http://dx.doi.org/10.21611/qirt.2012.107

GPU-based infrared thermography for NDE of minefields

by F.Pardo, P.López and D. Cabello

Centro de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela (CITIUS)
University of Santiago de Compostela, Santiago de Compostela, Spain, *fernando.pardo@usc.es*

Abstract

Infrared thermography is an attractive technique for non-destructive evaluation processes and particularly for detecting shallowly buried mines. Its use consists of subjecting the area under inspection to a source of natural or artificial heating/cooling process and studying the soil's response by means of the analysis of its thermal evolution given by a temporal sequence of infrared images. To this aim an efficient solution of the heat equation is required. Over the years, different software solutions have been proposed, taking advantage of today's impressive computing power of parallel machines. In this work, we consider a hybrid software–hardware approach making use of a Graphics Processing Unit (GPU) acting as a heat equation solver with the goal of obtaining a portable system to be used during field experiments. The system has been successfully used for the non-destructive inspection of soils in mine detection applications based on infrared thermography techniques.

1. Introduction

The dynamics of the heat equation have been widely known to have a crucial role on many physical problems. One such case is that of the non-destructive inspection of materials based on Infrared Thermography (IRT) techniques. IRT is based on studying the material's response to a given stimulus by means of the analysis of its thermal evolution given by a temporal sequence of infrared images that can be mapped into temperature values. Therefore, temperature distributions on the surface different from those that characterize the material reveal the presence of surface and internal anomalies, albeit the precise range of in-depth resolution depends on the material. A well-known application of IRT is on the detection of buried plastic antipersonnel mines, which are virtually undetectable by traditional methods, such as metal detectors. Other techniques, such as Ground Penetrating Radar (GPR), [1] have been investigated, but they fail to detect shallowly buried objects due to the large reflections coming from the soil-air interface. Given that unlike other types of mines, plastic antipersonnel mines are usually either laid on the surface or buried within a maximum depth of 10-15cm, special attention has been paid in recent years to the use of IRT for mine detection, [2].
An efficient way of extracting information from IRT data regarding the presence of mines using a 3D thermal model of the soil based on the solution of the heat equation was presented in [3]. This approach makes an intensive use of the 3D thermal model that needs to be solved for different configurations, resulting in a high computational cost. This makes a pure software solution impractical given that the final goal is to obtain a portable system to be used during field experiments. An alternative solution is the hardware implementation of the Finite Difference (FD) representation of the thermal model. In [4] we presented such a hardware implementation of the thermal model, by means of an FPGA, achieving an speed up of 10 compared to a purely software solution. However, the main drawback of this implementation is the cost of the FPGA system and its limited performance due to its little amount of memory. The approach presented in this paper makes use of a GPU platform to implement a 3D FD solver of the heat equation.

2. Infrared thermography for non destructive evaluation

IRT sensors, which respond to electromagnetic radiation in a sensor-specific wavelength range, constitute an attractive NDE technique because they can be used from a considerable standoff distance to rapidly survey large areas while providing information on several properties. Different materials show a characteristic response to a given thermal stimulus over time, a property that can be successfully used for the identification of buried targets. In this section, we will present a general procedure for the inspection of soils based on the solution of the heat equation applied, but not limited, to the detection of plastic antipersonnel mines. The analysis of the computational cost will result in the proposal of a hardware accelerator of the heat equation solver in Section 3.

2.1. Thermal model of the soil

We consider a soil volume, Ω, subjected to a known thermal stimulus through the soil–air interface, Γ, where both the soil and the targets are modeled as isotropic objects. We also assume that the thermal diffusivity is constant and that the temporal variation of the moisture content and the mass transference during the time of analysis are negligible, which are fair assumptions as long as the duration of the experiment does not exceed a couple of hours and the depth of inspection is limited to 10–15 cm. A typical example is antipersonnel mine detection, given that mines are usually either laid on the surface or only shallowly buried. Moreover, empirical evidence demonstrates that it is possible to reduce the interval of interest to roughly 1 hour around sunrise or sunset, as it is at these times when is more clearly manifested, [5]. Under these assumptions, the overall process is described by the time-dependent single phase 3D heat equation:

$$\frac{\partial T(r,t)}{\partial t} - div\left(\alpha(r) grad\left(T(r,t)\right)\right) = 0, \quad \alpha = \frac{k}{\rho c_p} \tag{1}$$

where r = (x, y, z) with $r \in \Omega$, ρ [kg/m^3] is the density, c_p [J/kg K] is the specific heat, k [W/m K] is the thermal conductivity, α [m^2/s] is the thermal diffusivity, and T [K] is the distribution of temperatures inside the soil. To solve this equation, the initial and boundary conditions are required:

$$k\frac{\partial T(r,t)}{\partial n} = q_{net}(t) \qquad for\ \tau \times \Gamma \tag{2}$$

$$k\frac{\partial T(r,t)}{\partial n} = 0 \qquad for\ \tau \times \partial\Omega \setminus \Gamma \tag{3}$$

$$T(x,y,z \to \infty) = T_\infty \tag{4}$$

$$T(r,t=t_0) = T_0(r) \qquad in\ \Omega \tag{5}$$

where τ is the time interval of analysis, q_{net} is the net heat that flows through the soil–air interface Γ, and n is the normal to the surface under consideration ∂Ω. Eq. 2 gives the boundary condition at the air– soil interface; Eq. 3 shows the boundary conditions applied to the sides of the volume not accessible for measurements, imposing a vanishing heat flux across them. Eq. 4 is the deep-ground condition, and it establishes that the temperature at a large enough depth remains constant. Finally, Eq. 5 gives the initial condition. The net heat flux at the the soil–air interface, q_{net}, in Eq. 2 can be written as:

$$q_{net}(t) = q_{sun}(t) + q_{rad}(t) + q_{conv}(t) \tag{6}$$

where q_{sun} is the short-wave radiation emitted by the sun and absorbed by the soil, q_{conv} represents the convection term at the soil–air interface, and q_{rad} is the heat flux exchange due to radiation. The first term of this expression, q_{sun}, can be estimated in a straightforward manner knowing the ground albedo and the local sun irradiation function [5]. In experimental setups, however, it can be easily measured with the help of low-cost equipment, which significantly reduces the complexity of the problem. The second term is given by $q_{rad}(t) = q_{sky}(t) - q_{soil}(t)$. The term $q_{sky}(t)$ is the long wave radiation from the atmosphere given by Stefan's law, $q_{sky}(t) = \sigma\, \varepsilon\, T_{sky}^4$ (t), where σ [W/m^{2}K^4] is the Stephen-Boltzman constant, ε is the emissivity, and T_{sky} [K] is the effective sky radiance temperature that can be estimated as $T_{sky}(t) = 0.9 \cdot T_{air}(t)$, T_{air} being the air temperature, [6]. On the other hand, the heat loss due to ground radiation is given by $q_{soil}(t) = \sigma\ \varepsilon\ T_{soil}^4$, where ε is the mean emissivity of the surface and $T_{soil}(t) = T(x,y,0,t)$ is the soil temperature at the soil-air interface. This boundary condition involves a non-linear function of *T*. Nevertheless, in the range of interest, this term can be linearized, [7].
To obtain a numerical solution of Eq. 1, we used an explicit FD method despite its conditional stability due to the need to accommodate changing boundary conditions during the simulation, which prevents the use of large temporal discretization steps typical of implicit or ADI methods [8]. Applying this discretization scheme to Eq. 1 and considering the neighborhood shown in Figure 1, we obtain the following equations for a surface and an internal node, respectively, assuming, without loss of generality, uniform spatial discretization steps, Δx = Δy = Δz:

$$T_{i,j,0}^{m+1} = T_{i,j,0}^{m} + F_0\left[\sum_{neigh}\left(T_{neighb}^{m} - T_{i,j,0}^{m}\right) + 2\left(T_{i,j,1}^{m} - T_{i,j,0}^{m}\right)\right] + 2\alpha_{sun}F_0 S q_{sun}^{m} + \left(2F_0 H + 8F_0 R T_{air}^3\right)\left(T_{air} - T_{i,j,0}^{m}\right)\ \forall i,j \tag{7}$$

$$T_{i,j,k}^{m+1} = T_{i,j,k}^{m} + F_0 \sum_{neighbor}\left(T_{neighbor}^{m} - T_{i,j,k}^{m}\right), \forall i,j,k > 0 \tag{8}$$

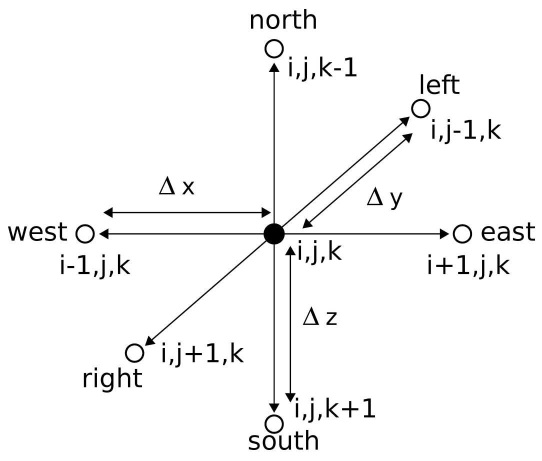


Fig. 1. *Node's neighborhood definition.*

where i, j, k are the discretization indexes in the x, y and z directions, m is the time discretization index, R, S, F_0 and H are defined as:

$$R = \frac{\sigma\varepsilon}{k}\Delta x \qquad S = \frac{\Delta x}{k} \qquad H = hS \qquad F_0 = \frac{\alpha \Delta t}{(\Delta x)^2} \tag{9}$$

The numerical stability criteria for Eqs. (7)-(8) is fullfilled if:

$$F_0 \le \frac{1}{6 + 2H + 8RT_{air}^3} \tag{10}$$

however in typical experimental setups $2H + 8RT_{air}^3 << 6$ and Eq. (10) can be approximated as:

$$F_0 \le \frac{1}{6} \tag{11}$$

This stability criteria is a trade-off between the spatial and temporal discretization steps.

2.2. Infrared thermography based soil inspection for the detection of buried objects

Based on the thermal model presented in Section 2.1, we will outline an IRT procedure for the identification of shallowly buried objects on the soil particularized for antipersonnel mine detection. The process starts with the acquisition of a sequence of IR images of the surface of the soil under known heating and atmospheric conditions, and goes on to determine error and classification maps with the location of buried targets. Usually, for a time range of 1 hour, a sequence of four images is enough.

We will assume that a pre-processing stage is run in order to align the images and map grayscale colors to temperature values. At the beginning of detection process, the starting conditions of the problem are set, most importantly, the initial assumption of the absence of buried targets and the definition of the initial entry on the error map, being a predefined threshold error. The detection of potential targets on the soil is done by comparing the measured IR image at a certain time with the simulated temperature on the soil surface at the same time given by the solution of the thermal model under the assumption of absence of mines on the field [3]. In so doing, the surface positions where the behavior is different from that expected under the assumption of mine absence are detected and classified as potential targets. Then, a quasi-inverse procedure is run which classifies the detected targets into either the mine or unknown categories. For the mine category, the depth of burial is also estimated. The iterative procedure is based on evaluating the deviation between the IR data and the solution of the thermal model for different soil configurations. Our initial guess will be to assume that all the potential targets are mines and the possible depths of burial constitute a finite set defined through the discretization parameter z. These two assumptions imply a reduction of the search space, therefore, the quasi-inverse nature of the classification effort that will either confirm or reject them. Based on this information, partial and global error and classification maps are calculated, which, in short, classify a target as *mine* if the global error is below a certain value and otherwise as *unknown*. Finally, a full inverse problem procedure is run for the targets classified as *unknown*, [3,9]. In this case, the simplifications made in the previous step no longer apply and the thermal model must be run for multiple configurations, resulting in at least 100–150 computations of the thermal model for each possible depth of burial.

2.3. Computational cost of the detection algorithm

The algorithm described above is based on iterative procedures involving multiple solutions of the heat equation for different soil configurations. This constitutes a time consuming process not feasible for its use on the field as the computational complexity of the FD method, if $N = n_x \cdot n_y \cdot n_z$ is the total number of grid nodes, is O(N·IT), where IT is the number of iterations. As an example, we consider the analysis of a piece of soil (α_{soil} = 6.4 $\cdot 10^{-7}$ m^2/s) of moderate dimensions of 1m x 1m with a shallowly buried mine (α_{mine} = 2.64 $\cdot 10^{-7}$ m^2/s). Even if the depth resolution of IRT is barely 10-15 cm, the depth of analysis must beset to at least 40-50 cm in order to apply the boundary condition in Eq. (4). Using a uniform spatial discretization of $\Delta x = \Delta y = \Delta z$ = 0.8 cm and assuming a temporal discretization step of Δt = 6.25 s (F0 = 0.06), for a typical example the simulation of the behavior of the soil during one hour using C++ (optimized for speed using O2 flag from Microsoft Visual C++ compiler) on a Intel Core2Duo 2.8GHz takes 30 seconds if single precision arithmetic is used to represent the temperatures. Taking into account that the proposed inverse procedure requires the solution of the model for multiple soil configurations, the total computing time assuming that only 100 iterations are needed (a soft approach) will add up to 50 minutes. As this jeopardizes its use for field experiments we have developed a hardware implementation of a heat equation solver. In [4, 10] we presented an FPGA-based implementation of such a solver. However, the main drawback of an FPGA implementation is the requirement of the system in terms of memory. The FPGA has a little amount of distributed memory and the FPGA's logic blocks can also be configured to behave like memory, however this is an inefficient way of FPGA using. Some vendors offer cards where external memory and FPGA are integrated on the same board, allowing using the FPGA to deal with processing issues. However, these are expensive solutions. GPUs offers a structure which perfectly fits with the proposed problem and they have the advantage of being cheaper than FPGAs. GPUs are present in all computers and therefore we avoid the necessity of having a dedicated and expensive hardware to deal with our problem. Moreover, the GPU implementation is hardware independent, in the sense that it can be used on GPUs from NVIDIA with none or little changes, depending on GPU's computing capabilities.

3. GPU implementation of the IRT-based mine detection system

The system that solves the thermal model using the explicit FD method was implemented using CUDA language, [11], and projected in a GPU from NVIDIA. The computing structure of GPUs makes them a suitable candidate to implement algorithms requiring high computing power. First we will introduce GPU characteristics and some basics about its programming mode. Then, we will present the proposed GPU implementation that simulates the thermal behavior of the soil and that speeds the computations up compared to a personnel computer.

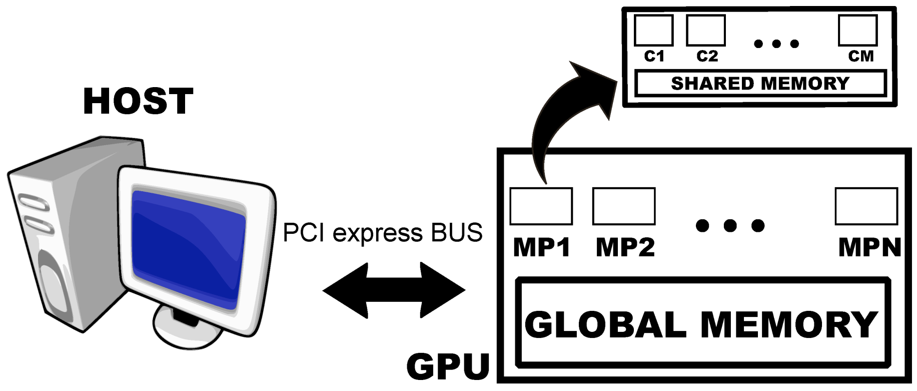


Fig. 2. *GPU internal structure and memory hierarchy.*

3.1 GPU structure

GPUs are made up of several multiprocessors that can perform parallel processing, which makes them suitable for processing in systems where the computations can be split up in independent portions and processed independently. The GPU is made up of several multiprocessors, labeled as MP1 ... MPN in Figure 2. Moreover, inside each multiprocessor there are several cores, labeled as C1 ... C. One important issue of GPU programming concerns the use of the different memories available in the GPU. A Global Memory is available to all multiprocessors and cores, whereas a Shared Memory inside each multiprocessor is only available to the corresponding multiprocessor's cores. Additionally, each core has its own and private memory space. One key aspect of a GPU-based system is the memory data organization and access, as they can impose a bottleneck in the system performance. The global memory has an access latency two orders of magnitude higher than the access to the shared memory. Thus, it is important to minimize the use of global memory and maximize, as far as possible, the use of shared memory because this will increase the performance of the system. Once the structure of the GPU has been briefly described we will introduce the basic aspects of GPU programming required to understand the structure of the proposed system. Functions in CUDA are called kernels and each kernel can be executed in parallel by several threads, as contrary to ordinary C/C++ functions that can only be executed by one processor. A kernel is not executed as a single thread, but it is executed as a block of threads, each of them processing the same function on different data, following a single-program multiple data (SPMD) computing model. Each thread inside the block has a 1D, 2D or 3D identifier (ID), depending on the applications, which distinguishes the concrete thread, to compute elements from a vector, matrix or volume of data. All the threads of a block are executed on the same multiprocessor and therefore they must fit within the available resources. This sets a limit on the maximum threads per block, which is limited to 512 in current GPUs. To avoid this limitation a kernel can be executed in several blocks of threads, which are organized as 1D or 2D groups of threads. The only requirement concerning the block of threads is that they must be independent from each other.

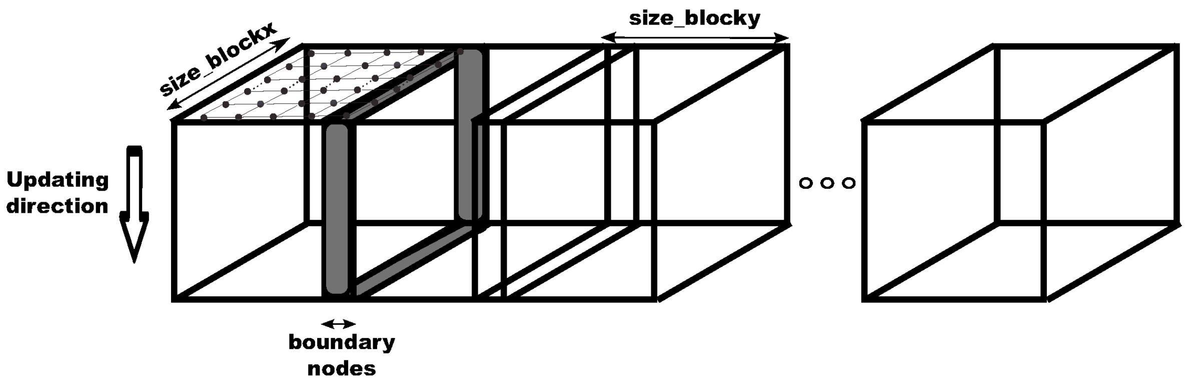


Fig. 3. *Temperatures updating scheme in the GPU*

3.2 GPU implementation of heat equation solver

GPU's structure fits perfectly our problem, where the full data can be split up in independent blocks that can be processed in parallel. Each multiprocessor can work with a portion of grid's nodes increasing the performance of the system. The GPU used in this work was a GTS-250 from NVIDIA (cost around 250 €). As was pointed, one of the main important aspects in an efficient CUDA-based system is the correct management of the memory to reduce the access to the global memory. To this aim the full grid of points, was divided into volume slices of size *size_blockx* · *size_blocky*, where the nodes' temperature of each slice is computed in a block of threads, see Figure 3. Each thread of the block is responsible for updating the temperature of the nodes with the same (x,y) coordinates within the considered piece of soil. The threads advance as a wave front, updating the nodes' temperature starting from the superficial layers to the inside of the soil, see Figure 3. It can be noted that there are overlapping areas between different blocks of threads, labeled as boundary nodes and indicated in grey in Figure 3, which must be taking into account to compute only once the new temperature value. Concerning the memory usage, the initial temperatures are stored in the global memory, and they have been transferred from the HOST memory to GPU global memory prior to the computation of the new temperatures. The temperatures are duplicated in the memory, as during one iteration we need to use one location to read temperatures and the other to write the updated values and in the following iteration the roles are interchanged. The remainder constant values needed in the computations, such as F_0 and values related to the boundary conditions, see Eq. (7), are also stored in the global memory. The access to the global memory should be minimized to increase the speed of the computations, because the global memory has a high latency access. Thus, we use, during the updating process, the shared memory of the multiprocessors to accelerate the access to the data. The memory operations are shown in Figure 4 where we can see the data transferences between the different memories of the GPU. In Figure 4 we consider the temperature updating process from nodes in Layer K. In STEP 1 the temperatures of Layer K-1 nodes are stored in both the multiprocessor's shared memory and in the cores' local memory. Moreover, nodes' temperatures from Layer k are read from the global memory and stored in the local cores' memory. During STEP 2 the nodes' temperatures from Layer K replace those from Layer K-1 in the shared memory, at the same time, the nodes' temperatures from Layer K+1 are read from global memory and stored in cores' local memory. In STEP 3 all data required to perform Layer K nodes' temperature updating is available on the local memory and shared memory. The same temperature of a Layer K is required to update the temperature of several nodes (the node itself and its north, south, west and east neighbors). If all nodes had to access global memory to read these values the process would be slowed, however once they are read from cores' local memory they are transferred to the shared memory, where they are available to all threads of the block, thus reducing the time access to the data. Once a thread has updated the temperature of a node, it uploads to the main memory the updated value and it continues computing the following temperature node updating. There is a synchronization process when a thread finishes one node's temperature updating because we must ensure that prior to continue with a node of the following layer all threads have finished the temperatures updating of the current layer.

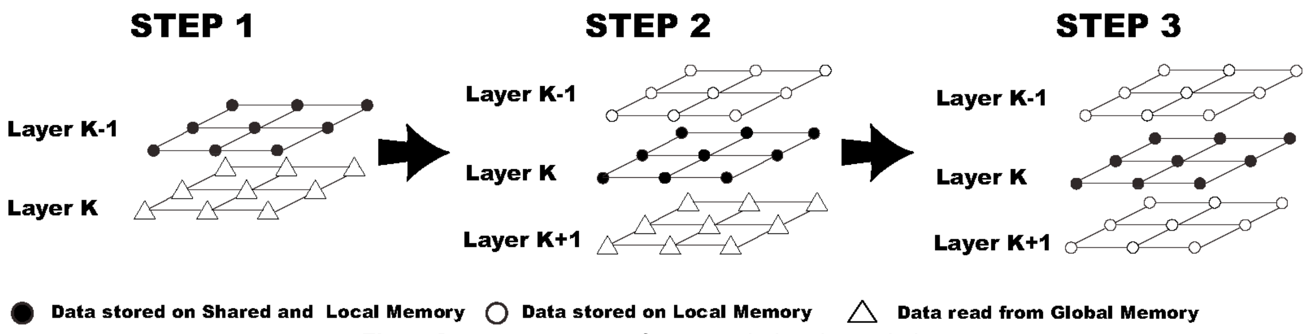

STEP 1
Layer K-1
Layer K
STEP 2
Layer K-1
Layer K
Layer K+1
STEP 3
Layer K-1
Layer K
Layer K+1
● Data stored on Shared and Local Memory
○ Data stored on Local Memory
△ Data read from Global Memory

Fig. 4. *Data memory transferences during the updating process.*

4. Results

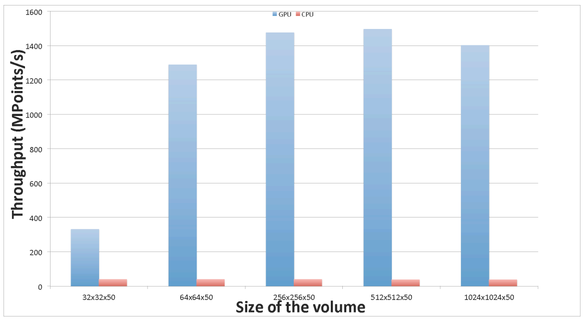
In this section we will introduce the results of the complete system. We divide this section into two main topics. On the one hand the results of the detection algorithm are shown. On the other hand, we will show the performance of the GPU implementation and how it improves the usability of the detection system reducing the processing time.

4.1 Infrared thermography-based landmine detection system

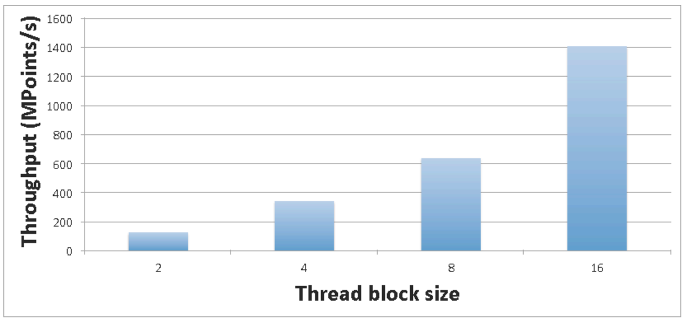
The detection algorithm proposed in [3] has shown very good results for the detection of antipersonnel landmines. It has been validated using infrared images from the TNO Physics and Electronics Laboratory, [12]. The validity of the algorithm was proved in [9, 13], its performance is summarized in Table 1, in which it is indicated the amount of mines correctly identified and detected.

Table 1*. Summary of the mines correctly detected and classified by the detection algorithm*

Mine location	Detected and classified	Total
Surface	12	12
Buried at 1 cm	6	6
Buried at 6 cm	5	6
Buried at 10 cm	1	1
Buried at 15 cm	0	1



(a) Throughput of GPU heat equation solver for different blocks' size.



(b) Throughput of GPU heat equation solvers implementations for different volume of simulated points. In the GPU implementation the block grids' size was set to 16x16.

Fig. 5. *Performance results comparing GPU and CPU throughputs for different setups.*

4.1 GPU heat equation solver

In this section we will show the performance of the GPU implementation and how it improves the usability of the detection system reducing the processing time. As was pointed a NVIDIA GTS 250 GPU, a low-cost GPU, was used to perform the comparison between a purely CPU implementation (Core2Duo 2.8 GHz implementation in C++) of the heat equation solver and a GPU implementation. One of the first issues is to think about the blocks threads' distribution and partitioning. The full volume of nodes which form the grid of points must be divided into blocks of threads, each of which is responsible for the nodes' temperature updating. The idea can be seen in Figure 3, where the volume has been divided into blocks of threads of size *size_blockx · size_blocky* which are sent to the MP of the GPU, in this case a 1D array of blocks is shown for the shake of clarity. Fig. 5(a) shows the performance of the GPU for various block sizes, where we have chosen *size_blockx = size_ blocky*, and a volume of 800 x 800 x 50 nodes. The results from these simulations can be seen in Table 2, where the speedup is compared to a purely CPU simulation of the full volume. It can be noted that the throughput of the system raises up as the size of the block is increased. This is due to the fact that when small blocks are used there are a lot of such small blocks spread over all MP, and therefore there will be a long queue of pending blocks to be processed. On the contrary, if the size of the blocks is increased we will have less blocks and the cue of pending blocks to be processed by the MP will be reduced. There is a limit, imposed by GPU's structure, given by the maximum number of threads that a MP can process (512). It can be seen from Fig. 5(b) that the throughput of the system is increased one order of magnitude when we change block thread size from 2 x 2 to 16 x 16 blocks of threads. Thus in the following simulations we will used this block's size for GPU simulations. Fig. 5(b) shows the GPU and CPU throughput for different nodes volumes, the data can be seen in Table 3. It can be noted how the performance of the GPU grows up one order of magnitude when the size of the volume is increased. This is due to the fact that for small volume of nodes not all GPU's resources are being used, whereas for big enough size volumes the inherent parallelism of the GPU increases the throughput of the system. It is obvious that for very big volumes the throughput will be low because there will be a cue of pending blocks to be processed which degrades the throughput of the system (note the reduction of the throughput for the 1024 x 1024 x 50 volume).

Table 2. *GPU throughput for a given volume of points and varying block thread's dimensions.*

block_dimx x block_dimy	***GPU throughput (Mpoints/s)***	***Speedup***
2 x 2	*120*	*3.1*
4 x 4	*336.5*	*8.7*
8 x 8	*634.3*	*16.7*
16 x 16	*1400.0*	*40.0*

In a previous work, [4], we presented an FPGA-based implementation of the heat equation solver. The system included a Virtex-II FPGA and a custom PCB with 6 memory banks. It achieved a speedup of 10 compared to a purely software solution. Moreover, an estimation of the performance of the system using a Virtex-5 FPGA was done, resulting in an speedup factor of 32. It can be noted that we have improved the performance of the heat equation solver using a GPU, which is also less expensive than modern, an expensive FPGA-based boards.

Table 3. *Throughput and CPU and GPU solvers of the heat equation solver for different size volumes.*

VOLUME	GPU THROUGHPUT (Mpoints/s)	CPU THROUGHPUT (Mpoints/s)	SPEEDUP
32 x 32 x 50	332	39.3	8.4
64 x 64 x 50	1289.33	39.3	32.6
256 x 256 x 50	1476.67	39.3	37
512 x 512 x 50	1496.02	39.3	39.3
1024 x 1024 x 50	1402.8	39.3	36.7

5. Conclusions

In this paper, we have introduced a landmine detection system based on infrared thermography using a 3D thermal model of the soil. However this is a very high time consuming algorithm, which jeopardizes its use in conventional computers. The efficient solution of the aforementioned procedures is successfully solved using a heat equation solver accelerator based on the use of GPUs, obtaining speed-up factors over 40. The speedup obtained with the proposed system with respect to nowadays computers, together with its low-cost and portability justifies the implementation as it permits its use on the field during demining operations.

REFERENCES

[1] Savelyev, T.G., van Kempen, L., Sahli, H.,Sachs, J., Sato, M. “Investigation of time-frequency features for GPR landmine discrimination”, IEEE Transactions on Geoscience and Remote Sensing 45(1), 118–129, 2007.
[2] Thanh, N.T., Sahli, H., Hao, D.N,. “Infrared thermography for buried landmine detection: inverse problem setting”, IEEE Transactions on Geoscience and Remote Sensing 46(12), 3987–4004, 2008
[3] López, P., van Kempen, L., Sahli, H., Cabello, D., 2004. “Improved thermal analysis of buried landmines", IEEE Transactions Geoscience and Remote Sensing 42(9), 1955–1964, 2004.
[4] Pardo, F., López, P., Cabello, D. and Balsi, M. "FPGA computation of the 3D heat equation", Computational Geoscience 14: 649–664, 2010.
[5] Khanafer, K., Vafai, K. "Thermal analysis of buried land mines over a diurnal cycle", IEEE Trans. Geosci. Remote Sens. 40(2), 461–473 (2002). doi:10.1109/36.992811
[6] Incropera, F., DeWitt, D. “Introduction to Heat Transfer”, 4th edn., John Wiley & Sons, 912pp, 2002.
[7] Watson, K.: Geologic applications of thermal infrared images. Proc. IEEE 63(1), 128–137 (1973)
[8] Wang, T., Chen, C.C.: 3-D thermal-ADI: a linear-time chip level transient thermal simulator. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 21(12), 1434–1445 (2002).
[9] López, P., Pardo, F., Sahli, H. and Cabello, D. "Non-destructive soil inspection using an efficient 3D software–hardware heat equation solver", In Inverse Problems in Science and Engineering, vol. 17, no. 6, pp. 755-775, September 2009.
[10] Pardo, F., López, P., Cabello, D. and Balsi, M. "Effcient software-hardware 3D heat equation solver with applications on the non-destructive evaluation of minefields", In Computers & Geoscience, vol. 35, pp. 2239-2249, 2009.
[11] NVIDIA (2010). NVIDIA CUDA C Programming Guide 3.1, NVIDIA Corporation Technical Staff.
[12] Jong, W., Lensen, H. and Janssen, H. "Sophisticated test facilities to detect land mines", Detection and Remediation Technologies for Mines and Minelike Targets IV, Vol. 3710 of Proceedings of the SPIE, pp. 1409–1418, 1999.
[13] Pardo, F., López, P. and Cabello, D. "Heat Transfer for NDE Landmine Detection", Developments in Heat Transfer, Dr. Marco Aurelio Dos Santos Bernardes (Ed.), ISBN: 978-953-307-569-3 (2011).