A Hybrid FPGA/Tilera Compute Element for Autonomous Hazard Detection and Navigation

Carlos Y. Villalpando, Robert A. Werner, John M. Carson III, Garen Khanoyan, Ryan A. Stern, Nikolas Trawny
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-7487
Carlos.Y.Villalpando@jpl.nasa.gov

Abstract—**To increase safety for future missions landing on other planetary or lunar bodies, the Autonomous Landing and Hazard Avoidance Technology (ALHAT) program is developing an integrated sensor for autonomous surface analysis and hazard determination. The ALHAT Hazard Detection System (HDS) consists of a Flash LIDAR for measuring the topography of the landing site, a gimbal to scan across the terrain, and an Inertial Measurement Unit (IMU), along with terrain analysis algorithms to identify the landing site and the local hazards. An FPGA and Manycore processor system was developed to interface all the devices in the HDS, to provide high-resolution timing to accurately measure system state, and to run the surface analysis algorithms quickly and efficiently. In this paper, we will describe how we integrated COTS components such as an FPGA evaluation board, a TILExpress64, and multi-threaded/multi-core aware software to build the HDS Compute Element (HDSCE). The ALHAT program is also working with the NASA Morpheus Project and has integrated the HDS as a sensor on the Morpheus Lander. This paper will also describe how the HDS is integrated with the Morpheus lander and the results of the initial test flights with the HDS installed. We will also describe future improvements to the HDSCE.**

TABLE OF CONTENTS

1. INTRODUCTION 1
2. HDS COMPUTE ELEMENT HARDWARE 1
3. HDS SOFTWARE 3
4. MORPHEUS INTEGRATION.....4
5. SUMMARY 5
ACKNOWLEDGEMENTS.....5
REFERENCES.....6
BIOGRAPHY 7

1. INTRODUCTION

Studies such as the Exploration Systems Architecture Study (ESAS) for future missions to planetary bodies have shown that technology must be developed to enable safe and precise landing on unknown and unprepared terrain, and in various lighting conditions. The Autonomous Landing and Hazard Avoidance Technology (ALHAT) Program has worked to develop sensor technology and sensor processing algorithms to be able to map terrain and obstacles autonomously. The project goals for ALHAT [1] are to:

1. Develop sensing technology to map terrain and hazards in all lighting conditions
2. Demonstrate computation needed for autonomous mapping and safe site selection based on data from the sensor in (1)
3. Demonstrate closed-loop avoidance of hazards detected in (2) with integration into the vehicle Guidance, Navigation & Control (GN&C) system.

The example mission for this technology is a landing on either a polar region or an unlit region of Earth's moon of a 3 meter or greater diameter lander, with sensing beginning at 750 to 1000 meters slant range distance from the desired landing area.

To this end, the ALHAT program conducted several studies with numerous field campaigns to determine what the ideal sensors, algorithms, and processing capabilities should be for a terrain sensing/hazard detection system (HDS) [2-10]. The current system consists of a 128x128 pixel flash LIDAR system developed by the NASA Langley Research Center capable of sensing ranges out to 1 kilometer. The LIDAR is mounted on a 2-axis gimbal developed by PVP Advanced Electro-Optical Systems. Also included is a LN200 IMU, and a compute and interface element. This paper will discuss the architecture and function of the HDS Compute Element (HDSCE).

2. HDS COMPUTE ELEMENT HARDWARE

The HDSCE integrates all the processing, sensing, and control in the HDS system. It takes commands and vehicle state from the host vehicle, interfaces and controls sensors and gimbals, processes data from all these data sources, and provides input to the host vehicle's GN&C system. It consists of three parts. The first part is a Xilinx FPGA-based I/O aggregation and time stamping system. The second part is a Tilera Tile64 based manycore processing system, and an optional third part is a PC-104-based debugging console/logger system for post-flight analysis. An overall block diagram of the system is illustrated in Figure 1. The processor board and the FPGA board are connected by a 10Gb/s XAUI link, and the external devices are connected by either their special I/O or via Ethernet, depending on the device.

978-1-4673-1813-6/13/$31.00 ©2013 IEEE

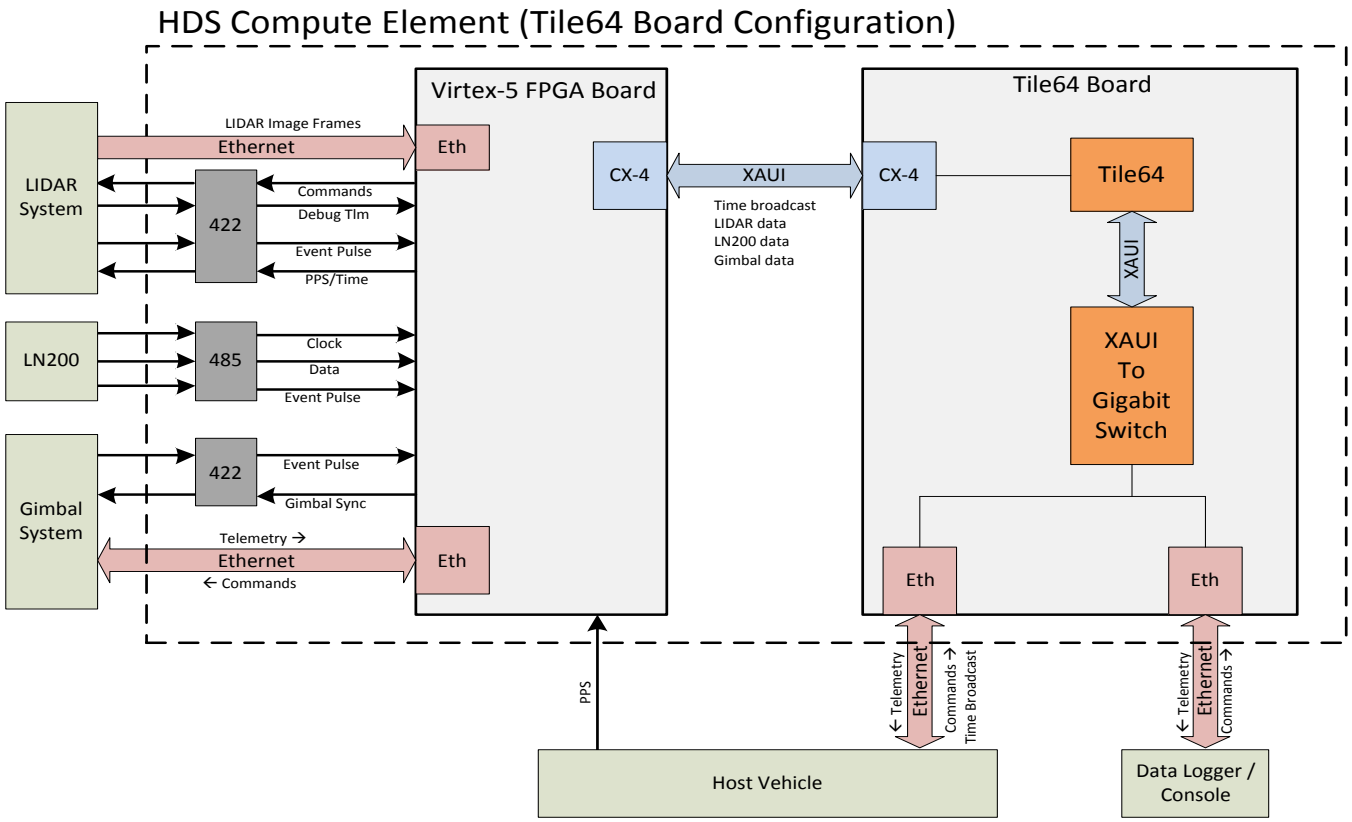


Figure 1 – Overall HDSCE Block Diagram and interface with external devices

The HDSCE box combines an FPGA board for sensor integration and timing, with a manycore computer board for processing. The FPGA does system level timing and data aggregation and acts as a go-between removing the real time requirements from the processor and labeling events with a high-resolution timestamp. The processor manages the behavior of the system, controls the instruments connected to the HDSCE, and services the "heavy lifting" computational requirements for analyzing the potential landing spots. The box itself is a custom enclosure with vibration isolation (Figure 2).

FPGA I/O and Timing board

In order to both reduce I/O requirements on the processor board and to eliminate tight real-time I/O requirements on the Operating System running on the processor board, it was decided that the FPGA shall both aggregate all I/O and annotate all data flowing through the FPGA with a system time associated with a measurement event. To that end, all sensor and gimbal interfaces go through the FPGA, and each device is required to also provide a discrete event signal flag that indicates to the FPGA when the device is taking a measurement.

The I/O is system bi-directional and is managed by the FPGA. Each device has its own dedicated data handler, with the gimbal and LIDAR each getting a dedicated Ethernet Interface, and the IMU getting a custom I/O controller and data pre-processor. All data flowing through the FPGA is tagged, annotated with system time (explained below) and is multiplexed/demultiplexed and shipped to/from the HDSCE processor via a 10 Gb/s XAUI link. The system time is managed by the FPGA and consists of two counters:

1) A whole number of seconds counter, incremented by a 1-Pulse-Per-Second (PPS) signal from the Host vehicle. This is essentially a PPS counter. At boot, this counter starts counting from zero, and waits for time messages from the host vehicle (HV) forwarded from the HDSCE processor. When HDS gets valid time messages from HV, this counter is set to the HV whole number of seconds. Once set, this counter continues to increment with PPS, and error flags are thrown if time messages stop agreeing with the current whole number of seconds value.

2) A subsecond counter counting at 100 MHz, reset by the rising edge of the external PPS. This counter nominally counts from zero to 100 million. This counter is not settable by anything from the outside world other than the PPS signal acting as reset. At the rising edge of the PPS, the current count gets copied to a “denominator” register and the subsecond counter starts counting again at zero. This denominator field is basically a register of the number of counts that the previous PPS interval saw. This allows the HDSCE to compensate for FGPA clock oscillator drift over time and report a correct fractional time.

When a measurement event signal (from the gimbal, IMU, LIDAR, etc.) is received, the current denominator, subsecond, and second registers are copied into an "event time" register associated with that device. When data for that device flows through the FPGA, its data packet gets annotated with the contents of the "event time" register and passed to the HDSCE processor.

The HDSCE processor takes the 3 values and turns it into a fractional time (seconds + subseconds/denominator) and assigns that as "HDS time" to the data packet received. The reader may notice that the subsecond count is for the current second, while the denominator was the count of the previous second, however, we have not observed more than 2 or 3 count differences from one second to the next in the delivered system. A 2-3 part per 100 million error is insignificant, and we are attempting to compensate long term clock oscillator instability, where over several hundred or thousands of seconds, the count may drift by significant amounts.

Tilera Manycore Processor

The HDSCE processor is based on the Tile64 Processor from Tilera Corporation. The Tilera Tile64 runs at 750 MHz and has a parallel architecture with 64 Processing Elements (PEs) arranged in an 8x8 grid. Each PE is a full-featured processor, with a 32-bit ALU and a three-way Very Long Instruction Word (VLIW) architecture allowing up to 3 instructions per cycle. A single PE contains 8KB of Instruction L1 Cache, 8KB of Data L1 Cache, and 16KB of combined L2 cache. The cache coherency mechanisms on the Tile64 also allow for a "Virtual L3" cache, where each PE can access the contents of any other PE's cache. Each PE also contains its own DMA engine and Translation Lookaside Buffer (TLB) allowing memory virtualization and the ability to run a modern O/S on each core. PEs are connected to their 4 nearest neighbors by 5 data channels. There are four external DDR2 memory channels and each processor can access external memory directly without explicit user code to transfer messages. The external memory architecture is monolithic and flat, as a specific memory location has the same physical address on every PE. The processor also includes multiple I/O ports, including two PCIe interfaces, multiple SERDES interfaces, General Purpose I/O, and XAUI. [11]

For this system, an off-the-shelf evaluation board from Tilera was used. The evaluation board was a full-length PCIe form factor board containing a Tile64 processor, 2.5 Gigabytes of DDR2 RAM, a PCIe interface, and all the high-speed interfaces brought out to connectors. The Gigabit Ethernet interface is brought out through a 12 port Gigabit Ethernet switch. The Tilera board contains enough resources to boot and run a Linux based O/S and enough peripherals to communicate with the rest of the components of the HDSCE.

Data Logger/Console system

The final piece of the HDSCE is the optional data logger and console system (DLC). This system is based on a COTS PC/104+ x86 single board computer connected to a solid-state drive. This system is connected to the HDSCE processor via Ethernet, and allows for rich logging of all data generated by the HDSCE for post-flight analysis as well as presenting a telemetry and console screen for real-time observation of the state of the HDS.

Figure 2: HDS-CE box

3. HDS SOFTWARE

The main HDSCE software suite runs on the Tile64 processor board. In general, the software consists of a main HDS Executive, instrument managers, and data and algorithm managers. Externally, the HDS Executive communicates with the Host Vehicle. Internally, it coordinates between instrument manager software modules such as a LIDAR manager, and Gimbal manager, as well as data handling and processing modules such as a data manager, navigation manager, and algorithm manager.

The HDS software has two major responsibilities. The first is to process and manage data from the Host Vehicle, IMU, LIDAR, and Gimbal. These data are processed and stored as needed, and provided to any HDS function that needs that data. The second responsibility is to perform Hazard Detection (HD) and Hazard Relative Navigation (HRN) by computing from that data a list of safe landing sites, as well as navigation inputs to the spacecraft based on tracking surface hazards. The software also contains a logging interface to the DLC.

The software is organized as a combination of task-parallel and data-parallel applications and its responsibilities and modules are spread amongst the available cores on the Tile64. The managers communicate amongst themselves via "messages" delivered via POSIX message queues. There are currently eight "managers," or executables, in the system. They are described as follows:

The **Algorithm Manager** handles CPU-intensive data processing algorithms for Hazard Detection, Hazard-Relative Navigation, and gimbal mosaic planning. Algorithms are being coded to run in parallel on several cores. Although this manager was initially coded as a single-threaded application to ease the development of the overall system behavior, this task is currently being coded as a hybrid single-threaded state machine controlling a separate Tilera parallel application for CPU-bound algorithms.

The **Annotator** interpolates recent gimbal and navigation poses into a single pose relating the sensor and map coordinate systems. Such a combined pose is associated with each range image.

The **Data Manager** underlies the entire Compute Element. Any messages generated by the system are routed via the Data Manager. The data manager distributes all messages according to a pre-compiled subscription list. Since all messages pass through the Data Manager, the Data Manager automatically forwards telemetry from all modules to the DLC if logging is enabled. This manager is a multi-threaded application in order to reduce message routing latency. The link between Data Manager and the DLC is bidirectional, allowing developers to "replay" log files through the HDS software for development and debugging.

The **HDS Executive** (abbreviated 'Exec') is the overall manager of the Compute Element. It accepts high-level commands from the Host Vehicle via UDP/IP Ethernet Messages and returns data such as a list of safe landing sites' coordinates, and tracking observations useful to the Host's navigation function. It also accepts commands from the console. Commands from whatever source are handled by issuing sub-commands to other managers, in some cases directly and in others from a state machine.

The **Gimbal Manager** controls the gimbal device by sending low-level commands, and by reading and analyzing status messages from the device. There are three pointing commands: (1) point the gimbal/sensor boresight relative to the vehicle body; (2) track a point on the ground; (3) execute a mosaic where the boresight is swept in a back-and-forth pattern covering a rectangular area on the ground. The Gimbal Manager requires knowledge of the current vehicle state (position, attitude) from the Nav Manager so it can compute pointing angles. The Gimbal Manager routinely provides a coordinate-transformation 'pose' to the Data Manager that describes the orientation of the LIDAR boresight relative to the HDS IMU.

The **LIDAR Manager** commands the LIDAR to power up, stand by, or collect data. It reads status messages from the LIDAR device. When running, The LIDAR Manager assembles individual LIDAR data packets into complete range images.

The **Nav Manager** propagates an initial navigation state using the HDS IMU. The navigation state is bootstrapped to the Host Vehicle's navigation state at the Host Vehicle's rate of 50 Hz. During mosaicking, the bootstrapping is suspended via a command by Exec, during which the navigation state is propagated using only the HDS IMU data. The navigation state is sent to the Gimbal Manager and Annotator (for annotating LIDAR images), Mosaic planner, and to Exec (to monitor range to target). The Nav Manager sends out the navigation state at the HDS IMU rate of 400 Hz.

The **XAUI Manager** manages the interface between the Tile64 and FPGA. It is a multi-threaded application and uses the low-level Tilera API to ensure the lowest latency of message delivery to and from the Data Manager as possible. All data to and from external HDS devices flows through the XAUI manager.

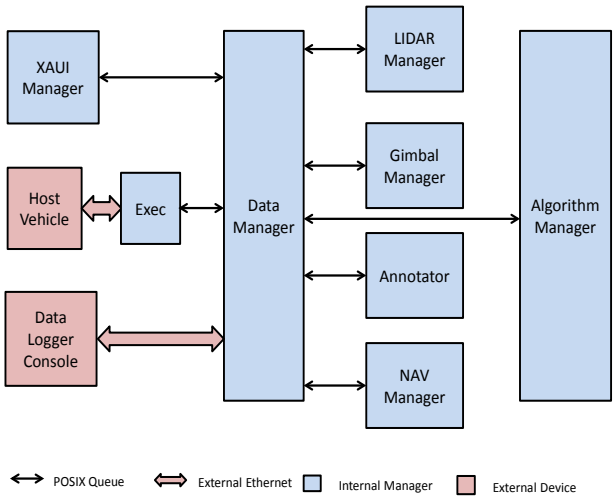


Figure 3: Software Block Diagram

4. MORPHEUS INTEGRATION

In order to test the performance of the HDS hardware and software, a flight with a realistic planetary landing trajectory is needed in order to present a high-fidelity scenario to evaluate against. The Morpheus Lander built by NASA's Johnson Space Center (JSC) was chosen as a testbed platform capable of flying a typical lunar landing profile. The Morpheus lander is a Liquid Oxygen/Methane-fueled vehicle capable of vertical landings and takeoffs [12].

The ALHAT program worked with the Project Morpheus team to integrate the HDS as an instrument on the lander. The entire ALHAT HDS system was mounted on the vehicle and basic interface checks were performed in the

laboratory. Static gimbal pointing knowledge and performance error was measured to be well within 0.15 degrees over intermediate and long distances. An illustration of the HDS mounted on Morpheus can be seen in Figure 4.

Figure 4: The ALHAT Hazard Detection System was physically integrated onto the Morpheus vehicle for static pointing tests and dynamic tether tests at JSC in May/June 2012.

A series of short tethered flights was performed during late spring and early summer of 2012 to evaluate dynamic performance of the Morpheus/ALHAT system and to ensure that the Morpheus/HDS system operated properly. Four test flights were conducted with high levels of success. A sample video illustrating the ALHAT HDS operating can be found in the Morpheus Tether Test 17 video [13], courtesy of the Morpheus Project.

Operation was as expected, however dynamic pointing performance did not meet the desired error budget. The issue has been identified and is being corrected at this writing.

At the end of the campaign, ALHAT and Morpheus were deintegrated for further development work on both platforms. As of this writing, integration of additional HDS capabilities into the combined ALHAT/Morpheus system is currently under development. Future flights will include observation and processing of a pre-constructed hazard field built at the Shuttle Landing Facility at the NASA Kennedy Space Center to evaluate HDS algorithm performance for selecting safe sites, and providing navigation input to the host vehicle.

5. SUMMARY

The combination of a manycore processor with an FPGA allowed great flexibility and ease of development for the HDSCE. This architecture combines the strengths of two architectures: the high-performance timing, I/O, and interface ability and processing of an FPGA, with the high-performance computing, flexibility, and programmability of a general-purpose manycore processor. This combination of an FPGA with a manycore processor, with both components being concurrently used for processing, has yet to be done for space applications.

This architecture is also useful for embedded robotic applications such as rovers. The FPGA/manycore combination allows the end user to place tasks on either the FPGA or the manycore processor, based on the strengths and weaknesses of each component.

The flexibility of the manycore processor running Linux allowed the development of a hybrid task and data parallel system. The large number of cores allowed each “manager” to be able to run on its own core without needed to compete for processor time.

The addition of the FPGA allowed not only I/O flexibility, but its high resolution timer removed many real-time requirements from the processor, allowing the system to know with very high fidelity when events occurred in the real world.

This integrated manycore/FPGA system is demonstrating to be a good fit for the HDSCE and ALHAT. We are looking for opportunities to integrate our current COTS solution into a custom single board computer/system in order to save weight, power, and integration effort.

Future, more focused papers will be presented discussing individual software component operation, performance, strengths, and weaknesses of the algorithms, computation, and their fit with the architecture described in this paper.

ACKNOWLEDGEMENTS

The work described in this publication was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract from the National Aeronautics and Space Administration under the Autonomous Landing and Hazard Avoidance Technologies program. Government sponsorship acknowledged.

REFERENCES

[1] Epp, C. and Smith, T., "Autonomous Precision Landing and Hazard Detection and Avoidance Technology (ALHAT)," Proc. IEEE Aerospace Conf., Big Sky, MT, March 2007.

[2] Johnson, A. and T. Ivanov, "Analysis and Testing of a LIDAR-Based Approach to Terrain Relative Navigation for Precise Lunar Landing," Proc. AIAA Guidance Navigation and Control Conference (AIAA-GNC 2011), August 2011.

[3] Keim, J., S. Mobasser, E. Bailey, A. Johnson, and G. Khanoyen, "Test Implementation to Evaluate Technologies for Safe Lunar Landing," Proc. IEEE Aerospace Conference (AEROCONF 2011), March 2011.

[4] Cheng, Y., D. Clouse, A. Johnson, W. Owen, and A. Vaughan, "Evaluation and Improvement of Passive Optical Terrain Relative Navigation Algorithms for Pin-Point Landing," Proc. AAS Space Flight Mechanics Meeting (AAS-SFM 2011), January 2011.

[5] Huertas, A., A. Johnson, R. Werner, and R. Maddock, "Performance Evaluation of Hazard Detection and Avoidance Algorithms for Safe Lunar Landing," Proc. IEEE Aerospace Conference (AEROCONF 2010), March 2010.

[6] Johnson, A., J. Keim, and T. Ivanov, "Analysis of Flash Lidar Data Field Test Data for Safe Lunar Landing," Proc. IEEE Aerospace Conference (AEROCONF 2010), March 2010.

[7] Keim, J., S. Mobasser, D. Kuang, Y. Cheng, T. Ivanov, A. Johnson, H. Goldberg, G. Khanoyan, and D. Natzic, "Field test implementation to evaluate a flash LIDAR as a primary sensor for safe lunar landing," Proc. IEEE Aerospace Conference (AEROCONF 2010), March 2010.

[8] Villalpando, C., A. Johnson, J. Oberlin, and S. Goldberg, "Investigation of the Tilera Processor for Real-time Hazard Detection and Avoidance on the Altair Lunar Lander," Proc. IEEE Aerospace Conference (AEROCONF 2010), March 2010.

[9] Johnson, A. and J. Montgomery, "An Overview of Terrain Relative Navigation for Precise Lunar Landing," Proc. IEEE Aerospace Conference (Aerospace08), Big Sky, MT, March 2008.

[10] Johnson, A.E., A. Huertas, R.A. Werner, and J.F. Montgomery, "Analysis of On-Board Hazard Detection and Avoidance for Safe Lunar Landing," Aerospace Conference, 2008 IEEE, pp.1-9, 1-8 March 2008.

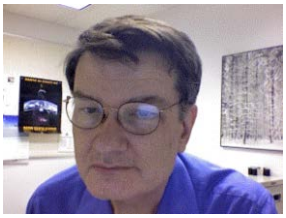
[11] Wentzlaff, D.; Griffin, P.; Hoffmann, H.; Liewei Bao; Edwards, B.; Ramey, C.; Mattina, M.; Chyi-Chang Miao; Brown, J.F.; Agarwal, A.;. "On-Chip Interconnection Architecture of the Tile Processor", IEEE Micro, September-October 2007, pp. 15-31.

[12]Olansen, J., Munday, S., Mitchell, J., Baine, M., (May 23-25, 2012). "Morpheus: Advancing Technologies for Human Exploration". Global Exploration Conference. GLEX-2012.05.2.4x12761.

[13] Morpheus Project, "ALHAT Hazard Detection System - Tether Test 17", July 19, 2012. YouTube, http://www.youtube.com/watch?v=MV7T1riox_g

BIOGRAPHY

Carlos Y. Villalpando *is a Senior Member of Technical staff in the Advanced Computer Systems and Technologies group at the Jet Propulsion Laboratory. He is currently a digital designer for advanced computing techniques for machine vision applications in FPGAs as well as system designer and programmer for machine vision tasks on multicore processors. He earned his Bachelor of Science degree in Electrical Engineering, Computer Block at the University of Texas at Austin in 1996 and a Master of Science in Electrical Engineering-VLSI at the University of Southern California in 2003. He has been a member of the JPL community continuously since 1993 and has worked primarily on Technology development tasks.*



Dr. Robert Werner *obtained a Bachelor of Arts in Mathematics (1972) from the University of Iowa and a Master of Science in Engineering (1990) and Doctor of Philosophy (1996) from the University of Texas at Austin, both in aerospace engineering under a NASA Graduate Student Research Fellowship. Before attending graduate school, he worked at the University of Iowa's academic computer center (1970-1988) where he was ultimately titled Research and Development Project Leader. After graduate school he joined NASA's Jet Propulsion Laboratory (1996-2012) where he has worked on autonomous navigation flight software ("AutoNav") for the Deep Space 1 and Deep Impact missions. Currently he is enhancing AutoNav, and working on the Autonomous Landing and Hazard Avoidance (ALHAT) project. His interests included computer graphics, operating systems, and microcomputers. Asteroid 3891 Werner is named in his honor. Dr. Werner recently retired from JPL in the Fall of 2012.*

John M. Carson III *received a Ph.D. in Mechanical Engineering (2008) from the California Institute of Technology, preceded by a B.S. in Aerospace Engineering (1997) and M.S. in Engineering Mechanics (1999) from the University of Texas at Austin. From 1999-2000 he performed flight and structural dynamics testing and analysis on the Joint Strike Fighter prototype at the Lockheed Martin Skunk Works. From 2001-present he has worked in guidance, navigation, and control design, analysis, and hardware development at the NASA Jet Propulsion Laboratory for several missions, including Cassini/Huygens and Mars Exploration Rover, and in technology development programs for small-body and planetary landing systems. From 2011-2012 he has been the JPL systems engineer and project manager for the Autonomous Landing and Hazard Avoidance Technology (ALHAT) project.*

Garen Khanoyan *is a senior member of the Advanced Computer Systems & Technologies group at JPL. He received his Bachelor of Science degree in Electrical Engineering (1999) followed by a Master of Science in Computer Science (2004) from the University of Southern California. In the recent years, he has been involved with development and field testing activities for the ALHAT, MSL, and LDSD projects at JPL. On the ALHAT project he developed the FPGA in the HDSCE and designed the DLC software.*

Nikolas Trawny *received a DEA in Automatic Control Systems (2003) from Supaero/ENSAE, Toulouse, France, a Diploma in Aerospace Engineering (2004) from the University of Stuttgart, Germany, an M.Sc. in Computer Science (2008) from the University of Minnesota, followed by a Ph.D. in Computer Science (2010) from the University of Minnesota. Since 2010, he has been working at the NASA Jet Propulsion Laboratory on projects such as MAVEN orbit determination and technology development programs for autonomous precision-navigation algorithms for small-body, lunar, and planetary landing applications. His interests lie in vision-aided inertial navigation and sensor fusion.*

Ryan A. Stern *is a member of the Advanced Computer Systems & Technologies group at JPL. He received his B.A. in Physics (1999) from Pomona College and his M.S. in Electrical Engineering (2003) from the University of California, Los Angeles. He has worked on a wide variety of technology tasks and flight projects since joining the NASA Jet Propulsion Laboratory in 1996. He was involved in avionics testing for the Mars Reconnaissance Orbiter, and he was one of the designers of the motor control assembly for the Mars Science Laboratory. He is currently working on an advanced, cold-capable motor controller for future Mars missions and supporting the FPGA and software development for ALHAT.*