SPARTAN Project: Efficient Implementation of Computer Vision Algorithms onto Reconfigurable Platform Targeting to Space Applications

K. Siozios, D. Diamantopoulos, I. Kostavelis[¥], E. Boukas[¥], L. Nalpantidis[¥], D. Soudris, A. Gasteratos[¥], M. Avilés[±] and I. Anagnostopoulos

School of Electrical and Computer Engineering, National Technical University of Athens, Greece
[¥] Dep. of Production and Management Eng., Democritus University of Thrace, Greece
[±] GMV, Spain

***Abstract*—Vision-based robotic applications exhibit increased computational complexity. This problem becomes even more important regarding mission critical application domains. The SPARTAN project focuses in the tight and optimal implementation of computer vision algorithms targeting to rover navigation for space applications. For evaluation purposes, these algorithms will be implemented with a co-design methodology onto a Virtex-6 FPGA device.**

Keywords: Computer Vision Algorithms, Space Application; FPGA implementation; Low-power; High-Performance

I. INTRODUCTION

The exploration of Mars is one of the main goals for both NASA and ESA, as confirmed by past and recent activities. The last 15 years there are a number of on-orbit and surface missions to Mars with remarkable results. The impressive success of previous missions (NASA's Mars Global Surveyor, Mars Odyssey, Phoenix, Mars Reconnaissance Orbiter, as well as ESA's Mars Express) enables a continued investment of efforts in this direction, and also facilitating international discussion for joint Mars exploration missions between ESA and NASA, such as in case of ExoMars (scheduled for 2018 [3]).

However, in order to accomplish this goal, a number of critical issues have to be resolved. Among others, efficient autonomous robots have to be designed. Autonomous robots' behavior greatly depends on the accuracy of their decision-making algorithms. Hence, vision-based solutions are becoming more and more attractive due to their decreasing cost, as well as their inherent coherence with human imposed mechanisms. In the case of stereo vision-based navigation, both the accuracy and the refresh rate of the computed disparity maps are the cornerstone of success. However, robotic applications place strict requirements on the demanded speed and accuracy of vision depth-computing algorithms.

In order to support this goal, the SPARTAN project aims to provide a sufficient implementation of computer vision (CV) algorithms for rover navigation. Rather than similar approaches that tackle this problem solely in software-level, our objective is to develop a number of CV algorithms in ESA compatible VHDL format, whereas the implementation of SPARTAN system will be demonstrated using a co-design methodology and a Virtex-6 FPGA device.

The rest of the paper is organized as follows: Section II describes the global SPARTAN system, whereas section III gives an overview of alternative computer vision algorithms candidate for rover navigation. Section IV discusses the implementation of the CV algorithmic approaches onto a Virtex-6 FPGA. Finally, conclusions are summarized in section V.

II. GLOBAL SYSTEM DESCRIPTION

The goal of the SPARTAN processing algorithms is to convert visual information from the rover cameras into 3D local maps and perform accurate location estimates useful for the navigation process. Image processing algorithms suitable for localization and 3D map reconstruction are selected and implemented into a parallel processing chain to achieve high performance while maintaining efficiency in terms of energy use, computing power and memory footprint.

The CV algorithms will be partitioned into a SPARTAN processing software component and a VHDL model component. The former will run on a standard central processing unit with the Linux operating system, whereas the latter will be implemented in a recent FPGA device. The system will use stereo-vision image information in order to feed SPARTAN algorithms. From a systems' perspective, SPARTAN algorithms need to interact with other components of the rover navigation application for them to be useful. Interfacing between the rover sensors and the processing elements will be accomplished through central processing component.

Figure 1 shows the interaction between different building blocks of SPARTAN system. All relevant SPARTAN functionalities have been grouped into a SPARTAN Localization and Mapping Engine. This block contains all subsystems needed for the realization of the algorithmic functions. As part of it, and resulting from a HW/SW partitioning procedure, several processes will be handled by the central processing software component, while others will be integrated into FPGA hardware.

The SPARTAN central processing will be built on top of ROS (Robot Operating System) framework [4], whereas the selected FPGA for implementing VHDL algorithms is a recent Virtex-6 device [20]. ROS provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. More specifically, the computer vision algorithms that will be implemented during the SPARTAN project are summarized as follows:

- Imaging: Implementing suitable local image processing that can serve image products.
- Visual Odometry: Provide an estimation of the displacement of the rover.
- Visual SLAM: Determine the current location of the rover.
- 3D Map reconstruction: Reconstruct the 3Dimensional shape of the terrain being imaged in front of the rover.
- Localization: Locates the new spatial location of rover at the map.

Finally, regarding demonstration purposes, the efficiency of SPARTAN system will be evaluated under two test platforms:

- Exomars-like rover under the 3DROV simulator: This first scenario involves a simulated rover (Exomars CAD model based) and a Martian surface modelled on the 3DROV framework. Realistic camera images will feed the vision based SPARTAN algorithms. This scenario will allow early testing of the algorithms during the development process.
- Outdoor ERA-Mobi rover: This second setup will test the same algorithms on a real rover in an outdoor environment.

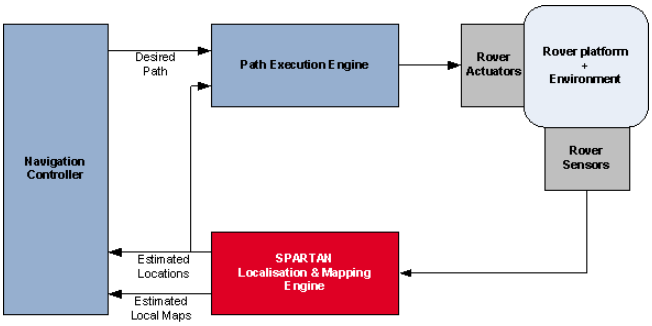

Figure 1. SPARTAN functional environment

III. COMPUTER VISION ALGORITHMS FOR SPARTAN

The main objective of our solutions is to achieve the desired rover navigation function, while reducing as much as possible the required overall budgets. For this purpose, a number of novel computer vision algorithms will be developed both in C/C++, as well as VHDL language [1].

This section provides a comprehensive analysis of the previously presented algorithms from the point of view of a hardware implementation, first presenting a set of criteria under which trading them off, and secondly performing a scoring procedure to identify which of those algorithms are better suited for the SPARTAN system. This study is performed under numerous design parameters that are summarized in Table I.

Since the previously discussed trade-off criteria impose a number of different implementation selections, next we have to clarify a potential scenario for weighting the importance of these criteria. Such a scenario will be also used during the evaluation of CV algorithms, as well as the in order to guide the selections performed during the physical co-design procedure (e.g. partitioning step). Table II summarizes the weights of importance of these trade-off criteria. The criterion with weight "8" indicates great importance for the SPARTAN system, and the criterion with weight "1" indicates minimal importance for the SPARTAN system. The algorithm that will be chosen is that that maximizes the following quantity S:

$$S = \sum_{i}^{N} \{W_i \times b\} \quad (1)$$

where W is the weight importance of every criterion, b is the binary decision for the criterion (i.e. "1" means that the criterion is fulfilled and "0" means that the criterion is not fulfilled) and $i=1, 2, ..., N$ the ID of the criteria. According to the previous equation the maximum score for an algorithm could is $S = 55$ and the minimum score is $S = 0$.

The most important goal for implementing a CV algorithm onto FPGA is the significant performance improvement. For this purpose, we will perform an appropriate analysis of candidate CV algorithms in order to evaluate a number of design parameters. Critical parameters among others are the computational complexity, as well as any other performance bottlenecks. This analysis will be performed for two scenarios: (a) each CV algorithm along, and (b) the entire SPARTAN system.

Based on the previous analysis, it is possible to identify either a number of functionalities per CV algorithm, or a number of CV algorithms, which are suitable for implementing in FPGA. Typical criteria for this selection were discussed in Table I. For instance, regarding the SLAM block discussed in this example, the components which are candidate for being implemented onto FPGA are the "Absolute Differences Computation", the "Aggregation of Absolute Differences" and the "Pixelwise Minimum Search", since they exhibit increased parallelism. Furthermore, the performance of these components can be further improved by performing pipeline extraction.

On the other hand, components of the SLAM block that found not to contain functionalities that are easily parallelized, or it is not possible to improve their performance by using pipeline techniques are candidate for implementation onto the CPU. For instance, regarding the SLAM depicted in Figure 2, the "Reliability Validation" and "Point Clouds Fitting" are such functionalities.

TABLE I. EMPLOYED TRADE-OFF CRITERIA FOR IMPLEMENTING CV ALGORITHMS

Criterion:	Description:	Importance:
Performance Improvement	• Different instantiations of the CV algorithms are available, each of which exhibits variations in terms of numerous design parameters (e.g., maximum operation frequency, power consumption, memory and bandwidth requirements).	The selection of CV algorithms should meet performance requirements/specifications posed by SPARTAN system.
Efficiency	• Another trade-off criterion affects the efficiency of studied CV algorithm to achieve the desired functionality. Note that such an efficiency does not impose the performance in terms of operation frequency, power consumption, memory/storage/IO requirements, etc.	It is very important in order the selected algorithm to meet the desired functionality.
Parallelism Extraction	• The fundamental way to improve the performance of the computer system is to exploit the parallelism. There are two kinds of parallelism: spatial parallelism and temporal parallelism. • Spatial parallelism tries to use duplicated function units (FUs), multiple cores or even multiprocessors to run at the same time on different data sets. • Temporal parallelism tries to use multistage pipeline or macro-pipeline to partition application to multiply phases and data sets and run simultaneously. • Clearly, temporal parallelism and spatial parallelism can yield the same potential speedup. Matrix multiplication and summaries, which are mostly used in CV algorithms, are typical kernels for performance improvement through parallelism extraction.	It is important to extract parallelism for CV algorithms (or parts of them) that will be implemented onto the FPGA.
Pipeline Extraction	• Apart from the latency improvement (i.e. input to output delay) achieved with parallelism extraction, applications also introduce limitations about their throughput (i.e. number of operations per time unit). For instance, since cameras typically provide stream of pixels, ideally they could be processing in pipelines. • Assuming that the latency of a module is T, then in order to achieve a throughput that is higher than 1/T, a number of operations must be overlapped with pipelining. • In order to achieve mentionable performance improvement, pipeline stages should exhibit the same delay; otherwise, we have to guarantee that the pipeline period (determined by slowest stages) meets system specifications.	Pipeline extraction is an ideal solution for increasing the throughput.
Regularity	• Algorithms (or parts of them) that are composed of regular and efficiently replicated circuit blocks are candidate for implementation onto FPGA.	Regularity is an important criterion for CV algorithms implemented onto FPGA.
Complexity Reduction	• The complexity of employed CV algorithms is another parameter that highly affects the efficiency of derived solution. • A number of source-to-source optimization techniques could be employed in order to reduce algorithmic complexity. Special care should be taken in order not to affect the functionality of final solution.	It is critical in order to alleviate requirements about operation frequency.
Memory Management	• The memory bottleneck is a usual problem for CV algorithms. This problem becomes even more important with the demand for bigger images, as well as for images with higher resolution. A number of source-to-source optimization techniques could be employed in order to reduce the memory requirements for selected algorithms. Special care should be taken in order to guarantee the functionality of final solution.	It is critical in order to alleviate memory storage limitation.
I/O Requirements	• The requirements for I/O communication among the building blocks of CV algorithms, affects highly the selection of implementation medium. • For instance, usually it is not preferable to transfer excessive high amount of data between FPGA (where some of the kernels will be implemented) and PC (where the camera is located). • Hence, either careful selection of CV algorithms (without excessive high amount of data among kernels), or partitioning, should be applied in order to guarantee that these algorithms will be implemented onto FPGA. • This problem could be even more important when we transfer images with increased size, resolution, color depth, etc.	Constraint to this selection is the selected interface medium between FPGA and PC.
Power Budget	• Since different instantiations of CV algorithms exhibit requirements regarding the data transfers and/or computations, each of them has unique power dissipation. • This is especially important, since the power budget of SPARTAN system is considerable limited. • As an improvement to this goal potentially we could employ some pre-defined features provided in FPGA devices (e.g. clock- and/or power-gating).	This criterion is important if our design is limited with a small power budget.
Adaptive at Run-time	• Both the selected CV algorithms, as well as the target implementation medium (FPGA + PC) should support some form of addictiveness. For instance, based on the availability of hardware resources, the system has to balance the performance and/or the accuracy. A suitable solution to this procedure is to build SPARTAN system into a modular fashion.	Important in order to enable different operation modes.
Available Implementation	• In case there are available implementations for the studied CV algorithms, it is possible to perform a more accurate profiling of this algorithm, as compared to the case where profiling is performed solely on the algorithm description in relevant references.	It is critical in order to perform a more accurate profiling.
Additional Criteria	• Finally, any other algorithmic-specific additional criteria should be taken into account.	-

Finally, there might be some other functionality, like the "Features Detection & Matching" for our example, which is not clear if they will be implemented onto the FPGA platform or the CPU. The final decision about these functionalities is based on the rest trade-off criteria depicted in Table 2 (labeled as "3" and "4" in the corresponding ordering list). Next, the CV trade-offs regarding three building kernels of CV algorithms are discussed.

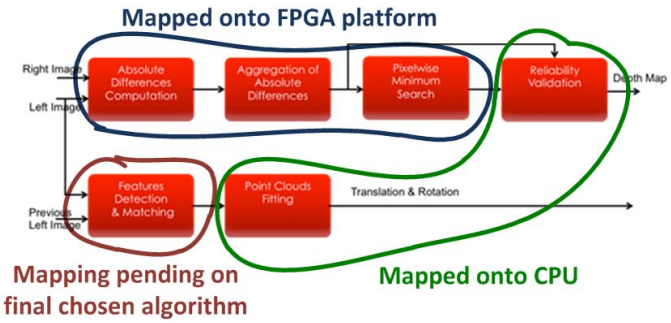

Figure 2. Block based partitioning of the visual SLAM function

A. *Evaluation of 3D Reconstruction*

Local algorithms aiming to 3D reconstruction are generally preferable for autonomous robotic applications, due to their higher refresh rates. As a result, a number of local methods have been selected for further inspection. Each of these algorithms has been evaluated according to set of defined trade-offs and criteria (depicted in Table I).

In order to show the importance among the trade-off criteria discussed previously, Table II gives their ordering importance. More specifically, the most important criterion affects the possibility of studied algorithm to meet the performance specifications required from SPARTAN system, while as a second criterion we have the accuracy. Next, trade-off criteria 3-6 are tightly firmed to the complexity of algorithmic solution (in terms of mathematical functions), whereas the criteria 7-8 affect the demand for resources (memory, storage and I/O, respectively). The opportunity of studied algorithm to be executed either as a stand-alone tool, or folded inside another algorithm, corresponds to criterion 9. Criterion 10 gives the power consumption for the selected algorithm. The availability of an existing implementation, as well as the difficulty to implement it from scratch is included in trade-off criterion 11. Finally, any other trade-off parameter corresponds to criterion 12.

Taking into consideration the previously mentioned trade-off parameters, a number of conclusions for the various methods are summarized in Table III. The values at this table corresponds to score based on the efficiency for each algorithm. Here, variables marked with 1 indicate very low efficiency and variables marked with 5 indicate the higher efficiency. The last column presents a total score for all the algorithms.

From the scores in this table it is not easy to conclude the most efficient algorithm. It is a trade-off between lots of different parameters:

- Considering that the most important parameter is the complexity and the speed then the most suitable choice is the [5] (folded in the OpenCV library). Also it must be pointed out that this algorithm is strongly depended on the texture and for that reason does not ensure the stability of the SPARTAN System 3D reconstruction module.
- Considering that the most important parameter is accuracy then [6] algorithm is the most appropriate. On the other hand this algorithm cannot produce fast stereo output that meets SPARTAN System requirements.
- Considering memory and storage requirements [9] algorithm is the less demanding one in resources. Also this algorithm could produce accurate stereo output able to meet SPARTAN System speed requirements.

TABLE II. ORDERING THE IMPORTANCE OF TRADE-OFF ALGORITHMS

ID	Name	Goal	Weight of Importance
1	Performance Improvement	Achieve (at least) the performance posed by SPARTAN specifications	8
2	Efficiency	Evaluate the algorithm efficiency	7
3	Parallelism Extraction	Improve the performance	6
4	Pipeline Extraction	Improve the throughput	6
5	Regularity	Extraction of regularity	5
6	Complexity Reduction	Reduce the complexity of CV algorithms	5
7	Memory Management	Minimize the memory requirements	4
8	I/O Requirements	Reduce the I/O requirements	4
9	Power Budget	Reduce power consumption	4
10	Adaptive at Run-time	Operation under different scenarios	3
11	Available Implementation	Evaluate the existence of this algorithm	2
12	Additional criteria	Any other additional criteria (e.g. CPU requirements)	1

TABLE III. EVALUATION FOR THE 3D RECONSTRUCTION ALGORITHMS

Method	Function-alities	Maximum Speed	Memory Requirements	Memory Storage	Accu-racy
ID (from Table II)	2,3,5,6	1	8	8	4
[5]	4	5	3	3	N/A
[6]	1	1	3	3	5
[7]	3	2	3	3	3
[8]	3	3	1	1	N/A
[9]	2,5	3	5	5	4

These conclusions could be further summarized and raked according to Table II in the next table taking into consideration SPARTAN system constrains.

TABLE IV. 3D RECONSTRUCTION ALGORITHMS SCORE, BASED ON THE ORDERING CRITERIA (TABLE II)

Met-hod	Criteria based on Table II												Weight Sum	Decision
	1	2	3	4	5	6	7	8	9	10	11	12		
[5]	1	0	1	0	1	1	0	0	1	1	1	1	33	Accept
[6]	0	1	0	0	1	0	0	0	0	1	1	1	18	Reject
[7]	0	1	0	1	1	0	0	0	1	1	0	0	25	Reject
[8]	0	1	0	0	1	0	0	1	0	1	0	0	19	Reject
[9]	1	1	1	1	1	1	1	1	1	1	1	0	54	Accept

B. *Evaluation of Visual Odometry*

Visual Odometry encloses four different modules as described in Figure 3. The first module is the landmark

detection, which has to do with the feature extraction methods of the scene. The second module is the 3D reconstruction of the scene, whereas the third module is the matching routines. These routines comprise algorithms that match the features in the sequential frames. The final module takes into consideration the information from the previous module and comes up with an estimation of the camera motion in every next step.

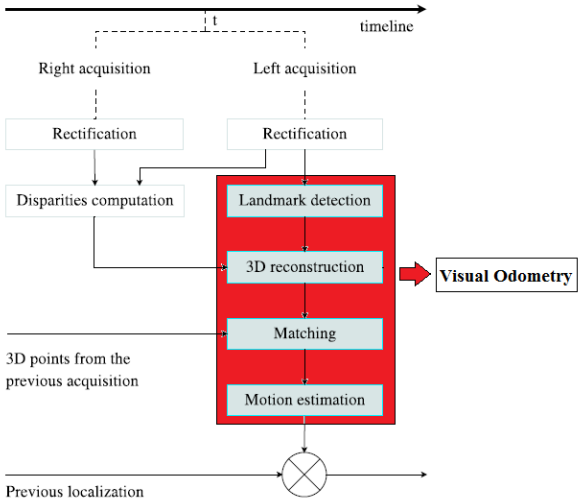

Figure 3. Visual Odometry module

In order to come up with a reasonable selection every different part should examined separately. The aforementioned Visual Odometry algorithms consist of separate modules. The analytically examination of every part separately will help to understand better SPARTAN constrains and potentials and can conclude to an efficient combination of these parts.

The most common detectors used in order to solve the localization problem are the SIFT [10], SURF [11] and Harris [12] corner algorithms. The first two detectors are accompanied from their own description and matching algorithms. Harris corner algorithm is only a detector which find corners and the most common way to match those points of interest is the correlation method. In this section a analysis of those three methods has been done in order to make decision for the most profitable method. All the algorithms have been examined in the same sequential images captured from the real Mars-like environment.

Based on our analysis, we found that SIFT algorithm detects more features than SURF and Harris detectors. On the other hand, SURF and Harris algorithms are almost ten times faster than SIFT. As far as it concerns the matching procedure SIFT succeeds more matches than SURF but SURF is real faster than SIFT. Correlation procedure for Harris detector in order to count the matches in sequential images has greater computational cost than SIFT and SURF.

These conclusions could be further summarized and raked according to Table II in the next table taking into consideration the SPARTAN system constrains.

TABLE V. FEATURE EXTRACTION ALGORITHMS SCORE, BASED ON THE WEIGHTING CRITERIA (TABLE II)

Met-hod	Criteria based on Table II												Weight Sum	Decision
	1	2	3	4	5	6	7	8	9	10	11	12		
[10]	0	1	1	1	1	0	0	0	0	0	1	1	27	Weak accept
[11]	1	1	1	1	1	0	0	0	0	1	1	0	37	Strong accept
[12]	0	0	1	0	0	1	1	0	1	1	0	0	22	Reject

The accuracy of matching in Visual Odometry techniques could significantly influence the total ability of a vision based algorithm to estimate the robot motion. Consequently, it is important to choose a method which could ensure robust matching of the detecting landmarks of a scene.

Usually, the landmark detection techniques provided with a matching routine. For these routines there is a trade between the number of correct matches and the time consumption. Although there are various matching techniques, each one is more efficient in solving different problems. Table IV summarizes and explains the advantages and denotes the disadvantages for most important techniques. We have assigned a value to each method based on Likert scale. Here, variables marked with 1 indicate very low efficiency and variables marked with 5 indicate the higher efficiency.

TABLE VI. MOTION ESTIMATION ALGORITHMS SCORE, BASED ON THE WEIGHTING CRITERIA (TABLE II)

Met-hod	Criteria based on Table II												Weight Sum	Decision
	1	2	3	4	5	6	7	8	9	10	11	12		
[13]	0	1	0	1	1	0	0	0	1	0	1	1	26	Weak Accept
[14]	0	0	0	0	1	0	1	0	0	1	0	0	12	Reject
[15]	0	1	0	1	1	0	0	1	0	0	0	0	23	Reject
[16]	1	1	1	0	1	0	1	1	1	1	1	0	35	Accept

From the previous table, we could easily come up with the conclusion that all the matching techniques have some advantages and disadvantages. The final selection is depended on the criteria that are more crucial for the SPARTAN system.

The last task in Visual Odometry involves the motion estimation. Hence, it is crucial to quantify the accuracy of these motion estimation methods respecting the computational cost.

Based on our exhaustive exploration, we found that the motion estimation module of Visual Odometry is very time consuming. Consequently, the proper selection of such an algorithm is crucial for the efficiency of the SPARTAN System. Based on this analysis, solutions affecting [16] and [17] seem to be the most candidates for hardware implementation. However, up to now, there is no comparison in term of the efficiency for these algorithms.

C. *Evaluation of Visual SLAM*

The SLAM problem is that of estimating a robot position and of progressively building a map of its environment. Typically is the extension of Visual Odometry since it takes into consideration all the past times t-1, t-2, t-N. Because it takes into consideration all the past instances the computational load of SLAM is very problematic. It has been developed both probabilistic and non-probabilistic methods. Generally, probabilistic methods demand more computation time and computational load as they take into account the inherent uncertainty in robot pose. On the other hand, non-probabilistic methods are less demanding, a fact that justifies their implementation in the SPARTAN Visual SLAM system. While there is an extended literature about SLAM methods, the purely vision-based ones (i.e. without measurements of any other sensors but the cameras) are limited.

The Visual SLAM problem is tightly interconnected with the aforementioned 3D reconstruction and Visual Odometry functionalities. Having covered these two modules, the missing part in order to obtain visual SLAM is the mapping of the environment. As a consequence, mapping algorithms are considered in this section, which in conjunction with the other two modules will comprise the visual SLAM module.

The main drawbacks of the probabilistic methods are, that the complexity is depended on the number of samples, and that the aforementioned assumptions may not always be correct (i.e. the Gaussian noise assumption or the considerations of linearity in non-linear circumstances). Such methods would limit SPARTAN system speed. Also these methods do not keep up with SPARTAN objectives to decrease computation time by taking into account energy consumption. Due to these constrains the examination of the non-probabilistic method for visual SLAM are more reasonable. Table VII summarizes the proposed non-probabilistic visual SLAM algorithms taking into consideration some characteristics (i. e. computational cost, efficiency, etc).

TABLE VII. COMPARISON OF NON-PROBABILISTIC VISUAL SLAM ALGORITHMS

Met-hod	Application	Computational Cost (1-high , 5 low)	Accuracy Range Resolution / Distance	VHDL development difficulty (1-high, 5 low)	Parallelism Extraction (1-low improvement, 5 high improvement)
[19]	Single robot usage	3	Simulation estimations	2	2
[20]	Single/Multi robot usage	3	Simulation estimations	2	2
[16]	Single robot usage	3,5	0.016m/2m	3,5	4

IV. PHYSICAL IMPLEMENTATION

The implementation of computer vision algorithms will be performed into VHDL description, while all the ESA requirements/guidelines will be meeting [2]. The purpose of these guidelines is to ensure a good coding standard for VHDL, w.r.t. to readability, portability and a high level of verification. Detail stages for system development (e.g. VHDL models for component simulation, board-level simulation, system-level simulation and test-benches) will be also described.

The target platform for our study is Xilinx Virtex-6 FPGA device [20]. Furthermore, during the SPARTAN activity we will analyze the efficiency of design implementation into such a new-technology (with 3-D integration) platform.

The proposed co-design VHDL model is depicted in Figure 4, while it consists of the following steps: profiling, behavioral optimization, partitioning, kernels mapping into FPGA, kernels mapping into CPU, system integration and back-end tools and configuration and run-time execution. This methodology provides a number of trade-offs that balance various design criteria among the FPGA device, the employed CV algorithms, as well as the entire SPARTAN system. More specifically, these trades-off parameters are summarized as follows: complexity reduction, performance improvement, memory management, parallelism and pipeline extraction.

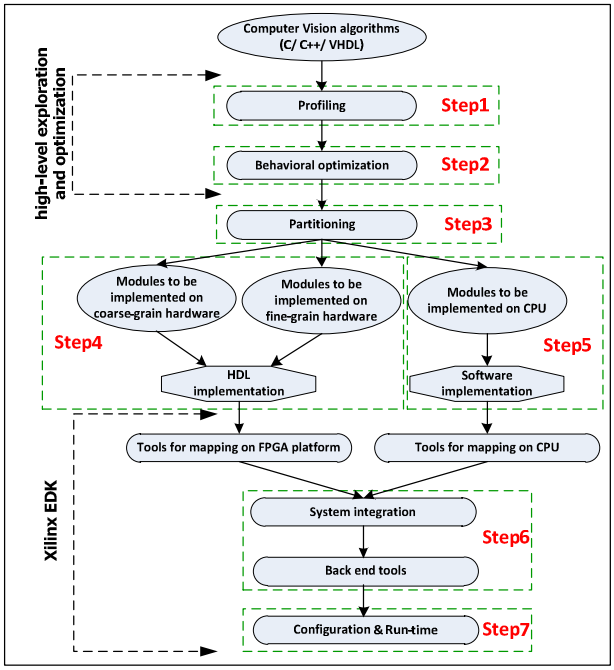

Figure 4. The employed co-design methodology

The input to our methodology is a high-level description of SPARTAN's architecture in C/C++, as well as VHDL format. Since the goal of this project is to provide efficient computer vision algorithms with increased performance, our methodology identifies those functionalities that that introduce performance bottleneck in systems' execution and handled appropriately through source-to-source modifications [1]. These modifications will take into account inherent constraints/features taken by the target platform.

The HDL descriptions of the target systems' modules will be mapped on reconfigurable hardware blocks of appropriate type using systematic approaches, mapping tools and reconfigurable hardware generators that are commercially available (Xilinx EDK Framework). The flow is completed with the system integration and back end design phases.

During system integration two different evaluation procedures will take place. Initially, each of the developed computer vision algorithms will be validated in order to verify its proper functionality in respect to timing, performance, and throughput constraints posed by the specifications of SPARTAN project. Possible violations to these constraints will be alleviated by applying focused optimization techniques (e.g. through usage of pipeline/parallelization techniques). Then, we can guarantee that all the developed algorithms meet systems' specifications. Then, one more evaluation step takes place, where the entire system is evaluated against to the same criteria (proper functionality in respect to timing, performance, and throughput constraints). Potential violations in this task will be overcome through appropriately tuning of kernels where validations are identified.

A. PC-FPGA Interfacing

Regarding the interface between host PC and the FPGA device we define two communication stages, the configuration stage and the debugging and execution stage. Figure 5 summarizes the communication protocols for both stages.

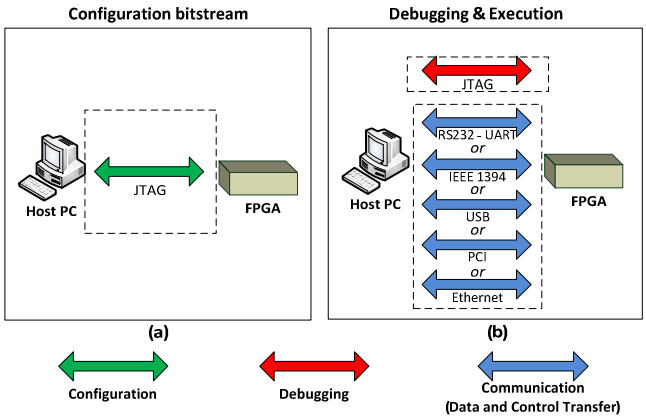

Figure 5. Communication protocols

The selected interface for FPGA to Central processing (Linux board) communications at the programming stage is the JTAG interface**.** FPGAs are JTAG-aware and so all the FPGA I/O pins can be controlled from the JTAG interface. FPGAs add the ability to be configured through JTAG (using proprietary JTAG commands). JTAG allows device programmer (hardware) to transfer data into internal non-volatile device memory. In the case of FPGAs, volatile memory devices can also be programmed via the JTAG port normally during development work. The Xilinx ISE tools can then be used to read-back the FPGA configuration for example or to even inspect internal signals, using "ChipScope Pro".

The SPARTAN system shall use extensively communication scenarios between the host PC and the FPGA during both the design and execution stage. Depending on amounts of data which should be transferred, different types of connections can be used. In order to determine those connections and provide better communication flexibility, we define the system architecture in terms of logic layers. The system layers, as shown in Figure 6, are: FPGA Hardware layer, FPGA Software Stack layer, Channel layer, Host PC Software Stack layer and Vision Algorithms Wrappers layer.

- Vision Algorithms Wrappers Layer: Contains the control section of computer vision algorithms of SPARTAN system written in C/C++. All requests to hardware are initiated by those programs.
- Host PC Software Stack Layer: It is responsible for translating the requests from Vision Algorithms Wrappers Layer to communication protocol oriented requests.
- Channel Layer: It is responsible for accessing hardware using selected physical interface.
- FPGA Software Stack Layer: This layer contains the appropriate libraries that support the communication protocol solution of Channel layer. These libraries may be simple RS232 software interfaces up to complete Linux kernel images containing i.e. the Ethernet software stack.
- FPGA Hardware Layer: Contains the architecture of vectorial processing. In this layer except of hardware logic describing computation intensive parts of vision algorithms, also embedded processors shall be utilized. Moreover external memories are included on that layer.

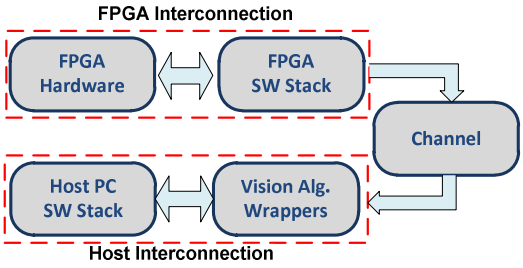

Figure 6. System layers

The communication between Vision Algorithms Wrapper Layer and Host PC Software Stack Layer is being established at the host operating system application layer. The vision algorithm wrappers are written in C/C++ and utilize pre-compiled libraries and/or API's of the host OS in order to transfer data to the Channel Layer.

In a similar way the communication between FPGA Hardware Layer and FPGA Software Stack Layer is being established exclusively on the FPGA device and depends on the characteristics of hardware architecture. There are two possible scenarios for the architecture that affect the aforementioned communication. These scenarios are depicted in Figure 7. The choice for the implemented architecture for the SPARTAN system will be based on the results of HW-SW partitioning of visual algorithms and the selection of the Channel protocol.

For the choice of the protocol for the channel, several parameters of the system architecture shall be taken into consideration. We extinguish the system communication needs in terms of required bandwidth. Typical solutions for this purpose are UART, PCI, USB, Firewire, and Ethernet interfaces. Based on SPARTAN specifications, the most candidates among them is the Ethernet and PCI for high

bandwidth connection links and RS232 for low-bandwidth links, respectively.

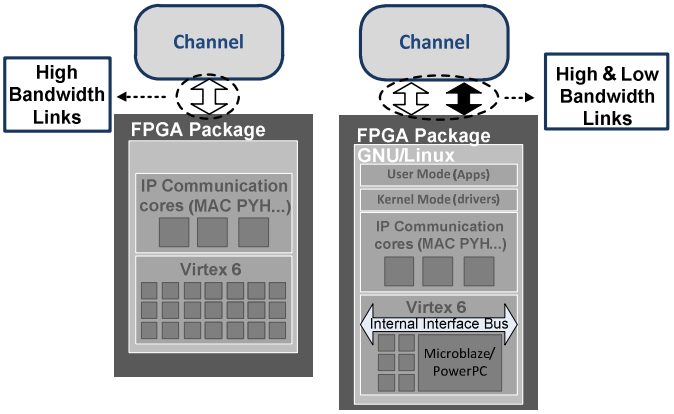

Figure 7. FPGA Hardware Architecture. (a) Implementation using IP cores and logic (b) Implementation using Embedded processor

This choice, depicted in Figure 8, is based on the fact that those protocols are well supported by software libraries both in the host pc and the FPGA. In addition to that, these protocols are well documented by Xilinx tool-flow, which will be used in SPARTAN project.

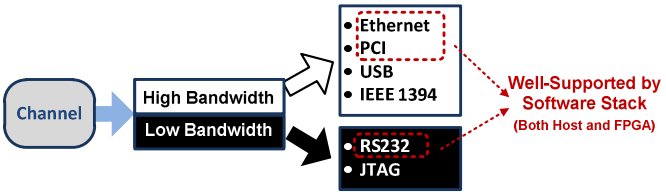

Figure 8. Selected Communication Protocols for SPARTAN system

However the JTAG protocol, except for bit-stream configuration on FPGA, shall be used in debugging time. In general the JTAG offers the following uses:

- Debugging: JTAG is used for accessing sub-blocks of integrated circuits.
- Boundary Scan Testing: JTAG manipulates the scan chain external interface (inputs and outputs to other chips) in order to test for certain faults. By using JTAG to manipulate its internal interface (to on-chip registers), the combinational logic can be tested.
- Debugging at run-time (during system integration step): By programming and read-back the configuration bit-stream, it is possible to find potential errors/upsets occurred at run-time. Due to the inherent constraints for fault masking posed by the target application domain (space application), such a mechanism can provide sufficient fault detection and correction (through partial and/or dynamic reconfiguration). Note that debugging at run-time affects only the development step of the prototype in order to ensure that the entire system works properly.

V. CONCLUSION

This paper discusses the computer vision algorithms that will be implemented during the SPARTAN project. The implementation medium for these algorithms is a Virtex-6 FPGA device.

ACKNOWLEDGEMENT

This work is supported by ESA (European Space Agency) project "Sparing Robotics Technologies For Autonomous Navigation (SPARTAN)" (ESA/ESTEC ITT Reference AO/1-6512/10/NL/EK).

REFERENCES

[1] S. Xydis, et.al., "Custom multi-threaded Dynamic Memory Management for Multiprocessor System-on-Chip platforms", Int. Conf. on Embedded Computer Systems (SAMOS), pp.102-109, 19-22 July 2010.

[2] ESA Compatible VHDL, (electronic document available at http://www.esa.int/TEC/Microelectronics/SEMS7EV681F_0.html).

[3] The ESA-NASA ExoMars programme 2016-2018.

[4] M. Quigley, et.al., "ROS: an open-source Robot Operating System", International Conference on Robotics and Automation, 2009

[5] K. Konolige, "Small vision systems: hardware and implementation," International Symposium on Robotics Research, pp. 111–116, 1997

[6] K.-J. Yoon and I. S. Kweon, "Adaptive support-weight approach for correspondence search," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, No. 4, pp. 650–656, 2006.

[7] L. Di Stefano, et.al., "A fast area-based stereo matching algorithm", Image and Vision Computing, Vol. 22, No. 12, pp. 983–1005, 2004.

[8] K. Muhlmann, et.al., "Calculating dense disparity maps from color stereo images, an efficient implementation," International Journal of Computer Vision, Vol. 47, No. 1-3, pp. 79–88, 2002.

[9] L. Nalpantidis, et.al., "A dense stereo correspondence algorithm for hardware implementation with enhanced disparity selection", in Hellenic conference on Artificial Intelligence, pp. 365–370, 2008.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. Journal of Computer Vision, Vol. 60, No. 2, pp. 91–110, 2004.

[11] H. Bay, A. Ess, et.al., "Speeded-up robust features (SURF)", Computer Vision and Image Understanding, Vol. 110, pp. 346–359, June 2008.

[12] C. Harris and M. Stephens, "A combined corner and edge detector", Int. Conf. Alvey, Vol. 15, pp. 50, 1988.

[13] C. Olson, et.al., , "Rover navigation usingstereo ego-motion," Robotics and Autonomous System, Vol. 4, No. 43, pp. 215–229, 2003.

[14] D. Nister, et.al., "Visual odometry for ground vehicle applications," Journal of Field Robotics,Vvol. 23, No. 1, pp. 3–20, 2006.

[15] F. Souvannavong, et.al., "Visual Odometry for autonomous localization on Mars", *ASTRA conference, 2008*

[16] L. Nalpantidis, et.al., "Non-probabilistic cellular automata-enhanced stereo vision simultaneous localisation and mapping (SLAM)," *Measurement and Science Technology*, in press.

[17] F. Souvannavong , et.al., Vision-Based Motion Estimation for the EXOMARS Rover, CNES (The French Space Agency), France.

[18] M. Di-Marco, et.al., "Set membership localization and mapping for autonomous navigation, Int. Journal on Robust Nonlinear Control, Vol. 11, No 7, pp.709–734.

[19] M. Di-Marco, et.al., "Simultaneous localization and map building for a team of cooperating robots: a set membership approach", Journal on Robotics Automation. vol. 19, no. 2, pp. 238–249.

[20] Virtex-6 FPGA Family, http://www.xilinx.com/products/silicon-devices/fpga/virtex-6/index.htm