A Software-Based Approach to Radiation Mitigation for Planetary Missions

Jeremy Nash
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
jeremy.nash@jpl.nasa.gov

Vandi Verma
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
vandi@jpl.nasa.gov

Evan Graser
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
graser@jpl.nasa.gov

Haoda Wang
Columbia University
530 W 120th St.
New York, NY 10027
haoda.wang@columbia.edu

Gerik Kubiak
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
gerik.kubiak@jpl.nasa.gov

Mark Maimone
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
mark.maimone@jpl.nasa.gov

Steven Myint
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
steven.myint@jpl.nasa.gov

Jeffrey Biesiadecki
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
jeffrey.j.biesiadecki@jpl.nasa.gov

Philip Romano
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
philip.j.romano@jpl.nasa.gov

***Abstract*— The Perseverance Mars rover has demonstrated a new onboard global localization capability that has the potential to significantly improve long-distance autonomous navigation [1]. It allows the rover to precisely determine its position by comparing panoramic images from its navigation cameras with high-resolution orbital maps, and use that knowledge to reset its onboard position uncertainty model back to nearly zero meters. Leveraging a fast, commercial off-the-shelf (COTS) Snapdragon 801 processor from the Ingenuity Helicopter Base Station contained within the rover body, the system performs complex image-processing tasks to accurately determine the rover's position relative to orbital maps. This approach of diverting compute-intensive operations to a faster but non-radiation hardened CPU overcomes the limitations of the rover's far slower, radiation-hardened primary processor, and is a paradigm shift in planetary robotics.**

We focus on the characterization, implementation and mitigation strategies required to use a non-radiation-hardened COTS processor in the harsh space environment. We describe the software-based memory checking and fault isolation techniques developed to address memory corruption potentially caused by aging or radiation effects. Furthermore, we present the performance and operational results from the deployed system on Mars, including its accuracy and robustness. The paper also discusses lessons learned during the development and flight deployment process and outline how this technology will fundamentally change future planetary navigation to support more ambitious and efficient autonomous missions by enabling continued use of commercial co-processors even in the presence of degraded memory capabilities.

TABLE OF CONTENTS

979-8-3315-7360-7/26/$31.00 ©2026 IEEE

1. INTRODUCTION

The NASA Mars 2020 mission, which includes the Perseverance rover and carried the Ingenuity helicopter, landed on Mars on February 18, 2021. It is collecting Martian samples for a subsequent mission to return them to Earth for scientific analysis. It recently collected the Chevya Falls rock sample, which contains organic compounds and mineral features that are considered "potential biosignatures," offering the strongest evidence yet of possible ancient microbial life on the planet. The farther it can drive between sample collection locations, the more diverse the samples can be.

Perseverance is the first NASA Mars rover that can drive using its autonomous driving capabilities (Hazard Detection and Avoidance, Visual Odometry) at close to its maximum electromechanical speed. As a result, its self-driving autonomous navigation system has become its primary means of driving: it has been used to execute or evaluate 90% of the total distance traveled [2]. As of this writing, Mars solar day (sol) 1775 for the mission (16 February 2026), the farthest distance it has driven without human review is 699.9m over three sols, which is a planetary rover record [3].

As the rover drives, the uncertainty regarding its actual position on the orbital map grows. This added uncertainty can result in drives ending early and block a narrow passage [1]. This can only be alleviated by re-localizing the rover's current location against the orbital map. Currently, human mapping specialists re-localize the rover before every drive plan (manually performing *global localization*). This requires a complete communications cycle, and the need for it limits how far Perseverance can autonomously navigate without ground-in-the-loop.

As a result, the maximum duration of any autonomous drive plan without human review has only been three Martian sols.

After several hundreds of meters, rover position uncertainty will grow to such an extent that it is impractical to navigate typical Martian terrain without using global localization to eliminate the accumulated uncertainty. This process is currently performed manually on Earth: rover operators downlink a panorama of stereo image pairs from the end of the drive and match them to an orbital map. Autonomous onboard global localization would eliminate the need for frequent communication with Earth, removing the limitation on drive distance due to uncertainty growth [1].

We implemented the core global localization image processing on the faster Snapdragon 801 processor, and updated the rover's flight software to enable its use during nominal operations [1]. But after deploying the software to Mars for an initial checkout, several issues arose with the non-radiation hardened CPU that had to be addressed.

This paper provides an outline of the development and successful deployment of our new, robust framework for performing onboard global localization system using a non-radiation hardened coprocessor on Perseverance. It describes:

- *The nature of radiation errors facing the non-radiation hardened Ingenuity helicopter base station COTS processor, the flagship Perseverance mission rover's co-processor in the harsh space environment.*
- *Implementation and mitigation strategies developed to enable use of the non-radiation hardened processor on mission-critical calculations.*
- *Software-based memory checking and fault isolation techniques developed to address memory corruption potentially caused by aging or radiation effects.*
- *The first demonstration using the non-radiation-hardened Ingenuity Helicopter Base Station as a rover co-processor.*

Beyond Perseverance, the absolute position estimation provided by global localization will be a key enabler for future planetary robotic missions. Lunar rover missions like Endurance [4] aim to traverse many kilometers across the Moon largely in Permanently Shadowed Regions with limited communication. The proposed system and its first application contributes a critical capability in the broader context of increasing demands for long-range autonomy in planetary robot navigation. And our planned use of the COTS processor as an integral mobility subsystem marks the first time a non-radiation hardened processor will actively support Mars Rover robotic operations.

2. ONBOARD GLOBAL LOCALIZATION

As described in [1] planetary mobile robots such as Perseverance and Ingenuity accumulate position uncertainty as they move. This is due to several factors including sensing and actuation accuracy, and accuracy of position estimation [5].

The general strategy of our new Perseverance application, which can be helpful to many planetary rover applications, is to determine the global location of the rover within an uncertainty envelope on a prior map using attitude knowledge and images taken with a rover-mounted stereo camera pair. In particular, this application performs a great deal of image processing, with raw images and maps making up the bulk of the RAM storage that is susceptible to modification by memory errors. While our algorithms for matching images are inherently robust to changes in individual pixels in those maps and images, our radiation mitigation approach is also adaptable to other applications that may not have the same characteristics.

Our algorithm for Global Localization is described in detail in [6], and the overall operations implementation strategy is described in [1]. We summarize the algorithm here to provide context for the kinds of operations being performed on our COTS processor.

Position uncertainty growth has typically been a problem for long drives (greater than 500m). Since the duration available for driving is limited each sol, these drives occur over multiple sols on Mars. Therefore to simplify operations onboard localization is planned to run just once per sol at the end of all driving, along with the complementary SunFind activity which reduces attitude uncertainty [7].

We model rover localization as an image registration problem between the rover images and the orbital map. With the SunFind activity running before global localization, the registration problem is simplified to a search over 2D translation.

Similar to current human-based approaches, we first produce a top-down rover stereo orthomosaic. The rover acquires a 360^o panorama of NavCam stereo pairs around the rover, typically five overlapping pairs. The images are radiometrically corrected to remove vignetting and exposure time differences, and each pair is stereo-matched using semi-global block matching to produce a point cloud. The point clouds are projected into a 2D grid that matches the resolution of the appearance map (25cm) and a Digital Elevation Map (DEM) (1m).

The rover stereo orthomosaics are then matched to the orbital map using a template-matching approach. The correlation scores from the appearance map and DEM are summed together, and the minimum score is converted into a delta position update.

The orbital map is a digital terrain model (DTM) and orthomosaic of HiRISE images produced by the USGS. HiRISE is the highest resolution pushbroom camera orbiting Mars with around 25cm resolution, mounted on the Mars Reconnaissance Orbiter (MRO).

In short, much of the Global Localization algorithm relies on image processing operations. As such, the bulk of the RAM storage used by the algorithm is comprised of raw greyscale pixel values from rover images or the terrain, resampled versions of those images viewed from overhead, and map intensity and height information. Map and image processing algorithms generally are robust to small variations in intensity, so any individual bit-level changes to pixel intensities will typically make only a small numeric impact on the overall results.

Onboard Global Localization Performance

Our approach achieves sub-meter accuracy in global localization performance on a real dataset from Mars of all the drives for which data was available [6]. Ground-based localization has been performed for Perseverance, manually on the ground by human localization experts, since the beginning of the mission.

Our approach achieves 0.36m accuracy across 264 panoramas with no significant outliers and 99% of results coming within 0.93m of human-generated ground truth. The mode is around one-pixel error in the orbital map (0.25m) [6]. We assume

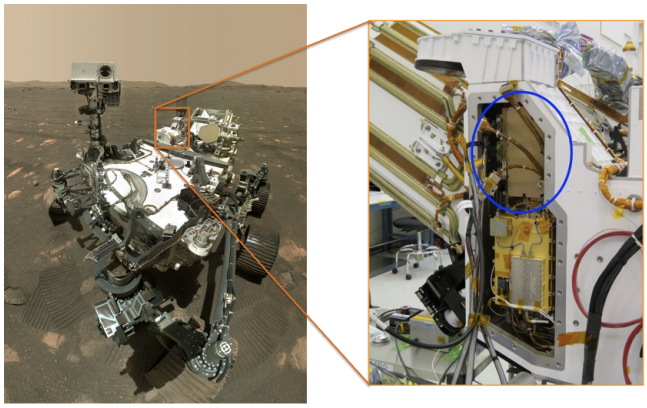

Figure 1. Location of the Helicopter Base Station on the Perseverance Rover. Its Snapdragon 801 processor is connected to the Rover's RAD750 by a serial cable, with its maximum bandwidth currently set to 10KB/s. Image courtesy [1].

a 30m search radius, which is based the max expected uncertainty after a long single-sol drive, though a larger search radius up to 100m does not significantly impact accuracy.

3. COTS Rover Co-Processing

Making Global Localization work onboard required us to find extra compute cycles. Perseverance's main computer is fundamentally specialized for reliability through hardware-based radiation mitigation. The difficulty, cost, and engineering compromises of developing and testing radiation-certified computers result in the computational capability of the RAD750 lagging decades behind modern computers. Fortunately, Perseverance also has a Snapdragon 801 computer, courtesy of the *daring and mighty* Ingenuity mission [8]. Perseverance uses it to relay data to and from Ingenuity via a Snapdragon 801 processor and radio mounted on the rover's Heli Base Station (HBS), shown in figure 1.

As described in [1] the Helicopter Base Station has several advantages as a powerful co-processor for Perseverance. The performance vs. reliability trade-offs between the RAD750 and Snapdragon 801 computers encourage a co-processing regime, where the RAD750 handles critical real-time processing, and the Snapdragon runs computationally intensive programs.

A need for radiation mitigation was clear even before Global Localization was deployed on Perseverance on Mars. The BAE RAD750 spec sheet shows an expected 0.02 bits corrupted per day given a radiation environment at 90% of worst-case GEO. In contrast, a CREME-MC simulation for the Snapdragon 801 using data from a previous study [9] shows roughly 4314.3 bits expected to be corrupted per day under similar conditions. Thus, despite the increased speed of the Snapdragon 801, ensuring that results from this COTS computer are correct in the presence of radiation is a challenge. A radiation management approach was therefore developed as described in [1] and summarized below.

Figure 2 shows the global localization steps and figure 3 shows activities in a multiple sol drive plan.

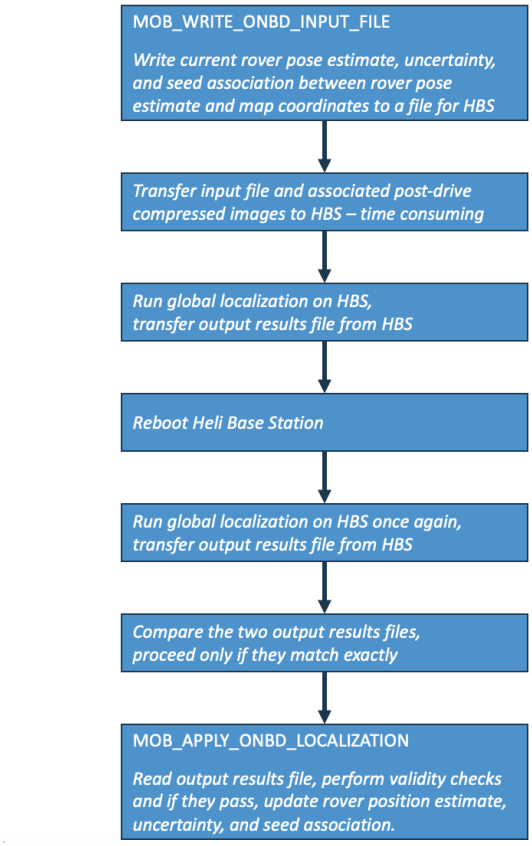

Figure 2. Steps to "Perform Onboard Global Localization[1]"

Checksums use the GNU coreutils `md5sum` program.

For hash collision to occur due to radiation, where 2 files can resolve to the same checksum [10], a specific set of bits must be affected by SEUs when the checksum is run, a virtually impossible event given the built-in ECC on the eMMC chip and the design of the ext4 filesystem onboard. Furthermore, MD5 is computationally inexpensive, which has minimal impact on overall runtime.

Newer C++ standards provide memory safety and type features that are not available in older C++ standards. However, the software on the HBS is relatively outdated, with the last software update having been delivered in 2017. As future missions will have newer computers with differing C++ libraries, we need a way to provide a generalized, consistent standard library for developers to use onboard the HBS.

To this end, we use LLVM, a widely-supported, open-source set of compiler tools that outputs optimized binaries from a variety of frontends, including C and C++ [11]. One key feature of LLVM is the unified intermediate representation (IR) that all optimizations are run on. By having one unified IR for all architectures, any language frontend that can be translated to LLVM IR is able to compile to any architecture. This allows us to easily cross-compile our project to a variety of targets, including x86_64 workstations as well as the ARM32 Krait architecture used by the HBS.

Another subproject of LLVM is libc++, which is a standards-compliant C++ function library built on top of LLVM. This

Feature	Perseverance RAD750	HBS Snapdragon 801
CPU Frequency	133 MHz (x2)	4 cores, 2.36 GHz
Memory	128 MiB ECC RAM	1.55 GiB non-ECC RAM
Storage	2GB validated ECC	32GB unvalidated ECC
Operating System	VxWorks 6.7	Linux 3.8
Architecture	PowerPC	ARM
Hardware Accelerators	FPGA (VCE)	GPU, DSP
Radiation Hardened	Yes	No
SEEs/bit/second (sim)	7.80716e−14	2.49620e−10
SEEs/day/device (sim)	13 (corrected)	4313 (uncorrected)

Table 1. Comparing Perseverance's main RAD750 computer to the Heli Base Station co-processor.

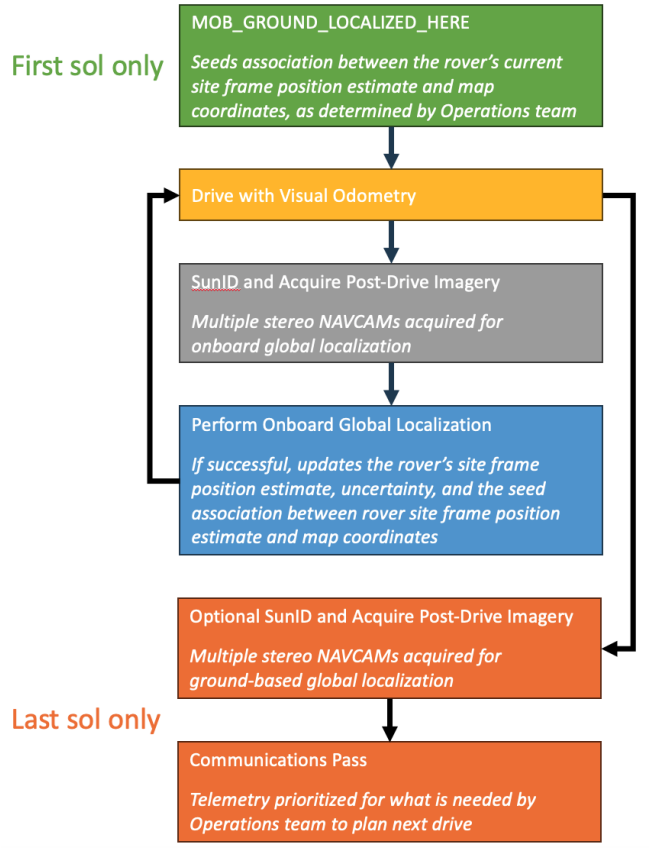

Figure 3. Global localization activities on a multi-sol plan

library is separated into 2 main parts: the core library, which defines high-level C++ language features; and the Application Binary Interface (ABI), which provides architecture-specific details such as exception handling and datatype sizes. By using the system ABI, while providing our own build of the libc++ core library, we can provide developers with modern C++ features while also keeping the binary size minimal, keeping uplink bandwidth consumption low. This is due to LLVM's ability to strip unused functions, which leaves *just the subset of the C++ library being used by the program* in the binary.

A majority of the time for running global localization is spent in transferring images and data from the primary RAD750, known as the RCE, to the HBS over the slow 10KB/s serial link as described in [1] . However, much of this can be run in parallel with other ongoing rover activities. So although the transfer takes 30 minutes, the entire activity can be safely run in parallel with other operations.

4. Flight Deployment

Because Perseverance is a science-critical multi-billion dollar mission, the main constraint is *risk*. New capabilities, however beneficial, need to be carefully proven to be safe. As described in [1] to achieve deployment to flight, our global localization project was broken into three distinct phases: software sandboxing demonstration, modular flight demonstration, and closed-loop flight demonstration. *Phase 1, Software Sandboxing* successfully demonstrated the safety of the software sandbox on the co-processor. *Phase 2, Modular Flight Demonstration* used previously acquired Mars data for the successful onboard demonstration. The resulting localization position was downlinked but not used to update the Rover's position.

Phase 3, Closed-loop Flight Demonstration

Phase 3 of this project includes rover flight software changes which interface between the Rover's main computer and the Heli Base Station. This enables the rover to take a panorama of NavCam images and send data to the Base Station for processing. Then, the global localization algorithm is run, and the result is sent back to the Rover's main computer. Finally, if the result passes multiple validity checks the rover flight software updates its onboard knowledge with the new position and uncertainty, and continues driving. These checks are summarized in Figure 4 from [1].

Phase 3, Part 1: New RCE Software and Orbital Maps— It took some time to update the flight software on the rover and corresponding ground software on Earth. In particular, the Ground Data System update process is slow and took nine months to update the command dictionary and perform a regression test. Orbital maps of the location the rover was heading to, called Lookout Hill, were uploaded.

This part ran on sol 1347 (2 December 2024).

Phase 3, Part 2.1: Shadow-mode Closed-loop Flight Demonstration

Since this phase updates the Rover's true position, which is mission-critical, the first test of the full global localization system was in a "shadow mode" where the updated position was not incorporated. But the results of running the closed loop activity in shadow mode were unexpected. It failed to transfer the 1st NAVCAM image from the panorama image set. All the other files and the executable image files transferred just fine prior to the failure. This was due to a

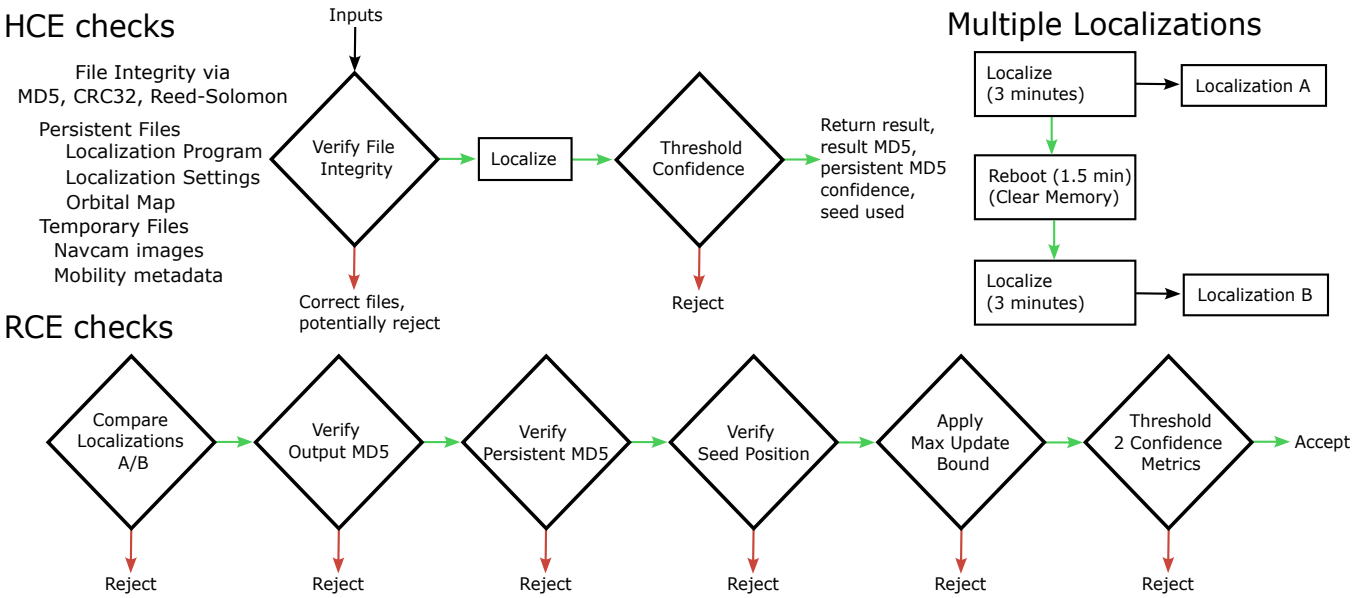

Figure 4. Checks done by both the HBS and the RCE to both mitigate radiation-induced SEUs and catch common possible localization errors. Under our probabilistic error model, the chance of performing an incorrect localization due to radiation errors is near zero. The localization algorithm itself is 100% accurate on all benchmarked data to within 1 meter [6]). Timeouts (parenthesized) on each localization run catch radiation errors that may cause the HBS to stall.

checksum error. The leading hypothesis was that a SEU had corrupted the first NCAM file transfer. This was based on a review of the HBS anomaly products and VSTB testing, where we successfully transferred the exact files downlinked off of the flight vehicle without issue. A low-effort method of either confirming or rejecting the SEU hypothesis in flight was conceived which would be a partial re-run of Part 2.

This part ran on sol 1360 (16 December 2024).

Phase 3, Part 2.2: Shadow-mode Closed-loop Flight Demonstration

A low-effort method of either confirming or rejecting the SEU hypothesis in flight was conceived which would be a partial re-run of Part 2.

In this run, all of the file transfers from the Rover Compute Element to the HBS succeeded as expected, supporting the hypothesis that the previous error was spurious and radiation induced and thus not a persistent issue.

Two independent runs of global localization both converged on a solution with excellent accuracy, which was very exciting! They did however differ by 1mm due to what was hypothesized to be another bit flip that impacted one of the NAVCAM panorama stereo imaging products. This kind of radiation event is something that the non-radiation-hardened HCE is susceptible to and is why we always run the processing at least twice and compare the results. Seeing this comparison work as expected in flight was good confirmation that our onboard checks were working correctly, but even a 1mm difference meant that the results were not accepted as identical, and therefore not correct.

This part ran on sol 1398 (24 January 2025).

We re-ran the test to ensure that the same stereo issue didn't happen twice in a row. Instead we found evidence for stuck or weakened bits in HBS memory, which experienced minor random data corruption across six runs on Mars.

HBS Memory Testing

Stuck or weakened bits became the leading hypothesis. Stuck bits are bits in memory that are stuck at a 0 or 1; weakened bits are bits in memory that can not reliably maintain a 0 or 1 state.

These errors occur as a DRAM chip is irradiated over time [12]. DRAM chips store bits in arrays of capacitors, where a charge in it represents a logical "1", and no charge a logical "0." However, these capacitors slowly leak the charge stored in them over time. To mitigate this, DRAM cells are refreshed constantly (usually every 64 ms) by reading the data stored in the cell, and re-writing the data stored in them. Long-term exposure to radiation will increase the leakage current of capacitors on the chip. This decreases the retention time of the capacitors, meaning that the charge can leak out before the refresh cycle occurs. This manifests as weakened or stuck bits.

A simple memory test was developed to characterize these bit errors. We allocate a large 1GB memory buffer in HBS RAM, and we write different patterns to it and read them back. If the pattern we read doesn't match the pattern we wrote, we record its (virtual) memory address. We run the memory test 3 times with different delays between reads and writes (1st test: 0 seconds, 2nd test: 5 seconds, 3rd test: 30 seconds). We use additional data from the HBS during downlink to map these (virtual) memory addresses to their physical memory address in HBS RAM.

On sol 1539 (18 June 2025) the first run of the memory testing program was a success. It detected 15 physical addresses in HCE RAM that showed weakened/stuck bits.

Many of the same physical addresses reappeared between runs. We detected 2 physical addresses in the first run, 14 in the second run, and 14 in the third run. The first 2 physical

addresses appear in all 3 runs. The 2nd and 3rd runs share 13 of the same physical addresses. All of the stuck bit values were 0, as opposed to an even distribution of 0's and 1's.

There was at least one bit that didn't register as 'stuck' until we read/write to it a couple times. For example, in our first memtest run, the bit at one virtual address didn't show up in our 1st and 2nd iteration through memory, but it showed up in the 3rd iteration. After that, it showed up every time in our second and third memtest run (with the longer delays), in all 3 iterations.

The HBS temperature plot is shown in Figure 5. HBS temperatures stayed relatively low (less than 60^{o}C) during the test, compared to temperatures we reach during a global localization activity. This was interesting, because we had hypothesized some connection between temperature and the number of memory errors due to stuck/weakened bits. However, there were also multiple global localization runs where we encountered memory errors at the lower temperatures. This test provides evidence that stuck/weakened bit memory errors can be present at relatively low temperatures.

We re-ran the activity on sol 1554 and and detected 20 weak bits in HBS RAM. It was an exciting result. It provided evidence that there were a handful of weak bits in HBS RAM. It provides a clear path forward and explained previously unexplained HBS errors during the Ingenuity mission as well.

On sol 1556 (6 July 2025) we then ran the HBS memory test preceded by HBS global localization activity. It detected 27 weak bits in HBS RAM. This confirmed that there were again a handful of weak bits, many of which were at fixed physical addresses.

HBS RAM weak bit quarantining

A HBS program was then developed to find and hold a list of Page Frame Numbers (PFNs) in HBS RAM that are known to have weak bits. The list is informed by the results from previous runs of the memory test program, and will be maintained as an onboard text file that can be updated with new vulnerable addresses as more are detected over time.

The program consists of a parent process that executes the memhold code in a child process. While PFNs are held, the parent process will return a 0 error code. If any failure occurs, the process exits with a non-zero error code.

The program also runs the same memory test code used in the memory test to find weakened bits, allowing us to continue detecting new suspect bits in memory whenever we use memhold alongside GBLC. So it has been incorporated into the nominal execution of global localization processing.

On Sol 1605 (25 August 2025) we ran the memhold program 3 times with 1 success and 2 failures:

- Run 1: HBS encountered an error about 10 minutes into running the program on the HBS, which would have been near the end of the script. The error was that the HBS was marked *sick* before transferring the log files. Marking hardware sick is standard protocol in the rover flight software when an unexpected signal is received from hardware that is outside the scope of programmed fault responses, and prevents it from being used again until that status has been cleared by a spacecraft command.
- Run 2: Perfect run. Run 2 was scheduled immediately after Run 1. The memory holding program jailed 29 pages, including 1 new unlisted bad page by testing memory. Running the memtest program 3 times afterwards showed 0 memory errors, because they had all been quarantined!
- Run 3: HBS went sick again, but this time only 1.5 minutes into running on the HBS. The memory holding program exited early and the HBS only went *sick* when the RCE attempted to transfer memtest result files from the HBS, which didn't exist because memtest never ran.

After these tests the memhold script was updated to:

- Reduce memory allocation from 940 MiB to 850 MiB.
- Reduce memhold iterations from 3 to 1.
- Stop using the 'sync' flag which attempts to mix up memory allocation by writing cached data to the disk.
- Use the 'verbose' flag to record additional debug info to the log file

With this update we did 3$\times$ runs again. All three runs succeeded this time.

CPU Throttling

The Snapdragon CPU can be prone to overheating when intensive calculations are run. If the temperature exceeds a preset limit (currently 90^{o}C) processing will halt, and the HBS will be marked *sick*. Fortunately a CPU throttling capability exists that will cause processing to slow down as the temperature increases. We have enabled this capability to mitigate against this potential environmental complication.

Mitigating future weak bits

Unfortunately masking known weak bits in RAM has not been sufficient to eliminate all memory issues. Our tests have revealed that new weak bits can still appear in areas of RAM that were not known to have problems, resulting in ongoing differences from one run to the next.

So we have updated our strategy for accepting results as valid. Our initial approach was to run the GBLC process twice to completion, each time writing overall results to an onboard file. We then compared the checksums of those files; any difference in checksum would cause us to declare the operation unsuccessful. But since so much of the RAM used in the GBLC processing corresponds to image pixel intensities, the *impact* of most weak bits on the overall result is typically minimal. The correlation score comparing rover-taken images against onboard map images changes, but only by a tiny fraction of a percent. And the resulting localization offset vector components typically change by less than 1cm. So we decided to accept each pair of checksum and localization position offset results as a "close enough" match so long as certain metrics agree to within a parameterized threshold.

The logic implementing the acceptance check runs on the rad-hardened RCE processor for safety. However, our new threshold-based checks are complex enough that they cannot be encoded directly in spacecraft commands, and instead must be at least partly implemented as changes to the RCE FSW. Normally the process for updating RCE FSW takes months or years to complete. But small changes can sometimes be implemented and validated in far less time as a "hot patch," which changes the FSW running in RAM without modifying the copy stored in persistent non-volatile memory [13]. In this case we created a new C language function which uses existing FSW to parse two GBLC result files written in YAML format to compute the absolute differences between three pairs of numbers: Uncertainty values, and 2D and 3D

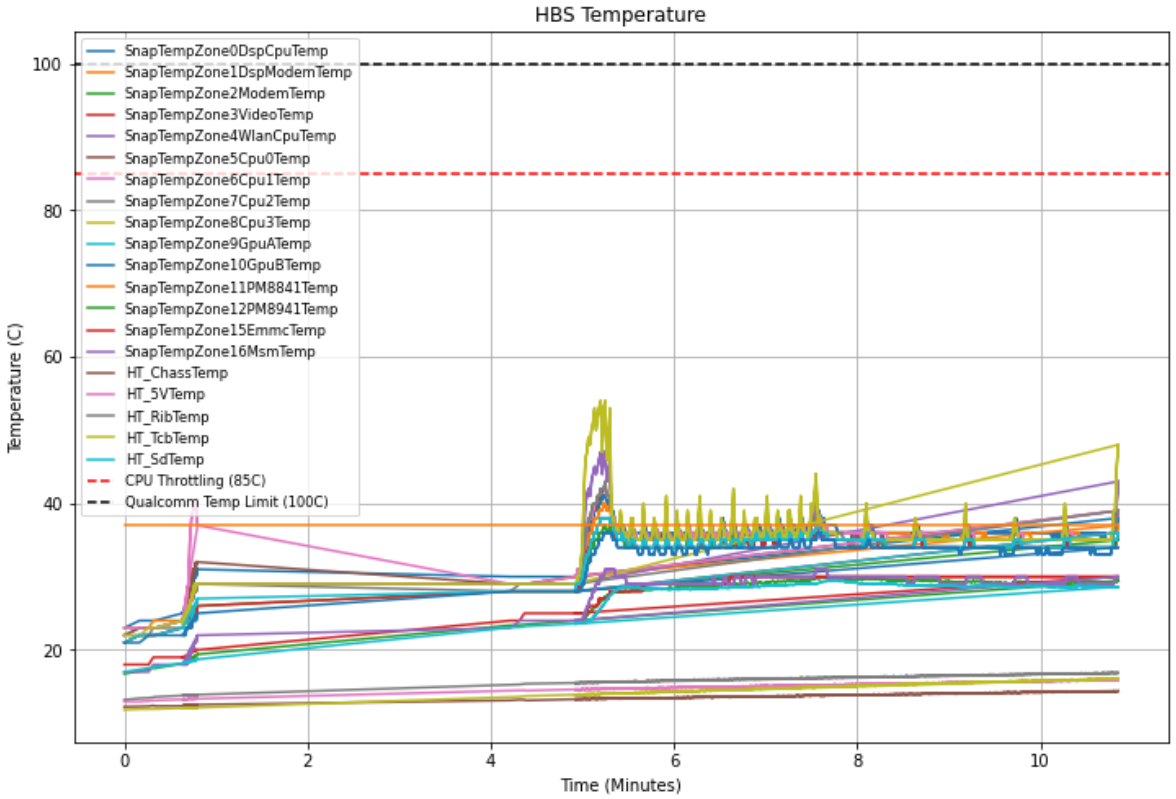

Figure 5. HBS temperature plot from the memory testing program on sol 1539 (18 June 2025).

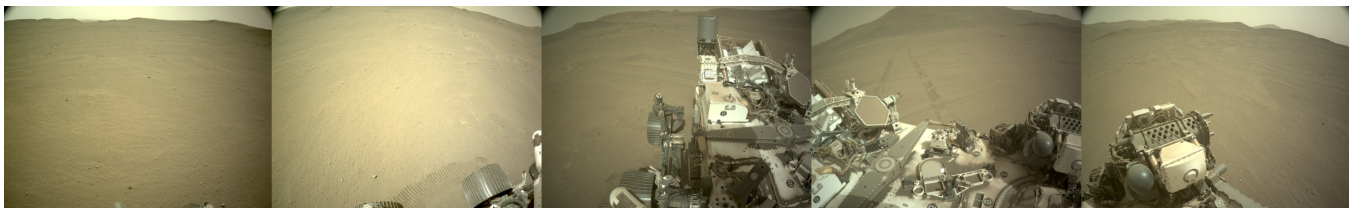

Figure 6. Sol 1762 (2 February 2026) post-drive Navcams collected during a successful test of our error-correcting framework on Mars. The Censible algorithm [6] worked well even in this relatively featureless terrain.

distances between localized positions. Each of those numbers has an independent parameterized tolerance (all initially set to 10cm), and any test can be disabled if desired by specifying a tolerance of -1. If any of those three numbers do not agree according to their parameterized tolerance, the computation is characterized as having failed.

To implement the hot patch, we upload a compiled version of the function as a `.o` object file, and run VxWorks shell commands to load the function into memory. We invoke the function by replacing the functionality of an existing utility test command in the RCE FSW, calling the new function instead of the nominal implementation of the command handler. The return status of the utility command tells us whether the two YAML files were successfully read, and the value of a spare RAM parameter communicates whether all the threshold constraints were met.

Rarely, something happens that prevents GBLC from producing an output file. That case is detected by checking the return status of the utility command, and if it has failed that will result in another attempt at running the GBLC process again.

In short, we (1) avoid using any pages that are known to contain weak bits, (2) re-run the processing when it fails to produce an output file, (3) throttle the CPU if we get close to overheating, and (4) compare two runs, relying on thresholding to accommodate previously unknown weak bits that result in small deviations to the numbers that do not significantly impact the validity of the overall result.

Successful Integrated Demo

The entire process ran successfully on Perseverance on sol 1762 (2 February 2026). After a 159 meter autonomous drive, five stereo Navcam image pairs were collected and transmitted to the HBS (Figure 6). The images were orthographically reprojected into an overhead view and compared against the orbital map data already stored onboard. The first execution of Global Localization failed due to a checksum mismatch in one of the Navcam images. That caused the retry strategy to be invoked, and the next two iterations both succeeded with identical answers. Snapdragon CPU temperatures remained in bounds throughout the processing (Figure 7). Since multiple runs succeeded and agreed to within the required tolerances, the new location and reduced

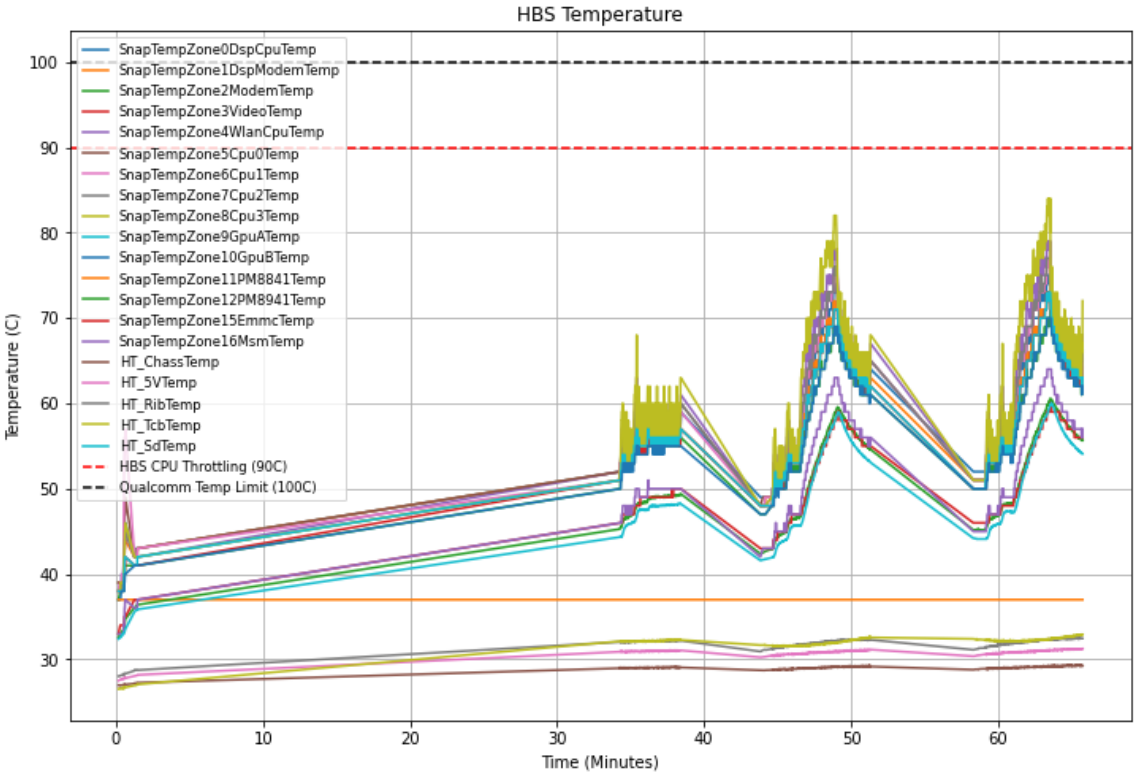

Figure 7. Sol 1762 (2 February 2026) Snapdragon CPU temperature profile. CPU throttling successfully kept the temperature below the 90^{o}C limit.

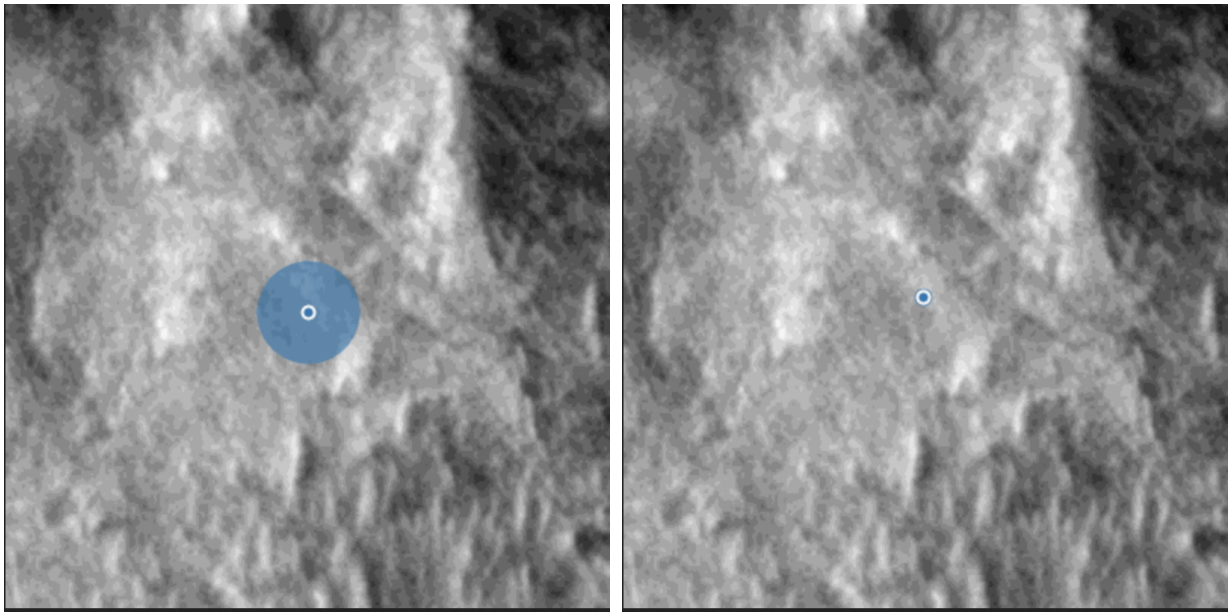

Figure 8. Sol 1762 (2 February 2026) Initial and final rover position and uncertainty estimates. The position was adjusted by 1.27m and the uncertainty (radius of the blue circle) shrank from 8.4m to 1.5m.

uncertainty were directly incorporated into the RCE FSW pose estimate (Figure 8).

The full process succeeded again on sol 1775 (16 February 2026). After a 36 meter human-directed drive, we collected five stereo pairs and compared them to the orbital map (Figure 9). In this case there were no new memory failures, the process converged twice in a row with identical answers so no third attempt was needed. A small correction of 35 cm was made and applied to the onboard position estimate, and the accumulated uncertainty was reduced.

We are now in the process of integrating Onboard Global Localization into our tactical operations processes, and look

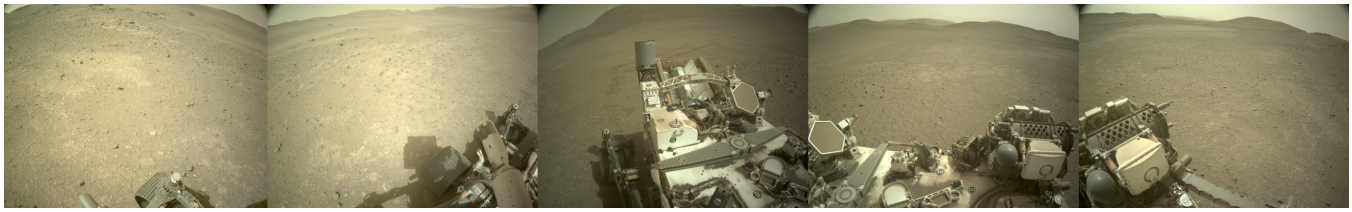

Figure 9. Sol 1775 (16 February 2026) post-drive Navcams collected during another successful test of our error-correcting framework on Mars.

forward to seeing it used to extend autonomous drive distances to new record-setting lengths.

5. Related Work

The concept of deploying non-radiation hardened commercial processors into a space environment has been discussed for some time [14], [15]. But software-based radiation hardening is a relatively new field, as actually bringing COTS hardware onboard space missions is a recent emerging trend. Previously, radiation hardening tended to be done at the chip level, which afforded more protection against all forms of both SEE and TID effects. However, radiation-hardened electronics rely on specialized older process nodes, redundant circuitry on the board to account for errors, and extra check circuits. As such, the performance of these chips lag behind the COTS state-of-the-art by about 20 years on average.

Application-specific radiation mitigations exist for deep learning [16], [17], PDE solving [18], and numerical analysis [19]. However, they rely on specific mathematical properties of the target computation, and cannot be easily adapted for general use.

For flight control, triple modular redundancy remains the state-of-the-art. SpaceX uses 3 COTS non-radiation-hardened processors in lockstep triple modular redundancy for the Falcon 9 rockets [20]. Software-only methods for flight control software that also rely on lockstep execution and replication have also been proposed [21]. This can be done at the level of a CPU core rather than that of an entire compute element, but doing so incurs additional risk of radiation damage in shared components [22].

In high-performance computing, which has no real-time deadlines, checkpoint-and-restart is often used in lieu of replication and voting. Such schemes involve storing checksums of critical memory that are recomputed every time memory is written and verified every time the memory location is read [18], [23], [24], [25]. This can be refined and optimized for non-real-time, compute-intensive tasks, where errors in memory and CPU cache can be accounted for as well [26].

6. Conclusion

We demonstrate how the COTS Helicopter Base Station can be used as a fast co-processor for the rover. The protocol for communication and quarantining of memory errors that we developed between the rover RCE and the HBS can be extended to a number of other applications in the future.

The combination of software sandboxing, memory page reservation, active memory testing, and implementation of sanity checks (including thresholding) on the radiation-hardened CPU provides the first demonstration incorporating non-radiation-hardened processing into the safety systems of a NASA surface spacecraft.

The presence of the more capable HBS Snapdragon 801 co-processor is what made it possible to deploy capabilities like onboard global localization. Phase 2 of global localization ran completely onboard the Perseverance rover on Mars in only 32s. In addition, use of an isolated co-processor limits the scope of any errors to its results, reducing risk and impact to the overall rover mission. Development on the HBS is also not restricted by existing rover RCE flight software practices. The combination of the radiation hardened RAD750 and the HBS Snapdragon allows separable software based radiation mitigations. For surface science missions, running the process multiple times is a very reasonable option since it is fast and the localization activity has no have real time constraints.

Lessons Learned

Develop and deploy in phases to reduce risk and limit impact—Much of the development was done over two summers. From the start we planned for three distinct phases of deployment to flight. Each focused on retiring one of the main elements of risk. Phase 1 demonstrated that global localization could be sandboxed on the HBS from HBS flight software. Phase 2 demonstrated the performance of global localization software in isolation on the HBS, all the data including the rover panorama were from a previous position and sent from the ground. Phase 3 was performed in shadow mode where we took rover panoramas on the rover and transferred them to the HBS and ran global localization, but sent the results to the ground without updating the rover pose. After we have tested the accuracy and performance of the results it will be fully deployed. The phases also served as a mechanism to help focus and re-prioritize if challenges, such as the memory errors described in the paper, are encountered.

Plan for software approaches for mitigating memory errors—When using non-radiation hardened COTS processors in planetary applications, there is a potential for memory errors. Developing a solution with built in checks allows for safe operations by detecting problems before accepting results.

Future Work

Our approach could be used for onboard global localization for Lunar rover missions such as Endurance. It could also be used for automated ground based localization by Lunar rover missions that may have limited onboard processing, but do have bandwidth to transmit panoramas to Earth. It also could be augmented for aerial applications.

Our sandboxing environment shows that HBS can be used as a coprocessor for other rover applications as well. A prime candidate among potential Mars 2020 updates is eliminating the ground in the loop cycles needed for placing arm turret-

mounted science instruments PIXL and SHERLOC safely into an abrasion patch [27]. That process currently takes 4 sols due to the need to reassess terrain clearances after the abrasion has completed, but by moving some of that processing onboard the duration could be reduced to 2 sols.

ACKNOWLEDGMENTS

This work is supported by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the with the National Aeronautics and Space Administration (80NM0018D0004). We thank the Ingenuity helicopter and Perseverance rover teams, including Robert Hogg, Danny Lam, Tim Canham, Katie Stack-Morgan, James Hazelrig, James Bhiel, Art Rankin, John Michael Morookian, Darwin Chiu, Nihal Dhamani, Sammi Lei, Brittany Seto, Art Thompson, Steve Lee, Daniel Zayas, Loren Jones, Douglas Sheldon, Cathleen Harris, and Sergeh Vartanian for their contributions and support of this work.

Pre-Decisional Information – For Planning and Discussion Purposes Only.

REFERENCES

[1] V. Verma et al., "Enabling long & precise drives for the perseverance mars rover via onboard global localization," in *2024 IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2024.

[2] M. Maimone et al., "Roving on the Edge: Robotic Operations Power Perseverance's Ascent of Jezero Crater Rim," in *48th Annual AAS Guidance, Navigation and Control Conference*, Berckenridge, CO, USA, Feb. 2026.

[3] V. Verma et al., "Autonomous robotics is driving perseverance rover's progress on mars," *Science Robotics*, vol. 8, no. 80, 2023.

[4] The National Academies of Sciences, Engineering, and Medicine: Committee on the Planetary Science and Astrobiology Decadal Survey, *Origins, Worlds, and Life: A Decadal Strategy for Planetary Science and Astrobiology 2023-2032*. Washington, DC USA: The National Academies Press, 2023.

[5] D. S. Bayard et al., "Vision-based navigation for the nasa mars helicopter," in *AIAA Scitech 2019 Forum*, 2019, p. 1411.

[6] J. Nash et al., "Censible: A Robust and Practical Global Localization Framework for Planetary Surface Missions," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2024.

[7] K. Ali et al., "Attitude and position estimation on the mars exploration rovers," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, 2005, 20–27 Vol. 1. DOI: 10.1109/ICSMC.2005.1571116.

[8] B. Balaram et al., "Mars helicopter technology demonstrator," in *2018 AIAA atmospheric flight mechanics conference*, 2018, p. 0023.

[9] S. M. Guertin, W. P. Parker, A. C. Daniel, and P. Adell, "Recent SEE results for snapdragon processors," in *2019 IEEE Radiation Effects Data Workshop*, 2019, pp. 1–5.

[10] J. Black, M. Cochran, and T. Highland, "A study of the md5 attacks: Insights and improvements," in *Fast Software Encryption: 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers 13*, Springer, 2006, pp. 262–277.

[11] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *International symposium on code generation and optimization, 2004. CGO 2004.*, IEEE, 2004, pp. 75–86.

[12] J. A. Pellish, "Radiation 101: Effects on hardware and robotic systems," Tech. Rep., 2015.

[13] E. Benowitz and M. Maimone, "Patching flight software on Mars," in *Workshop on Spacecraft Flight Software*, Laurel, MD, USA, Oct. 2015. [Online]. Available: http://flightsoftware.jhuapl.edu/files/_site/workshops/2015/.

[14] D. Chen et al., "Reliability and availability analysis for the jpl remote exploration and experimentation system," in *Proceedings International Conference on Dependable Systems and Networks*, 2002, pp. 337–342. DOI: 10.1109/DSN.2002.1028918.

[15] J. Samson, E. Grobelny, M. Clark, S. Driesse-Bunn, and S. Van Portfliet, "Nmp st8 dependable multiprocessor: Technology and technology validation overview," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010, p. 1495.

[16] J. Kosaian and K. Rashmi, "Arithmetic-intensity-guided fault tolerance for neural network inference on gpus," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[17] W. Zheng, B. Xu, J. Gu, and H. Chen, "Save: Software-implemented fault tolerance for model inference against gpu memory bit flips," in *2025 USENIX Annual Technical Conference*, USENIX, 2025.

[18] M. Salloum, J. R. Mayo, and R. C. Armstrong, "In-situ mitigation of silent data corruption in pde solvers," in *Proceedings of the ACM Workshop on Fault-Tolerance for HPC at Extreme Scale*, 2016, pp. 43–48.

[19] D. Nicholaeff, N. Davis, D. Trujillo, and R. Robey, "Cell-based adaptive mesh refinement implemented with general purpose graphics processing units," *Tech. Rep. LA-UR-11-07127*, 2012.

[20] H. Leppinen, "Current use of linux in spacecraft flight software," *IEEE Aerospace and Electronic Systems Magazine*, vol. 32, no. 10, pp. 4–13, 2017.

[21] A. G. Schmidt, M. French, and T. Flatley, "Radiation hardening by software techniques on FPGAs: Flight experiment evaluation and results," in *2017 IEEE Aerospace Conference*, IEEE, 2017, pp. 1–8.

[22] C. A. Skeggs, "Vivid: An operating system kernel for radiation-tolerant flight control software," Ph.D. dissertation, Massachusetts Institute of Technology, 2022.

[23] C. Borchert, H. Schirmeier, and O. Spinczyk, "Compiler-implemented differential checksums: Effective detection and correction of transient and permanent memory errors," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2023, pp. 81–94.

[24] D. Fiala, F. Mueller, and K. B. Ferreira, "Flipsphere: A software-based DRAM error detection and correction library for HPC," in *2016 IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, IEEE, 2016, pp. 19–28.

[25] M. Turmon, R. Granat, and D. Katz, "Software-implemented fault detection for high-performance space applications," in *Proceeding International Conference on Dependable Systems and Networks. DSN 2000*, IEEE, 2000, pp. 107–116.

[26] H. Wang, S. Myint, V. Verma, Y. Winetraub, J. Yang, and A. Cidon, "Radshield: Software radiation protection for commodity hardware in space," en, *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2026.

[27] V. Verma et al., "Robotic operations during perseverance's first extended mission," in *2025 IEEE Aerospace Conference*, Winner of M. Charles Fogg Best Conference Paper Award, Big Sky, MT, USA, Mar. 2025.

BIOGRAPHY

Jeremy Nash *is Group Lead of the Aerial and Orbital Image Analysis group at the Jet Propulsion Laboratory. His recent work includes creating Mars maps for the Sample Retrieval Lander (SRL) Entry, Descent, and Landing (EDL) system, and global localization for the Perseverance rover.*

Vandi Verma *is a Program Manager at NASA JPL, and the Chief Engineer of Robotic Operations for Mars 2020. Robotics capabilities she has worked on are in use on the Perseverance, and Curiosity rovers, and in human spaceflight projects. She worked on the Mars Exploration Rovers and Ingenuity helicopter Technology Demonstration. She led the onboard global localization effort.*

Evan Graser *is currently a deputy lead of the Mars 2020 Robotic Operations Rover Planner team and Strategic Route Planning Team. He has also served as a Systems and Mobility Downlink engineer for the MSL mission, and was previously the Mechansims Technical Authority for the MSL mission. He received his M.S. in Aerospace Engineering from the University of Colorado Boulder in 2017.*

Haoda Wang *is a PhD candidate at Columbia University applying software systems methods to spacecraft software. Previously, he has worked on flight software simulations on Mars 2020, and developed a fuzzer that has caught multiple flight software bugs. He is also a 2024 NSF GRFP fellow and a 2022 DoD NDSEG fellow, and holds a BS in Computer Engineering and Computer Science from the University of Southern California.*

Mark Maimone *is a JPL Principal in Autonomous Planetary Rover Navigation, and Mars 2020 Robotic Operations Deputy Team Chief, Mobility Technical Authority, and member of the Rover Planner and Flight Software development teams. He received NASA Exceptional Achievement Medals for designing and implementing GESTALT self-driving Flight Software for MER and MSL; contributed to the Mars 2020 ENav self-driving Flight Software; served as MSL Deputy Lead Rover Planner, Lead Mobility Rover Planner and Flight Software Lead; developed downlink automation tools for MER/MSL; and is on NASA's Lunar Terrain Vehicle Insight team. Mark has a Ph.D. in Computer Science and was also a post-doctoral researcher in Robotics at Carnegie Mellon University.*

Steven Myint *is a Group Lead in the Robotics section of the Jet Propulsion Laboratory. His recent work include developing flight software for the Mars 2020 rover, applying modern software fuzzing techniques to finding problems in safety-critical flight software, leading the development of SSim (Surface Simulation), the primary simulation tool for Mars 2020 and Mars Sample Return operations, and leading the development of RSVP (Robot Sequencing and Visualization Program) for Mars 2020. In Mars 2020 Robotic Operations, his roles include Rover Planner and Helicopter Integration Engineer.*

Jeffrey Biesiadecki *is a JPL Principal in Flight Software, the M2020 Sampling and Caching Subsystem Flight Software lead as well as a developer of M2020's mobility flight software including the Thinking While Driving design which enables continuous driving by acquiring and processing images during motion. Jeff also developed motor control and real-time mobility flight software for MER and MSL rovers, has been a rover driver on MER, MSL, and M2020, and has also served as Chief Engineer for the Mobility and Robotics Systems Section at NASA's Jet Propulsion Laboratory.*

Gerik Kubiak *is a Robotics Technolgist at the Jet Propulsion Laboratory. He works primarily as a flight software developer, supporting the Sample Recovery Helicopters, COLDArm, and Ingenuity, where he is the flight software lead. He has also worked as a systems engineer support the Mars 2020 Robotic Arm.*