Modeling of Partial Shading at the Cell Level on Photovoltaic Modules

Jean-Paul Calin[1,2,3], Jacques Levrat[2], Antonin Faes[1,2], Fahradin Mujovi[2], Paul Rémondeau[1], Kléber Nicolet-dit-Félix[1], Bénédicte Bonnet-Eymard[2], Didier Dalmazzone[3], Aïcha Hessler-Wyser[1], Christophe Ballif[1,2]
EPFL, PV-Lab, Maladière 71b, CH-2002 Neuchâtel[1]; CSEM, Rue Jaquet-Droz 1, CH-2002 Neuchâtel[2]; Institut Polytechnique de Paris, ENSTA, 828 Boulevard des Maréchaux, 91120 Palaiseau, France[3]

Introduction

Photovoltaic systems are highly sensitive to partial shading, which can significantly reduce their efficiency and overall energy output. In urban environments, where buildings and other structures frequently cast shadows, this issue is particularly pronounced. Efficient shadow modeling and electrical simulation are crucial for optimizing PV system performance.

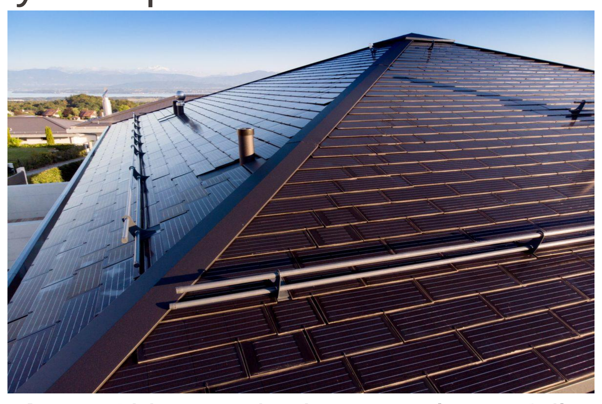


Fig. 1: Chimney shadow on solar roof tiles

Fig. 2: Roof structure shadow on PV modules

State of the Art:
PVSYST[1] is a professional software that supports fast shading factor estimation. It uses a pure geometric approach, neglecting important optical phenomena like reflection. Furthermore, the simulation is done at the module level, offering no visibility at the cell level.

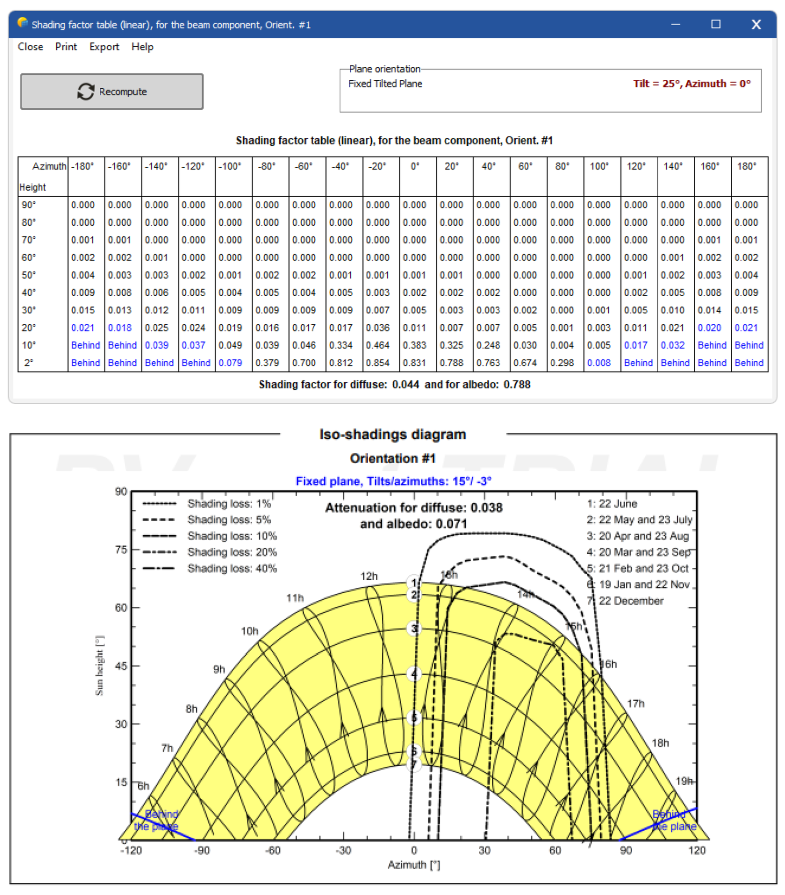


Fig. 3: Shading factor diagram from PVSYST simulation

Experimental Setup

Fig. 4: Obstacle installed in front of monitored outdoor roof tile modules

- Installed a chimney-like structure in front of roof tile modules
- Captured images of shadow projection throughout the day
- Monitored I-V curves, power, irradiance, and temperature

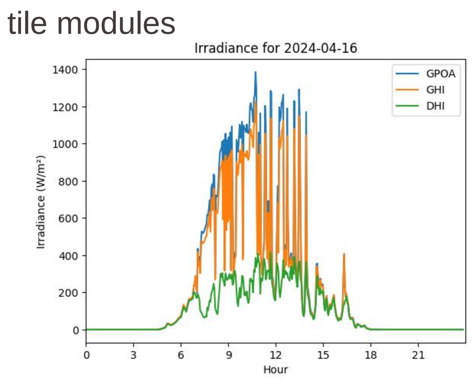
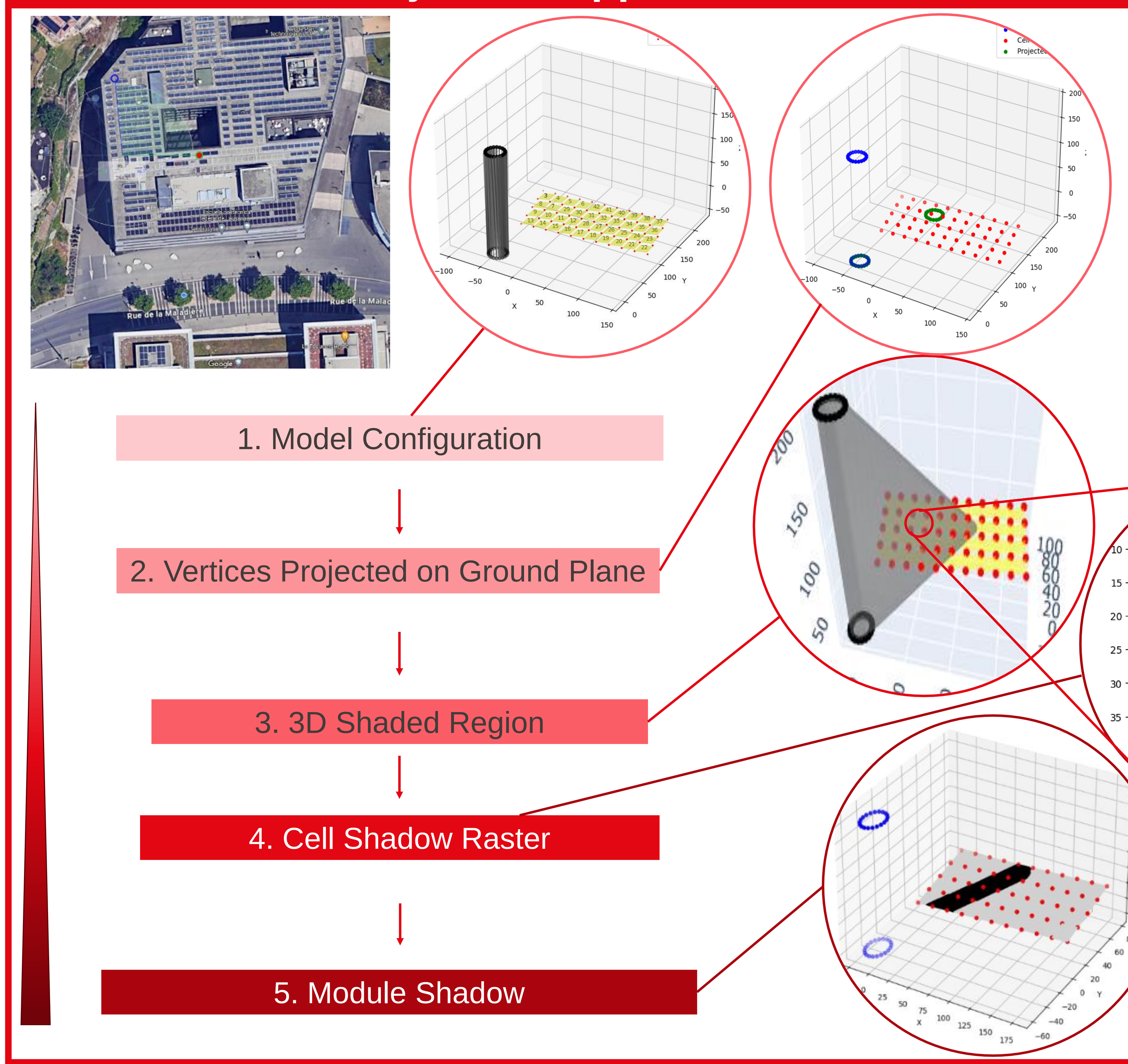
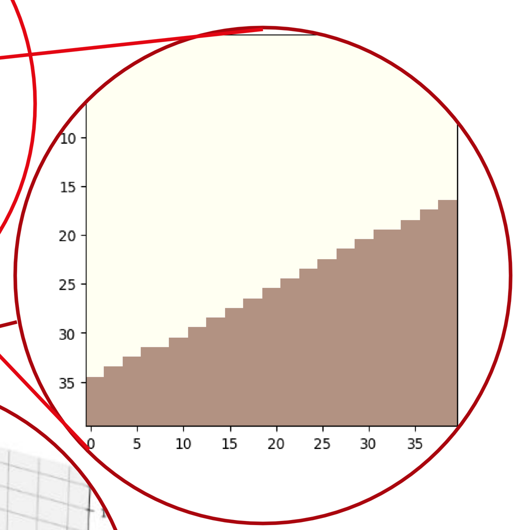


Fig. 5: Measured irradiance

Method: Vertex Projection Approach





- Pure geometric approach with **visibility at cell level**
- Original algorithm developed using pvlib-python[2] and inspired from de Sá et al.[3]
- Runtime optimization steps are described below

Method: Ray Tracing Approach

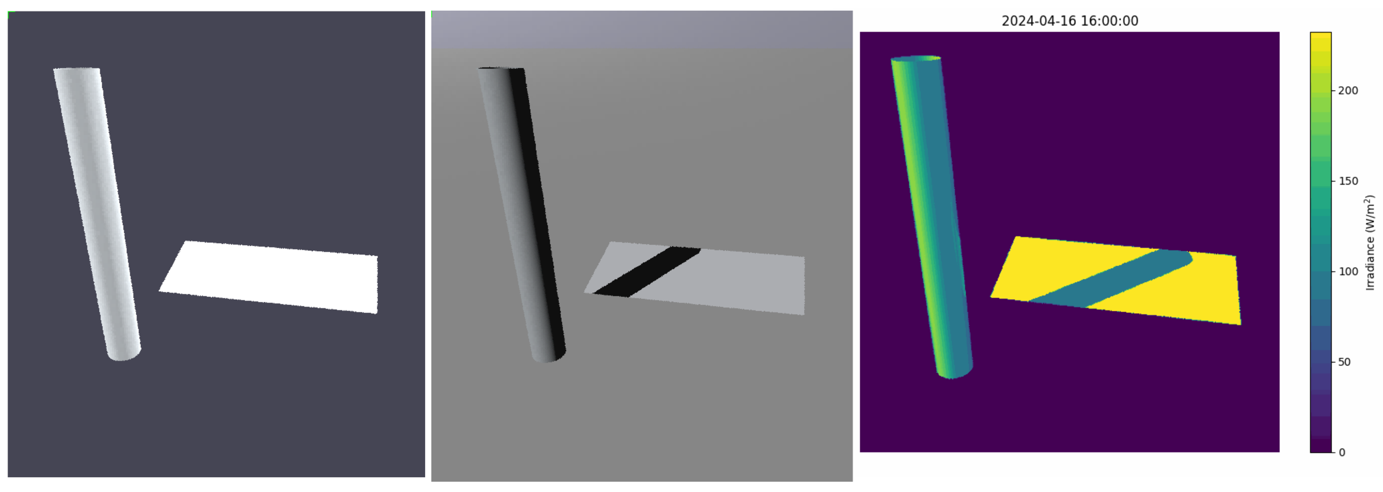
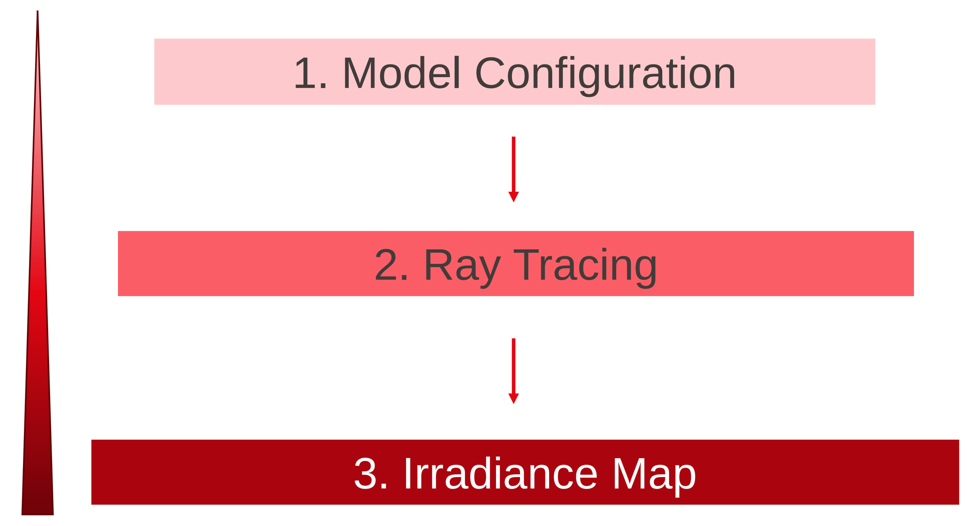


Fig. 6: Model configuration **Fig. 7:** Projected shadow **Fig. 8:** Irradiance map generated



Ray tracing provides **more precise irradiance** values by modeling:

- **Light ray interactions with materials**: reflection, refraction, absorption, transmission, and scattering

The **Radiance** package[4]:

- Calculates 3 components of light at each viewpoint: direct, specular indirect, and diffuse indirect component
- Uses a hybrid Monte Carlo and deterministic ray tracing method

Results

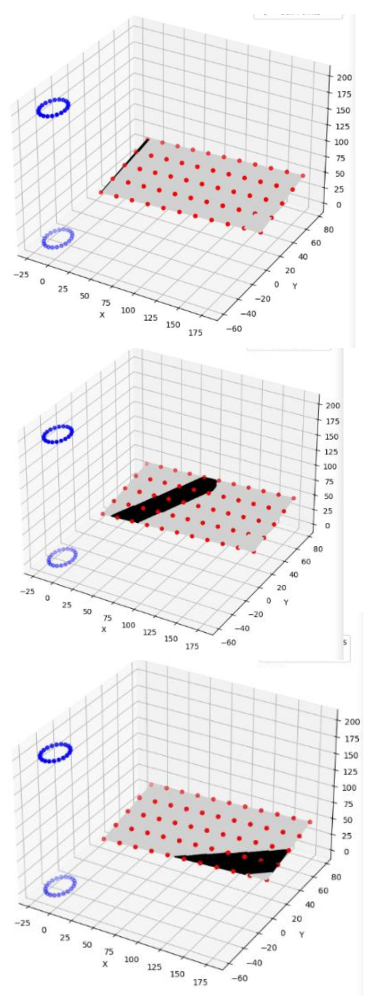


Fig. 9: Shadow simulated by vertex projection

Fig. 10: Experimental images of shadows

Vertex Projection Runtime Optimization

Goal : increase the speed of the vertex projection simulation

- The **most computationally intensive** step is the **shadow raster**
- Options for reducing the runtime:
  - ✓ Optimize data structures and computation
  - ✓ Check solar altitude angle: if negative, skip
  - ✓ Check the four corners of the cell: if all corners are shaded (illuminated), assume the whole cell is shaded (illuminated)
  - × Reduce cell grid size (= 40)
  - × Reduce the number of obstruction vertices (n points = 20)

→ The **runtime** of the simulation step **decreased from 3 minutes to 8 seconds**

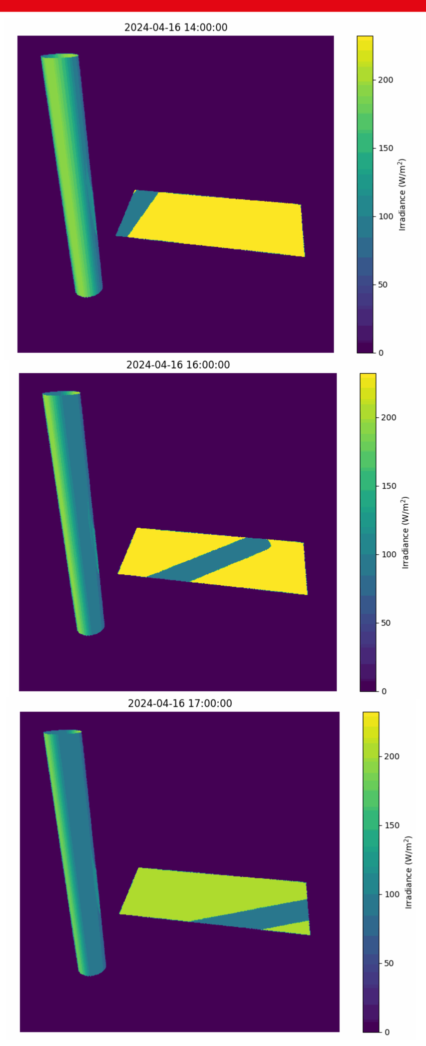


Fig. 11: Irradiance maps simulated with Radiance

Ray Tracing Runtime Optimization

Goal : increase the speed of the ray tracing simulation

- The **computational complexity** depends on the model configuration
- Options for reducing the runtime:
  - ✓ Use a cylinder to model the obstacle rather than 20 pairs of vertices
  - ✓ Use fewer light ray bounces (=2)

→ The **runtime** of the simulation step **decreased from 6.8 to 2.5 seconds**

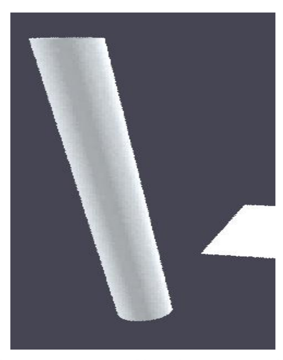


Fig. 12: Cylinder model configuration

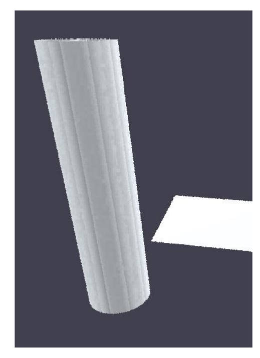


Fig. 13: Polygon model configuration

Conclusion

Experimentally validated shadow simulation on **cells in a module** using **vertex projection** and **ray tracing** methods. Both methods offer **cell-level visibility**, which is beyond the capabilities of PVSYST and **enables optimization of module parameters**, such as cell and string interconnections.

Timed the 1-year hourly simulation as typically **needed to compute Energy Yield**:

- **Vertex projection: offers cell-level visibility**: runtime improved from 14 days to 9.93 hours
- **Ray tracing: offers cell-level visibility** and **more precision** by modeling **light ray interactions**: runtime of 5.05 hours

Next steps:

- Electrical simulation from cell to module level
- Experimental validation on measured I-V curves

References
[1] A. Mermoud, "Pvsyst: a user-friendly software for pv-system simulation," in Twelfth European Photovoltaic Solar EnergyConference: proceedings of the International Conference, 1994.
[2] W. F. Holmgren, C. W. Hansen, and M. A. Mikofski, "pvlib python: A python package for modeling solar energy systems," Journal of Open Source Software, vol. 3, no. 29, p. 884, 2018.
[3] B. A. de Sá, T. Dezuo, and D. Ohf, "Shadow modelling algorithm for photovoltaic systems: extended analysis and simulation," Journal of Control, Automation and Electrical Systems, vol. 33, no. 5, pp. 1507–1518, 2022.
[4] G. W. Larson and R. Shakespeare, Rendering with Radiance: the art and science of lighting visualization. Morgan Kaufmann Publishers Inc., 1998.

Funded by the European Union's Horizon Europe, Innovation Actions programme under grant agreement No 101136112. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.
This work has received funding from the Swiss State Secretariat of Education, Research and Innovation (SERI).
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI