming messages from other modules as well as from itself, processes those received messages, performs various tasks such as calling NAVLIB's public interfaces, and sends out its own messages. The NAV module also interacts with the mobility manager (MOM) module, named MOM, which operates at a higher priority. For example, when MOM receives a GoTo command, it will send messages to NAV, which will internally call NAVLIB to find the best path to drive, then message that path back to MOM, which will then enqueue the specified maneuver into the Drive module, to be performed as soon as any previously enqueued maneuvers have completed. Meanwhile, as the rover drives the paths provided by NAVLIB, the MOM module relies on NAVLIB to perform mid-arc

path reevaluation (Section II-H), where it periodically checks that the path being driven remains safe, based on the latest estimate of the pose of the rover, which is updated by wheel encoder, VO, and IMU measurements.

The next two subsections provide more details on what a GoTo command is and how the NAVLIB, NAV, and MOM modules interact when executing such commands.

1) FSW MODULE INTERACTIONS IN A NOMINAL SCENARIO

Fig. 15 shows the interactions of the MOM, NAV, and NAVLIB modules in a nominal case where NAVLIB never fails to find a feasible path toward the goal. Each arc selected by NAVLIB is truncated to be 1-m-long or at most 30° for a TIP. While that arc is executing, the future pose of the rover at the end of that arc is predicted, together with its associated uncertainty due to slip and localization error, and a new path is planned while that arc is being driven so the rover can keep on driving and maximize its traverse rate. This parallel processing scheme gives the Perseverance rover the ability to drive continuously during autonomous drives, which we call TWD [28]. Although the wheel rotation rate of Perseverance is identical to that of Curiosity, TWD results in substantial increases in the average driving speed because Curiosity needs to stop at every planning step to "think" [18], [21].

2) FSW MODULE INTERACTIONS IN AN OFF-NOMINAL SCENARIO

Fig. 16 shows the interactions between the modules when NAVLIB is unable to find a feasible path to the goal and the GoTo fails. Nominally, when NAVLIB plans a new path through the environment, it relies on a mesh that corresponds to stereo disparity images taken at the start of the previously driven arc (i.e., from the previous planning cycle). As a result, the terrain mesh would be created from the images that had been taken up to 2 m before the start of the new path being planned. This is because each path-planning cycle starts as soon as the VO results are available, which is before the stereo disparity results needed to create a new terrain mesh are available. If NAVLIB fails to find a feasible path for the first time, it will try the path planning again once the newer stereo disparity results are received, in case the more up-to-date mesh, which should have fewer occlusions in the near field where the rover is predicted to start the new plan, helps find a feasible path. If NAVLIB still fails to find a path with this newer mesh, then it will command a stop and request new VO and stereo disparity at the stopped location and try one last time to find a path to the goal once both VO and stereo disparity are received. This is the best that can be attempted because, in that case, there is no uncertainty in the rover pose (unlike when it needed to be predicted while the rover was in motion and potential slip needed to be accounted for), and the mesh used for planning is as good as it gets (taken at the current pose of the rover). Therefore, if NAVLIB is still unable to find a path in that case, then it gives up, and NAV will declare a NoPath fault, which will stop the drive.

Note that NAVLIB will only return "no arc" if it has run out of all options, including retracing the way it came from, i.e., driving the previously selected arcs backward. To avoid undoing too much of the progress toward the goal that was accomplished before giving up, NAVLIB is only allowed to retreat up to a certain number of times (the current setting is 3 times), specified by a parameter. This limit is also a safety measure because the rover cannot image behind itself with the NavCams so driving backwards is performed using older map data (the rover will image the terrain in front of itself but not behind itself as it retreats) and we want to limit the amount of localization error that could accumulate since it would be dangerous for the rover to be too far off from where it thinks it is and potentially back up into a hazard it cannot perceive anew. ACE is still evaluated on the retreat path using the height map created during previous forward motions.

D. COMPUTATION

Making ENav work on the flight system required shifting much of the vision-related processing off the main computer onto a dedicated vision processor [14], [28]. Stereo image processing and much of the VO algorithm run on a secondary RAD750 and Virtex V FPGA, collectively known as the VCE, while ENav software runs on the primary RAD750 CPU called the RCE as part of the primary rover FSW.

ENav is only able to use a portion of the CPU time, since the RCE CPU also controls most of the subsystems on the Perseverance rover. There are over 100 FSW prioritized tasks running on the RCE using the real-time VxWorks operating system, some of which are always running and using up CPU time. As shown in Fig. 17, many maintenance tasks, including the Analog to Digital monitor (ADC), Remote Engineering Unit Manager (REUMGR), Motor Control Assembly (MCA), and 64-Hz Bus Controller (BCMGR_64), collectively use over 40% of available CPU time. When the motor (MOT) task is also active, the surface navigation (NAV) task that includes ENav is typically granted no more than 40% of the CPU. For example, the mean and median usage of the NAV task during the drive segment represented by Fig. 17 was just 32% of total CPU time (during those 1-s-slices when NAV was most active).

The Thinking While Driving framework described earlier means Perseverance can drive continuously without stopping, so long as ENav selects the next path step before the current one has completed. In Fig. 17, continuous driving can be found where the dark blue MOT task employs 10% or more CPU time without interruption over tens of seconds. Several issues can interrupt continuous driving. Perseverance (like Curiosity) only has eight motor controller boards due to resource constraints within the rover chassis, and thus can only actively control up to eight motors at once. As a result, whenever the arc curvature changes and the four corner wheels must be steered, the rover will stop driving its six drive motors to allow all four corner steering motors to work

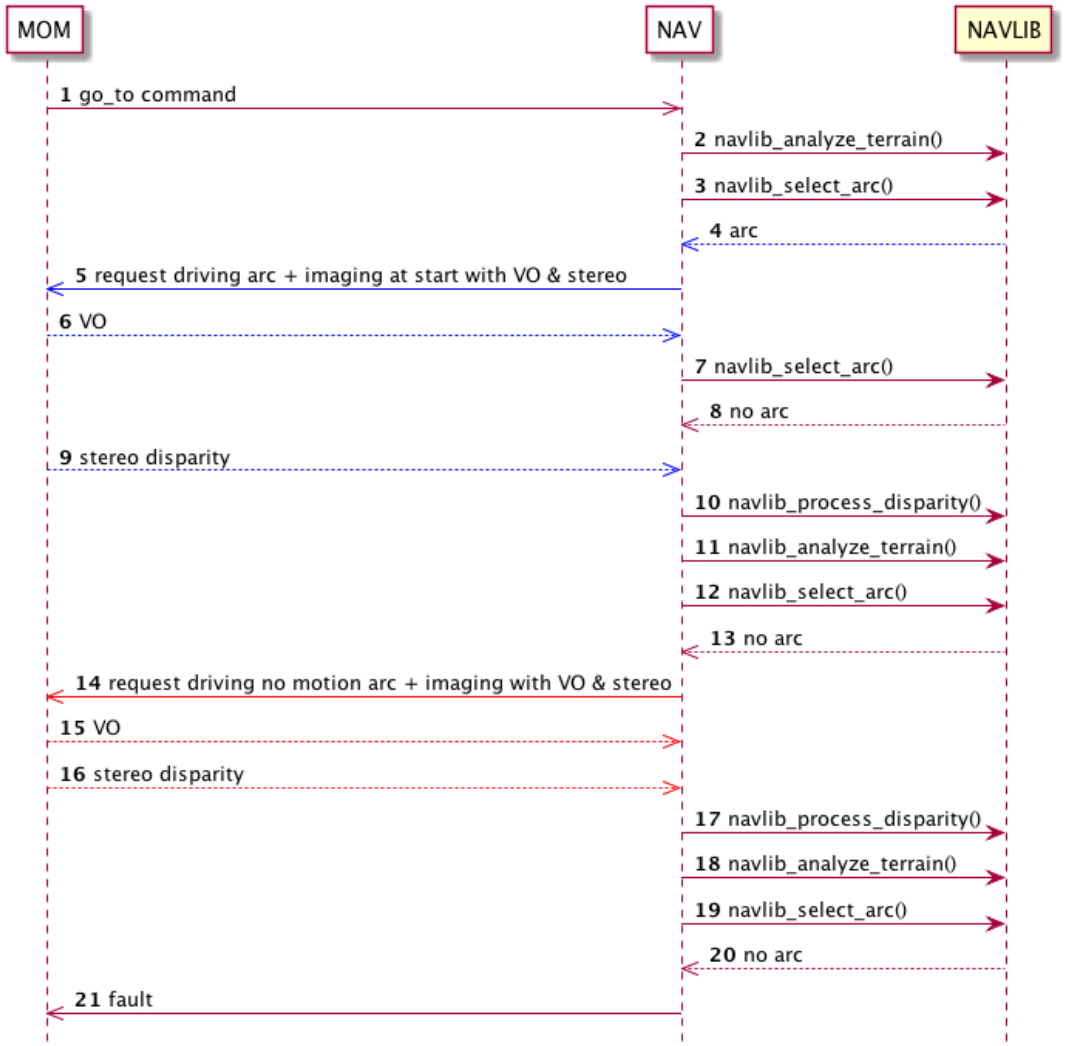


FIGURE 16. GoTo process in the off-nominal case (i.e., failure). Numbers 6 and 15 indicate forwarding of VO VCE results for NAV and ENav processing.

together. Drive pauses can also occur whenever any task delay results in the NAV task taking longer than expected to select the next path step. Although we have tried to configure ENav to minimize the chance of this occurring, there are still dozens of higher priority software tasks that might grab CPU time and force NAV to wait before it can select a path. Given the variability of CPU time allocation, the protocol between mobility FSW tasks was designed such that a drive will pause safely and gracefully without faulting, should any steps of the processing be unable to keep up with the physical drive motion.

Furthermore, our FSW tracks mobility data while driving in RAM buffers for later telemetering, including actuator/sensor/pose data at 8 Hz, terse step summary reports, and NAV/NAVLIB path-selection diagnostics. The mobility FSW pauses driving every 8 m to dump these RAM buffers to nonvolatile memory, driven by the amount of RAM allocated to them as well as ensuring most drive history is preserved should there be an unexpected CPU reset. This periodic dumping results in data management system (DMS) task processing that takes several seconds to complete, explaining the four widest and evenly spaced brown DMS bands in Fig. 17.

As currently configured, NAVLIB uses 18.3 MB of RAM memory, drawn from a total allocation of 28 MB in the FSW. The memory size is proportional to the size and squared resolution of the heightmap and costmap, the size of the tree of arcs used by the Local Planner, and the data products we choose to record, all of which are determined by parameter values. An additional 3.5 MB is used to import individual bands of 3-D stereo point cloud data into the heightmap.

IV. TESTING, V&V

Planetary robotic systems such as Perseverance have unique challenges in testing autonomy algorithms. First, it can never be tested in the real planetary environment until it lands there.

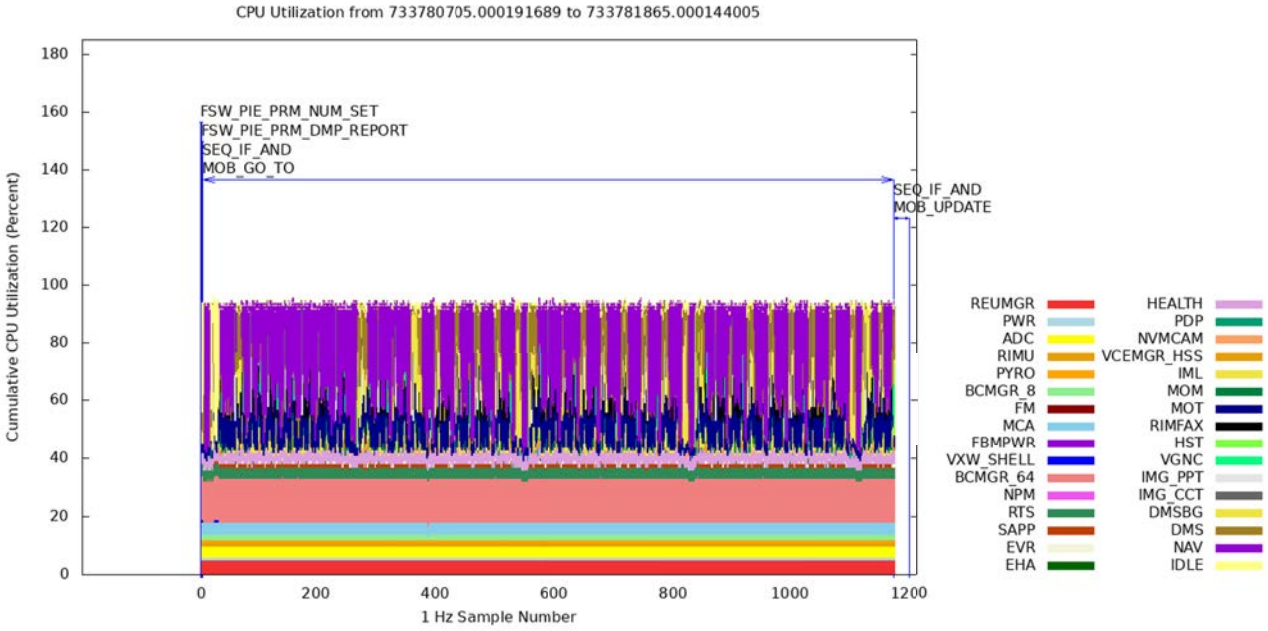


FIGURE 17. CPU Usage Plot generated during the first AVOID_ALL drive leg of the sol 753 drive (April 3, 2023), which covered 34 m. The plot shows the percentage of CPU used by individual VxWorks tasks, collected at 1-s intervals. The legend lists the names of all VxWorks tasks using at least 0.5% CPU throughout the 20 min represented in the plot. The task names in the legend are listed in bottom-to-top order: ADC is the yellow band on the bottom, and IDLE is the intermittent light yellow area on top [17]. The dark blue MOT task shows when the wheels are driving or steering, and the purple NAV task shows when ENav processing is active. While there is some yellow IDLE time, it is a small fraction overall. The brown DMS task is when our data management FSW is actively managing the creation of onboard data products in flash memory for later transmission to Earth.

Second, the flight system, including software and hardware, is highly complex and involves many subsystems that are not directly relevant to robotic functions, such as communication, thermal management, and data management. Although JPL did develop an engineering model of Perseverance, called the Vehicle System Testbed (VSTB), which faithfully replicates the flight hardware and software system, performing tests on VSTB involves highly complex and delicate procedures that take hours. Furthermore, due to scheduling priorities, the complete VSTB engineering model was not planned to be completed until relatively late in the development cycle. In fact, during the first 244 sols of surface operations, the flight vehicle ran a version of FSW with a mobility code that was frozen prior to VSTB coming online for mobility testing in JPL's Mars Yard. Perseverance uses a real-time Operating System, VxWorks, which is different from the main development environment, Linux, which further complicates our ability to perform tests efficiently. And finally, some of the requirements (drive success rate and drive efficiency) required statistical assessment, but it is practically impossible to run a statistically significant number of tests with VSTB hardware-in-the-loop experiments.

For these reasons, a number of different simulation and hardware testbeds with different fidelity are used for development and testing, as listed in Table 2. Roughly speaking, we used lower fidelity and more accessible testbeds during the early stage of development, such as the ROS-based ENav Sim [30] and the Scarecrow test vehicle, and moved on to higher fidelity testbeds such as the work station test set (WSTS) and surface system development environment (SSDEV) for formal verification and validation (V&V). WSTS runs a complete set of FSW on VxWorks in a virtual machine but lacks the ability to produce simulated camera images in the loop. SSDEV involves only the subset of FSW modules that are relevant to mobility and arm, and runs on Linux, and it emulates VxWorks' parallel task environment.

A limited number of hardware experiments at JPL's Mars Yard were complemented by Monte–Carlo simulations on ENav Sim and SSDEV for statistical evaluation of the performance metrics. More details of each test environment will be described in the remainder of this section.

A. SIMULATION

1) SIMULATION FRAMEWORKS

The following simulation frameworks were used during development, V&V, and operations.

a) ENav Sim

The ENav Sim, a low-fidelity and lightweight simulator that employed ROS, was developed to support early stage algorithm development. The emphasis was placed on the ability to run on regular Linux machines so that developers could easily test the code locally. It runs only the ENav library FSW, which was wrapped as a ROS module and interfaced

TABLE 2. Test Environments used during development and validation of ENav software. Type indicates whether the test environment is a software simulation, a HILS, or a physical vehicle moving through real terrain, SW platform indicates the operating system/framework for the test environment, terrain indicates what kinds of terrains could be easily tested (none, arbitrary simulated, and real), and FSW coverage indicates which of the 140+ FSW modules are being exercised, at least in part. SSDEV in the SW platform stands for surface system development environment, a FSW simulator explained in Section IV-A1.

Test Environment	Type	SW Platform	Terrain	FSW Coverage
ENav Sim	Sim	Linux/ROS	Sim	ENav library only
Rover Simulator (RSim)	Sim	Linux/SSDEV	Sim & Real	Mobility/Arm modules
ENav Unit tests	Sim	Linux/GoogleTest	None	Only NAVLIB module
NAV Unit tests	Sim	Linux	None	14 modules, 20 stubs
Work Station Test Set (WSTS)	Sim	VxWorks	None	Complete
Flight Software Testbed (FSWTB)	HILS	VxWorks	None	Complete
Surface Simulation (SSim)	Sim	Linux	Sim & Real	Complete with stubs
Scarecrow Vehicle	Real	Linux/SSDEV	Real	Mobility/Arm modules
Vehicle System Test Bed (VSTB)	Real	VxWorks	Real	Complete

to a 2-D kinematic simulator. One notable feature of this simulator is the ability to create synthetic camera images for a given digital elevation model (DEM) of the terrain, superimposed with a randomly generated texture. This capability is provided by the HyperDrive GUI, the visual interface of the rover operation software [40]. HyperDrive GUI was interfaced with ENav Sim through a wrapper called HDSim. This allowed us to test the end-to-end ENav capabilities, including perception and motion planning. More details on ENav Sim are described in [30].

b) Rover Simulator

Rover simulator (RSim) is a higher fidelity simulation environment consisting of the SSDEV and HDSim. SSDEV is a multithreaded emulation of all the FSW modules necessary for mobility, arm, sampling, and caching surface operations, with stubs for other functions and commands [4]. SSDEV is flexible enough to serve both as a simulation framework (when used in conjunction with a terrain simulator and image renderer) and also as a command interface for hardware testbeds like the Scarecrow rover. As in the ENav Sim, HDSim provided the ability to produce synthetic camera images during simulated drives such that the end-to-end ENav capabilities could be tested.

c) Work Station Test Set

The WSTS environment is a complete Linux build of all FSW running in the multithreaded VxSim framework, together with a separate emulation layer of all vehicle hardware. This framework does support limited use of pregenerated images, but does not provide full terrain simulation support.

d) FSW Testbed

The FSW testbed (FSWTB) is a hardware-in-the-loop simulator (HILS) of the Perseverance rover. It is able to run the real FSW on flight-equivalent avionics hardware. Software stubs are used to model the behavior of actuators and sensors not physically available. The venue also supports limited hardware testing, such as controlling a single mobility actuator or responding to telemetry from a physical flight-like IMU. We used FSWTB to evaluate the timing of each NAVLIB function: process disparity, analyze terrain, select arc, and evaluate arc. These early timing performance tests were critical to identify where computational bottlenecks were and allow the development team to make modifications. The timing results informed the design of module interfaces, especially those relying on synchronous computation and those performed while the vehicle was in motion.

e) Surface Simulation

Surface Simulation (SSim) is a single-threaded emulation of all of the FSW modules necessary for surface operations, with stubs for some functions and commands [32]. Unlike SSDEV, SSim only interfaces with the rover sequencing and visualization program (RSVP) tool, which simulates rover motion on arbitrary terrain meshes. The SSim environment was only used occasionally during FSW development; it primarily supports daily tactical operations [33].

2) TERRAINS

ENav Sim and RSim simulations required a terrain DEM to run. In particular, the Monte–Carlo simulation employed three standard types of terrains, as shown in Fig. 18.

1) *Uniform Slope Terrain:* Synthesized terrains with a uniform slope at a given rock abundance level, represented by the CFA [11], for evaluating performance against specific slope-CFA navigation requirements.
2) *Undulating Terrain:* Synthesized terrains with randomized, local undulation providing more challenging combinations of rock hazards and ground undulation as a form of "stress-testing."
3) *Jezero Terrain:* A set of terrains using real Mars orbital maps of Jezero Crater, along with a distribution of rocks matching the CFA measured from orbital imagery, was

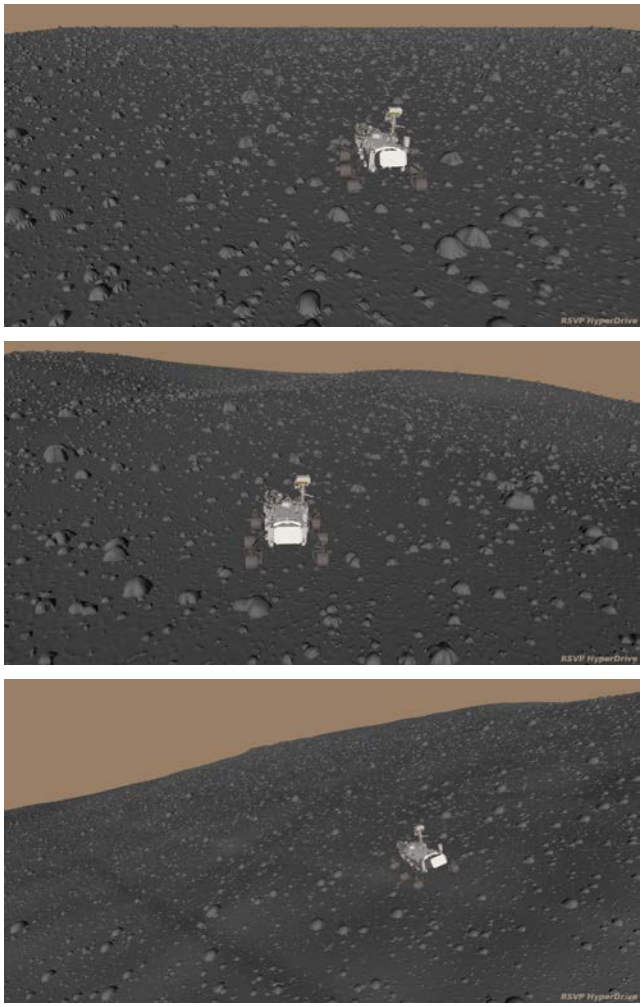


FIGURE 18. Examples of the three terrain types: 1) uniform slope, 2) randomized undulation, and 3) Jezero Orbital map.

used to investigate how the system navigated through realistic topographical features.

3) MONTE CARLO SIMULATION

An automated Monte–Carlo simulation framework was built on top of the ENav Sim and RSim to assess the navigation software performance and test new navigation parameters. This ability turned out to be important because any minor changes in algorithms can be statistically evaluated to identify if the changes positively contributed to the overall performance. Each Monte–Carlo simulation produced over 500 GB of simulation data products, which needed to be distilled into key metrics for useful analysis and sensitivity studies. A framework was developed that automatically filtered this data and auto-generated over 100 pages of graphs, statistics, and assessments of the current FSW performance.

4) KEY METRICS AND RESULTS

The metrics evaluated by our Monte–Carlo simulations fell into three primary categories: requirement-driven, vehicle safety, and diagnostic.

a) Requirement-Driven Metrics

The first category of the metrics directly corresponds to the requirements imposed on ENav. The two most important metrics of this kind are the drive success rate and the path inefficiency. The drive success rate is the ratio of the simulation runs in which the rover successfully arrived at a goal disk, set at 80 m away from the start. The path inefficiency measures the tortuosity of a path and is given by

$$\frac{\text{Wheel odometry}}{\text{Straight line distance}} - 1.$$

For example, if the rover's driving distance, measured by the wheel odometer, is 120 m to get to the goal 80 m away, the path inefficiency is 50%.

Separate sets of ENav requirements were defined for two terrain classes—"benign" and "complex" terrains. A terrain is considered "benign" if the slope is 15° or less and the rock abundance is 7% CFA or less. The simulations were run over a wide parameter space of slopes and rock abundance measured in CFA, uniformly distributed over the parameter ranges. However, the distribution of slope and rock abundance is, of course, not uniform on Mars. Therefore, we adopted a weighted sampling approach. We identified the approximate probability distribution of the slope and rock abundance CFA, shown in Table 3, from the long-range drive simulations on the landing site, Jezero Crater, using a high-resolution satellite imagery [22]. This probability distribution is used as the weight for computing the weighted average out of the uniformly sampled Monte–Carlo simulations. Each requirement metric was first calculated for the CFA-slope terrain cluster sims, then combined with all the other benign/complex values according to the weighting scheme to produce the expected value of the metric while driving through Jezero Crater.

The requirement-driven metrics were evaluated frequently throughout FSW development. The contour maps in Fig. 19 show the drive success rate and path inefficiency over the tested ranges of slope and rock abundance CFA on undulating terrains. As expected, the drive success rate decreases while the path inefficiency increases on the terrains with higher slopes and rock abundance. Table 4 shows the weighted averages of the two metrics on the version of ENav that was included in Perseverance's FSW at the landing on Mars. All the requirements were satisfied for the uniform slope and undulating terrains. Since Jezero terrains are not separated into benign or complex, it was not used for checking requirement satisfaction.

b) Vehicle Safety Metrics

The second category of metrics was related to vehicle safety, including identifying software issues. The safety guarantee of ENav is sourced from its kinematic collisions checking algorithm, ACE (Section II-G; [25]), which is designed to conservatively estimate key safety metrics, including the following list.

TABLE 3. Frequency distribution of slope and rock density, measured by the CFA, that were used to evaluate the requirement-driven metrics. The distribution is an approximation of that of Jezero Crater on Mars. Benign terrain was considered to be terrains with 7% CFA and ≤15° slopes. Complex terrain was all other terrains, and for the purpose of ENav requirements, this meant CFA up to 15% and slopes up to 20°.

Slope	CFA			
	7%	10%	12%	15%
20 deg	2%	1.5%	1%	0.5%
15 deg	5%	4%	1%	0.5%
10 deg	20%	10%	1%	0.5%
5 deg	40%	10%	2%	1%

TABLE 4. Requirement-driven metrics evaluated by Monte–Carlo simulations over three terrain types defined in Section IV-A2. These results are based on the version of ENav that was included in the first delivery to the FSW at the landing of Perseverance on Mars. Note that the Jezero terrains were not separated into benign or complex (cmplx) because the slope varied across each terrain.

Terrain	Success Rate		Path Inefficiency	
	Benign	Complex	Benign	Complex
Uniform slope	100%	91.6%	2.4%	12.8%
Undulating	98.5%	75.7%	4.4%	17.9%
Jezero	84.7%		14.1%	
Requirement	>90%	>75%	<15%	<35%

1) *Belly Pan Clearance:* The minimum distance between the ground and the rover's belly pan (the bottom panel of the rover chassis).
2) *Wheel Drop:* The maximum terrain height difference within each of the ACE wheelboxes. This conservatively bounds the worst case "drop" of a wheel.
3) *Rover Attitude:* The rover's roll, pitch, and overall tilt angle against the coordinate frame with its z-axis aligned with the gravity vector.
4) *Suspension Angles:* The joint angles (referred to as "rocker" and "bogie" angles) of the rover's passively articulated suspension system.

The conservatism of ACE's onboard evaluations of these metrics was tested by the Monte–Carlo simulations. The "true values" of these metrics were kinematically computed during the simulations at each time step and compared against the onboard evaluations. Fig. 20 shows the distribution of the simulated true values of belly pan clearance, wheel drop, and tilt angle, sampled at 8 Hz in all runs of a Monte–Carlo simulation. The black vertical lines in each plot show the minimum or maximum accepted values by ENav, meaning that ENav only selects a path if the ACE-estimated safety metrics are within these thresholds. The red vertical line represents the worst case "truth" value from the simulation.

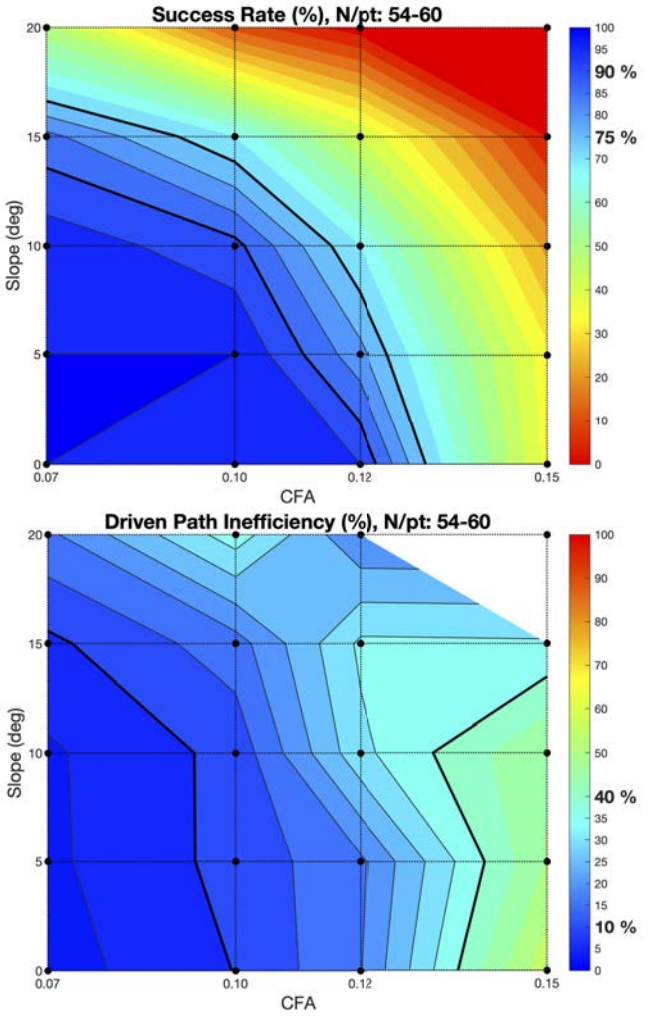


FIGURE 19. Success rate and driven path inefficiency plots generated by Monte–Carlo analysis.

The red line always lies on the safe side of the black line if ACE is conservative. This was not the case for wheel drop on the uniform slope and Jezero terrains. These were due to the limitation in the fidelity of the kinematic simulator that computed the "true values." For example, the simulator sometimes exhibited "chattering" behavior where the wheels instantaneously settled on the top of a rock in one timestep and then fell to the ground in the next, which is unrealistic. All such cases were manually investigated and verified by the test engineer running simulations. The bimodality of belly pan clearance in the uniform-slope and undulated terrains was due to an artifact in terrain generation where the starting position of the rover is kept free of rocks, thus a ∼66-cm clearance is measured for the first several meters of every drive.

Through repeated Monte–Carlo simulations with varying environmental inputs, we were able to validate that ENav's assessment of terrain hazards was always conservative and there were never any physical violations of the safety thresholds. This built confidence in the algorithm in a way that the limited hardware testing was unable to provide. In addition

to confirming the physical safety of the rover during driving, the simulations also exercised FSW logic many times using different inputs, exposing uncommon interface issues and nuances that would have otherwise been untestable manually. When reporting results from a Monte–Carlo, the specific metrics reported became a critical tool to help the developers focus on FSW issues that could improve the design.

c) Diagnostic Metrics

The third category of metrics was intended to provide various diagnostic information that was helpful for gaining deeper insights into errors and improving algorithm performance during FSW development. Examples include the percentage breakdown of the reasons paths were not selected [Fig. 21(a)], the number of ACE evaluations performed in each planning cycle [shown in the text box in the pie charts in Fig. 21(a)], and the visualization of all the paths [Fig. 21(b)]. Such diagnostic information was important when analyzing the sensitivity of FSW and parameter changes on ENav performance. For example, understanding that hazard avoidance was more sensitive to wheel drops than belly pan clearance or that unknown terrain was a more common issue on higher tilts provided crucial insights for improving ENav performance. Metrics in this category related to computation, such as the average number of ACE evaluations, were also important. Although the simulation was not able to fully model the timing and computational constraints of the real rover avionics, by measuring the number of times a computationally intense part of the algorithm was run, we were able to compare performance between simulations and extrapolate how these differences might relate to actual drive speed on hardware.

B. HARDWARE TESTS

Testing on hardware served multiple purposes during the different phases of the project. During the development phase, hardware tests supported the development of new navigation features and consisted of basic drive scenarios across varying terrain configurations. The goal of development testing on hardware was to augment our simulation tests in a real Mars-like environment with a physical rover system. The tests were performed in JPL's Mars Yard using the Scarecrow test vehicle, a stripped-down version of the rover that only represents the mobility system. Later on, we transitioned to V&V testing, where repeatable test cases were used to exercise specific system capabilities in flight-like scenarios. V&V testing typically used the higher fidelity VSTB hardware testbed as well, although the preparations for V&V testing were performed in simulation and Scarecrow to save time and cost.

1) VEHICLES

a) Scarecrow

Scarecrow (Fig. 22) is a mobility performance test vehicle for the Curiosity and Perseverance Mars rovers, built in 2007 [12]. It has the identical rocker-and-bogie suspension and wheels to the actual rovers, while its mass is approximately one-third of the flight vehicles, making its weight approximately the same on Earth as that of the flight rovers in Martian gravity. Being a "mass model" of the flight rover made it a helpful testbed for ENav algorithmic testing on high tilt and other more extreme terrain conditions. Scarecrow is battery-powered and uses WiFi to communicate with the test team, so no tether is needed. As the name suggests, the original Scarecrow did not have a "brain"—i.e., a flight-like onboard computer. For testing ENav, which obviously needed a brain, we strapped a Linux laptop computer that runs SSDEV onto its back. A raised mast with NavCams was also added to the test vehicle to provide a stereo perception system similar to the one on Perseverance. Scarecrow's NavCams were commercial-off-the-shelf units based on the same focal plane (CMOSIS CMV-20000) as the flight cameras [19]. To provide adequate image fidelity, custom-built (nonflight) lenses were made using the same optical prescription as the flight lenses.

b) Vehicle System Testbed

For formal V&V, a higher fidelity vehicle, the VSTB shown in Fig. 23 was used. This vehicle represents the "twin" of Perseverance in nearly all respects—actuators, avionics, software, mechanical mobility, and arm hardware. As such, it has a mass approximately equal to Perseverance, and therefore, on Earth, the ground pressure applied on the mobility system is 2.64 times greater due to the difference in surface gravity. This excess force on the system meant the VSTB could not operate exactly like the flight rover in challenging mobility situations, like driving over large rocks or on high slopes. In addition, the VSTB requires a thick tether for power and communication to ground support equipment (GSE). The tether is long enough to enable driving to most of the Mars Yard, but it does require a test operator to manage the tether, out of view behind the rover. And during the hottest parts of the summer, cool air was piped into VSTB using a flexible duct to an air conditioner alongside the power/comm tether. Despite these mechanical limitations, the VSTB was still critical for formal V&V testing of the navigation software because no other testbed included the fully integrated avionics, mobility systems, and the entire stack of FSW.

2) ENVIRONMENT—JPL MARS YARD

All ENav hardware-based tests were performed in the JPL Mars Yard [18], shown in Fig. 24. The Mars Yard is a 50×40 m^2-outdoor facility with terrain features representative of the Martian surface. There are slopes at multiple grades with different materials (flagstones, regolith, sand) used for mechanical stability testing and exercising navigation at extreme tilts. The majority of the yard is an open area that can be configured with craters or rock fields to explore navigation in varying levels of terrain complexity. The size of the VSTB, GSE, and tethers, and the limited availability of

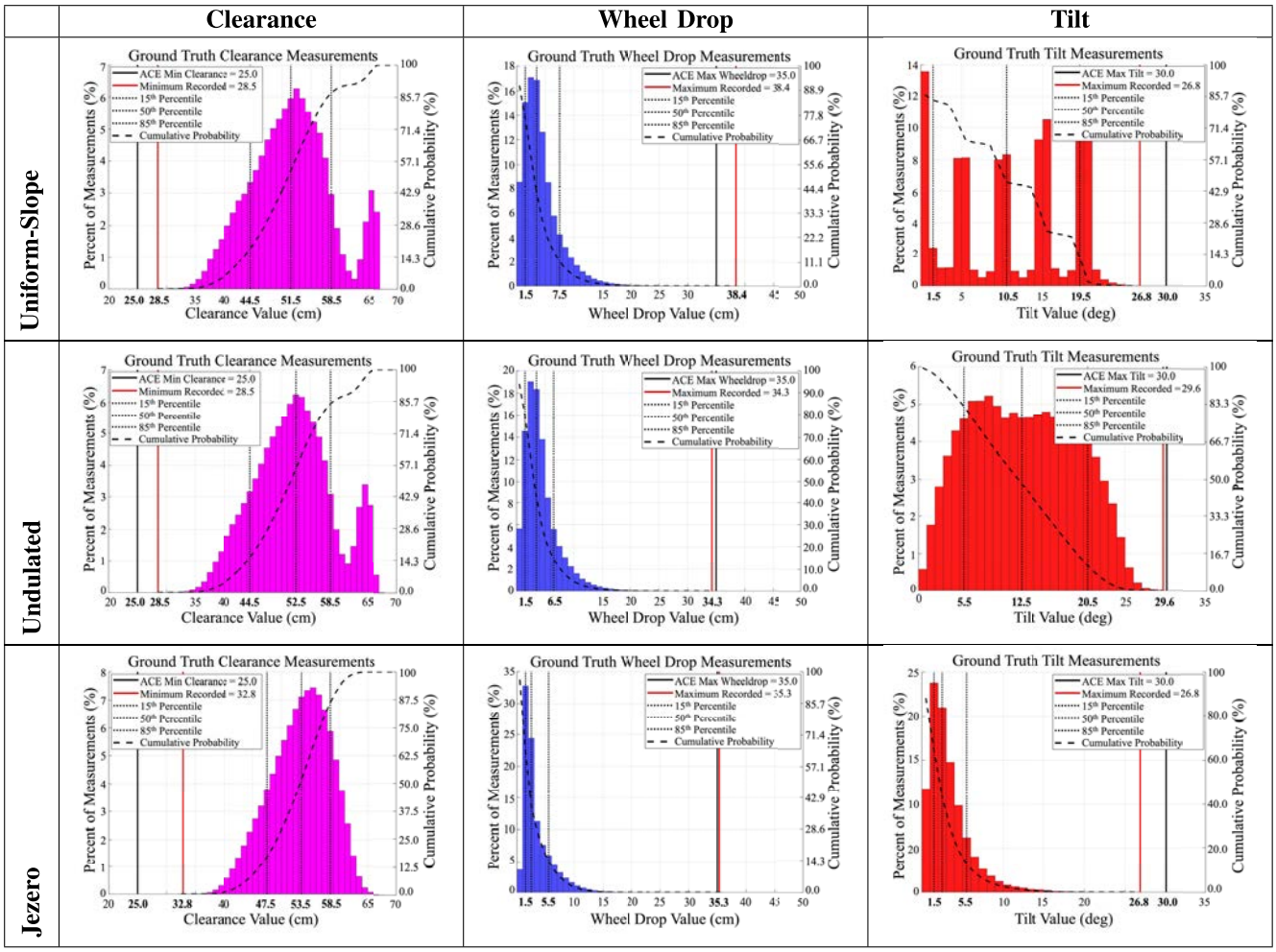


FIGURE 20. Examples of statistical plots generated by Monte–Carlo analysis, for three terrain types.

the testbed made it logistically impractical to take the VSTB to off-site locations for field tests.

3) TEST PROCEDURES

The test procedure with Scarecrow during the development phase was somewhat informal. Because many aspects of the flight system were still being developed, the team did not have the support of a formal ground system for command authoring and data processing. Instead, we used basic SSDEV command-line interfaces for commanding and tools that had been developed for data processing and analysis. The software under test during this time was primarily focused on ENav and associated FSW, which was exercised via the MOB_GO_TO command. Different path-selection modes (UNGUARDED, GUARDED, and AVOID_ALL) were tested. Complex Mars Yard configurations with slopes, rocks, and terrain undulations were used to challenge the navigation system. A typical day of testing included several approximately 40-m autonomous drives back and forth across the Mars Yard. During the peak of testing, developers would review the results of the drives and make updates to the software on-site so the team could test new features on the following day.

Working with the VSTB to perform the formal V&V was more challenging. The software version was frozen as a flight release candidate, and the objective of the testing was to prove that the candidate navigation software could safely and effectively handle specific flight-like scenarios. The parameter settings and system setup of the tests were highly controlled through command sequences that were included in formal test procedure documents, intended to mimic sequences as they would be used in flight. The test scenarios and Mars Yard configurations were each designed to evoke a specific response from the flight system. For example, the rover may be commanded to autonomously drive to a waypoint on the other side of an impassible ‘‘wall’’ of rocks to verify a failure response. Or hazards could be assembled as a corridor to confirm the vehicle’s ability to navigate through a narrow path of specific dimensions. Often, Scarecrow would first be used to refine the design of the scenario to ensure repeatable results before the team committed to running the test on the VSTB. This progression was done because VSTB required more testbed

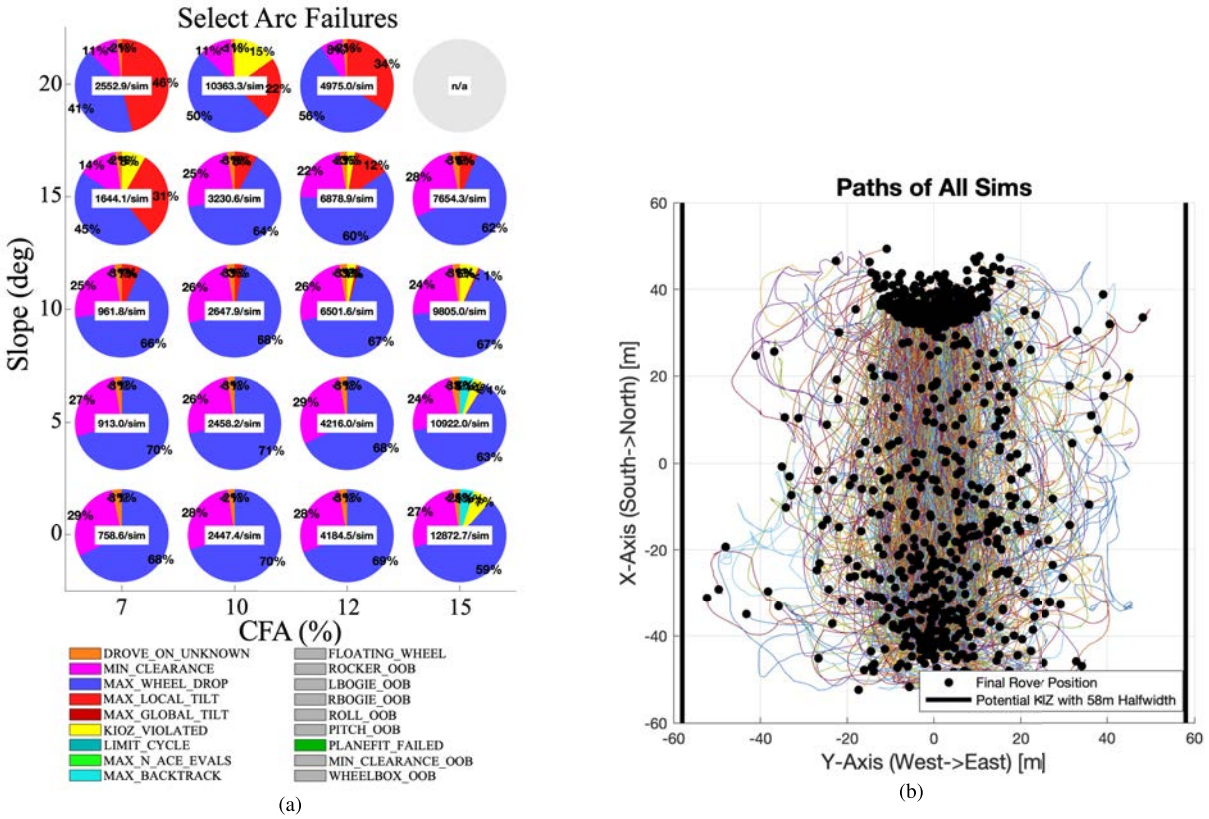


FIGURE 21. Examples of diagnostic metrics produced by the Monte–Carlo simulations. (a) Distribution of the reasons paths were not selected during ENav's path planning among successful drives. The average number of arcs that were evaluated as intraversable is shown in the text box in each pie chart. The result at 15% CFA and 20° slope is not shown because all runs failed to arrive at the goal. (b) Visualization of all paths in a Monte–Carlo simulation and the final rover positions. The rover started at (0, −40) and the goal was a disk with a 10-m radius centered at (0, 40). Note that the majority of the paths ended in the goal disk.

personnel and significantly more time to configure its flight system for each scenario than when running on Scarecrow. The data collection for V&V testing was more comprehensive as well and was typically presented at a format Data Review meeting as part of the V&V process. All images and state data from the rover were archived, and external measurements of pose using a total station were sometimes taken to help validate the pose estimation algorithm in use.

4) REPRESENTATIVE TEST RESULTS

A substantial number of tests were conducted in the Mars Yard over several years, initially to inform the development, and later as a part of V&V. A comprehensive description of all tests in the V&V program is not possible in this publication due to the quantity of tests performed and their varied purposes and procedures. Rather, in this section, we present two specific tests that are representative of the two ends of the spectrum. Movies summarizing these tests are included in the supplemental material for this publication.

a) General Obstacle Avoidance Tests

Fig. 25-top is a snapshot from a test performed on July 17, 2018. The Scarecrow rover with ENav started from one end of the Mars Yard and was commanded to autonomously drive to the other end, which is approximately 40 m away. Rocks were placed randomly in the Yard. In this particular test, the rover successfully avoided the obstacles and arrived at the goal. The class of tests represented by this one is meant to be a "stress test." The tests were performed with varying levels of rock density, and the outcome of the tests informed the development team as to potential vulnerabilities of the algorithm.

b) Specific Purpose Tests

The test shown in Fig. 25-bottom, conducted on February 7, 2018, represents a class of tests that targets more specific behaviors. In this case, a "wall" of rocks was built to block the direct line from the start to the goal, forcing the rover to go up the slope on the north side of the Mars Yard. The rover successfully traversed the ∼15° slope, avoided rocks, and arrived at the goal disk. Many tests of this kind were performed to test specific behaviors of automated mobility, such as going over steps, terrain undulations, and rough surfaces.

V. RESULTS ON MARS

This section describes some of the constraints imposed by Mars operations, and details the history and results of autonomously driving Perseverance on Mars with ENav.

FIGURE 22. Scarecrow rover. It has a copy of the flight mobility hardware, and its weight on Earth is the same as Perseverance's weight on Mars.

FIGURE 23. VSTB rover, which has mostly identical hardware (including avionics and mobility) and software to the Perseverance rover.

FIGURE 24. JPL Mars Yard, in which all the ENav hardware-based tests were performed.

FIGURE 25. Representative snapshots of Mars Yard tests with the Scarecrow rover. Full movies of these tests are included in the supplemental material.

A. OPERATIONAL CONSTRAINTS

There are several considerations that influence the amount of time that Perseverance can drive each sol.

General Resources: There must be sufficient power, time, and flash storage for any generated data before a drive can begin.

Good Lighting: Perseverance's NavCams (EECAMs [19]) have a wide field of view (73° vertical × 96° horizontal) and they work best if the Sun is kept out of their field of view. If it were to creep in, multiple artifacts could result when incoming sunlight is not blocked by the external lens hood/baffle (pixel saturation, blooming, ''ghost'' images due to light scattering within the lens assembly). To account for this, the NavCams are typically pointed 33° below the rover's *XY* plane (i.e., the horizon when the rover is level). Even so, the Sun will eventually enter the FOV. Therefore, we impose an operational constraint precluding driving when the Sun is low enough that it might become visible in the images. This time of day varies throughout the year, but is typically around 15:30 Local Mars Solar Time, and has been the latest time the rover can drive using autonomy that involves imaging.

Heating: Since Perseverance's wheel motor temperatures often drop below their operational range during the Martian night, any relatively early drive must allow time for the motors to be heated before the drive can begin. In practical terms, heating is typically scheduled to begin after the primary sequence controlling all rover activities has begun running. That typically happens between 09:00 and 10:00 Local Mars Solar Time, meaning the drive cannot begin until some time after that (the duration varies with the temperature, but often requires 30–60 min). This sets the earliest time the rover can drive.

Activity Conflicts: There are multiple activities performed by Perseverance that cannot be run concurrently while the rover is driving. Activities like Helicopter communication and UHF communication with orbiting relay satellites are typically planned one or many sols in advance, often during the same time the rover is available for driving. Any such conflicting activity will also shorten the duration available for driving.

Final Heading: The HGA needs to have an unobstructed view of Earth to enable reliable X-band communication. Therefore, some of the drive time must be reserved to accommodate having the rover TIP at the end of a drive to ensure nothing on the rover's deck will occlude HGA pointing to Earth during the next planned communication pass. In the case of AutoNav drives, there must be time for the autonomous turn for comm behavior (see Section II-I) to find a good location and turn to a desirable heading. Rover Planners typically allocate 3 min + worst case turn time (1°/s) for this in benign terrain, and 10 min in more complex obstacle-laden terrain.

Heading Uncertainty: Perseverance needs to be able to point the HGA back to Earth with sufficient accuracy, typically better than 5°. But over time, pointing uncertainty grows as the IMU propagates its attitude estimate while driving. Although Perseverance is now able to autonomously recalibrate its attitude by tracking the Sun [36], that was not the case during the first few hundred sols. Another constraint on drive distance early in the mission had been on the accumulated time spent steering and turning the wheels since the most recent attitude estimate recalibration, which was initially limited to less than 2 h. Now, such recalibration can be performed during AutoNav drives, so long as sufficient time is available, and its successful completion will reset the accumulated time back to 0.

Fig. 26 illustrates the drive modes and distances covered by every drive during the first 1312 sols of the mission. The daily drive distance was often limited by the desire to reach a specific location, or to avoid conflicting with another activity, rather than being primarily limited by the Sun's elevation.

B. JOURNEY ON MARS (ENav DRIVING JOURNAL THROUGH SOL 1312)

Since landing day, ENav has been an excellent driver of Perseverance with no major incidents. This section provides a region-by-region record of Perseverance's 32.1-km journey on Mars up to sol 1312, which corresponds to October 28, 2024, on Earth. In the discussions below, please refer to Fig. 2 for drive routes and names of particular surface regions, and to Fig. 26 to see how much AutoNav was employed on any given sol.

1) CHECKOUT AND COMMISSIONING

After landing at the Octavia E Butler Landing site, the operations team prioritized the checkout of Perseverance's many science and engineering subsystems, including the Ingenuity helicopter. That meant that we did not need to drive very far or quickly at first. Mars missions typically employ an initial commissioning phase for each high-level capability so that any unanticipated problems can be dealt with safely. For instance, during the first 64 sols on Mars, we used only the "human-directed" driving mode, without any imaging-assisted autonomy, while multiple other subsystems were being commissioned (see Fig. 26). VO was first used on sol 65 (April 26, 2021), and ENav processing was first demonstrated on sols 91 and 99 (May 23 and 31, 2021) in the "mapping" mode only: collecting images and generating a world model, but not using it to guide any driving decisions. This allowed us to confirm that the phasing of the many map-building subsystems was all in agreement, including the geometric camera lens models, the FPGA image rectification and stereo disparity generation, and the population of the resulting point cloud into the world map.

Next, we tested the guarded driving mode on sol 107. As explained in Section III-B, in that mode the rover uses ENav to model the world around it, but if a hazard should be found in the path, it will halt the drive with a NO_PATH mobility fault, rather than attempt to drive around it. That drive successfully demonstrated that guarded mode can stop the rover from driving into a perceived hazard (see Fig. 28).

The first demonstration of the full AutoNav system creating a world model and using the AVOID_ALL path-selection mode (Section III-B) occurred on sol 113 (June 14, 2021). That drive was over completely benign terrain, and worked well. For the next checkout drive, we looked for relatively benign terrain with a single (not too large) obstacle to confirm that the path-selection algorithm was working correctly. On sol 122, we found a good candidate region, drove toward a small obstacle, and successfully avoided it (see Fig. 29).

2) PRIME MISSION

Once the checkout had completed, ENav became available for regular use. The science team led us south and then west around the SE corner of the Seitah depression. We knew we were off to a great start when, over the next few months, AutoNav successfully performed multiple drives taking Perseverance onto the crater floor in the southern Seitah region around sol 204, with several drives breaking Opportunity's previous record for the longest single-sol AutoNav drive of 109 m. More detail on this initial phase of the mission is documented in [31]. Little driving was done for the next few months due to solar conjunction blocking communication and a desire to collect samples.

By sol 340, we were on our way again, with ENav taking us back to the Octavia E Butler landing site much more quickly, often driving over 200 m in a single sol (e.g., sols 340, 341, 351, 353, 355, and 360). It had taken months to reach the southern edge of Seitah, but only a few weeks to make the return trip. In all, we spent 13 months driving the first 5 km inside Jezero Crater.

Now having explored the landing site and southern part of Seitah, the science team was keen to drive toward the Delta

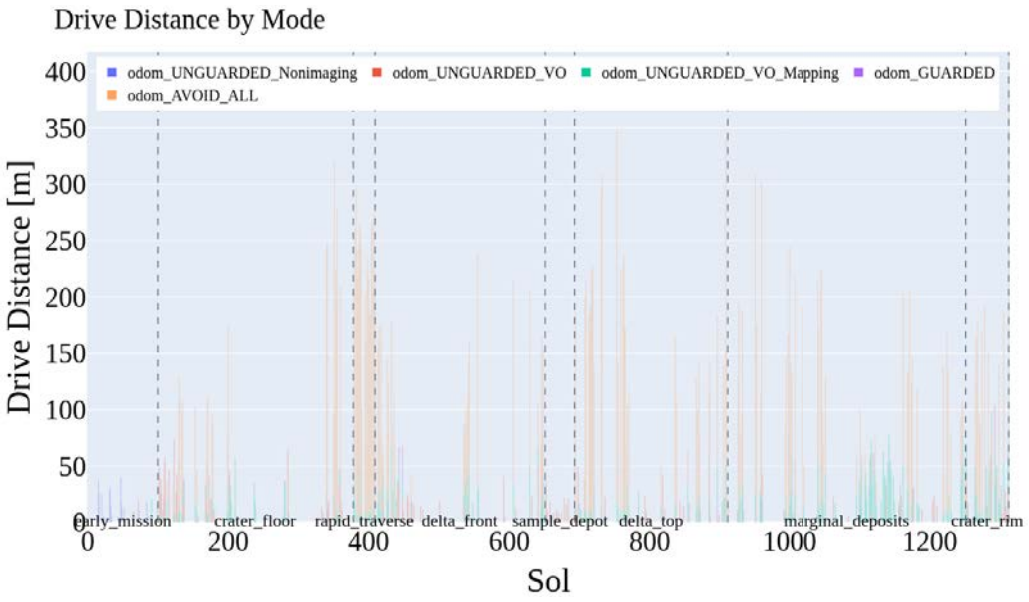


FIGURE 26. **Daily odometry through sol 1312 (October 28, 2024). Fig. 27 shows a cumulative plot of this data.**

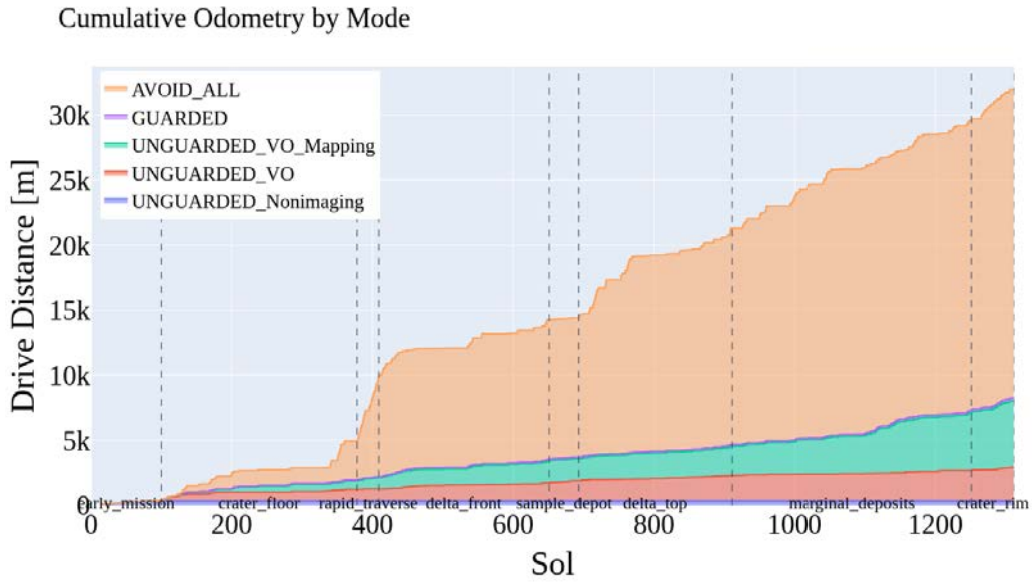


FIGURE 27. **Accumulation of daily odometry (shown in Fig. 26) through sol 1312 (October 28, 2024). The rightmost column corresponds to the data presented in Table 5.**

Fan inside the crater, located north of the Three Forks label in Fig. 2. They were so keen, in fact, that they were willing to forego science activities and just drive, drive, drive toward the delta. That was called the rapid traverse period, and from sol 379 to 409 (March 14 to April 14, 2022), ENav completed the next 5 km of driving in just 24 drive sols, averaging 201 m per sol. 95% of all driving that month was performed using AutoNav, an unprecedented level of autonomous driving [26]. The path from the landing site to the Three Forks area is labeled ‘‘Rapid Traverse 1’’ in Fig. 2. In all, we spent one month driving the second 5 km inside Jezero Crater.

Next, the science team explored the area southwest of the Three Forks location [33], and finished the first Martian Year back at Three Forks with a rich set of samples stored in tubes onboard the rover. In consultation with the Mars Sample Return project, a plan was devised to deposit 10 of these tubes onto the surface at the Three Forks area, which was found to be highly flat and conducive to later rendezvous with a future Mars Sample Return lander [35]. This activity resulted in the creation and documentation of the Three Forks Sample Depot, accomplishing one of the primary mission milestones in no small part thanks to ENav for having enabled long-

TABLE 5. Breakdown of Perseverance's total drive distance by available driving modes as of sol 1312 (October 28, 2024). AutoNav uses AVOID_ALL path-selection mode with ENav mapping and VO to analyze the terrain and select the next step while avoiding hazards and KOZs, Guarded uses ENav in GUARDED mode and VO to analyze the terrain but stops if a hazard or KOZ is detected rather than drive around it, Mapping runs UNGUARDED drives with ENav mapping and VO enabled to analyze the terrain passively, but does not act on its map (useful for data collection and seeding knowledge of the terrain under the rover), Nonimaging runs UNGUARDED drives without any vision-enabled autonomy (no ENav or VO), VO uses UNGUARDED path_mode and does not run ENav but does collect images and use them to relocalize the rover pose after each step. Distance is the idealized commanded distance covered by the center of the rover, which models no motion during turns-in-place (since the center stays in place). Drive rates are calculated by dividing the commanded distance driven by the duration of the MOB_GO_TO or primitive command that accomplished the drive (note this does not include the surrounding setup and cleanup command durations). The AVOID_KOZ path-selection mode has not been used in flight as of sol 1312.

Mode	ENav Use	Total Distance (m)	Duration (hours)	Rate (m/hour)	Percent of Total	Sol with Max Distance	Max Distance (m)
AutoNav	evaluate / select path	23,746.00	259.05	91.67	74.03	753	331
Guarded	evaluate / check safety	235.39	2.29	102.81	0.73	1292	39
Mapping	evaluate	5,105.54	51.14	99.84	15.92	1247	78
Non-Imaging	N/A	316.75	2.92	108.41	0.99	47	40
VO	N/A	2,651.36	26.96	98.33	8.27	1288	77
Totals	N/A	32,077.67	342.36	93.70	100.00	753	347

FIGURE 28. First guarded drive on Mars occurred on sol 107 (June 8, 2021), and successfully stopped before driving into the rock pile in front of the rover. This postdrive front HazCam view shows the rock pile is still well ahead of the rover's stopping point.

FIGURE 29. Second AutoNav checkout drive on Mars occurred on sol 122 (June 24, 2021), and was the first one to encounter (and successfully avoid) an obstacle. This postdrive NavCam view shows the rover tracks skirting around the obstacle.

distance drives, which increased the diversity of samples that could be included in the cache [29].

3) EXTENDED MISSION

What followed the sample depot construction was a northward climb up onto the Delta, beginning another exceptional period for ENav informally labeled "Rapid Traverse 2" in Fig. 2. Unlike the first rapid traverse, this time drives were interspersed among science activities. Yet even so, during the 20 AutoNav drives that took place from sol 694 (February 1, 2023) to sol 765 (April 15, 2023), AutoNav drive segments alone (not counting initial mapping portions) averaged 207 m per sol. That left the rover poised near the western edge of Belva crater, close to the Emerald Lake region.

The area to the west of Emerald Lake initially raised some concerns in the strategic route planning team. The many seeming boulders (visible in Fig. 30 as lighter toned spots) were thought to perhaps be difficult for AutoNav to successfully navigate. However, AutoNav ultimately had no problem navigating through this terrain, which is clear in the figure from the fact that the lighter white line illustrating the rover's actual path does not deviate much from virtual straight lines connecting each sol's end-of-drive location (bright white dot)

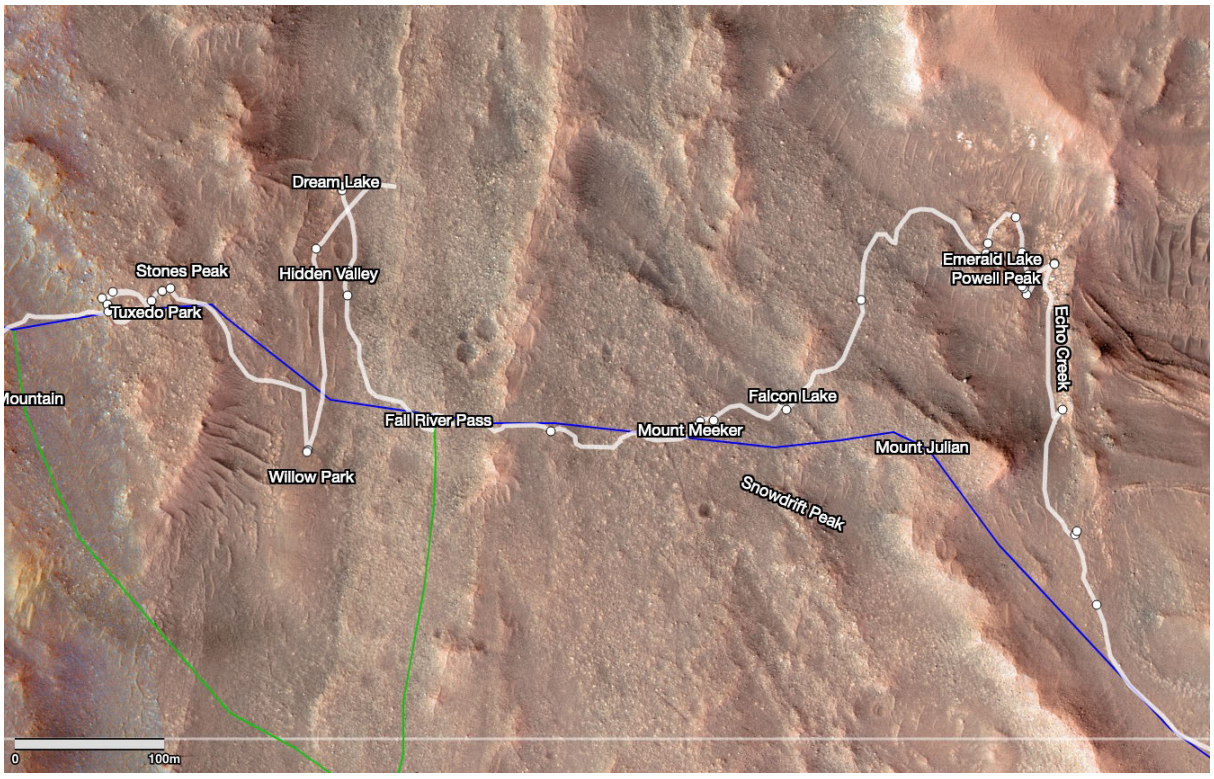


FIGURE 30. **Orbital view of sol 765–907 drives (April 15–September 8, 2023), through terrain initially thought to be problematic for AutoNav. In the end, AutoNav had no problem navigating here, with its 15 AutoNav drive segments averaging 98 m/sol. The dim white line represents the rover's actual path through the terrain, with each bright white dot representing its position at the end of one sol's drive. The blue and green lines indicate routes "suggested" by strategic route planners months before the rover actually arrived. The rover entered from the southeast at the bottom right of the image and exited on the left edge, west of the Tuxedo Park location. Image credit: NASA/JPL-Caltech/University of Arizona/USGS.**

to the next. Although this terrain was, in general, fine for AutoNav, the mobility system did encounter some steering difficulties at Tuxedo Park that delayed further progress for a few weeks [34].

The next terrain from sols 908 to 1052 (September 9, 2023–February 4, 2024) did not pose any major challenges for AutoNav. In that region, the 30 AutoNav drive segments averaged 153 m/sol.

At landing, ENav's parameters were configured conservatively, limiting planned drives to no more than 16° of tilt, based on the worst case terrain slope for cohesionless sand reported in [12]. Since much of the first three years was driven on relatively flat terrain, parameters remained conservative until higher sloped terrains were reached later in the mission. After three years of successful driving, we finally encountered some terrain that resulted in ENav declaring multiple times that it was unable to find a safe path. Labeled the "complex terrain" region in Fig. 2 and shown in higher resolution in Figs. 31–33, this terrain was predicted to be challenging for ENav, and indeed it was. During the two months spanning sol 1096–sol 1159 (March 20–May 24, 2024), no AutoNav driving was possible, so all driving was done in Mapping mode to collect example terrain images while slowly making our way through the terrain.

FIGURE 31. **Front Hazcam from sol 1100, illustrating the rocky, sandy, higher tilt complex terrain region that ENav was unable to navigate using its initial conservative parameter settings from sols 1096 to 1159.**

The greater rock density (see Fig. 31) and increasing slope in this terrain (see Fig. 34) led to greatly reduced drive performance, with the 35 human-directed VO drives from sol 1096 to 1159 averaging only 43 m/sol. Nonetheless, when

FIGURE 32. Orbital view of the complex terrain region through which AutoNav could not find a safe route. Instead, all drives from sol 1096 to sol 1159 were human-directed and, therefore, limited in distance by local terrain occlusions. Image credit: NASA/JPL-Caltech/University of Arizona/USGS.

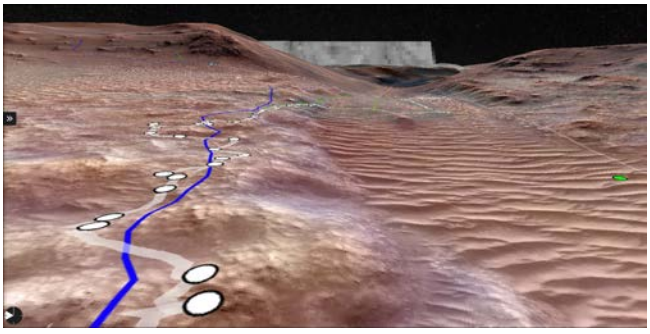


FIGURE 33. Perspective view of the complex terrain region in Fig. 32, looking west. Image credit: NASA/JPL-Caltech/University of Arizona/USGS.

AutoNav was used, it successfully avoided perceived obstacles when it could and stopped the rover when it could not, as designed. In other words, ENav kept the rover safe even in the highly complex terrain that prevented it from making forward progress.

This challenge led project management to make improving AutoNav performance a high priority for the robotic operations team [34], and caused us to reassess our conservative parameters for the first time. As a result, following substantive review of all ENav parameters and use of our Monte–Carlo simulation framework to validate new parameter settings, the maximum allowed planned tilt was raised from 16° to 20° on sol 1124.

Once we left the complex terrain region and entered the flat river valley on sol 1162 (May 27, 2024), AutoNav once again was able to take the lead. During the eight AutoNav drives in the river valley from the northern edge of the complex terrain region to Bright Angel, AutoNav drive segments averaged 134 m/sol.

Once science operations had completed at Bright Angel, we began our drive up the rim of Jezero Crater at Serpentine Rapids on sol 1244 (Fig. 35). The first two drives were stopped by terrain that exceeded the 20° maximum allowed planned tilt parameter. So after running Monte–Carlo simulations with the limit increased to 24°, that new value was applied on sol 1246. Several subsequent drives did stop early when the measured slip exceeded the limits that had been set, but the lessons learned at the complex terrain region and resulting parameter updates meant that ENav predictive analysis did not cause any more drives uphill on Serpentine Rapids to fault out. The terrain here slopes at high tilts between 16° and 23°, but fortunately, there was also exposed bedrock available almost everywhere. Therefore, when a drive faulted due to slip (almost always on sandy terrain), we could move to a more bedrock-exposed area and continue to make progress.

The next interesting area was Summerland Trail. This was an area with a high slope that the rover climbed from sols 1284 to 1298, which posed its own unique challenges. The tilts were somewhat lower on average (varying from 16° to 20°), but unlike Serpentine Rapids, there was little to no exposed bedrock (Fig. 36). After the first couple of drives, we learned that, although the terrain had various-sized pebbles on the surface, it was mostly loose material underneath. We had encountered such terrain on past missions, but never

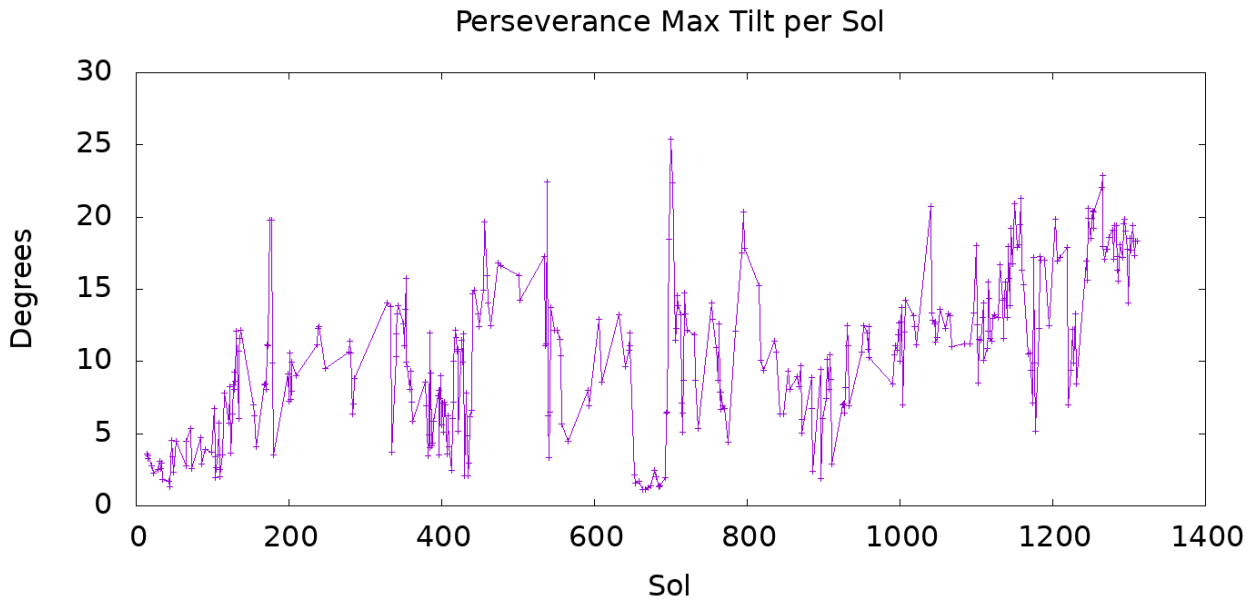


FIGURE 34. Maximum rover tilt on each sol through sol 1312 (October 28, 2024). Measurements only reflect sols during which some driving was done.

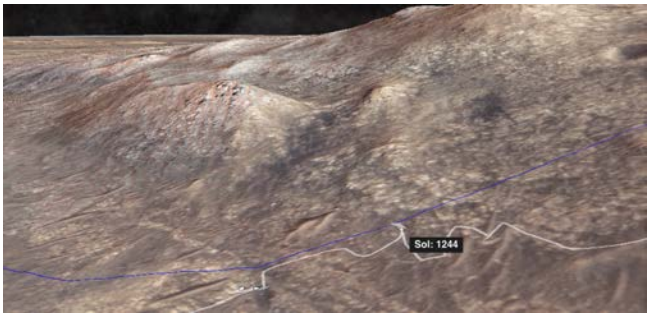


FIGURE 35. Perspective view of the initial climb up the Jezero Crater rim at Serpentine Rapids, looking southeast. Sol 1244 was the first drive up the steeper slope and reached 17° tilt before being stopped by predicted tilts above 20°. Image credit: NASA/JPL-Caltech/University of Arizona/USGS.

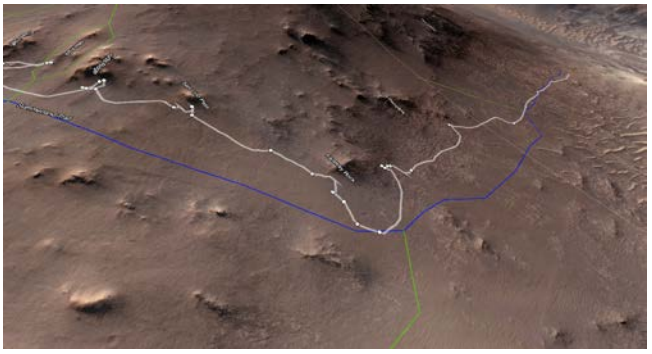


FIGURE 36. Perspective view of Summerland Trail (left) and Serpentine Rapids (right), looking north–northwest. The blue line indicates the strategically planned path that would have taken us through the middle of a valley. But the terrain at Summerland Trail has no obvious exposed bedrock, which resulted in high slip (as in Fig. 37) and led us to drive along the northern edge, closer to exposed outcrops. The northern edge provided firmer ground and allowed us to make progress. Image credit: NASA/JPL-Caltech/University of Arizona/USGS.

before at such high tilts. As a result, the rover experienced substantial slip that made uphill progress challenging, due to how much the wheels were churning through the terrain (Fig. 37). Fortunately, we were able to develop a strategy for updating the slip curve parameters (see Section II-G3) to support driving on slippery high tilt terrain with limited obstacles, and completed the drive up Summerland Trail [34].

ENav worked well during the remaining two weeks (through sol 1312 (October 28, 2024) as of this writing), with the rover staying at tilts below 20° (see Fig. 34) near the northern edge of the valley (Fig. 36). More information about Perseverance's overall robotics operations is available [31], [33], [34].

C. DRIVE RECORDS

ENav has been a critical resource during mission operations. Its use has enabled Perseverance to drive farther and faster than all prior NASA Mars rovers, setting new interplanetary distance records for autonomous driving.

Greatest Use of AutoNav: As of sol 1312 (October 28, 2024), ENav has evaluated the safety of Martian terrain during over 90% of all drives (29 087 out of 32 078 m), as shown in Table 5 and Fig. 27. This is a dramatic increase from the Curiosity mission, in which only 6.2% of the drive distance was accomplished using AutoNav or guarded modes during its first seven years of operation [27], and also the Spirit and Opportunity missions, which used AutoNav for over 23% and 5% of their total distances, respectively, [36].

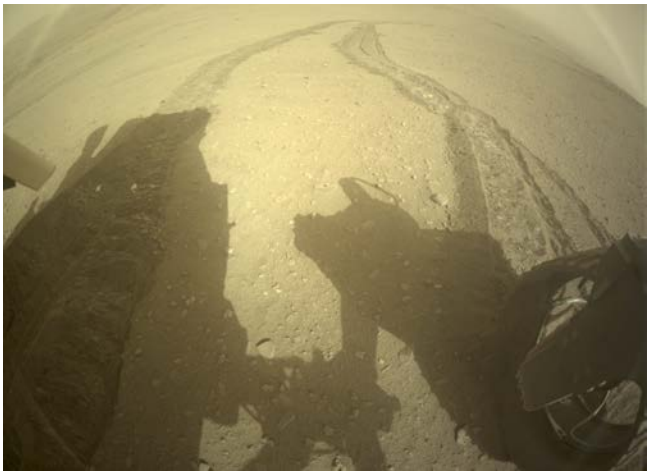


FIGURE 37. Rear Hazcam from sol 1286 with the rover at 15° tilt, illustrating the challenging terrain at Summerland Trail. The churned-up tracks behind the rover are reminiscent of high-embedding-risk terrains found on other missions, such as Opportunity sol 446 and Curiosity sol 672.

Longest Single-Sol AutoNav Drive: On sol 753 (April 3, 2023), Perseverance drove 331.74-m autonomously and 347.69 m in total (including the initial mapping phase of the drive), a new record on Mars [see Fig. 4(a)]. Before Mars 2020, the rover Opportunity had held the record with 109 m from its sol 384 drive in February 2005 [27]. Perseverance has exceeded the Opportunity record on 87 sols so far, the first time during a 119-m drive on sol 130 (July 2, 2021) which remarkably was only the second nominal AutoNav drive commanded after the initial checkout phase.[3]

Longest Multisol AutoNav Drive: On sols 717–719 (February 25–27, 2023), Perseverance drove 655.80-m autonomously. All three sols' drives were planned on February 24, 2023, and they executed on Mars in one command cycle, with no ground in the loop (i.e., no humans adjusted the plan after it was sent). Before that, the rover Opportunity had held the record with 390-m total (280-m autonomously) from its sol 383–385 drive in February 2005 [3].

Longest Continuation AutoNav Drive: On sols 407–409 (April 12–14, 2022), Perseverance drove 699.85-m autonomously [26]. Although the three sols' drives were planned on three different days, the last two were planned without foreknowledge of where the rover had ended on the prior sol, so the transitions to the later sols' drives were still truly autonomous.

Highest Tilt AutoNav: Earlier Mars rovers, being compute-limited, typically ran AutoNav without VO in most circumstances. As a result, their AutoNav was typically limited to lower tilt settings to avoid slip issues that might cause problems with map registration. In contrast, Perseverance has demonstrated successful drives using AutoNav and VO even at tilts up to 23°, e.g., on sol 1265 (September 10, 2024).

[3] After the initial submission of this article, Perseverance once again broke its own record, achieving 411.7 m during a single day's drive on sol 1540 (June 19, 2025), bringing its total commanded distance to 37.5 km.

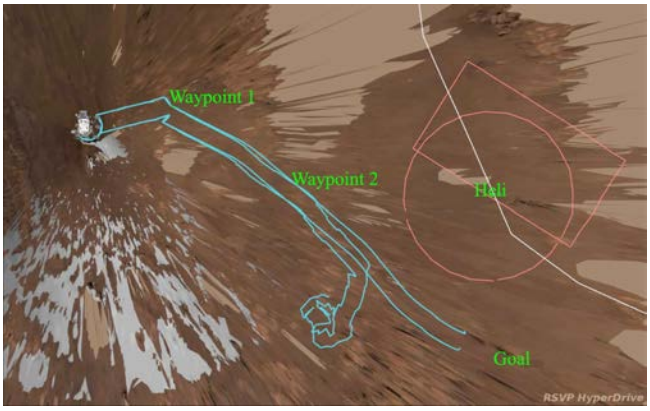


FIGURE 38. Plot of the simulated and actual ENav-selected paths driven on sol 129. The white line indicates the strategic path (planned weeks earlier on orbital maps alone), the red circle and rectangle indicate KOZs placed around the Ingenuity helicopter, labeled "Heli" in green. The rover started near the top-left corner of the image, where the rover is shown. The smooth set of light blue tracks was the simulated route, and the other light blue tracks indicate the path actually taken as ENav tried to find a way past the helicopter while avoiding perceived hazards. Waypoints and the goal along the planned path are also labeled in green.

FIGURE 39. NavCam image of the Ingenuity helicopter and some of the final hazard avoidance tracks from sol 129. This view looks back at some of the tracks made by the drive shown in Fig. 38. The white scale bar under Ingenuity indicates a 2.5-m length at that distance from the rover.

D. NOTABLE DRIVES

Section V-B described ENav performance in general over the first 1312 sols of operation. Here, we provide several single-sol example drives from Mars operations to illustrate ENav's performance.

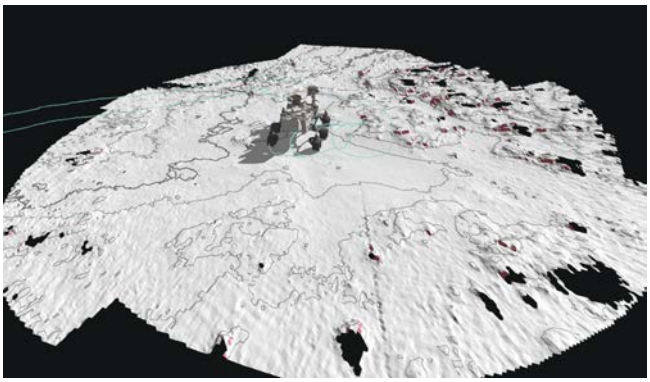


FIGURE 40. ENav Heightmap from the end of the sol 129 drive. The rocks in the front right of the rover are colored red, indicating they are perceived as being hazards. The is an "over the shoulder" view of the scene in Fig. 39.

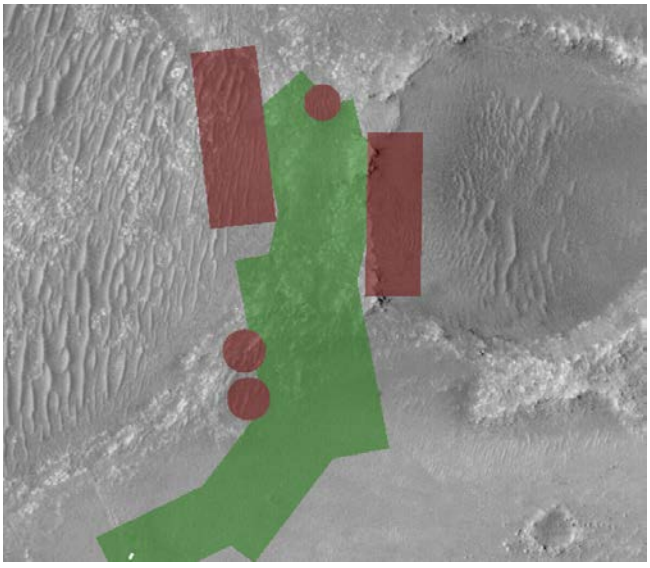


FIGURE 41. Sol 383 image of the keep-in (green) and keep-out (red) zones associated with the sol 383–385 three-sol 475-m drive plan. There are narrow gaps between the KOZs at the northern edge of the path. The left rectangular KOZ was added to prevent driving into the polygonally shaped ripples inside the crater on the left, the right rectangular KOZ blocked descent into the crater on the right, and the topmost circular KOZ was placed over a large sand ripple visible in the orbital map. A green KIZ is placed around each segment of a drive, normally with ever-growing width. But in this case, the northernmost KIZs were kept narrow enough so as to avoid allowing entry into the surrounding craters. Image courtesy [34].

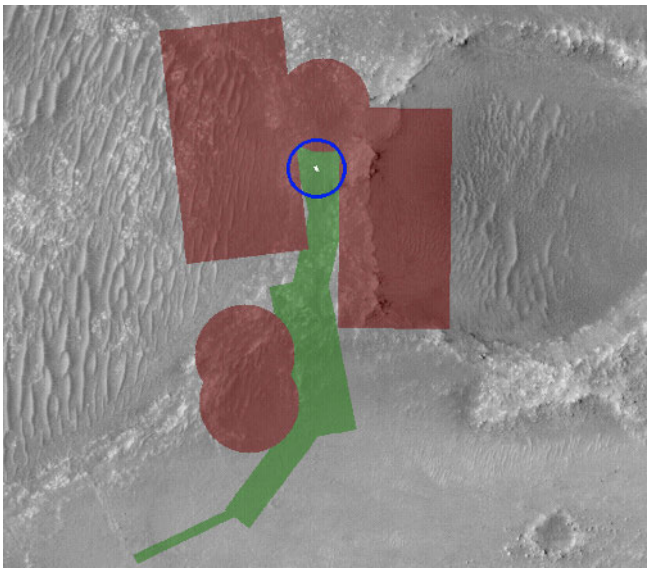


FIGURE 42. Sol 385 image of the keep-in and keep-out zones associated with the sol 383–385 three-sol 475-m drive plan. By the time the rover had reached the northern edge of the path, the position uncertainty had grown to 26.9 m (indicated by the blue circle in the figure). That caused the gaps of safety in the path to disappear, as the KOZs had grown so large that there was no longer any safe path through them. Image courtesy [34].

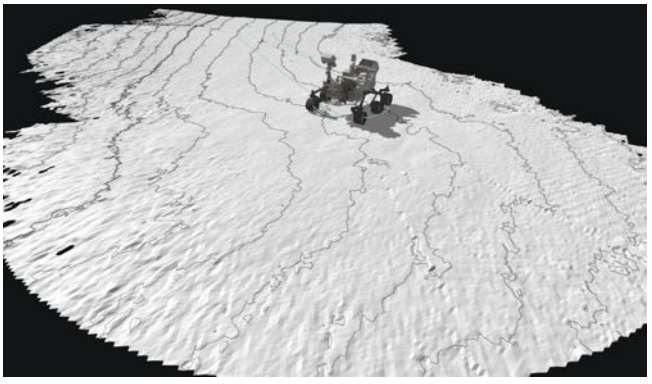


FIGURE 43. Sol 385 final ENav Heightmap shows that the terrain looked very safe. If the KOZs had not grown so large due to uncertainty, Perseverance could have kept driving.

1) SOL 129 (AVOIDING THE INGENUITY HELICOPTER)

On sol 129 (June 23, 2021), Rover Planners commanded a drive using 8.2 m of mapping and 44.3 m of AutoNav, assuming no hazard avoidance (Fig. 38). As the figure indicates, the rover did not have a good view of the terrain nearby: there were gaps in the NavCam-generated terrain mesh due to occlusions in the stereo data, and the terrain data 40+ m away was heavily interpolated (smoothed over). The intent was to bring the rover out of some rocky terrain, back onto a clearer path on the strategic route, shown as a white line in the figure. However, since the Ingenuity helicopter had landed on the strategic path nearby, two KOZs were placed around it, a 20-m-diameter disk and a 12×22 m^2-rectangle to ensure Perseverance would not get too close to it (Fig. 38).

The combination of the KOZs and some larger than-expected terrain features meant ENav was unable to find a safe path onto the strategic path that sol (Figs. 39 and 40). Instead, it kept searching for a way to reach its next goal, turning around and looping back before ultimately stopping due to too much uncertainty having accumulated in its attitude estimate. While it was unable to reach the goal that sol, ENav kept Perseverance and Ingenuity safe. And the sol

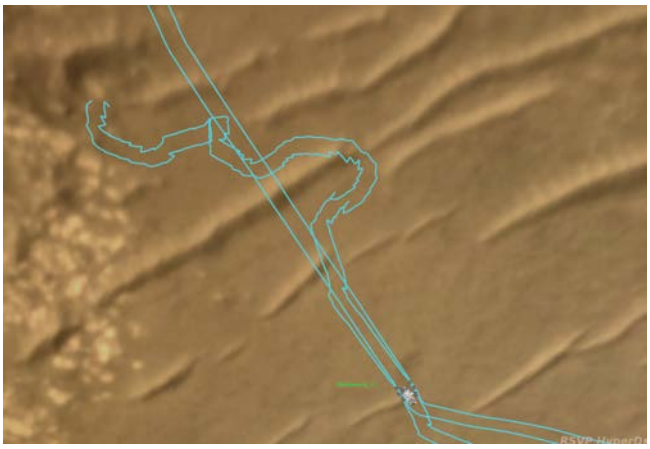


FIGURE 44. Plot of the simulated and actual ENav-selected paths through sand dunes on sol 909. The simulation path moves in straight lines between waypoints, because the orbital mesh used for planning does not provide detailed terrain features: so the sand dunes were smoothed out and not perceived as obstacles when the drive plan was initially simulated.

FIGURE 45. NavCam image looking back at the sol 909 ENav-selected path through sand dunes shown in Fig. 44.

130 drive the next day did succeed in rejoining the strategic path, in a 129.3-m mapping and AutoNav drive.

2) SOL 385 (LIMITED BY UNCERTAINTY)

On sol 385 (March 21, 2022), Perseverance was in the midst of its rapid traverse campaign. Rover Planners created a series of northward drives as part of a three-sol plan for sols 383–385, and the first two sols went very well, driving 241 and 215 m, respectively (see [26, Table 4]). However, the sol 385 drive only managed to drive an additional 14 m, in spite of having been allocated over 2 h for driving. The reason was that there was a narrow gap between KOZs at the northern edge of the drive plan (Fig. 41). By the time it had driven over 460 m of distance, it had accumulated 26.9 m of uncertainty.

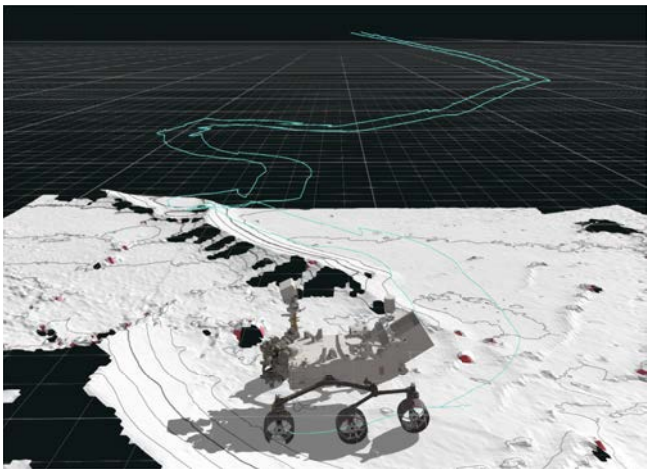


FIGURE 46. View of the final ENav Height Map and entire drive path through sand dunes shown in Figs. 44 and 45.

And since ENav grows KOZs (and shrinks Keepin Zones) by that uncertainty, that caused the KOZs to grow so large that no safe path remained between them (Fig. 42), and the drive terminated with a NO_PATH fault.

This is a prime example of the motivation for providing an autonomous capability for onboard global localization, something imminently planned for deployment on Perseverance [34]. Had the rover been able to localize itself and reduce its uncertainty autonomously, this drive could have covered over another 150 m of distance in the remaining time allocated (Fig. 43).

3) SOL 909 (AVOIDING SAND RIPPLES)

On sol 909 (September 10, 2023), Perseverance was commanded to drive autonomously northwest through a sandy area. Fig. 44 shows the circuitous path it selected to avoid having to climb over sand dunes in the region, Fig. 45 is a NavCam view of the tracks taken at the end of the drive, and Fig. 46 shows the ENav Heightmap model of the final 15 m of the drive. Although ENav did not have knowledge of the composition of the terrain, the unseen areas on the far side of the ripples (visible as gaps in the Heightmap terrain model) were enough to dissuade it from attempting to climb them.

4) SOL 1001

On sol 1001 (December 13, 2023) Perseverance entered some rock-abundant terrain (Fig. 47), and ENav was unable to find a safe path to go through the rock field and over a ridge line (Figs. 48 and 49). This was an early indication that the conservative parameters being used in ACE might keep the rover safe, but would be unable to navigate through more complex and higher tilt terrain.

VI. FUTURE WORK

Future surface exploration missions may choose to make use of ENav's capabilities. Lunar missions such as the Lunar Terrain Vehicle, the Endurance rover, and any platform that can

FIGURE 47. NavCams image on sol 1001.

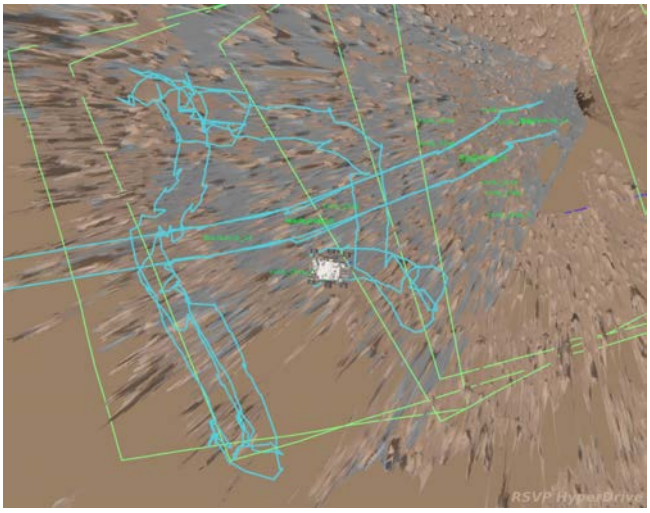


FIGURE 48. Drive route chosen by ENav on sol 1001. The plan had been to drive some 268 m mostly westbound, but the rover was unable to find a safe path down a nearby ridge line, indicated by the gaps in the terrain on the left side of the image due to occlusions in the stereo data. The rover drove 141-m trying to find a safe path, but ultimately stopped only 18 m from its starting location.

provide point cloud geometric terrain measurements might directly benefit from the existing ENav capabilities.

Further improvement of ENav's performance is possible with parameter tuning (as described in Section V-B3) or FSW upgrades. Multiple changes to the current FSW (version S8.0.0.3, which has been in use since sol 1151, or May 15, 2024) have already been implemented as a result of the steering and ENav issues encountered along the way, especially in the complex terrain region. Many of these changes (implemented in version S8.1) were designed to provide greater control over how to minimize the use of turns-in-place since they typically result in around 90° of steering motion on each corner wheel.

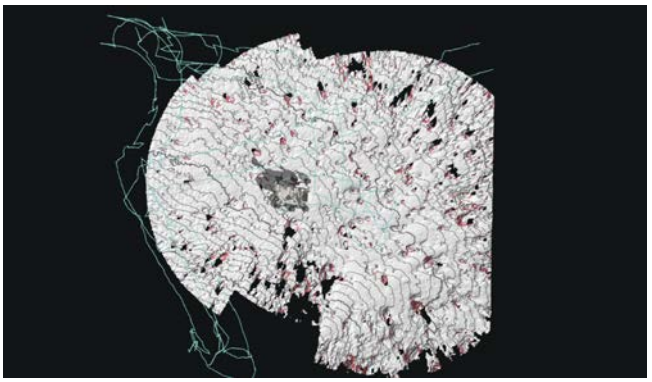


FIGURE 49. ENav heightmap from the end of the sol 1001 drive shown in Fig. 48. The red areas indicate perceived vehicle hazards, and this area had a large number of them.

More substantial improvements in autonomous driving capability on Mars would be possible if significantly more onboard computation resources were available. For example, ENav's traversability assessment is highly conservative, as exemplified by the failed drive on sol 1001 (Section V-D4) in a rock field. This is primarily because of the conservative approximation by ACE (Section II-G). With more abundant onboard computational resources, the conservatism could be eliminated by running exact kinematic settling, which would allow future rovers to drive autonomously on substantially more complex terrains. A preliminary implementation of such onboard kinematic settling (OBKS) exists [24], but it has not been deployed on actual missions.

Another major bottleneck to very long-range autonomous driving on Perseverance had been the inability to autonomously reduce the onboard position uncertainty due to a lack of onboard global localization (i.e., localization against an orbital map). This, in turn, limited the distance AutoNav could drive without requiring a ground-based localization, as exemplified in the sol 385 drive (Section V-D2). But recently, the project repurposed the Qualcomm Snapdragon processor on the base station for Mars Helicopter Ingenuity, which has substantially greater computation power than Perseverance's main RAD750 computer. Exploiting this additional computation resource, global localization capability was added to the Snapdragon co-processor, and the ability to invoke it was added to the rover FSW in version S8.0.0.3 (and updated in version S8.1 to simplify its operation) to better enable even longer autonomous drives. ENav heightmap structures automatically incorporate the new position and uncertainty knowledge generated by a successful global localization result. These changes have been completed and will be deployed in the next FSW upgrade (version S8.1) in August 2025 [34]. Further enhancements in perception, localization, and planning with enhanced compute resources would be possible in future missions.

Remarkable progress in autonomous driving on Earth in recent years was enabled by machine learning. With increased onboard computation power, extraterrestrial driving would

benefit from machine learning, too. For example, semantic segmentation on onboard images would allow future rovers to recognize terrain types and plan paths to avoid nongeometric hazards, such as slippery sand. Existing research showed that a convolutional neural network (CNN) trained on labeled rover images can classify terrain types with ∼97% accuracy compared to human expert labels [2]. Other research demonstrated that machine learning-based heuristics can sort the paths more efficiently and reduce the number of ACE runs up to 7 times when integrated with ENav [6]. There could certainly be more unexplored applications of machine learning that could improve future autonomous driving on Mars and beyond.

VII. CONCLUSION

This article presented ENav, the path-selection algorithm for autonomously driving the Perseverance rover on Mars. The main challenge was to provide a robust and provably safe path-planning capability with a highly limited onboard compute resource—the single-core RAD750 processor clocked at 133 MHz. To address this challenge, we developed a unique approximate kinematic collision checking algorithm, ACE, as well as a novel two-stage optimization approach that limits the number of ACE runs in nominal cases while ensuring that an efficient path is returned without stopping the rover for excessive computation time. ENav also provides robustness against slip by expanding the ACE wheel boxes to accommodate the predicted worst case slip for a given terrain. Before the launch to Mars, ENav was extensively tested in simulation and on hardware testbeds at JPL's Mars Yard. After its arrival on Mars in 2021, ENav was successfully validated through a number of check-out activities and approved for regular use on Mars. As of this writing, on sol 1312, or October 28, 2024, on the Earth calendar, 90% of all 32.1 km of driving has used ENav to evaluate the terrain. This is an order of magnitude improvement from her predecessor rover, Curiosity, which drove only 6.2% of the distance using AutoNav or Guarded modes. ENav made multiple driving records on Mars possible, such as the longest drive distance over one Martian day, which was 347.69 m, recorded on sol 753 (but see the footnote in Section V-C). ENav is one of the major enablers of Perseverance's ambitious mission, which is still ongoing as of this writing, to drive over unprecedented distances and perform scientific activities, including the collection of rock samples that may contain evidence of past life that might have existed on Mars long ago. ENav will continue to push the frontiers of exploration on Mars in the years to come.

APPENDIX
ENav IMPLEMENTATION DETAILS

This section describes details of the FSW implementation of ENav, which is encapsulated as the NAVLIB module, as explained in Section III.

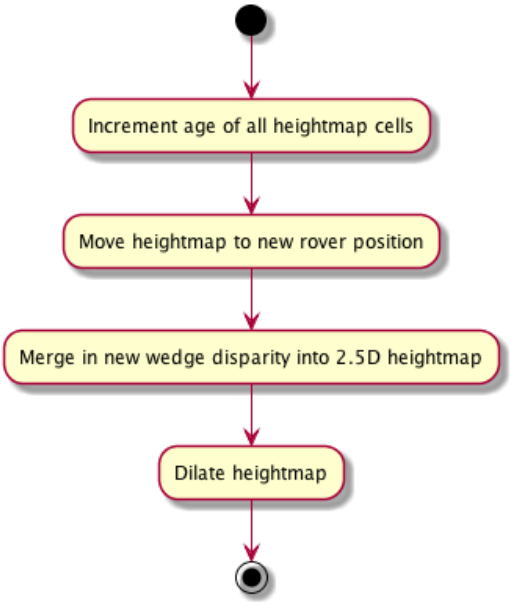


FIGURE 50. Activity diagram of the *NAVLIB_process_disparity* interface.

A. NAVLIB's PUBLIC INTERFACES

The main public interfaces of the NAVLIB library are used to perform the following operations. The corresponding function names are shown in italics. The details of the functions are described in the following subsections.

1) Ingest the stereo data (3-D mesh around the rover needed to plan safe paths around terrain hazards)—*NAVLIB_process_disparity*.
2) Update the internal costmap used for global planning once all the stereo data has been ingested—*NAVLIB_analyze_terrain*.
3) Generate the best path to drive—*NAVLIB_select_arc*.
4) Evaluate whether the provided arc is safe to drive—*NAVLIB_evaluate_arc*.
5) Set the keep-in and -out zones (KIOZs)—*NAVLIB_set_kiozs*.
6) Check if the "turn for comm" constraint is satisfied (i.e., whether the predicted rover tilt computed from the mesh is less than the max tilt at the rover's heading)—*enav_trees_search_tfc_tree*.
7) Save and load the world for multisol driving (to restore NAVLIB's internal state from one sol to the next, after FSW reboots)—*NAVLIB_save_world*[4] *NAVLIB_load_world*.

A more detailed description of the design of the first four key interfaces from the list above is provided in Section III.

1) PROCESS DISPARITY

Nav calls this NAVLIB function whenever new stereo data is available, so NAVLIB can update its internal represen-

[4]This is by far the most liked function name by the ENav Team.

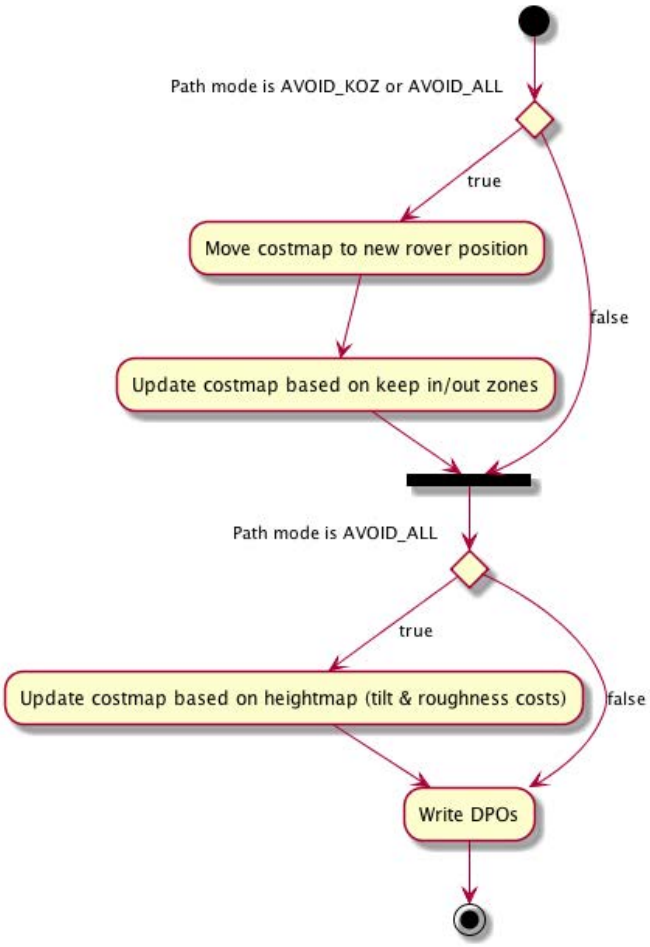


FIGURE 51. **Activity diagram of the *NAVLIB_analyze_terrain* interface.**

tation of the 3-D environment around the rover (i.e., its 2.5-D heightmap). Fig. 50 shows the activity diagram of the *NAVLIB_process_disparity* interface.

2) ANALYZE TERRAIN

Nav calls this NAVLIB function either right after all the new stereo data has been ingested by NAVLIB (via *NAVLIB_process_disparity*), or if the path mode has changed and the costmap needs to be updated accordingly: for example, if the rover crawled onto the map using an *UNGUARDED GoTo*, the costmap has not yet been updated (since it is not needed in *UNGUARDED* mode), and therefore gets updated via *NAVLIB_analyze_terrain* at the start of the next *GoTo* if the path mode changes (all the other three modes rely on the costmap for at least handling the KIOZs). Fig. 51 shows the activity diagram of the *NAVLIB_analyze_terrain* interface.

3) SELECT ARC

Nav calls this NAVLIB function when running a *GoTo* command, in order to determine which arc to drive to make progress toward the desired goal. Fig. 52 shows the activity diagram of the *NAVLIB_select_arc* interface.

4) EVALUATE ARC

Both the MOM and NAV modules call this NAVLIB function to evaluate if a given arc is safe to drive. Fig. 53 shows the activity diagram of the *NAVLIB_evaluate_arc* interface.

B. CODE TESTING

Due to the complexity and criticality of the NAVLIB FSW that must safely and autonomously drive our priceless national space asset on another planet, special emphasis was placed on unit testing. The NAVLIB module has 100% line, function, and branch coverage, as measured by the Gcov [9] code coverage tool. There are about 17 000 physical SLOC for the NAVLIB software that runs onboard the rover and over 35 000 SLOC for the associated unit tests. In other words, more than twice as much source code was written for unit testing as for the NAVLIB library itself. It should also be noted that unit testing, which consists of writing limited-scope tests for checking the behavior of specific functions within the library, is only a small part of the overall testing performed by the team to verify and validate the AutoNav software. Other tests performed include automated static and dynamic code analysis (using an updated version of [13]), fuzzy testing where the software behavior is checked when various mod-

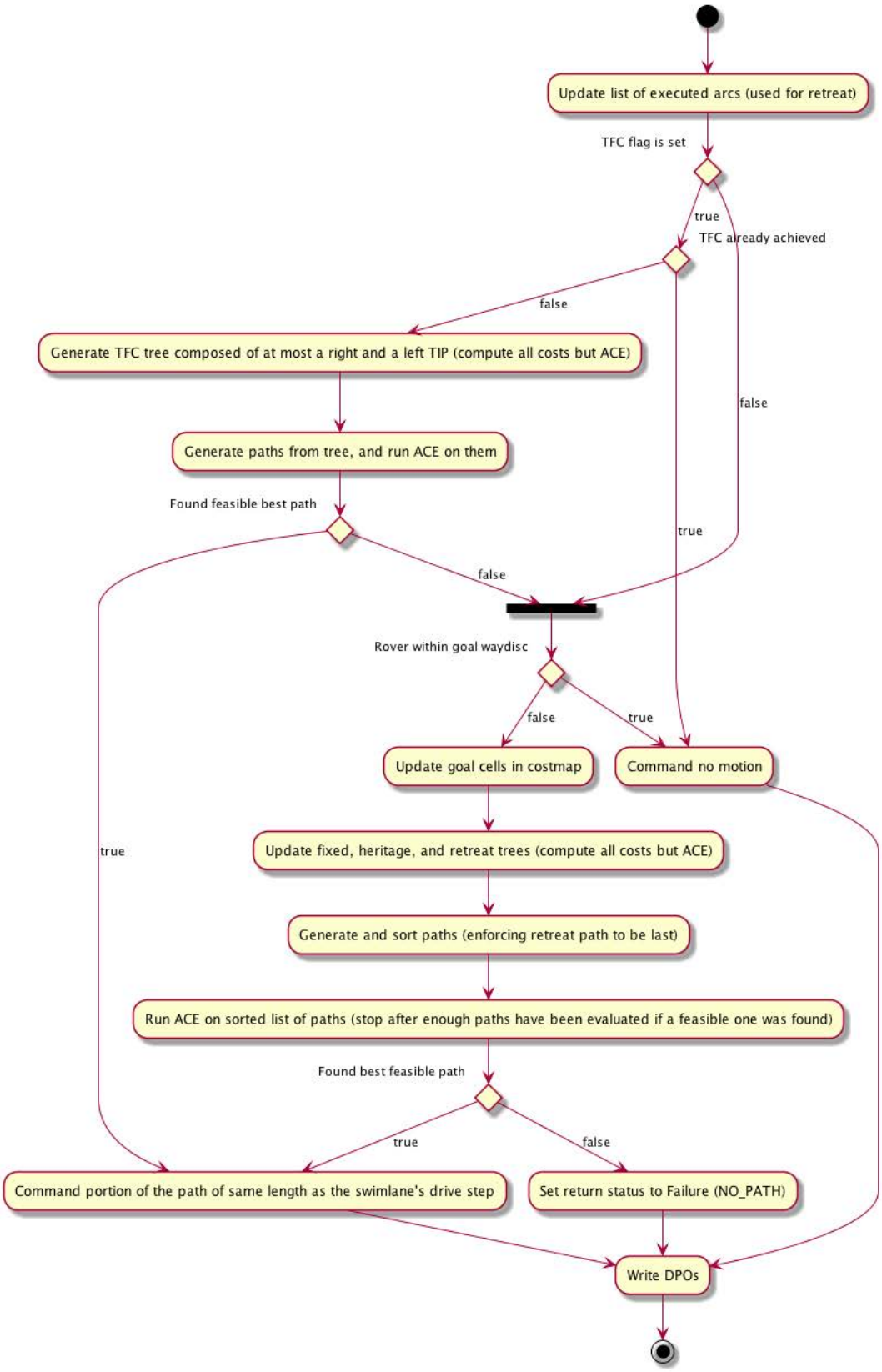


FIGURE 52. **Activity diagram of the *NAVLIB_select_arc* interface.**

ule parameters are randomly changed, and integrated system testing in various representative surface operations scenarios both in extensive Monte–Carlo simulations and hardware testbeds, as further described in Section IV.

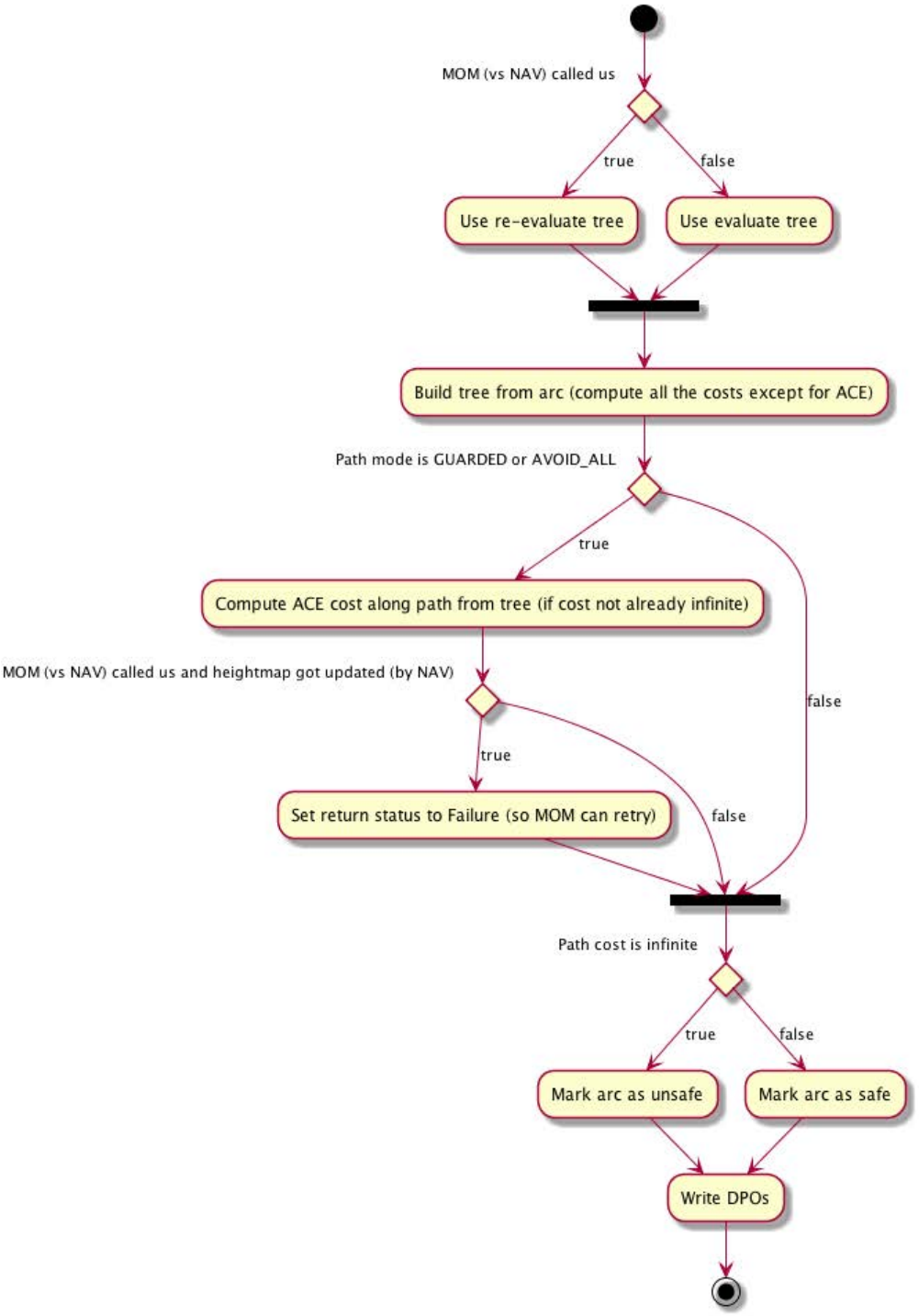


FIGURE 53. **Activity diagram of the *NAVLIB_evaluate_arc* interface.**

ACKNOWLEDGMENT

Thanks to Francois Ayoub for creating the sol 1540 driving movie, an excellent visualization of the longest autonomous drive ever achieved on Mars. The movie is included in the supplemental materials. Thanks to the Mars 2020 Robotic Operations Downlink Team (especially Evan Graser, Darwin Chiu, and Xianmei Lei) for downlink tables and plots. Thanks to Steven Myint for his year of development, getting the Nav flight software (FSW) module up and running. Thanks to Justin Maki for his help in understanding the lighting operational constraints. Thanks to Jeff Biesiadecki for the MOM FSW and very constructive paper review comments. And special thanks to the mobility FSW test teams for running the many shifts needed to validate the performance of the system, including Josh Collier, Tara Estlin, Heather Justice, Chris Matthes, Elio Morillo, Rich Rieber, Leann Scott, Bryan

Sonneveldt, Mikey Stragier, Jose Trujillo, Philip Twu, and Anais Zarifian.

The work of Olivier Toupet was completed when he was with the Jet Propulsion Laboratory, Pasadena, CA, USA.

REFERENCES

[1] K. S. Ali et al., "Attitude and position estimation on the Mars exploration rovers," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 1, Waikoloa, HI, USA, Oct. 2005, pp. 20–27.

[2] D. Atha, R. M. Swan, A. Didier, Z. Hasnain, and M. Ono, "Multi-mission terrain classifier for safe rover navigation and automated science," in *Proc. IEEE Aerosp. Conf. (AERO)*, Mar. 2022, pp. 1–13.

[3] J. J. Biesiadecki and M. W. Maimone, "The Mars exploration rover surface mobility flight software: Driving ambition," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2006, pp. 1–15.

[4] S. Brooks et al., "Testing Mars 2020 flight software and hardware in the surface system development environment," in *Proc. IEEE Aerosp. Conf. (AERO)*, Mar. 2022, pp. 1–13.

[5] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global path planning on board the Mars exploration rovers," in *Proc. IEEE Aerosp. Conf.*, Mar. 2007, pp. 1–11.

[6] S. Daftry et al., "MLNav: Learning to safely navigate on Martian terrains," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5461–5468, Apr. 2022.

[7] L. Ding et al., "A 2-year locomotive exploration and scientific investigation of the lunar farside by the Yutu-2 rover," *Sci. Robot.*, vol. 7, no. 62, Jan. 2022, Art. no. eabj6660.

[8] K. A. Farley et al., "Mars 2020 mission overview," *Space Sci. Rev.*, vol. 216, no. 8, pp. 1–41, 2020.

[9] Gcov. (2001). *Gcov Test Coverage Program*. [Online]. Available: https://gcc.gnu.org/onlinedocs/gcc/Gcov.html

[10] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," in *IEEE Proc. Aerosp. Conf.*, vol. 5, Mar. 2002, p. 5.

[11] M. Golombek and D. Rapp, "Size-frequency distributions of rocks on Mars and Earth analog sites: Implications for future landed missions," *J. Geophys. Res., Planets*, vol. 102, no. E2, pp. 4117–4129, Feb. 1997.

[12] M. Heverly et al., "Traverse performance characterization for the Mars science laboratory rover," *J. Field Robot.*, vol. 30, no. 6, pp. 835–846, Nov. 2013.

[13] G. J. Holzmann, "SCRUB: A tool for code reviews," *Innov. Syst. Softw. Eng.*, vol. 6, no. 4, pp. 311–318, Dec. 2010.

[14] T. M. Howard et al., "Enabling continuous planetary rover navigation through FPGA stereo and visual odometry," in *Proc. IEEE Aerosp. Conf.*, Mar. 2012, pp. 1–9.

[15] A. E. Johnson, Y. Cheng, J. F. Montgomery, N. Trawny, B. Tweddle, and J. X. Zheng, "Real-time terrain relative navigation test results from a relevant environment for Mars landing," in *Proc. AIAA Guid., Navig., Control Conf.*, Jan. 2015, p. 0851.

[16] R. Lange, T. L. Wagner, S. M. Milkovich, and M. Ono, "Mars 2020 surface mission performance modeling: Part 3. Mission performance modeling approach and results," in *Proc. AIAA SPACE Astronaut. Forum Expo.*, 2018, p. 5420.

[17] M. W. Maimone, "Visualizing multi-process CPU utilization using CUSP," in *Proc. IEEE Aerosp. Conf. (AERO)*, Big Sky, MT, USA, Mar. 2022, pp. 1–8.

[18] M. Maimone, N. Patel, A. Sabel, A. Holloway, and A. Rankin, "Visual odometry thinking while driving for the curiosity Mars Rover's three-year test campaign: Impact of evolving constraints on verification and validation," in *Proc. IEEE Aerosp. Conf. (AERO)*, Mar. 2022, pp. 1–10.

[19] J. N. Maki et al., "The Mars 2020 engineering cameras and microphone on the perseverance rover: A next-generation imaging system for Mars exploration," *Space Sci. Rev.*, vol. 216, no. 8, p. 137, Dec. 2020, doi: 10.1007/s11214-020-00765-9.

[20] J. Matijevic, "Autonomous navigation and the sojourner microrover," *Science*, vol. 280, no. 5362, pp. 454–455, Apr. 1998.

[21] M. McHenry et al., "Mars 2020 autonomous rover navigation," in *Proc. 43rd AAS Rocky Mountain Sect. Guid., Navig. Control Conf.*, Breckenridge, CO, USA, 2020, pp. 867–880.

[22] M. Ono et al., "Mars 2020 site-specific mission performance analysis: Part 2. Surface traversability," in *Proc. AIAA SPACE Astronaut. Forum Expo.*, Sep. 2018, p. 5419.

[23] M. Ono et al., "Data-driven surface traversability analysis for Mars 2020 landing site selection," in *Proc. IEEE Aerosp. Conf.*, Mar. 2016, pp. 1–12.

[24] M. Ono et al., "MAARS: Machine learning-based analytics for automated rover systems," in *Proc. IEEE Aerosp. Conf.*, Mar. 2020, pp. 1–17.

[25] K. Otsu, G. Matheron, S. Ghosh, O. Toupet, and M. Ono, "Fast approximate clearance evaluation for rovers with articulated suspension systems," *J. Field Robot.*, vol. 37, no. 5, pp. 768–785, Aug. 2020.

[26] A. Rankin et al., "Perseverance rapid traverse campaign," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2023, pp. 1–16.

[27] A. Rankin, M. Maimone, J. Biesiadecki, N. Patel, D. Levine, and O. Toupet, "Driving curiosity: Mars rover mobility trends during the first seven years," *J. Field Robot.*, vol. 38, no. 5, pp. 759–800, Aug. 2021.

[28] R. Rieber, M. McHenry, P. Twu, and M. M. Stragier, "Planning for a Martian road trip—The Mars2020 mobility systems design," in *Proc. IEEE Aerosp. Conf. (AERO)*, Mar. 2022, pp. 1–18.

[29] J. I. Simon et al., "Samples collected from the floor of Jezero Crater with the Mars 2020 Perseverance rover," *J. Geophys. Res., Planets*, vol. 128, no. 6, 2023, Art. no. e2022JE007474.

[30] O. Toupet, T. Del Sesto, M. Ono, S. Myint, J. vander Hook, and M. McHenry, "A ROS-based simulator for testing the enhanced autonomous navigation of the Mars 2020 rover," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2020, pp. 1–11.

[31] V. Verma et al., "First 210 solar days of Mars 2020 perseverance robotic operations–mobility, robotic arm, sampling, and helicopter," in *Proc. IEEE Aerosp. Conf. (AERO)*, Mar. 2022, pp. 1–20.

[32] V. Verma and C. Leger, "SSim: NASA Mars rover robotics flight software simulation," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2019, pp. 1–11.

[33] V. Verma et al., "Results from the first year and a half of Mars 2020 robotic operations," in *Proc. IEEE Aerosp. Conf.*, Mar. 2023, pp. 1–20.

[34] V. Verma et al., "Robotic operations during Perseverance's first extended mission," in *Proc. IEEE Aerosp. Conf.*, Mar. 2025, pp. 1–21.

[35] V. Verma et al., "Robotic operations for the first sample depot on Mars," in *Proc. IEEE Aerosp. Conf.*, Mar. 2024, pp. 1–12.

[36] V. Verma et al., "Autonomous robotics is driving Perseverance rover's progress on Mars," *Sci. Robot.*, vol. 8, no. 80, 2023, Art. no. eadi3099.

[37] V. Verma et al., "Enabling long & precise drives for the perseverance Mars rover via onboard global localization," in *Proc. IEEE Aerosp. Conf.*, Mar. 2024, pp. 1–18.

[38] C. Y. Villalpando, A. Morfopolous, L. Matthies, and S. Goldberg, "FPGA implementation of stereo disparity with high throughput for mobility applications," in *Proc. Aerosp. Conf.*, Mar. 2011, pp. 1–10.

[39] J. Wang et al., "Application of computer vision technology in collaborative control of the 'Zhurong' Mars rover," in *Image and Graphics Technologies and Applications*, W. Yongtian and W. Lifang, Eds., Singapore: Springer, 2023, pp. 425–439.

[40] J. Wright, F. Hartman, B. Cooper, S. Maxwell, J. Yen, and J. Morrison, "Driving on Mars with RSVP," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 37–45, Jun. 2006.

OLIVIER TOUPET received the M.S. degree in aeronautics and astronautics from MIT, Cambridge, MA, USA, in 2006.

He is currently a Distinguished AI Software Engineer at Zoox, Foster City, CA, USA, Amazon's self-driving robotaxi company. Prior to that, he was the supervisor of the Aerial Mobility Group, JPL, Pasadena, CA, USA, where he developed innovative technologies to enhance the autonomy of uncrewed aerial vehicles (UAVs) and supported surface operations of the Mars Perseverance rover and Ingenuity helicopter. He was the Lead of the Enhanced Navigation Team and owner of the NAVLIB flight software module described in this article, for which he was awarded the NASA Exceptional Technology Achievement Medal. He also designed the terrain-adaptive wheel speed control algorithm for the Curiosity Mars rover, and received the NASA Exceptional Engineering Achievement Medal for that work.

MASAHIRO (HIRO) ONO received the Ph.D. degree in aeronautics and astronautics from MIT, Cambridge, MA, USA, in 2012.

He is currently the Supervisor of the Robotic Mobility Group, JPL, Pasadena, CA, USA. Before joining JPL in 2013, he was an Assistant Professor at Keio University, Kanagawa, Japan. He was the PI of the EELS Project to create a highly versatile and intelligent snake-like robot for exploring unknown environments, such as Enceladus vents. As a member of the Mars 2020 Rover (M2020) Mission, he also supports tactical rover operations. Previously, he developed M2020's autonomous driving algorithm and also led the landing site traversability analysis. His research interest is centered around the application of robotic autonomy to space exploration, with an emphasis on machine learning applications to perception, data interpretation, and risk-aware decision-making.

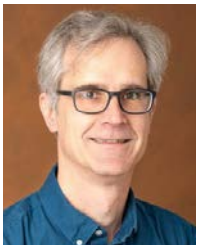
TYLER DEL SESTO received the M.S. degree in mechanical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2016.

At JPL, Pasadena, CA, USA, his work focuses on control and testing of mobile robots and improving the operability of robotic spacecraft. He was the test lead for the Perseverance rover's autonomous driving software during development. He was a robotics operator of the Curiosity rover for four years, and is currently an Operator, a Mobility Domain Expert, and the Deputy Rover Planner Lead for the Perseverance rover. He was the Mobility Lead for Three Forks Sample Depot Construction as well as the M2020 Strategic Route Planning Lead at that time.

MARK MAIMONE received the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, USA, in 1996.

He is a JPL Principal in Autonomous Planetary Rover Navigation and the Mars 2020 Robotic Operations Deputy Team Chief, a Mobility Technical Authority, and a member of the Rover Planner and FSW Development Teams. He was the MSL Deputy Lead Rover Planner, the Lead Mobility Rover Planner, and the Flight Software Lead; developed downlink automation tools for MER/MSL; and is on NASA's Lunar Terrain Vehicle Insight Team. He was a Post-Doctoral Researcher of robotics at Carnegie Mellon University, Pittsburgh, PA, USA.

Dr. Maimone received the NASA Exceptional Achievement Medals for Designing and Implementing GESTALT Self-Driving Flight Software for MER and MSL; he contributed to the Mars 2020 ENav Self-Driving Flight Software.



MICHAEL MCHENRY received the Ph.D. degree in computer science from the University of Southern California, Los Angeles, CA, USA, in 1998.

He is currently the Supervisor of the Robotics Systems Staff Group, Jet Propulsion Laboratory, Pasadena, CA, USA. He was also the Mobility Technical Authority and a Rover Planner for the Perseverance Rover. He has 25 years of experience in robotics research and flight. His JPL career started with DARPA's Tactical Mobile Robot Program (TMR), where he contributed to hazard avoidance, urban localization, user interfaces, and autonomous stair climbing capabilities. He led a variety of both reimbursable and NASA-focused research efforts involving the application of machine vision and motion planning to surface vehicles as well as other operations-focused tasks. More recently, he led Perseverance AutoNav Development, including the infusion of FPGA-accelerated image processing [via the vision compute element (VCE)], the ENav Hazard Detection and Avoidance software, and thinking-while-driving software updates.