SPARTAN: VISION-BASED AUTONOMOUS NAVIGATION SYSTEM FOR FAST TRAVERSAL PLANETARY ROVERS

Marcos Avilés[1], Manolis Lourakis[2], George Lentaris[3], Xenophon Zabulis[2], Ioannis Stamoulias[3], Konstantinos Maragos[3], Darío Mora[1], Dimitrios Soudris[3]

[1]*GMV Aerospace and Defence, S.A.U., Spain, E-mail: maaviles@gmv.com*

[2]*Foundation for Research and Technology – Hellas (FORTH), Greece, E-mail: lourakis@ics.forth.gr*

[3]*National Technical University of Athens (NTUA), Greece, E-mail: glentaris@microlab.ntua.gr*

ABSTRACT

Autonomous navigation of planetary exploration rovers has become a key element to attain mission success, as it can significantly improve the length of daily traverses, particularly when driving in unknown areas away from the lander. Future missions, such as the Mars Sample Return or the Lunar Polar Sample Return, already demand longer and faster traversals compared to past missions to maximize their scientific return and reduce operational costs and risks. Autonomous navigation is usually based on computer vision due to the passive mode of operation, simple hardware, small form factor and low power consumption of camera sensors. Computer vision nevertheless entails a rather high computational burden which puts a strain on the limited computational resources available onboard these rovers. This paper describes SPARTAN, an optimized, hardware embedded vision system for autonomous navigation of planetary rovers that meets the high demands of future missions requiring large and fast traversals.

1 INTRODUCTION

The exploration of Mars is one of the main goals of both NASA and ESA, as confirmed by past and recent activities as well as future plans. While multiple valuable investigations can be made on the surface of Mars, there is a definite consensus within the scientific community that the major scientific objectives of Martian exploration can only be achieved with the return of samples to Earth [1].

The Mars Sample Return (MSR) mission consists of bringing to Earth a number of Martian samples for thorough analysis in Earth laboratories, without the time, budget, and space constraints of a space mission. Irrespectively of the final mission architecture, it is widely accepted that the MSR mission will feature at least two surface elements, specifically a lander hosting a Mars Ascent Vehicle (MAV) and a Sample Fetching Rover (SFR).

The SFR will collect samples from the surface/subsurface of Mars, or pick up cached samples from a previous mission and return them back to the MAV. Long communication delays, limited bandwidth and narrow communication windows hamper teleoperation. Therefore, emphasis must be given to sufficient autonomous mobility of the rover, which must be at least in the range of future precision landing ellipse dimensions (< 10km) in the case of the SFR collecting cached samples or even up to 20 km in the scenario where the SFR will have to also do the sampling.

Similar demands are made in the context of the Lunar Polar Sample Return (LPSR) mission and its Lunar Prospecting Rover (LPR). The LPR mission concept objectives include determining the distribution of water and other volatiles on a local scale in the lunar polar regions at a location where a general enhancement of water is expected on the basis of orbital and other factors.

As a consequence, future planetary rovers will have to rely on autonomous navigation systems allowing faster traversals compared to current rovers [2], which only travel rather short distances per sol [4]. Basing navigation on computer vision is attractive owing to the passive mode of operation, simple hardware, small form factor and low power consumption of camera sensors. However, vision-based techniques are computationally demanding, which becomes a particularly challenging problem considering the limited computational resources available on board planetary exploration rovers.

The goal of the SPARTAN system is to provide a tight and optimal implementation of computer vision algorithms for rover navigation using vectorial processing. The SPARTAN system has been developed in the framework of different ESA activities, namely SPARTAN, SEXTANT and COMPASS. In this paper, we describe the outcome of the latter two, which are separate activities that supersede the system developed in the initial SPARTAN activity [3], even though the same name has been retained.

2 SYSTEM OPERATION

The importance of having the ability of autonomous navigation is emphasized by the limited

communication possibilities between the rover and the control center on Earth. This reduces the number of remote control commands sent to the rover from human operators, and so affects the efficiency of the rover activity. To maximize the length of traversals and be able to reach the goals set from Ground, the rover needs to be able to autonomously determine the quickest and most efficient path to the target location. On the other hand, it is important that these autonomy algorithms also keep the rover safe at all times and thus need to be designed to always work conservatively preventing any problem from ever occurring.

Based on these constraints, the navigation scenario for a planetary rover is typically divided into the following steps:

1. Global path planning or manual waypoints are determined and computed on Ground from a starting point P_0 to a target point P_g.
2. At P_0 the rover builds a 3D map of the terrain around it using a stereo bench at the top of its mast (navigation cameras or NavCams, placed at a height of approximately 1m). The map covers a distance of approximately 4m and a range of 120º.
3. The system finds a safe path of up to 4m to reach point P_1 taking into account the manoeuvrability of the rover and its ability to traverse the terrain.
4. The rover travels to P_1 relying only on its localization system. The rover location has to be accurate enough to safely reach the point and must be provided in real-time to insure the stability of the trajectory servo-control. A second stereo bench (localization cameras or LocCams) is used for the visual localization function.
5. Steps 2 to 4 are repeated as many times as required, from point P_i to P_{i+1}, until goal point P_g is reached.

Figure 1 illustrates the sequence of navigation tasks described above.

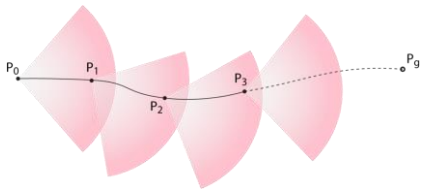

Figure 1: Rover navigation from starting point P_0 to target point P_g via a sequence of intermediate points.

In line with this scenario, SPARTAN provides two different operational modes: the mapping mode, where the system produces a 3D map of the environment, and the localization mode, where the system provides location estimates.

The imaging needs of the two modes are catered to by two pairs of stereo cameras mounted on the rover. Specifically, the high-definition "navigation" cameras are mounted high on a mast and used for terrain mapping, whereas the lower resolution "localization" cameras are placed lower (typically on the rover's body) in order to provide a closer view of the ground.

3 ALGORITHMIC DESCRIPTION

3.1 Mapping Mode

Mapping aims to determine the geometry of the environment around the rover via binocular stereo matching [8] and is essential for obstacle avoidance and path planning. Our stereo matching employs space sweep, an algorithm that performs dense stereo with arbitrary camera configurations [6]. It works by sweeping a set of virtual planes through the scene and measuring the photoconsistency of input images as they are re-projected onto these planes, without requiring prior image rectification. Thus, rather than computing image disparities, space sweep computes depths directly. Space sweep is attractive since it is amenable to parallel implementation that can achieve real-time performance. Hence, parallelized implementations of space sweep have been extensively employed in the real-time reconstruction of large-scale environments from binocular pairs mounted on mobile vehicles [7]. The accuracy of space sweep can be increased when the orientation of major structures in the scene, such as the ground plane, can be assumed known. Furthermore, its computational complexity can be directly modulated with respect to the precision of the obtained depth map, both regarding pixel resolution as well as depth precision. In this manner, it is possible to dedicate shorter computational times to obtain coarser reconstructions of the scene. Thus, an algorithm with anytime characteristics is derived, something that is not trivially feasible with other approaches.

3.2 Localization Mode

Localization is facilitated by visual odometry (VO), a process that determines the 6D position and orientation (i.e., pose) of a camera rig by analyzing the images it acquires over time. Any solution to VO is comprised of three distinct subtopics, namely feature detection, feature matching and motion estimation. Feature detection concerns the automatic extraction of sparse point features and their descriptors from a general scene, feature matching involves tracking them across a set of successive image frames and motion estimation regards the recovery of the incremental pose of the employed cameras as well some partial scene 3D information using structure from motion algorithms. This work employs the Harris corner detector for extracting natural features, SIFT descriptors and the Chi-squared ($\chi 2$) distance for matching and absolute orientation on sparse, stereo-triangulated 3D point clouds for motion estimation. The use of a stereo camera setup permits the depth/scale ambiguity to be resolved, something that is not possible with a single camera [12]. More details for each subtopic are

provided in the following paragraphs.

The Harris corner detector is an established interest point detector that is based on the local auto-correlation function (i.e. intensity variation) of an image [9]. The local auto-correlation function measures the local changes of the image using patches shifted by a small amount in different directions around a pixel. The shifted patches are approximated by a Taylor expansion truncated to the first order terms, which gives rise to a 2x2 matrix known as the structure tensor. The eigenvalues of this matrix capture the intensity structure of a point's local neighborhood. Various corner strength measures (i.e. "cornerness") have been proposed for determining whether a certain image pixel is a corner, avoiding the costly explicit computation of the eigenvalues.

SIFT (acronym for Scale Invariant Feature Transform), is a popular combined detector / descriptor [10]. SIFT computes point features that are invariant to image translation, scaling, rotation and partially invariant to illumination changes and affine transformations. SIFT starts by determining the locations of features as the local extrema of a Difference of Gaussians (DoG) function applied in scale space to a series of smoothed and re-sampled versions of an input image. Following the determination of their location, standard SIFT assigns an orientation to each feature by computing the maximum of an orientation histogram weighted by the magnitude of local image gradients. Feature descriptors are then determined by rotating the region surrounding each keypoint according to its dominant orientation and then dividing it into 4x4 patches. A histogram of image gradient directions quantized to a finite set of orientations is extracted from each such patch and the SIFT descriptor is finally formed by concatenating together the histograms of all patches.

In this work, the time- and memory-consuming scale-space filtering for obtaining the feature locations is avoided and only the descriptor part of SIFT is employed. More specifically, the feature locations are obtained from the Harris operator described above. To further reduce computation time and related FPGA implementation logic, the orientation normalization is removed in the computation of the SIFT descriptor. This choice is justified by the fact that in our application, the orientation of features is not expected to change between images and has been verified experimentally not to impair accuracy. Such a descriptor is referred to as "upright SIFT" in the literature. SIFT descriptors are matched using the standard ratio test defined with their two nearest neighbors. The distance function used is the Chi-squared distance which offers a good performance / computational cost trade-off [11]. In particular, compared to the commonly used Euclidean distance, the Chi-squared distance yields more matches with fewer mismatches. The spatial distribution of the detected Harris corners is improved by filtering them with the adaptive non-maximal suppression (ANMS) scheme of Brown et al. [15], which retains only those corners whose strength is locally maximal. Simpler, albeit less effective schemes based on image binning can be employed in place of ANMS.

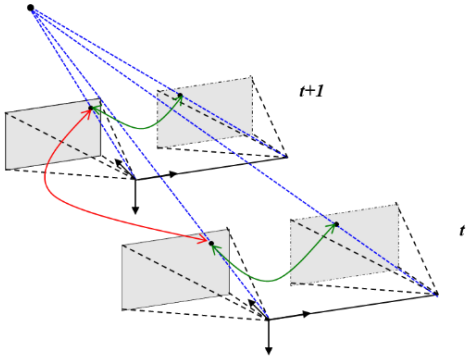

Figure 2: Motion estimation geometry. A 3D point seen in two stereo images over time. Projection rays are in blue, spatial and temporal matches are indicated with green and red arrows, respectively.

As illustrated in Figure 2, using the aforementioned feature detection and matching techniques, a number of points are detected and matched between the two stereo views at a certain instant in time t. Knowledge of the stereo calibration allows the estimation via triangulation of the 3D points giving rise to the matched image projections. As the rover moves, the latter are tracked over time in both stereo views to time t+1. By triangulating the tracked points in time t+1, the camera motion can be computed as the rigid transformation bringing the 3D points of time t in agreement with those obtained at time t+1.

Determining the rigid body transformation that aligns two matched 3D point sets is known as the absolute orientation problem. Its solution involves the minimization of the mean squared error between the point sets under the sought transformation and is expressible by closed form formulae using Horn's quaternion-based solution [13]. Faster algorithms such as [14] are also applicable. To safeguard against outliers, the estimation of motion is embedded in a robust regression framework (i.e. RANSAC [16]).

4 HW/SW IMPLEMENTATION

To meet the mission requirements regarding execution time on resource-constrained, space-grade HW platforms, system implementation was based on a custom HW/SW co-design methodology. Our approach involves in-depth algorithmic cost analysis, detailed HW/SW partitioning, parallel HW architecture design, parametric VHDL coding, integration with certain CPU-FPGA communication links, as well as design space exploration and parameter tuning for optimal adaptation to the Martian navigation scenarios. Our developments are briefly presented in the subsections that follow, more detailed descriptions can be found in [5] and [17].

Regarding the choice of FPGAs, in an extensive comparative analysis involving a large number of diverse processing platforms as candidate accelerators for space avionics [18], we have demonstrated that FPGAs offer the highest performance per watt among all choices. Therefore, they are the most viable option for accelerating computationally demanding operations in space.

4.1 System Description and Partitioning

We target a HW/SW system consisting of a space-grade LEON3 CPU with a processing power of approximately 150 MIPS, as well as an FPGA coprocessor, for which we consider radiation hardened devices from two distinct vendors: Virtex-5QV from Xilinx [23], and NG-MEDIUM from NanoXplore [22]. In both cases, the FPGA resources are limited with respect to the requirements of the aforementioned vision algorithms and mandate considerable design optimization to fit all VHDL components in the given devices. Especially for the NG-MEDIUM, our approach is to perform multi-FPGA partitioning.

Our HW/SW co-design methodology begins with profiling the computer vision pipelines. We combine automatic SW profilers and manual examination of the algorithms to evaluate as accurately as possible the requirements of each SW function. We focus on the analysis of time and memory, however, we also pay particular attention to the communication requirements of each function, to the reusability of variables among functions, as well as to the arithmetic requirements (fixed/floating point operations, dynamic range, accuracy, etc). We highlight that, instead of estimations based on conventional CPUs, the majority of our profiling is performed on an actual space-grade HW, which is highly representative of the platforms being used in similar missions: we port the entire localization and mapping pipelines on a soft-core LEON3 operating at 50 MHz in a Virtex4 FPGA (we project the results at 150 MIPS based on ordinary benchmarking).

Overall, our SW analysis can be summarized in a relatively simple rule, i.e., that most of the functions processing directly the pixels of the image become very slow due to the huge size of the input data and must be accelerated by the FPGA. In contrast, functions processing only those features extracted from the images are less computationally demanding and their mapping to HW or SW depends on secondary factors, such as their communication needs (e.g., their position in the pipeline, their memory footprint, etc.). Finally, functions processing only high-level information and/or solving numerical problems of increased SW complexity are most suitable for execution on the CPU.

Specific results on LEON3 show that space sweep requires more than an hour to extract depth from a stereo pair of 1120x1120 resolution. Therefore, 99.9% of its computations must be accelerated by FPGA, leaving on CPU only the final transformation of depth values to world coordinates (less than 2sec on LEON). For localization, the HW/SW partitioning becomes more challenging; on LEON3, for a single execution of the localization pipeline, the results show that Harris requires 2.8sec to process a stereo pair of 512x384 resolution, SIFT description requires 18.5sec to process 2.5K corners, feature matching requires 7.5sec for both spatial and temporal examination, outlier detection requires 0.06sec, and finally, motion estimation with absolute orientation requires only 0.09sec. Given the 1sec time budget imposed by the mission requirements, as well as all the remaining of our algorithmic analysis, we partition the localization pipeline in HW and SW components as shown in Figure 3 and we target system acceleration factors in the area of 30-60x.

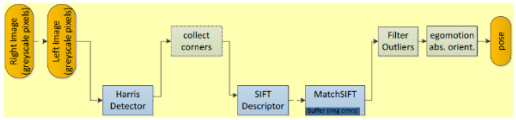

Figure 3: Proposed HW/SW partitioning of the localization algorithm (blue boxes, FPGA; grey boxes, LEON3).

With the proposed HW/SW partitioning, depending on the exact tuning of the algorithmic parameters, 82-99% of the computation is ported on FPGA. As shown in Figure 3, the left and right images of each stereo pair enter serially in the pipeline and an internal FPGA buffer (within matching component) stores all the necessary information for relating two images: it temporarily stores in RAMBs the feature descriptors of the previous image. The "collect corners" block on SW applies any level of sophistication to select the most prominent subset of features detected in each image, e.g., the ANMS algorithm requiring 3sec per image, or simpler adaptive thresholding techniques requiring only few milliseconds. The output of this pipeline is the 6D pose of the rover and is provided once every second. We note that the partitioning and the VHDL components described in the following subsection can be implemented in any FPGA technology, i.e., the well-established Virtex-5QV, the new NG-MEDIUM or the upcoming NG-LARGE, as well as on Commercial-Off-The-Shelf System-on-Chip devices such as the Zynq-7000 series.

4.2 VHDL IP-Cores Implementation

The components were developed with parametric VHDL by combining a number of digital design techniques and parallel architectures. We tackle the FPGA memory limitations by decomposing the input data and transferring each subset from the CPU to the FPGA, separately. That is, each stripe of the image is transferred, processed, and then uploaded to the CPU, before the following data can be processed by the same FPGA resources. We perform pipelining on a pixel basis leading to increased speed-up factors; by their nature, image processing algorithms involve repetitive calculations on successive pixels

and allow for efficient design of highly utilized pipelines. To facilitate the throughput of these pipelines, we develop parallel memories to allow multiple data to be fetched at a single cycle. Also, we parallelize the calculation of the algebraic formulas involved in each algorithm (e.g., the cornerness calculation).

Representatively, Figure 4 depicts the architecture of the Harris component. The pixels enter in a raster-scan order and are constantly forwarded to the stages of the pipeline. The most computationally intensive part is the 2D convolution, which is applied in 5 distinct stages of the algorithm. Hence, in the proposed architecture, the mask multiplication is fully parallelized and the accumulation circuitry is pipelined in order to achieve the throughput of one derivative value per cycle. In particular, the pixels are forwarded in raster-scan order into a serial-to-parallel buffer, which outputs a 5x5 sliding window at each clock cycle. The window is multiplied to the integer coefficient mask in parallel by using 5x5 constant multipliers and a pipelined adder-tree. Hence, our architecture executes one 512x384 image convolution in 512x384 cycles (derivatives and blurring use similar convolution components). We accelerate the computation of each pixel's cornerness value by storing the derivative values in parallel memory banks and allowing the parallelization of the arithmetic formula, which in a pipelined fashion processes one pixel location per cycle. Overall, our Harris component design achieves an efficient balancing of resources and read-cycles, tailored to the rover's requirements. The resulting FPGA cost of the Harris implementation is 14K LUT6 and 48 RAMB18.

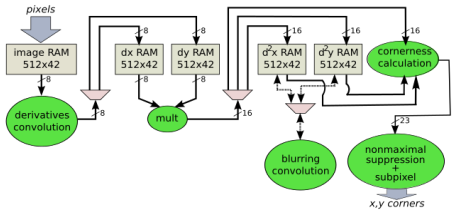

Figure 4: HW architecture of Harris Corner Detection: pipelined processing, with intermediate buffers, starting from image pixels and advancing to x-y vectors.

Similar design approaches were followed for developing the SIFT descriptor, the Matching module and the space sweep algorithm. The resulting costs were 12K LUT6 and 41 RAMB18 for SIFT, whereas the Matching module requires 11K LUT6 and 204 RAMB18 for storing the 128-byte feature descriptors. This increased memory requirement is also reflected on space sweep, which consumes 8K LUT6 and 201 RAMB18. Altogether, the mapping and localization pipelines, including an Ethernet controller and FSM arbiter, were fitted in a Xilinx Virtex-5QV with utilization 42-63% in logic resources and 98% in RAMB18.

When considering smaller FPGA devices, i.e., NG-MEDIUM and/or NG-LARGE, we devised HW/HW partitioning techniques and prototyped various implementations [17] on the HAPS-54 multi-FPGA platform [24]. Indicatively, based on the resources of our components and the size of the aforementioned BRAVE devices, we partitioned the *localization* and *mapping* pipelines as follows:

- 1 NG-MEDIUM, which includes the Ethernet communication module and the *mapping* pipeline, i.e., space sweep, however it excludes the subcomponent for updating the depth map
- 1 NG-MEDIUM, which includes the localization pipeline, however it excludes the Matching component
- 1 NG-LARGE, which includes the map updating and the Matching components, together with a buffer for storing the corners detected on each image

The above partitioning has led to a fully functional multi-FPGA implementation with inter-FPGA data transfers clocked at 150 MHz. The resource bottleneck, as expected, was the memory of the FPGAs with all 3 BRAVE devices being utilized at more than 90%.

5 RESULTS

5.1 Accuracy Evaluation

The accuracy of the mapping pipeline was evaluated with the aid of several synthetic stereo image pairs from two Mars-like terrains of different types ("rocky" and "flat", see Figure 5), for which the true depth maps are available as a byproduct of rendering.

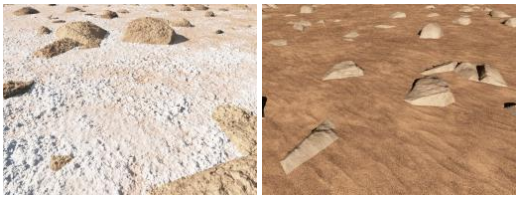

Figure 5: Sample frames from the "rocky" (left) and "flat" (right) terrains used in the experiments.

Figure 6 shows the mean reconstruction error in the 3D points up to 4 m recovered with space sweep from a set of 68 high-resolution (1024x1024) stereo pairs as a function of the distance of these points from the stereo camera system. Owing to the fact that the error in stereo reconstruction increases with the square of depth, the reconstructed points are grouped according to their depths and their errors plotted accordingly. As can be observed from the plot, the mean error always remains less than 1.6 cm and, as expected, is larger for further points.

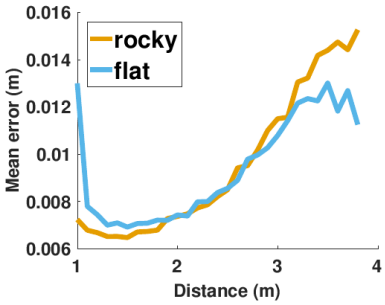

Figure 6: Average reconstruction error for 3D points recovered with space sweep stereo plotted against their distance from the stereo cameras.

To evaluate the accuracy of the localization pipeline, we employed synthetic sequences with a resolution of 512x384 pixels and known pose ground truth that depict the two Mars-like terrains mentioned above. A representative result from such an experiment is shown in Figure 7, which shows the cumulative translational error over time for two trajectories that are 100 m long. From this plot, it can be seen that the errors in both cases remain less than 1.1 m after the rover has traversed the entire trajectory.

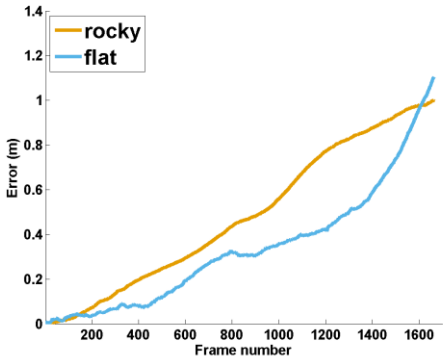

Figure 7: VO cumulative translational error for two 100 m long trajectories.

In addition to the experiments with synthetic images, localization was also tested with real sequences from Mars-analogue terrains from Teide in Tenerife [19], Atacama Desert [19] and Devon Island [21].

5.2 Execution Time

The developed HW/SW system was analyzed with respect to time while processing a challenging sequence from the Atacama dataset (8-bit image pairs of 512x384 pixels each). Our Virtex-5QV implementation achieved 120 MHz clock frequency, whereas the multi-FPGA implementation on HAPS achieved 136 MHz frequency. Therefore, on HW, our feature detection kernel consumed 18 msec, feature description 72 msec, SIFT matching 170 msec, and the overhead of HW-SW communication was 83msec (with Ethernet at 81 Mbps). The 150 MIPS LEON could complete filtering and egomotion estimation in 150 msec. We note that all of the above refer to the processing of 1243 features per image, on average, which imply almost 1K matches (control points) per rover step. In total, the developed HW/SW system was able to process the dataset at 2 FPS. When applying a more sophisticated selection of features, i.e., an initial crude selection on HW of ~1500 features via adaptive thresholding followed by a refined selection on SW via the ANMS algorithm, the processing rate decreased to 1 FPS in exchange for an increase in the robustness of localization.

For mapping, our space sweep implementation completed the processing of three high-definition pairs of images (resolution 1120x1120) in less than 12sec on HW, with an overhead of 6 sec on LEON for the coordinate transformation. We stress that the communication overhead in this case is zero, because we have achieved to pipeline the steps of CPU-FPGA transferring and processing, so that the communication time was completed masked.

The aforementioned results show immense acceleration factors when comparing to the all-SW execution on 150-MIPS LEON3, i.e., almost 800x for mapping mode and 60x for localization mode. Specifically for the latter, our feature detection kernel was accelerated by almost 160x, description by 260x, and matching by 70x. As a result, we demonstrated that the proposed system meets all mission requirements on time: 1 sec per step for localization mode and 20sec per step for mapping mode, by considering space-representative platforms, regardless of single- or multi-FPGA technology.

6 CONCLUSIONS

We have briefly presented SPARTAN, a highly optimized, hardware embedded vision-based system for autonomous navigation of planetary rovers meeting the ambitious demands of future missions where large and fast traversals are required.

SPARTAN is a SW/HW co-design combining the high computing power and low power consumption of an FPGA together with the higher floating point accuracy and flexibility of a processor. The choice of the employed algorithms was based not only on their raw performance in terms of accuracy or robustness, but also on their suitability for being implemented in a FPGA. This allows for a much more efficient system in terms of computing power, memory footprint, communication needs, energy use, speed and reconfigurability. The hardware logic has been optimized to fit into current space-grade FPGA devices but also into networks of smaller devices comparable in size and performance to the next generation European space-grade FPGA devices co-funded by ESA (i.e., the BRAVE family of FPGAs). The SW elements of the SPARTAN design were implemented on a LEON processor running RTEMS, thus resulting into a space-ready navigation component.

Acknowledgements

This work was supported by the European Space Agency via the SEXTANT and COMPASS activities of the ETP-MREP research programme. The authors would like to thank Kostas Siozios and Dionysios Diamantopoulos for their contribution in system integration at NTUA during the SEXTANT and COMPASS activities, Emmanouil Hourdakis for his assistance with the evaluation of results at FORTH during COMPASS, Javier Herrero for his contribution in the system integration at GMV during COMPASS, as well as Gianfranco Visentin from ESA for his valuable insights and comments during all activities.

References

[1] Ellery A (2015). Planetary Rovers: Robotic Exploration of the Solar System. Springer.

[2] Wong C et al. (2017), Adaptive and Intelligent Navigation of Autonomous Planetary Rovers - A Survey, NASA/ESA Conf. on Adapt. Hardw. Syst., Pasadena, CA, pp. 237-244.

[3] Kostavelis I, Nalpantidis L, Boukas E, Avilés M, Stamoulias I, Lentaris I, Diamantopoulos D, Siozios K, Soudris D, and Gasteratos A (2014), SPARTAN: Developing a Vision System for Future Autonomous Space Exploration Robots, J. Field Robot, 31:107–140.

[4] Matthies L, Maimone M, Johnson A, Cheng Y, Willson R, Villalpando C, Goldberg S, Huertas A, Stein A and Angelova A (2007). Computer Vision on Mars. Int. J. Comput. Vis., 75(1):67–92.

[5] Lentaris G, Stamoulias I, Soudris D, and Lourakis M (2016). HW/SW Codesign and FPGA Acceleration of Visual Odometry Algorithms for Rover Navigation on Mars. IEEE Trans. Circuits Syst. Video Technol., 26(8):1563–1577.

[6] Gallup D, Frahm JM, Mordohai P, Yang Q and Pollefeys M (2007), Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions. IEEE Conf. on Comput. Vis. and Pat. Rec., Minneapolis, MN, pp. 1-8.

[7] Pollefeys M et al. (2008), Detailed Real-time Urban 3D Reconstruction from Video. Int. J. Comput. Vis., 78(2-3):143–167.

[8] Scharstein D and Szeliski R (2002). A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. Int. J. Comput. Vis., 47(1-3):7–42.

[9] Harris C and Stephens M (1988). A Combined Corner and Edge Detector. In Proc. 4th Alvey Vis. Conf., U. of Manchester, UK, pp. 147–151.

[10] Lowe DG (2004). Distinctive Image Features from Scale-Invariant Keypoints. Int. J. Comput. Vis., 60(2):91–110.

[11] Rubner Y, Puzicha J, Tomasi C and Buhmann J (2001). Empirical Evaluation of Dissimilarity Measures for Color and Texture. Comput. Vis. Image Und. 84(1), 25–43.

[12] Lourakis M and Zabulis X (2013). Accurate Scale Factor Estimation in 3D Reconstruction. Int'l Conf. on Comput. Anal. Images and Pat., York, UK, pp. 498-506.

[13] Horn B (1987). Closed-Form Solution of Absolute Orientation Using Unit Quaternions. J. Optical Soc. Am. A 4(4):629–642.

[14] Lourakis M (2016). An Efficient Solution to Absolute Orientation, IAPR/IEEE Int'l. Conf. on Pat. Rec., Cancun, Mexico, pp. 3816-3819.

[15] Brown M, Szeliski R and Winder S (2005). Multi-image Matching Using Multi-scale Oriented Patches. IEEE Conf. on Comput. Vis. and Pat. Rec., San Diego, CA, pp. 510-517.

[16] Fischler MA and Bolles RC (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Commun. ACM, 24:381–395

[17] Lentaris G, Maragos K, Soudris D, Zabulis X and Lourakis M (2018). Single- and multi-FPGA Acceleration of Dense Stereo Vision for Planetary Rovers. *Under review*.

[18] Lentaris G, Maragos K, Stratakos I, Papadopoulos L, Papanikolaou O, Soudris D, Lourakis M, Zabulis X, Gonzalez-Arjona D and Furano G (2018). High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation. AIAA J. Aerosp. Inf. Syst., 15(4):178-192.

[19] Gandia F (2018). The Lucid Field Test Campaign – Results of Operations with a Rover in a Similar Lunar Environment, Int'l Symp. on AI, Robot. and Autom. (to appear).

[20] Woods M et al. (2014), Seeker—Autonomous Long - range Rover Navigation for Remote Exploration. J. Field Robot., 31: 940-968.

[21] Furgale PT, Carle P, Enright J, and Barfoot TD (2012). The Devon Island Rover Navigation Dataset. Int. J. Rob. Res., 31(6):707-713.

[22] Lepape E (2016). NanoXplore NXT-32000 FPGA (NG-MEDIUM) Presentation, https://indico.esa.int/indico/event/130/session/11/contribution/59/material/slides/0.pdf, 3rd SEFUW workshop, ESTEC, ESA.

[23] Xilinx (2018). Space-grade Virtex-5QV FPGA, https://www.xilinx.com/products/silicon-devices/fpga/virtex-5qv.html

[24] Synopsys (2018). HAPS Prototyping Solutions, https://www.synopsys.com/verification/prototyping/haps.html