2020 Building Performance Analysis Conference and
SimBuild co-organized by ASHRAE and IBPSA-USA

EFFICIENT COMPUTATION OF SURFACE SUNLIT FRACTIONS IN URBAN-SCALE BUILDING MODELING USING RAY-TRACING TECHNIQUES

Xuan Luo, Yu-hang Tang, Tianzhen Hong
Lawrence Berkeley National Laboratory, Berkeley, California, USA

ABSTRACT

For building energy simulation at an urban-scale, solar shading calculations can be significantly slow when a large number of shading surfaces are considered in the solar shading calculations, due to the computational complexity of the geometry calculations. We developed a new algorithm using the ray-tracing technique to pre-calculate the sunlit fractions of all exterior surfaces in an urban district altogether. The ray tracing-based calculator is accelerated using General Purpose Graphics Processing Units (GPGPUs) and the Optix ray tracing library, and provides an efficient, flexible, and robust means for computing the sunlit fraction of large numbers of urban surfaces of complex geometries.

INTRODUCTION

Buildings are responsible for about one-third of the global energy consumption and a quarter of carbon dioxide (CO2) emissions. Cities are paying greater attention to building energy efficiency in urban planning, and in meeting the city's goals for the reduction of greenhouse gas (GHG) emissions. To inform decision making in urban energy planning, it requires to design and operate urban buildings as a group rather than as single individuals, accounting for interactions among buildings (Hong et al. 2020). Modeling and optimizing the performance of a group of buildings at an urban scale provides quantitative insight for urban energy efficiency and resiliency.

Accurate calculation of solar shading on building exterior surfaces is of great importance in whole building energy modeling, especially in dense urban areas where buildings shaded each other. Building thermal load and energy simulation requires the calculation of the shadows caused by the building environment, building elements or shading devices based on the certain angle of incidence of the sun relative to the surface being considered (Dubois 1997). Algorithms for solar shading calculations have direct repercussions on the accuracy of the results and the computational times of building simulation tools. In building energy simulation tools, analytical solutions such as polygon clipping algorithms, are commonly used for shadow calculations for individual buildings with shading surfaces (Crawley et al. 2001). Research has also demonstrated the feasibility and advantage of using a computer's graphics processing unit (GPU) to accelerate the calculation of buildings' solar shading (Jones and Greenberg 2012).

Computational tools for UBEM can simulate the performance of buildings at urban-scale to provide quantitative insights for stakeholders into the impact of energy efficiency measures on buildings in urban environments. In UBEM, modeling the energy use of a building in an urban context requires simulating the buildings' shading effect with other buildings, including solar shading. To assess heat gains and daylight in buildings due to the sun for each of the building models, it is necessary to know how much of each part of the building is shaded and how much is in direct sunlight. However, modeling a building in an urban context may involve many shading surfaces from adjacent buildings and structures, which can significantly slow down building simulations using shading algorithms that calculate each building individually. In UBEM tools, GIS-based district level solar shading models have been developed to estimate the solar potential of a building group, including simplified geometry calculations (Melo et al. 2013; Karteris, Slini, and Papadopoulos 2013; Lilis, Giannakis, and Rovas 2017) and stochastic models with image processing (Liang et al. 2014).

Ray tracing was a computer graphics algorithm originally proposed for generating photorealistic renderings by emulating the transmission of light rays in 3D scene. In this algorithm, a number of rays are shot from a source region into the scene, while the intersection between the rays and scene objects are then solved and used to recursively compute a path of the ray bouncing between object surfaces. The ray tracing algorithm carries the following advantages:

1. The intersection of rays with many geometric objects can be solved analytically. This allows for efficient and robust handling of objects with non-convex and other complex shapes.

2. By manipulating the initiation and recursive generation of rays, it is straightforward to implement various surface/light interactions such as diffusion, transparency, and radiation.

3. The computation for tracing individual rays is strictly independent and hence could be parallelized easily.

Currently, ray tracing has found widespread application in tasks such as data visualization, indoor-light emulation, 3D model reconstruction, and optical design. The performance of ray tracing applications has also seen exponential growth thanks to the introduction of programmable GPGPUs and more recently RT cores in the CUDA architecture, which are specialized hardware for ray tracing. In particular, Monte Carlo Ray Tracing has been adopted in calculating solar shading and reflectance in urban canopy models for microclimate modeling (Wang et al. 2016; Krayenhoff et al. 2014; Erdélyi et al. 2014), but is less applied to detailed urban building geometries for UBEM.

In this study, we developed a new algorithm using the ray-tracing technique to pre-calculate the sunlit fractions of each exterior surface of all buildings at district- or city-scale. The ray-tracer we developed adopts the stochastic sampling techniques, which are used to simulate the interactions between rays and building objects, including exterior surfaces and shading surfaces in the building models. The calculator is then accelerated using General Purpose Graphics Processing Units (GPGPUs) and the Optix ray tracing library. We tested the tracer's performance using a standalone nine-zone EnergyPlus model with L-shape geometry, attached overhangs, and detached external shading surfaces. We conducted a sensitivity analysis with different number of rays as the input. We then applied the feature in EnergyPlus to batch import the pre-calculated sunlit fractions for annual simulation runs. A case study was conducted with a district of 22 high-rise buildings in downtown Chicago to evaluate the accuracy and computation cost using the ray-tracing technique, compared with using EnergyPlus to calculate solar shading for individual buildings. We also discussed the ray-tracer's performance regarding different number of urban surfaces considered in the shading calculation scene.

METHODOLOGY

Ray-tracing for shading calculation

To calculate building surface shading, the ray tracing algorithm works by shooting parallel rays, whose directions are determined by the solar angle and the geolocation of the building, onto the surfaces. The rays are generated from a randomized planar grid, whose direction is perpendicular to that of the rays. The spatial extent of the source grid is determined to be the most compact bounding box of the projection of all scene objects onto the source plane. The ray tracer will then compute the nearest hit point of each ray with the surfaces in the scene, and then respawn the rays beyond the hit point in order to discover all surfaces that lie along the direction of the rays. The respawned rays are necessary for estimating the total area of each surface and for computing shading behind semi-transparent surfaces, while the process will terminate when no more surface hits can be found for each ray. The sunlit fraction of a surface can then be deduced as the ratio between the energy of first-hit rays and respawned rays.

Our ray-tracing program is written in CUDA C++ and makes use of the NVIDIA Optix Prime library. CUDA C++ is an extension of the C++ language with syntactic features for executing codes on the CUDA GPUs. Optix Prime is a lower-level program engine that simplifies the design of ray tracing applications that executes on either CUDA GPUs or x86 CPUs. Specifically, we used Optix to compute the point of intersection between rays and building models, while our custom CUDA C++ code is used for scene modeling, ray generation and respawning, results consolidation and post-processing.

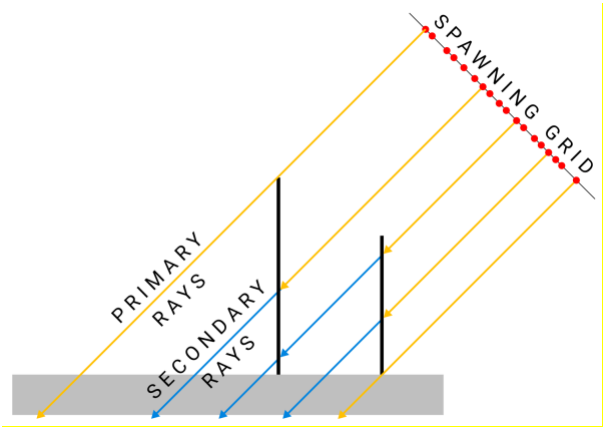

Figure 1 Illustration of the ray-tracing algorithm for calculating shadowing effect.

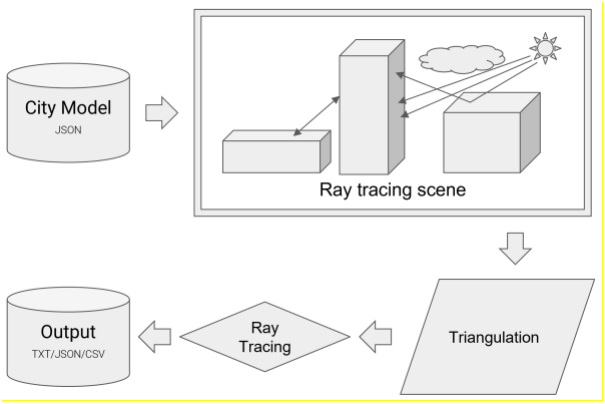

Figure 2 Flow chart of the ray tracer program.

For the input of the ray-tracer, we defined a "scene schema" in JSON format, describing the surfaces in the calculation scene. A scene specifies a list of the surfaces modeled by polygons. A surface is required to input its unique ID and 3D coordinate of the vertices. A surface can also specify its transmittance, and the default is opaque. A window surface is required to declare the ID of its attached wall surface.

The program produces plain text output which can be easily parsed and converted into common data file types such as CSV and JSON.

Applying the ray-tracer in EnergyPlus

When EnergyPlus assesses heat gains in buildings due to solar radiation, it needs to know how much of each part of the building is shaded and how much is in direct sunlight. Sunlit areas regarding the direct solar path determine the solar gain of the building. The sunlit area of each surface changes as the position of the sun changes during the day. At the hourly time step, the solar position is described in terms of three direction cosines that are convenient for determining the angle of incidence of the sun's rays on a building surface, calculated according to the solar declination angle and the equation of time. In an urban scene with a fixed site latitude and longitude, the annual schedule of solar position angles in three direction cosines (SUNCOS schedule) is also fixed and can be used as inputs of the ray-tracer.

For standalone building simulation, EnergyPlus employs polygon clipping algorithms to calculate sunlit fraction, including the Sutherlan-Hodgman algorithm and the Convex Weiler-Atherton algorithm. After polygon clipping, the solar module calculates the overlapping shadows to determine the final sunlit area for each surface. With the current analytical implementation, the results are accurate, but the calculation is computationally intensive and can take a long time for large building energy models with a lot of shading surfaces. We added a new feature to EnergyPlus (Version 8.9 and later) to optionally turn off the internal calculation of solar shading and import pre-calculated shading fractions as schedules for exterior surfaces. The new feature enables a significant speedup for parametric simulations that do not change building geometry or shading surfaces as the solar shading calculations do not need to be repeated between simulations. Currently, The *Schedule:File:Shading* object allows shading schedules to be imported altogether from a CSV file. The object can be used to read in hourly or sub-hourly schedules of the sunlit fraction of all exterior surfaces computed by other software or developed in a spreadsheet or other utility. We design the ray-tracer to output to the CSV format compatible with EnergyPlus, and we adopt this feature to import the sunlit fraction altogether to EnergyPlus calculated by the tracer.

RESULTS

We tested the calculation accuracy of the tracer compared to the analytical solution of EnergyPlus using (1) a fictional two-story, nine-zone building located in Phoenix, AZ, and (2) a high-rise office building located in downtown Chicago, IL. They are referred to as the Nine-story Building and High-rise Office, respectively. The Nine-story Building has detached shading, reveals, windows and self-shading, as visualized in Figure 3. The detached shadings, located in both the east and west side of the building, have a solar transmittance of 0.8. The High-rise Office is surrounded by other buildings and its surrounding surfaces are plotted in Figure 4. We ran EnergyPlus simulation for one week (January 1st to January 7th) at hourly timestep, and report the sunlit fractions of all walls, roofs, and windows.

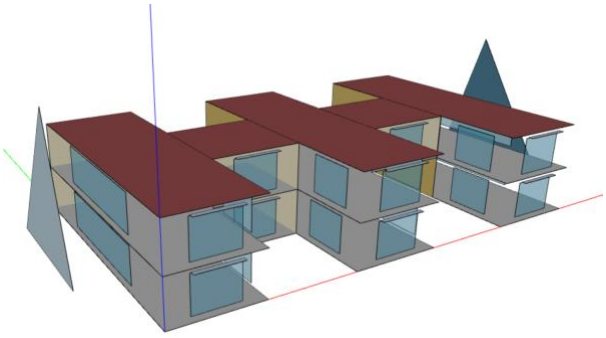

Figure 3 Geometry of a fictional nine-zone building.

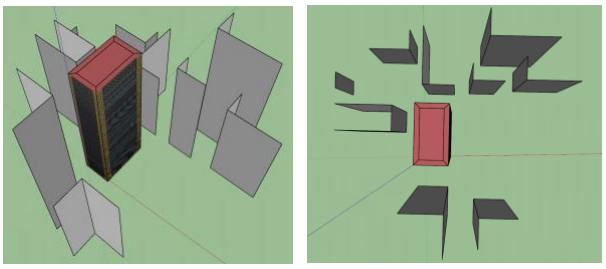

Figure 4 A high-rise building model and its shading surfaces.

The scene inputs for the models were created based on the exterior surfaces, fenestrations, and shading surfaces' geometry and transmittance inputs in EnergyPlus IDF input files. The SUNCOS schedules for Phoenix and Chicago were derived by EnergyPlus weather inputs.

For validation, we assume the sunlit fractions calculated by EnergyPlus' polygon clipping algorithms are the ground truth values, and the tracer should provide identical results with adequate sampling. However, it should be noted that the tracer currently has a simplified implementation for windows without considering the window recess, as illustrated in Figure 5. Differences are expected in window sunlit fraction calculation results due to neglecting the grey area shaded by the window recess.

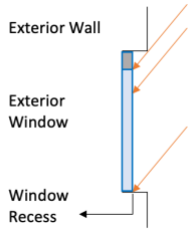

Figure 5 Shading of window recess

To measure accuracy, we reported both Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) between the ray-tracer result and EnergyPlus result as,

$$MAE = \frac{\sum_{t=1}^{k}\sum_{i=1}^{n}|f'-f|}{nk}, \quad (1)$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{t=1}^{k}\sum_{i=1}^{n}(f'-f)^2}{nk}}, \quad (2)$$

where f' is the ray-tracing result, f is the polygon clipping result, n is the total number of surfaces and k is the total number of time steps to compare.

We also compared the accuracy of introducing different number of rays in the ray-tracer calculation of the scene. For validation, the metrics were reported based on an average of (1) all surfaces and all timesteps (marked with subscript A, (2) partial shaded surfaces of those timesteps when the ground truth sunlit fraction is less than 1 and greater than 0 (marked with subscript P). For demonstration, windows are seperately compared. The accuracy measurements, excluding window surfaces, are listed in Table 1 and Table 2.

Table 1 Accuracy measurements of the Nine-zone Building scene

# OF RAYS	MAE_A	$RMSE_A$	MAE_P	$RMSE_P$
10,000	0.0030	0.0178	0.0081	0.0255
100,000	0.0009	0.0058	0.0024	0.0098
1,000,000	0.0005	0.0056	0.0014	0.0096
10,000,000	0.0005	0.0056	0.0014	0.0095

Table 2 Accuracy measurements of the High-rise Office scene

# OF RAYS	MAE_A	$RMSE_A$	MAE_P	$RMSE_P$
10,000	0.0619	0.2159	0.2403	0.4273
100,000	0.0021	0.0341	0.0120	0.0800
1,000,000	0.0002	0.0103	0.0008	0.0234
10,000,000	0.0000	0.0001	0.0000	0.0002

The results validate that with a reasonable number of rays (e.g., 100,000 for the Nine-zone Building and 1,000,000 for the High-rise Office) introduced to the scene, the tracer can calculate the sunlit fraction with an average RMSE of less than 0.01. The High-rise Office with many more surfaces requires more rays to achieve adequate accuracy. The error decreases with the number of rays increases from 10,000 to 1,000,000, while the effect gets marginal when the number of rays continues to go up.

Table 3 Accuracy measurements of the Nine-zone Building scene – window surfaces

# OF RAYS	MAE_A	$RMSE_A$	MAE_P	$RMSE_P$
10,000	0.0693	0.1313	0.1019	0.1591
100,000	0.0691	0.1312	0.1014	0.1589
1,000,000	0.0691	0.1312	0.1013	0.1590

For windows, however, the tracer results deviate from the EnergyPlus results as listed in Table 3 due to the simplification of the window recess. In these two cases, an average error of less than 0.1 of the sunlit fractions only causes an EnergyPlus simulated cooling and heating load difference of less than 0.1%, for both the annual total and peak loads. This is because the errors due to sampling are random, and can be either an over- or underestimation at each time step. If the errors are in one direction with an average overestimation of 0.1 (the window recess simplification), the annual energy difference can be over 4% for both test cases.

CASE STUDY

Domain and simulation settings

We chose a city block in the Chicago downtown area with highly dense high-rise buildings to conduct a case study, as shown in Figure 6. The block contains 20 buildings with various heights and geometry. Instead of conducting shading calculation building by building, with the ray-tracer, we consider the whole city block as a single urban scene to include all exterior surfaces in the block and calculate their shadowing effect.

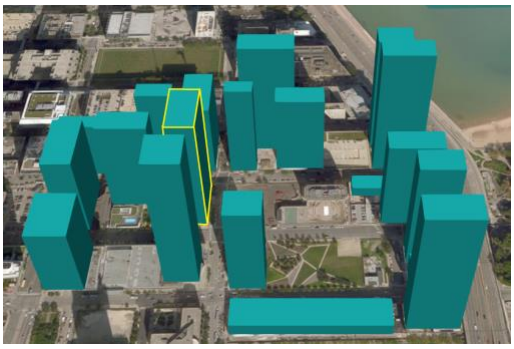

Figure 6 Building model and its shading surfaces.

We use the CityBES platform to model all buildings and compared the shading calculation between applying CityBES' internal shading models and applying the ray-tracer over this urban scene. CityBES is an open and free data and computing web platform which uses CityGML-based 3D city models to simulates building performance at an urban scale, adopting EnergyPlus and Openstudio as its simulation cores (Hong et al. 2016). CityBES models the neighborhood buildings as shading surfaces in EnergyPlus to consider the solar overshadowing effect between buildings (Chen, Hong, and Piette 2017). CityBES first processes the city GIS data to determine the shading/neighborhood buildings, and assigned the associated neighborhood exterior surfaces as the potential shading surfaces. The High-rise Building in Figure 4 is the 3D visualization of the highlighted building in Figure 6 along with its shading surfaces (grey) calculated by CityBES.

The internal model in CityBES still requires exhaust search of the potential shadowing surfaces in the building block or district. Beyond that, each building has to perform its own shading calculation. With a massive number of buildings in a larger simulation domain, the process can be heavy-lifting. Applying the ray-tracer in UBEM reduces the order of calculation by considering buildings as a group.

For EnergyPlus models in the urban block, the coordinates of each model are relative to its own origin. To generate a single simulation scene for the ray-tracer, we first converted the local coordinates to global based on each building's GIS location. Each surface in the scene was applied with a unique name, and the sunlit fraction was mapped back to the corresponding building surface after the ray-tracer calculation.

Run time analysis

The run time performance is measured by the computational time of the shading calculation functionality alone. In EnergyPlus, we set a customized timer measuring one-day simulation of the sunlit fraction computation. For the ray-tracer, we measured the total time of creating CUDA context, loading scene file, parsing, post-processing and recursive ray tracing. We further compared the ray-tracer's computing time of the three scenes to the calculation time in EnergyPlus in Table 4. The number of exterior surfaces and shading surfaces are also listed for reference. For the Chicago Urban District scene, there are no external shading or attached shading for buildings, while all buildings can shade each other. All measurements are based on the average hourly run time with a weekly simulation for both calculations.

Table 4 Sunlit fraction calculation computing time of the three scenes

CASE		NINE-ZONE OFFICE	HIGH-RISE BUILDING	CHICAGO URBAN DISTRICT
# Exterior Surfaces		72	517	3119
# Shading Surfaces		11	24	3119
Tracer Calculation Time (seconds)	10^4 rays	0.005	0.005	0.014
	10^5 rays	0.005	0.006	0.015
	10^6 rays	0.008	0.008	0.018
	10^7 rays	0.030	0.022	0.035
	10^8 rays	0.220	0.140	0.237
Tracer I/O Overhead (seconds)		0.150	0.159	0.278
EnergyPlus (seconds)		0.001	0.003	0.064

The run time benchmark results indicate the tracer takes only 0.5 seconds (calculation time + I/O overhead) to calculate the 20 buildings with 10^8 rays for a scene shot. The calculation run time is linear with the number of rays. The results show that for small models, the run time of shading calculation in EnergyPlus is neglectable. However, when the model becomes complex and the number of surfaces reaches a certain amount as in the high-rise building scene, the ray-tracer outperforms the polygon-clipping method in EnergyPlus. The run time of the ray tracer scales logarithmically with the scene size. The tracer has the advantage of calculating large urban

scenes with a group of buildings as it does not require to simulate building by building.

The report I/O overhead only includes the building scene input and processing time, and does not include I/O from the tracer back to EnergyPlus in this case study, which could be another overhead depending on the complexity of an urban scene. Efficiently adopting the ray-tracing technique requires estimation of the data I/O overhead as well.

CONCLUSION AND DISCUSSION

In this study, we introduced a ray tracing-based calculator for efficient computation of sunlit fractions in urban-scale building modeling. Tested with a sample building, the ray-tracer results' RMSE is less than 0.01 compared to EnergyPlus's analytical solution. Tested in an urban block with 20 high rise buildings, the tracer was able to process 10^8 rays in 0.5 seconds for a city-scale scene that contains 3120 surfaces. This approach provides an efficient, flexible, and robust means for computing the sunlit fraction of buildings that contain a large number of surfaces of complex geometries and absorbance characteristics. In an UBEM scene, the tracer can extract geometry information from EnergyPlus inputs and output sunlit fraction schedules back to EnergyPlus directly without extra pre- or post-processing. The workflow is also scalable to larger urban domains. Future work includes improving algorithms to handle window recess and applying ray-tracing considering solar reflectance for further tracing the solar beams. The work is also to be deployed in a larger UBEM domain for accuracy/performance analysis.

ACKNOWLEDGMENT

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration.

REFERENCES

Chen, Yixing, Tianzhen Hong, and Mary Ann Piette. 2017. "City-Scale Building Retrofit Analysis: A Case Study Using CityBES." In *Building Simulation 2017*. San Francisco, CA, USA.

Crawley, Drury B, Linda K Lawrie, Frederick C Winkelmann, W F Buhl, Y Joe Huang, Curtis O Pedersen, Richard K Strand, et al. 2001. "EnergyPlus : Creating a New-Generation Building Energy Simulation Program" 33.

Dubois, Mc. 1997. "Solar Shading and Building Energy Use." *Lund University*, no. 960480: 1–118. https://doi.org/TABK--97/3049.

Erdélyi, Róbert, Yimin Wang, Weisi Guo, Edward Hanna, and Giuseppe Colantuono. 2014. "Three-Dimensional SOlar RAdiation Model (SORAM) and Its Application to 3-D Urban Planning." *Solar Energy* 101: 63–73. https://doi.org/10.1016/j.solener.2013.12.023.

Hong, Tianzhen, Yixing Chen, Sang Hoon Lee, and Mary Ann Piette. 2016. "CityBES : A Web-Based Platform to Support City-Scale Building Energy Efficiency." In *Urban Computing 2016*. San Francisco, San Francisco, California, USA.

Hong, Tianzhen, Yixing Chen, Xuan Luo, Na Luo, and Sang Hoon Lee. 2020. "Ten Questions on Urban Building Energy Modeling." *Building and Environment* 168 (August 2019): 106508. https://doi.org/10.1016/j.buildenv.2019.106508.

Jones, Nathaniel L ., and Donald P. Greenberg. 2012. "Hardware Accelerated Computation of Direct Solar Radiation through Transparent Shades and Screens." *5th National Conference of the International Building Performance Simulation Association-USA*, 595–602.

Karteris, M., Th Slini, and A. M. Papadopoulos. 2013. "Urban Solar Energy Potential in Greece: A Statistical Calculation Model of Suitable Built Roof Areas for Photovoltaics." *Energy and Buildings* 62: 459–68. https://doi.org/10.1016/j.enbuild.2013.03.033.

Krayenhoff, E. S., A. Christen, A. Martilli, and T. R. Oke. 2014. "A Multi-Layer Radiation Model for Urban Neighbourhoods with Trees." *Boundary-Layer Meteorology* 151 (1): 139–78. https://doi.org/10.1007/s10546-013-9883-1.

Liang, Jianming, Jianhua Gong, Wenhang Li, and Abdoul Nasser Ibrahim. 2014. "A Visualization-Oriented 3D Method for Efficient Computation of Urban Solar Radiation Based on 3D-2D Surface Mapping." *International Journal of Geographical Information Science* 28 (4): 780–98. https://doi.org/10.1080/13658816.2014.880168.

Lilis, G N, G Giannakis, and D V Rovas. 2017. "Inter-Building Shading Calculations Based on CityGML Geometric Data."

Melo, Emerson G., Marcelo P. Almeida, Roberto Zilles, and José A.B. Grimoni. 2013. "Using a Shading Matrix to Estimate the Shading Factor and the Irradiation in a Three-Dimensional Model of a Receiving Surface in an Urban Environment." *Solar Energy* 92: 15–25. https://doi.org/10.1016/j.solener.2013.02.015.

Wang, Zhi Hua, Xiaoxi Zhao, Jiachuan Yang, and Jiyun Song. 2016. "Cooling and Energy Saving Potentials of Shade Trees and Urban Lawns in a Desert City." *Applied Energy* 161: 437–44. https://doi.org/10.1016/j.apenergy.2015.10.047.