Enabling Continuous Planetary Rover Navigation through FPGA Stereo and Visual Odometry

Thomas M. Howard, Arin Morfopoulos, Jack Morrison, Yoshiaki Kuwata, Carlos Villalpando, Larry Matthies, and Michael McHenry
Mobility and Robotics Systems (347)
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109
thomas.m.howard@jpl.nasa.gov

***Abstract*—Safe navigation under resource constraints is a key concern for autonomous planetary rovers operating on extraterrestrial bodies. Computational power in such applications is typically constrained by the radiation hardness and energy consumption requirements. For example, even though the microprocessors used for the Mars Science Laboratory (MSL) mission rover are an order of magnitude more powerful than those used for the rovers on the Mars Exploration Rovers (MER) mission, the computational power is still significantly less than that of contemporary desktop microprocessors. It is therefore important to move safely and efficiently through the environment while consuming a minimum amount of computational resources, energy and time.**

Perception, pose estimation, and motion planning are generally three of the most computationally expensive processes in modern autonomy navigation architectures. An example of this is on the MER where each rover must stop, acquire and process imagery to evaluate its surroundings, estimate the relative change in pose, and generate the next mobility system maneuver [1]. This paper describes improvements in the energy efficiency and speed of planetary rover autonomous traverse accomplished by converting processes typically performed by the CPU onto a Field Programmable Gate Arrays (FPGA) co-processor. Perception algorithms in general are well suited to FPGA implementations because much of processing is naturally parallelizable. In this paper we present novel implementations of stereo and visual odometry algorithms on a FPGA. The FPGA stereo implementation is an extension of [2] that uses "random in linear out" rectification and a higher-performance interface between the rectification, filter, and disparity stages of the stereo pipeline. The improved visual odometry component utilizes a FPGA implementation of a Harris feature detector and sum of absolute differences (SAD) operator. The FPGA implementation of the stereo and visual odometry functionality have demonstrated a performance improvement of approximately three orders of magnitude compared to the MER-class avionics. These more efficient perception and pose estimation modules have been merged with motion planning techniques that allow for continuous steering and driving to navigate cluttered obstacle fields without stopping to perceive. The resulting faster visual odometry rates also allow for wheel slip to be detected earlier and more reliably. Predictions of resulting improvements in planetary rover energy efficiency and average traverse speeds are reported. In addition, field results are presented that compare the performance of autonomous navigation on the Athena planetary rover prototype using continuous steering or driving and continuous steering and driving with GESTALT traversability analysis using the FPGA perception and pose estimation improvements.

TABLE OF CONTENTS

978-1-4577-0557-1/12/$26.00 ©2012 IEEE.
[1] IEEEAC Paper #1790, Version 1, Updated 01/4/2012.

1. INTRODUCTION

The computational power of space-qualified microprocessors has long lagged behind their commercial counterparts. This is due to the power constraints on the spacecraft itself a well as other factors unique to operating in the extreme environments of space such as extreme temperatures, vacuum, shock/vibration, and radiation. This results is a limited selection of microprocessors for spacecraft operations beyond low-Earth orbit. For example Spirit and Opportunity, the planetary rovers of the Mars Exploration Rovers (MER) mission, are powered by a 20 MHz RAD6000 computer providing approximately 22 MIPS. Curiosity, the planetary rover of the Mars Science Laboratory (MSL) mission, will utilize a newer RAD750 capable of operating at 200 MHz and 400 MIPS. This level of performance was exceeded over two decades ago in the consumer desktop market and has since been exceeded by more than two orders of magnitude (albeit with far greater power consumption).

Counter to these computational resource limits is the desire to operate with a high degree of autonomy that is essential for planetary rovers operating at such extreme distances. Communication with Spirit and Opportunity is limited to only several windows each day and exhibit communications latency that measures in the minutes. The Grid-based Estimation of Surface Traversability Applied to Local Terrain (GESTALT) algorithm is the method used by the MER rovers to autonomously navigate Martian terrain. When visual odometry is used in conjunction with GESTALT, each motion is limited to either a 75 centimeter straight line or arc or a turn-in-place of no more than 18 degrees. On the RAD6000 visual odometry processing can take two to three minutes [2] and significantly slow down autonomous traverse. The added computational cost of autonomous traverse and pose estimation slows the maximum average speed of MER from 124 meters per hour for a non-autonomous drive to just 36 meters per hour using wheel odometry. In environments where visual odometry is necessary, the maximum average speed of navigation (without AutoNav) is 10 meters per hour [3]. Use of both AutoNav and visual odometry reduces the maximum average speed of autonomous traverse so significantly that it has rarely been used in practice.

Field Programmable Gate Arrays (FPGA) provide the possi-

bility of improved performance and power efficiency for processes that can be divided and computed in parallel. Recent products such as Xilinx's Radiation-Hard-By-Design V5QV offer much more substantial resources than other currently available products. FPGA implementations of autonomous navigation processes can reduce or eliminate the performance penalty incurred by using radiation-hardened processors. The potential benefits for autonomous planetary exploration are significant as rover power efficiency, average traverse rates, and mission safety would all improve. Perception algorithms in particular are well suited to FPGA implementation because many of the processes are inherently parallelizable. This paper describes an FPGA implementation of stereo and visual odometry algorithm components mixed with modifications of contemporary planning algorithms to enable continuous autonomous navigation with always-on slip detection.

Related Work

Related work for this paper spans two fields: perception and autonomous navigation algorithms. The JPL stereo algorithm [4] estimates the range to all pixels in a set of images by determining a correlation between regions in each image. [5] adapted portions of this algorithm on an FPGA to achieve the performance necessary for high-speed field robot navigation.

Mobile robot navigation is a well studied research problem with applications in planetary exploration, agriculture, mining, transportation, and defense. Most modern mobile robot navigation algorithms are hierarchical in nature, where a high-fidelity local planner that generates safe trajectories is paired with a coarse approximation of vehicle motion at the global scale. Common types of local motion planners include potential field, sampling, and graph search techniques. Potential field methods follow gradients of potentials that map to hazards and goals in the environment [6], [7]. Sampling techniques include input space sampling [8] and state space sampling [9] methods that generate a set of possible motions using generalized predictive motion models. Graph search can be based on preconstructed edge sets (ego-graphs) [10], randomized sampling [11], or regular sampling in the state space [12]. This paper describes the modification of the GESTALT navigation algorithm for continuous steering or driving (CSOD) and a sampling-based approach for continuous steering and driving (CSAD) with faster perception and pose estimation implementations for continuous planetary rover navigation.

This work is distinct from other approaches because it addresses the problem of efficient planetary rover navigation with highly constrained computational resources. Specifically it integrates FPGA perception and pose estimation algorithm implementations with two different types of navigation algorithms to improve the efficiency of planetary rover traverse.

2. TECHNICAL APPROACH

This paper's approach to continuous planetary rover navigation using FPGA implementations of perception algorithms spans hardware design, software implementations, and algorithmic modifications. This section describes the architecture for a FPGA co-processor, the FPGA design and implementation of perception algorithms, and the modification of autonomous navigation algorithm to enable continuous navigation.

FPGA Architecture

There are many different approaches for integrating an FPGA with a CPU in an embedded system. Three examples of these include a single board computer with a PCI interface to an FPGA co-processor, a single board computer with an on-board FPGA that can communicate directory with the CPU, and a separate FPGA board that communicates with the CPU over SpaceWire. The first architecture was previously used on the Unmanned Ground Combat Vehicle PerceptOR Integrated (UPI) program using an Alpha-Data ADM-XRC4 board that hosted a Virtex4LX160 FPGA and transmitted image data over the PCI bus. The second architecture is more compact and has the potential for higher performance because the FPGA would not exhibit the latency of communicating over the PCI bus. The last architecture favors modularity and generality over performance as only the SpaceWire interface is shared between the FPGA and the CPU.

FPGA Implementation

The first step in designing an FPGA implementation of machine vision algorithms is to understand the autonomous navigation pipeline, which consists of: image acquisition, stereo, visual odometry, traversability analysis, and path planning and execution. Image acquisition involves capturing, downsampling, transmission, and storages of images acquired from rover-mounted cameras. Stereo decomposes into image rectification, filtering and disparity processing. Visual odometry involves identification and tracking of features between images to estimate pose changes. Traversability analysis transforms stereo data into a map used by a path planner that approximates the relative safety of traversing a particular region. Path planning and execution uses the estimated pose and traversability analysis to determine and command the next trajectory towards the overall goal region. This paper describes the acceleration of stereo and visual odometry with the use of an FPGA as a co-processor.

There are two basic FPGA resources that dominate our design choices: slices and blocks. Slices are the computational fabric of the FPGA and Block RAM (BRAM) store intermediate results. While resource consumption of the implementation is not a function of the image height, image width does influence the required number of Slices and BRAM per module.

FPGA Stereo—In [5] the computationally expensive portions of stereo were implemented in the FPGA: the rectification, filtering, and disparity calculations. The input to stereo is a pair of unrectified images, and a pair of rectification tables. The output from stereo is a pair of rectified images, a pair of filtered images, and a single disparity map (which contains range data for each pixel). Any or all of the output data products can be saved.

This implementation differs from the approach in [5] in two ways. The previous FPGA rectification implementation read the input image linearly and used the rectification table to determine where the output pixel is written. This implementation instead starts at the top left of the output image and uses the rectification table to determine which four input pixels should be used to determine that output pixel value. This process is continued linearly throughout the entire output image. This allows the output of rectification to feed directly into the filter module, and has higher performance. It is also the way that heritage flight software performs, and thus uses the same rectification table that flight software uses.

The second substantial difference is the interface between the

rectification, filter, and disparity stages of the stereo pipeline. Each module has been wrapped in a common interface, so that each module can communicate with each other or directly to memory. The data path between rectification, filter and disparity is configurable at run-time, so that intermediate results can be saved for debugging or omitted for improved performance.

FPGA Visual Odometry— Pose estimation through visual odometry is generally a four-step process. First, features are detected in each frame using a feature detection algorithm such as corner detection. Second, featured are matched using a sum of absolute differences (SAD) over local regions between subsequent image. Next, the largest set of inliers (self-consistent matches) is determined by RANSAC or other filtering techniques. Lastly, the relative motion between frames is estimated that minimizes the image space re-projection error for features in the inlier set. For our FPGA implementation of this algorithm, the first two steps are done by the FPGA and the CPU performs the last two steps.

In order to evenly distribute features across the image, the corner detector is run on separate local regions of the image. For each region the weights of the feature detector are iteratively modified to obtain a minimum/maximum number of features in each local region. Increasing the number of features found in each image increases the quality of the pose estimate at the cost of increased computational time.

The visual odometry SAD operator takes in a *set* and a *reference* image, with a feature coordinate list for set of N features, and a feature coordinate list for reference of M features. SAD then compares each feature from N with each feature from M. The performance bottleneck is in the memory access for the 15 pixel by 15 pixel local window around each feature in N and M. We use BRAM as a cache to load up to 512 feature sub-windows from N and compare each against one feature sub-window at a time from M. If more than 512 features exist in N, the process is repeated with the remaining features. The size of the feature cache (512) is configurable at build time, and is a trade between BRAM resources and speed.

Autonomous Navigation

The FPGA stereo and visual odometry improvements are generally beneficial to any type of autonomous navigation algorithm. This section describes how the GESTALT is modified for continuous steering or driving (CSOD) and the traversability analysis is used for continuous steering and driving (CSAD) using the higher-rate perception and pose estimation data.

Continuous Steering or Driving Navigation—The GESTALT system used by the Mars Exploration Rovers [4] includes the stereo image capture, stereo processing, visual odometry, terrain evaluation, and the path evaluation and selection. We have decoupled these processing into distinct software modules by replacing the stereo image capture, stereo processing, and the visual odometry of GESTALT with the fast FPGA implementation that was described above. The modified GESTALT algorithm takes as inputs the three-dimensional point cloud in the body-fixed frame, which is obtained from the stereo module, and the vehicle pose, which is obtained from the visual odometry module.

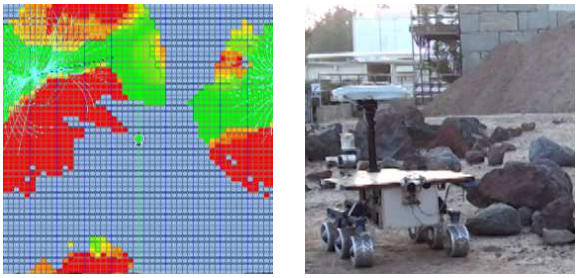
The terrain evaluation part of GESTALT works as follows. It first projects the three-dimensional point cloud to the local navigation frame, which is attached to the ground. GESTALT discretizes the terrain into square grid, and for each grid cell it computes the statistics (mean, variance, etc.) of the projected three-dimensional points. The statistics are then converted to the local slope and roughness, and finally a *goodness* value of each cell. The advantage of maintaining the statistics is that it also provides the information on how certain we are about the goodness of the terrain, which can be used to evaluate the risk of a traversal. The output of the terrain evaluation is the goodness map and the certainty map, to be used by path evaluation and selection.

The path evaluation part of GESTALT first generates a set of candidate motions, consisting of arcs of various curvatures or turn-in-place of different angles followed by a straight drive. It then evaluates each motion against the goodness map and the certainty map. By weighting the goodness, risk, and closeness to the final destination, GESTALT selects the best path to follow, and sends a command to the low-level vehicle controller.

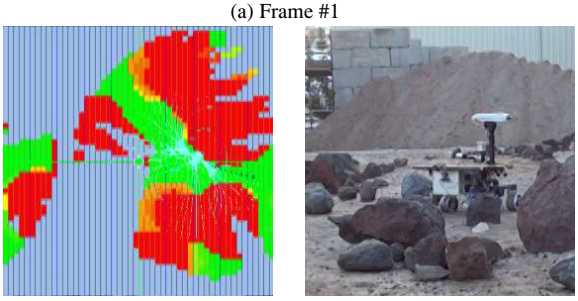
GESTALT executes a path in a receding horizon fashion and does not execute the entire path that was selected. To achieve continuous steering or driving, the behavior of the algorithm is modified to begin replanning with updated pose and range information once a fraction of the executed motion is complete. Once a new plan has been generated, the previous motion is interrupted and the new motion is executed. This process repeats until the goal has been reached. Figure 1 shows snapshots of a traversal of our research rover using FPGA stereo, FPGA VO, and the GESTALT. Note that GESTALT uses a fixed-size memory footprint and maintains the map only around the vehicle. The two-dimensional maps on the left are wrapped at the edges of the image.

Continuous Steering and Driving Navigation—One limitation of the current GESTALT implementation for planetary rovers that can continuously operate steering and drive motors is that the input space sampling based local motion planning search space is limited to sequences of steer-in-place, turn-in-place and constant curvature arc primitives. This is a significant limitation for planetary rovers whose electromechanical systems allow for simultaneous control of both steering and drive motors because more expressive and energy efficient motions can be planned and executed. Even with the non-FPGA implementations of stereo and visual odometry the efficiency of the stop, sense, and act behavior sequence could be made more efficient using continuous steering and driving techniques. The opportunity presented by exploiting the aforementioned perception and pose estimation performance improvements is however particular well suited for experimentation with continuous steering and driving techniques.

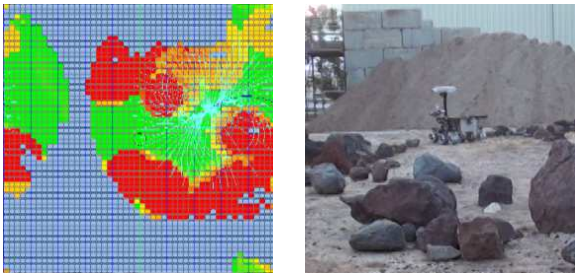
Since the original GESTALT navigation divides naturally into traversability analysis and path planning components, the same traversability analysis is used to provide fair comparisons of the efficiency of each approach. This navigation algorithm can be adapted for continuous steering and driving by replacing the path planner that generates sequence of turn-in-place and arc maneuvers with one that attempts to maintain the continuity of commanded curvature. A first-order implementation of this would replace the arc motions with Clothoids. Clothoid trajectories are motions whose curvature varies linearly with distance or time. Curvature continuity can easily be achieved by simply enforcing a constraint that the current curvature of the vehicle trajectory (at the point where the command is executed) matches the beginning of the planned motion. An example of this approach illustrated in Figure 2 where Clothoid trajectories are planned using



(a) Frame #1

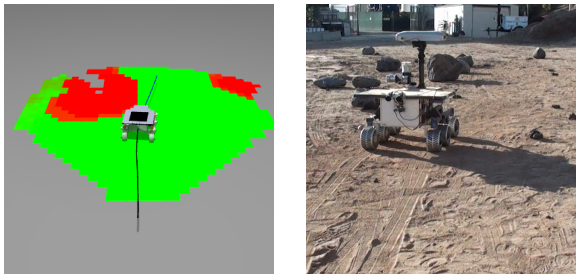


(b) Frame #2

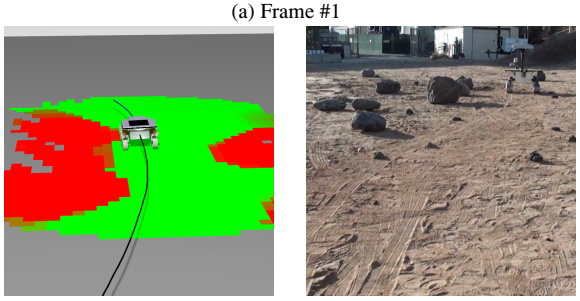


(c) Frame #3

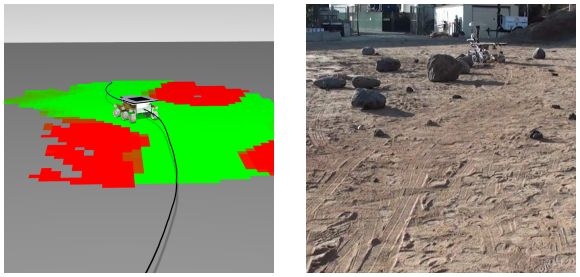
Figure 1: Traversability analysis, search spaces, and navigation of a planetary rover testbed with the GESTALT navigation algorithm. The images on the left of each frame show the goodness maps generated by the GESTALT traversability analysis tuned for this prototype planetary rover. Safe regions are indicated by the color green, dangerous regions are illustrated with the color yellow through red, and unknown regions are show in gray. The search space composed of steer-in-place, turn-in-place, and action primitives is overlaid in white. Images corresponding to the planned paths are shown on the right of each frame.



(a) Frame #1



(b) Frame #2



(c) Frame #3

Figure 2: Traversability analysis and planned paths of a planetary rover testbed with an input space sampling-based navigation algorithm. The images on the left of each frame illustrate the goodness maps generated by the GESTALT traversability analysis tuned for this prototype planetary rover. Similarly to Figure 1, safe regions are indicated by the color green, dangerous regions are illustrated with the color yellow through red, and unknown regions are show in gray. The next action selected from the clothoid-based search space is shown in blue. Images corresponding to the planned paths are shown on the right of each frame.

this receding horizon approach in an obstacle field. Updated maneuvers are generated and executed continuously in a manner until the rover reaches the goal similar to the continuous steering or driving modification of GESTALT described previously.

3. EXPERIMENTS

This section describes unit and system level experiments designed to measure the performance advantages of FPGA implementation of perception algorithms. Section 3 details the speed advantage of the FPGA versus non-FPGA stereo and visual odometry calculations. Section 3 exhibits the differences in autonomous planetary rover navigation performance with and without FPGA co-processing.

FPGA Implementation

In order to test our variations of the stereo and visual odometry algorithms were implemented on a Virtex 5 FPGA and compared the results to a similar non-FPGA implementation. We used the first architecture described in Section 2 where the CPU talks to the FPGA over the PCI bus. The implementation uses approximately 27% of the BRAM (136 of 576) and 73% of the slices (29,963 of 40,960) of the Virtex 5 FPGA. The breakdown of BRAM and slices for various components for an image width of 512 pixels is illustrated in Table 1.

The number slices required to compute the stereo disparity on the FPGA is linearly proportional to the width of the acquired image. The amount of BRAM for the rectification, filter, and Harris feature detector modules is similarly linearly proportional to the image width. The amount of BRAM for the disparity module is however proportional to the square of the image width. An illustration of the FPGA implementation is shown in Figure 3.

Algorithm	BRAM	Slices
Stereo Rectification	2	1,186
Stereo Filter	32	3,556
Stereo Disparity	36	13,275
VO Harris Feature Detector	2	10,567
VO SAD Operator	64	1,379

Table 1: Aspects of the Virtex 5 FPGA stereo and visual odometry implementation. Implementation of stereo and visual odometry algorithms on the Virtex5 FPGA take approximately 27% and 73% of the available BRAM and Slices respectively.

FPGA Stereo Implementation—The stereo performance of the FPGA implementation is evaluated by comparing the runtime performance to a CPU-only implementation on a RAD6000 for a variety of image widths. The stereo performance comparison is listed below in Table 2.

	Image Width (pixels)		
Implementation	**256**	**512**	**1024**
20 MHz RAD6000	∼24-30	N/A	N/A
Virtex 5 FPGA	0.005	0.019	0.082

Table 2: Stereo runtime (in seconds) on a 20 MHz RAD6000 and a Virtex 5 FPGA with varying image resolution. The FPGA implementation is faster at all three resolutions than the software implementation on the 20 MHz RAD6000 as demonstrated on MER.

On MER it takes 24 to 30 seconds to generate a disparity map from a pair of 256 pixel wide images on the 20 MHz RAD6000 Computer [13]. An approximation of the speed of stereo vision processing for MSL can be made by multiplying this number by the ratio of MIPS for the MER and MSL processors (22/400) which results in 1.32 to 1.65 seconds for a single stereo disparity map. The FPGA implementation performance data in Table 2 represent an improvement of several orders of magnitude over the current implementations of stereo vision on the CPU.

FPGA Visual Odometry Implementation—To evaluate the performance of the FPGA implementation of visual odometry we ran visual odometry over a set of one-hundred image pairs from MER data. For this implementation each 512 pixel by 384 pixel image is split into 16 separate regions and we require that 10 to 100 features are found. The FPGA implementation of the SAD operator is capable of comparing 400 features per image in 6.8 milliseconds.

	Image Width (pixels)	
Implementation	**256**	**512**
20 MHz RAD6000	∼160	N/A
Virtex 5 FPGA	N/A	0.016+CPU

Table 3: Visual odometry runtime (in seconds) on a 20 MHz RAD6000 and a Virtex 5 FPGA with varying image resolution.

On MER it takes approximately 160 seconds to estimate the relative pose change on the 20 MHz RAD6000 Computer [13], [14] using the visual odometry method described in [15]. An approximation of the speed of visual odometry for MSL can also be made by multiplying this number by the ratio of MIPS for the MER and MSL processors (22/400) which results in 8.8 seconds for a single visual odometry step. Directly comparing the performance numbers described in Table 3 is not entirely fair as the CPU used for feature tracking is far faster than the one used on-board Spirit or Opportunity. The first two steps of visual odometry that are computed by the FPGA (feature detection and feature matching) are known to take approximately 0.016 seconds for a pair of 512 pixel wide images. This implementation currently is capable of estimating pose at approximately 10 Hz on a Pentium-class CPU.

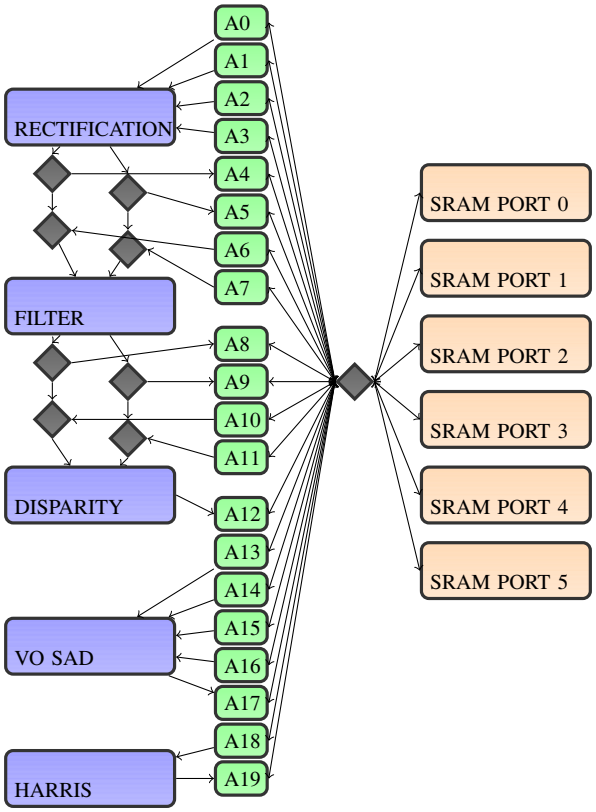


Figure 3: The architecture of the FPGA stereo and visual odometry implementation with the reusable and reconfigurable interfaces. Data paths for input and output of the rectification, filtering, and disparity stages of stereo vision are multiplexed so that the same implementation can be easily debugged and tested. In this figure orange boxes represent SRAM banks, gray diamonds illustrate multiplexers, blue boxes show aspects of the algorithm implementation on the FPGA, and green boxes represent reconfigurable inputs and outputs.

Autonomous Planetary Rover Navigation

To test the applicability of the FPGA accelerated stereo and visual odometry algorithms we have integrated these adaptations into our autonomous navigation architecture for prototype planetary rovers. Herein we describe the hardware and software of our evaluation platform and discuss the performance of the various algorithms using the FPGA stereo and visual odometry improvements.

Evaluation Platform—The experimental platform for this work was the Athena prototype planetary rover (Figure 4). The perception system for these planetary rover navigation

experiments consists of a pair of stereo cameras operating at 512 pixels by 384 pixels. Athena operates with a PC104+ Stack computer using a Pentium M 1.6 GHz processor for the mobility system and a co-processor with a Virtex 4 FPGA development board. The Stack and co-processor computers use VxWorks and Linux for the respective operating systems.

Figure 4: Athena, a six-wheeled prototype planetary rover with a rocker-bogie suspension. Athena was modified for this work to include a Virtex 4 FPGA development board for stereo and visual odometry processing.

The system software is based on [16] where each module is implemented as a hierarchical state machine that handles the interprocess communication. The navigation portion of the system architecture is shown in Figure 5.

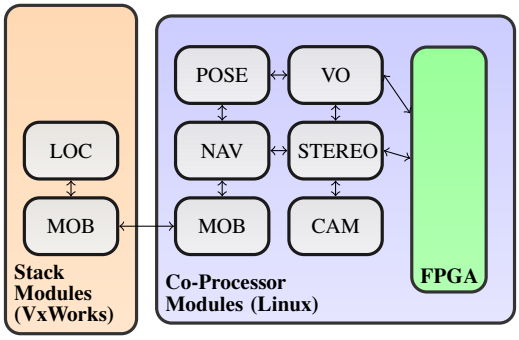


Figure 5: An overview of the abstraction between stack and co-processor navigation modules. The co-processor navigation modules contain the camera interface (CAM), stereo processing (STEREO), state estimation (VO/POSE), and trajectory planning (NAV/MOB). These modules exist on the co-processor in order to interface with the FPGA. The stack modules interface to the mobility system, handling and executing motions (MOB/LOC).

The separation of intelligence (on the co-processor) and actuation (on the stack) enables software portability. This system has been applied to two different mobile robots by only modifying the implementation of the locomotor module and by loading robot-specific navigation and camera model parameters. The co-processor board also has an interface to the FPGA for VO and STEREO modules.

Autonomous Traverse Experiments—In order to evaluate the average traverse speed achievable with the FPGA stereo and visual odometry improvements the environment depicted in Figure 4 was navigated using the GESTALT-based continuous steering or driving (CSOD) and input space sampling based continuous steering and driving (CSAD) techniques. For each experiment the rover was commanded to navigate to a point fifteen meters along the forward relative to the initial orientation of the rover. The maximum speed that could be issued by each algorithm was 7 cm/sec and is reduced only for high-curvature maneuvers. The continuous steering and driving navigator utilized a search space composed of 225 unique 3 meter long Clothoid curvature splines with knotpoints sampled between ± 1 rad/m. Figure 6 shows the position of each rover recorded by FPGA-based visual odometry during the experiment at approximately 5 Hz. Both algorithms navigated to the right of most of the rocks except when near the large rock near the goal where the rovers turned to the left. Excerpts of the continuous steering and driving navigation performance is shown in Figure 2.

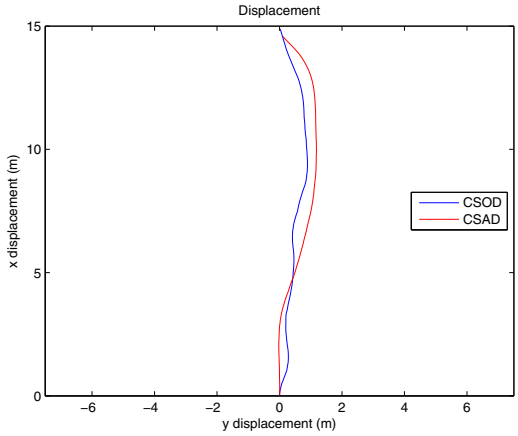


Figure 6: Paths recored by the FPGA visual odometry during autonomous navigation of the environment shown in Figure 4 using the continuous steering or driving (CSOD) and continuous steering and driving (CSAD) planners and the GESTALT traversability analysis. Both navigators exhibit similar maneuvers with the CSOD approach favoring more direct local paths over smoother motions. The accumulated distance over the traverse for the CSOD and CSAD approaches were 14.92 meters and 14.94 meters respectively.

Since both navigators followed similar paths that reached the goal position the average speed of the rover is an indicator for the efficiency of each approach. Figure 7 shows the proximity to the goal as a function of time. The CSOD and CSAD approaches took 287.58 and 231.97 seconds respectively to reach within 0.5 meters of the goal position. In this experiment not stopping to steer-in-place between constant curvature arcs resulting in a 19.3% improvement in traverse efficiency. The average speed of the CSAD approach (6.44 cm/sec) outpaced the CSOD technique (5.19 cm/sec) but is still close to the maximum speed that can be issued by the navigator because little processing time is dedicated to processing range data or estimating pose because of the FPGA co-processor. While a direct comparison to the average speed of autonomous traverse on MER is unfair because of the difference in computational resources and the maximum speed of the rover, the average speed of traverse can begin to approach these values by removing the range estimation and pose estimation bottlenecks through FPGA implementations of stereo and visual odometry algorithms.

The activity of the steering motors is yet another indicator of

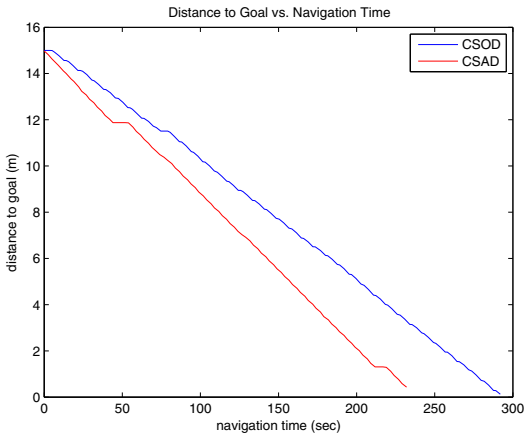


Figure 7: The distance to the goal as a function of time during autonomous navigation of the environment shown in Figure 4 using the continuous steering or driving (CSOD) and continuous steering and driving (CSAD) planners and the GESTALT traversability analysis. The CSOD and CSAD techniques reach within 0.5 meters of the goal in 287.58 and 231.97 seconds respectively.

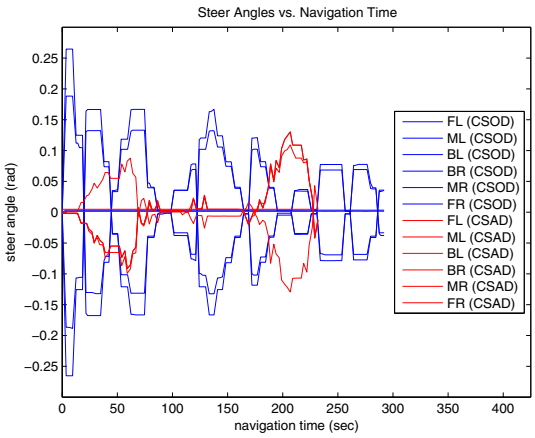


Figure 8: The steering angles of each wheel as a function of time during autonomous navigation of the environment shown in Figure 4 using the continuous steering or driving (CSOD) and continuous steering and driving (CSAD) planners and the GESTALT traversability analysis. Over the course of the autonomous traverse the average total displacement of each steering wheel for the CSOD and CSAD approaches were 2.03 and 0.93 radians respectively.

the efficiency of the navigator. Figure 8 shows the history of the each of the six steering angles for the CSOD and CSAD techniques during the autonomous traverse. The average aggregate displacement of a single steering motors for the CSOD approach is shown to be more than twice than that for the CSAD approach (2.03 radians versus 0.93 radians). Reducing the average displacement of the steering motors may also improve the expected lifetime of steering actuators by eliminating unnecessary steering activity during motion execution.

This experiment provided an example of how the continuous steering or driving and the continuous steering and driving modifications of the GESTALT navigation algorithm can improve the average speed and overall energy efficiency of autonomous traverse. In environments where costly turn-in-place maneuvers are not necessary trajectories that maintain the current curvature can reduce the time that it takes to reach a particular point or travel farther in the same window of time. The performance of both approaches was significantly improved by fast perception and pose estimation performance provided by the FPGA stereo and visual odometry implementations.

4. Conclusions and Future Work

Future missions of planetary exploration will likely require significantly greater traverse rates than those demonstrated by the MER mission. By moving computationally expensive components of the navigation algorithms into a FPGA the performance penalties associated with autonomous traverse can be reduced if not eliminated. This will result in greater average traverse rates and improve power efficiency. Of equal importance is the use of FPGA visual odometry to improve rover safety by enabling continuous obstacle detection and slip checking.

Extensions of this work will involve further analysis of the autonomous navigation algorithm pipeline to determine which components are applicable to the massively parallel implementations on an FPGA co-processor. In particular the traversability analysis and motion planning components appear as worthy candidates based on the inherently parallel nature of goodness evaluation and sampling-based trajectory generation. A new implementation of the co-processor architecture is also under development using a Leon2 CPU and Virtex 5FX130T FPGA to improve the speed of communication between the FPGA and the CPU.

Acknowledgments

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology under contract to the National Air and Space Administration.

References

[1] J. Biesiadecki and M. Maimone, "The Mars Exploration Rover Surface Mobility Flight Software: Driving Ambition," in *Proceedings of the 2006 IEEE Aerospace Conference*, 2006.

[2] Y. Cheng, M. Maimone, and L. Matthies, "Visual Odometry on the Mars Exploration Rovers," in *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, October 2005.

[3] J. Biesiadecki, C. Leger, , and M. Maimone, "Tradeoffs Between Directed and Autonomous Driving on the Mars Exploration Rovers," in *Proceedings of the 2005 International Symposium of Robotics Research*, October 2005.

[4] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo Vision and Rover Navigation Software for Planetary Exploration," in *Proceedings of the IEEE Aerospace Conference*, March 2002.

[5] C. Villalpando, A. Morfopolous, and L. Matthies, "FPGA Implementation of Stereo Disparity with High

Throughput for Mobility Applications," in *Proceedings of the 2011 IEEE Aerospace Conference*, 2011.

[6] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[7] K. Iagnemma, S. Shimoda, and Z. Shiller, "Near-Optimal Navigation of High Speed Mobile Robots on Uneven Terrain," in *Proceedings of the 2008 International Conference on Robotics and Automation*, 2008.

[8] A. Kelly and T. Stentz, "Rough Terrain Autonomous Mobility - Part 2: An Active Vision and Predictive Control Approach," *Autonomous Robots*, vol. 5, pp. 163–198, 1998.

[9] T. Howard, C. Green, A. Kelly, and D. Ferguson, "State Space Sampling of Feasible Motions for High-Performance Mobile Robot Navigation in Complex Environments," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 325–345, 2008.

[10] A. Lacaze, Y. Moscovitz, N. Declaris, and K. Murphy, "Path Planning for Autonomous Vehicles Driving Over Rough Terrain," in *IEEE ISIC/CIRA/ISAS Joint Conference*, 1998, pp. 50–55.

[11] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proceedings of the IEEE International Conference Robotics and Automation*, 2000.

[12] M. Pivtoraiko, R. Knepper, and A. Kelly, "Differentially Constrained Mobile Robot Motion Planning in State Lattices," *Journal of Field Robotics*, vol. 26, no. 3, 2009.

[13] M. Maimone, A. Johnson, Y. Cheng, R. Willson, and L. Matthies, "Autonomous Navigation Results from the Mars Exploration Rover (MER) Mission," in *Proceedings of the 9th International Symposium on Experimental Robotics (ISER)*, June 2004.

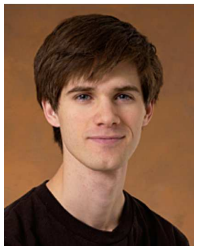
[14] M. Maimone, J. Biesiadecki, E. Tunstel, Y. Cheng, and C. Leger, "Surface Navigation and Mobility Intelligence on the Mars Exploration Rovers," pp. 45–69, March 2006.

[15] L. Matthies, "Dynamic Stereo Vision," Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, October 1989.

[16] B. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, and J. Morrison, "ATHLETE: A Cargo Handling and Manipulation Robot for the Moon," *Journal of Field Robotics*, vol. 24, no. 5, 2007.

BIOGRAPHY

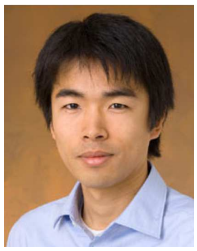
Thomas M. Howard recieved his Ph.D. in Robotics from Carnegie Mellon University's Robotics Institute in 2009 and earned B.S. degrees in Mechanical Engineering and Electrical and Computer Engineering from the University of Rochester in 2004. He is currently a Research Technologist at the Jet Propulsion Laboratory working in the Robotic Software Systems Group. Dr. Howard has seven years experience in robotics research focusing on model-predictive motion planning, navigation, and control of mobile robots and redundant manipulators. He contributed the model-predictive trajectory generator for local motion planning to Boss, winner of the 2007 DARPA Urban Challenge. Dr. Howard also led the motion planning and systems integration areas for JPL/Caltech during Phase I of the DARPA Autonomous Robotic Manipulation - Software Track (ARM-S) program.



Arin Morfopoulos is a member of the Robotic Actuation and Sensing Group at the Jet Propulsion Laboratory. He has been active in the FPGA design and implementation of vision algorithms on a half dozen DoD and NASA projects. He has been responsible for the system interfaces and integration of the FPGA vision algorithms on those tasks since 2007, and is the FPGA design lead on his current task to put stereo, visual odometry and path planning into an FPGA for the Mars 2018 rover mission.

Jack Morrison is a contractor for Northrop Grumman working at the Jet Propulsion Laboratory, designed the software on-board the Sojourner Mars rover, and was one of the primary "rover drivers" during Pathfinder surface operations. He also developed Sojourner's cameras and laser-based hazard avoidance system, and helped implement the 3D rover control station and parts of the ground data system. He currently supports a variety of flight and research robotics projects.

Jack is a native of Southern California, with a 1978 BS degree in Math from UCLA. His career has taken him on adventures in aerospace and commercial development, from air defense systems to video effects boxes to subsea electronics; from 8-bit microcontrollers to supercomputers; from air force bases to investment firms, and from Venus to Mars. His hobbies include computer animation, virtual air combat and literal equestrian jumping.



Yoshiaki Kuwata has 8 years of experience in developing planning and control systems for various types of autonomous vehicles, including small rovers, fixed-wing aircraft, quadrotors, balloon, and sea surface vehicle. He led the development of the planning and control system of the MIT vehicle in DARPA Urban Challenge, which used a simulation-based RRT path planner. He is currently JPLs technical task manager for ONR Littoral Combat Ships, while leading the trade study of EDL and rover mobility capabilities for future planetary missions.

Carlos 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.

Larry Matthies received his PhD in Computer Science from Carnegie Mellon University in 1989, then moved to the Jet Propulsion Laboratory, where he supervises the Computer Vision Group. His research interests include 3-D perception, state estimation, terrain classification, and dynamic scene analysis for autonomous navigation of unmanned vehicles on Earth and in space. His research on real-time stereo vision and accurate visual odometry in the early 1990s led to the incorporation of those capabilities in the twin Mars Exploration Rovers in 2003. His group also developed vision systems for estimating horizontal velocity of the Mars landers during descent and for detecting clouds and dust devils onboard the rovers. He has made numerous contributions to perception for unmanned ground vehicles in DARPA and Army robotics projects since the early 1990s. This includes work on passive 3-D perception with stereo vision day or night and multi-sensor terrain classification for off-road navigation, visual guidance of autonomous stair climbing for tracked vehicles, self-supervised learning of terrain trafficability, and pedestrian detection for safe operation of ground robots. He is an Adjunct Professor of Computer Science at the University of Southern California and is a member of the editorial boards of the Autonomous Robots journal and the Journal of Field Robotics. He is a Fellow of the IEEE and was a joint winner in 2008 of the IEEEs Robotics and Automation Award for his contributions to robotic space exploration.

Michael McHenry received his Ph.D. in Computer Science from the University of Southern California in 1998 and B.S. and M.E.E. Degrees in Electrical Engineering from Rice University. With more than 15 years of experience in software design applied to robotics and scientific computing, Dr. McHenry has contributed to and led a wide range of projects involving machine vision, LIDAR and autonomous navigation in both structured and unstructured environments. He currently leads a Mars Technology Program funded project to deploy FPGA-based vision and navigation processing to future Mars rovers. His research interests include unmanned ground systems perception and navigation, as well as component-based software architectures.