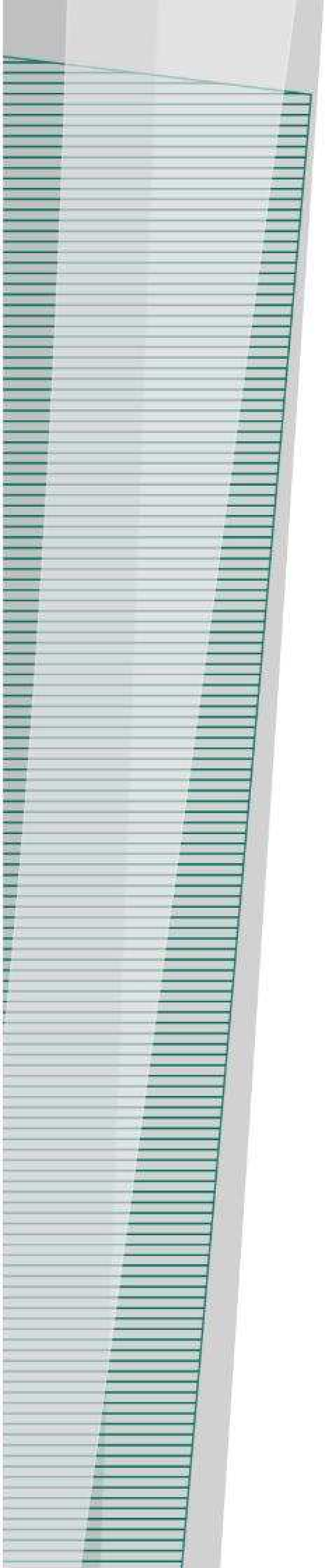

LNEC LABORATÓRIO NACIONAL DE ENGENHARIA CIVIL

INCD Infraestrutura Nacional de Computação Distribuída

Relatório de resultado – Piloto Experimentação Cloud Computing

(E7 – Relatório do estudo de viabilidade relativo à alínea (c) do nr. 1 da cláusula 3º do protocolo:

EVALUATION OF CIVIL ENGINEERING USE CASES IN A PUBLIC CLOUD COMPUTING FRAMEWORK- Assessment of GPU computational resources in images processing tests)

EXT/2016

15-November-2016

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

FCCN Computação Científica Nacional

Authors:
Alberto Azevedo
João Rogeiro
Anabela Oliveira
João Rico
António Inês
José Barateiro

Revised by:
André Fortunato

OUTLINE

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

We present herein the implementation, test and performance comparison of GPUs in physical and cloud/virtual environment of: i) an image processing algorithm for coastal applications using Synthetic Aperture Radar (SAR) imagery; ii) a hybrid programming algorithm (CPU/GPU) to automatically compute the correspondences between similar images. The first test is based on an algorithm developed by LNEC in OpenCL and the respective Python wrapper. The second test uses the photogrammetry software – MicMac – which can be compiled to run on both CPUs and GPUs.
Results show that GPUs as well as hybrid GPU/CPU approaches are an attractive alternative to CPUs for image processing codes. For the GPU SAR image processing test, the gain in execution time is over one order of magnitude relative to a single CPU processor. For the hybrid CPU/GPU tests, the gains are slightly smaller. In a virtual environment, the overhead is negligible for the GPU test as each virtual machine has a dedicated GPU and the majority of the processing is done at the GPU level. For the hybrid GPU/CPU tests, the exact same hardware was tested on the physical and the virtual environment. Results show that running times were on the order of 4 to 10% slower in the virtual environment. This overhead is, for most scientific computing applications, irrelevant, specially, if one considers the advantages from the virtual environment, such as the flexibility, scalability, cost and portability.
As the present analysis covers several image processing programs, the usefulness of GPUs for this field of application is clearly demonstrated and should be further explored in the future for more demanding applications such as the early detection of pollution events. Within the several applications in civil engineering, numerical models solving partial differential equations remain however as one of the most computational challenging tasks. The present analysis should thus be further extended in the future through the adaptation and application of some of these models in a GPU environment to assess its usefulness for engineering purposes.

Keywords: Image processing, GPU, hybrid GPU/CPU, cloud computing

1 INTRODUCTION

In the scope of the Portuguese National Distributed Computing Infrastructure (INCD, www.incd.pt), FCT/FCCN has established a protocol with LIP and LNEC to jointly promote an experimental pilot of cloud computing. LNEC's tasks in this pilot are the following:

- Test and evaluate the performance of several civil engineering applications in the pilot cloud developed by LIP and FCT/FCCN;
- Compare the tests performed in the previous task with commercial cloud infrastructures;
- Test the feasibility of using GPUs, also in a cloud computing approach, to accelerate remote sensing detection algorithms and numerical modeling programs through the implementation of hybrid codes (CPU+GPU).

This report presents the outcomes of the third task. LNEC has implemented, tested and compared the performance of GPUs in physical and cloud/virtual environment of: i) an image processing algorithm for coastal applications using Synthetic Aperture Radar (SAR) imagery; ii) hybrid programming algorithm (CPU/GPU) to automatically compute the correspondences between similar images.

The report is organized as follows, besides this Introduction. In Chapter 2, a description of the algorithms is made as well as the type of images and data used to perform the tests. The description of infrastructures used for the performance analysis is presented in Chapter 3. Chapter 4 presents the performance results of all tests in the physical environment of the CPU and GPU hardware. In this chapter a comparison between CPU and GPU performance is also made. In Chapter 5, all the benchmark tests executed in Chapter 4 are repeated, but now in a cloud/virtual environment, in order to assess the viability of the usage of the cloud-computing paradigm. Chapter 6 is dedicated to the presentation of the major conclusions of the present task in the scope of the Experimental Pilot Cloud Computing project.

2 USE CASES DESCRIPTION

2.1 OVERVIEW OF THE TEST CASES

SAR image processing for coastal applications

The Estuary and Coastal Zones Division of LNEC has been developing for the last 6 years several image processing algorithms for oil spill detection (Azevedo, 2010), coastline extraction and bathymetry assessment based on Synthetic Aperture Radar and Optical imagery (ongoing projects). The satellite images have become one of the best methods to extract useful information for coastal studies, taking advantage of the cost/benefit of this type of methodology, when compared with the more expensive traditional methods based on field campaigns. The satellite images can cover large areas, without the cloud and sunlight constraints (in the case of SAR images), with ground resolutions of 0.4-0.75m m and 1.25 m for Optical (from different satellites) and TerraSAR-X images, respectively.
In this study, a simple image-processing algorithm was developed by LNEC's team in order to test the performance of GPU and compare it with the CPU version. The algorithm performs the image reading and the application of several filters to a TerraSAR-X image with 12536 x 5647 pixels and a ground resolution of 3 m (Figure 1). The original image size was 25072 x 11293 pixels and 1.25 m resolution, but due to memory RAM limitations a smaller version was used in the performance tests.

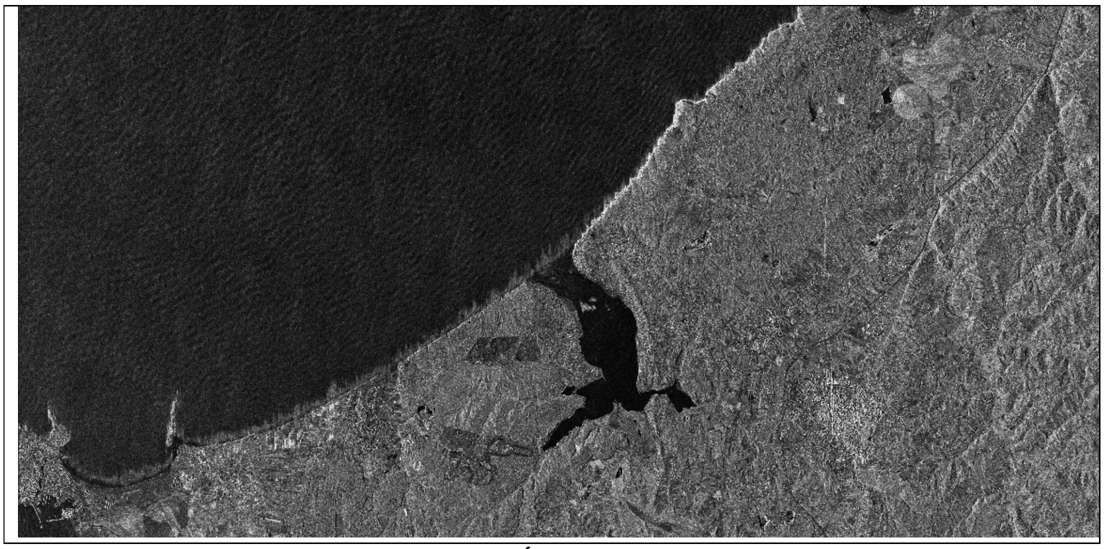
Figure 1 – TerraSAR-X image of the Óbidos lagoon, used in the present study.

MicMac Multi-image correspondence

As part of LNEC's mission to undertake, coordinate and promote scientific research and technological development, aiming to the continuous improvement and the good practice of Civil Engineering, it is often necessary to evaluate new technologies, their interest and applicability. Such was the case of multi-image correspondence technique using photographic images and orthomosaics. This study was carried out by Henriques (2015), and the technique is useful in general when there is a need to detect changes in an object or structure, compare similar objects or structures, or generate 3D models from 2D photographs. Example applications include the modelling of rocks and natural joints, safety inspections of concrete dam walls, and monitoring of piers or pier models (using short distance photographs or aerial photographs, respectively).

During the study of the site of large structures, such as concrete dams, a characterization of the mechanical properties of the rocks and rock masses is carried out. To that end, tests are carried out to obtain values that permit the characterization of the deformability and tension of rocks and joints. These can include the determination of the roughness of the surfaces, which can occur before and after the resistance tests. During the test one evaluates the effect of normal and tangential loads in samples, namely through the analysis of the change in roughness of the surfaces.

One usual method to estimate this change is based on a digital surface model (DSM) developed from the automatic survey in a three-dimensional scanning table. In this study case, a different method using photographs from different angles and multi-image correspondence processing using the MicMac software (described in section 3.2) is done and both methods are compared.

The sample used in this case study, referred to as 'castanha', is the surface of a joint, which is a granite parallelepiped with a 12×8 cm^2 top surface, embedded in a concrete block. Figure 2 shows one of the 9 pictures taken for the test, and used to create the orthomosaic and the point cloud (Figure 3). The second use case, 'gravillon', is very similar in approach to the 'castanha' case, although it uses a different subset of MicMac subprograms, as described in section 3.2.

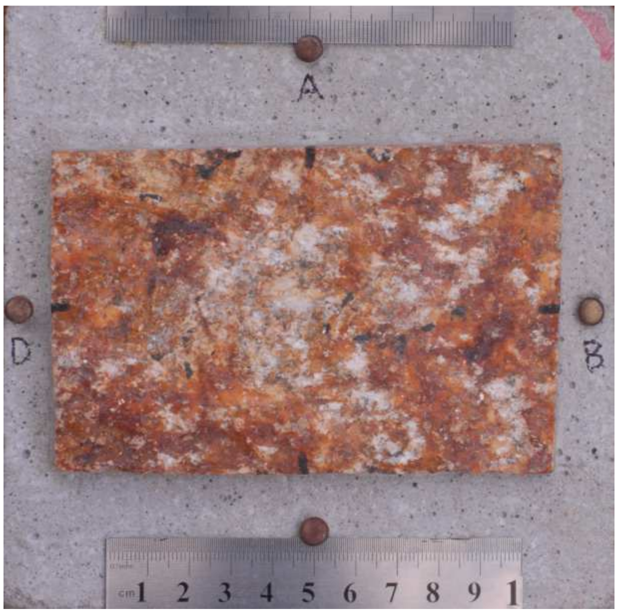

Figure 2 – Vertical photograph of one sample (12cmx8cm) of a rock embedded in a concrete block. The sample is used to characterize the mechanical properties of rocks and rocky massifs

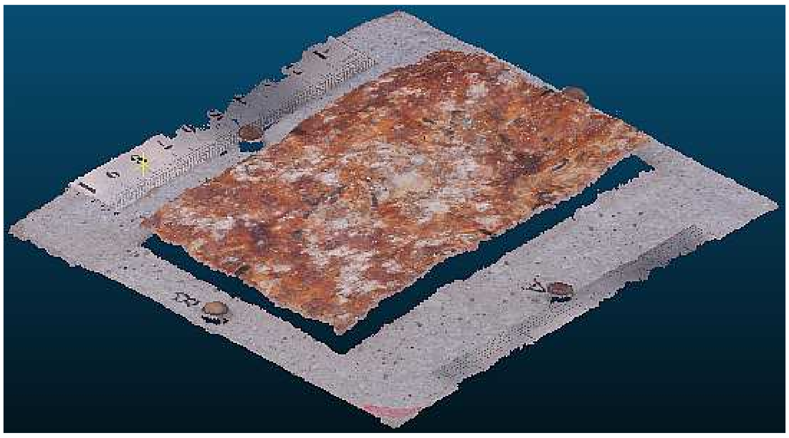

Figure 3 – Point cloud of the sample, generated with the MicMac program from 9 photographs from different angles

2.2 TECHNICAL DESCRIPTION OF THE USE CASES

OpenCL algorithm applied to a SAR image

The algorithm developed by LNEC's team was coded in OpenCL (Open Computing Language, Version 2.0 - https://www.khronos.org/opencl) and the respective Python wrapper (PyOpenCL - https://pypi.python.org/pypi/pyopencl). OpenCL is a royalty-free standard for cross-platform, parallel programming of modern processors found in personal computers, servers and handheld/embedded devices. This framework greatly improves speed and responsiveness of applications since it can dynamically interrogate the system load and balance work across available processors (CPUs, GPUs, DSPs, FPGA and hardware).

In the present project the OpenCL framework is used to manage the workload to the GPU. The CPU version of the image-processing algorithm was developed using the Python-Pillow library (Pillow - https://github.com/python-pillow/Pillow), which is a fork of the former Python Imaging Library (PIL - http://www.pythonware.com/products/pil/). The Python-Pillow Library offers extensive image-processing capabilities to the standard Python interpreter.

The pseudo-code of the algorithm for both versions (CPU and GPU) is presented below. The code was developed in Python with the exception of the KERNEL file that was developed in C (Image_filters.cl).

OpenCL pseudo-algorithm:

1. Import the modules pyopencl and PIL
2. Read image
3. Convert image to RGBA
4. Create an OpenCL context on available platform
5. Create a command-queue on the GPU device on the created context
6. Load input image from file and load it into an OpenCL image object
7. Create output1 and output2 image objects in order to create filtering workflow inside KERNEL
8. Create sampler for sampling image object
9. Create/load OpenCL program (KERNEL with filters and saved in "Image_filters.cl" file)
10. Define Global and Local work size buffers
11. Call the several filters from kernel (smooth_strong, detail, smooth and blur filters). The user defines the number of times the last filter (blur filter) is called (from 1 up to 200 times)
12. Transfer output image from GPU memory to global memory
13. Save final image to the output file

The pseudo-code for the serial version of the algorithm is presented in the following lines:

1. Import the modules pyopencl and PIL
2. Read image
3. Convert image to RGBA
4. Apply the smooth_strong filter (5x5)
5. Apply the detail filter (3x3)
6. Apply the smooth filter (3x3)
7. Apply the blur filter (5x5) inside a for-loop (with a length define by the user, from 1 up to 200 times)
8. Save final image to the output file

MicMac multi-image correspondence

MicMac (http://logiciel.ign.fr/?Micmac) is a photogrammetry software developed by the French *Institut National de l'Information Géographique et Forestière* (IGN) and designed to automatically compute the correspondences between similar images. It is written in C++ and provides solutions (of equivalent quality of ad hoc methods) to the following types of calculations: numerical models of elevation from high-resolution aerial images, numerical models of terrains from satellite images, connection points in aero-triangulation, and homological points for the superposition of multi-channel images.
MicMac has a multi-resolution approach and, at a given resolution, seeks to minimize an energy function (related to the coordinates and the regularity of the images). It is essentially a pipeline of several subprograms, which the user can control by a parameterization system through a xml file.
MicMac programs can be compiled to run on both CPU (including usage of OpenMP - http://openmp.org) and GPU. It additionally uses a parallel computing approach by generating a makefile that is executed in parallel depending on the available cores ('-j' option). A typical MicMac program may use other ancillary programs such as ImageMagick - http://www.imagemagick.org, exiv2 - http://www.exiv2.org and proj4 - https://github.com/OSGeo/proj.4, which are integrated into the MicMac workflow.
The 'castanha' case study uses all of the following 7 subprograms, while the second case study, 'gravillon', uses only the last 2 (Malt and Nuage2Ply) since its input is already pre-processed:

1. Tapioca (full automatic tie points computation)
2. Tapas (full automatic orientation computation)
3. AperiCloud (generation of visualization of camera position and sparse 3d model)
4. SBGlobaBascule ("score based global" bascule, used when no absolute information is available)
5. Campari (compensation of heterogeneous measures)
6. Malt (matching in ground geometry and ground-image geometry)
7. Nuage2Ply (transform depth map in cloud point into ply format (Polygon File Format))

3 DESCRIPTION OF THE INFRASTRUCTURES USED FOR THE PERFORMANCE ANALYSIS

3.1 BASELINE SYSTEMS FOR COMPARISON

Two distinct sets of resources and use cases at LNEC were used for the present analysis. The first set was developed using the GPU resources made available by FCCN for the project and the resources of the Hydraulics and Environment Department team, while the second test used only the GPU resources made available by FCCN. This approach allowed for two distinct environments to run the GPU performance/viability tests. In the present section a brief description of the hardware for both environments is made.

DHA/LNEC computational environment

The two different environments used by the Hydraulics and Environment Department team for the image processing algorithm test are (details are available in Table 1.):

i) a desktop setup with a dual core intel i5-2300 CPU @ 2,8GHz with a AMD Radeon HD 6450 GPU; and

ii) a 12 core Intel Xeon E5-2620 v2 @ 2.10GHz with a NVIDIA Tesla K20m GPU, provided by LIP.

3.2 PILOT CLOUD COMPUTING RESOURCES

The present pilot cloud is composed of the following physical resources:

- blades HP ProLiant BL460c with two quad core Xeon E5440 processors

Table 1 - Computational Resources Description CPU

host / resource	**setup**	**model**	**cores (total)**	**speed (turbo)**	***l2 size***	***l3 size***	***bus***	**ram size**	***ram type***
DHA- Desktop	1 × CPU	Intel Core i5-2300	4 (4)	2.8Ghz (3.1)	4 × 256KB	6 MB	-	4 GB	2 ch DDR
FCCN - CPU	2 × CPU	Intel Xeon E5-2620	6 (12)	2 Ghz (2.5)	6 × 256 KB	15 MB	2 × 7.2 GT/s QPI	32 GB	4ch DDR3

Table 2 - Computational Resources Description GPU

host / resource	**model**	**engine** (single/double precision)	**memory**	***power***
DHA- GPU	AMD Radeon HD 6450	160C @625Mhz = 185/n.a. Gflops	1GB DDR3 @ 600Mhz = 9.6GB/s	15W
LIP – GPU aurora	NVIDIA Tesla K20m	2494C @706Mhz = 3.52/1.17 TFlops	5GB GDDR5 @ 5.2Ghz = 208GB/s	225W
LIP – GPU nimbus	NVIDIA Tesla K40	2880C @745 (875)Mhz = 4.29(5.04)/1.43(1.68) TFlops	12GB GDDR5 @ 6Ghz = 288GB/s	235W

4 EVALUATION ANALYSIS

4.1 DESCRIPTION OF EVALUATION PROCEDURE

In all cases the execution time is the metric used to evaluate the benchmarking performance. Therefore, only the global CPU and GPU performance were evaluated.

4.2 EVALUATION OF PERFORMANCE IN THE PHYSICAL ENVIRONMENT

OpenCL algorithm applied to a SAR image

The first tests were run in a physical environment, where the user had full access to the machine devices. Figure 4 and Figure 5 show the performance times of the algorithm, for the serial and GPU version in each host (DHA and FCCN).

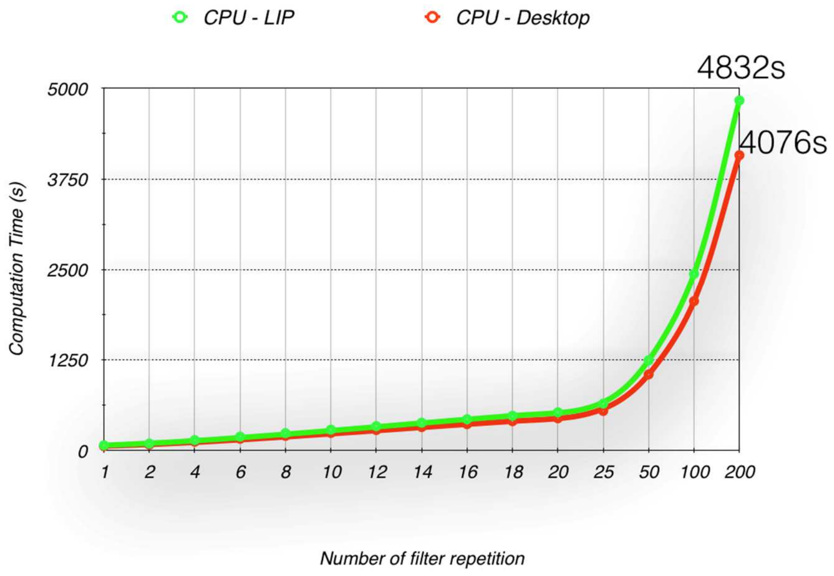

Figure 4 – Results of the serial/CPU version of the image-processing algorithm.

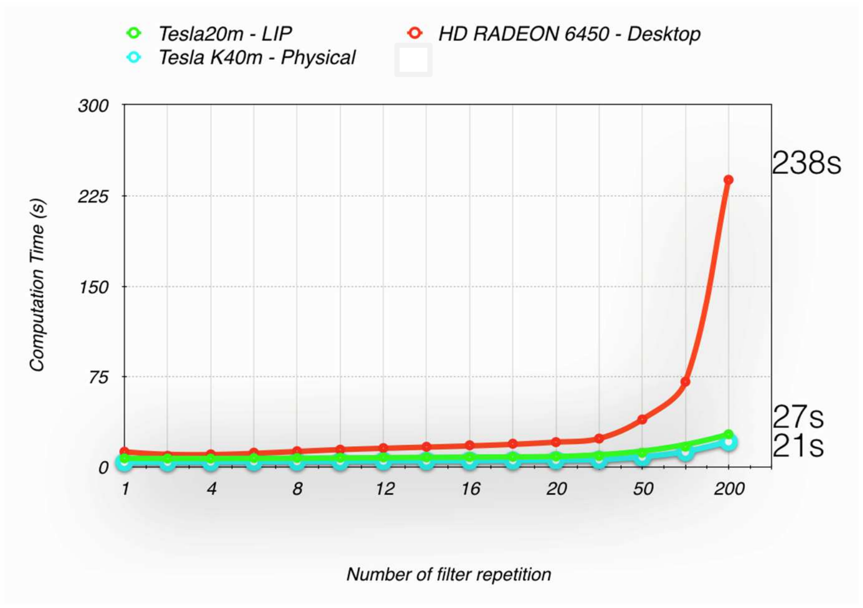

Figure 5 – Results of the OpenCL (GPU) version of the image-processing algorithm in a physical environment.

The results show that the execution time differences, in the CPU devices, are very small and result from the distinct CPU clock speeds. In this test, each execution of the filters runs for a period of approximately 20-24 seconds. Therefore, the computational time grows linearly with the number of repetitions of the last filter.
The OpenCL version of algorithm takes advantage of the GPU capabilities and reduces the computational time to 21s, 27s in the Tesla GPUs, K40m and K20m, respectively, and 238s for the AMD RADEON HD6450 GPU, considering the most demanding test, with 200 repetitions of the blur filter.

MicMac multi-image correspondence

The results for the physical environment for the case studies 'castanha' and 'gravillon' are presented on Table 3 and Table 4, respectively.
Tapioca and Malt are the two subprograms which the program spends the most time on and the only two which are parallel, as can be confirmed from the differences between real and user times, and between different numbers of cores. Tapioca is parallelized at the makefile level as described in section 3.2, while Malt is parallelized using openMP or CUDA. The running time of Tapioca decreases as the number of cores increases and it is essentially the same for the OMP-8 and CUDA runs since both use 8 cores. The running time of Malt also decreases as the number of cores increases, and is lowest for the CUDA run. The running times for the Tapioca and Malt runs are plotted in Figure 6 for the 'castanha' case (the values shown in this figure were normalized so that Malt's OMP-4 run has the same value as Tapioca OMP-4's run).

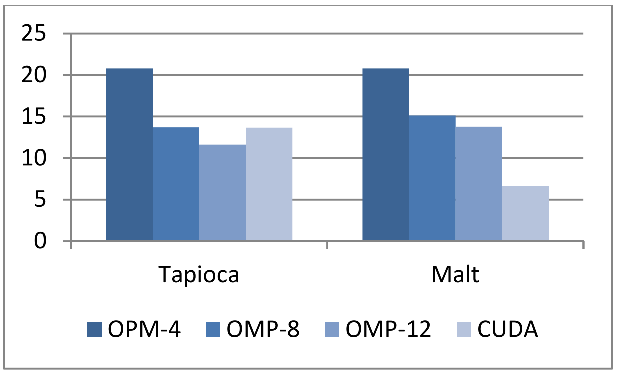

Figure 6 – Normalized running times for the Tapioca and Malt subprograms for the 'castanha' case study.

Table 3 - Running times for the different MicMac subprograms for the 'castanha' case. OMP-n: run with openMP option with n processor cores; CUDA: run with CUDA option and 24 cores.

	OMP-4			OMP-8			OMP-12			CUDA		
time (sec)	real	user	sys	real	user	sys	real	user	sys	real	user	sys
Tapioc a	20.81	88.05	6.40	13.67	91.16	6.96	11.63	94.37	7.40	13.64	90.87	6.91
Tapas	11.14	10.27	0.11	10.94	10.15	0.12	11.12	10.37	0.11	10.75	10.03	0.12
Aperi-Cloud	11.44	11.40	3.49	12.40	11.38	3.25	11.96	10.61	3.14	11.92	10.41	3.20
SBGloB ascule	1.04	0.90	0.05	1.01	0.89	0.06	1.03	0.90	0.06	1.02	0.90	0.07
Campa ri	2.07	1.86	0.06	2.07	1.83	0.07	2.08	1.86	0.07	2.06	1.85	0.06
Malt	399.7 9	1314. 7	14.21	291.1 7	1353.02	15.7 8	265.3 3	1393.69	18.8 5	127.0 0	646.0 3	102.4 8
Nuage 2Ply	20.47	19.81	0.16	20.69	20.02	0.16	20.50	19.83	0.16	22.62	21.94	0.17

Table 4 - Running times for the different MicMac subprograms for the 'gravillon' case

	OMP			CUDA		
time (seconds)	real	user	sys	real	user	sys
Malt	29.86	109.66	8.12	21.18	19.60	7.55
Nuage2Ply	1.87	1.75	0.05	2.16	2.05	0.05

4.3 EVALUATION OF PERFORMANCE IN CLOUD/VIRTUAL ENVIRONMENT

In the present section both tests, SAR image-processing and MicMac multi-image correspondence, are repeated in a cloud/virtual environment in order to evaluate the performance differences between virtual and physical environments.

OpenCL algorithm applied to a SAR image

In the present section all the tests regarding the processing of a SAR image are repeated in a virtual/cloud environment, using a NVIDIA Tesla K40m. Figure 7 shows the performance times of the GPU version of the algorithm, and the differences between virtual and physical performances.

The results show that the performance differences between virtual and physical environments are negligible (virtual k40m is approximately 1% faster than in physical configuration). The difference between k20m and k40m are mainly due to the hardware differences between both GPUs.

The present test also demonstrates that the virtualization of GPU hardware doesn't have a significant impact on the performance of the algorithms, because each virtual/cloud configuration has its own dedicated GPU. The GPUs cannot be shared between different cloud systems therefore they can be used in cloud environment maintaining their physical performance.

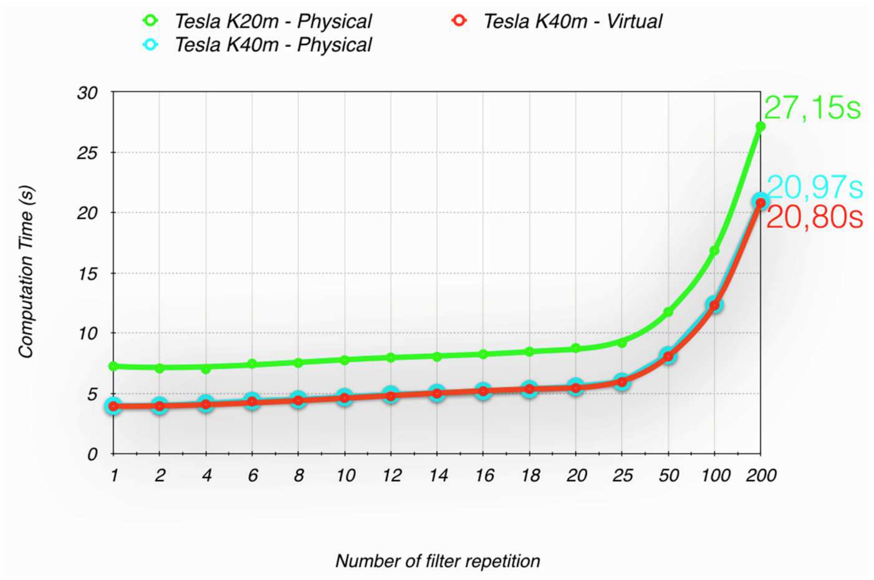

Figure 7 – Results of the OpenCL (GPU) version of the image-processing algorithm in physical and virtual environments for the NVIDIA GPUs (Tesla K20m and K40m).

MicMac multi-image correspondence

Virtual environment

Table 5 and Table 6 show the absolute running times for the 'castanha' and the 'gravillon' cases. The relative and absolute running times of the virtual environment were in general very similar, with an overhead of circa 3-10%. Again in this case, the running times of the parallel sub-programs, Tapioca and Malt, decrease when a larger number of cores is used.
In every case, the virtual environment completely satisfied the criteria of functional completeness and functional correctness[1]. The programs were run in the same manner as in the local environment and the results produced were the same.

Table 5 - Running times for the different MicMac subprograms for the 'castanha' case. OMP-n: run with openMP option with n processor cores; CUDA: run with CUDA option and 24 cores.

	OMP-4			OMP-8			OMP-12			CUDA		
time (seconds)	real	user	sys	real	user	sys	real	user	sys	real	user	sys
Tapioca	24.44	91.86	8.14	18.29	95.49	8.92	15.75	99.65	9.49	15.77	92.68	5.56
Tapas	12.04	11.00	0.14	11.67	10.84	0.15	12.11	11.19	0.14	11.99	10.83	0.10
Aperi-Cloud	12.23	10.52	4.73	12.64	10.99	4.72	12.21	10.83	4.68	12.30	10.89	4.14
SBGloBascule	1.25	1.05	0.09	1.23	1.04	0.08	1.20	1.01	0.09	1.22	1.05	0.05
Campari	2.37	2.07	0.08	2.32	2.07	0.09	2.35	2.07	0.10	2.39	2.11	0.04
Malt	411.24	1340.26	15.43	303.53	1387.24	18.38	270.28	1418.71	21.09	131.72	597.38	103.35
Nuage2Ply	21.35	20.67	0.26	21.45	20.77	0.23	21.87	21.09	0.23	24.04	23.32	0.20

Table 6 - Running times for the different MicMac subprograms for the 'gravillon' case

	OMP			CUDA		
time (seconds)	real	user	sys	real	user	sys
Malt	36.84	105.54	11.56	23.29	20.58	7.54
Nuage2Ply	2.18	1.90	0.08	2.28	2.14	0.05

[1] Functional completeness: degree to which the set of functions covers all the specified tasks and user objectives. Functional correctness: degree to which a product or system provides the correct results with the needed degree of precision.

Comparison between the physical and the virtual environments

The running times for the Malt program (the only sub-program that uses the GPU) for both test cases 'castanha' and 'gravillon' are shown in Table 7 (these values are shown in the previous 4 tables, but are repeated here for comparison). As expected, the running times in the virtual environment were larger, by a factor of circa 4-10%. This overhead is, for most scientific computing applications, irrelevant, specially, if one considers the advantages from the virtual environment, such as the flexibility, scalability, cost and portability. The user time in the 'castanha' case for the virtual environment was significantly faster (7.5%) than the physical case, which might be caused be several factors, including the fact that for the virtual environment a more recent version of CUDA (v7.5) was used, compared the physical environment (v6.5).

This test was carried out to assess the overhead of the virtual environment, which required using the exact same GPU. However in the vast majority of the cases one does not expect this scenario. Instead, one will most likely be facing (at least) two alternatives: the local (physical) setup and whichever virtual/cloud environment is being considered – likely, a different set of physical machines (e.g., different model and vendor). This difference in hardware is expected to have an impact in the running times of the same magnitude as the overhead of the virtual environment. In particular, when the GPU of the virtual environment is better than the one of the physical environment, the performance of the virtual environment will be better than the physical.

Table 7 - Running times for the Malt sub-program of the 'gravillon' and the 'castanha' cases, using CUDA and 24 cores.

	'gravillon'			'castanha'		
	real	user	sys	real	user	sys
physical	21,18	19,60	7,55	127,00	646,03	102,48
virtual	23,29	20,58	7,54	131,72	597,38	103,35
increase (%)	10,0	5,0	-0,1	3,7	-7,5	0,9

5 CONCLUSIONS

We presented herein the implementation, test and performance comparison of GPUs in physical and cloud/virtual environments of: i) an image processing algorithm for coastal applications using Synthetic Aperture Radar (SAR) imagery; ii) a hybrid programming algorithm (CPU/GPU) to automatically compute the correspondences between similar images.

Results show that GPUs as well as hybrid GPU/CPU approaches are an attractive alternative to CPUs for image processing codes. For the GPU SAR image processing test, the gain in execution time is over one order of magnitude relative to a single CPU processor. For the hybrid CPU/GPU tests, the gains are slightly smaller. In a virtual environment, the overhead is negligible for the GPU test as each virtual machine has a dedicated GPU and the majority of the processing is done at the GPU level. For the hybrid GPU/CPU tests, the exact same hardware was tested on the physical and the virtual environments. It was found that running times were on the order of 4 to 10% slower in the virtual environment. This overhead is, for most scientific computing applications, irrelevant, specially, if one considers the advantages from the virtual environment, such as the flexibility, scalability, cost and portability.

As the present analysis covers several image processing programs, the usefulness of GPUs for this field of application is clearly demonstrated and should be further explored in the future for more demanding applications such as the early detection of pollution events. Within the several applications in civil engineering, numerical models solving partial differential equations remain however as one of the most computationally challenging tasks. The present analysis should thus be further explored in the future through the adaptation and application of some of these models in a GPU environment to assess its usefulness for engineering purposes.

References

Azevedo, A., 2010. Sistema Integrado de Modelação para Apoio à Prevenção e Mitigação de Acidentes de Hidrocarbonetos em Estuários e Orla Costeira. Lisboa, Portugal: Faculdade de Ciências da Universidade de Lisboa, Ph.D. Thesis, 224 pp. http://hdl.handle.net/10451/2332.

Henriques, M. J., Braz N., Roque D. (2015). Nuvens de Pontos e Ortomosaicos. A sua Utilização num Laboratório de Engenharia Civil, VIII Conferência Nacional de Cartografia e Geodesia.