Fast Flux Density Distribution Simulation of Central Receiver System on GPU

Caitou He[a], Jieqing Feng[a], Yuhong Zhao[b,*]

[a] *State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou, 310058, China*
[b] *Institute of Industrial Process Control, College of Control Science and Engineering, Zhejiang University, Hangzhou, 310027, China*

Abstract

The simulation of the light flux density distribution on a receiver plays an important role in energy estimation, design and optimization of a heliostat field and the focusing strategy for a central receiver system (CRS). However, this simulation is a time-consuming procedure. In this paper, we propose a fast simulation method that fully exploits the tremendous rendering and parallel computing capacities of contemporary graphics processing units (GPUs). First, an auxiliary spatial data structure is employed to organize the heliostats in the field, and a parallel light beam traversal algorithm is designed and performed on the GPU to determine the shadowing and blocking heliostats for each reference heliostat. Then, the flux spot reflected by each heliostat on the receiver is computed using the HFLCAL model and accumulated for the final flux density distribution. Both the computing stage and accumulation stage are accomplished via GPU rendering pipeline. The proposed method is verified by taking the PS10 power plant as an example. Because this method considers both shadowing and blocking effects, the simulation results are consistent with those in the official report. Due to its high efficiency, the proposed method has potential applications in CRS design and optimization.

Keywords: Flux Distribution Simulation, Central Receiver System, Heliostat Field, Rendering Pipeline, DirectCompute, GPGPU

1. Introduction

Solar thermal power plants with central receiver systems (CRSs) (Vant-Hull and Hildebrandt, 1976) are attracting increasing attention from both the academic and industrial communities. In this system, hundreds or thousands of heliostats reflect sunlight and focus this sunlight onto the receiver to heat a working medium in order to generate electricity. When designing a CRS, many

*Corresponding author
Email addresses: 11221024@zju.edu.cn (Caitou He), jqfeng@cad.zju.edu.cn (Jieqing Feng), yhzhao@iipc.zju.edu.cn (Yuhong Zhao)

stages and factors influence the electrical generation efficiency. Among these factors, the simulation of the flux distribution on the receiver surface is a fundamental problem (Biggs and Vittitoe, 1976).

There are two main types of flux distribution simulation methods, i.e., the ray tracing approach and the analytical approach. Based on these methods, many practical simulation tools have been developed and released, including SolTrace by NREL (Wendelin, 2003), STRAL by DLR (German Aerospace Center) (Belhomme et al., 2009), and TieSOL by Tietronix (Izygon et al., 2011). Among these methods, accuracy and efficiency are always contradictory. The ray tracing approach computes the flux distribution on a receiver by tracing millions of rays or more between the heliostat field and receiver. Obviously, its computational complexity is very high. The analytical approach efficiently evaluates the flux using an empirical analytical formula. Shadowing and blocking effects are difficult to fully take into account, which is one of the reasons that the analytical approach is less accurate than the ray tracing approach.

In this paper, a fast and accurate analytical simulation approach is proposed, that considers both shadowing and blocking effects. The main idea is to convert the computation of the flux spot on the receiver surface contributed by each heliostat into a rasterization procedure through rendering pipeline on graphics processing units (GPUs), i.e., "drawing" the flux spot on the receiver surface, as shown in Figure 1. The flux accumulation is accomplished via the alpha blending operation (Wallace, 1981) in the rendering pipeline. Furthermore, the shadowing and blocking effects are processed in parallel on the GPU. As a result, the proposed method can accurately simulate the flux density of a field containing hundreds of heliostats within dozens of milliseconds.

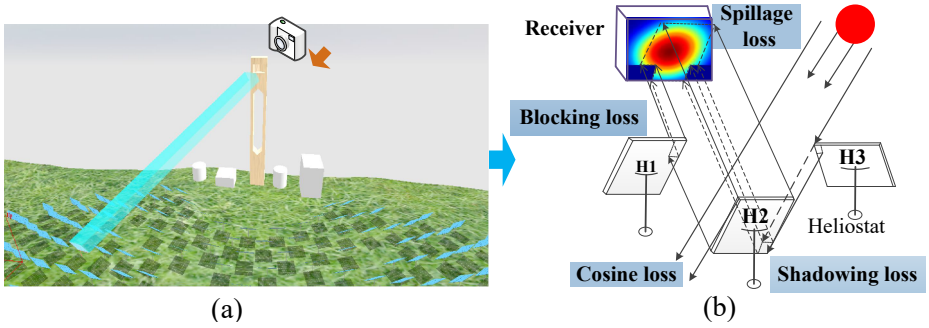

Figure 1: The flux simulation is performed as a graphics rendering procedure: (a) the receiver surface is treated as the rendering target, and (b) the flux density distribution on the receiver surface cast by the parabolic heliostat H2 is shown, where the flux at the corner is eliminated due to shadowing and blocking effects caused by H3 and H1, respectively.

The rest of the paper is organized as follows: The related work is briefly introduced in Section 2. Some relevant preliminary knowledge is presented in Section 3. The proposed fast flux density distribution simulation method using a GPU is described in detail in Section 4. Validation, experiments and discussion are given in Section 5. Finally, in Section 6, the conclusions are drawn, and

future work is proposed.

2. Related Work

2.1. Simulation of Flux Distribution on the Receiver

Basically, two types of flux distribution simulation methods exist: the ray tracing method and the analytical method. Instead of simulation, the flux mapping method measures the flux distribution on the receiver directly; this method is based on the measurement equipment and the image processing techniques. Garcia et al. (2008) gave a detailed comparison between the ray tracing approach and the analytical approach and also provided an overview of relevant simulation tools. Bode and Gauche (2012), from the South African Solar Thermal Energy Research Institute, gave another survey emphasizing the progress of the simulation tools.

Ray Tracing Method Ray tracing is an image synthesis algorithm in computer graphics that simulates the light reflection, refraction and scattering among a light source, objects, and a viewer (Glassner, 1989). This method can naturally take shadowing and blocking effects into account for the flux simulation on the receiver and can be further classified into three categories according to where the rays are generated: the forward ray tracing method, the bidirectional ray tracing method, and the reverse ray tracing method.

Wendelin (2003) developed a forward ray tracing simulation tool based on Monte Carlo method named SolTrace. This tool is capable of modeling general geometries of optical systems with the normal distribution captured via the Video Scanning Hartmann Test (VSHOT). However, as the author indicated, the simulation speed needs to be improved.

Belhomme et al. (2009) generated the rays directly on the reflective surface of the heliostat and accelerated the bidirectional ray tracing process by exploiting SIMD architecture on a multi-core CPU. Later, they achieved ray tracing speed of more than 60 million rays per second on an 8 core PC(Ahlbrink et al., 2012). It is worth mentioning that they employed two methods to model the surface errors of a heliostat mirror, i.e., the common statistical approach based on the Gaussian distribution of surface normal vectors (Belhomme et al., 2009), and the approach using measured surface normal vectors of high resolution (Ulmer et al., 2011). Izygon et al. (2011) achieved state-of-the-art ray tracing simulation speed by taking advantage of the parallel computing capacities of contemporary GPUs. Tracing 100 million rays took approximately 887 ms using 3 GTX 570 GPUs.

In the reverse ray tracing method (Chiesi et al., 2013; Pancotti, 2007), rays are emitted from the receiver surface rather than the energy source. They are traced from the receiver to the heliostat and finally to the sun. The flux density is obtained by integrating over the corresponding area of the sun model, which is modeled as a Lambertian surface. Chiesi et al. (2013) achieved 52 times faster speed with heterogeneous systems (8 CPUs and two NVIDIA GTX 570 and one GTX 480 graphics cards) compared to multi-core CPUs.

Analytical Method In a mathematical sense, the analytical method is the convolution of the sun brightness profile with the reflective surface slope errors (Dellin, 1979; Walzel et al., 1977). An analytical solution to this problem is not known so far; instead, numerical approximation approaches (Biggs and Vittitoe, 1976; Vittitoe and Biggs, 1981) have been proposed. The HFLCAL model is a simplification of the convolution approach, and was initially designed to model the flux distribution on the receiver surface with a circular normal distribution (Schwarzbözl et al., 2009). Many researchers have used this model for comparisons with measured data (Collado, 2010) and heliostat aiming strategies (Besarati and Goswami, 2014; Salomé et al., 2013).

García et al. (2015) revised the HFLCAL model by modeling the flux distribution on the heliostat reflecting surface rather than on the receiver and then mapping the two-dimensional Gaussian distribution to the receiver through homography transformation (García et al., 2015). As a result, the revised model gives a more precise representation of the actual flux spot on the receiver.Huang and Xu (2014) employed the Gauss-Legendre integration method to calculate the intercept factor of each heliostat on the receiver surface, thus obtaining a coarse numerical estimation of the received energy.

Flux Mapping Method The flux mapping method is not a simulation method but is instead a measurement method. First, it calibrates the photographic image data of the receiver surface and then establishes a look-up table between the pixel value of the image and the flux intensity. Finally, it obtains the flux distribution map on the receiver panel. This method can be further classified into two types, namely, the direct measurement based method (Blackmon, 1985) and the indirect evaluation based method (Saade et al., 2014; Ulmer et al., 2002). The coverage range and precision are two problems of these methods (Ho and Khalsa, 2012). Obviously, the flux mapping method is not suitable for energy forecasting because it requires an online image capture of the receiver surface.

2.2. Shadowing and Blocking Effects

In a heliostat field, the shadowing effect refers to the incoming sunlight occluded by adjacent heliostats, while the blocking effect refers to reflected sunlight that is lost on the way to the receiver, as shown in Figure 1(b). These are important factors for precisely evaluating the flux distribution on the receiver, especially for a large-scale heliostat field. In the past few decades, many elaborate methods have been proposed to accelerate the processing of these effects.

Belhomme et al. (2009) employed a hierarchical spatial structure and the separating axis theorem (SAT) to organize the heliostats, thus accelerating the shadowing and blocking heliostat identification. As a simple solution, Izygon et al. (2011) suggested keeping two lists of potential shadowing and blocking heliostats for each heliostat. Each list has a fixed number of heliostats, which are identified from the neighboring heliostats. Obviously, the estimation is not accurate, especially when the sun angle is small, i.e., during early morning or late afternoon. Based on the observation that a heliostat will not cast a shadow to those heliostats ahead of it relative to the sun, Besarati and Goswami (2014)

proposed partitioning the neighbors of the current heliostat into two parts and considering only the front part in relation to the sun or the receiver. Obviously, this is still a conservative estimation that takes many irrelevant heliostats into account.

From the perspective of computer graphics, both the shadowing effect and the blocking effect can be regarded as a hidden surface removal problem (Sutherland et al., 1974), which has been a fundamental but well-investigated problem in computer graphics over the past few decades, e.g., in the hierarchical z-buffer algorithm (Greene et al., 1993). The idea of decreasing computational complexity is to exploit spatial coherence in the scene. Accordingly, there are two main spatial partition approaches: the adaptive approach represented by octree (Glassner, 1984) and the uniform grid partition (Fujimoto et al., 1986). The former is top-down, which suits scenes where the objects are irregularly distributed. Conversely, the latter suits scenes in which the objects are uniformly distributed. The accompanying ray traversal algorithm was elaborately designed to efficiently find potential hitting objects (Amanatides and Woo, 1987).

2.3. Atmospheric Attenuation Model

Some empirical atmospheric attenuation models have been proposed that describe energy attenuation travelling from each heliostat to the receiver. The model proposed by Leary et al. (1979) is adopted in our method:

$$\eta_{aa} = \begin{cases} 0.99331 - 0.0001176 * d + 1.97 * 10^{-8} * d^2 & d \leq 1000m \\ \exp(-0.0001106 * d) & d \geq 1000m \end{cases} \tag{1}$$

where d is measured in meters. Noone et al. (2012) noted that the difference among atmospheric attenuation models is small and not as sensitive as other effects, e.g., shadowing effect, blocking effect, cosine loss and spillage loss. If possible, we strongly recommend that the local atmospheric attenuation model regressed by measured data be adopted.

3. Preliminary Knowledge

In this section, we briefly introduce some preliminary knowledge underlying our modeling.

3.1. The Coordinate Systems

In the proposed simulation, three Cartesian coordinate systems are defined to describe a CRS hierarchically. The global coordinate system **XYZ** is used for the heliostat field, the local heliostat coordinate system **UNV** for each heliostat, and the local planar coordinate system $\{\mathbf{S}i\mathbf{T}i\}$ (i=0,1,2,...n-1, where n is the panel count of the receiver surface) for the receiver surface, as shown in Figure 2. All the coordinate systems are left-handed to match the Direct3D's convention (Coordinate Systems (Direct3D)).

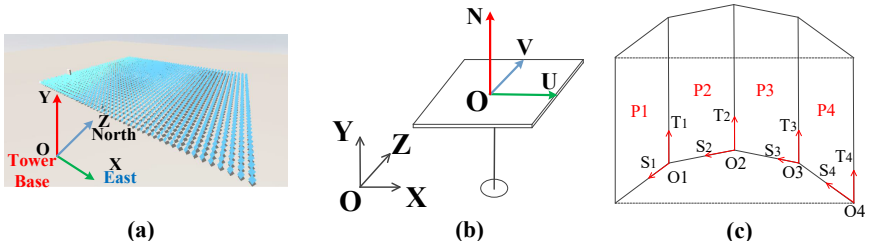

Figure 2: Three coordinate systems: (a) global coordinate system for the heliostat field, (b) local coordinate system for each heliostat and (c) local coordinate system for the receiver panel.

3.2. Energy Model

The revised HFLCAL model (García et al., 2015) is adopted in our method to describe the flux spot reflected by each heliostat. The flux density on the receiver surface is modelled in two steps.

First, a circular Gaussian distribution is used to model the flux density in w/m^2 at point $P(u, v)$ on the heliostat, which is described in the heliostat local coordinate system

$$\mathrm{F}(u, v) = \frac{P_h}{2\pi {\sigma_{HF}}^2} e^{-\frac{(u^2+v^2)}{2{\sigma_{HF}}^2}} \tag{2}$$

where P_h is the total energy reflected by the heliostat. P_h is determined by solar DNI (I_D) at the moment,heliostat surface area (S_H), heliostat reflectivity (ρ) and heliostat surface cosine loss ($\cos w$) as follows:

$$P_h = I_D * (S_H * \cos w) * \rho \tag{3}$$

For a parabolic reflection surface, the cosine loss can be approximated by taking the $\cos w$ value at the heliostat center. σ_{HF} is the effective deviation of the Gaussian model, which approximates the flux spreading effect contributed by four factors, i.e., the sun shape, the mirror slope error, the astigmatic error and the tracking error (Besarati et al., 2014; Schwarzbözl et al., 2009).

Second, the two dimensional Gaussian distribution is projected to the receiver surface through homography transformation. Supposing the point $P(u, v)$ is mapped to $P'(s, t)$ of the receiver surface, the flux density $F'(s, t)$ at P' is calculated as:

$$F'(s, t) = F(u, v) * \eta_{aa} * P_{abs} \tag{4}$$

where η_{aa} is the atmospheric attenuation effect, and P_{abs} is receiver surface absorptivity factor. P_{abs} is assigned to be 0.9 according to Izygon et al. (2011).

3.3. Rendering Pipeline and DirectCompute on GPU

The rendering pipeline (Graphics Pipeline) refers to a sequence of procedures used to create a 2D image representation of a 3D scene. Many algorithms are

involved in achieving this goal, including the z-buffering algorithm and the alpha blending algorithm and are currently implemented on special hardware, i.e., GPUs.

The z-buffering algorithm for hidden surface removal (Sutherland et al., 1974) and the alpha blending operation (Wallace, 1981) for accumulation are two hardware-supported procedures in rendering APIs, e.g., DirectX. They are executed sequentially as follows: The value of a pixel is added to the accumulation buffer via alpha blending if and only if it passes the depth test, which means that it is closer to the viewpoint than the corresponding pixel in the accumulation buffer. We exploit this mechanism to efficiently eliminate shadowing and blocking effects and simultaneously achieve energy accumulation for all of the heliostats in a single rendering pass.

Initially, GPUs were designed as special-purpose hardware to efficiently perform a graphics rendering pipeline. Gradually, they evolved to become a general-purpose hardware (GPGPU) that performs not only graphics task but also parallel computing tasks. DirectCompute (ComputeShader Overview) is an application programming interface for parallel computing on GPUs. In our simulation, this technique is employed to detect shadowing and blocking heliostats for each heliostat in parallel.

4. GPU-based Simulation of Flux Density Distribution on Receiver

The simulation process contains three steps, as shown in Figure 3. First, for a given time, each heliostat is adjusted to redirect the incident sunlight to the target according to various parameters, such as sun direction, heliostat location, and target position. Second, uniform spatial grid encapsulating the heliostat field is built. A light beam traversal algorithm for spatial uniform grid is designed and performed in parallel to efficiently detect the potential shadowing and blocking heliostats for each heliostat. Finally, the flux spot on each panel of the receiver surface contributed by each heliostat is obtained through rasterization and z-buffering algorithms, taking the shadowing and blocking effects into account, and is then accumulated through the alpha blending operation.

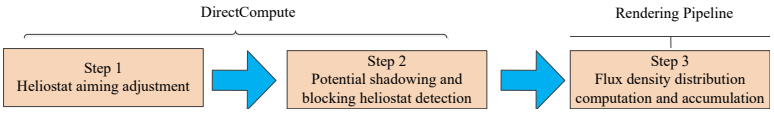

Figure 3: The three steps of the simulation process.

The first two steps are performed in parallel through DirectCompute on the GPU, with one thread taking charge of one heliostat. The last step fully exploits the tremendous parallel computing capacities of GPU rendering pipeline. The intermediate data transfer among different steps always resides in the GPU structured buffer during the simulation process. Thus, data IO overhead between the CPU and GPU is avoided.

4.1. Heliostat Aiming Adjustment

The purpose of the aiming adjustment is to track the sunlight so that the heliostat always reflects the incident sunlight to the receiver target, which is generally the center of receiving surface. In the initial state, the local coordinate system **UNV** of each heliostat is consistent with the global coordinate system **XYZ** other than a translation of their origins. In this paper, we will not address the aiming strategy problem due to the length of the paper. Moreover, tracking error can be easily added by imposing a normal disturbance on the theoretical heliostat normal (Izygon et al., 2011).

Currently, there are two main types of sun tracking models, namely, spinning-elevation type and azimuth-elevation type (Chen et al., 2004), which are suitable for a moving target (e.g., parabolic dish) and a fixed target (e.g., CRS), respectively. Therefore, the azimuth-elevation heliostat tracking model is adopted in our simulation. Each heliostat has two degrees of freedom in this tracking model, i.e., azimuth rotation and elevation rotation, as shown in Figure 4. Let $\mathbf{U'N'V'}$ be the local heliostat coordinate system for a heliostat after adjustment.

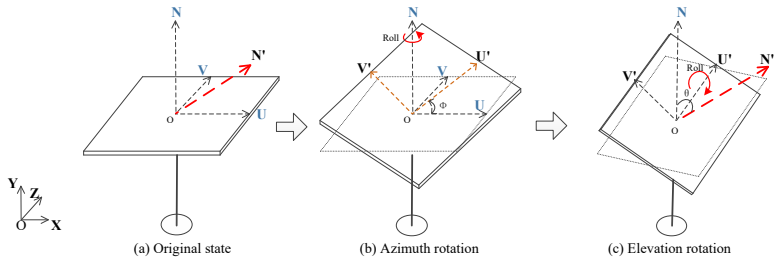

Figure 4: Rotations of a tracking heliostat: (a) initial state, (b) azimuth rotation, and (c) elevation rotation.

The normal $\mathbf{N'}$ of the adjusted heliostat can be deduced by the law of reflection using the following formula:

$$\mathbf{N'} = \frac{\mathbf{s} + \mathbf{r}}{\|\mathbf{s} + \mathbf{r}\|} \tag{5}$$

where $\mathbf{s}$ and $\mathbf{r}$ are normalized vectors of incident light and reflection light respectively, and $\mathbf{s}$ can be determined by the underlying geographical coordinates and solar time (Duffie et al., 2013). Then, the axis $\mathbf{U'}$ is determined by:

$$\mathbf{U'} = \mathbf{N'} \times \mathbf{N} \tag{6}$$

Finally the $\mathbf{V'}$ axis is determined so that $\mathbf{U'N'V'}$ forms a left-handed coordinate system.

4.2. Detection of Shadow and Block

To accelerate detection, the spatial coherence of the heliostats in the field should be exploited, which can be expressed via an auxiliary data structure,

i.e., a uniform spatial grid. Further, a light beam traversal algorithm is designed based on the uniform spatial grid structure, that can efficiently search the potential shadowing and blocking heliostats in parallel on GPU. To process the shadowing and blocking effects uniformly, each shadowing heliostat is converted into an equivalent virtual blocking heliostat.

4.2.1. Uniform Spatial Grid Construction

A uniform spatial grid encapsulating the heliostat field is constructed offline following two steps. First, an axis-aligned bounding box (AABB) is built for each heliostat in the global coordinate system. The size of the AABB edge equals the length of the heliostat diagonal, as shown in Figure 5(a). Thus, a heliostat at any time will be contained within its conservative AABB. Second, an axis-aligned bounding box for the heliostat field in the global coordinate system is uniformly subdivided along the x- and z-directions. A uniform spatial grid is generated, where the size of each grid cell is determined tactically, as shown in Figure 5(b). The purpose of using an auxiliary data structure is to exploit the spatial coherence of heliostats in the field. Each cell of the grid can be regarded as a proxy or index of a heliostat. Thus, a grid cell that contains a heliostat is theoretically better (few heliostats straddle the boundary of two or more grid cells). For a uniformly distributed rectangular heliostat field, the above assertion can be achieved straightforwardly. Otherwise, the grid cell's size will be empirically determined by the user. For example, in PS10, as shown in Figure 6(a), which is a radially staggered distributed field, the dimensions of the cells are set as the heliostat diagonal in our test. Under this condition, a grid cell will be the index or proxy of several heliostats, whose AABBs intersect with the cell.

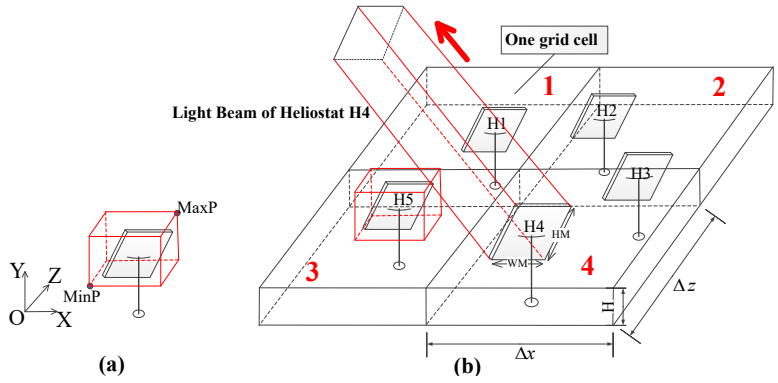

Figure 5: The auxiliary grid data structure and modeling of the light beam received or reflected by one heliostat: (a) axis-aligned bounding box (red) of a heliostat and (b) uniform spatial grid of the heliostat field. The light beam of the heliostat is modeled as a parallelepiped.

Figure 6 shows two types of heliostat field layout and their uniform spatial grid. One is the famous PS10 heliostat field. Because the heliostats are installed

on a slope, the uniform grid consists of two layers along the vertical direction. The other is a uniformly distributed rectangular field containing 1800 heliostats.

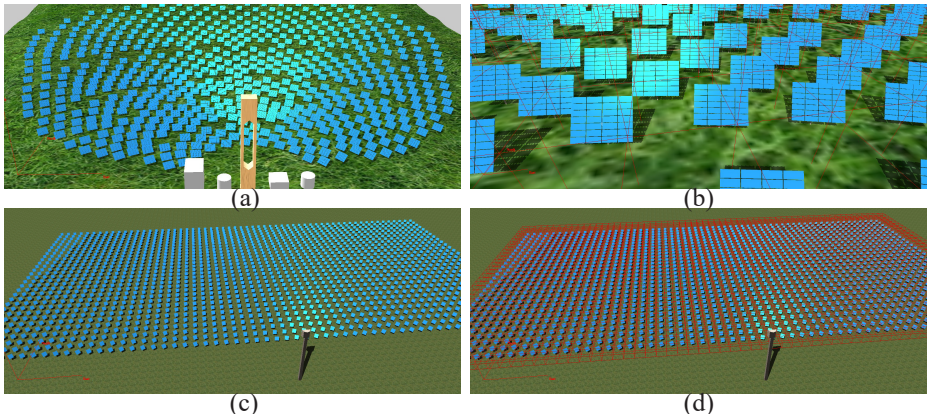

Figure 6: Two popular types of heliostat field layouts and their uniform spatial partition grids, visualized in the form of wireframes: (a) PS10, a radially staggered distributed field, (b) zoom-in of the PS10 field and the related grid cells, (c) a uniformly distributed rectangular field, and (d) the spatial partition grid of the field.

4.2.2. Light Beam Traversal Algorithm

Inspired by the incremental grid traversal algorithm for ray tracing (known as the 3DDDA Algorithm) (Amanatides and Woo, 1987; Fujimoto et al., 1986), we develop a robust light beam traversal algorithm to search the grid cells intersected by incident light beams or reflected light beams and then determine the potential shadowing or blocking heliostats.

Fast Beam-Grid Cell Intersection Test. A light beam can be represented as a parallelepiped consisting of five planes (the heliostat reflecting surface plus four side planes), which encompass the volume translated by the heliostat surface along light beam direction **r** to the infinity, as shown in Figure 5(b). Without loss of generality, we assume that the face normals of the five planes point to the inside of the light beam volume, as shown in Figure 7

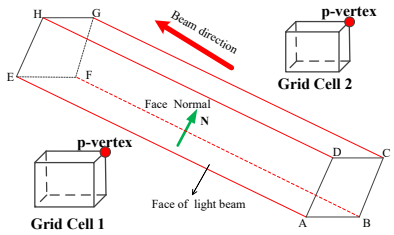

Figure 7: The face normals of the five planes of the light beam point to the inside of the light beam volume. The p-vertex of grid cell 1 is on the back side of one beam plane (ABFE), so it will be rejected by the beam.

Inspired by the view frustum culling algorithm for bounding boxes (Assarsson and Mller, 2000), the idea of the proposed fast beam-grid cell intersection test is to reject the cells by each face of the light beam as early as possible, where the light beam can be regarded as the view frustum. For each face plane of the beam, a cell vertex is identified, namely, a p-vertex, the one vertex among the eight vertices of the grid cell that has the greatest signed distance from the considered plane. Then, the cell can be safely rejected by the beam if the p-vertex is on the back side of the plane, such as Grid Cell 1 in Figure 7. If a cell cannot be rejected by the five face planes of a beam, then the beam will intersect with the cell. Because the beams are independent with each other, the proposed beam-grid cell intersection algorithm is well adapted to the SIMD architecture of GPU and can be implemented in parallel.

Beam Traversal in the Uniform Grid. A traversal algorithm is designed to efficiently find the grid cells intersecting with a light beam, using the beam-grid cell intersection test algorithm above. The basic idea of the algorithm is to scan the grid cells layer by layer, then row by row, starting at the first cell where the beam is emitted, as shown in Figure 8. If a heliostat crosses several grid cells, the first cell along the counter light beam direction is chosen as the first one to be scanned. The scanning for the current layer terminates if the intersected cells are all surrounded by non-intersected cells along the beam's heading direction, as shown in Figure 8(g).

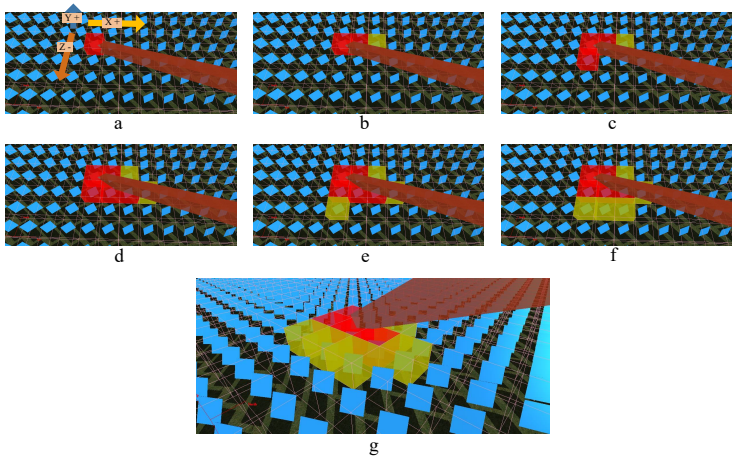

Figure 8: Illustration of light beam traversal algorithm in the uniform grid. The red and yellow grid cells indicate intersecting and non-intersecting cells, respectively.

Beam-Heliostat Intersection Test. After the intersected grid cells are determined via the above beam traversal algorithm, the heliostats in the intersected cells are checked to determine whether they are intersected by the light beam. The intersection test is accomplished by checking the four corners of the heliostat against the five faces of the beam, which is similar to the beam-grid cell intersection test above. If all the corners of the heliostat are on the backside

of one beam face, the heliostat is separated from the beam. Otherwise, the heliostat intersects the beam, which will cause a shadowing or blocking effect.

4.2.3. Implementation Details on GPU

To implement the beam grid traversal algorithm on GPU with DirectCompute API, some data structures are elaborately designed, i.e., the uniform grid cells containing heliostat partition information are first mapped to the GPU memory. To this end, the parameters of the uniform grid structure, such as cells' dimensions, are packed and passed to the GPU through a constant buffer. The heliostat index information of each cell is organized serially as a heliostat index array. The index of each grid cell ends with -1. Another array keeps the starting index of each cell in the heliostat index array in sequence. Addressing the heliostat list information of a specific cell consists of two steps: first, fetching the starting heliostat index location of the cell and second, reading the index continuously in the heliostat index array until -1 is met. These two arrays can be perfectly mapped to the GPU memory through structure buffers (Figure 9).

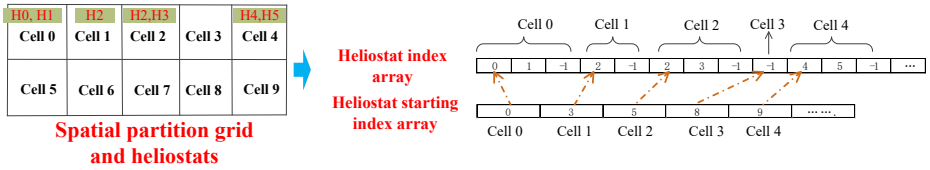

Figure 9: The data of the uniform grid cells are organized contiguously and passed to the GPU memory through structure buffers.

4.3. Shadow to Block

In general, the shadowing and blocking effects are processed separately in the flux density distribution simulation. The net results of flux loss on the receiver caused by shadowing effects or blocking effects are the same. That is to say, we cannot distinguish between the two effects in the final result. This observation means that the two effects can be processed uniformly by converting the shadowing effect into an equivalent imaginary blocking effect, as shown in Figure 10. In this way, the shadowing heliostats can be converted into blocking heliostats so that the two flux loss effects can be processed uniformly. A blocking or imaginary blocking heliostat is generally refer to as an occluder.

4.4. Flux Simulation

In this section, we will introduce the proposed flux simulation method based on GPU rendering pipeline in detail. At the beginning, parameters of all of the heliostats (i.e., position, orientation, geometry, reflectivity, etc.) are packed and streamed into the rendering pipeline at the input-assembler stage. First, the flux density distribution on the receiver surface contributed by each heliostat is computed through a rasterization procedure based on the revised HFLCAL

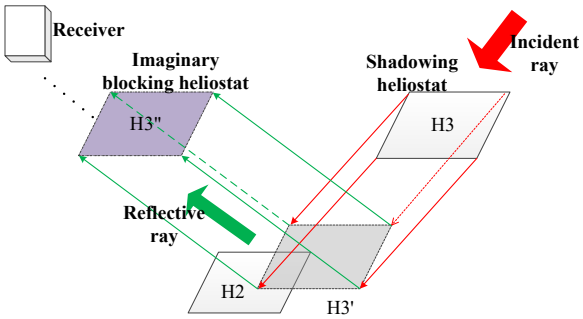

Figure 10: Conversion of shadowing heliostat H3 into imaginary blocking heliostat H3". H3" is obtained by offsetting the shadow silhouette along the reflective ray direction. The distance between H2 and H3 equals that between H2 and H3".

model. Then, all of the flux density distributions are accumulated through an alpha-blending operation, and the blocking effects are eliminated through z-buffering algorithm. The flux accumulation and blocking elimination process for all of the heliostats are accomplished in a single rendering pass, which is therefore extremely efficient.

4.4.1. Render Target Mapping

In our simulation, the receiver is regarded as the rendering target, i.e., a frame buffer, and the flux spot is the synthesized image on the rendering target, where the pixel value is the flux density. Each pixel in the frame buffer corresponds to a physical area on the receiver surface, and vice versa. If the receiver contains several physical panels, each panel will correspond to a rendering target. The correspondence between each receiver panel and the rendering target can be established using a $2D$ affine transformation. For example, if the size of the receiving panel is $w \times h$ (m^2), the resolution of the frame buffer can be set as $pw \times ph$. The unit of p is pixel width/m, representing the resolution parameter from the receiver panel to the render target. In theory, the larger p is, the higher the simulation accuracy is, and the more time the simulation procedure needs.

4.4.2. Flux Density Distribution on the Receiver via Flux Mapping

If the receiver contains several panels, the following procedures will be performed for each panel. The heliostat is projected onto the panel of the receiver surface along reflection direction, as shown in Figure 11. Based on the projected heliostat corners, a mapping between the projected heliostat and the heliostat local coordinate system is deduced. Then, the flux density of each pixel on the render target is calculated according to the process described in section 3.2. This procedure is performed in parallel via pixel shader. We call this process flux mapping.

If the heliostat has some occluders, these occluders should also be projected onto the receiver panel. Intuitively, the projection areas on the receiver panel

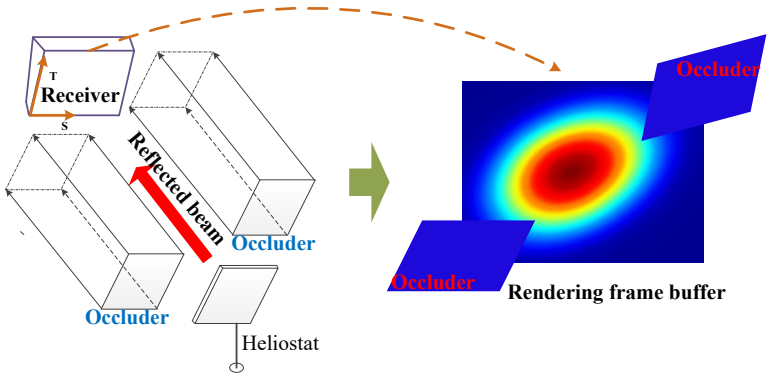

Figure 11: The heliostat and its occluders are projected onto the receiver along the reflection direction, and then assigned using the flux mapping procedure.

will receive no flux spot. However, the flux density on the receiver along the silhouette of the blocked area should be smooth transition from lightness to darkness , rather than sharp excisions, which is essentially due to the projection of the sunshape. The problem can be solved by improving the energy model and is out of the range of our problem. In our solution to the blocking effects, the projection area of occluder is simply assigned as 0, i.e., no flux contribution. Thus, this process and the following rasterization can also be implemented in the rendering pipeline in parallel.

4.4.3. Streaming Data in the Rendering Pipeline

To achieve the flux accumulation and blocking effects elimination in one rendering pass, the depth values of the projection of the occluders and heliostats are elaborately assigned according to the following rules:

$$d_p = \begin{cases} 1.0 - \frac{2*ID+2}{2*N+1}, & \text{the occluders of the heliostat indexed ID} \\ 1.0 - \frac{2*ID+1}{2*N+1} & \text{the heliostat indexed ID} \end{cases} \tag{7}$$

where N is the number of heliostats in the field, ID is the heliostat index and $0 \leq ID \leq N-1$. In the z-buffering algorithm, the smaller the depth is, the closer it is to the render target. Thus, in this setting, the occluders are always in front of the corresponding heliostat. Furthermore, the heliostat and its occluders are strictly organized in the pipeline as shown in Figure 12.

4.4.4. Block Elimination and Flux Accumulation

Blocking effect elimination and flux accumulation are realized using the z-buffering and alpha blending functions in the rendering pipeline. First, the depth map is initialized by assigning the maximum depth value of 1.0. The flux spots and their occluders in terms of stream data are rendered to the frame buffer, which successively undergo the depth test and alpha blending. Once a fragment passes the depth test, its flux density value is directly added to

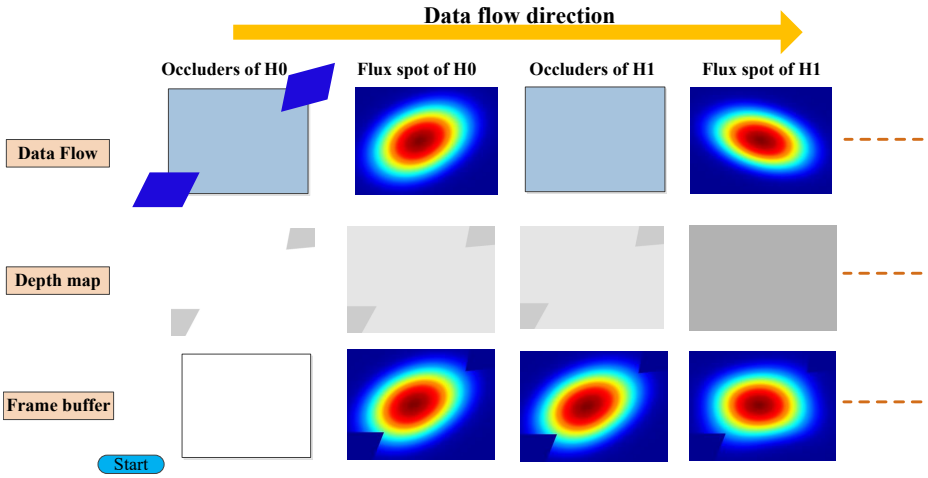

Figure 12: Process of flux accumulation and blocking effects elimination regarding the steaming data. The depth buffer keeps the closest depth value per pixel, while the frame buffer records the accumulated flux density.

the frame buffer via alpha blending; otherwise, it will be discarded. In this way, the fragment of flux spot in the blocking area are automatically eliminated before accumulation. Since the flux density value of the occluders equals zero, the frame buffer remains unchanged when they are accumulated to the frame buffer. The evolution of the frame buffer and the attached depth buffer during this process is illustrated in Figure 12. The frame buffer keeps the accumulating result, while the depth map records the nearest depth value (the darker the map is, the smaller the depth value is).

4.5. Total Energy

When the rendering procedure terminates, the accumulated flux density distribution on the receiver is obtained. Each pixel in the frame buffer represents the solar energy density on the receiver surface in W/m^2. The total flux on the receiver surface can be computed through integration over the frame buffers.

$$E_{total} = \sum\nolimits_{NP} E_i * S_{pixel} \tag{8}$$

where NP is the number of pixels in all frame buffers and S_{pixel} equals the receiver surface area per pixel (m^2/pixel).

5. Validation, Experiments and Discussions

5.1. Validation of Simulation Method for PS10

The proposed simulation method has been implemented on a desktop PC with an Intel Core (TM) i5-3450 @3.10 GHz CPU and an NVIDIA GeForce

GTX 670 GPU. The OS is Windows 10, and the graphics API is Direct3D 11.0. We tested the proposed method for PS10, which is a famous commercial solar power tower plant that has some measured results published in an official technical report (Osuna et al., 2004). We constructed the PS10 heliostat field, receiver panels and its surrounding terrain via reverse engineering according to the information in that technical report, where 624 curved heliostats of 121 m^2 each are arranged with a radially staggered pattern, as shown in Figure 6(a).

The PS10 has a cavity receiver consisting of four panels (Figure 2(c)). Each panel is 5.36 m $\times$ 12.0 m. These panels are arranged in a semi-cylinder with a radius of 7.0 m. All of the heliostats aim at the center of the aperture, which is approximately 100.5 m above the ground.

The simulation was performed using the revised HFLCAL model and the atmospheric attenuation model in Equation 1. The HFLCAL model is perfectly suitable for PS10 because the heliostat in PS10 has a parabolic reflecting surface. σ_{HF} in Equation 2 was set as 1.80 according to our experiments. Other environmental parameters such DNI and the sunlight direction were specified according to the technical report.

Figure 13 shows the graphical comparison between our flux simulation results and the official results in the technical report at two times. The numerical comparison among the official data, the results by Chiesi et al. (2013) and our results is given in Table 1. These results show that the proposed simulation is very consistent with the ground truth. Furthermore, the simulation time for the PS10 heliostat field with the resolution parameter of p=40 pixels width/m is approximately 39.2 milliseconds. The energy loss due to the shadowing and blocking effects at 12:00 on March 21st is 565 kW, accounting for as much as one percent of the total energy received.

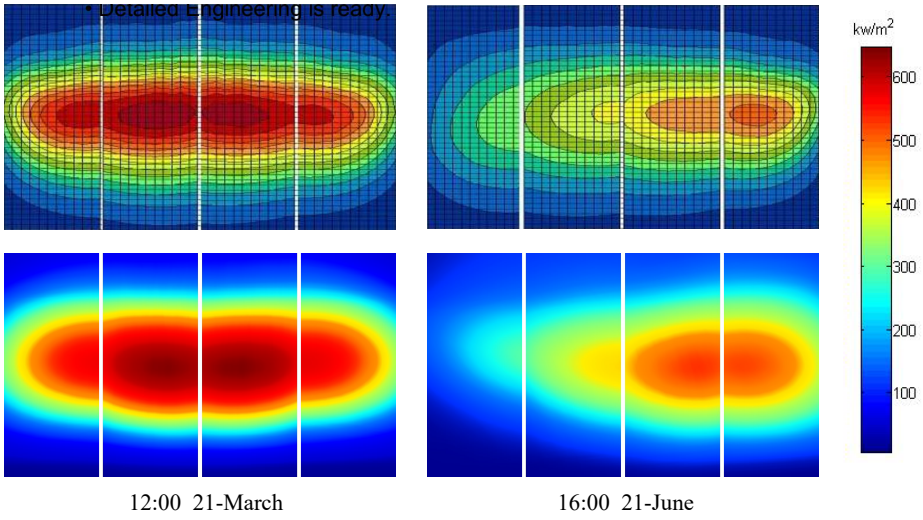

Figure 13: Graphical comparisons between the simulated flux density distribution (bottom) and the official result (top) for two specific times.

Table 1: Numerical comparison among the official results, the results from Chiesi et al. (2013) and our simulation results.

Net Power(kW) Data / Time	Official data	Chiesi M. et al. (2013)	Our results
12:00 March 21^{st}	51953.86	51325	52491
16:00 June 21^{st}	34456.32	34403	35119
Peak (kW/m^2) Data / Time	**Official data**	**Chiesi M. et al. (2013)**	**Our results**
12:00 March 21^{st}	644	——	676.5
16:00 June 21^{st}	469	——	449.1

5.2. Scalability of the Simulation Algorithm

In this section, we discuss the scalability of the proposed flux simulation algorithm for both the receiver resolution, i.e., the parameter p (pixel width/m) of the receiver panel, and the heliostat field scale.

First, the impact of the receiver resolution on the PS10 simulation result was studied. The receiver resolution parameter can be adjusted for different applications. Different receiver resolutions mean different sampling granularities of the energy spot, which will affect the accuracy of the simulation in theory. As shown in Table 2, the net power of PS10 tends to converge to a stable state when p is greater than 20. Meanwhile, the simulation time increases quadratically. We chose $p = 40$ pixel width/m as a tradeoff between simulation accuracy and efficiency.

Table 2: The simulated net power of PS10 at 12:00 on March 21^{st} and the simulation time with respect to receiver resolutions.

p	1.0	5.0	20.0	40.0	50.0	80.0	100
Time (ms)	8.1	9.8	24.2	39.2	67.1	155.7	256.3
Net Power (kW)	58037	52946	52342	52491	52540	52352	52303

Second, we studied the scalability of the simulation algorithm for the heliostat field scales. Here, ten synthesized rectangular heliostat fields were generated as examples. The basic parameters of the synthesized scene are listed in Table 3. Five groups of experiments were conducted with receiver resolutions of p=65, 50, 40, 30 and 20.

The runtime statistics are shown in Figure 14. The results show that the simulation time varies approximately linearly with the heliostat number for a specific receiver resolution.

5.3. Performance Comparison with Ray Tracing Methods

In the proposed method, one pixel in the render target can be approximately regarded as a ray in the ray tracing method in the sample rate. In the validation

Table 3: Basic field parameters of the synthesized scene.

Site longitude	Site latitude	Solar azimuth angle	Solar altitude angle
6.23°	37.4°	90°	52°
Receiver position	Receiver type	Receiver width	Receiver length
(0.0, 100.0, 0.0)	One panel	8 m	8 m
Heliostat width	Heliostat length	Field type	
6 m	6 m	Rectangular	

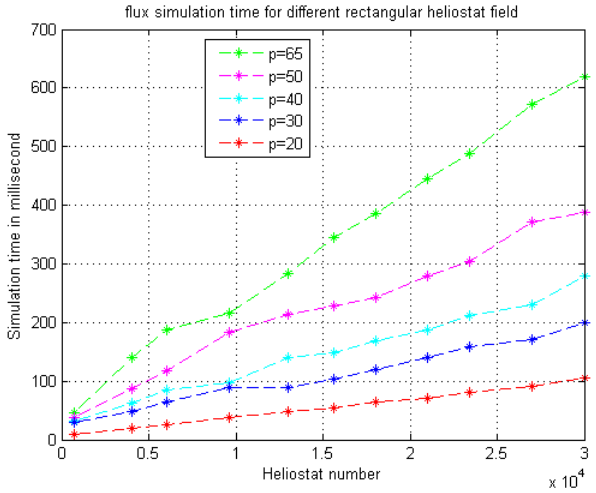

Figure 14: Flux simulation time for different scales of heliostat fields with various p.

for PS10 when p=50, the number of rays is:

$$N = 600 * 268 * 624 * 4 = 401,356,800 \tag{9}$$

while the corresponding run time is approximately 67.1 milliseconds. To our knowledge, TieSOL (Izygon et al., 2011) is the most efficient GPU-based ray tracing software for flux simulation of CRS. As reported by Izygon et al. (2011), tracing 100 million rays took approximately 887 milliseconds on 3 GTX 570 GPUs. Comparably, our method achieves much faster performance. Of course, the simulation accuracy of TieSOL is higher than ours due to the methods adopted.

The proposed method shares many similarities with the reverse ray tracing method in which the receiver surface is discretized. As indicated in the work by Chiesi et al. (2013), they required 4.33 s to simulate a field of 12,000 heliostats. The corresponding discretization schemes of each receiver panel and heliostat are 40×40 and 20×20, respectively, which is approximately equivalent to 7.68 billion rays. In our experiments, as shown in Figure 14, the maximum ray

number appears in the simulation of a field containing 30,000 heliostats and p=65 pixel width/meter. Comparably, the proposed method needs only 0.618 s to trace $65 * 65 * 8 * 8 * 30000 = 8.11$ billion rays.

5.4. Comparison with Analytical Methods

One advantage of analytical methods is the fast evaluation of the field performance. Thus, these methods can be used for heliostat field design and aiming optimization. Because the proposed method is fully implemented on GPU, it is sufficiently fast to fulfil this task. Another experiment was conducted to evaluate the annual performance of the PS10 plant from year 2005 to 2014, with a simulation time interval of 5 minutes. The plant was assumed to run every day in a year, 10 hours per day. The proposed method took approximately 7.5 seconds to simulate for a day and 2698 seconds (0.75 h) for a year. Compared with the work by Huang and Xu (2014), which reports approximately 2.67 hours for simulating a year, the proposed method not only runs much faster but can also provide the instantaneous flux density distribution on the receiver surface.

6. Conclusion

We presented a GPU-based algorithm that is capable of simulating the flux distribution on the receiver for central receiver systems. To exploit the spatial coherence of a heliostat field, a uniform grid structure is built to organize the heliostats. A beam traversal algorithm is designed and implemented in parallel to efficiently detect shadowing and blocking heliostats for each heliostat. The flux spot of each heliostat on the receiver is computed through a rasterization operation in the rendering pipeline. All of the flux spots are accumulated in a single rendering pass, taking the shadowing and blocking effects into account. By exploiting the advantages of the parallel computing capacity and the rendering pipeline of GPUs, the algorithm can achieve an almost real-time simulation for a medium-scale field.

In theory, the proposed simulation framework is applicable to a receiver with an arbitrary shape as long as the receiver surface can be approximated by a series of planar panels, e.g., the commonly used cylindrical receiver. In this case, the computational complexity of the simulation is proportional to the number of panels in the approximation.

The proposed method is independent of flux spot models, whether reflected by planar or parabolic heliostats with multi-facet mirrors. In addition this method is independent of the heliostat field layout, the aiming strategy of each heliostat, and the receiver shape, as long as the configuration of the CRS is given. To the best of our knowledge, an accurate analytical model of flux spot reflected by a planar heliostat remains a challenge, which will be more suitable for current central receiver systems. We would like to investigate this issue in the future. Furthermore, fast simulation is the basis of a series of optimization problems in CRS simulation. Therefore, we will investigate the optimization problems based on our fast simulation method in the future, including aiming strategy, and heliostat field layout.

7. Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 61173128.

8. Appendix:

A list of symbols and variables used in the paper.

Nomenclature

- CRS: Central Receiver System.
- GPU Graphics Processing Unit.
- η_{aa} Atmospheric attenuation factor.
- d Distance from the heliostat to the receiver aiming point.
- P_h Total energy reflected by a heliostat.
- I_D Direct Normal Irradiance.
- S_H Heliostat reflective surface area.
- ρ Heliostat reflectivity.
- $\cos w$ Cosine loss of a heliostat.
- σ_{HF} Effective deviation of the HFLCAL model.
- P_{abs} Receiver surface absorptivity factor.
- $\mathbf{N}'$ The normal of the adjusted heliostat.
- $\mathbf{s}$ Normalized vector of sun direction in the global coordinate system.
- $\mathbf{r}$ Normalized vector of reflection light direction.
- p The resolution parameter from the receiver panel to the render target.
- d_p Virtual depth value of the projection in the rendering pipeline.
- N Number of heliostats.
- ID Heliostat index.
- E_i Flux density value of one pixel in W/m^2.
- S_{pixel} Receiver surface area represented by a pixel.

References

Ahlbrink, N., Belhomme, B., Flesch, R., Maldonado Quinto, D., Rong, A., Schwarzbözl, P., 2012. Stral: Fast ray tracing software with tool coupling capabilities for high-precision simulations of solar thermal power plants, in: Proceedings of the SolarPACES 2012 conference.

Amanatides, J., Woo, A., 1987. A fast voxel traversal algorithm for ray tracing, in: Eurographics', pp. 3–10.

Assarsson, U., Mller, T., 2000. Optimized view frustum culling algorithms for bounding boxes. Journal of Graphics Tools 5, 9–22. doi:10.1080/10867651.2000.10487517.

Belhomme, B., Pitz-Paal, R., Schwarzbözl, P., Ulmer, S., 2009. A new fast ray tracing tool for high-precision simulation of heliostat fields. Journal of Solar Energy Engineering volume 131, 376–385. doi:10.1115/1.3139139.

Besarati, S.M., Goswami, D.Y., 2014. A computationally efficient method for the design of the heliostat field for solar power tower plant. Renewable Energy 69, 226–232. doi:10.1016/j.renene.2014.03.043.

Besarati, S.M., Goswami, D.Y., Stefanakos, E.K., 2014. Optimal heliostat aiming strategy for uniform distribution of heat flux on the receiver of a solar power tower plant. Energy Conversion and Management 84, 234–243. doi:doi:10.1016/j.enconman.2014.04.030.

Biggs, F., Vittitoe, C.N., 1976. The helios model for the optical behavior of reflecting solar concentrators. Sandia National Laboratories Report Report N0.SAND76-0347 .

Blackmon, J.B., 1985. Development and performance of a digital image radiometer for heliostat evaluation at solar one. Journal of Solar Energy Engineering 107, 315–321. doi:10.1115/1.3267699.

Bode, S., J., Gauche, P., 2012. Review of optical software for use in concentrating solar power systems, in: Proceedings Southern African Solar Energy Conference.

Chen, Y.T., Kribus, A., Lim, B.H., Lim, C.S., Chong, K.K., Karni, J., Buck, R., Pfahl, A., Bligh, T.P., 2004. Comparison of two sun tracking methods in the application of a heliostat field. Journal of Solar Energy Engineering 126, 638–644. doi:10.1115/1.1634583.

Chiesi, M., Vanzolini, L., Scarselli, E.F., Guerrieri, R., 2013. Accurate optical model for design and analysis of solar fields based on heterogeneous multicore systems. Renewable Energy 55, 241–251. doi:10.1016/j.renene.2012.12.025.

Collado, F.J., 2010. One-point fitting of the flux density produced by a heliostat. Solar Energy 84, 673–684. doi:10.1016/j.solener.2010.01.019.

ComputeShader Overview, 2016. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/ff476331(v=vs.85).aspx. [Online; accessed 02-September-2016].

Coordinate Systems (Direct3D), 2016. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/bb204853(v=vs.85).aspx. [Online; accessed 02-September-2016].

Dellin, T.A., 1979. An improved hermite expansion calculation of the flux distribution from heliostats. Report No. SAND79-8619 .

Duffie, J.A., Beckman, W.A., Mcgowan, J., 2013. Solar Engineering of Thermal Processes. John Wiley & Sons.

Fujimoto, A., Tanaka, T., Iwata, K., 1986. Arts: Accelerated ray-tracing system. IEEE Computer Graphics & Applications 6, 16–26. doi:10.1109/MCG.1986.276715.

García, L., Burisch, M., Sanchez, M., 2015. Spillage estimation in a heliostats field for solar field optimization. Energy Procedia 69, 1269–1276. doi:10.1016/j.egypro.2015.03.156.

Garcia, P., Ferriere, A., Bezian, J.J., 2008. Codes for solar flux calculation dedicated to central receiver system applications: A comparative review. Solar Energy 82, 189–197. doi:10.1016/j.solener.2007.08.004.

Glassner, A.S., 1984. Space subdivision for fast ray tracing. IEEE Computer Graphics & Applications 4, 15 – 24. doi:10.1109/MCG.1984.6429331.

Glassner, A.S., 1989. An introduction to ray tracing. Elsevier.

Graphics Pipeline, 2016. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/ff476882%28v=vs.85%29.aspx. [Online; accessed 02-September-2016].

Greene, N., Kass, M., Miller, G., 1993. Hierarchical z-buffer visibility, in: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 231–238.

Ho, C.K., Khalsa, S.S., 2012. A photographic flux mapping method for concentrating solar collectors and receivers. Journal of Solar Energy Engineering 134. doi:10.1115/1.4006892.

Huang, W., Xu, Q., 2014. Development of an analytical method and its quick algorithm to calculate the solar energy collected by a heliostat field in a year. Energy Conversion & Management 83, 110–118. doi:10.1016/j.enconman.2014.03.065.

Izygon, M., Armstrong, P., Nilsson, C., Vu, N., 2011. TieSOL–a GPU-based suite of software for central receiver solar power plants, in: SolarPACES Granada, Spain.

Leary, P.L., Hankins, J.D., Leary, P.L., Hankins, J.D., 1979. A User's guide for MIRVAL: a computer code for comparing designs of heliostat-receiver optics for central receiver solar power plants. Technical Report Sandia Laboratories Report, Albuquerque, NM, Report No. SAND77-8280.

Noone, C.J., Torrilhon, M., Mitsos, A., 2012. Heliostat field optimization: A new computationally efficient model and biomimetic layout. Solar Energy 86, 792–803. doi:10.1016/j.solener.2011.12.007.

Osuna, R., Fernandez, V., Romero, S., 2004. PS10: a 11.0-mw solar tower power plant with saturated steam receiver, in: Proceedings of the 12th Solar PACES International Symposium on Concentrated Solar Power and Chemical Energy Technologies, Oaxaca, México.

Pancotti, L., 2007. Optical simulation model for flat mirror concentrators. Solar Energy Materials & Solar Cells 91, 551–559. doi:10.1016/j.solmat.2006.11.007.

Saade, E., Clough, D.E., Weimer, A.W., 2014. Use of image-based direct normal irradiance forecasts in the model predictive control of a solar-thermal reactor. Journal of Solar Energy Engineering 136, 83–99. doi:10.1115/1.4025825.

Salomé, A., Chhel, F., Flamant, G., Ferrire, A., Thiery, F., 2013. Control of the flux distribution on a solar tower receiver using an optimized aiming point strategy: Application to themis solar tower. Solar Energy 94, 352–366.

Schwarzbözl, P., Pitz-Paal, R., Schmitz, M., 2009. Visual HFLCAL - a software tool for layout and optimisation of heliostat fields, in: Proceedings of 15th International SolarPACES Symposium, pp. 15–18.

Sutherland, I.E., Sproull, R.F., Schumacker, R.A., 1974. A characterization of ten hidden-surface algorithms. ACM Computing Surveys (CSUR) 6, 1–55. doi:10.1145/356625.356626.

Ulmer, S., Marz, T., Prahl, C., Reinalter, W., Belhomme, B., 2011. Automated high resolution measurement of heliostat slope errors. Solar Energy 85, 681–687. doi:10.1016/j.solener.2010.01.010.

Ulmer, S., Reinalter, W., Heller, P., Lpfert, E., Martnez, D., 2002. Beam characterization and improvement with a flux mapping system for dish concentrators, in: ASME Solar 2002: International Solar Energy Conference, pp. 285–292.

Vant-Hull, L.L., Hildebrandt, A.F., 1976. Solar thermal power system based on optical transmission. Solar Energy 18, 31–39. doi:10.1016/0038-092X(76)90033-5.

Vittitoe, C.N., Biggs, F., 1981. A user's guide to helios: A computer program for modeling the optical behavior of reflecting solar concentrators. Sandia National Laboratories, Albuquerque, NM, SAND81-1180 .

Wallace, B.A., 1981. Merging and transformation of raster images for cartoon animation, in: ACM SIGGRAPH Computer Graphics, ACM. pp. 253–262. doi:10.1145/800224.806813.

Walzel, M.D., Lipps, F.W., Vant-Hull, L.L., 1977. A solar flux density calculation for a solar tower concentrator using a two-dimensional hermite function expansion. Solar Energy 19, 239–253. doi:10.1016/0038-092X(77)90067-6.

Wendelin, T., 2003. Soltrace: A new optical modeling tool for concentrating solar optics, in: ASME 2003 International Solar Energy Conference, pp. 253–260.