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

Florian Kotthoff [1,2¶], **Konrad Heidler** [1], **Martin Großhauser** [1], **Mino Estrella** [3], and **Korbinian Pöppel** [1]

1 OpenPV GbR, Munich, Germany **2** OFFIS Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany **3** Chair of Renewable and Sustainable Energy Systems, Technical University of Munich, Germany ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Adam R. Jensen

Submitted: 12 July 2025
Published: unpublished

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. Simshady provides both a package for web development, as well as a Command-line interface (CLI) tool for simulations on your machine.

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

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

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

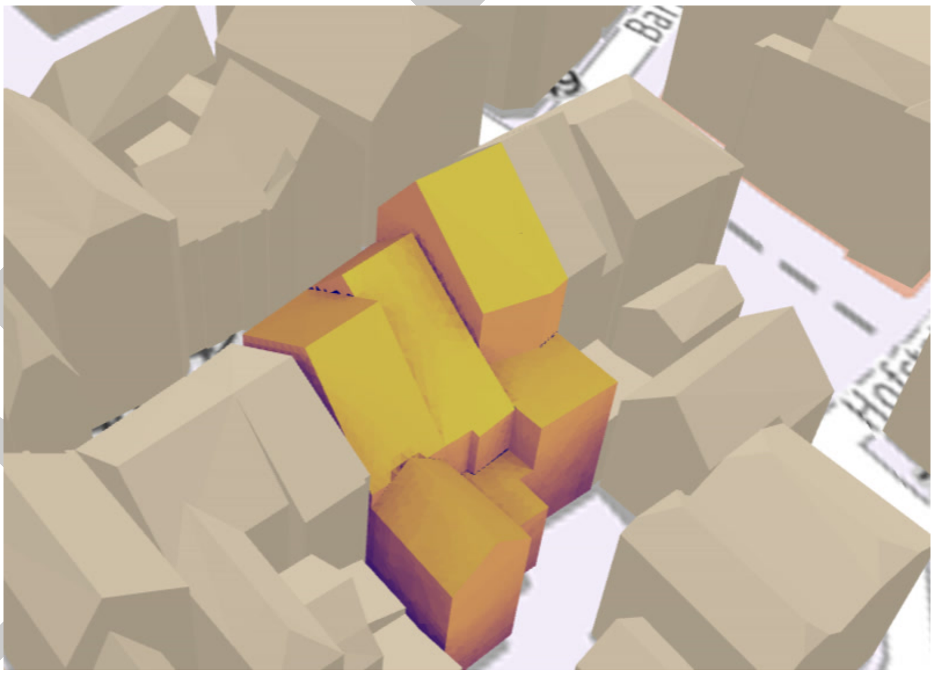

Figure 1: A simulated building with its solar yield, where dark purple represents low yields and light yellow represents high yields. The simulated shading from neighboring buildings is clearly visible.

The CLI tool

The `simshady` CLI is a thin wrapper around the core WebGL-based simulation engine that enables batch processing of photovoltaic-yield analyses on a headless server. It first parses a small set of required arguments (the simulation geometry and the irradiance data). The supplied geometry files, either JSON objects or Wavefront OBJ files, are handed to a headless Chromium instance launched via Puppeteer (Google Chrome Team, 2025).

Inside the browser context the full simshady package is injected, the scene is reconstructed, and the GPU-accelerated Möller-Trumbore ray-tracing routine is executed. When the calculation finishes, the CLI extracts the mesh data from the browser, writes binary artefacts (positions.bin, colors.bin, intensities.bin), a colour-coded OBJ file, a top-down snapshot, and the simulation results into the user-specified output directory.

Conclusion

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

Credit Authorship Statement

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

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

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

ME: Software, Writing – review & editing

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

Acknowledgements

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

References

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

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

Google Chrome Team. (2025). *Puppeteer: Headless chrome node.js API* (Version v24.31.0). Google. https://github.com/puppeteer/puppeteer

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

Holmgren, W. F., Hansen, C. W., Stein, J. S., & Mikofski, M. A. (2018). Review of open source tools for PV modeling. *2018 IEEE 7th World Conference on Photovoltaic Energy*

Conversion (WCPEC)(a Joint Conference of 45th IEEE PVSC, 28th PVSEC & 34th EU PVSEC), 2557–2560. https://doi.org/10.1109/PVSC.2018.8548231

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

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

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

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

Pvfactors: Open-source view-factor model for diffuse shading and bifacial PV modeling (Version 1.5.2). (2022). https://github.com/SunPower/pvfactors

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