Accelerating Ray-Tracers for Central Receiver Systems using a GPU

Lukas Aldenhoff, Pascal Richter [a)]

RWTH Aachen University, Department of Computer Science, Ahornstr. 55, 52074 Aachen, Germany.

[a)]Corresponding author: pascal.richter@rwth-aachen.de

Abstract. An accurate and computationally fast ray-tracer is the key part for the simulation of the optical irradiation of a heliostat field in a solar central receiver system. Within this work we present how the ray-tracing for central receiver systems can be accelerated while obtaining highly accurate results. The runtime improvement is achieved by an enhanced blocking and shading routine and a parallelized evaluation of the ray-tracer using a graphical processing unit (GPU). Our new *SunFlower* GPU ray-tracer is implemented with the Monte-Carlo methodology such that direct run-time comparisons to other Monte-Carlo ray-tracers are possible. Within a case study we demonstrate that the improved and GPU-parallelized ray-tracer computes up to 99.95% accurate annual simulations of heliostat fields consisting of 8600 heliostats in less than 2.5 seconds.

INTRODUCTION

A meaningful optimization of the heliostat field layout in a central receiver system is a computationally immensely expensive task as high accuracy is paramount. Furthermore, modern heliostat fields constantly grow in number of heliostats which makes the problem harder. Layout optimization typically needs to re-evaluate the annual energy production of the power plant multiple thousand times. Thus, the accurate simulation of annual energy production needs to be done in seconds to finish the optimization within an acceptable time frame. This makes the development of a simulation tool, which is useful for optimization of heliostat fields, a challenging task especially regarding performance and accuracy.

There have been multiple comparative reviews of existing central receiver system ray-tracing tools. To date the focus of these studies has been on the accuracy and the flexibility of the simulations [1, 2]. However, as powerful, flexible and accurate simulation tools are achieved more commonly, the performance of the simulations is quickly becoming the focus.

Some state-of-the-art tools like TieSOL [3], QMCRT [4] and sbpRAY [5] push in this direction of timely and accurate simulation of heliostat field output. The most important innovation in this regard is the utilization of the powerful GPU architecture to accelerate the highly parallel computation. Still, this field of work remains in high demand to allow and improve heliostat field layout optimization even for future central receiver systems with several tens of thousands of heliostats.

This work reintroduces the *SunFlower* [6,7] tool, which through continuous development now offers a GPU adaptation with comparable performance to the above-mentioned tools[1]. To accurately represent optical errors of real-world power plant production, the classical Monte Carlo ray tracing approach is implemented. The tool covers heliostat field layout optimization and allows simulation for any setting of a central receiver system, including every geometry of receiver (e.g., flat plate). We will show that our GPU implementation of the Monte Carlo ray tracer computes moment simulations for power plants with nearly ten thousand heliostats in close to 50 msec, while the accuracy is higher than 99.95%. In combination with our novel annual integration method [8] this makes us able to evaluate annual simulations with accuracy above 99.9% in less than five seconds.

RAY-TRACER

The incorporated ray tracing procedure generally consists of three steps:

1. **Ray generation**: heliostat facets of each heliostat are partitioned into smaller cells. Subsequently a ray is generated for each cell. The ray's power is then scaled to the cell area. Using Monte Carlo sampling we duplicate the generated rays and perturb each instance according to a random evaluation of a horizontal and vertical gaussian distribution to simulate inaccuracies of heliostat surface and heliostat alignment as well as deviations introduced by modelling the sun as a disk.

2. **Blocking and shading check**: the generated rays are checked for blocking and shading effects. Only those rays that are neither blocked nor shaded are evaluated in the final step.

3. **Ray evaluation**: rays that are not shaded nor blocked are traced against the receiver surface to verify that they indeed hit the receiver. To identify flux distribution the receiver panels are partitioned into smaller pieces and individual pieces are checked for intersection with the rays. If the ray hits a receiver piece, we add flux to the piece corresponding to the heliostat cell area minus losses from optical errors such as atmospheric attenuation, cosine effect and imperfect heliostat reflectivity. Reflection effects at receiver surfaces onto neighboring receiver surface are neglected, as this effect has a very low influence on the solution.

GPU ACCELERATED RAY-TRACING

For the implementation of the *SunFlower* ray-tracing module on GPU, the entire optical model of *SunFlower* was revised. A more lightweight optical model was reimplemented in CUDA code to make it compatible with and better suited to the GPU architecture. This includes implementing various efficient geometry operations and switching calculations to the 32bit float datatype where it does not significantly harm accuracy. Intrinsic functions are employed to make calculations as efficient as possible. Any information that is not essential to the calculations done on the GPU is omitted from the data model to reduce memory footprint. Compressed row storage arrays are implemented to store potential blocking and shading candidates. Tree traversal algorithms were adapted to work efficiently with the warp execution model of the GPU architecture. To reach a maximum speed-up, the GPU implementation was again optimized based on continuous investigation with the NVIDIA Nsight Compute performance analysis tool. Various code sections were further adapted to mitigate execution divergence during the highly dynamic simulation. As threads on the GPU work in warps, we must ensure that these threads stay synchronized. Furthermore, the parallelization methodology was tailored to optimize the occupancy of the GPU architecture and the efficiency of memory access. An enormous amount of tiny work chunks is created to allow the architecture to keep all parallel processors busy. We additionally coalesce memory access and reduce the number of necessary load operations by having threads within a warp work on a block of closely packed data. This is achieved by letting a block of at least 64 threads work on a single heliostat of the heliostat field. The final *SunFlower* GPU ray-tracer was validated against the previously validated CPU based *SunFlower* ray-tracer.

ACCELERATING BLOCKING & SHADING

To further speed up the simulation, we tackled the most expensive task in central receiver system simulation: blocking and shading. The state of the art was revised, introducing new approaches to reduce the computational intensity of blocking and shading. The main improvement is a new method to pre-calculate potentially blocking and shading heliostats based on the bounding sphere method. We filter the preselected candidates in two additional steps to rigorously reduce the number of expensive checks needed during ray tracing. The first additional step filters candidates for each heliostat facet using the bounding sphere method. In the final step each heliostat facet is projected towards the receiver (blocking) and sun (shading). Making use of the axis-aligned bounding boxes of heliostats we check whether this projection might intersect with any heliostat. An illustration of the bounding box filter is shown in Fig. 1. In this case one of the top corners of the AABB of heliostat A intersects the projection of the heliostat facet towards the sun. Therefore, this heliostat is confirmed as a potential shading candidate for the checked facet and must be checked for each ray cast towards this facet.

The very few remaining candidates for blocking and shading of each ray are tested with a greatly accelerated intersection-check due to the use of AABB trees [9].

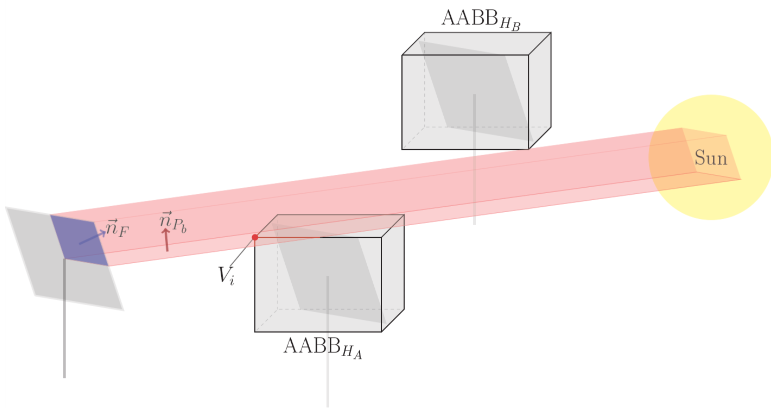

FIGURE 1. Facet projection plane intersection check against AABBs of potentially shading heliostats. In this case heliostat A is confirmed as potentially shading while heliostat B is rejected and can be ignored for shading considerations of the checked heliostat facet.

Case Study

To evaluate the performance of our *SunFlower* GPU ray-tracer we compare it to the performance of other state-of-the-art GPU implemented central receiver system ray-tracers. For the test case the SunFlower GPU ray-tracer is simulated for three different power plants with different heliostat field sizes and different receiver types, see Table 1.

TABLE 1. For the test case three different power plants were used: PS10, Gemasolar and AbengoaCRS.

	PS10	Gemasolar	AbengoaCRS
Location	Seville, Spain	Seville, Spain	Chile (hypothetical)
Number of heliostats	624	2650	8600
Total mirror surface	75 715 m^2	291 500 m^2	1 192 773 m^2
Receiver type	Cylindrical cavity	External cylindrical	External cylindrical
Receiver surface	165.36 m^2	425.09m^2	934.77 m^2

Thus, we can confidently judge the performance for different settings and any field size.

All tests are done on an otherwise idle system using a Ryzen 7 3700x 8-core CPU, 32 GB RAM and a NVIDIA Geforce RTX 2070. First the convergence of the *SunFlower* GPU ray-tracer was examined for all three power plants and different sun positions, using validated results of the previous *SunFlower* ray-tracer as reference [7]. The reference result was produced by a simulation with 100 rays per square meter of mirror surface. The reference was also validated with the established SolTrace and Tonatiuh tools.

Fig. 2, 3 and 4 depict the results for each of the three power plant configurations. Each simulation was run 30 times to account for Monte Carlo fluctuations. The shaded regions show the maximum and minimum reported optical power throughout these 30 simulations. With a sufficient amount of traced rays per square meter, the *SunFlower* GPU ray-tracer deviates by less than 0.05% from our validated reference result for all three configurations.

We determine that 50 rays per square meter (rpsm) for PS10, 35 rpsm for Gemasolar and 25 rpsm for AbengoaCRS are sufficient for accuracies above 99.95%. This means e.g., for AbengoaCRS we trace a total of 30 million rays per simulated moment. The convergence of the result to a normalized value of above 1 in Fig. 2 and 4 can be explained by the float datatype of the GPU and intrinsic functions. The inaccuracy of the GPU used floating point numbers is about 0.01 %.

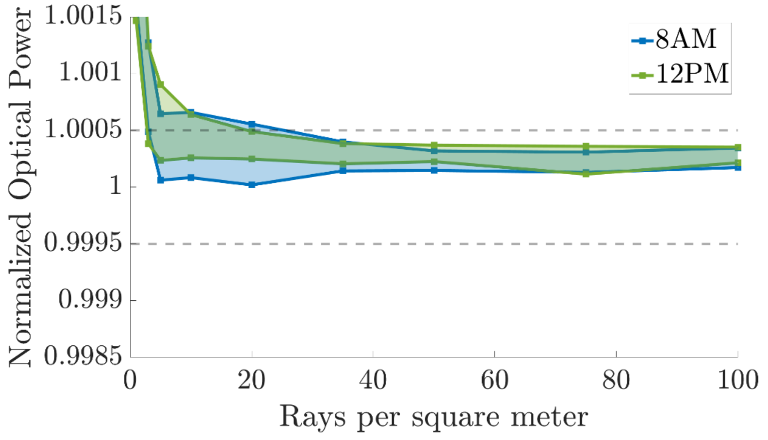

FIGURE 2. Normalized optical power for an annual simulation with different amounts of traced rays by our *SunFlower* GPU ray-tracer using the PS10 test case. The converged reference result was produced by a previous, already validated *SunFlower* ray-tracer (on a CPU with double precision) using 100 rays per square meter [7].

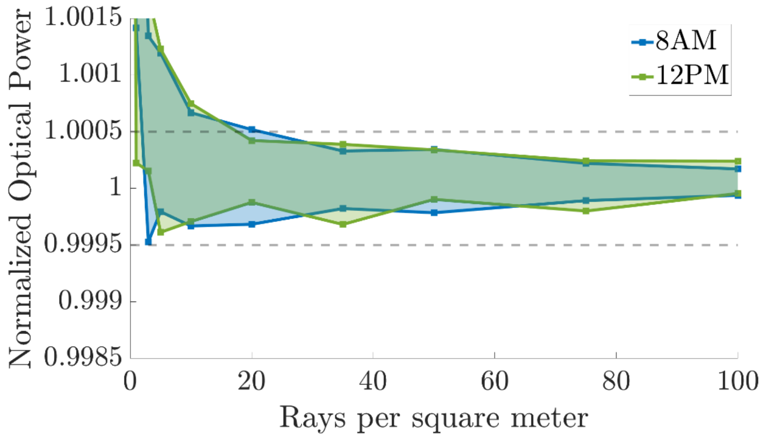

FIGURE 3. Normalized optical power for an annual simulation with different amounts of traced rays by our *SunFlower* GPU ray-tracer using the Gemasolar test case. The converged reference result was produced by a previous, already validated *SunFlower* ray-tracer (on a CPU with double precision) using 100 rays per square meter [7].

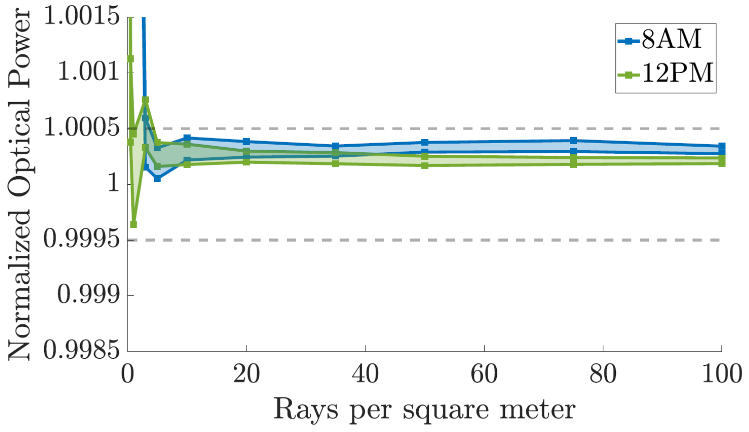

FIGURE 4. Normalized optical power for an annual simulation with different amounts of traced rays by our *SunFlower* GPU ray-tracer using the AbengoaCRS test case. The converged reference result was produced by a previous, already validated *SunFlower* ray-tracer (on a CPU with double precision) using 100 rays per square meter [7].

As Fig. 5 illustrates, the *SunFlower* GPU ray-tracer finishes an annual simulation for any of the power plant configurations in less than five seconds. This is orders of magnitude faster than typical CPU ray-tracers.

The simulation runtime scales roughly linearly with the heliostat field size. Even the annual simulation of the AbengoaCRS heliostat field with 8600 heliostats and close to 30 million traced rays per simulated moment finishes the around 50 required moment simulations in roughly three seconds. Annual simulations for smaller heliostat fields such as PS10 are computed in about 400 milliseconds.

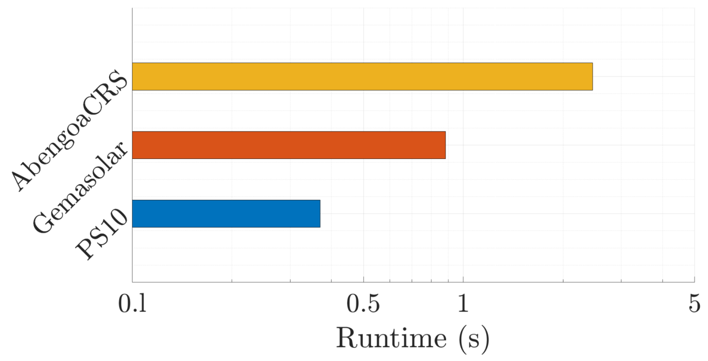

FIGURE 5. Runtimes for an annual simulation by our *SunFlower* GPU ray-tracer for three different power plants.

Fig. 6 shows that the *SunFlower* GPU ray-tracer is faster than other state-of-the-art tools with GPU parallelization, without making any compromises in terms of accuracy. The well-established TieSol ray-tracing tool is reported to compute just over 100 million rays per second [3]. The quasi-Monte Carlo ray-tracer QMCRT is reported to be slightly faster [4] and the recent commercial sbpRAY achieves significantly higher performance of multiple hundred rays per second [5]. Our experiments with the *SunFlower* GPU ray-tracer show that it is even faster than sbpRAY for any heliostat field size.

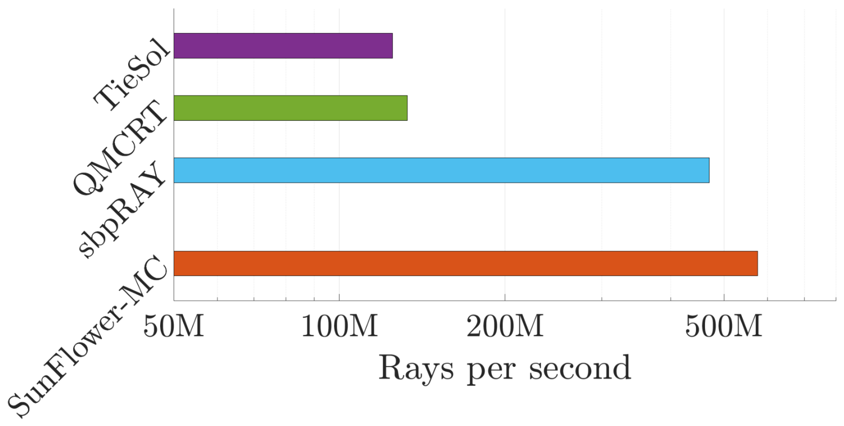

FIGURE 6. Comparison of the traced rays per second for all three power plant simulations with our *SunFlower GPU* ray-tracer to the reported performance of other state-of-the-art Monte Carlo GPU ray-tracers.

This difference between the four tools can be attributed to the efficient utilization of GPU resources and our accelerated blocking and shading method, as this is in general the most time-consuming unit for ray-tracers.

Conclusion

The above shown investigation underlines the potential of the GPU architecture in the context of optical simulations of central receiver systems. The investigation also confirms the advantages of our newly implemented approaches to reduce computational load of the simulation particularly in the context of blocking and shading tests. Altogether, the optimized GPU parallelization allows for annual simulations within very few seconds, making it possible to conduct heliostat field layout optimization in minutes.

The presented GPU parallelization and acceleration of the Monte Carlo based *SunFlower* ray-tracer can similarly be applied to analytical ray-tracing methods. As analytical ray-tracing methods typically allow even faster simulation of collected solar power and avoid Monte Carlo fluctuations, this is a promising topic for future research.

References

1. 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*(3), 189-197.
2. Cruz, N. C., Redondo, J., Berenguel, M., Álvarez, J., & Ortigosa, P. (2017). Review of software for optical analyzing and optimizing heliostat fields. *Renewable and Sustainable Energy Reviews*, *72*, 1001-1018.
3. Izygon, M., Armstrong, P., Nilsson, C., & Vu, Ngoc.: TieSol – A GPU-based suite of software for central receiver solar power plants. In *Proceedings of SolarPACES*, 2011.
4. Duan, X., He, C., Lin, X., Zhao, Y., & Feng, J.: Quasi-Monte Carlo ray tracing algorithm for radiative flux distribution simulation. *Solar Energy* 211 (2020): 167-182.
5. Gebreiter, D., Weinrebe, G., Wöhrbach, M., Arbes, F., Gross, F., & Landman, W.: sbpRAY – A fast and versatile tool for the simulation of large scale CSP plants. In *AIP Conference Proceedings*, 2019.
6. Richter, P., Heiming, G., Lukas, N. & Frank, M.: SunFlower: A new solar tower simulation method for use in field layout optimization. In *AIP Conference Proceedings*, 2018.
7. Richter, P. & Hövelmann, F.: Computationally fast analytical ray-tracer for central receiver system. In *AIP Conference Proceedings*, 2021.
8. Richter, P., Tinnes, J., & Aldenhoff, L.: Accurate interpolation methods for the annual simulation of solar central receiver systems using celestial coordinate system. *Solar Energy* 213 (2021): 328-338.
9. Williams, A., Barrus, S., Morley, R. K., & Shirley, P.: An efficient and robust ray-box intersection algorithm. In *ACM SIGGRAPH 2005 Courses* (2005).