

openpv/simshady: A Javascript Package for Photovoltaic Yield Estimation Based on 3D Meshes

Florian Kotthoff^{1,2}, Konrad Heidler¹, Martin Großhauser¹, and Korbinian Pöppel¹

¹ OpenPV GbR, c/o Martin Großhauser, Arnulfstrasse 138, 80634 München, Germany ² OFFIS Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany ¶ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [✉](#)

Submitted: 12 July 2025

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

openpv/simshady is a JavaScript package for simulating photovoltaic (PV) energy yields. It integrates local climate data and 3D objects into its shading simulation, utilizing Three.js meshes for geometric modeling. The package performs shading analysis using a WebGL-parallelized implementation of the Möller-Trumbore intersection algorithm Möller & Trumbore (1997), producing color-coded Three.js meshes that represent the expected PV yield.

Statement of need

To meet global climate targets, solar photovoltaic (PV) capacity must expand significantly. Tripling renewable energy capacity by 2030 is essential to limit global warming to 1.5°C (International Energy Agency, 2023). The expansion of PV plays a crucial role, and PV systems offer an additional benefit: small-scale house-mounted PV systems enable public participation and legitimize the energy transition.

For calculating the yield of PV systems, various factors are important, including the location of the planned installation, local climate, surrounding objects such as houses or trees, and terrain. To provide accurate estimates of expected yields, simulation tools are essential in both research and practical PV system planning.

For these reasons, a variety of software tools for simulating photovoltaic systems already exist (Holmgren et al., 2018; Jakica, 2018). One widely used software is the Python package pvlib (Anderson et al., 2023), which offers a range of functionalities. However, the rather niche topic of shading simulation with 3D objects is not included in this package. Another Python-based software that enables irradiance modeling in two dimensions is pvfactors (Anoma et al., 2017; Pvfactores, 2022).

Web-based tools for solar panel simulations, such as PVGIS, PVWatts, and RETScreen, provide an accessible means for non-technical individuals to estimate energy yields based on geographic location and building geometry (Psomopoulos et al., 2015). However, these tools lack the capability to perform shading simulations using 3D geometries.

Package description

openpv/simshady simulates the yield of photovoltaic (PV) systems by considering weather/climate data and shading from local 3D geometry. The model represents the environment through a 3D scene setup, comprising primary objects for simulation (e.g., PV panels or target buildings) and surrounding objects that may cast shadows (e.g., neighboring buildings, trees). Weather and climate data are integrated using Global Horizontal Irradiance (GHI) and Direct

39 Normal Irradiance (DNI) datasets, which are reconstructed to include directional irradiance
40 information using the HEALPix framework (Górski et al., 2005; Zonca et al., 2019).

41 The simulation utilizes the Möller-Trumbore intersection algorithm (Möller & Trumbore, 1997)
42 to determine if any shading objects obstruct the view between a sky pixel and the main
43 simulation geometry. For each triangle in the simulation geometry, a shading mask is generated,
44 indicating whether an object blocks the line of sight from the sky pixel to the triangle. The
45 shading mask values range from 0 to 1, where 0 indicates that an object shades the triangle,
46 1 signifies that there is no obstruction and the line of sight is perpendicular to the triangle,
47 and values between 0 and 1 represent cases where there is no obstruction but the angle of
48 incidence is not perpendicular. The aggregated radiance values from all sky dome pixels
49 are then multiplied by the corresponding shading mask values and summed to calculate the
50 total energy received by each triangle. This computation is fully parallelizable and has been
51 implemented using WebGL, allowing for GPU acceleration.

52 The package finally returns a color coded Three.js mesh, as shown in Figure 1. Additionally,
53 each triangle of the simulated buildings has its annual solar yield assigned as an attribute for
54 further processing.



Figure 1: A simulated building with its solar yield, where dark purple represents low yields and light yellow represents high yields. The simulated shading from neighboring buildings is clearly visible.

55 Conclusion

56 The openpv/simshady package serves two primary purposes: it provides a solution for scientific
57 calculations of PV yield, while also facilitating science communication through interactive and
58 user-friendly simulations that can be run directly within a web browser. This eliminates the need
59 for specialized software or programming knowledge, making it accessible to a broader range
60 of users. Furthermore, by implementing the main algorithm in WebGL, the package achieves
61 higher performance than a pure Javascript implementation, and it offers a JavaScript wrapper
62 around PV simulation in WebGL. This is particularly beneficial because WebGL is a language
63 that is not widely known among scientists, and thus can be challenging for them to implement
64 their own code, making the openpv/simshady package a valuable tool for simplifying this

65 process.

66 Credit Authorship Statement

67 FK: Conceptualization, Software, Funding acquisition, Writing – original draft

68 MG: Conceptualization, Software, Funding acquisition, Writing – review & editing

69 KH: Conceptualization, Software, Funding acquisition, Writing – review & editing

70 KP: Conceptualization, Software, Funding acquisition, Writing – review & editing

71 Acknowledgements

72 The development of this software was funded by the German Federal Ministry of Research,
73 Technology and Space within the “Software Sprint - Support for Open Source Developers”
74 program by Prototype Fund.

75 References

76 Anderson, K. S., Hansen, C. W., Holmgren, W. F., Jensen, A. R., Mikofski, M. A., & Driesse,
77 A. (2023). Pvlb python: 2023 project update. *Journal of Open Source Software*, 8(92),
78 5994. <https://doi.org/10.21105/joss.05994>

79 Anoma, M. A., Jacob, D., Bourne, B. C., Scholl, J. A., Riley, D. M., & Hansen, C. W.
80 (2017). View factor model and validation for bifacial PV and diffuse shade on single-
81 axis trackers. *2017 IEEE 44th Photovoltaic Specialist Conference (PVSC)*, 1549–1554.
82 <https://doi.org/10.1109/PVSC.2017.8366704>

83 Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., &
84 Bartelmann, M. (2005). HEALPix: A framework for high-resolution discretization and
85 fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2), 759.
86 <https://doi.org/10.1086/427976>

87 Holmgren, W. F., Hansen, C. W., Stein, J. S., & Mikofski, M. A. (2018). Review of open
88 source tools for PV modeling. *2018 IEEE 7th World Conference on Photovoltaic Energy
89 Conversion (WCPEC)(a Joint Conference of 45th IEEE PVSC, 28th PVSEC & 34th EU
90 PVSEC)*, 2557–2560. <https://doi.org/10.1109/PVSC.2018.8548231>

91 International Energy Agency. (2023). *Tripling renewable power capacity by 2030
92 is vital to keep the 1.5°C goal within reach*. [https://www.iea.org/commentaries/
93 tripling-renewable-power-capacity-by-2030-is-vital-to-keep-the-150c-goal-within-reach](https://www.iea.org/commentaries/tripling-renewable-power-capacity-by-2030-is-vital-to-keep-the-150c-goal-within-reach).

94 Jakica, N. (2018). State-of-the-art review of solar design tools and methods for assessing
95 daylighting and solar potential for building-integrated photovoltaics. *Renewable and
96 Sustainable Energy Reviews*, 81, 1296–1328. <https://doi.org/10.1016/j.rser.2017.05.080>

97 Möller, T., & Trumbore, B. (1997). Fast, minimum storage ray-triangle intersection. *Journal
98 of Graphics Tools*, 2(1), 21–28. <https://doi.org/10.1080/10867651.1997.10487468>

99 Psomopoulos, C. S., Ioannidis, G. C., Kaminaris, S. D., Mardikis, K. D., & Katsikas, N.
100 G. (2015). A comparative evaluation of photovoltaic electricity production assessment
101 software (PVGIS, PVWatts and RETScreen). *Environmental Processes*, 2, 175–189.
102 <https://doi.org/10.1007/s40710-015-0092-4>

103 *Pvfactories: Open-source view-factor model for diffuse shading and bifacial PV modeling* (Version
104 1.5.2). (2022). <https://github.com/SunPower/pvfactories>

105 Zonca, A., Singer, L., Lenz, D., Reinecke, M., Rosset, C., Hivon, E., & Gorski, K. (2019).
106 Healpy: Equal area pixelization and spherical harmonics transforms for data on the sphere
107 in python. *Journal of Open Source Software*, 4(35), 1298. [https://doi.org/10.21105/joss.](https://doi.org/10.21105/joss.01298)
108 [01298](https://doi.org/10.21105/joss.01298)

DRAFT