J Supercomput
DOI 10.1007/s11227-013-0891-1

Optimal tilt and orientation maps: a multi-algorithm approach for heterogeneous multicore-GPU systems

S. Tabik · A. Villegas · E.L. Zapata · L.F. Romero

Abstract This paper presents a new Geographic Information Systems (GIS) tool to compute the optimal solar-panel positioning maps on large high-resolution Digital Elevation Models (DEMs). In particular, this software finds out (1) the maximum solar energy input that can be captured on a surface located at a specific height on each point of the DEM, and then (2) the optimal tilt and orientation that allow capturing this amount of energy. The radiation and horizon algorithms we developed in previous works were used as baseline for this tool (Romero et al. in Comput. Phys. Commun. 178(11):800–808, 2008; Tabik et al. in Int. J. Geogr. Inf. Sci. 25(4):541–555, 2011). A multi-method approach is analyzed to make the hybrid implementation of this tool especially appropriate for heterogeneous multicore-GPU architectures. The experimental results show a high numerical accuracy with a linear scalability.

Keywords Maximum solar energy map · Optimum tilt map · Optimum orientation map · GIS · DEMs · Multicore-GPU heterogeneous system

1 Background and motivation

Knowledge of solar energy input that can be collected at a given terrain is essential to maximize energy production output. There exists a large number of applications where the investment depends directly on the accuracy of this knowledge. Some useful applications in this context are:

1. To preliminary know the solar energy input and deduce the electric energy output that could be produced by a given photovoltaic panel if installed on a specific place [1–3].

S. Tabik (✉) · A. Villegas · E.L. Zapata · L.F. Romero
Dept. Computer Architecture, University of Málaga, 29080 Málaga, Spain
e-mail: stabik@uma.es

Published online: 23 February 2013

2. To find out the most suitable sites for the installation of a solar plant and determine the optimal tilt and orientation of each one of their modules.
3. To preliminary know the quantity of solar energy that can be captured by a hypothetical solar plant. This is especially useful when a company or institution have to decide whether to invest in solar or eolic energy in a specific terrain and needs to compare the electric energy production of a hypothetical solar plant versus the energy output estimations of a hypothetical eolic plant in that specific place.
4. The need to determine the optimal tilt and orientation angles of solar collectors dedicated to drying crops in emerging countries for different periods [4–6].

There exist several baseline or kernel radiation models for calculating the incoming radiation in global areas represented by DEMs that fit in the main memory of single processor desktop. However, very few derived GIS-tools respond to the applications enumerated previously due to the high computational cost of the used kernel models. For instance, baseline models such as Solar Spatial Analyst tool implemented under the commercial ArcGIS 10 [7] and r.sun [8] implemented under GRASS GIS environment [9] calculate the clear-sky global irradiation maps considering the shadowing of the local terrain (i.e., produced by the DEM itself). Nevertheless, due to the computational limitations of the used shadow and radiation methods, both models can be applied only to small and medium DEM sizes.

Some works solve these limitations by using alternative numerical methods and parallel computing the model. For instance, in Refs. [10, 11] we presented a high performance radiation model faster by many orders of magnitude than all the current models for the same terrain sizes. We also developed a highly scalable multi-level horizon algorithm to make the computation even more efficient [12]. The radiation software is publicly available through URL [13].

All the models described above consider only the heights, tilts and aspects of the points of the land surface, i.e., the data provided by the DEM. However, they do not consider placing the collecting surface at a different or even variable heights, tilts and orientations. Recall that any modification of these parameters may change considerably the received radiation. Few related works have addressed applications affine to this problem but only on fix surfaces. In [1], the authors developed a tool called PVGIS based on r.sun [8] for estimating the photovoltaic potential on specific building surfaces, generally roofs with a predefined tilt and azimuth. A similar tool called PV was proposed in [3] to be implemented under ArcGIS. Both tools are limited to small urban areas due to the fact that the used sequential baseline model per se is costly from a computational point of view and building new tools on the top of it makes it even more expensive. A comparative summary of all the software described above is provided in Table 1.

This work came to improve the accuracy and computational performance of a preliminary implementation described in [14]. In particular, this paper presents a new tool which computes, first, the maximum solar energy input that can be collected at each solar module situated on each point of a vast high-resolution DEM and the optimal tilt and orientation maps that allow capturing that maximum solar radiation during different periods of time. We analyzed three search methods on GPU and multicore based systems: (1) an exhaustive method used for validation purposes,

Table 1 The difference between the proposed tool and related tools

GIS-Tool	Environment	DEM size	Target systems	Calculates	Horizons
Solar Analyst (free license)	ArcGIS	Small	Mono-processor	Solar radiation	Local
PV Analyst	ArcGIS	Small	Mono-processor	PV potential on roofs (based on Solar Analyst)	Local
r.sun (free license)	GRASS	Small	Mono-processor	Solar radiation	Local
PVGIS	GRASS	Small	Mono-processor	PV potential on roofs (based on r.sun)	Local
Our baseline model (free license)	–	Unlimited	Multi-processor	Solar radiation	Self+near+far No edge effect
The proposed GIS-tool	–	Unlimited	Multi processor	Maximum solar energy map. Optimal tilt map. Optimal orientation map.	Self+near+far No edge effect

(2) a divide-and-conquer algorithm, and (3) a gradient ascent algorithm. For numerical and performance comparisons we used two DEMs from the region of Andalusia, Spain, a DEM of a non-urban area of resolution 10×10 m^2 and a DEM of an urban area, which includes buildings, of the city of Málaga of resolution 1×1 m^2.

The calculated maximum solar energy and optimal angles maps show coherent results. The computational results show that the irregular load of the gradient ascent method makes it more suitable for multicore processors while the regular load of divide-and-conquer algorithm is more appropriate for GPU systems. The hybrid parallel implementation exploits these results by assigning the gradient ascent method to CPUs and divide-and-conquer to GPUs. The findings presented in this paper can be easily incorporated into GRASS GIS [9] and ArcView GIS [7] environments.

This paper is organized as follows. The baseline horizon and radiation algorithms, and their adaptation to calculate the maximum solar radiation, are provided in Sects. 2 and 3, respectively. A description and analysis of three search methods is provided in Sect. 4. The parallel CPU- and GPU-implementation and their optimizations are given in Sect. 5. The hybrid CPU-GPU implementation is evaluated and analyzed in Sect. 6. The numerical results are provided in Sect. 7 and conclusions are drawn in Sect. 8.

2 Horizon computation

The calculation of the shape of the horizon, also called shadow problem, of all the points of the terrain is essential to accurately calculate the radiation input. The multi-

level composition algorithm we developed in a previous work [12] is used as kernel for the tool presented in this work. This kernel was designed to take advantage of multiprocessor computers, which in consequence allowed us to increase the accuracy of the results and overcome memory limitations. It calculates the total horizon of a given point as the maximum elevation of three components: (1) an exact ground horizon, (2) a high-precision near-end horizon, and (3) a low-precision far-end horizon. These three horizons are computed separately using the appropriate DEM resolution as follows. First, the space around each point of the terrain is divided into $s = 360$ sectors, where $\hat{s} = 1°$, and then the horizons of all the points are calculated sector by sector till sweeping the entire 360 sectors. Upon completion, each point of the terrain will get an array of 360 elevation angle values. The optimizations of this algorithm for single and multi-processors are described in detail in [12].

A brief description of the three components of the horizon together with an analysis of their adaptation to calculate the maximum energy on solar modules can be summarized as follows.

2.1 Ground horizon

A horizontal surface represented by a point in the DEM views at most one half of the sky dome because the ground hides the other half. On inclined surfaces, there is an additional fraction of the sky dome that is not visible from the points that belong to that surface due to self-hiding, which we call ground horizon. This component is calculated in each sector using a formula $hor_g = \max(0, \alpha - 90°)$, where α is the angle between the normal to the surface (i.e., the point of the DEM) and the search direction $\overrightarrow{u_s}$.

Usually the center of the panel is placed at an altitude in the order of 1 m above the ground surface. In most cases, this height is negligible compared with the resolution of the DEM. In consequence the panel self shading must be added to the ground horizon. However, if the resolution of the used DEM is comparable with the panel altitude then the ground horizon must be replaced by the panel self horizon.

2.2 Near-end horizon

The near horizon produced by the closest points in the terrain is calculated using the full resolution DEM that fits entirely in the RAM memory. If the DEM does not fit in the memory hierarchy, a block partition strategy can be applied [12]. Placing a solar panel at a given altitude affects slightly this horizon if and only if the precision of the DEM is comparable with the size of the panel. Generally, this component is not required to be recalculated.

2.3 Far-end horizon

The far horizon is calculated using a low resolution halo DEM, that surrounds the high-resolution one taking into account the elevations placed behind a certain limit, roughly 1 to 10 km. This limit depends on the resolution of the used DEM, the desired accuracy and, the memory hierarchy of the used computing system. The fact of placing the solar panel at a certain elevation above the DEM does not affect at all this horizon component.

3 Solar radiation computation

The algorithm used to compute the solar irradiation that can be captured at each pixel of the DEM can be summarized as follows:

- The sky dome is discretized and the sun trajectory is calculated. The number of times the sun has passed from each pixel of the sky dome is calculated for all the considered time period. This calculation is done only once for all the points of the terrain.
- Using the previously calculated data, the energy matrices, $N1$, $N2$, $N3$, $N4$, $N5$ and, $N6$ are calculated on the considered area for a discrete set of horizon elevations, azimuths and, heights. These matrices, which are described in Ref. [10, 11], are shared between all the points of areas of similar atmospheric and astronomical conditions, usually areas of up to 100 km^2. The calculation at this stage does not depend on the DEM, it only needs the global latitude, longitude and some atmospheric parameters such as the turbidity factor of the considered region.
- The global solar irradiance, G_{ic} (W m^{-2}), that can be captured at point P of the DEM with tilt, also called inclination, γ_N, azimuth, also called aspect A_N and, height, also called elevation z, is calculated using the precomputed horizon as follows.

For each point P of the terrain
/* A_N *and* γ_N *must be known at this stage* */
  For $A_0 = 0,\ 360° - 1°$
    /* The near_hor() and far_hor() are already calculated */
    hor = max(ground_hor(), near_hor(), far_hor())
    val1+ = N_1(z, A_0, hor)
    val2+ = N_2(z, A_0, hor) · cos($A_0 - A_N$)
    val3+ = N_3(A_0, hor)
    val4+ = N_4(z, γ_N, A_0, hor)
    val5+ = N_5(z, A_0, hor) · cos($A_0 - A_N$)
    val6+ = hor/90
  End For
$G_{hc} = val1 + val3 \cdot F(\gamma_N) + val4$
$G_{ic} = k_c \cdot (\cos(\gamma_N) \cdot val1 + \sin(\gamma_N) \cdot val2 + val3 \cdot F(\gamma_N) + val4 + \sin(\gamma_N) \cdot val5 + \rho_g(x, y) \cdot G_{hc} \cdot val6/360)$
End For

Here, *val* with $(i = 1, 6)$ are auxiliary variables; h_0 and A_0 are the altitude and azimuth of the Sun, respectively; $F(\gamma_N)$ is a trigonometrical expression to consider the non-circumsolar component of the diffuse radiation; k_c is the clear-sky index, which represents the attenuation of the irradiation due to clouds. The global irradiation, G_h (W h m^{-2}), can be calculated using a numerical integration of the irradiance over a specific time interval on the order of hours, days, months or, years and using time steps of few minutes.

4 Optimal tilt- and orientation-maps computation

We developed three methods to compute the maximum solar energy and optimal angles maps: (i) the global maximum algorithm, used only for validation purposes since it is very costly from a computational point of view, specially for vast DEMs, (ii) the divide-and-conquer algorithm, and (iii) the gradient ascent algorithm. The three proposed methods are based on the horizon and irradiation kernels described in Sects. 2 and 3.

4.1 Global maximum algorithm

The first algorithm is a naive method which consists of exhaustively finding, for each point of the terrain, the combination (tilt γ_N, orientation A_N) that captures the maximum solar irradiation. This calculation can be carried out by sweeping all possible 90 tilt and 360 *orientation angles*, 1° by 1°. The found optimal combination corresponds to the maximum irradiation input that can be captured at that surface. The computational cost of this method can be reduced by considering only the tilt and orientation intervals that receive direct solar irradiation from the east to the west passing by the south. In fact, we experimentally found that the optimal tilt and orientation in most regions of the northern hemisphere belong to [30°, 75°] and [90°, 270°], respectively. Notice that this algorithm is regular since it iterates over the same number of solutions for each point.

4.2 Divide-and-conquer algorithm

For each point of the DEM, this method divides the space (γ_N,A_N) into n regions and evaluates the calculated irradiation at their centers. Afterwards, it iterates over the neighborhood of the center with the maximum energy till it reaches again the maximum value. Successive iterations will lead to the tilt and orientation in the solution space, which will be considered as the optimal solution.

This method applies successive partitions in tilt and orientation. Experimentally, we found that odd numbers of partitions ensures more accurate solutions for the majority of the points. We also found that carefully selecting the prime factorization of the tilt and orientation intervals reduces the total number of iterations. That is, the tilt interval, [30°, 75°], has $75 - 30 + 1$ tilt values; the nearest value to this number (46) whose prime factorization gives the smallest addition of factors is 45 ($3 \times 3 \times 5$). Consequently the range should be changed to [31°, 75°]. Specifically, we performed a partition of $5 \times 5 \times 7$ in orientation and $3 \times 3 \times 5$ in tilt as can be seen in Fig. 1. Where, first the orientation interval is partitioned into 5 blocks, then the tilt interval of the block with the largest value of irradiation is partitioned into 3 blocks and successively the block with the maximum value of irradiation in the current partition is partitioned into 5 blocks and so forth.

This algorithm is substantially faster than the global maximum algorithm. In addition, in DEMs that do not include buildings, it provides identical maximum energy and optimal angles in 100 % of the points. Nevertheless, in DEMs that do include buildings, the divide-and-conquer method show different results with respect to the global maximum in 6 % of the points.

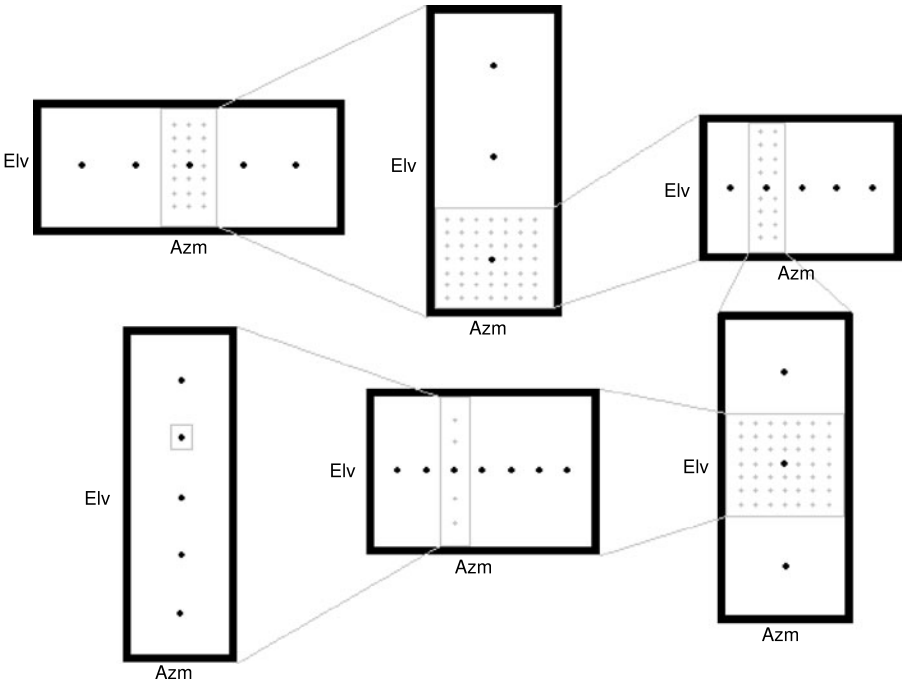

Fig. 1 Data partition of the orientation (Azm) and tilt (Elv) intervals in divide-and-conquer method

A further analysis showed that these points receive few direct solar radiation and the difference between their local and global maximum irradiation is not considerable, which means that they cannot be candidates for placing solar panels.

The computational load required for each point of the DEM is similar, which make this algorithm also regular.

4.3 Gradient ascent algorithm

The third algorithm consists of starting from an initial value of tilt and orientation, e.g., $\gamma_N = 34°$ and $A_N = 180°$, and moving in azimuth in the direction of the fastest grow irradiation values, until a maximum is reached. Then, using that value of the azimuth, the algorithm moves in orientation in the direction of fastest grow of irradiation values. Similarly to the second algorithm, this method stops the search when a maximum is reached. The obtained values of tilt and orientation will be considered as the optimal values.

Similarly to divide-and-conquer method, calculating the maximum energy and optimal angles using gradient ascent in DEMs that do not include buildings provide the global maximum in 100 % of the points. However, in DEMs that include buildings, the gradient ascent method gives different results with respect to the global maximum method in around 7 % of the points. Where the calculated panel position of almost 100 % of these erroneous points could receive an amount of energy very similar to the one obtained from the optimal position. A further analysis of these points showed that

all of them have very large obstacles in the south which prevent them from receiving direct solar energy in all orientations. Recall that diffuse and albedo radiations are too irregular and less important, which could imply the existence of several local maximum values. As these points receive few energy, about $<1/5$ Emax (Emax: the maximum energy received per point over all the DEM), they cannot be eligible to install a solar panel anyway. This finding can be used to further optimize both divide-and-conquer and gradient ascent algorithms by restricting the calculation only to the points that do not have important obstacles in the south.

The number of iterations performed by this method varies from point to point. For instance, in the DEM of the urban area of the city of Málaga of resolution 1×1 m^2, the experiment shows that each point needs an average of 7.74 iterations.

5 Parallel implementations for multicore- and GPU-based systems

Several optimizations were applied to make the parallel CPU and CUDA implementations more efficient.

5.1 CUDA-implementation

The CUDA implementation of the three algorithms is based on the next approach: Each thread is assigned one point of the terrain to calculate its maximum energy, optimal tilt and, optimal orientation. Due to memory limitations, for large DEMs, the global workload is organized into several calls to the CUDA-kernel, which can be either maximum global method, divide-and-conquer method or, gradient ascent method. For example, for a DEM of one million points, the global workload was organized into 2000 calls to one of the search algorithms to work on a pool of 500-point DEM-blocks. Experimentally, the optimal thread-block configuration for the GPU described in Table 2 is 5×100.

The parallel approach used for multicore systems is similar to the one used for GPU. That is, each thread (per core) is assigned one DEM-block to calculate its maximum solar energy and optimal orientation and tilt using one of the three previously described algorithms.

5.2 Data organization for GPU memory hierarchy

Our algorithm deals with these data structures: horizon, energy matrices, precalculated $\sin(\gamma_N)$, $\cos(\gamma_N)$, $\cos(A_0 - A_N)$ arrays and, the resulting optimal slop, optimal orientation, maximum energy maps. An appropriate mapping of these data in the GPU memory hierarchy is essential for the performance.

The horizon is an array that stores the elevations, seen in 360 directions, of each point of the terrain. For a block of 500 points, we need to allocate 180,000 bytes for the horizon. This information is too large to fit in registers or shared memory. It can only fit in global memory. Thus, first the horizon is allocated in the global memory, then each thread copies the horizon of the point it processes into local memory. These accesses should be synchronized to ensure that all the threads access the same local position at the same time.

Table 2 Architectural summary of the multicore and GPU used in the experiments

	Quad-Core AMD Phenom II X4 840T	NVIDIA GeForce GTX 480
# Cores	4	480
Clock speed	2.9 GHz	1.43 GHz
Main memory	8 Gb	1 Gb
Memory hierarchy	L1: 64 K + 64 K	Registers: 32768
	L2: 4 × 512 K	Shared memory: 48 Kb
	L3: 6 MB	L2: 524 K

Table 3 Average runtime per 500-point DEM block of the three search algorithms: global maximum, divide-and-conquer and, gradient ascent algorithms on a multicore processor (column 2) and on GPU (column 3); in addition to the speedup of the GPU-implementation with respect to the parallel cpu-implementation (column 4)

	CPU time (ms/block)	GPU time (ms/block)	Speedup ($\frac{CPU\ time}{GPU\ time}$)
Global maximum	13371.25	5108.80	2.62
Divide-and-conquer	88.10	12.70	6.94
Gradient ascent	37.15	60.41	0.62

The energy matrices are read by all the threads to compute the irradiation for all the points. Due to their high storage requirement, about 24 Mb, and irregular accesses, it is appropriate to allocate them in the global memory.

The calculated trigonometric values stored in arrays can fit in the shared memory. However, in this memory, as threads are synchronized, all the simultaneous accesses to the same values are serialized. Thus, we keep $\sin(\gamma_N)$, $\cos(\gamma_N)$ arrays in the shared memory and allocate the $\cos(A_0 - A_N)$ array in the local memory to get advantage of the aligned accesses.

We did not map data into constant and texture memories due to their low capacity and synchronization constraints.

5.3 Performance analysis

A comparison of the average runtime per DEM-block (among 2000 total number of blocks) and speedup of the global maximum, divide-and-conquer and, gradient ascent algorithms, on a multicore processor and one single GPU (whose characteristics are provided in Table 2) is shown in Table 3. For each method, we performed 10 executions and took the average runtime.

Constant atmospherical parameters are copied from CPU to GPU only once before the first kernel call and takes about 10,48 ms. The 500-point DEMs and their horizons are copied from host to device before each kernel call, for a total number of 2000 calls. The resulting maps are also copied back from device to GPU once per each kernel call. The runtime and speedup shown in Table 3 include the time needed to move data from and to GPU.

As it can be seen in Table 3, the gradient ascent algorithm is the most efficient method on CPUs since it gives substantially smaller runtime than the exhaustive and divide-and-conquer methods. However, the divide-and-conquer algorithm overcomes both the exhaustive and gradient ascent algorithms on GPUs showing a good scalability. This can be explained by the fact that divide-and-conquer has a well balanced load among all the points of the terrain which makes it specially appropriate for GPUs. Whereas, the gradient ascent algorithm has a low but irregular load which makes it more suitable for multicore systems.

6 Hybrid CPU-GPU implementation

This section provides, first, a description of the hybrid implementation, the used experimental setup and performance evaluation on a system that combines the CPU and GPU summarized in Table 2. Then it gives a discussion of the obtained results.

The calculation of solar radiation at a given point of the terrain does not require any information from the points of the neighborhood. Therefore, we implemented our algorithm using work-sharing data-parallel model based on block partition. Where first, the terrain is partitioned into blocks that fit entirely in both, the last level shared memory of multicore system and the global memory of the used GPUs.

Each block can be assigned to either an available core or GPU. Each running thread per core picks up a non-processed DEM-block from the pool of blocks and processes it using the gradient ascent algorithm. While, the GPU processes individual DEM-blocks received form the host using divide-and-conquer algorithm. The parallel implementation can be sketched as follows:

```
//Initialize the DEM-blocks counter
numBlocks = 0
//Create a number of threads = #cores-1+#GPUs
nthreads = deviceCount+coreCount-1
while(numBlocks<totalBlocks)
  if(Im==GPU)
    numBlocks++ // atomic operation
    execute(devide&conquer)
  endif
  if(Im==CPU)
    numBlocks++ // atomic operation
    execute(gradientAcent)
  endif
end
```

The parallel implementation of the hybrid algorithm was carried out using C++ programming language. To create and manage threads on the GPU and CPU we used NVIDIA CUDA [15] and openMP [16], respectively. All the executions were performed on Windows 7 64-bit OS. The optimal configuration for the experiments,

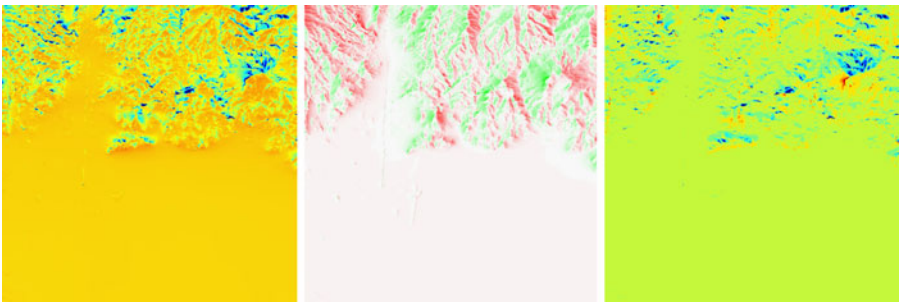

Fig. 2 (**a**) The annual maximum solar energy map that could be captured in the region of Málaga, where the *blue* and *red colors* correspond to 6000 and 7800 MW h m^{-2}, respectively; (**b**) optimal orientation map of the same region, where *green* and *red colors* correspond to the west and east, respectively, (**c**) optimal tilt map of the same region, where *blue* and *red colors* correspond to 30° and 50°, respectively (Color figure online)

which provides a good affinity, consists of dedicating one CPU-thread to copy data between host and GPUs and to be in charge of OS tasks. The parallel code is available via email from the corresponding author.

Executing the hybrid implementation on a system that combines both the multicore processor and GPU described in Table 2 takes an average time per block equals 9.5 ms, where the CPU and GPU process respectively 516 and 1484 DEM-blocks. Which means that the GPU executed around 74.20 % of the total workload.

Using the runtimes of the GPU-implementation and CPU-implementation given in Table 3, the hybrid implementation would obtain a linear scalability if the GPU processes 1489 blocks, which represents 74.45 % of the global workload. Notice that the global workload corresponds to 2000 DEM-blocks. The difference between the expected and the experimental performance is <1 % which corresponds to a number of DEM-blocks $<\pm 5$. This means that the produced interferences, intercores and between cores and GPU, are negligible and consequently the proposed hybrid implementation scales linearly with respect to the used number of cores and GPUs.

7 Numerical results

The numerical correctness and accuracy of the proposed high-performance algorithm is showed in two DEMs of different resolutions. As it can be verified in Fig. 2, the north-western faces of some mountains of the region of Málaga captures substantially less solar irradiation due to the shadow in the south produced by these mountains themselves. Constant values of irradiation (7800 MW h m^{-2}), tilt (34°) and orientation (180°) correspond to hypothetical panels at sea level, where are no obstacles. Similarly in Fig. 3, the shadow of the buildings affects substantially the quantity of the energy that can be produced in nearer locations.

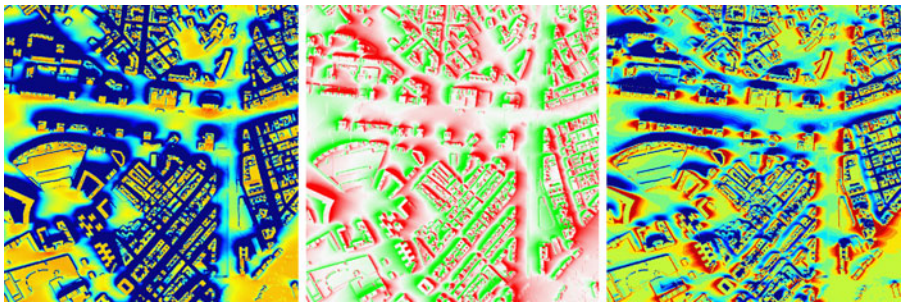

Fig. 3 (**a**) The annual maximum solar energy map that could be captured at the urban area of Málaga, where the *blue* and *red colors* correspond to 0 and 7800 MW h m^{-2}, respectively; (**b**) optimal orientation map of the same area, where *green* and *red colors* correspond to the west and east, respectively; (**c**) optimal tilt map of the same area, where *blue* and *red colors* correspond to 0° and 90°, respectively (Color figure online)

8 Conclusions

This paper presented a new fast, accurate, and parallel algorithm able to compute the maps of the maximum solar radiation input and the optimal angles, appropriate for large high-resolution DEMs on heterogeneous multicore-GPU systems. The performance analysis of different search methods on multicore processors and GPUs shows that certain algorithms that were ousted for CPUs due to its low efficiency should be reconsidered for GPUs as new factors must be taken into account. This applies especially to algorithms without conditional clauses such as the divide-and-conquer technique employed here. An other important conclusion in this context is that hybrid implementations should consider multiple algorithms, the most efficient one for each architecture.

The proposed GIS-tool is useful for many professionals in academia and industry. It provides a high accuracy, especially for areas with stable climatic conditions (i.e., where the cloudiness does not change too much with respect to the average value). Actually, it can be easily included in Geographical Information System (GIS) tools such as ArcGIS [7] and GRASS [9] environments.

Acknowledgements This work was supported by the Spanish Ministry of Science and Innovation TIN2010-16144 and the postdoc grant funded by the University of Málaga. We thank Nvidia for hardware donation under Professor Partnership 2008–2010, CUDA Teaching Center 2011–2012 and CUDA Research Center 2012 Awards.

References

1. Hofierka J, Kanuk J (2009) Assessment of photovoltaic potential in urban areas using open-source solar radiation tools. Renew Energy 34(10):2206–2214
2. Cellura M, Di Gangi A, Longo S, Orioli A (2012) Photovoltaic electricity scenario analysis in urban contests: an Italian case study. Renew Sustain Energy Rev 16(4):2041–2052
3. Choi Y, Rayl J, Tammineedi C, Brownson JRS (2011) PV analyst: coupling ArcGIS with TRNSYS to assess distributed photovoltaic potential in urban areas. Sol Energy 85(11):2924–2939

4. Bari S (2000) Optimum slope angle and orientation of solar collectors for different periods of possible utilization. Energy Convers Manag 41(8):855–860
5. Bakirci K (2012) General models for optimum tilt angles of solar panels: Turkey case study. Renew Sustain Energy Rev 16(8):6149–6159
6. Jafarkazemi F, Saadabadi SA (2012, in press) Optimum tilt angle and orientation of solar surfaces in Abu Dhabi, UAE. Renew Energy
7. ESRI (2011) ArcGIS Desktop: Release 10. Redlands, CA: Environmental Systems Research Institute
8. Suri M, Hofierka J (2004) A new GIS-based solar radiation model and its application to photovoltaic assessments. Trans GIS 8:175–190
9. GRASS Development Team (2012) Geographic resources analysis support system (GRASS) software. Open source geospatial foundation project. http://grass.osgeo.org
10. Romero LF, Tabik S, Vias J, Zapata EL (2008) Fast clear-sky solar irradiation computation for very large digital elevation models. Comput Phys Commun 178(11):800–808
11. Tabik S, Vias JM, Zapata E, Romero LF (2007) Fast Insolation Computation in Large Territories. In: International conference on computational science, vol 1, pp 54–61
12. Tabik S, Romero LF, Zapata EL (2011) High-performance three-horizon composition algorithm for large-scale terrains. Int J Geogr Inf Sci 25(4):541–555
13. www.ac.uma.es/~siham/research/radiation.tar.gz
14. Tabik S, Villegas A, Zapata EL, Romero LF (2012) A fast GIS-tool to compute the maximum solar energy on very large terrains. Proc Comput Sci 9:364–372
15. NVIDIA CUDA C programming guide. Available from URL. http://docs.nvidia.com/cuda/index.html
16. http://openmp.org/wp/openmp-specifications/