IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 18, 2020, accepted February 29, 2020, date of publication March 5, 2020, date of current version March 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2978590

GPU-Based Dynamic Solar Potential Estimation Tool Using 3D Plans

SÜMEYYE KAYNAK[1], BARAN KAYNAK[2], AND AHMET ÖZMEN[1]
[1]Department of Computer Engineering, Sakarya University, 54050 Sakarya, Turkey
[2]Computer Research and Application Center, Sakarya University, 54050 Sakarya, Turkey

Corresponding author: Ahmet Özmen (ozmen@sakarya.edu.tr)

ABSTRACT Estimations of the solar potential from the building design files may affect placement considerations in favor of more sunlight reception that reduces the energy costs and saves the environment. In this study, a GPU based system (GPU-DSRM) is proposed to estimate direct and diffuse solar radiation aggregated on 3D structures at urban or individual scale. In the proposed approach, finite element method, back-face detection and ray-tracing algorithms are customized to run in parallel to reduce the execution time. Thus, real-time shadow analysis with adjustable sampling rate and time scale can be performed without compromising precision and accuracy of the estimations. The most important novel aspect of the study is that it can be used anywhere in the world without the need for meteorological data. Some of the test results obtained from a site with 10 buildings are presented in this paper that shows a speedup value of 45 with the new GPU-based implementation compared to the CPU-based model. The GPU-DSRM tool has also been compared with geometric tools in the literature. Solar energy potential analysis of building designs or existing urban formations can be completed faster and more precisely with this new approach.

INDEX TERMS Solar potential analysis, GPU, 3D City Model.

NOMENCLATURE

α	Hourly elevation angle
β	Slope
δ	Declination angle
ω	Hour angle
θ	The angle between the normal of the sloped surface and the direction of the sun. The angle of the sun to the surface
ε	Elevation angle constant
φ	Latitude angle
H_b	Direct solar radiation on a horizontal surface
H_d	Diffuse solar radiation on a horizontal surface
H_{td}	Diffuse solar radiation on a sloped surface
S	Daily duration of daylight
θ_z	The zenith angle of the sun
H_0	Extraterrestrial irradiation
H	Global solar radiation value for horizontal surfaces
K_t	$\frac{H}{H_0}$, *clearness index*
R_d	Conversion factor of diffuse radiation
S_0	Maximum duration of daylight possible

The associate editor coordinating the review of this manuscript and approving it for publication was Sungroh Yoon.

I. INTRODUCTION

One of the most important indicators of development is the state of cities. Hence, the provision of quality housing and the formation of sustainable cities are necessary in order to improve living standards. The design of long-lasting, resilient housing against future global climate changes and the ability to work together with climate play central roles in the formation of sustainable cities. Many parameters should be analyzed in detail for a qualified housing design such as selection of land, location of the building facades, building geometry, lighting etc. Consideration of these parameters during the design phase will enable to achieve better performance from the active systems installed homes. Aside from this, retrofitting changes which is made to sustain an existing building or cities change the physical, social structure and solar potential of the building [1].

Solar energy is one of the renewable sources where active and passive systems are widely used to utilize in a sustainable city. The installation of buildings that optimally exploit solar energy potential provides the power to meet both thermal and electrical energy requirements [2]. In order to increase the utilization of solar energy in houses and cities, radiation analysis on 3D structures should be done in the most

accurate way to obtain better planning of energy resources and energy distribution systems [3]–[5]. The accuracy of the analysis depends on the holistic analysis of the 3D structures in complex urban environments and the effectiveness of the model.

There are many studies in the literature to estimate solar potential of buildings. As a result of these studies, two types of solar potential calculation software tool have been developed: rendering and geometric. Rendering tools are difficult to use and less preferable for general solar potential analysis since they require meteorological data and detailed material information of the structures. DIVA-for-Rhino and Honeybee/Ladybug are examples of rendering tools. On the other hand, geometric tools are less complex to use because they only use geometric relationships between the sun and the sky. Geometric tools consider only direct reflection, and usually don't rely on climate data and building material properties. ArcGIS and GRASS GIS are examples of geometric tools.

When geometric tools are examined, it is seen that there are few studies that perform radiation analysis of buildings in the design stage, consider the shadow effect [6], [7]. However, only the roofs of the buildings were focused and their facades were not taken into consideration in solar estimations. In these pixel-based studies, roof shadows are obtained by using the shading maps which is taken only a few times in a day. Hence, a dynamic shadow analysis is not possible in the pixel based systems.

Solar potential estimation becomes an iterative process when optimal position and facade forms are searched for a building. This may require many trials to obtain the best home design parameters, and when the problem becomes urban size, radiation analysis may take weeks to get the results [8]. Hence, high performance computers and appropriate algorithms are needed to obtain the results in shorter time.

Based on these shortcomings, a geometric based solar potential analysis tool development study was carried out in accordance with the following objectives: 1) The tool must be able to analyze solar potential for any part of the world (no meteorological data is needed), 2) it must take into account all building surfaces either in design phase or completed, 3) it must do dynamic shading analysis, 4) it must produce high precision and accurate results (by dividing surfaces into smaller pieces and dynamic sampling), 5) it must complete the analysis in a reasonable time.

During the study, it was understood that the conventional methods are not be suitable for this task; hence, a new Graphical Processing Unit (GPU) based high-performance parallel algorithm has been developed. Solar analysis can be carried out at any location in the world within a reasonable time using this new approach, including shadow analysis with roofs and facades using 3D building plans provided in CityGML format. The tool developed in this study is targeting urban planners, architects, civil engineers, energy investors and even individuals.

II. RELATED WORK

In literature, many models have been developed for solar radiation analysis, some of which are linear and the others nonlinear mathematical models, there are also artificial intelligence techniques and hybrid models [9], [10]. Solar geometry information (solar incidence angle, azimuth, latitude, longitude and hour angle etc.), atmospheric conditions, physical properties of the area to be analyzed (albedo value etc.) play important roles in determining the amount of radiation on horizontal and inclined surfaces.

Some models have been used for the radiation analysis of 3D buildings in an urban environment but these models alone are not sufficient. For example, shadows caused by structures or other objects in the environment must be taken into account. However, this require a detailed 3D information about the area and/or structure to obtain more accurate results. The detailed information is usually obtained using airborne and satellite images, Airborne Laser Scanning (ALS), Terrestrial Laser Scanning (TLS) techniques or Light Detection and Ranging (LIDAR) systems [11]–[13]. These technologies are used in many studies, such as analysis of solar radiation of building roofs in urban areas [14], [15], estimation of PV potential of solar radiation on roofs [16]–[18], determination of optimum position of PV panels on roof [19], evaluating solar radiation over facades [20], [21], and analysis of solar energy potential of buildings [22]–[24].

These technologies are costly and not suitable for newly designed buildings or living areas. Hence, a pixel-based approach is proposed to estimate solar potential on flat roofs which doesn't rely on technologies such as LiDAR, ALS, MLS [6]. In this study, buildings with flat roofs in a newly planned construction area are chosen as a case study. In another study, researchers are focused on estimating the solar potential of pitched roofs based on the pretext architectural design drawings which use a pixel based approach without technologies such as LIDAR, ALS, MLS [7]. A typical Australian house with nine roofs is chosen for case study. In this study, shadows are also considered by using shading maps.

Shadows have large impact on solar potential, therefore shading analysis should be done continuously for high accuracy estimates. A dynamic solar radiation model (DSRM) is presented at another work to evaluate the solar potential on 3D structures (facades, roofs) which can be either in planning stage or completed at the urban or individual scale [25]. A real-time shadow analysis in the desired sensitivity and time scale can be achieved using the proposed approach where the analysis takes approximately 11 hours. To reduce the response time, a new geometric method has been developed by another group to estimate the solar radiation of buildings using rooftop by 3D models [26].

The response time becomes an important factor when large scale solar radiation applications are considered including the dynamic shading analysis. High performance computer clusters and software techniques such as parallel computing

environments or specialized hardware devices are preferred for solar potential analysis or weather forecasting [27], [28]. Although cluster computing provides more flexible computing environment, multi-core systems are more preferred in existing studies due to their ease of programming and better price-performance ratio within smaller size [29].

Use of Graphical Processing Unit (GPU) architecture to meet high performance computing demand has been a frequently used approach in last decade. GPU architecture implements the Single Instruction Multiple Data (SIMD) model with a large number of cores [30]. The scientific community takes advantage of this technology to accelerate their applications by performing concurrent computation on GPU cores. This is usually referred as General Purpose Computation on GPU (GPGPU). Parallelization of serial code is a tedious task; however, it becomes more complex when GPU is considered due to its unique architecture [31]. If an algorithm was originally coded for a traditional computer, then the best is to redesign the algorithm for a target GPU architecture, then code it from scratch to obtain well performing concurrent execution.

Compute Unified Device Algorithm (CUDA) is a parallel computing platform and software programming model developed by NVIDIA to utilize GPU architecture for high performance problems. With the support of CUDA programming model, NVIDIA becomes dominant GPU architecture for many applications as well as geographic information systems applications [32].

A software developer with basic parallel programming knowledge and high-level programming skills can develop applications using with the NVIDIA-CUDA programming platform. In this study, CUDA programming model is preferred because it is more mature and has extensive documental support that allow coding in high-level languages. In order to benefit from the capabilities of the GPU architecture, applications running on the CPU can be moved onto the GPU; however, the correctness and performance analysis of the work must carefully be carried out.

Many high performance methods have been developed for faster solar radiation applications. Parallelization are the most preferred approach in order to meet high performance needs. There are many studies that benefit from the GPU-based parallelism in order to meet the objectives such as faster filtering of data used in solar radiation analysis studies [33]–[36], reducing the response time of the solar radiation analysis application [29], [37]–[39] and existing radiation analysis tools [40]. GPU method that can perform solar potential analysis of 3D building(s) which can either be in design stage or completed hasn't been encountered in the literature. The developed method uses 3D input files of the target building(s) at any scale. The better response time of the application allows examining alternative building designs and placement layouts within shorter time in solar potential perspective. The proposed method is described in detail in section IV.

III. PARALLELISM FOR PERFORMANCE AND CUDA ARCHITECTURE

Multi-core CPU architectures implement limited amount of parallelism via multi-threaded programming model and they are still far away to meet the increasing demand for high performance. The successive execution of the commands makes the performance of the CPUs inadequate for high computational requirements. Parallelism is a solution for this demand and can be achieved with several hardware models: distributed memory, shared memory or hybrid models.

In distributed memory model, a parallel system can be organized by connecting many computers using high speed network infrastructure, and parallelism can be obtained in the task level granularity. Distributed nature of the system causes significant communication and synchronization overheads while passing data between processes at separate computers. This type of systems is scalable, flexible; but they are hard to program.

There are several shared memory parallel architecture approaches in use. One of them is very well known multi-core CPU system. The other approach is to use multiple separate CPUs on the same board that share the main memory. This type of computers is usually implemented as workstation or server systems. Multiple-CPU systems are expensive, have fixed structure in nature and do not scale well with the problem size. However, they are comparatively easy to program, because compiler does all the tedious complex part of the task distribution between CPUs.

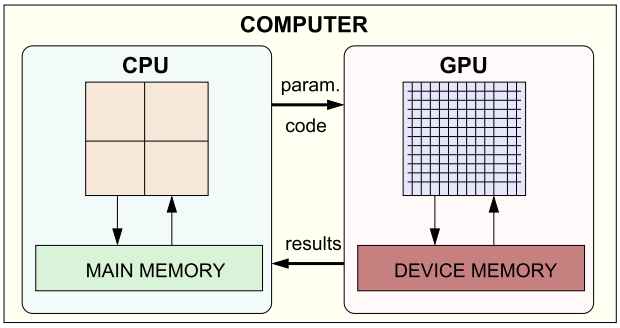

FIGURE 1. General purpose CPU-GPU architecture for high performance computing.

Like other systems, GPU architecture also implements a shared memory model with thousands of processing elements (GPU cores) connected to a unified memory. However, GPU computing differs from above parallel approaches because it comes as an attachment to a main system (see Figure 1). Moreover, the programmer must know internal details of the interested GPU architecture to develop application. Interestingly, GPU architecture allows data level parallelism which makes it superior for some applications [41]. As a solution to high performance requirements, GPU-supported high-performance computing systems are widely preferred because of their ease of use and price-performance advantage.

Multiple GPU boards can be installed into one case, and multiple GPU servers can be interconnected via a high speed network to form a hybrid parallel system.

NVIDIA created a model called Compute Unified Device Architecture (CUDA) to enable graphic cards to be used for high performance calculations [42]. CUDA allows code developers to program NVIDIA brand graphic cards in ordinary computers for their high performance workloads. CUDA architecture has a hierarchical layout in each device: grids, blocks and threads. A programmer can address the concurrent executions (threads) by using 3D arrays. CUDA source code running on the CPU is called ''host code'', and the code running on GPU is called the ''device code''. The functions required to run on the GPU are called ''kernel functions''. During the run, a new grid is created logically for each kernel call and the kernel function is divided into thread blocks [43]. The number of blocks and threads varies depending on the application, the most appropriate thread and block counts are obtained from the experiences. All blocks of the GPU must be in use for optimum performance.

IV. GPU-BASED SOLAR POTENTIAL ESTIMATION

In this study, a software tool that implements real-time shading analysis with GPU accelerated DSRM model have been developed to estimate solar energy potential more quickly and precisely. Detailed information on the dynamic solar radiation model is given in section IV-A. Section IV-B describes GPU based DSRM model in detail.

A. RESEARCH METHODOLOGY

Dynamic global solar radiation model has been developed to predict solar radiation potential of 3D structures at the individual/urban scale (see Figure 2). The model takes 3D plans and sun information such as altitude, azimuth, hour angles of the sun as input. The sun information such as declination and zenith angles are calculated using the equations introduced in a book by Duffie and Beckman [44]. The back-face detection algorithm is used to detect sun-facing surfaces. In the algorithm, all surfaces of interested buildings are divided into small triangles at the desired scale by using the finite element method. Each triangle on the front surface is subjected to ray-triangle-intersection test with the others, and this test determines the triangles that the sunlight directly reach. For the shadow-less triangles, global radiation amount equals the sum of direct and diffuse radiation. For shaded areas, the global solar radiation amount is taken as only the diffusion radiation value.

In this study, Angstrom–Prescot model is used to obtain the global solar radiation amount on horizontal surfaces [45], [46].

$$\frac{H}{H_0} = a + b * \left(\frac{S}{S_o}\right) \tag{1}$$

The coefficients a and b in Equation 1 are called Angstrom coefficients. The coefficient values $a = 0.25$, $b = 0.50$ proposed in [47] are used. The performance of the Angstrom-Prescot model with these suggested coefficient values has been evaluated for Kastamonu, Sakarya, Adıyaman, Afyon, Diyarbakır and İzmir. The model outputs are compared with global solar radiation data measured by the ''General Directorate of Renewable Energy/Turkey'' (http://www.eie.gov.tr). The model was tested for the cities mentioned in the previous work and verified that it produces acceptable results [25].

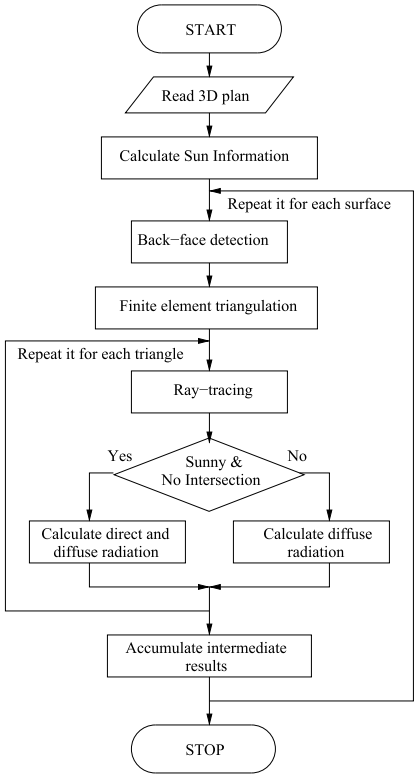

FIGURE 2. Basic steps of the DSRM model.

The direct radiation value on the horizontal surface is calculated on the basis of `Elevation Angle Constant` (EAC) method recommended by [48]. The general formulas of the EAC method are given in Equation 2 and Equation 3.

$$H_b = (H - H_d)/\varepsilon \tag{2}$$

$$\varepsilon = \begin{cases} \dfrac{\sum_{sunset}^{sunrise} \sin\alpha}{12} & for \ \ (S_0 \leq 12) \\ \dfrac{\sum_{sunset}^{sunrise} \sin\alpha}{day\ length} & for \ \ (S_0 > 12) \end{cases} \tag{3}$$

The diffuse radiation value on the horizontal surface is calculated on the basis of quadratic model recommended by [49]. The general formulas of the quadratic method are given in Equation 4.

$$\frac{H_d}{H} = 0.9885 - 1.4276K_t + 0.5679{K_t}^2 \tag{4}$$

The direct radiation value on a sloped surface depends on the permeability of the atmosphere and the direct radiation parameters on the horizontal surface. The permeability of the

atmosphere can be calculated by Equation 5.

$$R_b = max(0, \frac{\cos\theta}{\cos\theta_Z}) = max(0, \frac{\sin\delta\sin(\varphi-\beta)+\cos\delta\cos(\varphi-\beta)}{\sin\delta\sin\varphi+\cos\delta\cos\varphi\cos\omega}) \quad (5)$$

The diffusion radiation value on a sloped surface is calculated on the basis of Liu-Jordan model [50]. The Liu-Jordan model is obtained by multiplying the skewness of the oblique surface by the visual field value (Rd) with the horizontal surface diffusion radiation value:

$$H_{td} = H_d * R_d \quad (6)$$

In Liu-Jordan model, the Rd value is calculated using Equation 7.

$$R_d = \frac{(1+\cos\beta)}{2} \quad (7)$$

B. GPU BASED DSRM MODEL

The GPU-DSRM model has been developed to make the solar potential on 3D structures (facades, roofs) more precise and faster which can be either in planning stage or completed at the urban or individual scale. In some tools presented in the literature, the users had to wait long periods of time (such as 6 to 11 hours) to analyze solar potential of 3D plans using defined scenarios [25], [26]. This waiting period is very annoying for a user, and it deters the use of valuable solar analysis tools to obtain the best passive design parameters that provide optimum utilization of solar potential.

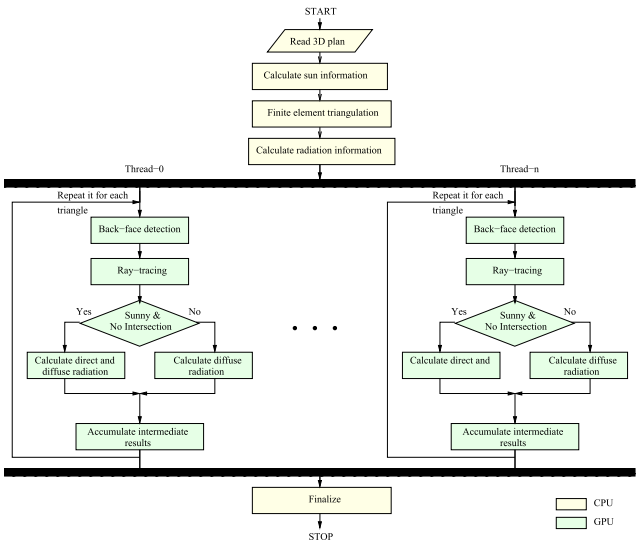

FIGURE 3. Flow diagram of GPU-DSRM algorithm.

GPU-DSRM tool uses the finite element method to divide the building facades into triangles at the desired scale for realization of the high precision analysis. The subject is studied before; however, the conventional computers with limited computation power cause long delays when the input size scales up [25]. In this context, the use of GPU has been found as a solution to the problem, and the GPU-DSRM algorithm has been developed to work on GPU architecture (see Figure 3).

Algorithm 1 CUDA Kernel Function for GPU-DSRM

Input : triangleList, sunInfoList
Output: diffuseRad, directRad, globalRad

```
tList ← triangleList
while time < sunInfoListLength do
    sunNormal ← calcSunNormalize(azimuth, altitude)
    for x ← threatIdx.x + blockDim.x * blockIdx.x;
    x < tListLength; x ← x + blockDim.x * gridDim.x
    do
        triangleNormal ←
        calcNormal(tList[x].v0, tList[x].v1, tList[x].v2)
        seeSun ← sunSee(sunNormal, triangleNormal)
        if seeSun then
            intersected ← false
            midPoint ← tList[x].midPoint
            for
            y ← threatIdx.y + blockDim.y * blockIdx.y;
            y < tListLength;
            y ← y + blockDim.y * gridDim.y do
                if !intersected then
                    intersected ← testIntersection(
                    midPoint, sunNormal, tList[y])
                end
            end
            diffuseRad[x] ←
            tList[x].diffuseList[time] + diffuseRad
            if intersected then
                globalRad[x] ←
                globalRad[x] + diffuseRad[x]
            else
                directRad[x] ←
                directRad[x] + tList[x].directList[time]
                globalRad[x] ← globalRad[x] +
                directRad[x] + diffuseRad[x]
            end
        end
    end
    time ← time + 1
end
```

The parallel algorithm requires two parameters. The first parameter is triangle array. These triangles are generated from 3D plans of interested structures using finite element method. The second parameter is sun information array. The sun information is used to calculate solar radiation amount of surface and to analyze whether a surface is shaded.

The following operations are performed during the analysis of each triangle:

- Using the back-face detection algorithm, it is determined whether the triangle is exposed to the sun.

- Using the ray tracing algorithm, it is determined whether the sunlit triangles are in the shadow of another triangle.
- If the triangle is exposed to direct sunlight, direct and diffusion solar radiation is calculated for the global solar radiation calculation.
- If the triangle is not exposed to direct sunlight, the diffusion solar radiation is calculated for the global solar radiation calculation.

The pseudo code of algorithm can be seen in Algorithm 1.

The serial version of the algorithm was coded using Visual Studio platform in C-Sharp programming language, aimed to run on classical multi-core CPU computers. When parallelization study has started, the language and platform became problem since the CUDA architecture do not support C-Sharp language. While searching a solution, a research study called Hybridizer by Altimesh was came across that develops converter tools for Visual Studio applications, and the research group was contacted for collaboration [51]. Hybridizer provides tools and libraries to generate source code or binaries from C-Sharp applications optimized for multi-core CPUs and GPUs. As a result, Hybridizer libraries are used while transferring some parts of our previous code to C++ that will run on GPU. The recommendations obtained from similar work are followed carefully while porting the application into GPU platform [27], [52].

When algorithm is parallelized, both correctness and performance tests must be performed to get the same output in less time with high utilization. Especially profiling tools become more critical since they show idling parts of the parallel run, and allows fine-tuning in the code. For this reason, the testing and optimization processes of parallel code are quite complex from those of serial programming.

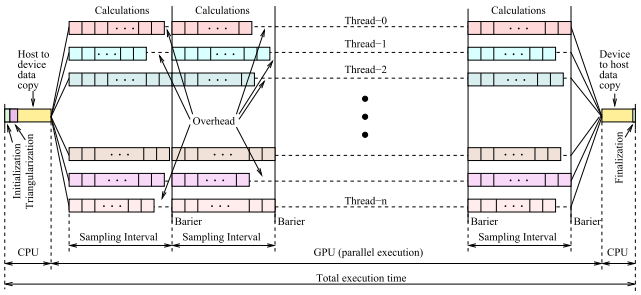

FIGURE 4. Parallel program activities and synchronization barriers.

At the left hand side of Figure 4, the input data are transferred from the host to the device, and this data is used throughout the time periods to be analyzed. The input data formed at the CPU side for each of the requested sampling time zones that affect the sensitivity of the analysis. Based on the data transferred, operations are performed on the GPU for the first sampling time interval, and immediately after all these processes are finished, the threads are activated for the next sampling time period. For a desired analysis time, the operations are executed repeatedly and the resulting data is transferred from the device memory to the host memory.

When profile output is examined, the triangularization process takes 1% and triangle intersection tests correspond to 99% of total execution time. Hence, the first part is decided to run on the CPU and the later part on the GPU (see Figure 4).

The data transfer from the CPU to the GPU and reverse may take quite large fraction in total execution time, hence the amount of data to transfer must be minimal. In this study, the following items are transferred to GPU memory before parallel execution starts:

- Coordinate information of triangles and calculated sun angle information for simulation time intervals.
- Lists of the direct and diffusion radiation values of the whole year for each triangle within the specified time interval.

The input and output data take up nearly 140 MB of memory in the GPU for the case studies presented in this work.

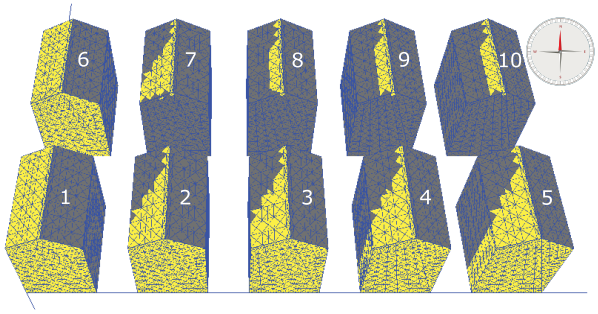

FIGURE 5. Sun seeing facades are divided into small triangular areas for calculation.

The Möller-Trumbore ray-triangle intersection algorithm used for shading analysis and back-face detection algorithm is implemented in a kernel function and executed in parallel [53]. The scenario consisting of 10 buildings with a surface area of 2453.64 m^2 is divided into 5694 triangles with an area of 4.3 m^2 (see Figure 5). Based on this scenario, with a half-hour sampling time resolution, the ray-triangle intersection algorithm corresponds to about 99 percent of the execution time of the DSRM model. An improvement in the ray-triangle intersection algorithm will greatly affect the DSRM model. Therefore, the ray-triangle intersection algorithm was executed in parallel on the GPU (see Figure 4).

The CUDA architecture allows a programmer to address the threads in x, y and z dimensions. In this work, the threads in x-dimension represent the triangles to be analyzed, and threads in the y-dimension represent the triangles of all buildings in the site in the GPU-DSRM model. The triangle to be analyzed is subjected to the intersection test in parallel manner with all other triangles. If any thread in x-dimension returns true in response to the intersection test, it is determined that the triangle is in shadow and does not receive direct solar radiation. The results of solar radiation are calculated according to these process outputs. An overview of the kernel function of the GPU-DSRM model is given in Algorithm 1.

Even though threads in x-dimension perform the same operations, they may not be able to finish their operations evenly. If a triangle in x-dimension intersects with one of the triangles in the y-dimension, the thread terminates the operation immediately and passes to the next job. Idle threads take the jobs in the queue. This procedure is repeated for the desired sampling interval period.

In the GPU-DSRM algorithm, threads do not need to communicate with each other, and the synchronization of threads is ensured by the barriers that are activated at certain intervals. The barriers can be considered to cause inefficiency as a common thought, and they are detrimental when used on all threads to meet the requirements of just a few threads [42]. However, this is not the case in this application. To calculate the next time interval, a common variable needs to be increased. To be able to update this variable for the next time period, we need to make sure that all threads are complete with barriers.

During the test phase, the accuracy analysis was performed before the performance review and the CPU versus GPU based results were compared for errors that might occur due to parallelism or floating point rounding differences. For this purpose, many test scenarios were created and the results were found to be the same. The test scenarios were conducted with different input parameters and comparative performance results are given in detail in the next section. Table 1 shows hardware and software components used for computations throughout the study.

TABLE 1. Test Computer Hardware and Software Components Used During the Experiments.

Component	Features
CPU	Quad Core Intel Core i7-7700HQ.
Main memory	16 GB RAM, 2400 MHz bandwidth.
OS	Windows 10.
GPU Card	Nvidia GeForce GTX 950M, 640 core. Maxwell architecture, Compute Capability 5.0.
Platform	CUDA, Version 9.1.

V. RESULT AND DISCUSSION

A new settlement with 10 buildings in Yazlık region of Sakarya is selected as testbed to validate the model and measure its performance (see Figure 5). The layout of the buildings are 2 rows and 5 columns with the distances between the rows and the columns are approximately 10 m and 15 m respectively. The buildings have the same dimensions: 25 m height, 17 m width and 23 m depth. The yellow triangles in the figure represent the ones that receive direct sunlight, and the gray ones show not receiving. This image is obtained from the GPU-DSRM algorithm for just specific time; e.g, it was captured for the January the 1st, at 16:30 when the sun is in the west direction and it is about sunset. Based on the sampling interval, this calculations are repeated several times even in one day throughout the observation period.

Test scenarios were created by using different triangle and sampling intervals, and then experiments were conducted to compare the performance of the GPU-DSRM model.

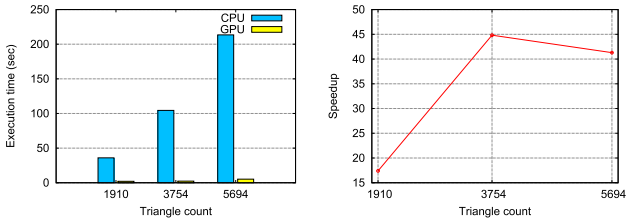

FIGURE 6. Comparison between CPU and GPU based implementation of DSRM algorithm: a) Execution time, b) Speedup.

Figure 6.a shows execution time comparison between CPU versus GPU implementation for 1910, 3754 and 5694 triangle counts and 1 hour sampling interval time for 31 days of data. As it can be seen from the figure, significant performance improvement has been achieved, and the execution time dropped down from 213.46 s to 5.13 s. with a speedup value of 41.29. Figure 6.b shows speedup values, where it initially increases with triangle count input linearly, and then becomes almost steady due to hardware limits of the GPU card is reached.

Triangle count and sampling interval time for a same target area effect the resolution and the accuracy of the output. The more triangle for the same surfaces makes the area smaller in the computations ends up higher precision. Similarly, the shorter sampling interval allows the use of more current solar beam angle for each triangle that increases the accuracy of the predictions. The goal of this research was to reduce the execution time of the tool without losing precision and accuracy at the output. Hence, in the experiments, both triangle count and sampling interval time are changed to see their impact over the execution time. Figure 7.a shows execution time for CPU and Figure 7.b for GPU with respect to triangle count and sampling interval time. The tool is executed 30 times according to different thread counts, and the average value is reported in the experiments.

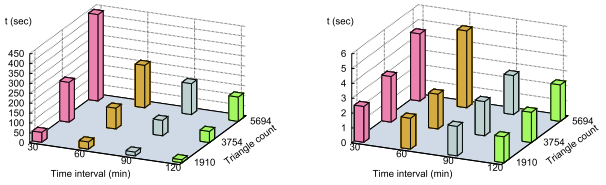

FIGURE 7. DSRM execution time for time interval 30, 60, 90, 120 minute versus triangle count 1910, 3754, 5694, a) CPU-DSRM, b) GPU-DSRM.

During the development of the GPU-DSRM model, it is observed that multiple triangle assignments to each thread increases the average waiting time and irregularity of threads. Considering the study model shown in Algorithm 1, it can be said that the algorithm can be scaled depending on the hardware, and the average waiting times of the threads are minimized. Aside from concurrent execution in multiple threads, some other improvements also employed in this new implementation, some of them are itemized as follows:

- Since the execution time of the serial version is very long, the results of the analysis for a desired time interval could be obtained at the end of long periods. Therefore, the radiation calculations for each triangle had to be performed at each time interval and these calculations were kept in a database. In the CPU-DSRM model, radiation calculations for any time interval could only be achieved quickly in this way. Due to the fact that the calculations made in the GPU are too short, and it is now unnecessary to keep the individual radiation values for each triangle separately. Instead, the average values of the radiation data for each triangle are calculated, and they kept in the memory rather than on the disk. In this way, the requested intermediate values can be retrieved in a very short time.
- In GPU-DSRM, primarily all surfaces are divided into triangles and all triangles are subjected to back-face detection algorithm. Then, the sun exposed triangles are subjected to the intersection test with all other triangles and the radiation calculations are made according to the results of this test. If this algorithm was applied to the surfaces first and the triangles were split, then it would be necessary to keep the information on which surface each triangle belongs to in GPU memory. This calculation is easily accomplished by scalar multiplication of the surface normal and the solar normal data. In this way, a search process and extra GPU memory data transfer is eliminated. This results in both accuracy and performance improvement. Therefore, the back-face detection algorithm is performed on a triangular scale on the GPU. Then, intersection testing and shading analysis are performed. While the calculations in the CPU-DSRM model last for several hours, the GPU-DSRM model is completed in seconds and there is no need to keep the temporal data generated in a database during the calculations.

As the problem size (i.e. triangle count) increases, the thread count also increases without affecting the execution time of the application till the hardware limits are reached. In fact, it has been observed that if the number of threads increased too much, this causes a negative effect on execution time. Therefore, it is necessary to take the optimum number of threads to meet the requirements and in order to massively exploit GPU capabilities [54]. As a result of the analysis, 4096 threads are found suitable for this application.

It is seen that different block-thread combinations in GPU do not change the execution time when the total thread count is kept constant. For this reason, we primarily focused on the total number of threads and it is set always as 4096. Then, the number of threads per block is set as 128.

Further experiments have also been conducted to see how triangle count affects the execution time while sampling interval is kept constant, and how sampling interval time affect the execution time while the triangle county kept constant. Figure 8.a and Figure 8.b show the effect of the number of triangles and the sampling interval on the execution time, respectively.

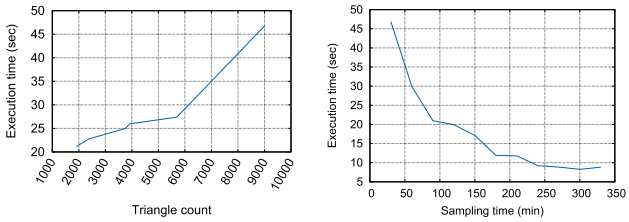

FIGURE 8. The effects of triangle count and sampling interval time on execution time: a) The triangle count is changing while the sampling interval time is constant, b) The sampling interval is changing and while the triangle count is constant.

As shown in Figure 8.a the whole year-round analysis process takes an average of 46.86 second with the triangles having a surface area of 2.71 m^2 each (makes 9024 triangles total) with constant 30-min time intervals. When the surface area of the triangles is increased to 4.3 m^2 (makes 5694 triangles), the execution time reduces to 27.38 second. Some more experiments are conducted in a similar way with the surface area of 12.84, 10.35, 6.53 and 6.26 m^2 (1919, 2370, 3754 and 3920 triangles respectively), it is observed that the execution time decreases linearly with the number of triangles when the sampling interval time is constant.

Another parameter that affects the accuracy and execution time of the tool is the sampling interval. Sampling interval tells how often the calculations should be done considering changing sun position. In order to see the effect of sampling interval on execution time, initially sampling interval with triangles having an area of 2.71 m^2 is set to 30 min. Figure 8.b shows when the sampling interval is taken for 30 minutes, the radiation analysis of the whole year takes place at an average of 46.863 seconds. It is obvious that the execution time decreases as sampling interval increases, and after a certain point of sampling interval increase, there is not much reduction in execution time because of the hardware limits of the GPU card.

As a result, the number of triangles and the sampling period affect the execution time, precision and accuracy. With the earlier version of the serial DSRM model run for annual analysis on the CPU, it took about 11 hours to complete the job. When the serial version of the algorithm was improved by eliminating the disk accesses, the execution time was reduced to 6 hours. Although this is a successful performance improvement, it is still far from an acceptable value for a tool that is supposed to work interactively on the design files. However, in the GPU based implementation, the same job takes approximately 47 seconds for the same input parameters. In this example, the working area is divided into 2.71 m^2 triangles and 30-minute time slices are taken for sampling interval time for both runs. As it can be seen from this comparative study, the GPU-DSRM model runs about 842 times faster than the earlier DSRM model for the same parameters. It is expected that this acceleration will be much better when

GPU cards are used with more cores. Therefore, the results obtained in this study will lead to new researches for different GPU cards.

The GPU-DSRM has been compared with other software tools such as ArcGIS and ''skymapping'' in terms of performance, easiness of usage, input data type and precision [26]. The common characteristics of these tools are that they all belong to geometric tools and neglect the reflected radiation.

Although ArcGIS is used for mapping purposes in general, it is also a widely preferred software tool that performs solar potential analysis. However, ArcGIS cannot read 3D data as input [55]; hence, the input image must be annotated with auxiliary data by the user to compensate for missing information before performing the solar analysis. Nevertheless, the annotated input of structures does not provide as much information as 3D plans have. The GPU-DSRM tool accepts 3D plans and performs radiation analysis using more comprehensive knowledge such as width, length, height, slope, etc. about the buildings under consideration. As a result, GPU-DSRM tool produces higher accuracy and resolution results in solar estimations.

On the other end, the skymapping tool performs radiation analysis using only the rooftops of buildings extracted from 3D plans. For example, the performance of our early DSRM model was compared with the skymapping tool using the same test bed in [26]. The GPU-DSRM developed in this study is over performed the skymapping tool by employing novel approaches in the algorithm. Since both applications used the same test bed in their studies, comparative execution times can be given as follows: the skymapping tool completed the task using 69726 triangles in approximately 3 hours, while the GPU-DSRM tool did the same task in approximately 6 minutes. As it can be seen from the results, the performance of the DSRM algorithm is greatly improved by implementing the algorithm in parallel using the GPU hardware with considerably less cost.

Limitations: The tool developed in this study has the following limitations when compared with other tools in the literature:

- In this study, the analysis covers only solar radiation falling on the exterior of a building.
- The reflected radiation is neglected because it generally constitutes only a small proportion of total radiation.
- The software tool is implemented using CUDA architecture because it is a widely used model, hence the tool runs only on NVIDIA Graphics cards.
- The input files can be only in CityGML format because it is a standard for 3D representation of structures.
- Covering material types over the facades was not considered in this research because our goal was to find the maximum solar potential that a building may receive.

VI. CONCLUSION AND FUTURE WORK

Utilizing solar energy at buildings saves energy and reduce the environmental pollution, hence the solar analysis is an important task to explore the potential before or after the construction. On the other hand, the demand for high precision and accurate predictions enforce algorithm developers involve more parameters into their algorithm which causes longer time to produce the expected results. As a result, the existing solar analysis tools either produce limited outputs or they are not practical to use due to long running times.

In this paper, a new geometric-based solar radiation estimation tool (GPU-DSRM) development study is presented. The tool analyzes buildings holistically using 3D building plans in CityGML format. It also takes into account dynamic shadow analysis and can be used anywhere in the world to perform solar analysis. It is observed that it can take hours or even days to complete a solar analysis task for sites with multiple buildings when used similar tools in the literature [8], [25], [26]. The long execution time had a deterrent effect on the examination of alternative designs. Hence, the software tool has been developed using up-to-date parallel GPU technologies which provides parallelism at the data level, ease of use and price-performance advantage.

The algorithm has been implemented using NVIDIA GTX950M GPU hardware and its performance measured and compared against existing tools in the literature such as 'skymapping' and 'DSRM'. The measured speedups are average of 30, 45 and 842 fold against the skymapping, improved-DSRM and initial version of DSRM respectively.

It can be concluded that high resolution solar potential analysis in 3D plans can be done in an acceptable time period by using better GPU hardware which have more cores or by using several GPU servers that contain multiple GPU boards. In this way, architects and urban planners will be able to examine alternative home designs in terms of solar potential in shorter time.

In future studies, it is aimed that users will be able to upload 3D building plans to the system with a web interface and make all calculations on a remote server located in a cloud. In this way, users who do not have high performance hardware will be able to use this software remotely. The calculations can be further accelerated by installing multiple GPUs on the server(s) in the cloud.

REFERENCES

[1] B. Ozarisoy and H. Altan, ''Low energy design strategies for retrofitting existing residential buildings in cyprus,'' *Proc. Inst. Civil Engineers Eng. Sustainability*, vol. 172, pp. 241–255, Jul. 2018.

[2] B. Ozarisoy and H. Elsharkawy, ''Assessing overheating risk and thermal comfort in state-of-the-art prototype houses that combat exacerbated climate change in UK,'' *Energy Buildings*, vol. 187, pp. 201–217, Mar. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778818331839

[3] A. S. B. M. Shah, H. Yokoyama, and N. Kakimoto, ''High-precision forecasting model of solar irradiance based on grid point value data analysis for an efficient photovoltaic system,'' *IEEE Trans. Sustain. Energy*, vol. 6, no. 2, pp. 474–481, Apr. 2015.

[4] A. Shakya, S. Michael, C. Saunders, D. Armstrong, P. Pandey, S. Chalise, and R. Tonkoski, ''Using Markov switching model for solar irradiance forecasting in remote microgrids,'' *Proc. IEEE Energy Convers. Congr. Expo. (ECCE)*, Sep. 2017, vol. 8, no. 3, pp. 895–905.

[5] A. Bracale, G. Carpinelli, and P. De Falco, "A probabilistic competitive ensemble method for short-term photovoltaic power forecasting," *IEEE Trans. Sustain. Energy*, vol. 8, no. 2, pp. 551–560, Apr. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7570246/

[6] Y. Li, D. Ding, C. Liu, and C. Wang, "A pixel-based approach to estimation of solar energy potential on building roofs," *Energy Buildings*, vol. 129, pp. 563–573, Oct. 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0378778816307101

[7] Y. Li and C. Liu, "Estimating solar energy potentials on pitched roofs," *Energy Buildings*, vol. 139, pp. 101–107, Mar. 2017.

[8] M. Bayrakci, K. Calvert, and J. Brownson, "An automated model for rooftop PV systems assessment in Arcgis using lidar," *AIMS Energy*, vol. 3, pp. 401–420, Sep. 2015.

[9] A. Asrari, T. X. Wu, and B. Ramos, "A hybrid algorithm for short-term solar power prediction—Sunshine state case study," *IEEE Trans. Sustain. Energy*, vol. 8, no. 2, pp. 582–591, Apr. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7577881/

[10] R. H. Inman, H. T. C. Pedro, and C. F. M. Coimbra, "Solar forecasting methods for renewable energy integration," *Progr. Energy Combustion Sci.*, vol. 39, no. 6, pp. 535–576, Dec. 2013, doi: 10.1016/j.pecs.2013.06.002.

[11] M. Boulifa, A. Adane, A. Mefti, S. Ameur, and Z. Ameur, "The solar radiation evaluation by satellite images processing," in *Proc. Int. Conf. Renew. Energies Power Qual. (ICREPQ)*, 2010, pp. 1–4. [Online]. Available: http://www.icrepq.com/icrepq'10/324-Boulifa.pdf

[12] A. W. Azhari, K. Sopian, A. Zaharim, and M. Al Ghoul, "A new approach for predicting solar radiation in tropical environment using satellite images-case study of Malaysia," *WSEAS Trans. Environ. Develop.*, vol. 4, no. 4, pp. 373–378, 2008.

[13] A. Olpenda, K. Stereńczak, and K. Będkowski, "Modeling solar radiation in the forest using remote sensing data: A review of approaches and opportunities," *Remote Sens.*, vol. 10, no. 5, p. 694, May 2018. [Online]. Available: http://www.mdpi.com/2072-4292/10/5/694

[14] N. Lukac, D. Zlaus, S. Seme, B. Zalik, and G. Stumberger, "Rating of roofs' surfaces regarding their solar potential and suitability for PV systems, based on LiDAR data," *Appl. Energy*, vol. 102, pp. 803–812, Feb. 2013.

[15] M. Brumen, N. Lukac, and B. Zalik, "Gis application for solar potential estimation on buildings roofs," Int. Acad., Res., Ind. Assoc., Chamonix, France, Tech. Rep., 2014.

[16] C. Carl, "Calculating solar photovoltaic potential on residential rooftops in Kailua Kona, Hawaii," Ph.D. dissertation, USC Graduate School, Univ. Southern California, Angeles, CA, USA, 2014. [Online]. Available: https://search.proquest.com/docview/1560671278?accountid=13654

[17] M. C. Brito, N. Gomes, T. Santos, and J. A. Tenedório, "Photovoltaic potential in a lisbon suburb using LiDAR data," *Sol. Energy*, vol. 86, no. 1, pp. 283–288, Jan. 2012.

[18] T. Santos, N. Gomes, S. Freire, M. C. Brito, L. Santos, and J. A. Tenedório, "Applications of solar mapping in the urban environment," *Appl. Geography*, vol. 51, pp. 48–57, Jul. 2014.

[19] C. D. Brandt, "Rooftop solar capacity modeling using GIS within the city of Stillwater, MN," Ph.D. dissertation, Univ. Wisconsin-River Falls, River Falls, WI, USA, 2014. [Online]. Available: http://digital.library.wisc.edu/1793/68317

[20] A. Martínez-Rubio, F. Sanz-Adan, J. Santamaría-Peña, and A. Martínez, "Evaluating solar irradiance over facades in high building cities, based on LiDAR technology," *Appl. Energy*, vol. 183, pp. 133–147, Dec. 2016.

[21] A. Jochem, B. Höfle, and M. Rutzinger, "Extraction of vertical walls from mobile laser scanning data for solar potential assessment," *Remote Sens.*, vol. 3, no. 4, pp. 650–667, Mar. 2011.

[22] P. Redweik, C. Catita, and M. Brito, "Solar energy potential on roofs and facades in an urban landscape," *Sol. Energy*, vol. 97, pp. 332–341, Nov. 2013.

[23] P. Huang, M. Cheng, Y. Chen, D. Zai, C. Wang, and J. Li, "Solar potential analysis method using terrestrial laser scanning point clouds," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 3, pp. 1221–1233, Mar. 2017.

[24] C. Catita, P. Redweik, J. Pereira, and M. C. Brito, "Extending solar potential analysis in buildings to vertical facades," *Comput. Geosci.*, vol. 66, pp. 1–12, May 2014.

[25] S. Kaynak, B. Kaynak, and A. Özmen, "A software tool development study for solar energy potential analysis," *Energy Buildings*, vol. 162, pp. 134–143, Mar. 2018.

[26] M. U. Yousuf and S. M. Shere, "A novel computational methodology to estimate solar energy on building rooftops," *Environ. Progr. Sustain. Energy*, Jan. 2020. [Online]. Available: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/ep.13385, doi: 10.1002/ep.13385.

[27] J. P. Silva, J. Hagopian, M. Burdiat, E. Dufrechou, M. Pedemonte, A. Gutiérrez, G. Cazes, and P. Ezzatti, "Another step to the full GPU implementation of the weather research and forecasting model," *J. Supercomput.*, vol. 70, no. 2, pp. 746–755, Apr. 2014. [Online]. Available: http://link.springer.com/10.1007/s11227-014-1193-y

[28] R. M. Gualan-Saavedra, L. D. Solano-Quinde, and B. M. Bode, "GPU acceleration of the horizontal diffusion method in the weather research and forecasting (WRF) model," in *Proc. Asia–Pacific Conf. Comput. Aided Syst. Eng.*, Jul. 2015, pp. 284–289. [Online]. Available: http://ieeexplore.ieee.org/document/7287033/

[29] Y. Huang, Z. Chen, B. Wu, L. Chen, W. Mao, F. Zhao, J. Wu, J. Wu, and B. Yu, "Estimating roof solar energy potential in the downtown area using a GPU-accelerated solar radiation model and airborne LiDAR data," *Remote Sens.*, vol. 7, no. 12, pp. 17212–17233, Dec. 2015.

[30] J. Salvador, O. Sofía, F. Orte, E. D. Santos, H. Oyama, T. Nagahama, A. Mizuno, and R. Uribe Paredes, "Performance improvements of an atmospheric radiative transfer model on GPU-based platform using CUDA," in *Proc. 12th Congreso Argentino Ciencias Comput.*, Jun. 2015.

[31] K. Thouti and S. Sathe, "Comparison of Openmp & Opencl parallel processing technologies," *Int. J. Adv. Comput. Sci. Appl.*, vol. 3, pp. 56–61, Nov. 2012.

[32] C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 508–527, Sep. 2011.

[33] X. Hu, X. Li, and Y. Zhang, "Fast filtering of LiDAR point cloud in urban areas based on scan line segmentation and GPU acceleration," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 308–312, Mar. 2013.

[34] S. Pečnik and B. Žalik, "Real-time visualization using GPU-accelerated filtering of lidar data," *World Acad. Sci., Eng. Technol., Int. J. Comput., Elect., Automat., Control Inf. Eng.*, vol. 8, no. 12, pp. 2108–2112, 2014.

[35] R. Sugumaran, O. Dossay, and P. Gray, "GPU-based cloud performance for LiDAR data processing," in *Proc. Int. Conf. Comput. Geospatial Res. Appl.*, May 2011, p. 48.

[36] J. Li, Y. Xu, H. Macrander, L. Atkinson, T. Thomas, and M. A. Lopez, "GPU-based lightweight parallel processing toolset for LiDAR data for terrain analysis," *Environ. Model. Softw.*, vol. 117, pp. 55–68, Jul. 2019.

[37] S. Tabik, A. Villegas, E. L. Zapata, and L. F. Romero, "A fast GIS-tool to compute the maximum solar energy on very large terrains," *Procedia Comput. Sci.*, vol. 9, pp. 364–372, 2012, doi: 10.1016/j.procs.2012.04.039.

[38] Z. Yanjie, Y. Guoqing, and D. Yanqing, "Parallel computation of ground radiation simulation based on GPU," *Procedia Comput. Sci.*, vol. 107, pp. 9–14, 2017, doi: 10.1016/j.procs.2017.03.049.

[39] N. Lukač and B. Žalik, "GPU-based roofs' solar potential estimation using LiDAR data," *Comput. Geosci.*, vol. 52, pp. 34–41, Mar. 2013.

[40] T. V. Russkova, "Monte Carlo simulation of the solar radiation transfer in a cloudy atmosphere with the use of graphic processor and NVIDIA CUDA technology," *Atmos. Ocean. Opt.*, vol. 31, no. 2, pp. 119–130, Apr. 2018.

[41] M. A. Rahman and R. C. Muniyandi, "Review of GPU implementation to process of RNA sequence on cancer," *Informat. Med. Unlocked*, vol. 10, pp. 17–26, 2018, doi: 10.1016/j.imu.2017.10.008.

[42] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," *J. Parallel Distrib. Comput.*, vol. 68, no. 10, pp. 1370–1380, Oct. 2008.

[43] M. Gebremedhin, A. H. Moghadam, P. Fritzson, and K. Stavåker, "A data-parallel algorithmic modelica extension for efficient execution on multi-core platforms," in *Proc. 9th Int. MODELICA Conf.*, Munich, Germany Sep./Nov. 2012, pp. 393–404. [Online]. Available: http://www.ep.liu.se/ecp/article.asp?issue=076%26article=41

[44] J. A. Duffie and W. A. Beckman, *Solar Engineering of Thermal Processes*. Hoboken, NJ, USA: Wiley, 2013.

[45] A. Angstrom, "Solar and terrestrial radiation," *Quart. J. Roy. Meteorol. Soc.*, vol. 50, pp. 121–126, Apr. 1924.

[46] J. Prescot, "Evaporation from a water surface in relation to solar radiation," *Trans. Roy. Soc. South Aust.*, vol. 64, pp. 114–118, 1940.

[47] B. Delgado, M. Paredes, and M. Martínez, "Software application for calculating solar radiation and equivalent evaporation in mobile devices," *Agricult. Water Manage.*, vol. 151, pp. 30–36, Mar. 2015.

[48] C. S. Solanki and C. S. Sangani, "Estimation of monthly averaged direct normal solar radiation using elevation angle for any location," *Sol. Energy Mater. Sol. Cells*, vol. 92, no. 1, pp. 38–44, Jan. 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0927024807003121
[49] S. Tarhan and A. Sari, "Model selection for global and diffuse radiation over the central black sea (CBS) region of turkey," *Energy Convers. Manage.*, vol. 46, no. 4, pp. 605–613, Mar. 2005.
[50] B. Liu and R. Jordan, "Daily insolation on surfaces tilted towards equator," *ASHRAE Trans.*, vol. 10, Oct. 1962.
[51] Altimesh. (2017). *Hybridizer*. Accessed: Apr. 10, 2019. [Online]. Available: http://www.altimesh.com/
[52] J. Delgado, "A case study on porting scientific applications to GPU/CUDA," *J. Comput. Interdiscipl. Sci.*, vol. 2, no. 1, pp. 3–11, 2011.
[53] T. Möller and B. Trumbore, "Fast, minimum storage ray/triangle intersection," in *Proc. ACM SIGGRAPH Courses*, 2005, p. 7-es, doi: 10.1145/1198555.1198746.
[54] M. Chiesi, L. Vanzolini, E. Franchi Scarselli, and R. Guerrieri, "Accurate optical model for design and analysis of solar fields based on heterogeneous multicore systems," *Renew. Energy*, vol. 55, pp. 241–251, Jul. 2013.
[55] A. Chow, A. Fung, and S. Li, "GIS modeling of solar neighborhood potential at a fine spatiotemporal resolution," *Buildings*, vol. 4, no. 2, pp. 195–206, May 2014.

SÜMEYYE KAYNAK received the B.S., M.S., and Ph.D. degrees in computer engineering from Sakarya University, Sakarya, Turkey, in 2012, 2014, and 2019, respectively.

Since 2012, she has been a Research Assistant with the Department of Computer Engineering, Sakarya University. Her research interests include energy, software system development, blockchain, and artificial intelligence.

BARAN KAYNAK received the B.S. and M.S. degrees in industrial engineering from Sakarya University, Sakarya, Turkey, in 2011 and 2013, respectively, where he is currently pursuing the Ph.D. degree in industrial engineering.

He joined the Computer Research and Application Center, Sakarya University, in 2011. His current research interests include software system development, cloud manufacturing, and blockchain.

AHMET ÖZMEN received the B.S. degree in electronics and communication engineering from the Department of Istanbul Technical University, Istanbul, Turkey, in 1987, and the M.S. and Ph.D. degrees from the Electrical Engineering Department, University of Kentucky, Lexington, KY, USA, in 1996 and 2000, respectively. He is currently with the Department of Computer Engineering, Sakarya University. His main research activity is related to data analysis and software system development.