



**Politecnico
di Torino**

Master's Degree Course in Digital Skills for Sustainable Societal Transi-
tions

A.Y. 2026/2027

July 2026

**GIS-based Platform for Solar Radiation and Photovol-
taic Production Simulation and Analysis**

Supervisor:

Prof. Lorenzo Bottaccioli

Candidate:

Farinaz Goli

Acknowledgement

I would like to express my deepest gratitude to God for His countless blessings, guidance, and strength throughout this academic journey. This thesis would not have been possible without the support, patience, and encouragement of the people who stood by me during this important stage of my life.

I am sincerely grateful to my family for their unconditional love, continuous support, and belief in me. Their encouragement has always been a source of motivation and resilience, especially during the most challenging moments of my studies.

I would also like to extend my sincere appreciation to my supervisor and professors, whose guidance, knowledge, and valuable feedback helped me develop a deeper understanding of my research field and improve the quality of this thesis. Their academic support has played an important role in shaping this work.

A special thanks goes to my dear friend Jalal, whose support, kindness, and encouragement have meant a great deal to me throughout this journey. His presence and motivation helped me move forward with more confidence and determination.

Finally, I would like to thank everyone who, directly or indirectly, contributed to my academic and personal growth during these years. Every piece of advice, encouragement, and support has been truly appreciated.

Abstract

Urban rooftop photovoltaic assessment requires more than estimating available roof area or applying general solar-resource values. In dense urban environments, photovoltaic potential depends on rooftop geometry, slope, aspect, local obstruction, installation suitability, weather variability, and system operating conditions. Existing studies have addressed many of these components individually, but they are often treated as separate geospatial or photovoltaic modelling tasks rather than as parts of a coherent and reproducible computational workflow.

This thesis proposes, implements, and validates a backend-oriented framework for urban rooftop photovoltaic simulation that integrates geospatial preprocessing, clear-sky solar-radiation modelling, rooftop suitability analysis, structured artefact generation, and weather-aware photovoltaic estimation within a single reproducible workflow. The framework integrates GRASS GIS and the `r.sun` model through the Actinia execution environment for rooftop-scale geospatial and solar-radiation processing. A defining design feature is that intermediate outputs, including rooftop sampling points, suitable-area datasets, and rooftop-level clear-sky radiation baselines, are materialised as reusable Apache Parquet artefacts, decoupling the computationally intensive geospatial stage from downstream simulation and enabling different weather scenarios to be evaluated without repeating upstream computation. Two simulation paths are supported: a workflow-based real-sky path that scales precomputed clear-sky baselines with observed meteorological data before `pvlb`-based photovoltaic estimation, and a computationally lighter fast horizon-based path that combines locally derived horizon profiles with PVGIS horizon data for centroid-level estimation.

The framework is validated against measured generation data from four photovoltaic systems in the La Crocetta district of Turin, Italy, spanning nominal powers from 13.20 kWp to 631 kWp and two independent monitoring periods a decade apart (2010 to 2011 and 2020). The ERBS irradiance decomposition model is iden-

tified as the most suitable choice for the Turin urban context, outperforming three alternative decomposition variants and direct irradiance component supply from reanalysis products across all systems. For normally operating systems driven by the local Polito Weather Station, the framework achieves daily RMSD values between 9.46% and 14.68% across systems spanning nearly fifty times the nominal power, demonstrating scale-invariant accuracy without system-specific calibration. The local Polito source consistently outperforms the Open-Meteo reanalysis product across all systems, with daily RMSD improvement factors ranging from 1.05 to 2.25, most pronounced under cloudy and rainy conditions. The framework additionally demonstrates utility as an operational fault detection tool, identifying a hardware malfunction through a systematic divergence between simulated and measured generation, quantified as an increase in daily RMSD from 8.98% to 16.17% and a reversal in mean bias sign.

These results establish the proposed framework as a robust, reproducible, and scalable infrastructure for urban rooftop photovoltaic assessment, with validated capabilities spanning energy yield estimation, cross-scale deployment analysis, weather source evaluation, and operational performance monitoring.

Table of Contents

Introduction	1
Background and Literature Review	5
2.1 Introduction.....	5
2.2 Urban rooftop photovoltaics as a research domain.....	6
2.3 Geospatial data and rooftop representation in PV assessment	8
2.4 Terrain-derived factors and rooftop solar exposure.....	12
2.5 GIS-based solar-radiation modelling in urban rooftop studies.....	14
2.6 Rooftop suitability and deployable installation areas.....	17
2.7 Weather-aware and time-resolved photovoltaic estimation.....	19
2.8 Computational workflows, web-based platforms, and service-oriented infrastructures.....	21
2.9 Research gap and positioning of the thesis	24
2.10 Summary	26
Enabling Technologies.....	27
3.1 Introduction.....	27
3.2 Geospatial data representations used in the framework.....	28
3.3 GRASS GIS, r.sun, and Actinia.....	30
3.4 Photovoltaic, weather, and external solar data technologies	32
3.5 Artefact and persistence technologies	35
3.6 Backend and deployment technologies.....	38
3.7 Summary.....	41
Methodology	43
4.1 Introduction.....	43
4.2 System Overview.....	43
4.2.1 System goals and design principles	43
4.2.2 High-Level Architecture.....	44
4.2.3 End-to-end workflow summary	54

4.3 Access control and identity management.....	58
4.3.1 Authentication model.....	58
4.3.2 Token lifecycle and security measures.....	59
4.3.3 User management capabilities.....	62
4.4 Data management and reproducibility.....	63
4.4.1 Storage layout and dataset organisation.....	63
4.4.2 Metadata and job tracking.....	65
4.4.3 Validation and quality gates.....	66
4.5 Core geospatial processing pipeline (Actinia/GRASS).....	68
4.5.1 Actinia integration and execution model.....	68
4.5.2 Location and mapset management.....	69
4.5.3 Data import: raster and vector.....	71
4.5.4 Derived-surface workflow.....	72
4.5.5 High-resolution DSM workflow.....	74
4.5.6 Geospatial preprocessing.....	75
4.6 Simulation methodology.....	77
4.6.1 Transformation of GRASS outputs into analytical artefacts.....	77
4.6.2 Real-sky computation and weather handling.....	83
4.6.3 Fast horizon-based simulation approach.....	86
4.6.4 Parallel execution model.....	90
4.6.5 Output products and formats.....	93
4.7 Implementation notes.....	96
4.7.1 API surface and endpoint conventions.....	96
4.7.2 Error handling and logging.....	97
4.7.3 Configuration management.....	99
4.8 Summary.....	101
Results and Discussion.....	103
5.1 Introduction.....	103
5.2 Case Study and Experimental Setup.....	104
5.3 Performance Indicators.....	107
5.4 Decomposition Model Selection.....	108
5.5 PV System Simulation Validation.....	113

5.5.1 Sommelier.....	114
5.5.2 Galfer.....	115
5.5.3 I3P.....	117
5.5.4 Cittadella	123
5.6 Weather Source Comparison: Polito vs Open-Meteo.....	127
5.7 PV System Operation Assessment	129
5.8 Discussion	131
5.9 Summary.....	134
Conclusion	136
6.1 Summary of the work.....	136
6.2 Main findings	137
6.3 Limitations	139
6.4 Future directions	139
6.5 Concluding remarks.....	140
References	141
Appendix: Implementation-oriented UML Diagrams	145
A.1 Core backend service structure.....	145
A.2 Solar service package structure.....	146
A.3 Selected implementation excerpts	147

List of Figures

Figure 2.1. Rooftop slope, aspect, and directional obstruction in urban PV assessment	13
Figure 3.1. DTM vs. DSM and raster vs. vector representations in the framework	30
Figure 3.2. GRASS GIS, r.sun, and Actinia in the geospatial computation stack...	32
Figure 3.3. PV modelling, weather input, and external solar-resource data relationships.....	35
Figure 3.4. Parquet, PostgreSQL, and Valkey in the persistence layer.....	38
Figure 3.5. Backend service layer and containerised deployment environment...	40
Figure 4.1. High-Level Architecture of the Proposed Framework.....	45
Figure 4.2. Input Configurations and DSM Preparation Workflow	55
Figure 4.3. Clear-sky preprocessing workflow	56
Figure 4.4. Authentication-Layer Token Lifecycle and Security Flow	60
Figure 4.5. Entity-Relationship Diagram of the Metadata and Tracking Layer	65
Figure 5.1. Geographic locations of the four PV systems and Polito Weather Station, La Crocetta, Turin.....	105
Figure 5.2. Sommelier – daily energy comparison, July 2010, Model 2 variants vs. measured (Polito Weather Station).....	112
Figure 5.3. Sommelier – daily energy comparison, July 2010, Model 1 & 2 variants vs. measured (Open-Meteo).	113
Figure 5.4. Sommelier – simulated vs. measured instantaneous power profiles by weather type.	115
Figure 5.5. I3P East – simulated vs. measured instantaneous power profiles by weather type.	120

Figure 5.6. I3P West — simulated vs. measured instantaneous power profiles by weather type.	122
Figure 5.7. Cittadella — simulated vs. measured instantaneous power profiles by weather type.	126
Figure 5.8. GalFer — simulated vs. measured power profiles by weather type, before and after the May 2010 malfunction.....	130
Figure A.1. Core backend service class diagram.....	151
Figure A.2. UML diagram of the solar service package.	152

List of Tables

Table 2.1. Main geospatial representation types used in rooftop photovoltaic assessment, together with their typical data	9
Table 2.2. Representative tools and platforms in rooftop solar and photovoltaic assessment	16
Table 2.3. Representative computational approaches and platforms related to rooftop solar and photovoltaic assessment	22
Table 4.1. User-related API Endpoints and Responsibilities	62
Table 4.2. Validation Checks and Failure Modes in the Ingestion Layer.....	67
Table 4.3. Representative Monthly Average Days and Corresponding Day-of-Year (DOY).....	77
Table 4.4. Parallel Execution Strategies Across the Simulation Pipeline	90
Table 4.5. Artefacts, path organisation, format, and purpose across the simulation pipeline.....	94
Table 5.1. Technical characteristics of the four PV systems used for validation.	107
Table 5.2. I3P East array – daily model selection, 2020 (Polito Weather Station)	109
Table 5.3. Daily model selection summary – all systems and irradiance sources	110
Table 5.4. Sommelier – ERBS simulation performance indicators, Polito Weather Station, March 2010 to February 2011.	114
Table 5.5. Sommelier – ERBS simulation performance indicators, Open-Meteo, March 2010 to February 2011.....	114
Table 5.6. GalFer – ERBS performance indicators, Polito Weather Station, 2010–2011 (pre/post malfunction).	116

Table 5.7. GalFer — ERBS performance indicators, Open-Meteo, 2010–2011 (pre/post malfunction).	116
Table 5.8. I3P East — seasonal performance indicators by weather type, 2020 (15-min, Polito & Open-Meteo).	118
Table 5.9. I3P West — seasonal performance indicators by weather type, 2020 (15-min, Polito & Open-Meteo).	121
Table 5.10. Cittadella — seasonal performance indicators by weather type, 2020 (15-min, Polito & Open-Meteo).	124
Table 5.11. Cross-system ERBS daily performance — Polito vs. Open-Meteo, all four PV systems.	127

List of Listings

Listing 4.1. Two-stage DSM smoothing and slope/aspect derivation in GRASS..	76
Listing 4.2. Resampling heterogeneous weather inputs to a uniform 15-minute grid	84
Listing 4.3. Real-sky irradiance scaling and PVWatts DC-power estimation	86
Listing A.1. Centralized backend configuration and path normalisation	148
Listing A.2. Executing GRASS commands inside the Actinia container	149
Listing A.3. Persisting workflow result manifests	150
Listing A.4. Canonical weather retrieval from Open-Meteo	152
Listing A.5. Scaling clear-sky area means with real weather	153
Listing A.6. Azimuth convention harmonisation for fast horizon PV	154
Listing A.7. Per-timestep real-sky photovoltaic computation	155

List of Equations

Equation 4.1. Inward buffer spacing in a projected (metric) CRS.....	73
Equation 4.2. Inward buffer spacing in a geographic (angular) CRS	73
Equation 4.3. Building height from DSM and DTM.....	73
Equation 4.4. Effective raster resolution.....	74
Equation 4.5. High-resolution DSM acceptance condition	74
Equation 4.6. Rooftop slope suitability criterion	80
Equation 4.7. Rooftop aspect suitability criterion	80
Equation 4.8. Suitability raster definition.....	81
Equation 4.9. Effective rooftop installation area.....	81
Equation 4.10. Raster cell area in a projected CRS.....	81
Equation 4.11. Mean direct (beam) clear-sky irradiance per rooftop area.....	82
Equation 4.12. Mean diffuse clear-sky irradiance per rooftop area	82
Equation 4.13. Direct-irradiance scaling coefficient	85
Equation 4.14. Diffuse-irradiance scaling coefficient.....	85
Equation 4.15. Rooftop-area centroid	89
Equation 4.16. GRASS-to-pvlib azimuth conversion.....	87
Equation 4.17. Merged horizon profile.....	88
Equation 4.18. Global horizontal irradiance reconstruction	88
Equation 4.19. Horizon-masking condition.....	88
Equation 4.20. DC power output (PVWatts model).....	89
Equation 4.21. Photovoltaic cell temperature	91
Equation 4.22. DC power output (simplified form)	92

List of Acronyms

AC	Alternating Current
API	Application Programming Interface
CPU	Central Processing Unit
CRS	Coordinate Reference System
CSV	Comma-Separated Values
CRUD	Create, Read, Update, Delete
DC	Direct Current
DHI	Diffuse Horizontal Irradiance
DISC	Direct Insolation Simulation Code (decomposition model)
DIRINDEX	Direct Irradiance Index (decomposition model)
DIRINT	Direct Irradiance (decomposition model)
DNI	Direct Normal Irradiance
DOY	Day of Year
DSM	Digital Surface Model
DTM	Digital Terrain Model
EPW	EnergyPlus Weather (file format)
EPSG	European Petroleum Survey Group (spatial reference identifier)
ERBS	Erbs-Klein-Duffie irradiance decomposition model

GHI Global Horizontal Irradiance

GIS Geographic Information System

GRASS Geographic Resources Analysis Support System

GTI Global Tilted Irradiance

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

JWT JSON Web Token

LCE Legates Coefficient of Efficiency

MAD Mean Absolute Difference

MBD Mean Bias Difference

NOCT Normal Operating Cell Temperature

OGC Open Geospatial Consortium

ORM Object-Relational Mapping

PV Photovoltaic

PVGIS Photovoltaic Geographical Information System

REST Representational State Transfer

RMSD Root Mean Square Difference

SAPM Sandia Array Performance Model

UI User Interface

UML Unified Modelling Language

URL Uniform Resource Locator

UTC Coordinated Universal Time

WIA Willmott Index of Agreement

Chapter 1

Introduction

The transition towards low-carbon and decentralised energy systems has increased the importance of photovoltaic generation in urban environments. Among the available urban surfaces, rooftops represent one of the most promising resources for photovoltaic deployment because they allow renewable electricity production to be integrated into the built environment without requiring additional land. This is particularly relevant in dense cities, where available land is limited, energy demand is concentrated, and the built fabric itself can become part of the distributed energy infrastructure. In this context, rooftop photovoltaic systems are not only individual energy installations but also spatially distributed assets whose potential depends on the physical, geometric, and environmental characteristics of the urban landscape.

Assessing rooftop photovoltaic potential in cities, however, is not a simple task. Urban rooftops are heterogeneous in size, height, slope, orientation, surrounding obstruction, and installation suitability. Two buildings located close to each other may receive very different amounts of solar radiation because of differences in rooftop geometry, shading from neighbouring structures, or exposure to the sky dome. Moreover, photovoltaic production is not determined by solar availability alone: it also depends on weather conditions, irradiance composition, module temperature, system configuration, and the temporal variability of atmospheric conditions. For this reason, reliable rooftop PV assessment requires more than estimating available roof area or applying regional solar-radiation averages. It requires a workflow capable of connecting spatial representation, terrain-derived analysis, solar-radi-

ation modelling, rooftop suitability detection, weather-aware photovoltaic simulation, and validation against measured production data.

Existing research has addressed many of these components individually. Geospatial and GIS-based approaches have been widely used to represent buildings and rooftops, derive terrain-related variables such as slope and aspect, and estimate solar radiation across urban surfaces. Other studies have focused on photovoltaic performance modelling, weather-aware simulation, irradiance decomposition, or the use of web-based solar assessment platforms. These contributions provide important methodological and technical foundations for rooftop PV analysis. Nevertheless, the existing literature often treats these components separately: rooftop assessment is frequently presented either as a spatial mapping problem or as a photovoltaic performance-estimation problem, and less attention is given to the reusable intermediate products that connect one stage to the next. These intermediate products are not mere by-products of the workflow: they affect reproducibility, traceability, and the ability to reuse earlier results under changing downstream conditions, a point that becomes critical as assessment moves from static mapping towards weather-aware, time-resolved simulation. The research gap addressed in this thesis is therefore methodological as much as analytical: what is missing is not another isolated photovoltaic model, but a structured, backend-oriented framework that connects the full rooftop PV assessment chain through explicit workflow organisation, reusable intermediate artefacts, and controlled execution logic.

The objective of this thesis is to develop and validate a reproducible computational framework for urban rooftop photovoltaic simulation. The proposed framework integrates geospatial preprocessing, solar-radiation modelling, rooftop suitability analysis, weather-aware photovoltaic estimation, and structured artefact generation within a coherent backend workflow. A central feature of its design is that intermediate products, including rooftop sampling points, suitable-area datasets, point-level clear-sky radiation outputs, and rooftop-level radiation baselines, are materialised as persistent, reusable datasets rather than treated as transient outputs. This artefact-driven approach decouples the computationally expensive geospatial preprocessing stage from downstream simulation, allowing the same spatial and radiative baseline to be reused across different weather scenarios and simulation branches without repeating the upstream computation. The framework supports two downstream simulation paths: a workflow-based real-sky path that scales precomputed

clear-sky baselines with observed meteorological data before photovoltaic estimation, and a computationally lighter fast horizon-based path for selected rooftop areas. Together, these paths allow the framework to accommodate different balances between analytical detail, data availability, and execution efficiency.

The framework is validated against measured generation data from four real photovoltaic systems installed in the La Crocetta district of Turin, Italy: *Sommelier*, *GalFer*, *I3P*, and *Cittadella*. These systems differ in size, installation type, and orientation, ranging from small rooftop installations to the large 631 kWp *Cittadella* system, and span two independent monitoring periods a decade apart. The validation evaluates irradiance decomposition model selection, compares two irradiance sources of different quality, assesses simulation accuracy across time resolutions and weather conditions, and explores the potential of the framework as a diagnostic tool for operational fault detection.

The contribution of this thesis is twofold. From a methodological perspective, it proposes a staged and reproducible framework that connects rooftop representation, geospatial pre-processing, solar-radiation modelling, suitability detection, and weather-aware photovoltaic simulation within a single backend-oriented workflow. From a practical perspective, it evaluates this framework on real urban PV systems and demonstrates how simulation accuracy depends on weather-source quality, temporal resolution, installation type, and system operating conditions. In doing so, the thesis positions rooftop photovoltaic assessment not as a sequence of isolated modelling tasks, but as an integrated computational process whose organisation directly affects reproducibility, traceability, and analytical usefulness.

The remainder of the thesis is organised as follows. Chapter 2 reviews the literature on urban rooftop photovoltaic assessment, covering geospatial representation, terrain-derived exposure factors, GIS-based solar-radiation modelling, rooftop suitability analysis, weather-aware photovoltaic estimation, and computational workflow design, before identifying the research gap that motivates the present work. Chapter 3 introduces the enabling technologies on which the framework is built. Chapter 4 presents the methodology in full, describing the system architecture, the geospatial processing pipeline, the artefact generation logic, the simulation paths, and the implementation. Chapter 5 reports the validation results and discussion, including decomposition model selection, system-level simulation accuracy, weather-source comparison, and the operational fault detection case study. Chapter 6 pre-

sents the conclusions, including the main findings, limitations, and directions for future work.

Background and Literature Review

2.1 Introduction

Rooftop photovoltaic assessment in urban environments has shifted from a broad concern with solar availability towards a more structured problem: how individual rooftop surfaces can be represented, evaluated, and compared under realistic spatial and environmental conditions [1], [2]. In dense urban areas, rooftops cannot be treated as uniform receiving planes, since their photovoltaic potential depends on interacting factors such as rooftop geometry, surface orientation, local obstruction, solar-radiation patterns, deployable area, and weather-dependent operating conditions [2]– [5]. Rooftop PV analysis therefore requires methods that connect the physical characteristics of the built environment to computational procedures supporting rooftop-level interpretation and simulation [1], [2]. This relevance is reinforced at the policy level by international decarbonisation objectives and clean-energy transition pathways [11], [13] and, in European contexts, by the incentive frameworks that have shaped PV deployment [12].

Accordingly, the body of literature relevant to this thesis spans several interrelated domains rather than a single narrowly defined research field. It encompasses urban rooftop photovoltaics as a distributed energy resource, geospatial data and rooftop representation, terrain-derived exposure factors, GIS-based solar-radiation modelling, rooftop suitability analysis, weather-aware photovoltaic estimation, and the computational organisation of multi-stage workflows [1], [2], [5]. These themes are closely interconnected. The way rooftops are represented determines which geometric properties can be extracted; these properties, in turn, influence how solar exposure is modelled; and the resulting solar information must ultimately be translated into deployment-oriented and simulation-oriented outputs if rooftop photovoltaic potential is to be meaningfully interpreted at the urban scale [2]– [5].

Within this broader context, the present thesis is concerned not only with rooftop photovoltaic estimation as an isolated modelling task, but also with the methodological structure required to support such estimation through a reproducible, workflow-oriented framework [1], [5], [6]– [10]. This makes it necessary to review the literature in a manner that moves gradually from the general urban significance of rooftop photovoltaics to the specific geospatial, radiative, and computational components required for rooftop-level analysis [1], [5].

The review is therefore organised as a connected progression. It begins by examining rooftop photovoltaics as an urban research domain and then considers the principal geospatial foundations of rooftop assessment, with particular emphasis on rooftop representation and terrain-derived exposure factors [3]– [5]. It then addresses GIS-based solar-radiation modelling, rooftop suitability and deployable installation areas, and weather-aware photovoltaic estimation [2]– [5]. Finally, it discusses computational workflows, web-based platforms, and service-oriented infrastructures, since rooftop photovoltaic analysis increasingly depends not only on the modelling techniques themselves, but also on the way these techniques are organised into coherent and reusable processing frameworks [6]– [10]. Through this progression, the chapter establishes the background necessary to position the contribution of the present thesis within the existing literature.

2.2 Urban rooftop photovoltaics as a research domain

Photovoltaic systems have become one of the most prominent technologies in the transition towards low-carbon energy systems. Their importance is especially evident in urban environments, where energy demand, building concentration, and infrastructure density are particularly high [1], [2]. In this context, rooftop photovoltaic systems are widely regarded as an attractive option because they allow renewable electricity generation to be integrated into existing built surfaces without requiring additional land [2], [5]. This gives rooftops a distinctive role in urban energy discussions, as they constitute a distributed and spatially embedded resource that can contribute to local electricity production while remaining physically tied to the existing urban fabric [1], [5]. At the same time, their growing relevance should also be understood within the broader expansion of clean energy transition strategies and the practical search for scalable urban renewable-energy solutions [11], [13].

Rooftop photovoltaics are therefore increasingly examined not only at the scale of individual buildings but at the scale of neighbourhoods, districts, and cities [1], [2]. Beyond a single

installation, rooftop PV becomes part of a wider urban-energy problem in which many rooftops, each with different geometric and environmental conditions, must be considered together [2], [4]. This matters because urban rooftop stocks are heterogeneous: roofs vary in size, shape, elevation, orientation, and surrounding context, and these differences directly affect their suitability for deployment [4], [14]. In dense settings, neighbouring buildings further modify local solar access in ways that coarse, generalised estimates cannot capture [4], [14].

This is one reason rooftop photovoltaic assessment has become a distinct research domain rather than a subtopic of general solar-energy studies. Because urban rooftops are fragmented, varied, and embedded in a built context that both enables and constrains deployment [1], [14], the relevant question is no longer whether a city receives enough solar radiation overall, but how that resource is distributed across specific rooftop surfaces and how urban form governs its conversion into usable generation [2], [4]. Methods designed for regional resource estimation remain useful for large-scale comparison but are often insufficient for rooftop-level variation across a real urban fabric [5], [14].

The growing relevance of rooftop photovoltaics is also closely connected to practical planning needs. In many studies, rooftop PV assessment is expected to address questions that are spatial, technical, and deployment-oriented at the same time: where photovoltaic systems can be installed, how much suitable surface is available, which rooftops are more promising than others, and how expected electricity generation may vary over time [2], [15]. This gives the field a strongly applied character. Rooftop photovoltaic analysis is intended not only for theoretical interpretation, but also for technical screening, urban energy planning, prioritisation, and support for implementation-oriented decision-making [1], [15].

Rooftop photovoltaic assessment also sits at the intersection of several fields: renewable-energy research, because it concerns distributed solar generation; urban-energy research, because it ties buildings and districts to the energy transition; and geospatial analysis, because rooftop conditions can only be assessed once the built environment is represented spatially [1], [5]. This breadth explains the methodological diversity of the literature, which ranges from urban-energy planning to rooftop geometry and solar access, and on to time-resolved, performance-oriented estimation [2], [4], [14].

Framed this way, rooftop photovoltaics are less a matter of placing panels on buildings than an urban-scale problem of turning built surfaces into distributed renewable-energy opportunities. This is central to the present thesis, which addresses rooftop-level assessment within a framework linking geospatial representation, solar modelling, and the analytical stages that follow [2], [4], [14]. A first issue then becomes unavoidable: how rooftops are represented in spatial terms. Before exposure, suitability, or performance can be analysed, the rooftop domain must be described through data structures that capture its geometry and urban context. The next section therefore turns to the geospatial foundations of rooftop representation.

2.3 Geospatial data and rooftop representation in PV assessment

A central issue in rooftop photovoltaic assessment concerns the way rooftop surfaces are represented spatially. Before solar exposure, suitability, or photovoltaic performance can be analysed, the built environment must first be translated into a geospatial form that is suitable for computation [1], [5]. For this reason, the quality, structure, and level of detail of the input data play a decisive role in shaping the analytical possibilities of rooftop PV studies. In practice, different representations of the urban surface do not simply provide alternative visual descriptions of the same environment. They also determine which rooftop characteristics can be extracted, which spatial constraints can be captured, and how accurately rooftop-level photovoltaic potential can be assessed [1], [4]. This dependence on spatial representation is also visible in broader integrated urban-modelling environments, in which the structure and granularity of geospatial inputs directly influence the fidelity of downstream analytical results [18].

At the simplest level, some studies begin from two-dimensional building representations such as cadastral maps or footprint layers [15], [16]. Bergamasco and Asinari, for instance, base a scalable rooftop-potential method on available roof-surface area extracted from such data and refined through ortho-image analysis [15], while Kabir et al. estimate usable rooftop area for PV across Dhaka from building outlines [16]. Footprint data are widely available and inexpensive, and in broad screening they support useful first-level estimates of building location and gross roof area [15]. They remain, however, a planar description: they do not encode rooftop three-dimensional structure, height variation, inclination, or the sur-

rounding context affecting solar exposure [1], [5], so their value diminishes once the objective shifts from inventory towards detailed rooftop-oriented assessment.

To clarify the main alternatives found in the literature, **Table 2.1** summarises the principal geospatial representation types commonly used in rooftop photovoltaic studies, together with their typical data sources, strengths, limitations, and analytical roles [1]– [3].

Typical use in rooftop PV studies	Main limitations	Main strengths	Typical data source	Representation type
Broad urban screening, gross rooftop-area estimation, linkage of rooftops to parcels or building units	Provide only 2D geometry, do not directly capture rooftop height, tilt, aspect, or surrounding obstructions	Widely available, easy to manage, suitable for first-level rooftop inventory and area estimation	Cadastral datasets, municipal building layers, parcel-linked building polygons	Building footprints / cadastral maps
Rooftop-oriented solar modelling, terrain-derivative extraction, detailed rooftop PV assessment	Quality depends on spatial resolution and preprocessing; may be computationally demanding at fine scale	Capture rooftop elevation and surface form, support derivation of slope, aspect, and exposure-related variables	Raster elevation products representing the upper surface of terrain and buildings	Digital Surface Models (DSMs)
Fine-resolution rooftop analysis, shading-aware PV assessment, building-scale or district-scale detailed studies	Data acquisition and processing may be expensive or unavailable for some areas	High geometric detail, strong suitability for rooftop morphology and urban obstruction analysis	Airborne or terrestrial LiDAR point clouds processed into DSM-like products	LiDAR-derived surface models
Roof segmentation, rooftop classification, geometric enrichment of urban rooftop datasets	Quality varies with source and reconstruction method; may require substantial preprocessing and assumptions	Can enrich rooftop interpretation beyond simple 2D outlines, useful where explicit rooftop form can be reconstructed	Aerial imagery, orthophotos, photogrammetric reconstruction, 3D city models	Ortho-image / photogrammetric / 3D-derived representations
Synthetic or reconstructed rooftop-surface generation for downstream solar and PV workflows	Usually rely on assumptions or indirect derivation, may simplify fine rooftop detail	Useful when high-resolution measured surfaces are not available; flexible under heterogeneous data conditions	Footprints combined with building-height attributes, raster-vector integration, derived elevation information	Enriched or derived building-based representations

Table 2.1. Main geospatial representation types used in rooftop photovoltaic assessment, together with their typical data sources, strengths, limitations, and analytical roles.

As shown in **Table 2.1**, rooftop representation in photovoltaic studies ranges from simple planar building layers to more explicitly surface-aware and elevation-aware models. The progression across these representation types is not merely technical. It reflects different levels of analytical ambition. Footprint-based approaches may be sufficient for broad inventory-level screening, whereas DSM-based, LiDAR-based, or otherwise enriched rooftop representations are more suitable when the objective is to analyse rooftop geometry, local exposure conditions, and deployment-relevant rooftop variation in greater detail [5], [14], [17]. The table therefore makes clear that rooftop representation should not be treated as a neutral preprocessing choice. It defines the geometric basis of the entire assessment workflow and directly influences what later stages of solar modelling and photovoltaic interpretation can realistically achieve.

For this reason, a substantial part of the literature has turned towards richer geospatial representations that capture the vertical and surface-related characteristics of the built environment more explicitly. Among these, digital surface models have played a particularly important role [5], [14]. Unlike a footprint layer, a DSM describes the elevation of the upper surface, including buildings and other above-ground objects, and therefore provides a spatial representation from which rooftop geometry can be analysed more directly. This is highly relevant in rooftop PV studies because solar exposure is not determined only by the horizontal extent of a roof, but also by its elevation profile, inclination, aspect, and relation to surrounding obstructions [5], [14]. A rooftop represented only as a polygon and the same rooftop represented as a surface model do not offer the same analytical possibilities. The latter makes it possible to derive geometric variables that are central to solar and photovoltaic analysis, whereas the former often requires additional assumptions or supplementary datasets before similar information can be approximated [1], [14].

Within this elevation-aware category, the literature draws on several data sources. Several studies derive DSMs from LiDAR: Lukac et al. assess buildings-roof PV potential directly from LiDAR returns [4], Brito et al. apply LiDAR to a Lisbon suburb [17], and Jakubiec and Reinhart combine LiDAR-derived geometry with hourly Daysim simulations to predict city-wide PV gains [3]. Others reconstruct rooftop geometry from photogrammetry or ortho-image analysis [15], [3]. Such approaches are most valuable in dense environments, where rooftops are rarely flat or geometrically simple, because they preserve the morphological complexity that governs rooftop-level solar access [4], [17]. Their cost is practical: availability, resolution, and preprocessing demands mean detailed 3D urban data are not obtainable for every study area [1], [16].

This has encouraged approaches that adapt rooftop representation to the available inputs rather than assuming an ideal dataset. Some enrich planar footprints with explicit building-height attributes [15], [16]; others infer rooftop elevation by combining raster and vector sources [1], [5]. Rooftop representation is thus not a neutral preprocessing step but a methodological decision with consequences for the whole workflow [5], [14]: a coarse representation may suffice for high-level screening, whereas detailed rooftop analysis requires more than simple building outlines [4], [14].

Scale matters as well. At city or regional scale, simplified representations may suffice when the goal is broad comparison rather than fine local differentiation [15], [16]. As the focus narrows to individual rooftop units, candidate installation surfaces, or time-dependent estimation, the need for explicit, locally meaningful representation grows [5], [4], especially in heterogeneous fabrics where adjacent buildings differ within the same district. There, assessment depends not on the mere presence of a roof but on how that roof is modelled within its wider surface and spatial context [14], [17].

Geospatial representation therefore amounts to more than data description: it defines the geometric basis from which later variables are derived, conditions the quality of solar-exposure modelling, and sets the level of detail at which suitability and performance can be interpreted [5], [14]. It is, in effect, the foundational layer of the analytical chain, with different datasets supporting different degrees of realism, granularity, and assumption about rooftop form and surrounding structure [1], [16]. Open-source environments such as GRASS GIS have helped make these representation choices usable within reproducible spatial workflows [19].

For the present thesis, this discussion is particularly relevant because the methodological framework is designed to accommodate more than one geospatial entry condition. Rooftop analysis may begin from a direct high-resolution surface representation or from building-based data that require further transformation before a usable rooftop-oriented surface can be obtained. This flexibility reflects a broader issue already visible in the literature: rooftop photovoltaic assessment often has to operate under heterogeneous data conditions rather than under a single idealised input scenario [1], [5]. Once the rooftop domain has been represented in a geospatially meaningful way, the next analytical question concerns the variables that can be derived from that representation. Rooftop assessment does not end with surface description alone. It requires the extraction of geometric and exposure-related factors that help explain how different rooftop surfaces interact with incoming solar-radiation. For this reason, the next section turns to terrain-derived and solar-exposure factors, with particular attention to the role of slope, aspect, and related geometric conditions in rooftop photovoltaic analysis.

2.4 Terrain-derived factors and rooftop solar exposure

Once rooftop surfaces have been represented, the next stage concerns the variables that describe how those surfaces are positioned and exposed within the urban environment. A rooftop is not analytically relevant merely because it exists; its photovoltaic value depends on its geometric form and on how far that form allows access to incoming solar radiation. Rooftop studies therefore emphasise terrain-derived and exposure-related factors obtained from elevation-aware representations such as digital surface models [1], [5], [14].

Slope and aspect are the two most frequently used factors: slope is the local inclination of the surface, aspect its orientation. Both shape how solar radiation reaches the roof through the day and year [1], [2], [17], so a roof section with a favourable tilt and orientation can receive markedly different exposure from a neighbouring section facing elsewhere. Rooftop-level analysis therefore differs from general resource estimation precisely because the receiving surface is not geometrically neutral [2], [14].

The importance of slope and aspect is especially evident in urban environments, where rooftop forms are rarely homogeneous. Some roofs may be relatively flat, others pitched, and others composed of multiple segments with different orientations. Even when buildings are adjacent, their rooftops may respond differently to solar exposure because of local differences in shape or azimuth. For this reason, many studies treat slope and aspect not merely as auxiliary descriptors, but as fundamental analytical variables that shape later stages such as suitability screening, solar modelling, or photovoltaic estimation [5], [2], [4]. A surface that is geometrically available may still be less attractive from a photovoltaic perspective if its orientation or inclination reduces the useful solar input reaching it.

Surface geometry alone, however, does not explain rooftop exposure. Urban rooftops sit within a built context that can obstruct the sky and cut solar access from particular directions: a roof that looks suitable in terms of slope and aspect may in practice be shaded by neighbouring buildings or rooftop structures [5], [14]. The literature therefore extends assessment beyond local geometry to directional exposure, typically through GIS measures of sky visibility, shadowing, or horizon obstruction that describe how the surroundings constrain radiation from each azimuth [1], [3], [14].

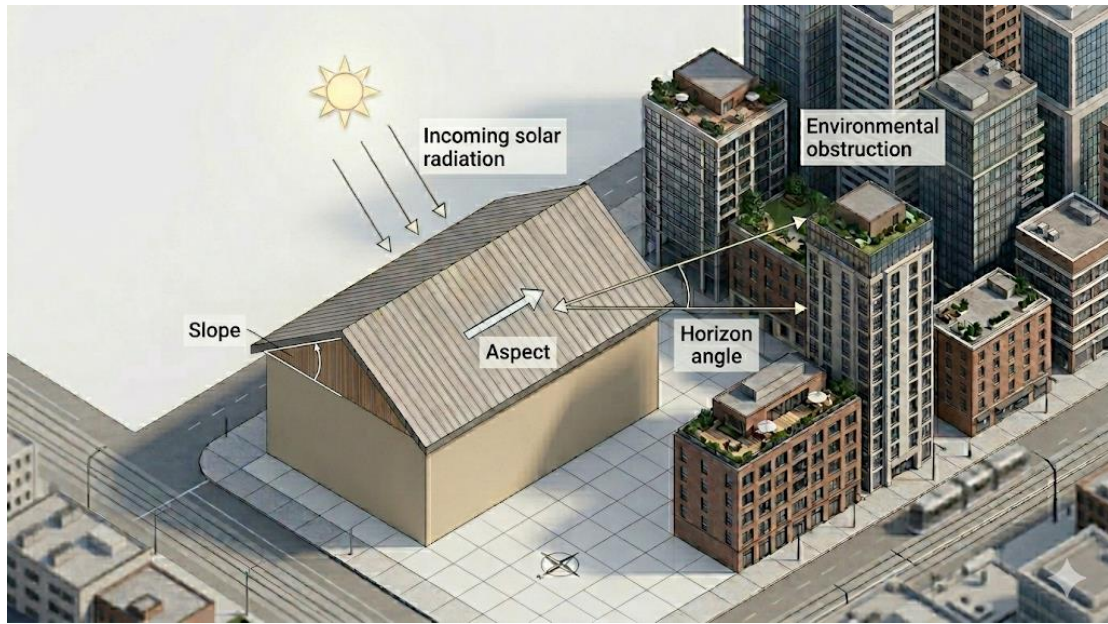


Figure 2.1. Rooftop slope, aspect, and directional obstruction in urban PV assessment.

Source: Author's elaboration based on the reviewed literature.

The conceptual elements illustrated in [Figure 2.1](#) summarise the three main components discussed above. First, rooftop slope defines the inclination of the receiving surface. Second, aspect indicates the direction towards which that surface is oriented. Third, the surrounding built environment introduces directional obstruction, expressed here through the horizon angle, which limits the portion of the sky dome from which direct solar radiation can reach the roof. The figure is therefore not intended as a technical measurement scheme, but as a visual synthesis of the analytical relationship between rooftop geometry, directional exposure, and urban obstruction. In this sense, it clarifies why rooftop photovoltaic assessment cannot rely on rooftop area alone but must also account for how the roof is oriented and how open or obstructed its surrounding sky conditions remain.

Directional obstruction adds a second layer of interpretation. Slope and aspect describe the surface itself but say nothing about whether the sky is open in the relevant directions – a distinction that matters most in dense fabrics, where a well-oriented roof can still lose direct radiation to nearby structures. Rooftop exposure thus emerges from the interaction between local surface properties and the wider spatial setting [\[3\]](#), [\[14\]](#), and the literature moves correspondingly from simple geometry towards explicit modelling of exposure conditions.

This broader view has three consequences for assessment. It is more realistic, since roofs of similar geometry are no longer treated as equally promising. It explains why detailed rooftop analysis needs more than a two-dimensional inventory: without support for geometric derivation and directional exposure, key differences between roofs stay hidden [5], [17]. And it bridges urban morphology and solar modelling, since once geometry and sky conditions are characterised, radiation can be modelled to reflect not only general climate but local, surface-specific exposure [1], [4].

At the same time, the use of terrain-derived variables introduces methodological choices. Different studies may apply different thresholds, different levels of geometric detail, and different assumptions about what constitutes a favourable rooftop condition. Some approaches use slope and aspect primarily to exclude clearly unsuitable surfaces, whereas others incorporate them more directly into subsequent simulation stages [2], [17]. Similarly, directional exposure may be represented through different shadowing or horizon-based techniques depending on the data and software environment available [1], [3]. This variation reflects the broader diversity of the field, but it also confirms a common underlying idea: rooftop photovoltaic analysis depends on the extraction of interpretable geometric and exposure-related variables from the urban surface.

For this thesis, these variables form the bridge between rooftop representation and solar-radiation modelling: a rooftop surface becomes useful only when its geometric and directional properties are translated into exposure information. Terrain-derived variables are therefore not secondary outputs but the basis of the next stage. The following section examines GIS-based solar-radiation modelling and how rooftop geometry and obstruction are turned into explicit representations of solar input.

2.5 GIS-based solar-radiation modelling in urban rooftop studies

With geometry and exposure established, the focus shifts to modelling solar radiation itself, because geometric interpretation alone does not yet quantify the solar input available to a roof. A roof may be large, well oriented, and apparently unobstructed, yet the distribution of radiation across it must still be computed. GIS-based solar-radiation modelling has become central to this task, building in particular on physically based models such as r.sun, introduced by Šuri and Hofierka for GIS-based solar assessment [20] and applied to urban PV potential by Hofierka and Kanuk [14].

The importance of GIS in this context lies in its ability to connect spatial data, surface geometry, and solar computation within a common analytical environment. In urban studies, solar radiation is not treated as a uniform climatic quantity applied equally across all rooftops. Instead, it is modelled as a spatially differentiated phenomenon whose local variation depends on surface inclination, orientation, surrounding obstructions, and the broader geometry of the built environment [14], [3]. GIS-based approaches are particularly well suited to this task because they make it possible to compute solar-radiation over spatially explicit surface representations while preserving the local variability that is often critical in rooftop-level analysis. In practice, this line of work has also been supported by established open-source geospatial environments such as GRASS GIS, which have provided an operational basis for spatially explicit solar-radiation modelling and related environmental analysis [19].

Many studies use GIS methods to estimate rooftop solar potential, typically starting from DSMs or LiDAR-derived surfaces. Hofierka and Kanuk, for example, combine open-source radiation tools with urban surface data to map rooftop potential [14], and Brito et al. derive roof-level estimates from LiDAR [17]. The combination of geospatial data and solar models moves analysis beyond regional irradiance averages, producing radiation maps and roof-level suitability layers that show how uneven solar availability is within a single city [17], [20].

Not all approaches share the same temporal resolution or ambition. Some report long-term summaries – annual or monthly potential – which give a useful first approximation across large areas and underpin web tools such as PVGIS [8] and PVWatts [9]. These are effective for screening and city-scale comparison, but by compressing the temporal dynamics of exposure into aggregates they limit how far rooftop conditions can later feed time-dependent photovoltaic simulation [9], [20].

For this reason, more detailed spatio-temporal approaches have become increasingly important in the literature. Rather than relying only on annual or monthly aggregates, these methods preserve shorter time intervals and model how radiation changes throughout the day and across representative periods of the year [3], [21]. This temporal dimension is especially relevant when rooftop photovoltaic assessment is intended to support not only static solar-potential mapping, but also subsequent estimation of photovoltaic behaviour under varying operating conditions. Once time is introduced more explicitly into the analy-

sis, rooftop solar modelling becomes more than a descriptive cartographic exercise. It begins to function as a structured basis for later analytical stages [20], [21].

A further distinction concerns the components of radiation. Rather than treating incident radiation as one quantity, many studies separate direct and diffuse contributions [3], [14], because the direct component is strongly shaped by geometry and obstruction while the diffuse component follows a different spatial pattern. Keeping the two apart improves interpretability and eases later integration with weather-aware modelling; it reflects a wider tendency to represent rooftop solar conditions in physically interpretable rather than purely aggregated terms [3].

GIS-based modelling also involves a recurring trade-off between realism and practicality: detailed models capture rooftop-level variation and the effect of geometry and obstruction, but urban-scale computation grows costly when many roofs, timesteps, and high-resolution surfaces are combined [20], [21]. The literature accordingly spans both detailed modelling environments and more accessible platform-oriented solutions, summarised in Table 2.2, ranging from research models to user-facing services such as PVGIS [8], iGUESS [22], I-SCOPE [23], and Mapdwell [26].

Typical output	Scale / resolution	Main focus	Tool / platform
Beam and diffuse irradiance maps	Fine spatial detail, workflow-dependent time resolution	Spatial solar-radiation modelling	r.sun / GIS-based solar modelling
Irradiation and PV summaries	Broad geographic coverage, mainly aggregated outputs	Web-based solar and PV estimation	PVGIS
PV production estimates	System-oriented, hourly or aggregated	Simplified PV performance estimation	PVWatts
Solar-related planning outputs	Multi-scale urban planning	Urban energy planning support	iGUESS
Solar maps and urban-service outputs	Urban platform scale	Smart-city solar services	I-SCOPE
Roof solar-potential outputs	Building-roof screening level	Roof-level solar potential mapping	Brumen et al. application
Energy-yield and system outputs	Strong temporal modelling, limited GIS rooftop focus	Detailed PV system simulation	SAM

Table 2.2. Representative tools and platforms in rooftop solar and photovoltaic assessment.

As shown in Table 2.2, rooftop solar and photovoltaic assessment has developed through several computational forms rather than through a single dominant modelling tradition. Some tools are primarily designed to compute spatially explicit solar-radiation fields, while

others focus on the accessible delivery of solar and photovoltaic information through web platforms, and others again emphasise system-level photovoltaic performance simulation. This distinction is important because it shows that rooftop PV analysis increasingly depends not only on how solar conditions are modelled, but also on how these modelling capabilities are organised, delivered, and reused within broader analytical environments [8], [9], [22]– [25].

For this thesis, solar-radiation modelling is not an end in itself but an intermediate stage: its role is to turn rooftop geometry and exposure into a structured radiative basis for later processing. It matters less as a map-making step than as the bridge between geospatial pre-processing and rooftop-level photovoltaic estimation.

Once solar radiation has been modelled over rooftop surfaces, the next question is whether all of those surfaces should be treated as equally relevant for deployment. In practice, rooftop photovoltaic assessment often moves beyond radiation modelling alone and begins to identify which portions of the rooftop domain remain suitable for actual photovoltaic installation. For this reason, the next section turns to rooftop suitability and the identification of deployable installation areas.

2.6 Rooftop suitability and deployable installation areas

Rooftop assessment does not end with radiation modelling. Even with exposure modelled explicitly, a further step is needed to decide which parts of the rooftop domain are realistically suitable for installation, and much of the literature treats suitability as a distinct stage [2], [14]. The question becomes not how much radiation reaches the surface, but which portions qualify as deployable installation areas under geometric, spatial, and practical constraints.

Solar availability and deployment potential are not equivalent. A roof section may receive good radiation yet be unusable because of limited size, poor inclination, awkward orientation, or fragmentation [2], [3]; conversely, a promising building may offer only a few realistically usable sub-areas. Many studies therefore treat the roof not as one homogeneous unit but as a set of candidate installation surfaces [14], [15], turning a general rooftop-solar problem into a deployment-oriented one.

A common strategy in the literature is to define suitable rooftop areas by combining terrain-derived variables with rooftop masks or building-based constraints. Among the most frequently used criteria are slope and aspect, since these variables directly influence the expected photovoltaic usefulness of a surface [14], [4]. Rooftop sections with very steep inclinations, highly unfavourable azimuths, or implausible surface conditions are often excluded from the candidate set, while geometrically more favourable portions are retained for further analysis. In many cases, this selection is implemented not at building level but at the level of rooftop cells, pixels, or segmented sub-surfaces, which allows suitability to vary internally across the same roof [2], [4]. This point is important because it reflects a broader methodological shift in rooftop PV studies: the unit of interpretation increasingly moves from the building to the deployable rooftop area.

Available area is another recurring criterion. Suitability is rarely defined by orientation and irradiance alone; it also depends on whether a region is large and coherent enough to host panels [3], [15]. Some studies apply minimum-area filtering or remove fragmented patches too small for deployment [14], [24]; others retain the link between candidate surfaces and their parent buildings or parcels [15]. The aim is consistent: outputs that are not only solar-relevant but installation-relevant.

Suitability is defined at varying levels of complexity. In some studies, it is a simple screening result from threshold filters on slope, aspect, or exposure [4], [14]; in others it is embedded in a wider framework where suitable areas become the units for later estimation, aggregation, or planning [2], [24]. The difference is one of role: an endpoint in screening studies, but an intermediate analytical entity in workflow- or simulation-oriented frameworks.

This transition from rooftop description to deployable installation units is particularly relevant in urban contexts, where rooftop geometries are often fragmented, internally heterogeneous, and constrained by the surrounding built form [2], [3]. It is therefore not sufficient to state that a building has photovoltaic potential in general. What is usually needed is a more refined identification of the specific rooftop portions that can support installation and can then serve as meaningful carriers of subsequent photovoltaic calculations [14], [15]. In this sense, rooftop suitability analysis introduces a practical interpretative layer into the literature: it converts solar-exposed rooftop surfaces into analytically and operationally relevant candidate areas.

This thesis does not stop at rooftop solar modelling: it explicitly identifies candidate installation surfaces and uses them as structured units in the downstream stages. Suitable rooftop areas thus bridge geospatial and radiative processing on one side and photovoltaic estimation on the other. The next section turns to weather-aware and time-resolved estimation, where suitability and solar information meet actual photovoltaic performance.

2.7 Weather-aware and time-resolved photovoltaic estimation

Solar-radiation modelling represents surface exposure but not full photovoltaic behaviour under real conditions. A roof may receive favourable radiation in radiative terms, yet output still depends on atmospheric variability, irradiance composition, temperature, and the temporal structure of conditions. Much of the literature therefore moves beyond static or clear-sky assessment to weather-aware, time-resolved estimation [2], [21].

Photovoltaic output is inherently time-dependent: even with fixed geometry, incident conditions vary through the day and year. This matters when the goal is not broad availability but rooftop-level behaviour under realistic conditions [2], [9]. Annual or monthly aggregates remain useful for screening but fall short when later stages need time-stamped performance, comparison across weather states, or integration with energy-oriented workflows [21], [27].

A second issue is irradiance composition. Weather records usually provide global horizontal radiation, whereas detailed PV modelling needs direct-normal and diffuse-horizontal components. Direct radiation is tied to solar position, orientation, and obstruction, while diffuse radiation behaves differently and can remain significant when direct radiation is reduced. A body of work therefore develops decomposition models that estimate the two components from global irradiance – for example the correlations of Erbs et al. [28], Reindl et al. [31], and Skartveit and Olseth [32], later compared against measurements by Ineichen [33].

No decomposition model is universally optimal; performance shifts with climate, latitude, temporal resolution, and atmospheric conditions [29], [34]. Weather-aware estimation is thus not simply a matter of adding meteorological data, but of how irradiance inputs are represented and transformed before use. The underlying physics follows standard solar-engineering treatments [39], and validation studies confirm that diffuse-fraction estimation stays strongly model-dependent across climates, resolutions, and data conditions [40], [41].

Module temperature also affects estimation. Conversion efficiency depends not only on incident irradiance but on the module's thermal state, which in turn responds to ambient temperature, wind, and system behaviour [35], [36]. Mattei et al. estimate module temperature through a simple energy-balance method [35], while Schwingshackl et al. show the marked influence of wind on module temperature [36]. The choice of model shapes how strongly output responds to environmental conditions under similar radiative input. As a result, rooftop PV modelling gradually shifts from pure solar-exposure analysis towards a broader representation of system behaviour under weather-driven operating conditions.

This perspective ties into time-resolved simulation. Once the aim is generation profiles rather than static potential, the question changes from where conditions are favourable to how a rooftop system behaves over time, how output responds to meteorological states, and how that behaviour can be organised for later use [2], [9], [27]. Time-resolved estimation thus bridges rooftop solar-resource analysis and rooftop energy modelling.

There is, again, a balance between physical detail and tractability. Weather-aware, time-resolved approaches add realism but also demand more meteorological inputs, stronger assumptions about decomposition and temperature, and a more explicit computational organisation [34], [37], [38]. Rooftop PV estimation is therefore as much a problem of methodological structuring as of physics, and studies strike different balances according to purpose, data, and intended downstream use.

In this thesis, photovoltaic estimation is not an isolated final calculation but a downstream stage building on rooftop geometry, solar-radiation modelling, and structured intermediate outputs while incorporating time-varying weather. Weather-aware estimation matters here not only for realism but because it connects earlier exposure analysis to rooftop-level energy estimation within an organised workflow.

Once photovoltaic estimation is considered in this way, a further issue becomes important: how the underlying modelling stages are organised, stored, and delivered within broader computational systems. Rooftop PV analysis increasingly depends not only on the modelling choices themselves, but also on the infrastructures through which these modelling steps are connected, reused, and exposed. For this reason, the next section turns to computational workflows, web-based platforms, and service-oriented infrastructures in rooftop photovoltaic assessment.

2.8 Computational workflows, web-based platforms, and service-oriented infrastructures

As rooftop photovoltaic assessment has become more detailed and data-intensive, the methodological challenge has extended beyond modelling techniques alone. In earlier studies, the central concern was often how to estimate rooftop solar potential or photovoltaic opportunity across a given urban area. However, as workflows became more spatially explicit, more time-dependent, and more strongly connected to heterogeneous data sources, an additional issue emerged: how the different analytical stages should be organised, executed, and delivered within broader computational systems [6], [22]. In this sense, rooftop photovoltaic assessment is no longer only a question of physical or geospatial modelling, but also of computational workflow design.

A first line of work concerns web-based access to geospatial and solar capabilities. Rather than standalone desktop tools, several systems expose solar and geospatial functions as web services or platforms: Li et al. share GRASS algorithms in a service-oriented environment [6], Qiu et al. build the GWASS web application on GRASS [7], and platforms such as I-SCOPE deliver urban solar services [23]. A local, isolated modelling chain has limited reusability, whereas a service-oriented one supports repeated execution, broader access, and integration with other tasks [8], [23] – consistent with established principles of web architecture such as REST [46] and OGC interoperability standards [47].

One important example is the use of geospatial web-service environments for exposing GIS algorithms and workflows. Studies such as those by Li et al. and Qiu et al. show that geoscience and GRASS-based algorithms can be organised within service-oriented environments rather than remaining confined to local software execution [6], [7]. In these cases, the focus is not specifically limited to rooftop photovoltaics, but the broader implication is highly relevant: solar and rooftop workflows can be structured as reusable services rather than as one-off scripts or manual desktop operations. This idea is particularly important for multi-stage photovoltaic assessment, where repeated execution, controlled inputs, and workflow modularity can strongly affect reproducibility and usability.

A second line of development concerns broader web platforms and urban systems that incorporate solar or photovoltaic analysis within larger decision-support environments. Platforms such as *PVGIS*, *iGUESS*, *I-SCOPE*, and related systems illustrate how solar and

photovoltaic information can be embedded within user-facing tools, planning interfaces, or interoperable smart-city services [8]– [42]. These systems differ substantially in purpose and methodological depth. Some are primarily oriented towards broad public access to solar and PV information, while others are intended to support urban-scale planning, energy assessment, or multi-scale spatial decision-making. To clarify these distinctions more explicitly, **Table 2.3** summarises a set of representative computational approaches and platforms discussed in the literature, with attention to their main role, delivery type, and relevance to rooftop photovoltaic workflows [6]– [45].

System / Approach	Main Role	Delivery Type	Main Relevance
Li et al. / GWASS	Geospatial algorithm sharing and service exposure	Web service / web GIS	Reusable geospatial processing
PVGIS	Solar and PV information access	Web platform	Public-facing solar and PV estimation
iGUESS	Urban energy planning support	Web-based planning system	Multi-scale urban energy analysis
I-SCOPE	Smart-city solar services	Interoperable urban platform	Urban-scale service integration
Microservices literature	Modular workflow organisation	Software-architecture principle	Supports staged backend design
Bottaccioli et al. Infrastructures	Distributed PV workflow organisation	Service/infrastructure	Closest conceptual precedent to this thesis

Table 2.3. Representative computational approaches and platforms related to rooftop solar and photovoltaic assessment

As shown in **Table 2.3**, the computational landscape surrounding rooftop solar and photovoltaic assessment is not limited to one type of system. Some contributions focus on exposing geospatial processing capabilities as reusable services, others on providing user-facing solar or photovoltaic information through web platforms, and others on structuring broader distributed or modular infrastructures for workflow execution [6], [45]. This distinction is important because it shows that rooftop PV analysis increasingly depends not only on modelling techniques themselves, but also on how those techniques are delivered, connected, and reused across different computational settings. Related work has also explored integrated tool chains and urban-scale modelling environments in which photovoltaic analysis forms part of a wider computational ecosystem rather than remaining a standalone task [48], [18].

This also raises the importance of organising multi-stage workflows. Rooftop analysis usually involves a sequence of interdependent operations — input handling, geospatial prepro-

cessing, solar modelling, suitability detection, estimation — each with different data structures and computational needs. The value of a framework then lies not only in the correctness of each stage but in how stages are linked, stored, and made reusable [10], [22], [43]. Reproducibility in such workflows depends as much on how intermediate results are preserved and execution controlled as on the underlying models.

These concerns connect to service-oriented and modular software design. Service-oriented and microservice approaches, as set out by Fowler and Lewis [10] and Newman [43], emphasise loose coupling, modular execution, and clear separation of responsibilities. The rooftop PV literature rarely uses this vocabulary, but the need is increasingly evident: once analysis spans multiple stages, heterogeneous data, and repeated execution, monolithic implementations become hard to maintain, extend, and reuse, whereas a modular structure gives clearer boundaries between data ingestion, geospatial processing, weather handling, and simulation.

This is particularly relevant in workflows that combine computationally intensive geospatial processing with later analytical or simulation-oriented stages. Raster-based preprocessing, rooftop-geometry extraction, weather-aware photovoltaic estimation, and urban-scale result handling do not always have the same runtime characteristics or storage requirements. For this reason, a computational infrastructure capable of separating, coordinating, and re-connecting these stages becomes methodologically valuable rather than merely technically convenient [22], [44], [45]. In such cases, the computational architecture itself begins to influence the scientific usability of the workflow by determining whether intermediate products can be reused, whether different analytical branches can be supported, and whether the full process remains traceable and reproducible across repeated runs. This is particularly relevant when a single framework must accommodate different balances between computational detail, execution cost, and downstream reuse.

For this thesis, the framework is not a single isolated model but a staged, implementation-aware workflow connecting authentication and access control, geospatial preprocessing, solar-radiation modelling, structured intermediate artefacts, weather-aware processing, and rooftop-level estimation. The literature on web-based platforms, reusable geospatial services, and modular infrastructures thus underpins its methodological choices, and the

contribution lies as much in how the modelling stages are organised into a coherent, reusable, service-oriented framework as in the stages themselves.

2.9 Research gap and positioning of the thesis

The literature reviewed in the previous sections shows that urban rooftop photovoltaic assessment has developed along several important lines of research. Existing studies have made substantial contributions to the representation of rooftop geometry, the derivation of exposure-related variables, the modelling of solar radiation in urban contexts, the identification of suitable rooftop areas, and the estimation of photovoltaic performance under different temporal and environmental conditions [1]– [5]. At the same time, web-based platforms, geospatial services, and broader computational infrastructures have demonstrated that rooftop and solar analysis is increasingly shaped not only by modelling itself, but also by the ways in which analytical capabilities are delivered, reused, and integrated into wider systems [6]– [10].

This body of work is, however, fragmented in emphasis. Some studies focus on rooftop representation and solar-potential mapping, others on performance modelling, web-based delivery, or planning support [1], [3], [6], [7]. Valuable as these are, they typically address only part of the chain: rooftop assessment is presented either as a geospatial exercise or as a photovoltaic estimation problem, rather than as a reusable, multi-stage workflow linking heterogeneous inputs, preprocessing, suitability, structured intermediate products, and weather-aware output generation.

A related limitation is the treatment of intermediate stages. Many studies emphasise final solar or photovoltaic outputs and give less attention to the reusable intermediate products that connect one stage to the next. Yet these are not mere by-products: they affect reproducibility, tractability, traceability, and the ability to reuse earlier results under changing downstream conditions [2], [44], [45] – a point that becomes critical as assessment moves from static mapping towards weather-aware, time-resolved simulation.

Another point emerging from the literature is that rooftop photovoltaic workflows may benefit from more than one computational path. Detailed geospatial and photovoltaic pipelines can provide rich and physically meaningful outputs, but they may also become computationally intensive, data-demanding, and operationally burdensome when high spatial detail,

dense temporal resolution, and repeated scenario evaluation are required [2], [4], [5]. At the same time, simpler or more aggregated approaches may improve efficiency and accessibility, but risk losing part of the spatial and analytical richness needed for rooftop-level interpretation. This suggests that the methodological challenge is not only to improve rooftop PV modelling accuracy in isolation, but also to design frameworks capable of preserving coherence while accommodating different balances between analytical detail, data availability, and execution efficiency. More broadly, the literature on computational infrastructures for photovoltaic assessment indicates that modular and distributed workflow design can play an important role in supporting this balance, particularly when rooftop-level modelling must remain reusable across different analytical conditions [44].

The present thesis is positioned at the intersection of rooftop photovoltaic modelling and computational workflow design. Rather than a single isolated calculation, it develops a staged, implementation-aware framework linking geospatial input handling, terrain and solar preprocessing, suitability analysis, structured artefacts, weather-aware processing, and rooftop-level estimation within a backend-oriented workflow. Its contribution lies as much in this reproducible, reusable organisation as in the individual stages. The approach extends earlier GIS-based PV infrastructures operating at fine spatio-temporal resolution [49] towards a broader staged workflow built around reusable analytical layers and backend orchestration.

The framework is also designed for heterogeneous inputs rather than one idealised configuration. Assessment may begin from direct surface representations or from building-based inputs needing transformation before analysis, and the workflow does not depend on a single monolithic simulation path: it supports staged processing and alternative branches responding to different demands of detail, speed, and reuse. This flexibility reflects a practical reality in the literature, where rooftop assessment must operate under varying combinations of data quality, computational cost, and objectives.

For these reasons, the present thesis addresses a gap that is methodological as much as analytical. It does not seek merely to add another rooftop photovoltaic model to the literature. Rather, it proposes a structured framework in which rooftop representation, solar modelling, suitability analysis, and weather-aware photovoltaic estimation are connected through reusable intermediate layers and explicit workflow organisation. In doing so, it po-

sitions rooftop photovoltaic assessment not simply as a sequence of isolated modelling tasks, but as a coherent computational process whose design has direct implications for reproducibility, interpretability, and subsequent analytical reuse.

2.10 Summary

This chapter reviewed the literature on urban rooftop photovoltaic assessment and showed that the problem extends well beyond general solar-resource description. Rooftop PV analysis depends on several interconnected components: geospatial representation, terrain-derived exposure factors, GIS-based radiation modelling, suitability analysis, weather-aware and time-resolved estimation, and the computational organisation of multi-stage workflows. Together these show that assessment is not one modelling task but a structured process in which representation, geometry, radiative modelling, and downstream estimation are closely interdependent.

The review also identified a methodological gap. Existing studies contribute across each area but usually address only part of the chain, creating a need for frameworks that link heterogeneous inputs, geospatial preprocessing, reusable intermediate artefacts, suitability, weather-aware processing, and rooftop-level estimation within a coherent, reproducible workflow. The next chapter presents the methodological framework developed here and how these components are organised in the proposed backend-oriented system.

Enabling Technologies

3.1 Introduction

The proposed rooftop photovoltaic simulation framework is built on a combination of geospatial, photovoltaic, data-management, and backend technologies that together make its analytical workflow possible. While the previous chapter reviewed the broader literature and positioned the thesis within existing research, the purpose of the present chapter is different. Rather than discussing the state of the art at a general level, it introduces the principal technologies that enable the framework itself and clarifies why they matter for this kind of system.

This distinction is important because the framework developed in this thesis is not a single isolated model or script. It is a backend-oriented computational system that integrates terrain-related geospatial processing, solar-radiation modelling, weather-aware photovoltaic estimation, the generation of structured analytical artefacts, and service-based execution. For that reason, understanding the methodological choices developed later in the thesis requires familiarity not only with the analytical workflow, but also with the technological components that support it. These components are not presented here as a generic software stack. Instead, they are introduced selectively in terms of the specific capabilities they contribute to the proposed framework.

Accordingly, the chapter is organised around five technology groups that are central to the system. It begins with the geospatial data representations on which rooftop analysis depends, including the distinction between raster and vector forms and the role of terrain-oriented surface models. It then examines the geospatial computation stack formed by *GRASS GIS*, the *r.sun* radiation model, and Actinia as the service-based execution environment. The chapter next turns to the photovoltaic and weather-related technologies that support irradiance and power estimation, including *pvlb*, weather-data sources, and external solar-resource services. This is followed by the technologies used for structured data per-

sistence and analytical artefact management, and finally by the backend and deployment technologies that support the framework as an operational service.

The aim of this chapter is therefore not to explain how each stage of the framework is executed in practice, since those implementation details are presented in the methodology chapter. Rather, it provides the technical background needed to understand the kinds of technologies on which the framework relies, the conceptual role played by each of them, and the way they collectively support a reproducible and deployable rooftop photovoltaic simulation system.

3.2 Geospatial data representations used in the framework

The proposed rooftop photovoltaic framework operates with geospatial data representations that capture both the discrete geometry of buildings and the continuous spatial structure of the urban surface. This distinction is important because rooftop photovoltaic analysis cannot rely on building location alone. Solar exposure, shading, and roof suitability depend on surface geometry, rooftop elevation, and the spatial relationship between buildings and their surroundings. For this reason, the framework relies on both vector and raster representations, each serving a different analytical purpose.

Vector data are used to represent discrete urban objects such as building footprints and candidate rooftop areas. In this form, buildings are described as bounded geometric entities with explicit spatial identity. This is important because photovoltaic suitability and simulation results must ultimately be linked to meaningful rooftop units rather than to anonymous grid cells. Vector representations therefore support object-level interpretation and allow the framework to preserve the connection between geometric entities, their attributes, and the final rooftop-level outputs.

Raster data, by contrast, are used to represent spatially continuous surfaces and derived fields. Elevation models, slope and aspect layers, horizon-related outputs, and solar-radiation surfaces are naturally expressed in raster form because they vary continuously across space. This makes raster representation especially suitable for terrain-oriented preprocessing and solar-exposure analysis, where the framework must evaluate local surface conditions cell by cell over the study area. In the context of rooftop photovoltaic assessment,

raster-based representation is therefore essential for deriving the surface properties that later influence suitability detection and photovoltaic estimation.

Within this raster-based context, a central distinction is that between a DTM and a DSM. A DTM represents the bare-earth surface, excluding above-ground objects such as buildings and vegetation, whereas a DSM represents the elevation of the uppermost visible surface, including rooftops and other surface features. For rooftop photovoltaic analysis, this difference is fundamental. A bare-terrain model may be sufficient for topographic description, but it does not represent the elevated urban surfaces on which photovoltaic systems are actually installed. By contrast, a DSM preserves the rooftop surface geometry required for the later derivation of slope, aspect, and obstruction-related quantities.

This distinction becomes especially important in a framework that supports more than one input condition. Where a sufficiently detailed real DSM is available, it can provide the geometric basis for rooftop-oriented analysis directly. Where such a surface model is not available, a rooftop-oriented surface representation must instead be constructed indirectly from building-based inputs. In both cases, the requirement is essentially the same: the framework must operate with a surface representation that preserves elevation, rooftop form, and surface continuity in a way that supports downstream solar and photovoltaic computation.

The relationship between raster and vector data is therefore not incidental, but methodologically necessary. Raster representations provide the continuous geometric context required for terrain and solar preprocessing, while vector representations provide the discrete spatial units through which rooftop conditions can be interpreted, filtered, and aggregated. The framework depends on both forms because rooftop photovoltaic analysis requires a transition from surface-based spatial modelling to rooftop-level analytical units. This dual representation makes it possible to preserve spatial detail without losing building-level interpretability, and it forms one of the basic geospatial foundations on which the rest of the framework is built. **Figure 3.1** provides a conceptual illustration of the distinction between DTM and DSM, together with the different analytical roles of raster and vector representations within the framework.

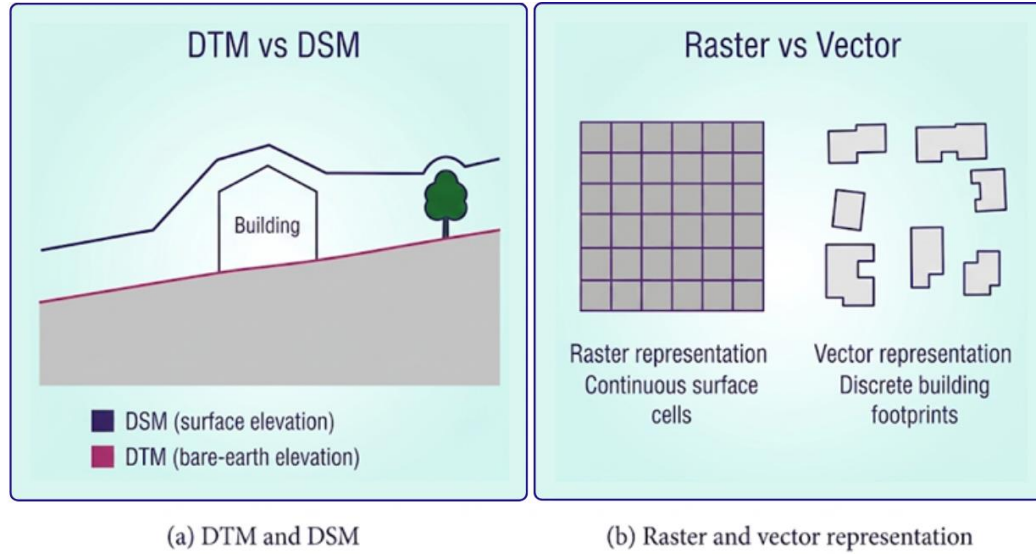


Figure 3.1. DTM vs. DSM and raster vs. vector representations in the framework.

3.3 GRASS GIS, *r.sun*, and Actinia

The geospatial computation layer of the proposed framework is built around three closely related enabling technologies: GRASS GIS, the *r.sun* solar-radiation model, and Actinia as the service-oriented execution environment through which geospatial processing is managed. Together, these technologies provide the computational basis for terrain preprocessing, horizon-aware solar analysis, and the controlled execution of raster-based operations within the broader rooftop photovoltaic framework.

GRASS GIS plays a central role because the framework depends heavily on raster-oriented geospatial computation. Rooftop photovoltaic analysis requires operations that act directly on elevation surfaces and derived spatial fields, including terrain smoothing, slope and aspect derivation, horizon analysis, and clear-sky solar-radiation modelling. GRASS GIS is particularly well suited to this kind of workflow because it provides a mature environment for high-resolution raster processing, a consistent spatial-workspace structure based on locations and mapsets, and a broad set of validated modules for terrain and solar analysis. In the context of the present framework, its importance lies not simply in being a GIS platform, but in functioning as the primary computational engine for the geospatial stages of the workflow.

Within this stack, the *r.sun* module is especially important because it provides the clear-sky solar-radiation modelling capability on which later photovoltaic estimation depends. Con-

ceptually, *r.sun* is a physically based solar model that computes surface irradiance by combining solar geometry with terrain-related inputs such as slope, aspect, and topographic obstruction. This is a critical enabling capability for the framework, since rooftop photovoltaic analysis requires more than geometric rooftop representation alone. It also requires an explicit radiative interpretation of the urban surface, so that rooftop units can be analysed not only in terms of form, but also in terms of their potential solar exposure under clear-sky conditions. In this respect, *r.sun* provides the link between terrain-derived rooftop geometry and the irradiance context later used in weather-aware photovoltaic simulation.

A related enabling component is Actinia, which provides a service-oriented execution environment for GRASS GIS processing. Rather than treating geospatial computation as a purely local desktop activity, Actinia enables GRASS-based operations to be managed within a backend-oriented and controlled execution context. This is important for the present framework because the system is not organised as an interactive GIS project, but as a reproducible computational service in which geospatial stages must be invoked programmatically and linked to a broader workflow. Actinia therefore supports the framework not simply by exposing geospatial capabilities, but by embedding them within a structured execution model compatible with backend services, user sessions, and workflow orchestration.

The relevance of these technologies becomes even clearer when they are considered together. GRASS GIS provides the computational environment for raster-based terrain and solar processing, *r.sun* provides the clear-sky radiation model that transforms rooftop geometry into solar-exposure information, and Actinia provides the service-oriented layer through which these capabilities can be invoked as part of a controlled analytical workflow. Their combined use is therefore not accidental. It reflects the need for a rooftop photovoltaic framework that can move from geospatial input handling to solar analysis in a way that is computationally robust, reproducible, and suitable for integration into a larger backend system.

For the proposed framework, this stack is particularly valuable because it supports the transition from raw or reconstructed urban surface data to structured solar-relevant outputs without requiring geospatial analysis to remain isolated from the rest of the system. Instead, terrain processing and solar modelling become integrated components of the

broader framework architecture. In this sense, *GRASS GIS*, *r.sun*, and *Actinia* should be understood as enabling technologies not only because they perform essential geospatial tasks, but because they make it possible to embed those tasks within a rooftop-oriented photovoltaic workflow that is service-based, modular, and analytically reusable. **Figure 3.2** illustrates the conceptual role of *GRASS GIS*, *r.sun*, and *Actinia* within the geospatial computation stack of the proposed framework.

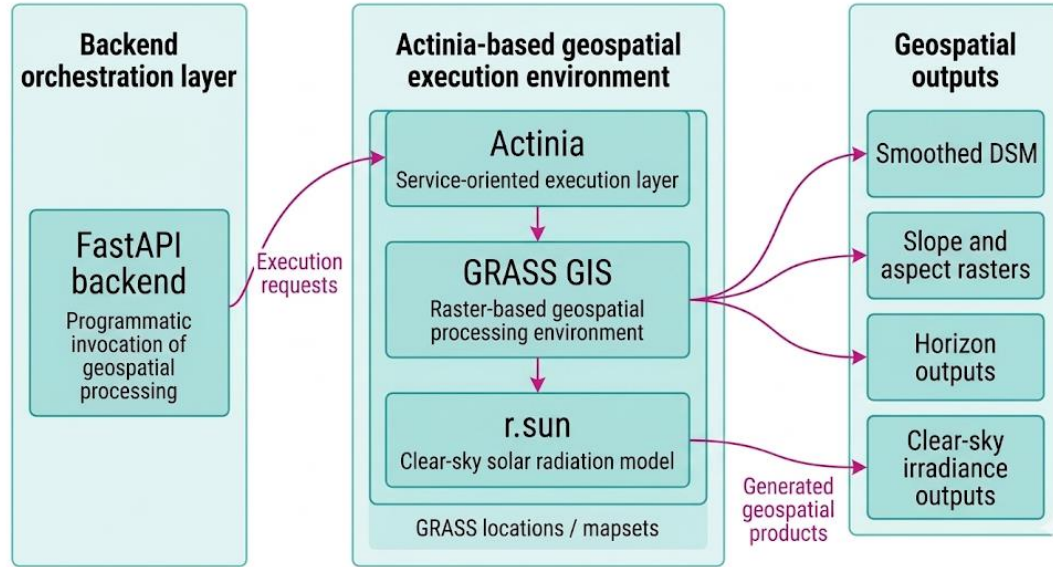


Figure 3.2. GRASS GIS, *r.sun*, and Actinia in the geospatial computation stack.

3.4 Photovoltaic, weather, and external solar data technologies

The proposed framework combines photovoltaic modelling tools, weather-data sources, and external solar-resource services in order to move from geospatial solar-exposure analysis to rooftop-level energy estimation under realistic operating conditions. While the previous section described the geospatial computation stack responsible for terrain and clear-sky preprocessing, the present section focuses on the technologies that enable photovoltaic conversion modelling, weather-aware adjustment, and the use of external solar-related data within the broader system. These technologies are particularly important because rooftop photovoltaic simulation depends not only on geometric and radiative context, but also on atmospheric variability, thermal effects, and photovoltaic performance models capable of translating irradiance into electrical output.

A central enabling technology in this part of the framework is *pvlb*, which provides the main computational environment for photovoltaic performance modelling. In the proposed system, *pvlb* supports several functions essential to rooftop-level simulation, including solar-position calculation, irradiance processing on tilted planes, temperature-dependent photovoltaic conversion, and, where required, inverter-related AC estimation. This is important because photovoltaic output cannot be inferred directly from rooftop irradiance alone. Instead, a rooftop photovoltaic framework must convert irradiance into electrical power through physically interpretable models that account for panel orientation, solar position, and thermal operating conditions. In this respect, *pvlb* provides the modelling layer through which solar-resource information is transformed into photovoltaic production estimates.

Within this modelling layer, the *PVWatts* approach is especially relevant because it offers a practical and widely used way of relating irradiance and module temperature to photovoltaic power output. Its importance in the framework lies in providing a computationally efficient and physically interpretable power-conversion model that can be integrated into time-resolved rooftop simulation. This is particularly useful in a framework that must support repeated evaluation across multiple rooftop units and time steps, while still preserving the influence of changing irradiance and thermal conditions on final photovoltaic output.

For this reason, photovoltaic simulation also depends on technologies for cell-temperature estimation. In the proposed framework, *pvlb* supports both the SAPM and PVsyst temperature models, which provide parameter-based approaches for relating irradiance, ambient temperature, wind speed, and mounting characteristics to module operating temperature. The enabling role of these models lies in their ability to connect environmental conditions and photovoltaic conversion in a physically meaningful way. Without a temperature model, irradiance would effectively be treated as if it acted on a module under fixed thermal conditions, reducing the realism of the power-estimation stage. Temperature-aware modelling, by contrast, allows the framework to reflect the fact that photovoltaic performance is shaped jointly by solar input and thermal behaviour.

A second enabling group concerns weather-data technologies, since weather-aware simulation requires meteorological information beyond the clear-sky radiative conditions computed in the geospatial stack. In this framework, the conceptual distinction between clear-sky and real-sky conditions is fundamental. Clear-sky radiation represents the theoretical

irradiance expected under cloud-free atmospheric conditions and provides a useful baseline for solar-resource analysis. Real-sky simulation, by contrast, requires observed or reconstructed weather variables that reflect actual atmospheric conditions. This distinction matters because photovoltaic systems operate under real weather rather than idealised clear-sky conditions, and rooftop output must therefore be modelled in relation to variable irradiance, ambient temperature, and wind conditions.

Within this context, *Open-Meteo* provides an important enabling service because it supplies structured weather-data that can support time-resolved photovoltaic simulation. Conceptually, this kind of weather service matters not merely because it provides atmospheric variables, but because it allows the framework to connect rooftop-level solar-resource information to time-stamped environmental conditions in a reproducible manner. Weather services therefore function as more than external data repositories. They act as a bridge between geospatial solar preprocessing and real operating-state simulation, enabling photovoltaic output to be estimated under temporally varying meteorological conditions rather than under static or idealised assumptions alone.

A further enabling component is *PVGIS*, whose role in the proposed framework is especially relevant to the fast horizon-based simulation branch. *PVGIS* is not used here simply as a general solar-information platform, but as an external source of horizon-profile data associated with specific geographic locations. This is methodologically important because rooftop shading and solar obstruction are not determined only by local rooftop form. They may also depend on broader directional terrain conditions or the surrounding obstruction context. In the fast horizon-based branch, external *PVGIS* horizon profiles provide an additional directional description of sky obstruction that complements the horizon information derived locally from the framework's own geospatial preprocessing. Before these external horizon profiles are combined with local GRASS-derived horizon information, the corresponding directional conventions are harmonised so that both sources can be interpreted consistently within the *pvlb-based* simulation step. This makes *PVGIS* a meaningful external solar-resource technology within the framework rather than a peripheral reference service.

Taken together, these technologies enable the transition from geospatially derived solar context to weather-aware rooftop photovoltaic estimation. *pvlb* provides the computational

environment for photovoltaic modelling, *PVWatts* provides a practical power-conversion approach, *SAPM* and *PVsyst* temperature models support thermal realism, *Open-Meteo* provides weather input for real-sky simulation, and *PVGIS* contributes external horizon information for horizon-aware photovoltaic assessment. Their combined role is therefore not simply to add external data or software dependencies, but to supply the conceptual and computational components required for transforming rooftop solar exposure into photovoltaic production estimates under realistic environmental conditions. In the proposed framework, these enabling technologies support not only the workflow-based photovoltaic simulation branch, but also the fast horizon-based alternative, and they remain relevant across different execution settings, including batch simulation and co-simulation. To support the discussion above, **Figure 3.3** conceptually illustrates the relationship between photovoltaic modelling, weather input, and external solar-resource data within the proposed framework.

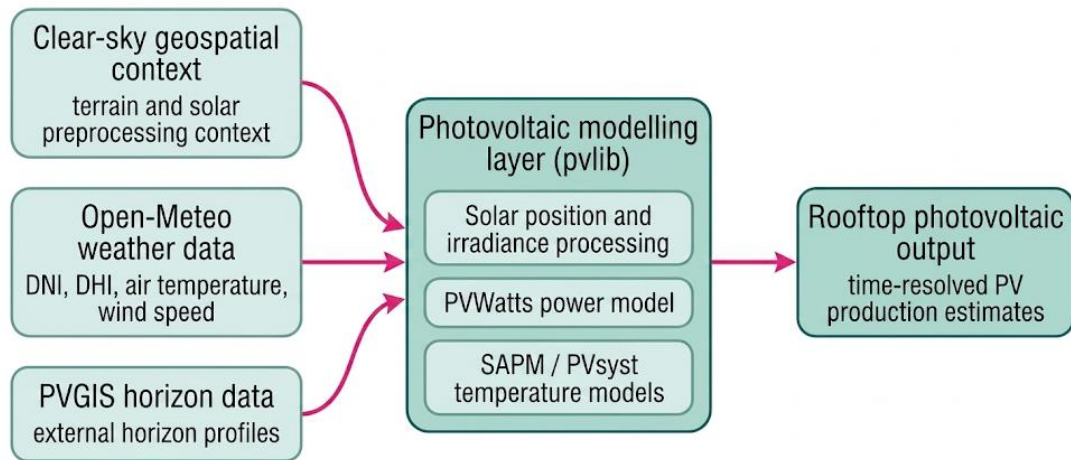


Figure 3.3. PV modelling, weather input, and external solar-resource data relationships

3.5 Artefact and persistence technologies

A central enabling principle of the proposed framework is that intermediate and final computational products are not treated as transient outputs that disappear once a processing step has finished. Instead, the system is organised around the explicit materialisation, storage, and retrieval of structured analytical artefacts. This is an important technological feature because the framework combines computationally intensive geospatial processing with later photovoltaic simulation stages that may need to reuse previously generated results. For this reason, artefact design and data persistence are not secondary implementa-

tion details, but enabling technologies that support scalability, traceability, and controlled reuse across the broader workflow.

The most important artefact technology in this context is Parquet, which is used as the main format for storing structured analytical datasets generated by the framework. Parquet is a columnar storage format particularly well suited to large analytical tables in which repeated access is often limited to selected variables rather than entire records. This is relevant for rooftop photovoltaic simulation because many of the framework’s outputs are naturally organised as structured datasets indexed by rooftop unit, point, timestamp, or simulation stage. In such settings, a columnar format offers clear advantages, including compact storage, efficient reading of selected fields, and practical reuse in downstream processing. Within the framework, Parquet therefore functions not simply as a file format, but as a data-layer technology that makes intermediate and final analytical products manageable at the scale required by rooftop-level and time-resolved simulation.

The technological role of Parquet becomes especially clear at the boundary between geospatial computation and later analytical processing. The framework performs terrain-related preprocessing and solar-radiation analysis within the GRASS-based geospatial environment, but later stages such as rooftop aggregation, weather-aware adjustment, and photovoltaic estimation operate more naturally on structured tabular artefacts than on native raster-processing outputs alone. For this reason, the system includes an explicit transition from GRASS-side products to reusable analytical datasets. This architectural boundary is important because it separates geospatial computation from later data-oriented analysis while preserving continuity between them. In effect, Parquet enables the framework to convert raster-derived geospatial outputs into persistent analytical artefacts that can be read, filtered, aggregated, and reused without repeating the full upstream computation.

Alongside these artefacts, the framework also relies on PostgreSQL as the principal relational persistence technology for metadata and execution tracking. In this context, PostgreSQL is not used to store the full geospatial or time-series outputs of the simulation pipeline. Instead, its enabling role is to maintain the structured system information required to operate the framework as a controlled computational service. This includes user-related information, session context, job tracking, and references to generated outputs. The distinction between metadata persistence and artefact persistence is technologically im-

portant. Large analytical products are stored separately in a form suited to structured data access, while relational persistence is reserved for the system state required to manage execution, maintain traceability, and support reproducibility across multiple runs and users.

A further supporting technology in this layer is Valkey, which provides a lightweight key-value persistence mechanism for session-scoped intermediate context. Its role within the framework differs from that of PostgreSQL and Parquet. Whereas PostgreSQL preserves structured relational metadata and Parquet stores reusable analytical products, Valkey supports rapid retrieval of intermediate workflow context needed by later stages within the same execution sequence. In a multi-stage geospatial and photovoltaic pipeline, later processing steps often depend on knowledge of what earlier stages have already produced. A session-oriented cache is therefore useful because it allows the framework to resolve prerequisite context efficiently without requiring repeated reconstruction of the same transient execution state. In this sense, Valkey supports workflow continuity and controlled stage-to-stage coordination rather than long-term analytical storage.

Taken together, these technologies define the persistence layer of the framework in a functionally differentiated manner. Parquet supports the storage of reusable analytical artefacts, PostgreSQL supports the persistence of relational metadata and execution state, and Valkey supports session-scoped intermediate context across linked processing stages. Their combined use is important because the proposed system is not a single one-step computation, but a staged framework in which outputs must remain accessible, interpretable, and reusable beyond the moment of their initial generation. This makes artefact and persistence technologies a core part of the enabling structure of the framework rather than a purely technical afterthought. To support the discussion above, **Figure 3.4** conceptually illustrates the different roles of Parquet, PostgreSQL, and Valkey within the persistence layer of the proposed framework.

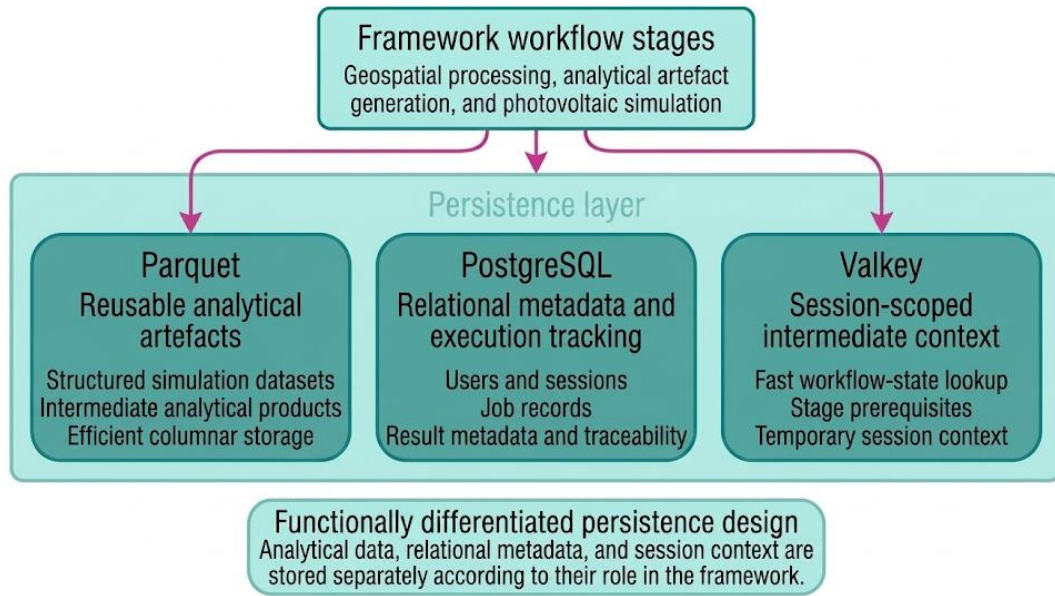


Figure 3.4. Parquet, PostgreSQL, and Valkey in the persistence layer.

3.6 Backend and deployment technologies

The proposed rooftop photovoltaic framework is not implemented as a standalone geospatial script or a local desktop workflow. Instead, it is designed as a backend-oriented computational service in which geospatial processing, artefact generation, weather-aware photovoltaic simulation, and result retrieval are exposed through a structured programmatic interface. For this reason, backend and deployment technologies play an important enabling role within the system. They do not contribute directly to terrain modelling or photovoltaic physics, but they make it possible for the framework to operate as a reproducible, controllable, and deployable service rather than as an isolated sequence of manual procedures.

A central technology in this layer is FastAPI, which provides the web framework through which the main capabilities of the system are exposed. Its enabling role lies in supporting a clear API structure, validated request and response handling, and automatically generated interactive documentation. In the context of the proposed framework, this is especially important because the system is organised as a set of interdependent processing stages rather than as a single one-step calculation. A backend framework is therefore needed not only to expose computational functions, but also to structure how those functions are invoked, validated, and connected to authenticated execution contexts. FastAPI supports this

role by providing a lightweight and well-structured foundation for the service layer of the framework.

A particularly important architectural feature of the system is that it operates through a session-scoped, multi-stage workflow model. In a simple stateless API, each request is typically expected to contain all the information required to complete a single task independently. The proposed framework is organised differently, because later stages often depend on intermediate products generated by earlier ones. Terrain preprocessing, artefact generation, rooftop aggregation, weather handling, and photovoltaic simulation are therefore not isolated service calls, but linked stages within a broader computational sequence. In this setting, the backend must support not only request handling but also controlled continuity across stages. This makes the service layer itself an enabling technology, since it provides the operational structure within which complex rooftop photovoltaic workflows can be executed in a coordinated manner.

A second important enabling technology in this layer is Docker, whose role in the framework extends beyond general deployment portability. In many software systems, containerisation is used primarily to simplify packaging and environment management. In the proposed framework, however, containerisation is also part of the computational architecture itself. The framework depends on a containerised geospatial execution environment in which Actinia and the underlying GRASS-based processing stack are available within a controlled runtime context. This means that Docker is relevant not only because it makes the system easier to reproduce across machines, but also because it supports the operational separation between the backend service and the geospatial execution substrate on which key processing stages rely.

This deployment-oriented role becomes especially important in a framework that combines heterogeneous technologies with different runtime requirements. Geospatial processing, analytical artefact management, metadata tracking, and photovoltaic simulation do not all depend on the same software environment, yet they must remain interoperable within a single coherent system. Containerised deployment helps maintain this interoperability by isolating execution environments while preserving controlled communication between them. In this respect, Docker contributes not only to reproducibility and portability, but also to the practical stability of the framework as a service-based computational platform.

Authentication and access control also contribute to this layer, although their role is secondary to the main computational and deployment technologies. In the framework, authenticated access is necessary because workflow execution is associated with user-specific and session-specific processing context. This is important in a system where generated artefacts, metadata, and execution state must remain linked to the correct workflow instance. At the level of enabling technologies, however, authentication is better understood as supporting infrastructure rather than as a primary analytical component. Its importance lies in enabling controlled service access and execution isolation rather than in shaping the geospatial or photovoltaic logic of the framework itself.

Taken together, these technologies define the operational service layer through which the broader rooftop photovoltaic framework becomes deployable and executable as a backend system. FastAPI provides the structured API environment, session-aware service handling supports linked multi-stage execution, Docker provides both deployment reproducibility and computational-environment separation, and authentication supports controlled access to workflow context. Their combined role is therefore not to replace the analytical core of the framework, but to make that core accessible, reproducible, and operationally manageable within a service-oriented implementation. To support the discussion above, **Figure 3.5** conceptually illustrates the role of the backend service layer and containerised deployment environment within the proposed framework.

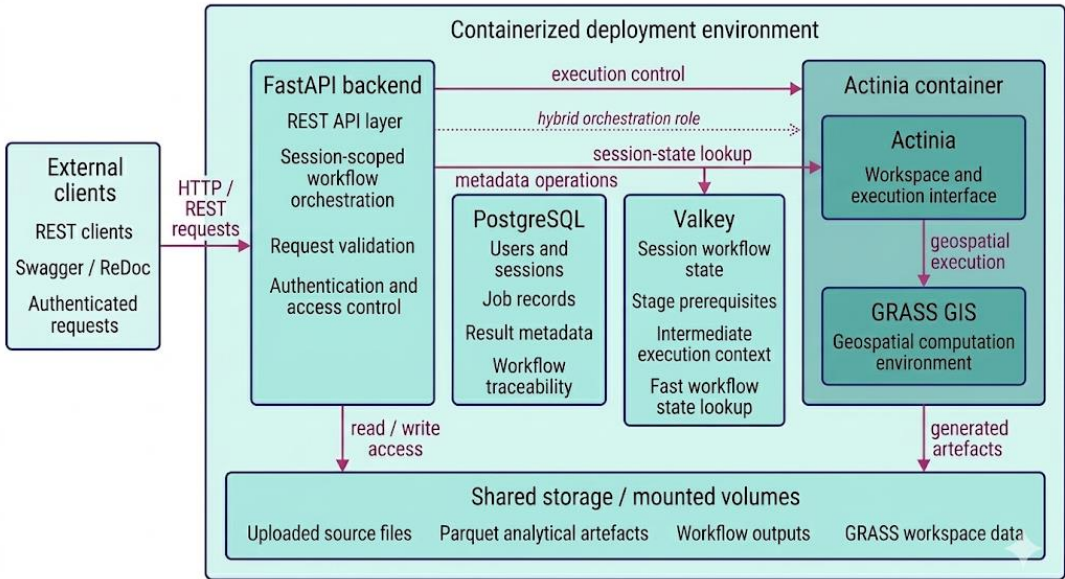


Figure 3.5. Backend service layer and containerised deployment environment

3.7 Summary

This chapter has outlined the main technologies that enable the proposed rooftop photovoltaic simulation framework and clarified their role within the broader system. Rather than describing the workflow in terms of implementation steps, the discussion has focused on the technological foundations that make the framework possible, including geospatial data representations, geospatial computation technologies, photovoltaic and weather-aware modelling tools, artefact and persistence technologies, and backend and deployment components.

The chapter first showed that rooftop photovoltaic analysis depends on geospatial representations capable of preserving both continuous surface structure and discrete building-related entities. In this context, raster and vector forms, together with terrain-oriented surface models such as DSMs, provide the spatial basis for rooftop-oriented analysis. It then examined the geospatial computation stack formed by GRASS GIS, *r.sun*, and Actinia, highlighting how terrain preprocessing and clear-sky solar-radiation modelling are supported within a service-oriented execution environment.

The discussion also introduced the photovoltaic and weather-related technologies that support the transition from solar-exposure analysis to rooftop-level energy estimation under realistic operating conditions. In this respect, *pvlb*, *PVWatts*, temperature models, weather services such as *Open-Meteo*, and external solar-resource services such as *PVGIS* together provide the computational and data-oriented basis for weather-aware photovoltaic simulation. Their combined role is especially important in a framework that supports both workflow-based and horizon-aware photovoltaic estimation paths.

A further enabling layer is defined by artefact and persistence technologies. The chapter explained that Parquet supports the storage of reusable analytical artefacts, PostgreSQL supports relational metadata and execution tracking, and Valkey supports session-scoped intermediate workflow context. Together, these technologies make it possible for the framework to preserve structured outputs across stages and maintain controlled continuity in a multi-stage computational setting. Finally, the chapter considered the backend and deployment technologies that allow the framework to function as an operational service rather than as an isolated analytical script, with FastAPI supporting the service layer and Docker supporting controlled deployment and execution-environment separation.

Taken together, these enabling technologies form the technical foundation on which the methodology of the proposed framework is built. They do not define the workflow sequence by themselves, but they provide the capabilities that allow that workflow to be implemented in a reproducible, service-oriented, and analytically reusable manner. For this reason, the present chapter serves as the technological bridge between the broader research context discussed in the literature review and the implementation-oriented methodological structure presented in the following chapter.

Methodology

4.1 Introduction

This chapter describes the methodological framework adopted for the proposed system: how the implemented platform transforms raw geospatial data into datasets suitable for rooftop photovoltaic (PV) analysis and energy-production estimation. The system is a backend processing service that combines web-based orchestration with geospatial analysis tools, exposing through a REST API a sequence of computational tasks that integrate spatial data processing, solar-radiation modelling, and photovoltaic simulation.

The approach follows a multi-stage pipeline. From geospatial inputs such as building footprints and terrain models, the system generates intermediate datasets – digital surface models, terrain attributes, and solar-radiation estimates – which are stored and reused in later stages, separating computationally intensive geospatial preprocessing from the final PV simulation. The following sections describe the architecture and the main components of this workflow.

4.2 System Overview

4.2.1 System goals and design principles

The system is a backend framework for rooftop photovoltaic simulation that integrates geospatial preprocessing, solar-radiation estimation, and photovoltaic estimation into a multi-stage workflow. Its aim is not merely to execute individual tasks but to maintain methodological consistency across the whole workflow: it operates on heterogeneous spatial data, transforms it into a standardised form, and reuses that data downstream without re-executing earlier steps. Five design principles follow from this aim.

The first is reproducibility. The system uses controlled execution contexts, session scoping, persistent intermediate artefacts, and centralised configuration of external services, removing ambiguity about how inputs are processed and allowing the outputs of earlier steps to be reused later in the workflow.

The second is modularity. Authentication and session control, user management, geospatial preprocessing, analytical artefact generation, and real-sky simulation are separated into distinct layers and route groups. This improves extensibility and maintainability and keeps the GRASS-based geospatial preprocessing clearly distinct from the later analytical and simulation steps.

The third is artefact-driven execution. Intermediate data are treated not as transient by-products but as explicit, reusable artefacts, which decouples the expensive geospatial processing from the later analytical and simulation steps. Clear-sky baselines, rooftop sampling data, suitable-area data, and aggregated rooftop-level data can therefore be reused without recomputing the upstream stages.

The fourth is portability through configuration. No deployment-specific values are hard-coded in the service logic; external dependencies — the Actinia base URL, storage roots, JWT settings, database connections, and container identifiers — are set through a single configuration layer, so the same pipeline adapts easily to different environments.

The fifth is accommodation of stage-specific computational needs. Because the workflow's stages do not share the same runtime characteristics, the system applies different execution models to different components: the geospatial runtime, the analytical artefact generation, and the photovoltaic simulation.

4.2.2 High-Level Architecture

This section describes the high-level architecture of the proposed rooftop photovoltaic simulation framework and the role of its principal components. As shown in **Figure 4.1**, the system is presented not as a generic processing chain but as a structured methodological infrastructure in which access, execution control, core analytical services, persistence, and heterogeneous inputs are explicitly differentiated. This separation matters because the framework does not perform isolated geospatial operations; it coordinates a sequence of interdependent services that transform heterogeneous rooftop, terrain, atmospheric, and weather inputs into reusable intermediate artefacts and, ultimately, photovoltaic simulation outputs.

Operationally, the backend acts as a controlled orchestration environment. It exposes the framework through REST endpoints, governs execution through authentication and ses-

sion-aware coordination, organises the analytical stages as a set of processing services, and relies on dedicated persistence and execution substrates to store metadata, manage artefacts, and support spatial computation. Making the execution logic explicit improves reproducibility, maintainability, and extensibility as new services or branches are added.

Figure 4.1 presents the framework through five complementary groups: the Access Components, which expose the REST interface; the Control and Coordination group, which handles authentication and workflow orchestration; the Core Processing Services, which implement the analytical stages of the workflow; the Persistence and Execution Substrate, which stores metadata and artefacts and hosts spatial computation; and the Input Data and External Sources, which supply the heterogeneous inputs that initiate the workflow. This grouping reflects the implemented system more faithfully than a simplified layered description, because it makes explicit not only the methodological stages of the workflow but also the supporting components that enable controlled execution, artefact reuse, and traceable storage.

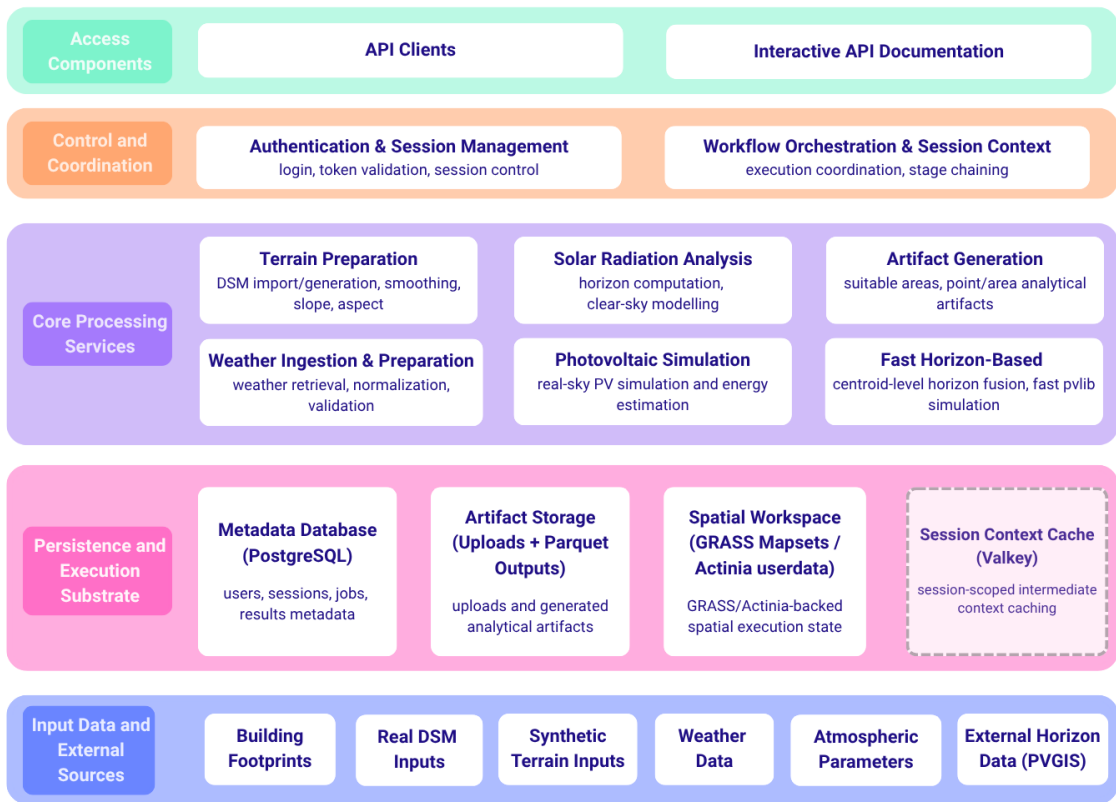


Figure 4.1. High-Level Architecture of the Proposed Framework

Access Components

The Access Components define how the framework is exposed to users and external clients. Concretely, the framework is implemented as a FastAPI application that publishes a set of REST endpoints, grouped by function into authentication, user-management, and solar-processing routes. Alongside these endpoints, the same application exposes interactive API documentation generated from the OpenAPI schema and served through Swagger UI and ReDoc, which lets users inspect the available operations and their parameters directly.

These components do not perform analysis or simulation themselves; they are the entry points through which requests and parameters are submitted and through which results and generated artefacts are retrieved. The REST clients issue the calls that trigger the workflow, while the OpenAPI documentation describes how those calls are structured. Separating this access surface from the rest of the architecture keeps user-interaction concerns distinct from the internal execution logic, so that the way the framework is invoked remains independent of how the analytical stages are implemented.

Control and Coordination

The Control and Coordination group governs how requests are validated, organised, and propagated through the workflow. It consists of two principal components: *Authentication and Session Management* and *Workflow Orchestration and Session Context*.

The Authentication and Session Management component governs controlled access to the framework. It is implemented through JWTs signed with the HS256 algorithm, issuing short-lived access tokens together with longer-lived refresh tokens; the refresh tokens are persisted in the metadata database as keyed hashes, with explicit expiry and last-use timestamps so that sessions can be tracked and revoked. Dedicated login and logout endpoints establish and terminate a session, and every protected request is validated before it reaches the analytical stages. The output of this component is therefore not only an accept/reject decision but a validated, session-aware identity: the authenticated user is the key that scopes downstream execution, since it maps to a per-user spatial workspace in the execution substrate. In this way the component does more than restrict access — it fixes the controlled, session-aware environment in which the rest of the workflow runs.

The Workflow Orchestration and Session Context component coordinates the execution of the analytical stages. Taking the validated session as input, it connects the framework's services into a coherent sequence by chaining the per-stage runners that drive the geospatial and simulation steps – for example, invoking GRASS operations inside the Actinia execution environment under the authenticated user's mapset – and by maintaining session-scoped context so that the outputs of one stage remain available to the next. This matters because the framework is not a single monolithic process, but a set of service blocks whose intermediate outputs must remain reusable and traceable across the execution sequence. Its output is thus an ordered, context-preserving execution flow in which each Core Processing Service is invoked with the inputs and identity it requires.

Core Processing Services

The Core Processing Services are the analytical core of the framework, implementing the stages that turn raw or semi-processed inputs into photovoltaic outputs. Consistent with the access surface described above, each block is exposed as an independently invocable service under the framework's REST interface, so that a stage can be triggered on its own or chained into the full workflow by the orchestration component. The blocks are organised so that the static geometric and radiative context is established first, reusable artefacts are then materialised, time-varying weather is ingested separately, and the prepared products finally converge in a simulation stage that estimates photovoltaic output.

Six blocks make up this group: *Terrain Preparation*, which builds the rooftop and terrain representation and its derivatives; *Solar-Radiation Analysis*, which adds the horizon and clear-sky radiative context; *Artefact Generation*, which materialises the reusable intermediate products consumed downstream; *Weather Ingestion and Preparation*, which supplies the time-dependent meteorological drivers; *Workflow-based Photovoltaic Simulation*, the main path in which these products are combined to estimate real-sky photovoltaic behaviour; and *Fast Horizon-based Simulation*, a computationally lighter alternative path to the same estimation. The following subsections describe each service in turn, including its inputs, the tools and methods it applies, and the outputs it produces for the subsequent stages.

Terrain Preparation

The Terrain Preparation service prepares the rooftop and terrain representation, together with its derivatives, required for all downstream analysis. Its defining feature is that it does not rely on a single fixed input path: the service accepts different entry configurations depending on data availability and reconciles them into a common terrain representation before any solar computation takes place.

When a high-resolution real DSM is available — typically at 1 m resolution or finer — it is taken as the input and imported into the spatial workspace with the GRASS module *r.in.gdal*. The raster resolution is then read and validated against the high-resolution acceptance condition, so that only sufficiently detailed surfaces enter the pipeline as a direct terrain representation. When no such DSM is available, the service follows a synthetic-terrain route built from the building footprints: if the footprints already carry a height attribute, the footprints are rasterised by that attribute to produce a synthetic DSM; if they do not, representative building heights are first derived from auxiliary elevation inputs — as the difference between an upper-surface and a bare-earth elevation — and then attached to the footprints to construct the synthetic surface. Both branches converge on the same product: a usable DSM raster held in the per-user GRASS mapset.

Once the terrain representation is established, the service computes the geometric derivatives needed downstream. The DSM is first smoothed through a short *r.neighbors* filter chain — a size-7 moving average followed by a size-3 median — to suppress acquisition noise and remove features too small to host a panel, and the smoothed surface is then passed to *r.slope.aspect*, which derives the per-cell slope and aspect rasters. The outputs of this service — the smoothed DSM and its slope and aspect rasters, all stored in the spatial workspace — constitute the geometric basis consumed by the Solar-Radiation Analysis service and by the later suitability and photovoltaic computations.

Solar-Radiation Analysis

The Solar-Radiation Analysis service establishes the radiative context. Once the terrain is prepared, rooftop orientation and exposure must be interpreted in solar rather than purely geometric terms, and this service performs that interpretation in two steps. It takes as input the smoothed DSM and the slope and aspect rasters produced by Terrain Preparation, together with a monthly series of Linke turbidity coefficients, a ground-albedo value, and the set of representative days and time steps for which radiation is to be evaluated.

First, horizon computation is carried out with the GRASS module `r.horizon`, which sweeps the surrounding surface around each cell at a fixed angular step, up to a bounded search distance, to obtain the local horizon angles that describe how the terrain itself obstructs incoming radiation. Second, clear-sky modelling is performed with `r.sun`, which combines the elevation, slope, aspect, and horizon information with the Linke turbidity and albedo values to compute beam (direct) and diffuse clear-sky irradiance for each representative day and time step. The outputs – the per-cell horizon profiles and the clear-sky direct and diffuse irradiance maps, stored in the spatial workspace – are the solar-relevant products consumed by the later suitability, artefact-generation, and energy stages. In particular, the clear-sky maps provide the reference against which real-sky conditions are subsequently scaled, which is why these service marks the transition from geometric terrain preparation to radiative analysis.

Artefact Generation

The Artefact Generation service materialises the intermediate analytical products consumed later in the workflow, and this is a central characteristic of the framework: rather than treating intermediate results as transient states, the system writes them as durable, reusable artefacts that can be stored, indexed, and read back downstream without recomputing the upstream stages. It takes as input the geospatial and solar products of the preceding services – the suitability rasters, the slope and aspect maps, the cadastral footprints, and the clear-sky irradiance maps – and turns them into a small set of explicit, well-typed outputs.

Concretely, the service produces several artefact families. The deployable rooftop areas are obtained by vectorising the suitability raster with `r.to.vect`, exporting the resulting polygons through `v.out.ogr`, and clipping them against the cadastral footprints; the polygons, enriched with geographic coordinates, are written as a GeoParquet file (`suitable_areas.parquet`). In parallel, point- and area-based products are derived from the geospatial and solar stages and written as Apache Parquet tables: per-point static attributes (position, slope, and aspect of each sampling point), per-point clear-sky time series across the representative time steps, and per-area mean clear-sky irradiance, in which the beam and diffuse components are averaged over the sampling points of each rooftop area.

All of these artefacts persisted in the framework's artefact storage as Parquet datasets, organised so they can be queried selectively rather than reloaded in full. This design both improves efficiency, by enabling reuse instead of recomputation, and strengthens traceability, by making the intermediate outputs explicit and addressable. The artefacts produced here, deployable areas, static point attributes, point time series, and area clear sky means, are exactly the inputs later combined by the photovoltaic simulation services.

Weather Ingestion and Preparation

The Weather Ingestion and Preparation service handles the time-dependent meteorological information required for real-sky simulation. Whereas terrain and solar preprocessing fix the static geometric and radiative context, realistic estimation also depends on time-varying conditions, so weather data are treated as a dedicated service that retrieves, normalises, and validates these inputs into a form the simulation can consume directly.

The service accepts weather from more than one source. It can query the Open-Meteo web service for a given location and time range, or it can ingest user-supplied files in CSV or EPW format, which lets measured or station-specific records be used in place of, or alongside, the online source. Whatever the origin, the raw records are mapped onto a canonical schema, DNI, DHI, GHI, ambient air temperature, and wind speed, and their timestamps are parsed into timezone-aware UTC, so that heterogeneous inputs are reconciled onto a common reference.

The normalised series are then placed on a uniform 15-minute time grid: each variable is aggregated and gap-filled through interpolation and forward/backward filling, and the result is validated to ensure that the fields required for simulation are present and complete. The output is a clean, regularly sampled, UTC-referenced weather table providing the irradiance and ambient drivers, temperature and wind speed in particular, that the Workflow-based and Fast Horizon-based simulation services use to translate clear-sky references into real-sky photovoltaic estimates.

Workflow-based Photovoltaic Simulation

The Workflow-based Photovoltaic Simulation service is the main downstream path, where the outputs of all preceding stages converge. It draws together the deployable rooftop areas and the per-area clear-sky means produced by Artefact Generation, the normalised real-

sky weather table from Weather Ingestion and Preparation, and the geometric description of each rooftop area, and combines them to estimate photovoltaic behaviour under real-sky conditions. The estimation itself is carried out through a pvlib-based simulation step applied to each deployable area over the simulation horizon.

For every time step, the service first aligns the real weather record with the matching clear-sky reference, mapping each real timestamp to the nearest pre-computed 15-minute clear-sky snapshot for that rooftop area. It then forms the real-sky irradiance by scaling the stored clear-sky means with two coefficients derived from the weather data — a direct coefficient, the ratio of measured to clear-sky DNI, and a diffuse coefficient, the analogous ratio for the diffuse component. The scaled beam and diffuse contributions are combined into the irradiance incident on the tilted rooftop surface, the operating cell temperature is estimated from ambient air temperature and wind speed, and the DC power is finally computed with the PVWatts formulation, scaled by the effective installation area and the assumed power density of the array.

The output of this service is therefore a real-sky photovoltaic estimate for each deployable rooftop area across the requested time horizon, the time-resolved power, and by integration the energy, attributable to each candidate surface. As the computational endpoint of the primary workflow, it converts the structured intermediate artefacts of the earlier stages into the final energy-related products that the framework is designed to deliver.

Fast Horizon-based Simulation

The Fast Horizon-based Simulation service provides an additional simulation path designed as a computationally lighter alternative to the workflow-based estimation stage. Instead of relying on the full solar-map computation sequence during downstream simulation, it operates on a selected set of PV system areas and builds, for each one, a directional description of the surrounding obstruction. For every area the service first reduces the rooftop to its centroid and samples the locally derived horizon profile at that point from the *r.horizon* outputs, converting the GRASS azimuth convention into the convention expected by the simulation library. It then retrieves an external horizon profile for the same centroid from the PVGIS service — the centroid being transformed to geographic coordinates for the query — and interpolates it onto the same azimuth grid. The two profiles are combined into a single

merged horizon by retaining, for each azimuth, the more obstructing of the local and external angles, so that both near-field terrain and far-field obstruction are represented.

This merged horizon is then applied directly inside a pvlib-based simulation step. The surface orientation is taken from the area's mean aspect, the incoming irradiance is transposed onto the tilted rooftop plane, and the direct component is masked whenever the sun lies below the merged horizon at its current azimuth; the operating cell temperature and DC power are then obtained as in the main path, using a SAPM or PVsyst temperature model. The output is a per-area, horizon-aware photovoltaic time series. The significance of this branch is that it preserves directional obstruction as an explicit part of the estimation while avoiding the full workflow-based solar-preprocessing chain at the final stage; it does not replace the main path but complements it as a lighter alternative within the same architecture.

Persistence and Execution Substrate

The Persistence and Execution Substrate provide the components through which the framework stores metadata, preserves outputs, and executes spatial computations in a controlled environment: the Metadata Database (PostgreSQL), Artefact Storage (uploads and Parquet outputs), Spatial Workspace (GRASS mapsets / Actinia userdata), and Session Context Cache (Valkey).

The Metadata Database (PostgreSQL) holds the structured metadata that supports the framework's operational logic. Accessed through an ORM layer, it maintains the core tables of the system — users, sessions, uploaded files, jobs, and results — so that every execution is tied to an authenticated user and a tracked job, and each output is catalogued through a result manifest that records where the corresponding artefact resides. This matters because the framework does not rely on file-based outputs alone: structured metadata persistence is what tracks execution, preserves state across stages, and supports reproducibility.

The Artefact Storage (uploads and Parquet outputs) holds the uploaded input files and the generated analytical artefacts, particularly the Parquet datasets produced by the Artefact Generation service. Because the workflow is artefact-driven, intermediate and final outputs are materialised and preserved — and referenced from the metadata database through the

result manifest — so they can be reused downstream or retrieved later without repeating the full chain of computations.

The Spatial Workspace (GRASS mapsets / Actinia userdata) is the spatial execution substrate for the geospatial stages. It is organised as per-user Actinia-managed GRASS locations and mapsets, created per CRS and addressed under the authenticated user's namespace, within which the raster and vector operations of the pipeline are executed and maintained. It is therefore not merely passive storage but the controlled environment in which the geospatial computations run, and their persistent layers accumulate across the analysis.

The Session Context Cache (Valkey) plays a secondary supporting role, maintaining session-scoped intermediate context that aids workflow continuity across stages. It is not the primary system of record, but it supports efficient chaining of stages within the controlled execution environment.

Input Data and External Sources

The Input Data and External Sources group represents the heterogeneous information required to initiate the workflow. In the implemented architecture, this group includes Building Footprints, Real DSM Inputs, Synthetic Terrain Inputs, Weather Data, Atmospheric Parameters, and External Horizon Data (PVGIS). Some of these are typically supplied by the end-user for the study area, while others are retrieved on demand from external web services; representing them explicitly is important because the framework is designed to accommodate different data-availability conditions while preserving a coherent downstream structure.

From a spatial standpoint, the entry conditions are set by the available rooftop and terrain information. Building Footprints are vector polygons that give the geometric basis for rooftop analysis and are central to the synthetic-terrain route, where they are combined with height information to construct a usable terrain representation. Real DSM Inputs provide a direct high-resolution raster representation when available, whereas Synthetic Terrain Inputs cover the alternative case, in which the terrain must be built from the footprints and indirect or derived elevation information. Together these define which of the two terrain-preparation branches the workflow follows.

Meteorological and atmospheric information define the environmental side of the workflow. Weather Data supply the time-dependent drivers for real-sky simulation — direct, diffuse, and global irradiance together with ambient air temperature and wind speed — and are obtained either from the Open-Meteo service or from user-provided CSV or EPW records. Atmospheric Parameters serve a different role: these are the monthly Linke turbidity coefficients, together with the ground-albedo value, that parameterise the clear-sky model and therefore influence the attenuation of solar radiation in the radiative stage, even before any real-sky behaviour is estimated.

A further external input is the External Horizon Data retrieved from PVGIS. This source is specifically relevant to the Fast Horizon-based Simulation service, where externally retrieved horizon information is associated with the centroid of each selected PV system area and merged with the locally derived horizon profile from the study-area terrain. Its explicit presence is methodologically important because it shows that the framework can integrate not only conventional geospatial and meteorological inputs, but also complementary external horizon information when a lighter downstream simulation strategy is required.

Taken together, these input categories define the admissible starting conditions of the framework and clarify how heterogeneous real-world data — whether user-supplied or retrieved from external services — are incorporated into a controlled, reusable, and implementation-aware methodological infrastructure.

4.2.3 End-to-end workflow summary

The workflow begins by preparing a DSM as the geometric basis for the geospatial stages. When a high-resolution DSM is directly available, the framework treats it as the primary surface representation: the uploaded raster is validated, its spatial-reference context resolved, and the dataset accepted only if it meets the resolution requirements of the real-DSM workflow. A raster that is too coarse does not enter this branch.

If no suitable high-resolution DSM is available, the system follows a derived-surface workflow, reconstructing rooftop geometry from building footprints — using height attributes already present in the vector data or deriving representative heights from external elevation datasets when these are absent. Once heights are assigned, a synthetic DSM is generated as the geometric basis for the solar-preprocessing stages.

Although these two branches differ in how the initial surface model is obtained, they converge towards the same methodological objective: producing a DSM suitable for downstream computation of rooftop terrain derivatives, solar exposure, and photovoltaic potential. The operational details of CRS handling, raster and vector import, resolution checks, and synthetic DSM generation are discussed later in **Section 4.5**. **Figure 4.2** summarises the alternative input configurations and DSM-preparation workflow.

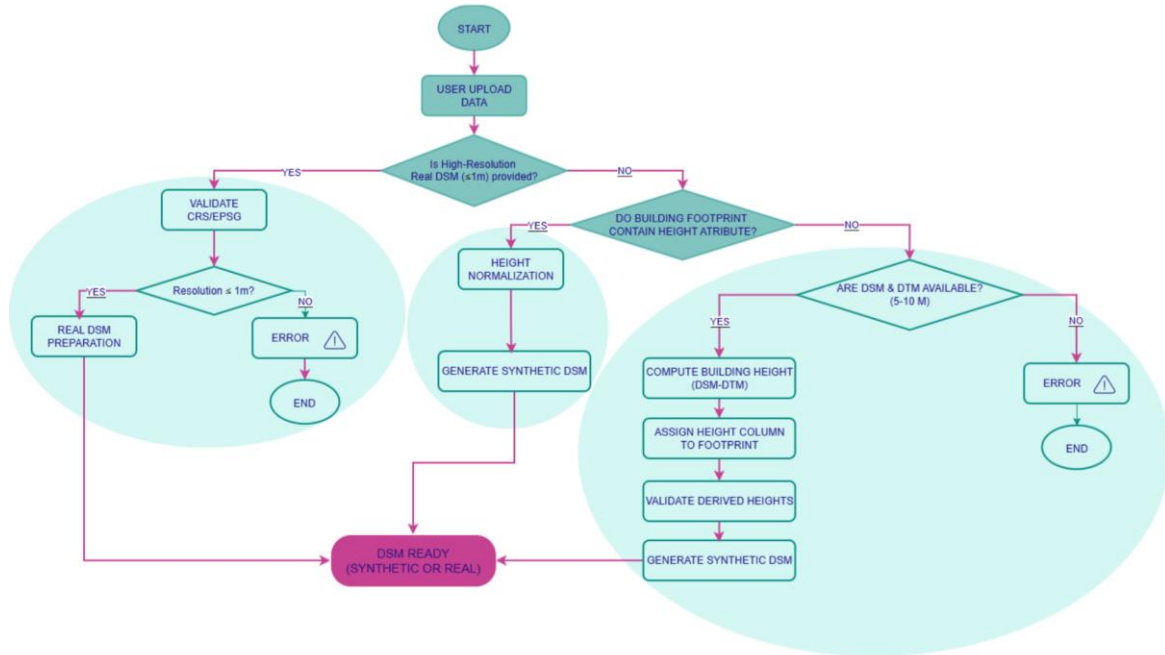


Figure 4.2. Input Configurations and DSM Preparation Workflow

Once the DSM representation has been prepared, the system proceeds with geospatial pre-processing to derive the terrain attributes required for clear-sky solar-radiation analysis. This stage includes DSM smoothing, the derivation of slope and aspect, and horizon analysis across multiple azimuth directions. Together, these raster products describe rooftop surface geometry and the extent to which surrounding terrain or structures obstruct incoming solar-radiation. On this basis, the workflow computes clear-sky solar irradiance using the GRASS GIS *r.sun* model. To capture seasonal variation while maintaining computational tractability, the simulation is evaluated for a set of representative days distributed across the year and sampled at regular time intervals during the daylight period. The resulting irradiance layers are not treated as final outputs, but are instead transformed into structured

intermediate artefacts for later reuse. The implementation details of geospatial preprocessing, clear-sky modelling, and execution strategy are discussed later in **Sections 4.5** and **6**. **Figure 4.3** summarises the clear-sky preprocessing pipeline.

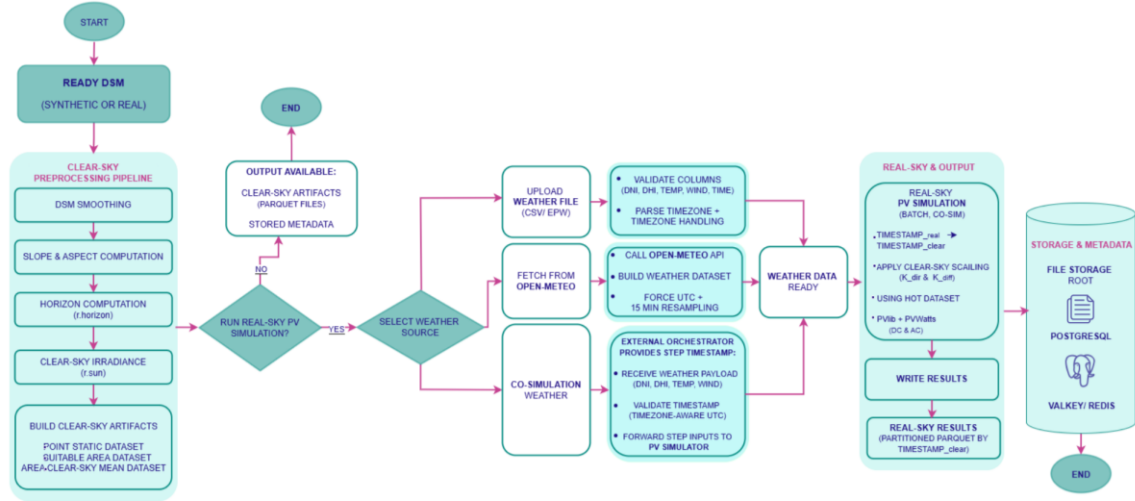


Figure 4.3. Clear-sky preprocessing workflow

To make the clear-sky raster outputs usable downstream, the framework reorganises them into structured analytical artefacts. Rather than continuing in raster form, it converts rooftop information into a point-based representation that links static rooftop geometry to time-dependent radiation efficiently. Representative sampling points are generated across rooftop areas, each carrying the geometric attributes needed later: location, surface orientation, and directional sky-visibility.

In parallel, the clear-sky irradiance outputs are reorganised into a time-indexed dataset keyed by timestamp and sampling location. This yields two complementary artefacts: a static dataset of rooftop sampling geometry and a temporal dataset of clear-sky radiation over the simulation grid. Separating static from time-dependent information is intentional – it reduces redundancy and supports efficient reuse in later stages. Implementation details are given in **Section 4.6**.

These artefacts are then used to identify rooftop regions suitable for installation and to reorganise radiation at the level of candidate surfaces. Suitability analysis has two alternatives: suitable polygons may be supplied directly by the user or detected automatically from terrain-derived attributes and rooftop masks produced earlier. In the automatic branch, roof-

top geometry is evaluated by inclination and orientation, and candidate surfaces are derived with their geometric properties. The workflow then computes additional attributes per surface, including representative geometric descriptors and effective rooftop area, producing a reusable artefact that links rooftop geometry to downstream estimation.

In a subsequent aggregation stage, the point-level clear-sky radiation data generated earlier in the workflow are associated with the suitable rooftop surfaces and transformed into rooftop-level radiation statistics. This produces a structured clear-sky baseline at the scale of candidate installation areas rather than individual sampling points. Organising the workflow in this way separates computationally intensive geospatial preprocessing from later photovoltaic simulation stages and enables reusable, storage-efficient analytical products. The implementation details of suitability detection, effective-area computation, artefact persistence, and rooftop-level radiation aggregation are discussed later in Sections 4.4 and 4.6.

From here the framework offers two downstream simulation paths. The first, the workflow-based path, builds on the rooftop-level clear-sky baseline and incorporates real atmospheric conditions to estimate production under non-ideal weather. Meteorological data may enter through several routes – uploaded weather files, external weather services, or co-simulation inputs supplied at runtime – and are transformed into a representation aligned with the simulation timestep. Each real timestamp is then matched to the closest precomputed clear-sky snapshot, so the baseline can be reused instead of recomputing the full solar model for every weather scenario.

Once the clear-sky reference is identified, the system scales the clear-sky irradiance with the meteorological observations to estimate the effective real-sky irradiance on each rooftop. These values, with temperature and wind, are passed to a dedicated photovoltaic simulation component that computes power output per timestep. This simulation layer is a reusable component supporting both batch execution over a full weather series and stepwise execution in co-simulation. Its structured outputs are the final stage of the pipeline, giving rooftop-level production estimates under realistic conditions.

In addition to this primary path, the framework also supports a fast horizon-based simulation approach as a lighter downstream alternative for selected PV system areas. Rather than relying on the complete solar-map computation sequence during final simulation, this branch operates on selected rooftop units and combines locally derived horizon-related in-

formation with externally retrieved horizon data associated with the centroid of each selected area. The resulting directional horizon profile is then used directly within a *pvl*-based simulation step to estimate irradiance and photovoltaic output. This alternative path is methodologically important because it preserves horizon-aware photovoltaic estimation while reducing dependence on the full downstream workflow-based solar-simulation chain. The implementation details of weather handling, temporal mapping, irradiance scaling, workflow-based photovoltaic conversion, and the fast horizon-based simulation approach are discussed later in [Section 4.6](#).

4.3 Access control and identity management

Alongside the computational pipeline, the platform implements an access-control and identity-management layer governing how users authenticate, obtain authorised API access, and interact with protected workflows. It ensures that only authenticated users can invoke backend operations and ties each interaction to a persistent server-side session. In implementation, it combines external credential validation against Actinia with internal JWT-based authentication and database-backed session tracking, so the system can authenticate users against the geospatial backend, issue application-level tokens, and maintain controlled session state across requests.

4.3.1 Authentication model

The authentication model of the platform follows a hybrid design that links external identity verification to internal API access control. At login time, the backend first validates the submitted username and password against the Actinia service. This external validation step ensures that the credentials correspond to a valid Actinia account before the user is allowed to enter the application-level workflow. After successful validation, the backend synchronizes a local user record containing Actinia-derived identity attributes used by the application, including the Actinia username, role, and group. At the API layer, authenticated access is implemented using JWTs in two forms: an access token and a refresh token. Token creation is handled explicitly by the authentication service, where each token includes standard identity and session-related claims. In particular, issued tokens contain the user identifier (sub) and the server-side session identifier (sid), thereby linking bearer tokens to a concrete database session. The access token additionally carries contextual identity information used by the backend during protected operations, including the Actinia username, role, and

group. Both token types are signed using the configured JWT secret and algorithm, and token validation routines distinguish between access and refresh tokens through dedicated token-type checks. A key characteristic of the implemented model is that authentication is not fully stateless. Before tokens are issued, the backend creates a session record in the *models.sessions* table. This session stores the authenticated user identifier, expiration time, refresh-token binding, device information, IP address, activity status, usage timestamps, and optional revocation metadata. The generated session identifier is then embedded into both JWTs, so that each authenticated request can be resolved not only to a user but also to a specific active session. This design allows protected endpoints to enforce a two-level authentication check. First, the incoming bearer token is decoded and validated cryptographically. Second, the backend verifies that the referenced session exists, belongs to the expected user, remains active, and has not expired. The `/auth/me` endpoint illustrates this mechanism clearly: the access token is accepted only if both token validation and session validation succeed, after which the backend returns the authenticated user identity together with the active session context.

The authentication model is thus a JWT-based access architecture with database-backed session control. It keeps the efficiency of bearer-token authorisation while avoiding the limits of purely stateless authentication, since session activity can be checked centrally and revoked explicitly when needed.

4.3.2 Token lifecycle and security measures

The authentication layer defines an explicit lifecycle for login, token renewal, and logout, with security checks at each stage. By pairing JWT issuance with database-backed session validation, token validity depends not only on cryptographic verification but on the state of the corresponding server-side session — allowing controlled revocation and immediate logout when a session is deactivated. **Figure 4.4** summarises the operational flow of login, refresh-token rotation, and logout in the implemented authentication layer, highlighting the interaction between the client, the API backend, the external Actinia service, and the database-backed session store.

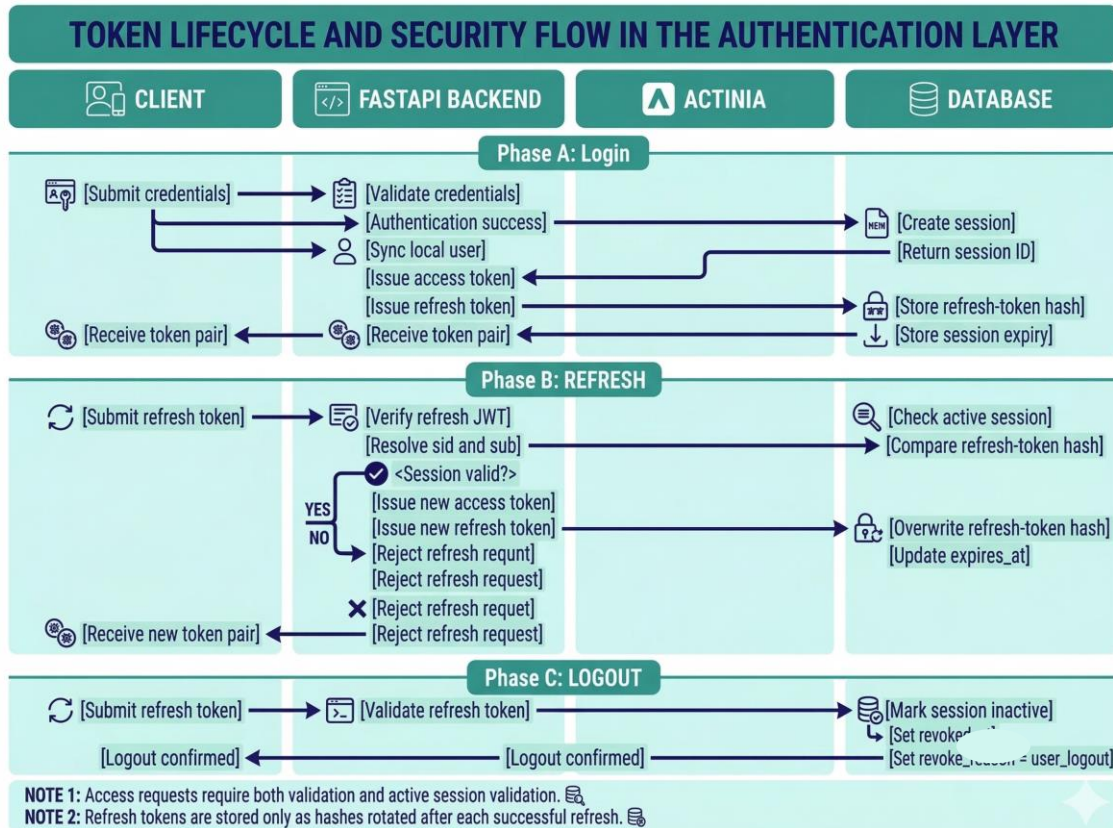


Figure 4.4. Authentication-Layer Token Lifecycle and Security Flow

At login, the client submits username and password to `/auth/login`. The backend validates them against Actinia, synchronises the local user record, and – before issuing any token – creates a new session row in the database. This step is essential: the generated database identifier becomes the session identifier (sid) embedded in both tokens. The response returns the token pair with basic user identity.

Access and refresh tokens are distinguished clearly. Both are JWTs signed with the configured secret and algorithm and carry an expiration claim (exp) and a token-type claim (type): the access token is `type="access"`, for authorising protected requests, and the refresh token is `type="refresh"`, reserved for renewal. The access token also carries contextual claims – internal user identifier (sub), session identifier (sid), Actinia username, role, and group – while the refresh token holds only the identity and session linkage needed for renewal. Validation rejects any token whose type does not match its expected purpose. Refresh-token storage is a notable security measure. The refresh token is never stored in plain text: after issuance, the backend computes a keyed hash and stores only that hash and aligns the

session expiration timestamp with the token's exp claim. The session row thus becomes the authoritative server-side binding for refresh validation, without persisting the plaintext token.

Access-token use follows a two-step validation. First the backend verifies the JWT signature, expiration, and type; then it resolves the session via the "sid" and sub claims and checks that it exists, is active, and has not expired, using the database "expires_at" field tied to the refresh-token lifetime. Each successful validation updates the session's "last_used_at". As a result, even valid access tokens stop being accepted once the linked session is deactivated or expires.

Token renewal is implemented through the /auth/refresh endpoint using a **sliding refresh** strategy. When a client presents a refresh token, the backend first validates the refresh JWT itself and then verifies that its hash matches the one stored for the active session identified by the token's "sid" and sub claims. If the token-session binding is valid and the session is still active and unexpired, the backend issues a new access token and a new refresh token bound to the same session identifier. The session row is then updated with the hash of the newly issued refresh token, and the expiration timestamp is reset to match the new refresh-token expiry. Because the old refresh-token hash is overwritten, previously issued refresh tokens for that session are no longer valid. This implements token rotation while preserving session continuity.

Logout, at /auth/logout, is based on refresh-token validation rather than access-token submission. The refresh token is validated against the active session; the session is then marked inactive, the revocation timestamp recorded, and the revoke reason set to "user_logout". From that point both refresh and access tokens fail, because the session-state check no longer succeeds — providing immediate server-side invalidation without external token blacklists.

The token lifecycle therefore rests on four principles: separating short-lived access tokens from longer-lived refresh tokens, binding tokens to a server-side session through sid, storing refresh tokens only as keyed hashes, and rotating the refresh token on each successful renewal. Together these allow access to be revoked centrally by changing session state, rather than relying on client-side token disposal alone.

4.3.3 User management capabilities

Beyond authentication and token handling, the platform exposes a structured set of user-management capabilities through the API. These capabilities are divided into two categories: **self-service operations** available to authenticated users for managing their own profile state, and **administrative operations** reserved for users with elevated roles. **Table 4.1** summarises the main user-related API endpoints and the responsibilities associated with each operation.

Endpoint	Method	Access scope	Responsibility
/me	GET	Authenticated user	Return the current user's local profile fields
/me	PATCH	Authenticated user	Update self-service profile fields (full_name, email)
/me/preferences	GET	Authenticated user	Retrieve stored user preferences
/me/preferences	PATCH	Authenticated user	Merge and persist user preference values
/users	GET	Admin/Superadmin	List usernames retrieved from Actinia
/users/{user_id}	POST	Admin/Superadmin	Create user in Actinia and upset local user record
/users/{user_id}	GET	Admin/ Superadmin	Read local user details by internal identifier
/users/{user_id}	DELETE	Admin/ Superadmin	Delete user from Actinia and remove local user record if possible

Table 4.1. User-related API Endpoints and Responsibilities

This separation allows the platform to support routine profile maintenance while restricting higher-impact account management tasks to administrators. At the self-service level, the API allows an authenticated user to retrieve their current local profile representation through the `/me` endpoint. The returned representation is intentionally limited to a defined subset of exposed fields, including the internal identifier, email, full name, role, group, preferences, and record timestamps. Profile modification is also supported through a dedicated update endpoint, but the implementation constrains writable fields to a narrow set, namely “full_name” and email. This design prevents authenticated users from modifying security-relevant attributes such as role or group through self-service requests. In addition to basic profile fields, the platform supports user-specific preference storage. Preferences are retrieved through `/me/preferences` and updated through a corresponding patch operation. Rather than replacing the entire preference object blindly, the implementation merges incoming key-value pairs into the existing preference structure stored in the local user record. This provides a lightweight mechanism for storing per-user application settings while preserving previously defined values unless they are explicitly overwritten.

A notable feature of the user model is the separation between Actinia-owned identity attributes and locally managed profile attributes. Synchronisation from Actinia updates only the

authoritative external fields — mapped role and group — while local attributes such as preferences are left untouched, so identity refreshes never overwrite application-level personalisation held in the local database.

The **administrative** layer exposes a broader set of capabilities. Users whose local role is admin or superadmin can request the list of users available in Actinia through an API endpoint that proxies the remote `/api/v3/users` resource and extracts usernames from the returned payload. Administrative users can also create new user accounts. In this workflow, the backend first creates the user in Actinia, then reads back the authoritative Actinia profile, maps role and group information into the local representation, and finally inserts or updates the corresponding local user row. Local profile fields such as email and full name can optionally be added during this process, while passwords are never stored in the local database. Administrative read and delete operations are also implemented. The API provides an admin-protected endpoint for reading local user details by identifier from the database. For deletion, the workflow is designed conservatively: the backend first verifies that the target user exists in Actinia, then deletes the remote Actinia account, and finally attempts to remove the corresponding local user record if present. If the local deletion fails because related records still exist, the transaction is rolled back locally and an explicit conflict response is returned, while preserving the fact that the remote Actinia deletion has already occurred. This behaviour reflects the system's practical integration with an external identity source while maintaining database integrity checks in the local layer.

Overall, user management is deliberately limited but well scoped: self-service for profile and preference maintenance, with account creation, listing, inspection, and deletion reserved for administrator roles. Throughout, the design keeps a clear distinction between authoritative identity data synchronised from Actinia and the local user metadata the application maintains itself.

4.4 Data management and reproducibility

4.4.1 Storage layout and dataset organisation

To support reproducibility and controlled execution across multiple processing stages, the platform organises uploaded files and generated artefacts through an explicit storage layout rather than storing files in an unstructured or case-by-case manner. The storage layer

distinguishes between persistent shared storage and temporary workspace storage. In the current configuration, persistent artefacts are stored under a host-side **userdata** root, while temporary intermediate files are written under a separate workspace directory intended for transient execution products. Matching container-side roots are defined as well, allowing the backend to translate host-visible file paths into paths accessible from the containerised GRASS/Actinia runtime. Within this layout, files are organised primarily by processing context. Persistent uploads and generated datasets are not kept in one flat directory but grouped under deterministic subdirectories reflecting the GRASS processing context – in particular the resolved location and the session-specific mapset. This keeps outputs for different spatial-reference systems or user sessions separated at the filesystem level and lets the provenance of a stored artefact be reconstructed directly from its path. The ingestion layer uses a staged writing strategy. Newly uploaded files are first written to a temporary workspace area for the current session; after validation and context resolution, the relevant files are moved to their persistent location under the shared userdata storage. This matters for workflows that need temporary extraction or inspection – compressed vector uploads, transient CSV exports, intermediate GeoJSON – and, by separating staging from persistent storage, avoids polluting the long-term artefact store with invalid or partial files.

The persistent store is further organised by dataset category. Uploaded sources live under an uploads branch, while major analytical artefacts – point-level clear-sky data, suitable rooftop surfaces, rooftop-area clear-sky aggregates, and real-sky simulation outputs – are written to dedicated category-specific directories. This separation improves navigability, reduces ambiguity between intermediate and final outputs, and aids retrieval by downstream stages. A second principle distinguishes temporary execution products from reproducible analytical artefacts. Temporary products – staging files, extracted archives, transient CSV exports, intermediate outputs – are written under the workspace area and may be cleaned up or overwritten. Persistent analytical outputs, by contrast, are materialised in stable, reusable formats, primarily Parquet datasets and partitioned result directories, so they can be reloaded by later stages without rerunning the earlier geospatial steps, strengthening traceability and reproducibility.

Finally, the storage layer is closely integrated with metadata persistence. File paths are normalised before being stored in the relational database, allowing the system to reference artefacts in a stable manner without depending excessively on raw absolute host paths.

This design supports later provenance tracking, job-to-artefact linking, and cross-stage artefact reuse, all of which are essential for a multi-stage computational pipeline in which outputs from one stage become structured inputs to subsequent stages.

4.4.2 Metadata and job tracking

Alongside filesystem storage, the framework keeps a relational metadata layer recording uploaded files, computational jobs, generated results, and authenticated sessions. Although some of this could be inferred from the filesystem, storing it explicitly in database tables supports structured tracing, status inspection, and an execution history that can be queried.

At the centre of this layer is the `models.jobs` table, which records computational runs as typed operations. Each job is assigned a unique database identifier and linked to a user through a foreign key relationship. Additional fields include the job type, *status*, optional error message, creation timestamp, and a *parameters* field stored in JSONB format. This design allows each run to be represented not only as an execution event, but also as an operation with an explicit type, state, and parameter context. An index on *(user_id, created_at)* supports chronological retrieval of jobs associated with a given user. **Figure 4.5** summarises the main entities of the metadata layer and the relationships used to track users, sessions, uploaded files, jobs, and generated results.

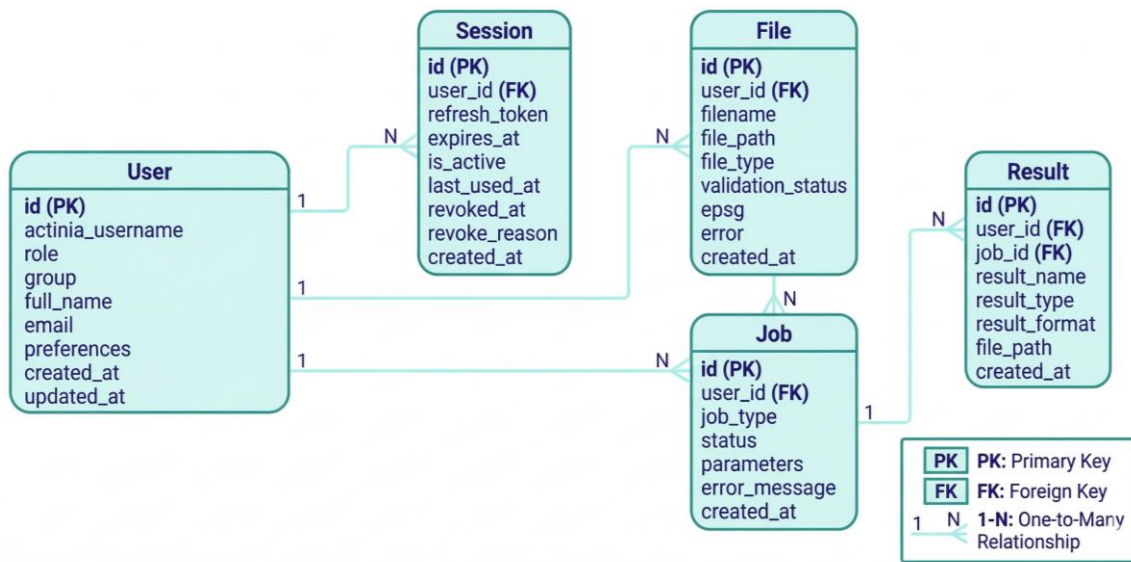


Figure 4.5. Entity-Relationship Diagram of the Metadata and Tracking Layer

Generated outputs are represented separately in the “models.results” table. Rather than embedding result information in the job record, each artefact is registered as its own entity linked to a user and a job by foreign keys, with a row storing result name, type, format, filesystem path, and creation timestamp. A uniqueness constraint on (job_id, file_path) prevents the same path being registered twice for one job.

Input data are traced through the “models.files” table. This table stores metadata about uploaded source files, including the original filename, stored filesystem path, file type, validation status, EPSG code, optional error message, and creation timestamp. This separation is methodologically useful because it distinguishes user-provided inputs from outputs generated by the analytical workflow, even though both are ultimately linked to the storage layer through file paths.

The metadata layer also includes the “models.sessions” table, which supports the access-control mechanism while providing additional execution-tracing context. This table stores session-related information such as the session identifier, expiration time, last activity time, and revocation status. In the current design, however, jobs are not linked to sessions through a foreign key relationship. Instead, jobs, files, and results are linked directly to users, while session records remain an auxiliary layer associated with authentication and execution context rather than a required dependency for the operation of the workflow itself.

Together these tables give a compact but effective structure for tracing execution: they distinguish uploaded inputs from generated outputs, record job status and parameter context, and preserve timestamps for chronological inspection. Though intentionally simple, the model is sufficient to support reproducibility, operational tracing, and structured linkage between users, jobs, files, results, and sessions.

4.4.3 Validation and quality gates

Before datasets enter the main pipeline, the backend applies a validation gate that rejects structurally invalid files, flags missing spatial-reference information, and enforces the domain-specific constraints required by later GRASS and Actinia stages. Validation is therefore not one binary decision but a sequence of format-, content-, and workflow-specific checks whose outcomes are recorded in the metadata layer. **Table 4.2** summarises the main validation checks applied during ingestion, together with their failure conditions and system responses.

Validation check	Applies to	Failure Condition	System Response
Shapefile Package completeness	ZIP-packaged Shapefiles	Missing .shp, .shx, .dbf, or .prj / .qj	Reject as invalid content
GeoPackage Layer Presence	GeoPackage vector uploads	No readable vector layer found	Reject as invalid content
File Format Detection	Raster and vector uploads	Format cannot be recognized or is unsupported	Reject as invalid format
CRS / EPSG Detection	Raster and vector uploads	Projection cannot be detected and no usable EPSG is available	Reject as invalid content
EPSG Mismatch Handling	Mixed raster/vector workflows	Dataset CRS differs from supplied override or companion layer	Continue with warning or override-based import path, depending on workflow
DSM Resolution threshold	Real-DSM raster workflow	nsres > 1.0 m or ewres > 1.0 m	Reject as resolution too coarse
Metadata Validation Status Persistence	All uploaded files	Validation fails at any stage	Store failure state and validation error in metadata table

Table 4.2. Validation Checks and Failure Modes in the Ingestion Layer

Validation distinguishes between raster and vector uploads and applies different checks depending on the data type. For vector uploads, the framework first determines whether the input is a direct vector file or a ZIP archive containing vector data. In the case of ZIP-packaged shapefiles, the archive is checked for the required structural components, including *.shp*, *.shx*, and *.dbf*, together with projection metadata in *.prj* or *.qj* form. If these components are missing, the upload is rejected as invalid content. A similar content gate is applied to GeoPackage uploads, where the dataset must contain at least one readable vector layer before it can proceed further in the workflow.

A second class of checks concerns spatial-reference integrity. For both raster and vector datasets, the backend tries to detect the CRS and derive a usable EPSG code; if it cannot, and no valid override is supplied, the import is rejected — missing CRS information is treated as a hard error, not a warning. The implementation also handles conflicts between user-supplied CRS and metadata detected from the dataset, continuing with the dataset's valid CRS where appropriate and recording a warning when an external EPSG override is ignored or a mixed raster–vector workflow proceeds through an override-based import.

In addition to general file validation, the platform applies domain-specific quality gates derived from the scientific assumptions of the workflow. A clear example is the real-DSM ingestion branch. After raster import, the spatial resolution is queried and compared against the threshold required for the high-resolution DSM pathway. If either *nsres* or *ewres* exceeds 1.0 m, the dataset is rejected with an explicit resolution failure. This is not merely a technical

import condition, but a methodological quality requirement intended to ensure that the real-DSM branch is used only when rooftop geometry can be represented with sufficient spatial detail.

Validation outcomes also persisted. Uploaded files are recorded with states such as `validated`, `invalid_format`, or `invalid_content`, together with EPSG information and any error messages. Validation therefore operates not only at request time but also as part of post-execution traceability: accepted and rejected uploads can be inspected later through the metadata layer, giving an auditable record of ingestion decisions.

Overall, validation combines format checks, CRS-integrity checks, and workflow-specific quality constraints, ensuring inputs are both structurally valid and methodologically suitable for the next stages. Making validation explicit at ingestion avoids deferring these problems to later computation and improves reproducibility and operational transparency.

4.5 Core geospatial processing pipeline (Actinia/GRASS)

4.5.1 Actinia integration and execution model

The geospatial runtime follows a hybrid integration with Actinia. Actinia is not used as a general-purpose remote execution engine; instead, the backend relies on it mainly to manage the GRASS workspace context, while the geospatial commands run directly inside the running Actinia container. The application backend handles orchestration, validation, and stage ordering, and the containerised Actinia environment provides the controlled runtime for GRASS operations.

At the configuration level, the integration depends on a few explicit environment parameters — the Actinia base URL, administrator-level backend credentials, and the running container’s identifier — which form the HTTP-facing layer. Through it, the backend uses protected Actinia endpoints for authentication, validating user credentials, and admin-scoped access for retrieving user metadata. Within the geospatial workflow itself, only a few Actinia API calls are used, mainly to verify that the required GRASS locations and mapsets exist.

Once the workspace context is ready, execution moves from HTTP-level interaction to container-level command execution. Rather than sending remote execution specifications and polling for completion, the backend resolves the GRASS mapset path inside the container

and invokes GRASS commands through restricted docker exec calls. Each service function prepares the parameters for the relevant GRASS module, resolves the location and mapset for the active session, and runs the command through a shared runner utility responsible for mapset-path discovery, optional overwrite handling, and error propagation when a GRASS command fails.

The execution model is therefore stage-oriented rather than job-queue-oriented. Each service function corresponds to a methodological stage – real-DSM import, DSM smoothing, slope and aspect derivation, horizon analysis, or clear-sky modelling – and the backend prepares and runs the required GRASS commands within the session-specific container, then handles artefact persistence, metadata registration, and error reporting. Workflow monitoring thus relies on direct command success or failure plus backend-side job and result tracking, not on remote Actinia job polling.

This choice has two main implications. It keeps the geospatial runtime tightly controlled, since all GRASS operations run inside the same containerised stack; and it gives the backend precise control over stage sequencing, validation gates, and artefact registration without depending on a separate remote execution lifecycle. The trade-off is that command orchestration stays an explicit backend responsibility rather than being delegated to higher-level Actinia services – but for this design it provides a reproducible, transparent bridge between the FastAPI backend and the GRASS runtime.

4.5.2 Location and mapset management

A further constraint is that every stage must run within a valid GRASS workspace context, defined here by a GRASS location combined with a session-specific mapset. Unlike a traditional approach where a user manually manages a working environment, the backend derives the spatial-reference system from the uploaded data, checks whether the corresponding location exists, and creates or reuses a mapset tied to the active session. This isolates the context of different sessions while keeping consistency across all stages of the same run.

The first step is selecting the GRASS location, which is defined by the input data’s spatial-reference system and so acts as a projection-specific workspace rather than a generic directory. Once a valid EPSG code is determined for the uploaded raster or vector, the backend checks whether the matching Actinia location exists and, if not, creates it through

an idempotent request; if it already exists, the request succeeds and processing continues. This avoids duplicate locations while ensuring a valid projection-specific workspace before analysis begins.

Mapset management is treated separately from location management. For user-initiated runs, the mapset name is derived from the session identifier following the pattern *session_<session_id>*. This effectively turns the mapset into a namespace scoped to a single session rather than a workspace shared across unrelated runs. The backend ensures that the required mapset exists within the selected location through a second idempotent call to Actinia. As a result, all uploaded datasets, intermediate GRASS products, and derived geospatial artefacts associated with the same analysis session remain contained within a stable and session-specific workspace context.

A key design choice is that the processing environment is adapted to the CRS of the uploaded data rather than forcing reprojection during workspace preparation: location selection follows the projection of the supplied raster or vector. GRASS operations then run in a spatial-reference context taken directly from the source data, reducing ambiguity in raster resolution, buffering, and other projection-sensitive calculations.

Reprojection or CRS override, where required, is therefore handled as a workflow-specific import concern rather than part of location selection — separating the definition of the workspace from any later decision about data adaptation or harmonisation during import.

The framework also separates the logical identity of the mapset in the application layer from the physical mapset path resolved inside the container. To support different deployment layouts, it does not assume a single fixed path but builds a list of candidates mapset roots from the configured userdata root, common Actinia GRASS-database layouts, temporary workspace layouts, and owner-specific namespace variants. The runner probes these candidates and validates each as a GRASS mapset by checking its structure — the mapset directory, an associated WIND file, and a valid PERMANENT directory in the parent location — performing a short scan of known roots before failing if none is found. This makes mapset resolution resilient to deployment differences while still enforcing the requirements of a valid GRASS workspace.

A further level of scoping comes from the Actinia resource owner. When ensuring locations or mapsets, the backend may include an X-Actinia-User header, so resources are created within the authenticated user's namespace; likewise, when resolving candidate mapset paths in the container, the runner considers multiple owner-name variants from both the configured Actinia service user and the effective per-request owner. This matters where Actinia stores resources under user-specific directories rather than a single shared root.

Together these rules give a reproducible workspace model: the GRASS location supplies the projection context, the mapset supplies the session context, and the backend resolves both consistently at the API level and inside the container. Later stages can then reference a stable, well-defined processing context without hard coding a filesystem layout or collapsing all runs into one shared workspace.

4.5.3 Data import: raster and vector

After the location and session-specific mapset are created, the backend imports the data. Import is staged and modular rather than a single direct transfer: uploaded files are first written to a temporary inspection area, where structure and metadata can be examined before they move to persistent shared storage. This is especially relevant for vector uploads arriving as ZIP archives of several related files, where the inspection directory may differ from the final persisted path.

Raster and vector imports are handled differently from the outset. For vector data, the backend first determines the nature of the uploaded object, distinguishing between direct vector-file uploads and ZIP-based uploads. When the input is a ZIP archive, the archive is extracted into a temporary directory, and its contents are inspected before import proceeds. In the current implementation, this inspection checks whether the archive contains a structurally valid Shapefile package. In practice, this means verifying the presence of the required *.shp*, *.shx*, and *.dbf* components together with projection information provided through either a *.prj* or *.qpj* file. Only once this structural check has succeeded can the dataset proceed to the next import step.

CRS resolution is central to import. For vector data, the backend tries to resolve a usable EPSG code from the dataset's embedded spatial-reference information; if none can be derived and no valid override is supplied, the import is rejected. Where CRS information is present, it sets the context for import. Crucially, import involves no generic reprojection:

rather than converting incoming datasets to a fixed standard CRS, the workspace is prepared to match the dataset's CRS, and the import runs in that context. The framework thus adapts the workspace to the data, not the data to a fixed workspace.

Once the CRS has been resolved, the backend verifies the existence of the matching GRASS location and session-specific mapset before continuing with the import. The uploaded file is then moved into the persistent uploads branch under the appropriate location and mapset structure. For ZIP-based vector uploads, the effective import source may be the extracted Shapefile path rather than the original archive file itself. Vector import is carried out through the GRASS *v.import* command, while raster import, including DSM and DTM inputs, is performed through *r.in.gdal*.

Import therefore acts as both a validation gateway and a spatial-context enforcement layer, ensuring files are structurally valid, projection-aware, and imported into the correct GRASS context before any later stage runs. It also reinforces a core rule of the framework: spatial processing stays anchored to the input data's CRS, and reprojection occurs only where a specific workflow branch requires it, never as a default preprocessing step.

4.5.4 Derived-surface workflow

If no suitable high-resolution DSM is available, the framework uses a derived-surface workflow to generate a synthetic surface for the solar-preprocessing stages. The goal is not to import an existing raster but to construct a rooftop-oriented elevation model from building footprints and, where needed, auxiliary elevation sources — the operational implementation of the derived-surface concept introduced earlier.

The first configuration applies when the footprint dataset already carries a height attribute per structure. The backend extracts these heights from the vector attribute table and normalises them before later raster operations, excluding buildings with missing or invalid heights so unreliable values do not propagate. A new attribute field is then added to store the processed heights used in the remaining stages.

At the same stage, an additional field stores slope-related information for later approximation of rooftop inclination, since explicit roof-shape geometry is absent from footprint-based input. Slope values are assigned within a predefined range so the workflow can pro-

ceed without detailed roof-form data; this does not reconstruct full morphology but gives a practical approximation for the later solar-preprocessing and suitability stages.

The footprint layer is then rasterised, each cell within a footprint taking that building's elevation, producing an initial rooftop surface. On its own this would render each building as a uniform prism, a poor approximation of rooftop form. To improve it, the workflow applies a progressive inward-buffering strategy: a sequence of progressively smaller regions is generated inside each footprint, and these nested regions are later stacked to construct the synthetic rooftop surface.

The spacing between successive buffer levels depends on the CRS of the input data. If the source data use a projected CRS with metric units, the inward buffer step size is equal to the user-defined metric distance d_m , so that the buffer distance is simply:

$$d = d_m \quad (4.1)$$

If the source data use a geographic CRS with angular units, the same metric step must first be converted into angular distance while taking local latitude into account. In that case, the spacing between successive inward buffers is given by:

$$d_{deg} = \frac{d_m}{R \cdot \cos(\phi)} \cdot \frac{180}{\pi} \quad (4.2)$$

where ϕ denotes latitude. This conversion ensures that the effective physical spacing between successive inward buffer levels remains approximately stable even when coordinates are expressed in degrees.

A second configuration applies when the footprint layer has no height attribute, in which case structural height is derived from auxiliary elevation rasters. When both a DSM and a DTM are available, building height is estimated as the difference between the upper-surface and bare-earth elevations:

$$H_{building}(x, y) = DSM(x, y) - DTM(x, y) \quad (4.3)$$

These values represent the height of structures above the terrain and are linked back to the footprint polygons to give each building a representative height attribute. From there, the workflow converges with the branch in which heights were already present.

The final product is a synthetic DSM that approximates rooftop geometry for the later pipeline, supporting smoothing, terrain-derivative calculation, horizon analysis, and solar-radiation modelling within the same framework used for real DSM inputs. Its value is that the system keeps operating when high-resolution DSM data are unavailable: rather than terminating, the framework reconstructs an approximate rooftop surface from more common, lightweight sources, extending its applicability to a wider range of inputs.

4.5.5 High-resolution DSM workflow

When a high-resolution DSM is supplied directly, the framework treats the uploaded raster as the primary surface representation. Unlike the derived-surface workflow, it does not reconstruct rooftop geometry from footprints or auxiliary sources but assumes the raster already captures rooftop form at a resolution suitable for direct solar-radiation modelling. The challenge here is therefore not surface synthesis but validating whether the raster is detailed enough for rooftop-scale analysis.

The workflow first analyses the raster’s spatial-reference system and resolves the CRS/EPSG definition needed to place it in the GRASS context. As before, the raster is not reprojected during ingestion; instead, the workspace is prepared to match its CRS. Once the location and session-specific mapset exist, the raster is imported and becomes the reference surface for the remaining preprocessing.

A dedicated quality gate is then applied in order to determine whether the uploaded raster is suitable for the real-DSM workflow. In particular, the north-south and east-west raster resolutions are obtained from the dataset metadata, and a single effective resolution is defined as:

$$\textit{resolution} = \max(\textit{nsres}, \textit{ewres}) \quad (4.4)$$

This definition ensures that the acceptance decision is governed by the coarser of the two raster dimensions. As a result, an anisotropic raster cannot be accepted solely on the basis of one finer resolution value while the other dimension remains too coarse. The DSM is accepted only if:

$$\textit{resolution} \leq 1 \text{ m} \quad (4.5)$$

If this condition is not satisfied, the workflow does not proceed along the real-DSM branch. Instead, the raster is rejected for this pathway, and the user is informed that the provided DSM is too coarse for rooftop-scale processing under the high-resolution workflow.

This gate is methodological, not merely technical. The downstream solar-modelling stages assume rooftop geometry is represented in enough detail to preserve meaningful information about inclination, aspect, and directional obstruction. A too-coarse raster may remain computable but no longer matches the workflow's scientific assumptions, so continuing would risk outputs that are formally computable yet unreliable. The resolution filter therefore protects the scientific validity of the branch, not just execution feasibility.

Once it passes validation, the DSM underpins all subsequent GRASS-side preprocessing – smoothing, terrain-derivative calculation, horizon analysis, and clear-sky modelling. The real-DSM branch is more direct than the derived-surface workflow, since no rooftop surface must be reconstructed first; its advantage is preserving the geometric detail already in the raster while enforcing a strict quality gate for rooftop-scale work.

4.5.6 Geospatial preprocessing

With the DSM established, the system derives the terrain attributes needed for solar-radiation modelling. These preprocessing steps run inside the containerised GRASS GIS environment to ensure reproducibility.

First, the DSM is smoothed to reduce local noise and small-scale artefacts that would otherwise degrade the derived attributes – important especially for synthetic DSMs or heterogeneous source data. Smoothing uses neighbourhood raster filters via the GRASS module `r.neighbors` in two stages: an averaging filter with a neighbourhood size of seven removes high-frequency irregularities, followed by a median filter with a smaller neighbourhood that preserves rooftop geometry while avoiding over-smoothing.

From the filtered surface, the GRASS module `r.slope.aspect` computes the slope and aspect rasters, representing the inclination at each cell and the orientation of the surface normal. Both are essential for solar-irradiance modelling, since rooftop inclination and orientation directly affect the radiation received.

```
# Two-stage smoothing: average (size 7), then median (size 3)
smoothing_pipeline = [("average", 7), ("median", 3)]
```

```

prev = dsm_name
for idx, (method, size) in enumerate(smoothing_pipeline, start=1):
    is_last = idx == len(smoothing_pipeline)
    out_name = final_smoothed_name if is_last else f"{final_smoothed_name}_s{idx}"
    run(["r.neighbors", f"input={prev}", f"output={out_name}",
        f"method={method}", f"size={int(size)}", "--overwrite"])
    prev = out_name

# Slope and aspect from the smoothed DSM
run(["r.slope.aspect", "-e", f"elevation={final_smoothed_name}",
    f"slope={slope_name}", f"aspect={aspect_name}",
    "format=degrees", "min_slope=0.0", "--overwrite"])

```

Listing 4.1. Two-stage DSM smoothing and slope/aspect derivation in GRASS.

In addition to local surface orientation, another important factor influencing solar exposure is the visibility of the sky dome. For this reason, the preprocessing stage also includes a horizon analysis in which directional obstructions are evaluated around each raster cell. This computation is performed using the GRASS GIS module *r.horizon*, which derives horizon elevation angles along multiple directions surrounding the area of interest. The directional indices of these horizon outputs follow the GRASS *r.horizon* convention, where 0° corresponds to East, 90° to North, 180° to West, and 270° to South. These horizon angles represent directional constraints on incoming solar-radiation and are later used to account for shading effects caused by surrounding terrain or structures. Because this analysis requires the evaluation of multiple directional profiles across the DSM, it can become computationally demanding for large raster datasets. To address this issue, the implementation executes the horizon calculations in parallel within the GRASS GIS environment.

As a result of this preprocessing stage, the system generates a set of raster layers describing the geometric properties of the surface, including the smoothed DSM, slope raster, aspect raster, and a family of GRASS-convention horizon rasters representing sky obstruction in different directions. Together, these layers provide the geometric basis for the subsequent clear-sky solar-radiation modelling stage.

After the derivation of terrain attributes, the workflow continues with the clear-sky solar-radiation modelling stage. In this phase, the GRASS GIS model *r.sun* is employed to compute solar irradiance under clear-sky atmospheric conditions using the previously derived

surface geometry and horizon constraints. More specifically, the smoothed DSM together with the slope and aspect rasters describes the geometric configuration of the scene, while the computed horizon layers define the shading constraints that affect incoming solar radiation from specific azimuth directions.

To account for seasonal variability while keeping the problem computationally tractable, the approach is implemented using a set of twelve representative days distributed throughout the year. The selected monthly reference dates and their corresponding day-of-year (DOY) indices are reported in **Table 4.3**, following the monthly average-day convention adopted in the reference infrastructure. For each of these representative days, solar radiation is evaluated at multiple timesteps during the daylight period. The temporal sampling is performed at fifteen-minute intervals between 04:00 and 22:00, resulting in a dense representation of daily irradiance conditions. Because the number of required simulations becomes large when combining representative days with multiple timesteps, the radiation modelling stage is also parallelized within the GRASS environment.

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Date	17th	16th	16th	15th	15th	11th	17th	16th	15th	15th	14th	10th
DOY	17	47	75	105	135	162	198	228	258	288	318	344

Table 4.3. Representative Monthly Average Days and Corresponding Day-of-Year (DOY)

For each simulated timestep, the *r.sun* model produces separate raster layers for the direct (beam) and diffuse components of solar radiation. These layers represent the theoretical solar irradiance that would reach each rooftop surface in the absence of short-term atmospheric variability such as cloud cover. At this stage, the GRASS-based part of the workflow is complete. The resulting raster outputs are then transformed into structured analytical artefacts for the subsequent stages of the pipeline, as discussed in **Section 4.6**.

4.6 Simulation methodology

4.6.1 Transformation of GRASS outputs into analytical artefacts

Rather than using the large set of raster-based radiation outputs directly in the final stage of the photovoltaic simulation process, the framework transforms these outputs into structured analytical artefacts that can be reused efficiently in later stages of the workflow. These artefacts serve as intermediate data products that make it possible to evaluate dif-

ferent weather conditions and simulation scenarios without repeating the computationally expensive process of clear-sky radiation modelling. In this sense, the transformation of GRASS outputs into analytical artefacts is not merely a change in storage format, but a methodological transition from raster-oriented geospatial computation to data-oriented analytical processing.

To accomplish this transition, a dedicated data-engineering stage converts the raster-based outputs into structured tabular datasets. The spatial information describing rooftop geometry and solar exposure is extracted from the GRASS raster layers and reorganised into a point-based representation: rather than continuing in raster form, the system samples rooftop surfaces, generating analytical points across the available rooftop area. Each point is a spatial sampling unit carrying the local geometric and radiative characteristics of the surface.

Sampling points are created from the rasterised building or rooftop mask produced earlier, converted into a vector point dataset marking where surface attributes will be evaluated. Each point receives a unique identifier, keeping results reproducible across runs and providing a stable reference to the same locations across the datasets produced later. The point representation thus becomes a persistent scaffold linking geometric attributes, radiation values, and later rooftop-level aggregates.

For each point, the system extracts attributes from the rasters computed in the earlier GRASS stages: geographic coordinates, plus slope and aspect from the smoothed DSM, describing the surface geometry at the sampling location. The directional horizon information from the horizon-analysis stage is also attached to each point, giving a local sky-visibility profile of angular obstruction in different directions. The resulting dataset records not just position but a compact description of rooftop geometry and directional exposure at each sampled location. The directional horizon values stored in this static point artefact retain the original GRASS *r.horizon* directional convention. In this convention, 0° corresponds to East, 90° to North, 180° to West, and 270° to South. They are therefore not directly equivalent to *pvl* or *PVGIS* azimuths. When these horizon values are used in *pvl*-based calculations, they must first be converted to the *pvl*/*PVGIS* azimuth convention, as described later for the fast horizon-based simulation branch.

At this stage of the workflow, two analytical datasets are produced. The first dataset contains the static geometric attributes of the rooftop sampling points. These attributes include the point coordinates, the slope of the rooftop surface at the point, the aspect of the rooftop surface at the point, and the directional horizon profile that characterises sky visibility in the original GRASS *r.horizon* directional convention. Since these properties are derived from the rooftop geometry itself and do not vary over time, they form a static dataset describing the spatial characteristics of the rooftop sampling points.

The second dataset contains the clear-sky solar-radiation time series associated with each rooftop sampling point. This dataset follows the same temporal sampling strategy adopted during the clear-sky radiation modelling stage. In the current implementation, the clear-sky radiation model is evaluated for twelve representative days distributed throughout the year, and for each of these days radiation values are computed at 15-minute intervals during the daylight period. For each timestamp in this temporal grid, the dataset stores the corresponding radiation values for each rooftop point. In particular, the direct and diffuse components of solar radiation derived from the *r.sun* model are stored together with the temporal identifier of the simulation step. This dataset therefore captures the time-dependent variation of rooftop solar irradiance under clear-sky assumptions, while maintaining a direct link to the underlying spatial sampling structure.

Separating static geometric attributes from time-dependent radiation values is a deliberate architectural choice. Storing both together would repeat each point's static properties for every timestamp, creating substantial redundancy. Keeping one dataset for static attributes and another for time-varying radiation reduces duplication while still allowing the two to be recombined efficiently downstream, improving storage efficiency and flexibility for large point sets and dense temporal grids.

Because extraction and transformation may involve large point sets and many timestamps, the framework parallelises this stage. The aim is not to change the data's structure but to speed up converting GRASS outputs into structured datasets, keeping the stage compatible with the scale of the sampling problem while preserving a reproducible representation of the geospatial results.

Both point-level datasets are stored in Parquet for efficient compression, structured storage, and analytical querying. They are the first major analytical artefacts of the workflow,

providing the basis for later simulation while decoupling expensive geospatial processing from subsequent computation. To stay consistent across stages, the required spatial products — smoothed DSM, slope and aspect rasters, horizon layers — are retrieved through a session-based context-resolution mechanism that links each artefact to the correct execution context.

With the point-level artefacts in place, the workflow moves from describing rooftop geometry point by point to identifying rooftop surfaces that meet the geometric requirements for panel placement. This stage detects or ingests rooftop installation areas and represents them as a second structured artefact feeding later estimation.

Suitable-area generation has two configurations. In the first, the user supplies a dataset of suitable rooftop polygons — perhaps from prior segmentation or external workflows. The backend then validates the upload, imports it into GRASS, extracts the required attributes, and integrates the polygons without automatic detection, treating them as the authoritative candidate surfaces.

In the second, the system detects suitable regions automatically, using terrain-derived attributes — chiefly slope and aspect — together with a rooftop mask from the building footprints. Surface inclination and orientation are the primary criteria, since they strongly affect photovoltaic exposure and performance, and rooftop cells are evaluated by threshold constraints on the slope and aspect rasters, restricted to rooftop pixels.

If $S(x, y)$ denotes the slope at location (x, y) and $A(x, y)$ denotes the aspect angle at the same location, a pixel is considered geometrically suitable when it satisfies the following inequalities:

$$S_{\min} \leq S(x, y) \leq S_{\max} \quad (4.6)$$

$$A_{\min} \leq A(x, y) \leq A_{\max} \quad (4.7)$$

These inequalities define the preferred ranges of slope and azimuth orientation for photovoltaic exposure. Because the objective is to identify installation areas on rooftops rather than on the surrounding terrain, the suitability analysis is constrained by the rooftop mask derived from the building footprint dataset. This prevents terrain pixels or non-rooftop sur-

faces from being incorrectly classified as candidate photovoltaic installation locations. Accordingly, the suitability raster can be expressed as

$$\mathbf{Suitable}(x,y) = \mathbf{RoofMask}(x,y) \wedge \mathbf{SlopeMask}(x,y) \wedge \mathbf{AspectMask}(x,y) \quad (4.8)$$

This raster marks the pixels meeting the geometric suitability criteria. To obtain candidate surfaces, the suitable regions are converted to vector polygons using GRASS raster-to-vector conversion, each polygon representing a rooftop surface that may host an installation.

For each candidate polygon, the workflow computes descriptive attributes. These include representative geometry such as mean slope and mean aspect, giving a compact summary of each surface, and the effective installation area – the total rooftop area available for placement within the suitable region.

The effective area is computed from the number of suitable raster cells contained within the polygon. Let N_{cells} denote the number of suitable raster cells associated with a given rooftop region and let A_{cell} denote the area of a single raster cell. The effective rooftop area can then be written as:

$$A_{effective} = N_{cells} \cdot A_{cell} \quad (4.9)$$

The calculation of A_{cell} depends on the coordinate system of the dataset. If the raster grid is defined in a projected coordinate system with metric units, the cell area can be computed directly from the raster resolutions:

$$A_{cell} = r_{ns} \cdot r_{ew} \quad (4.10)$$

where r_{ns} and r_{ew} are the north-south and the east-west resolution of the grid, respectively.

If the dataset is defined in a geographic coordinate system using angular units, cell dimensions must be converted into metric units by taking latitude into account. In this case, the cell-area calculation depends on the Earth's radius and the local latitude, since the east-west dimension varies with $\cos(\phi)$. This distinction is important because it ensures that effective rooftop area is computed consistently even when the source data do not originate in a projected metric CRS.

After geometric attributes and effective area are computed, each suitable polygon is linked to its source building footprint through a spatial join that assigns a building identifier. This preserves the relation between candidate areas and their buildings, supporting later building-level aggregation and interpretation. Centroid coordinates are also computed as a representative reference point. The dataset is then stored in Parquet as the second major analytical artefact, holding the geometric and spatial properties of the suitable surfaces.

Next, an aggregation stage integrates the earlier artefacts to compute rooftop-level clear-sky radiation statistics. From here the workflow no longer uses GRASS for the main computations; the remaining steps operate directly on the Parquet datasets, marking the transition from geospatial preprocessing to data-oriented analysis.

This stage turns point-level clear-sky radiation into rooftop-surface-level statistics, since installation and energy estimation ultimately occur at the level of candidate areas rather than individual points. It combines three datasets produced earlier. The first dataset contains the static attributes of rooftop sampling points, including coordinates, slope, aspect, and the GRASS-convention horizon profile. The second holds the clear-sky radiation time series for each rooftop point – direct and diffuse components at each simulation time. The third holds the suitable rooftop surfaces from the suitability stage, with their geometric descriptors, identifiers, and effective areas.

The first task is to associate each sampling point with the suitable surface that contains it, establishing the link needed to transfer radiation from point to surface level. The system then aggregates clear-sky radiation for each suitable region at each simulation timestamp.

Let $R_{beam}(p, t)$ and $R_{diff}(p, t)$ denote the direct and diffuse radiation components computed at point p and time t . For a rooftop area A_i containing N_i sampling points, the mean radiation values associated with that rooftop surface are computed as:

$$\bar{R}_{beam}(A_i, t) = \frac{1}{N_i} \sum_{p \in A_i} R_{beam}(p, t) \quad (4.11)$$

$$\bar{R}_{diff}(A_i, t) = \frac{1}{N_i} \sum_{p \in A_i} R_{diff}(p, t) \quad (4.12)$$

These values give the average clear-sky irradiance on each rooftop surface at a given timestep. Aggregating to surface level reduces dimensionality while retaining the essential spatial variability of exposure across regions, yielding a dataset of timestamped clear-sky statistics per suitable area — the baseline that is later adjusted for real atmospheric conditions.

This aggregated dataset is also stored in Parquet, using columnar storage, compression, and fast access, and is partitioned by timestamp so later stages can read specific timesteps without scanning the full series. This is especially useful for batch and weather-driven real-sky simulation, where timestamp-specific access is repeated.

This design decouples clear-sky modelling from later simulation: because the statistics are computed once and stored as artefacts, the framework can evaluate many weather conditions and scenarios without repeating the expensive GRASS-based processing, improving efficiency and scalability when the same baseline is reused under different meteorological inputs.

The rooftop-level clear-sky dataset is therefore a key intermediate result: it holds the baseline statistics for each suitable installation area and feeds directly into the real-sky photovoltaic simulation stage.

4.6.2 Real-sky computation and weather handling

With the clear-sky baseline ready, the workflow incorporates real atmospheric conditions. Clear-sky radiation is the theoretical irradiance available at a rooftop, but actual production also depends on short-term weather — cloud cover, atmospheric turbidity, air temperature, wind speed. The framework integrates external meteorological data to transform the baseline into a realistic real-sky irradiance representation for downstream estimation.

Weather data can be ingested in three modes, reflecting different uses of the workflow. In the first, the user uploads a weather table in CSV or EPW format, expected to contain time-indexed variables such as direct-normal irradiance, diffuse-horizontal irradiance, air temperature, and wind speed; timestamps are parsed and normalised to UTC for consistency.

In the second mode, meteorological data are retrieved automatically from an external web-based service. In the current implementation, *Open-Meteo* is used as a lightweight and freely accessible source of meteorological time series. To query this source, the framework first

determines a representative geographic location for the study area. When rooftop coordinates are available, the mean latitude and longitude of the candidate surfaces are used. Otherwise, a centroid-based approximation of the rooftop dataset is used instead. This representative location is then passed to the external service in order to retrieve the required weather data.

In the third mode, weather is supplied at runtime by an external simulation environment. In this co-simulation setting, variables are not read from files or fetched from a service but received step by step from an externally controlled process, letting the photovoltaic workflow act as a component in a broader coupled simulation.

Whatever the source, all weather inputs pass through a common preprocessing stage. Raw series may have non-uniform intervals or resolutions that do not match the workflow, so the data are resampled to a uniform 15-minute interval, matching the clear-sky stage. Gaps or irregular samples are filled by interpolation followed by forward- and backward-fill, keeping the required fields defined across the simulation horizon.

```
# Per-variable aggregation / interpolation policy
policy = {
    "dni": {"agg": "mean", "interpolate": True, "required": True},
    "dhi": {"agg": "mean", "interpolate": True, "required": True},
    "temp_air": {"agg": "mean", "interpolate": True, "required": True},
    "wind_speed": {"agg": "mean", "interpolate": True, "required": True},
}

# Resample to a fixed 15-minute grid, then fill remaining gaps
resampled = deduped.resample(target_freq).agg(agg_map) # target_freq = "15min"
interp_cols = [c for c in value_cols if policy[c]["interpolate"]]
resampled[interp_cols] = resampled[interp_cols].interpolate(
    method=interpolation_method, limit_direction="both")
resampled = resampled.ffill().bfill()
```

Listing 4.2. Resampling heterogeneous weather inputs to a uniform 15-minute grid.

Once the weather-data have been normalised and resampled, the corresponding real timestamps must be linked to the precomputed clear-sky reference states generated in earlier stages. Because the clear-sky model is evaluated only for a limited set of representative days and discrete daylight timesteps, real timestamps cannot be matched directly to the baseline outputs. Instead, the framework applies a deterministic two-stage mapping proce-

cedure. First, the time-of-day component of each real timestamp is mapped to the nearest 15-minute interval within the clear-sky simulation window, defined in the current implementation as 04:00 to 22:00. Second, the day-of-year component is mapped to the nearest representative day used in the clear-sky simulations, according to the monthly reference day set reported in **Table 4.3**. Through this procedure, each real timestamp is associated with a corresponding clear-sky snapshot defined by a representative day and a discretised time of day.

After the appropriate clear-sky reference has been identified, the system adjusts the clear-sky irradiance values to reflect actual atmospheric conditions described by the weather data. This is done through a scaling procedure that compares observed direct and diffuse irradiance values with their corresponding clear-sky counterparts. Let DNI_{real} and DHI_{real} denote the direct and diffuse irradiance values provided by the weather dataset and let DNI_{clear} and DHI_{clear} denote the corresponding precomputed clear-sky values. The scaling coefficients are defined as:

$$k_{dir} = \frac{DNI_{real}}{DNI_{clear}} \quad (4.13)$$

$$k_{diff} = \frac{DHI_{real}}{DHI_{clear}} \quad (4.14)$$

These coefficients represent the attenuation of solar radiation under real atmospheric conditions, including cloud cover and related variability. They are applied to the precomputed clear-sky baseline to estimate the effective irradiance on each rooftop surface under real-sky conditions.

```
# Clear-sky reference at this timestep (pvlib)
cs = self.pv_location.get_clearsky(pd.DatetimeIndex([ts_utc]), model=self.clearsky_model)
dni_clear, dhi_clear = float(cs["dni"].iloc[0]), float(cs["dhi"].iloc[0])

# Scaling factors from real vs clear-sky irradiance
k_dir = _safe_ratio(step_weather["dni"], dni_clear)
k_diff = _safe_ratio(step_weather["dhi"], dhi_clear)

# Apply to the precomputed clear-sky area means -> real-sky GTI
gti_area = beam_clear_area_mean * k_dir + diff_clear_area_mean * k_diff
```

```
# PVWatts DC power
pdc_w = pvlib.pvsystem.pvwatts_dc(
    effective_irradiance=gti_area, temp_cell=temp_cell_c,
    pdc0=self.pdc0_W, gamma_pdc=self.gamma_pdc, temp_ref=25.0)
```

Listing 4.3. Real-sky irradiance scaling and PVWatts DC-power estimation.

The output of this stage is a weather-aligned real-sky irradiance representation linked to the rooftop-level clear-sky baseline. In methodological terms, it acts as the bridge between the precomputed solar-radiation artefacts and the subsequent stage of photovoltaic energy conversion.

4.6.3 Fast horizon-based simulation approach

In addition to the workflow-based photovoltaic simulation path described in the previous sections, the framework also supports a fast horizon-based simulation approach as an alternative downstream branch for selected rooftop units. The purpose of this branch is not to replace the main artefact-driven workflow, but to provide a lighter simulation strategy that can estimate photovoltaic behaviour at the level of suitable rooftop areas without relying on the full downstream clear-sky area-means scaling chain. In methodological terms, this approach remains consistent with the broader framework because it still depends on rooftop geometry, horizon-related information, and weather inputs. However, it simplifies the downstream simulation logic by moving to a centroid-level horizon-aware photovoltaic computation performed directly with *pvlib*.

The branch operates on selected PV system areas, given as suitable rooftop polygons – supplied directly by the user or resolved from a previously generated suitable-area artefact in the same session. For each area it requires at least a stable identifier, polygon geometry, effective area, and representative slope and aspect. It also depends on the earlier horizon-analysis outputs, which provide the local directional obstruction needed for the centroid-based simulation, and on weather input, resolved through the same preparation logic as before, including timestamp normalisation and column mapping.

A central methodological feature of this branch is that horizon information is resolved at the level of the centroid of each selected rooftop area. Let A_i denote the geometry of a suitable rooftop polygon. Its representative point for the fast simulation branch is taken as the polygon centroid,

$$C_i = \text{centroid}(A_i) \quad (4.15)$$

where C_i is the centroid associated with rooftop area i . This centroid is used in two different ways. First, it is used in the native CRS of the study area to sample local horizon values from the horizon rasters generated by the framework itself. These values are extracted from the family of direction-specific horizon outputs and together form a local horizon profile for the selected rooftop area. Second, the same centroid is transformed into geographic coordinates in order to retrieve an external horizon profile from PVGIS through the corresponding *pvlib* interface. As a result, each selected rooftop unit is associated with two directional horizon descriptions: one derived locally from the framework's own geospatial preprocessing outputs, and one retrieved externally from PVGIS.

Before the local and external horizon profiles are combined, their azimuth conventions must be harmonised. This step is necessary because the horizon rasters generated by GRASS *r.horizon* and the horizon profiles used by *pvlib*/PVGIS do not follow the same directional convention. In the GRASS *r.horizon* convention, directions are expressed with 0° corresponding to East, 90° to North, 180° to West, and 270° to South, increasing counterclockwise from East. By contrast, *pvlib* and PVGIS use the common solar-azimuth convention in which azimuths are expressed clockwise from North, with 0° corresponding to North, 90° to East, 180° to South, and 270° to West.

For this reason, each local horizon direction extracted from the GRASS outputs is converted to the *pvlib*/PVGIS convention before interpolation, profile merging, and solar-azimuth lookup. The conversion is defined as

$$\theta_{pvlib} = (90 - \theta_{GRASS}) \bmod 360 \quad (4.16)$$

where θ_{GRASS} is the direction associated with the local *r.horizon* output and θ_{pvlib} is the corresponding azimuth in the *pvlib*/PVGIS convention. For example, a GRASS direction of 270° corresponds to South and is therefore converted to a *pvlib* azimuth of 180°. After this conversion, the local horizon profile and the PVGIS horizon profile are expressed on a common azimuth convention and can be compared direction by direction.

Because the local and external horizon descriptions may differ, the framework combines them into a single merged horizon profile before photovoltaic simulation is performed. Let

$h_i^{local}(\theta)$ denote the local horizon elevation angle for rooftop area i at azimuth direction θ , after conversion to the *pvl*ib/*PVGIS* azimuth convention. Let $h_i^{ext}(\theta)$ denote the corresponding external horizon angle retrieved from *PVGIS* and interpolated onto the same azimuth grid. The merged horizon profile is then defined direction by direction as

$$h_i^{merged}(\theta) = \max(h_i^{local}(\theta), h_i^{ext}(\theta)) \quad (4.17)$$

This maximum rule is methodologically important because it preserves the most restrictive horizon condition at each azimuth direction. In other words, the resulting horizon profile reflects the higher of the two obstruction estimates for each angle and therefore avoids under-representing directional sky obstruction in the downstream irradiance calculation. The merged profile is thus treated as the effective horizon condition for the selected rooftop unit. In implementation terms, the *GRASS*-derived local horizon profile is first converted to the *pvl*ib/*PVGIS* azimuth convention, and the external *PVGIS* horizon profile is then interpolated onto this common azimuth grid so that the two profiles can be compared consistently before the maximum is taken.

Once the merged horizon profile has been constructed, the framework performs a direct weather-driven photovoltaic simulation for each selected rooftop area using *pvl*ib. For each simulation timestamp, solar position is first computed at the centroid location. The weather-data provide the DNI, DHI, air temperature, and wind speed required by the branch. If GHI is not explicitly available in the weather input, it is reconstructed from the direct and diffuse components according to

$$GHI = DNI \cdot \cos(z) + DHI \quad (4.18)$$

where z is the solar zenith angle. The merged horizon profile is then interpolated to the solar azimuth corresponding to each timestamp. Since both the merged horizon profile and the solar azimuth are expressed in the *pvl*ib convention, the lookup is performed within a consistent directional reference system. If the solar elevation angle is lower than or equal to the merged horizon elevation for the same direction, the direct component is treated as blocked by the horizon. In simplified form, this condition can be written as

$$\alpha_{sun}(t) \leq h_i^{merged}(\theta_{sun}(t)) \quad (4.19)$$

where $\alpha_{sun}(t)$ is the solar elevation angle and $\theta_{sun}(t)$ is the solar azimuth at time t . Under this condition, the DNI contribution is set to zero in the adjusted irradiance calculation, while the diffuse contribution is retained. In this way, the branch applies an explicit horizon-based masking procedure before plane-of-array irradiance is computed.

After horizon masking has been applied, the framework computes the incident irradiance on the tilted rooftop plane using *pvlb* irradiance models. The representative slope and aspect of the selected rooftop area are used as the tilt and azimuth parameters for this calculation. However, the representative aspect values originate from the GRASS-derived suitable-area workflow and therefore follow the GRASS directional convention. Before being passed to *pvlb* as the surface azimuth, they are converted to the *pvlb* convention using the same angular transformation applied to the local horizon directions. This ensures that both the obstruction profile and the receiving surface orientation are interpreted consistently within *pvlb*.

The resulting plane-of-array irradiance is then combined with the ambient weather variables in order to estimate photovoltaic cell temperature. In the current implementation, the branch supports both SAPM- and PVsyst-based temperature models, computed through the corresponding *pvlb* routines using plane-of-array irradiance, air temperature, wind speed, and preset model parameters. The estimated cell temperature is subsequently used in a *PVWatts* power model to compute direct-current photovoltaic output. In implementation-faithful form, the DC power calculation follows

$$P_{dc} = P_{dc0} \left(\frac{E_{effective}}{1000} \right) [1 + \gamma_{pdc}(T_{cell} - T_{ref})] \quad (4.20)$$

where $E_{effective}$ corresponds to the plane-of-array irradiance, γ_{pdc} is the temperature coefficient used in the *PVWatts* model, and T_{ref} is the reference cell temperature. When AC output is requested, the framework then applies *pvlb*'s *PVWatts* inverter model to derive alternating-current power from the DC result.

The branch's outputs are stored as structured time series per suitable rooftop area: one Parquet file per area, holding timestamped results with the relevant geometric, solar-position, irradiance, and power variables — among them timestamp, centroid coordinates, slope, aspect, effective area, solar azimuth and elevation, merged horizon elevation, irradiance

components, cell temperature, and photovoltaic power – together with per-area annual summaries. The fast horizon-based approach thus keeps the framework’s artefact-oriented output logic even though it skips the heavier downstream clear-sky aggregation and scaling.

The significance of this branch is the balance it strikes between reuse and simplification: it reuses artefacts already produced by the main workflow – suitable rooftop areas, prior horizon-analysis results, prepared weather data – but bypasses the area-level clear-sky aggregation and batch-scaling chain of the workflow-based path. It is, in effect, a centroid-level, horizon-aware alternative within the same architecture, retaining directional obstruction, weather-driven irradiance adjustment, and physically interpretable computation while reducing dependence on the full downstream sequence.

4.6.4 Parallel execution model

The framework does not apply one uniform parallelisation scheme. Parallelisation is introduced selectively, stage by stage, according to each task’s computational characteristics – reflecting the pipeline’s heterogeneity, which combines GRASS raster computation, analytical artefact generation, and rooftop-level time-dependent simulation. **Table 4.4** summarises the main execution strategies adopted across the workflow, together with the computational units to which they are applied.

Stage	Parallelization mechanism	Parallel unit	Purpose
Horizon analysis	Multiprocessing within the GRASS environment	Subsets of azimuth directions	Reduce the cost of directional horizon computation over large raster surfaces
Clear-sky radiation modelling	Multiprocessing within the GRASS environment	Representative day and time-step combinations	Accelerate the generation of clear-sky irradiance rasters
Analytical artifact generation	Parallel data-processing operations	Point and timestamp transformation tasks	Speed up the conversion of GRASS outputs into structured analytical datasets
Real-sky batch simulation	Shared in-process simulator reuse with buffered result writing	Sequential timestep evaluation across rooftop areas	Improve batch efficiency without repeatedly constructing simulator objects
Co-simulation execution	Stepwise execution controlled by an external orchestrator	Single timestep payload	Support deterministic interaction with external simulation environments

Table 4.4. Parallel Execution Strategies Across the Simulation Pipeline

The first level of parallelisation is introduced within the GRASS-based geospatial stages. Horizon analysis is computationally demanding because the raster surface must be evaluated across multiple azimuth directions. For this reason, the horizon stage is parallelised within the GRASS environment over subsets of directional computations. A similar strategy is applied to the clear-sky radiation stage, where the *r.sun* model must be evaluated repeatedly across representative days and daylight timesteps. Parallel execution in these stages reduces the cost of large raster computations while preserving the methodological consistency of the geospatial workflow.

A second level applies during the transformation of GRASS outputs into analytical artefacts. Generating rooftop sampling datasets and time-indexed products can be expensive given the number of points and timestamps, so extraction and transformation are parallelised to speed the conversion into structured datasets for downstream simulation.

The downstream photovoltaic stage follows a different execution model. Rather than relying on the same type of task-level parallelisation used in the geospatial stages, it is organised around reusable photovoltaic simulation components. In the workflow-based simulation path, this component transforms weather-adjusted irradiance and meteorological variables into electrical power estimates while remaining independent of the control mechanism used to invoke it. During execution, the simulator receives effective irradiance values together with the relevant meteorological inputs, including air temperature and wind speed. These variables are used to estimate photovoltaic cell temperature through empirical temperature models provided by the *pvlb* library, including the SAPM temperature model and the PVsyst temperature model. In simplified form, cell temperature may be expressed as

$$T_{cell} = T_{air} + f(G, v) \quad (4.21)$$

where T_{air} is ambient air temperature, G is the irradiance incident on the module plane, and v is wind speed.

The estimated cell temperature is then used to compute photovoltaic power output through the *PVWatts* model. In simplified form, DC power may be written as

$$P_{dc} = P_{dc0} \cdot \frac{G}{G_{ref}} \cdot [1 + \gamma(T_{cell} - T_{ref})] \quad (4.22)$$

where P_{dc0} is nominal DC capacity, G is incident irradiance, G_{ref} is reference irradiance, γ is the temperature coefficient of power, and T_{cell} is the estimated cell temperature. Where required, the simulator can also estimate AC output by incorporating conversion losses and DC/AC sizing effects.

The photovoltaic simulator is a reusable object that stores each candidate area's static properties at initialisation – geometric characteristics such as tilt and orientation, and installation parameters such as effective area and nominal power density – avoiding repeated reconstruction of the same configuration during time-dependent evaluation.

The workflow-based simulator runs in two modes. In batch mode, it processes a complete weather time series over the whole horizon: the weather dataset is prepared once, simulator instances are created for the relevant areas, and the workflow iterates over the real timestamps, resolving each weather state, passing it to the appropriate instances, and writing results incrementally to partitioned storage. This improves efficiency through simulator reuse and buffered writing, but, despite the name, the batch runner is not a distributed execution layer.

In co-simulation mode, execution proceeds one step at a time under an external orchestrator. The aim is not batch throughput but deterministic interaction with another simulation environment: at each step the runner receives a weather payload for the current timestamp and passes it to the same simulator used in batch mode. Keeping a common physical core across both modes, while letting the surrounding control flow vary by deployment, is a deliberate design choice.

The fast horizon-based approach follows a lighter pattern. Because it bypasses the clear-sky area-means aggregation and weather-scaling chain of the workflow-based path, it does not use the same batch-oriented artefact traversal; execution is organised per selected rooftop area. For each area, the framework resolves the centroid, samples local horizon values, retrieves the corresponding PVGIS horizon profile, harmonises the GRASS-derived and PVGIS azimuth conventions, constructs the merged horizon description, and then evaluates a weather-driven *pvl* simulation over the prepared timestamp series. In this branch, the

main computational load lies not in repeated raster-scale processing, but in per-area horizon-aware time-series evaluation combined with persistent writing of one structured output file per rooftop unit.

One implementation detail is that the current codebase does not actively use a Ray-based distributed execution layer, although interface-level CPU-allocation parameters remain present; the retained batch-core code explicitly logs that the Ray CPU setting is ignored. In practice, runtime behaviour comes from the parallel strategies applied to GRASS-side processing, artefact generation, simulator reuse in batch evaluation, and per-area iteration in the fast horizon-based branch, rather than from Ray-based fan-out across rooftop areas.

Overall, the execution model is stage-specific and workload-aware: GRASS raster analysis is accelerated by internal parallelisation of directional and temporal computations, artefact generation by parallel transformation of large point and timestamp datasets, and workflow-based real-sky batch execution by simulator reuse and buffered writing. The fast horizon-based branch uses a lighter per-area strategy, while co-simulation keeps a stepwise pattern for deterministic coupling with external orchestrators.

4.6.5 Output products and formats

The framework produces structured output artefacts at several stages, each with a distinct role in the pipeline. Some preserve the results of expensive geospatial processing; others feed the real-sky simulation stage or are final outputs. A core requirement is that these products are materialised and stored for reuse rather than recomputed on demand. **Table 4.5** summarises the major artefacts generated by the system together with their logical storage paths, formats, and roles within the pipeline.

Artifact	Logical path / storage branch	Format	Purpose
Uploaded source datasets	uploads/<location>/<mapset>/...	Original uploaded format	Preserve validated source inputs used by the workflow
Rooftop static point dataset	points_clearsky/<location>/<mapset>/points_static.parquet	Parquet	Store static rooftop-point geometry and horizon attributes
Point-level clear-sky radiation dataset	points_clearsky/<location>/<mapset>/points_clearsky_timeseries.parquet	Parquet	Store direct and diffuse clear-sky irradiance values for each rooftop point and timestamp
Suitable rooftop surface dataset	suitable_areas/<location>/<mapset>/suitable_areas.parquet	Parquet	Represent candidate rooftop installation surfaces and their geometric properties
Rooftop-level clear-sky baseline dataset	area_clearsky_means/<location>/.../<mapset>/	Parquet	Store aggregated clear-sky radiation statistics at the rooftop-area level
Real-sky simulation results	real_sky_results/<location>/<mapset>/...	Partitioned Parquet	Store weather-adjusted photovoltaic production outputs for later retrieval and analysis
Fast horizon-based simulation results	fast_horizon_pv_results/<location>/<mapset>	Parquet	Store centroid-based, horizon-aware photovoltaic time-series outputs for selected suitable rooftop areas

Table 4.5. Artefacts, path organisation, format, and purpose across the simulation pipeline

The first group of artefacts originates from the GRASS-based preprocessing stages and the clear-sky radiation modelling stage. At this point in the workflow, the outputs of geospatial analysis are transformed into structured analytical results. These include the static rooftop-point dataset, which stores geometric attributes such as coordinates, slope, aspect, and directional horizon information in the original GRASS *r.horizon* convention, as well as the time-dependent point-level clear-sky radiation dataset, which stores direct and diffuse irradiance values for each rooftop point at each simulation timestep. Together, these two artefacts preserve the link between rooftop geometry and the temporal structure of clear-sky solar exposure. The horizon information stored in the static rooftop-point artefact should therefore be interpreted as a GRASS-derived directional profile rather than as a *pvl*ib/*PVGIS* azimuth profile. When these values are used by downstream *pvl*ib-based components, the directional convention must be harmonised as described in [Section 4.6.3](#).

A second major artefact is the dataset of rooftop surfaces deemed suitable for installation, generated either from user-supplied polygons or by automatic geometric detection. Besides identifiers and geometry, it stores descriptive information per candidate unit — representative slope and aspect, centroid coordinates, building association, and effective installation area — and constitutes the structured representation of installation units on which later aggregation and estimation are based.

Once suitable areas are identified, the workflow generates a rooftop-level clear-sky baseline by aggregating point-level irradiance to the scale of rooftop regions. This artefact holds the direct and diffuse clear-sky statistics for each candidate surface over time and is the principal radiation input to the workflow-based real-sky stage, marking the transition from point-level to rooftop-area-level representation – the scale at which installation and energy estimation are performed.

The final artefact groups are produced during the downstream simulation stages. In the workflow-based branch, the system generates weather-adjusted photovoltaic outputs for rooftop regions under real-sky conditions. Depending on the execution mode, these stored results may represent either a complete time series generated from an entire weather dataset or a sequence of outputs produced step by step during co-simulation. In addition to photovoltaic production estimates, these outputs may also include associated timestamps and simulation parameters. In the fast horizon-based branch, the framework generates a separate family of per-area photovoltaic time-series outputs, written as one Parquet file per selected rooftop unit. These outputs preserve centroid-based horizon-aware simulation results for the selected suitable areas, including the merged horizon condition used during *pvl*ib-based irradiance and photovoltaic calculation. Unlike the upstream static point horizon artefact, the merged horizon information used in this branch is expressed after harmonisation with the *pvl*ib/*PVGIS* azimuth convention. These outputs therefore constitute the final artefacts of the lighter alternative simulation path.

Parquet is the preferred format for analytical results throughout. Columnar storage reduces overhead through compression, supports fast access to large datasets, and lets later stages read only the columns a given operation needs – important when static geometry, time-dependent radiation, and rooftop-level results are accessed repeatedly. Artefacts queried mainly by simulation time may additionally be partitioned to improve access.

The output-artefact layer follows the same principles as the rest of the framework: it separates expensive geospatial computation from later simulation, preserves reusable intermediate results, and maintains structured products that can be traced, reloaded, and analysed independently of the original run. Artefact storage is therefore not merely persistence but a central part of the design – true of both the workflow-based path and the fast horizon-based branch, whose outputs are likewise materialised as explicit artefacts.

4.7 Implementation notes

4.7.1 API surface and endpoint conventions

The implemented API is organised into three main route groups, each corresponding to a distinct functional responsibility: *auth*, *users*, and *actinia*. Together, these groups separate authentication and session control, user management, and access to the geospatial and photovoltaic computation pipeline.

The *auth* route group contains the endpoints responsible for authentication and session lifecycle management. These include *POST /auth/login*, *POST /auth/refresh*, *POST /auth/logout*, and *GET /auth/me*. Their role is limited to token issuance, token renewal, log-out handling, and retrieval of the current authenticated user context. This design keeps session-related concerns isolated from the rest of the API surface.

The *users* route group is dedicated to user management. It includes self-service endpoints such as *GET /users/me*, *PATCH /users/me*, *GET /users/me/preferences*, and *PATCH /users/me/preferences*, which allow authenticated users to manage their own profile and preferences. It also includes administrative endpoints such as *GET /users*, *POST /users/{user_id}*, *GET /users/{user_id}*, and *DELETE /users/{user_id}* for broader user administration. These administrative operations are restricted to users holding *admin* or *superadmin* roles.

The *Actinia* route group provides the main interface to the computational workflow. Unlike the authentication and user-management routes, endpoints in this group are named after specific stages of the implemented methodology. The available routes include synthetic DSM generation (*POST /actinia/upload-and-synthetic-dsm*), real DSM upload (*POST /actinia/upload-real-dsm*), DSM smoothing and slope/aspect derivation (*POST /actinia/dsm-smooth-and-slope-aspect*), horizon analysis (*POST /actinia/horizon*), clear-sky solar-radiation modelling (*POST /actinia/solar-radiation*), generation of point-level static artefacts (*POST /actinia/points-static-parquet*), suitable-area detection (*POST /actinia/suitable-areas-parquet*), rooftop-level clear-sky aggregation (*POST /actinia/area-clearsky-means-parquet*), workflow-based real-sky batch simulation (*POST /actinia/real-sky-batch*), fast horizon-based simulation (*POST /actinia/fast-horizon-pv-simulation*), and weather artefact generation through *Open-Meteo* (*POST /actinia/weather/open-meteo*). This naming convention

makes the external API closely reflect the internal structure of the computational pipeline described in the methodology.

Another convention is that most computational endpoints use bearer-token authentication to resolve execution context from the validated token, so clients need not pass location or mapset identifiers for routine operations. The backend instead derives the execution namespace from the authenticated user and session, using the resolved user and session identifiers to determine the correct storage and processing scope for the requested stage.

More generally, the route design follows a descriptive naming scheme rather than a generic CRUD-oriented pattern. Endpoint names such as *upload-and-synthetic-dsm*, *upload-real-dsm*, *dsm-smooth-and-slope-aspect*, *area-clearsky-means-parquet*, and *fast-horizon-pv-simulation* encode the methodological purpose of the corresponding operation directly in the path. This improves readability in automatically generated interactive documentation and makes the API easier to use from external REST clients. In effect, the API functions not only as a service surface, but also as an orchestration interface whose route structure mirrors the main stages of the geospatial and photovoltaic workflow.

4.7.2 Error handling and logging

The backend handles errors in layers. At the API level, invalid requests become explicit HTTP responses; at the service level, structured logging captures operational context and execution details. Together these give predictable client behaviour and enough observability for debugging and monitoring.

At the API boundary, invalid requests are rejected with appropriate HTTP exceptions. Typical examples include invalid EPSG codes, missing input files, inconsistent slope or resolution parameters, invalid session states, or missing authenticated users. In such cases, the routers raise *HTTPException* instances with specific status codes, including *400 Bad Request*, *401 Unauthorized*, *404 Not Found*, and *422 Unprocessable Entity*. Alongside the status code, the response includes a detail field that identifies the error condition and provides an explanatory message.

A second layer of error normalisation is applied inside the service functions. At this level, lower-level failures such as database write errors, invalid ZIP packages, missing CRS information, file-not-found conditions, invalid weather inputs, missing rooftop areas, or failed

GRASS command execution are translated into *HTTPException* responses with consistent status semantics. Depending on the case, the response body may include structured fields such as *error*, *message*, *details*, and *path*, allowing service-level failures to be exposed through a uniform HTTP interface.

The implementation also distinguishes between expected methodological failures and unexpected execution failures. Expected failures include invalid credentials, missing session state, unsupported or incomplete geospatial uploads, and weather-data that do not conform to the required schema. These conditions are represented as explicit HTTP errors with meaningful status codes. Unexpected failures are typically caught at service boundaries, logged together with contextual information, and then re-raised as internal server or dependency-related errors depending on whether the source lies within the local backend or in an external dependency such as Actinia or *Open-Meteo*. This preserves consistent client-facing behaviour while retaining diagnostic information on the backend side.

Logging is emitted primarily at the service and utility layers rather than at the schema or model layer. The codebase defines module-level loggers through the standard *logging.getLogger(__name__)* pattern, and these loggers are used to report both workflow progress and failure conditions. Informational log messages are typically emitted at major stage boundaries, including synthetic DSM generation, smoothing and terrain-derivative computation, *r.sun* execution, *Open-Meteo* retrieval, and real-sky batch processing. These log messages may include contextual identifiers such as user ID, session ID, mapset name, file name, or stage name in order to support tracing of request execution.

Error-level logging is used when failures occur in internal service components, including database persistence, GRASS command execution, and interaction with external services. Typical examples include failed job-row insertion, unsuccessful *r.sun* execution, unreadable weather datasets, or exceptions raised during batch processing. These failures are logged before being propagated as HTTP responses. Warning-level logging is used for situations in which execution continues but some aspect of the workflow is degraded or unexpected, such as fallback behaviour, ignored compatibility parameters, or partially inconsistent intermediate inputs. This approach provides detailed operational visibility without exposing unnecessary implementation detail to API clients.

Logging also occurs at multiple levels of abstraction. Utility functions for path resolution or command execution inside the Actinia container emit low-level runtime information, while higher-level services record workflow progression in stage terms, so execution can be traced both as technical operations and as workflow stages.

Overall, the backend pairs normalised HTTP error behaviour with service-level logging, supporting inspection of execution failures, external-dependency issues, and intermediate stages during development and debugging while presenting a consistent interface to clients.

4.7.3 Configuration management

The implemented framework includes an explicit configuration layer that separates deployment-specific settings from the core application logic. This is achieved through a combination of a `.env` file and a typed configuration mechanism based on the *pydantic-settings* framework.

At startup, environment variables are loaded from the project-level `.env` file into a central settings object. This defines the configuration for system operation — Actinia connection details, storage roots, container identifiers, JWT settings, database connection parameters, and development and Valkey options — so environment-specific assumptions are not embedded in the service logic.

One important group of settings concerns the geospatial runtime environment used to execute the GRASS-based processing stages. These settings include the base URL of the Actinia service together with backend credentials used for authenticated interaction. They also define the logical storage roots used to persist data both on the host filesystem and inside the container runtime. In the current implementation, *FILE_STORAGE_ROOT* identifies the host-side user-data directory, while *USERDATA_CONTAINER_ROOT* identifies the corresponding path inside the containerised runtime environment. This distinction is essential because uploaded files and intermediate artefacts must be visible both to the backend on the host side and to the GRASS software stack executing inside the container.

The configuration layer also defines the container identifiers needed for command execution: the Actinia and Valkey container names are exposed as environment variables rather than hard-coded, as are the database connection settings. The same backend code can

therefore be moved to different deployment topologies by changing configuration values rather than the implementation.

Security-related parameters are managed in the same way. These include the secret key, signing algorithm, access-token lifetime, and refresh-token lifetime. As a result, authentication behaviour can be adjusted without changing the underlying code. Similarly, operational parameters such as *DEBUG* and *LOG_LEVEL* are defined in the configuration layer, allowing the same codebase to run with different verbosity and diagnostic behaviour depending on the target environment. Valkey-related connectivity settings are also defined centrally within this layer.

Configuration also handles filesystem paths. The module resolves host-side paths relative to the project root, converting storage locations to absolute paths before use elsewhere, which reduces dependence on the current working directory and keeps runtime behaviour stable across environments. This matters most for storage roots, which must stay consistent for file persistence, artefact retrieval, and interaction with the containerised geospatial runtime.

Overall, the configuration design supports both reproducibility and deployment portability: external services, storage locations, authentication parameters, and runtime options are declared explicitly and loaded centrally, so the same pipeline can be reproduced across development and deployment environments while keeping the system's operational assumptions transparent.

To complement the implementation notes presented in this section, Appendix A provides two implementation-oriented UML-style diagrams documenting the internal backend organisation of the proposed framework. The first diagram summarises the core backend service structure, including API modules, authentication and user services, grouped solar services, ORM models, repositories, core utilities, and external execution or persistence systems. The second diagram expands the internal organisation of the *app.services.solar* package by showing the terrain, radiation, artefact, weather, and photovoltaic service groups and their main dependency flows. These diagrams are not part of the main methodological workflow itself but provide supporting documentation for the implemented service structure.

4.8 Summary

This chapter has presented the methodology of the proposed rooftop photovoltaic simulation framework, from its architecture to the implementation-oriented organisation of its geospatial and simulation stages. It first described the overall architecture – the separation of access control, orchestration, geospatial processing, and simulation – and then the end-to-end pipeline through which heterogeneous spatial inputs become simulation-ready artefacts and, ultimately, rooftop-level production estimates.

A central feature is the explicit separation between geospatial computation in GRASS and the data-oriented analytical and simulation stages that follow. For rooftop geometry, the framework supports both a high-resolution DSM-based approach and a derived-surface approach, so the methodology applies both to rooftops given directly through suitable DSM inputs and to rooftops reconstructed from building footprints and elevation-derived heights.

The chapter also showed that GRASS outputs are not consumed directly by the final simulation stage but transformed into structured analytical artefacts – rooftop sampling points, suitable-area datasets, and rooftop-level clear-sky baselines. This separation of expensive geospatial analysis from later weather-aware estimation enables efficient reuse of intermediate results and evaluation of multiple weather scenarios without repeating the upstream workflow.

Building on this baseline, the workflow-based real-sky stage incorporates external meteorological data, aligns real timestamps with precomputed clear-sky reference states, and estimates effective irradiance under actual conditions, passing these values to a reusable simulation component that supports both batch and co-simulation. The chapter also introduced a fast horizon-based alternative that operates on selected suitable areas, combines locally derived horizon information with external PVGIS data, harmonises the azimuth conventions before merging, and performs direct pvlib-based estimation at centroid level – so the methodology offers both a complete workflow-based path and a lighter horizon-aware alternative in one architecture.

Finally, the chapter summarised the implementation layers supporting the methodology: authentication and session management, metadata and job tracking, validation, API design, error handling, configuration management, and stage-specific execution models. This

structure matters because the framework is defined not only by its computational formulas but by the execution logic, artefact-persistence strategy, and service organisation that make the workflow reproducible, traceable, and deployable across environments.

Overall, the methodology described in this chapter extends beyond the computational model alone. It also defines the execution logic, data structures, and system interfaces required to make the framework reproducible, traceable, and operationally extensible. In this sense, the chapter establishes the full methodological and implementation foundation for the system evaluated in the following chapters, while additional implementation-oriented UML diagrams and selected code listings are provided in **Appendix A** as supporting technical documentation.

Results and Discussion

5.1 Introduction

This chapter presents and discusses the experimental results obtained by applying the proposed simulation infrastructure to four real PV systems located in Turin, Italy, and evaluates its accuracy through systematic comparison with measured generation data. While the previous chapters described the architecture of the framework, the geospatial preprocessing pipeline, and the photovoltaic estimation methodology, the focus here shifts to the empirical validation of those methods under real-sky conditions and over two independent monitoring periods.

The validation is built around four case studies that together cover a wide range of system scales, installation types, and rooftop orientations, spanning a nominal power range from 13.20 kWp to 631 kWp and two monitoring periods separated by a decade. This diversity makes it possible to examine not only whether the simulation infrastructure produces accurate results for a single system, but whether its accuracy is consistent across fundamentally different installation configurations and meteorological years.

The chapter addresses several interconnected objectives. It first identifies the most appropriate irradiance decomposition model for the Turin urban area, then quantifies the simulation accuracy achieved at multiple time resolutions and across different seasonal and weather conditions and finally compares two irradiance sources of different spatial and temporal characteristics. Beyond energy yield estimation, the chapter also demonstrates how the same simulation infrastructure can be applied to the detection of hardware malfunctions in deployed PV systems, extending its practical relevance beyond planning and assessment contexts.

The results are presented progressively, from the case study description and the definition of the statistical indicators, through the model selection and system-level validation results,

to a consolidated cross-system comparison and a broader discussion of the findings and their implications.

5.2 Case Study and Experimental Setup

To validate the simulation capabilities of the proposed infrastructure, four PV systems located in Turin, Italy, were selected as the case study. Turin is a city in the Piedmont region in north-west Italy, characterised by a temperate continental climate with well-defined seasonal variations, frequent winter overcast periods, and a significant proportion of cloudy and rainy days throughout the year, making it a challenging and representative environment for PV simulation at the urban scale.

All four systems are situated within the La Crocetta district, a densely built residential and institutional area covering approximately 3.7 km² in the city centre and containing around 2,200 buildings of heterogeneous construction type and period. The geographic location of all four systems within La Crocetta is shown in **Figure 5.1**. The geospatial data required for simulation were provided by the Turin city council. The DSM has a spatial resolution of 1 m², sufficient to resolve rooftop obstructions such as chimneys, dormers, and HVAC equipment that affect both the irradiance received by the PV modules and the accuracy of shadow simulations. Cadastral building footprint maps were also provided, together forming the spatial foundation applied uniformly to all four systems in the simulation pipeline.

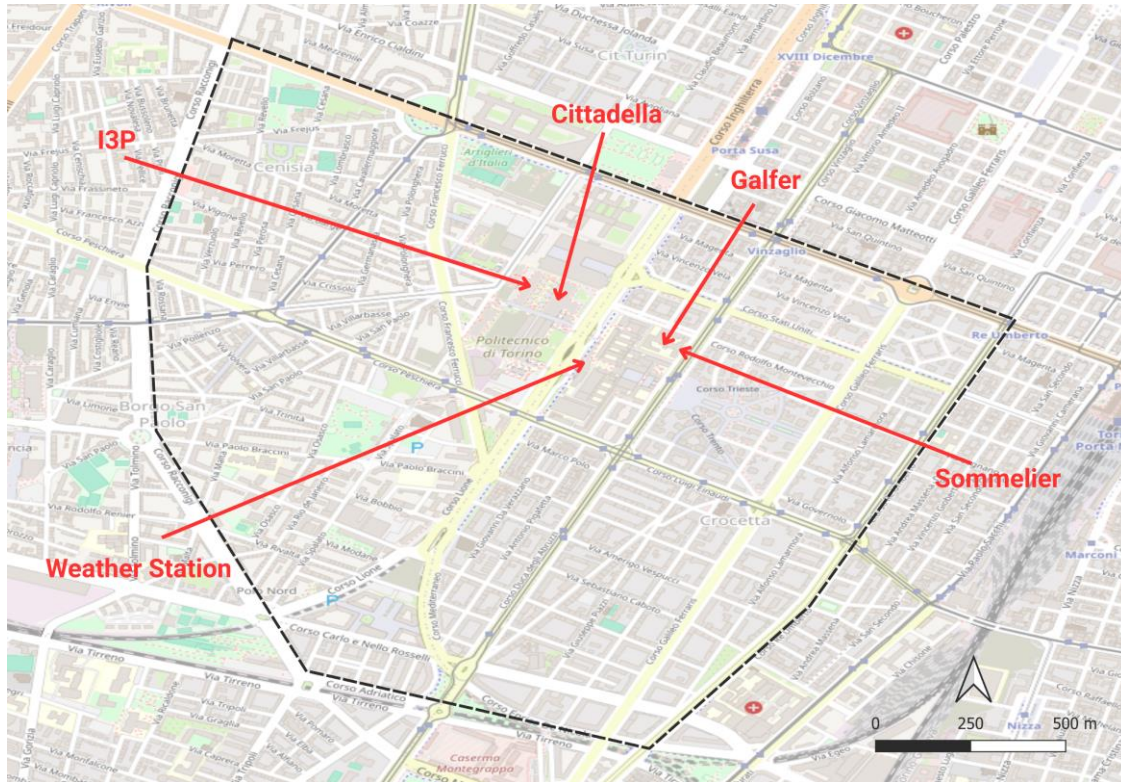


Figure 5.1. Geographic locations of the four PV systems and Polito Weather Station, La Crocetta, Turin.

Two independent irradiance sources were used throughout this study, enabling a systematic assessment of the effect of weather data quality on simulation accuracy. The first source is the Polito Weather Station, located on the Politecnico di Torino campus at the centre of the La Crocetta district. The station is equipped with a first-class pyranometer measuring GHI at one-minute intervals, subsequently averaged to 15-minute resolution. Quality filtering was applied following Younes et al. [37] to exclude samples in which the solar altitude was below 7° , the clearness index fell outside the physically valid range of 0 to 1, or the measured GHI exceeded the theoretical clear-sky value. For the 2010-2011 monitoring period covering Sommelier and GalFer, the Polito station provided GHI measurements only, without concurrent DNI or DHI values. For these two systems, the direct and diffuse irradiance components therefore had to be estimated from GHI using irradiance decomposition models, as described in **Section 5.4**. For the 2020 monitoring period covering I3P and Cittadella, the Polito station data files additionally include a pre-computed Model 1 Closure simulation providing DNI and DHI directly, enabling all five decomposition model variants to be evaluated for those systems. The second source is Open-Meteo, an open-access reanalysis API providing modelled GHI, DNI, and DHI at hourly resolution on a grid-

ded spatial domain. Since Open-Meteo supplies all three irradiance components for all systems and monitoring periods, it allows the full set of five model variants to be evaluated consistently across the entire study. Whether modelled DNI and DHI from a reanalysis product translates into better simulation accuracy than GHI-only decomposition is one of the central questions addressed in this chapter.

The four PV systems span a wide range of scales, installation types, orientations, and monitoring periods, as summarised in **Table 5.1** Sommelier (Istituto Sommelier) and GalFer (Istituto Galileo Ferraris) are two poly-crystalline systems installed in 2004 on school buildings within the La Crocetta district, both oriented SSW at 240° azimuth. Sommelier is building-integrated at a slope of 20°, while GalFer is free-standing at 35°. Both systems were monitored from 1 March 2010 to 22 February 2011 at 15-minute resolution, which analysed the energetic performance of PV and solar thermal plants. The I3P system (Ex Fucine building) is installed on a dual-pitch roof with two independent arrays on opposite faces of the building: an East-facing array and a West-facing array, each monitored throughout the full calendar year 2020. The contrasting orientations of these two arrays produce fundamentally different generation profiles, with the East array peaking in the morning and the West array peaking in the afternoon, making I3P a unique case study for validating the infrastructure's capability with non-south-facing and geometrically complex multi-array installations. It should be noted that the West array experienced a sensor outage from approximately 14 June 2020, eliminating most of the summer period coverage; West array results for Summer 2020 are therefore treated as indicative only and are clearly flagged in the relevant analyses. Finally, Cittadella is the large-scale rooftop PV installation on the main building of the Politecnico di Torino university campus, comprising 1,836 mono-crystalline modules with a total nominal power of 631 kWp distributed across approximately 3,000 m² of rooftop area, also monitored throughout 2020. At nearly 50 times the nominal power of I3P and 32 times that of Sommelier, Cittadella enables a direct assessment of the infrastructure's cross-scale generalisability from small residential-scale arrays to large commercial and institutional installations.

The two monitoring periods, covering 2010 to 2011 for Sommelier and GalFer and the full year 2020 for I3P and Cittadella, are a decade apart. This temporal separation is not a limitation but an asset: consistent results across two independent monitoring campaigns and

two distinct meteorological years strengthen the generalisability of the conclusions drawn from the study.

Parameter	I3P (East & West)	GalFer	Sommelier	Cittadella
Nominal Power [kW]	15.28	13.2	19.8	631
Module Power [W]	283	165	165	327
PV Modules Number	54	80	120	1836
Module Technology	Mono-cryst.	Poly-cryst.	Poly-cryst.	Mono-cryst.
Module Efficiency [%]	20.2	13.1	13.1	20.10%
Module Temp. Coef. [%/°C]	0.38	0.48	0.48	0.33
Slope [°]	26	35	20	25
Aspect [°] (South 270°)	23	240	240	180
Installation Type	Dual-pitch rooftop	Free-standing	Building-integrated	Rooftop
Installation Year	2008	2004	2004	2016
Monitoring Period	Jan–Dec 2020	Mar 2010–Feb 2011	Mar 2010–Feb 2011	Jan–Dec 2020
Data Resolution	15 min	15 min	15 min	15 min

Table 5.1. Technical characteristics of the four PV systems used for validation

5.3 Performance Indicators

To evaluate the accuracy of the simulations, the statistical performance indicators proposed by Gueymard [38] are adopted throughout this chapter. These indicators are applied consistently across all four systems, both irradiance sources, and all-time resolutions, allowing direct and meaningful comparisons between results. All indicators except for the coefficient of determination are expressed as a percentage of the mean measured value, rather than in absolute units, as recommended in [38].

Four indicators quantify the dispersion of simulation errors. The MBD measures the average systematic deviation of simulated values from measured values, where a positive MBD indicates a tendency to overestimate and a negative MBD indicates underestimation. The RMSD represents the standard deviation of the differences between simulated and observed values and is sensitive to large individual errors. The MAD is defined as the average of the absolute differences between simulated and measured values and provides a measure of the typical error magnitude without sign cancellation. The coefficient of determina-

tion (r^2) indicates the proportion of the variance in the measured data that is explained by the simulation, ranging from 0 to 1, where values close to 1 indicate a strong linear relationship between simulated and measured outputs.

Two additional indicators characterise the overall model performance. The WIA [50] is a standardised measure of the degree of model prediction error, varying between 0 and 1, where 1 corresponds to perfect agreement between simulation and measurement. The LCE [51] is defined as unity minus the ratio of the mean square error to the variance in the observed data. LCE can range from negative infinity to 1, where 1 represents a perfect model and values below zero indicate that the simulation performs worse than a naive predictor based on the observed mean alone.

For yearly aggregates, LCE, WIA, and r^2 are not computed, since these indicators require a time series of values rather than a single scalar, and are therefore not applicable to a single annual total.

Weather conditions are classified throughout this chapter using the clearness index k_t , defined as the ratio of measured GHI to the theoretical extraterrestrial horizontal irradiance at the same location and time. Three categories are adopted: Sunny days ($k_t \geq 0.65$), Cloudy days ($0.30 \leq k_t < 0.65$), and Rainy days ($k_t < 0.30$). This classification is applied consistently across all systems and monitoring periods.

5.4 Decomposition Model Selection

The accuracy of irradiance decomposition models is known to depend strongly on geographic location, local climate, and the characteristics of the irradiance data source [38]. For this reason, all available model variants were evaluated for each PV system and each irradiance source before selecting the model to be applied in the simulation validation reported in **Section 5.5**. The evaluation was performed at the daily timeframe over the full monitoring period of each system. Two categories of model are compared: **Model 1 Closure**, which uses DNI and DHI values supplied directly by the data source without decomposition, and **four Model 2** decomposition variants, namely DISC, DIRINT, DIRINDEX, and ERBS, which estimate DNI and DHI from GHI measurements alone. Model 1 Closure is not applicable for Sommelier and GalFer under the Polito source, since the 2010–2011 Polito station data provides GHI measurements only, without concurrent DNI or DHI values.

Table 5.2 reports the performance of all five model variants for the I3P East array with the Polito Weather Station, which is used here as the representative case to illustrate the selection rationale. The same evaluation was carried out for all remaining systems and sources.

Model	LCE	MAD [%]	MBD [%]	r ²	RMSD [%]	WIA
Model 1 – Closure (GHI and computed DNI/DHI)						
Model 1 – Closure	-0.617	18.92	17.58	0.759	24.06	0.953
Model 2 – Decomposition models (GHI only)						
Model 2 – DISC	0.622	8	3.01	0.944	11.64	0.987
Model 2 – DIRINT	0.623	8.01	2.89	0.944	11.62	0.987
Model 2 – DIRINDEX	0.613	8.04	3.43	0.942	11.77	0.987
Model 2 – ERBS (★)	0.681	7.54	-1.61	0.952	10.69	0.989

Table 5.2. I3P East – daily model selection, 2020 (Polito Weather Station)

The results in **Table 5.2** show a clear hierarchy. Among the four decomposition variants, **ERBS** achieves the best overall performance, with the lowest RMSD (10.69%), the lowest MAD (7.54%), the highest LCE (0.681), and the highest WIA (0.989). The differences between ERBS and the other three decomposition variants are modest in absolute terms, since DISC, DIRINT, and DIRINDEX all achieve similar indicator values. However, ERBS consistently produces the smallest systematic bias, with MBD of -1.61% compared to values between 2.89% and 3.43% for the other variants, indicating a better ability to avoid systematic overestimation of the irradiance components. Model 1 Closure performs substantially worse than all decomposition variants, with an RMSD of 24.06% against 10.69% for ERBS, and a strongly positive MBD of 17.58%, confirming that the DNI and DHI values derived through the closure computation introduce a systematic positive bias that decomposition from measured GHI avoids.

This pattern is consistent across all four systems and both irradiance sources, as summarised in **Table 5.3**, which reports the ERBS and Model 1 Closure results for all systems. **ERBS** is therefore adopted as the selected decomposition model for all systems and all irradiance sources throughout the remainder of this chapter. This uniform selection ensures full cross-system comparability of results and is supported by the consistent superiority of ERBS observed in every evaluated case.

System	Source	Model	LCE	MAD [%]	MBD [%]	r ²	RMSD [%]	WIA
Sommelier	Polito	Closure	—	—	—	—	—	—
		ERBS	0.41	36.27	36.12	0.406	44.41	0.899
	Open-Meteo	Closure	-1.27	75.41	73.8	0.81	86.46	0.73
		ERBS	-0.82	65.05	63.16	0.8	77.32	0.76
GalFer	Polito	Closure	—	—	—	—	—	—
		ERBS	0.93	8.92	2.29	0.94	14.68	0.984
	Open-Meteo	Closure	0.63	26.75	19.63	0.78	34.93	0.912
		ERBS	0.78	19.72	5.75	0.8	27	0.941
I3P East	Polito	Closure	-0.617	18.92	17.58	0.759	24.06	0.953
		ERBS	0.681	7.54	-1.61	0.952	10.69	0.989
	Open-Meteo	Closure	-0.64	17.7	12.31	0.755	24.22	0.944
		ERBS	-0.232	16.41	-1.61	0.816	20.99	0.952
I3P West	Polito	Closure	-2.771	45.17	-7.85	0.353	60.05	0.74
		Model 2 – ERBS	-3.182	48.28	-22.29	0.282	63.24	0.684
	Open-Meteo	Closure	-3.239	46.81	-19.25	0.272	63.6	0.666
		ERBS	-3.585	49.03	-22.54	0.213	66.15	0.635
Cittadella	Polito	Closure	-28.602	11.74	0	0.901	16.08	0.974
		ERBS	-9.247	6.68	0	0.966	9.46	0.991
	Open-Meteo	Closure	-52.652	17.29	0	0.82	21.65	0.942
		ERBS	-51.052	16.7	0	0.825	21.32	0.947

Table 5.3. Daily model selection summary – all systems and irradiance sources.

The cross-system summary in **Table 5.3** indicates that ERBS provides better or comparable performance to Model 1 Closure across all applicable cases. The margin of improvement is particularly pronounced for Cittadella, where Model 1 Closure yields LCE values of -28.6 and -52.7 under the Polito and Open-Meteo sources respectively, suggesting that directly supplied DNI and DHI values from both sources introduce severe systematic errors for this system. For Sommelier and GalFer under Open-Meteo, Model 1 Closure is also substantially worse than ERBS, with RMSD values of 86.46% and 34.93% respectively against 77.32% and 27.00% for ERBS. The only case where Model 1 Closure and ERBS produce comparable results is I3P East, where the RMSD difference is less than 4 percentage points under both sources. However, even here ERBS achieves better bias control, making it the more consistent choice. For the I3P West array, all models produce poor results due to the sensor outage and geometry-related factors discussed in **Section 5.2**, and this case is therefore not used as decisive evidence for model selection. ERBS is adopted uniformly across all systems and sources to ensure cross-system comparability of results.

Two additional observations emerge from **Table 5.3**. First, the I3P West array results are poor under all models and both sources, with RMSD values exceeding 60% in every case. As discussed in **Section 5.2**, this is attributable to the sensor outage affecting the West array from mid-June 2020 and does not reflect a limitation of the decomposition model. Second, a consistent pattern is visible across all systems: Polito-based simulations with ERBS achieve substantially lower error values than Open-Meteo-based simulations with ERBS, even though the same decomposition model is applied in both cases. This finding, which points to the dominant role of irradiance input quality rather than decomposition model choice in governing simulation accuracy, is examined in detail in **Section 5.6**.

The visual confirmation of the model selection results is provided in **Figure 5.2** and **Figure 5.3**, which report the daily energy production for all model variants compared against measured values for Sommelier throughout July 2010, for the Polito and Open-Meteo sources respectively. Both figures clearly show that ERBS produces the closest agreement with measured daily energy across the full month. Model 1 Closure systematically overestimates daily production in both cases, particularly on high-irradiance days, which is consistent with the strongly positive MBD values reported in **Table 5.3**. Among the Model 2 decomposition variants, DISC, DIRINT, and DIRINDEX also tend to overestimate production, while ERBS achieves the most balanced distribution of errors across the month. This supports the selection of ERBS as the most appropriate decomposition model for the Turin urban area under both irradiance sources.

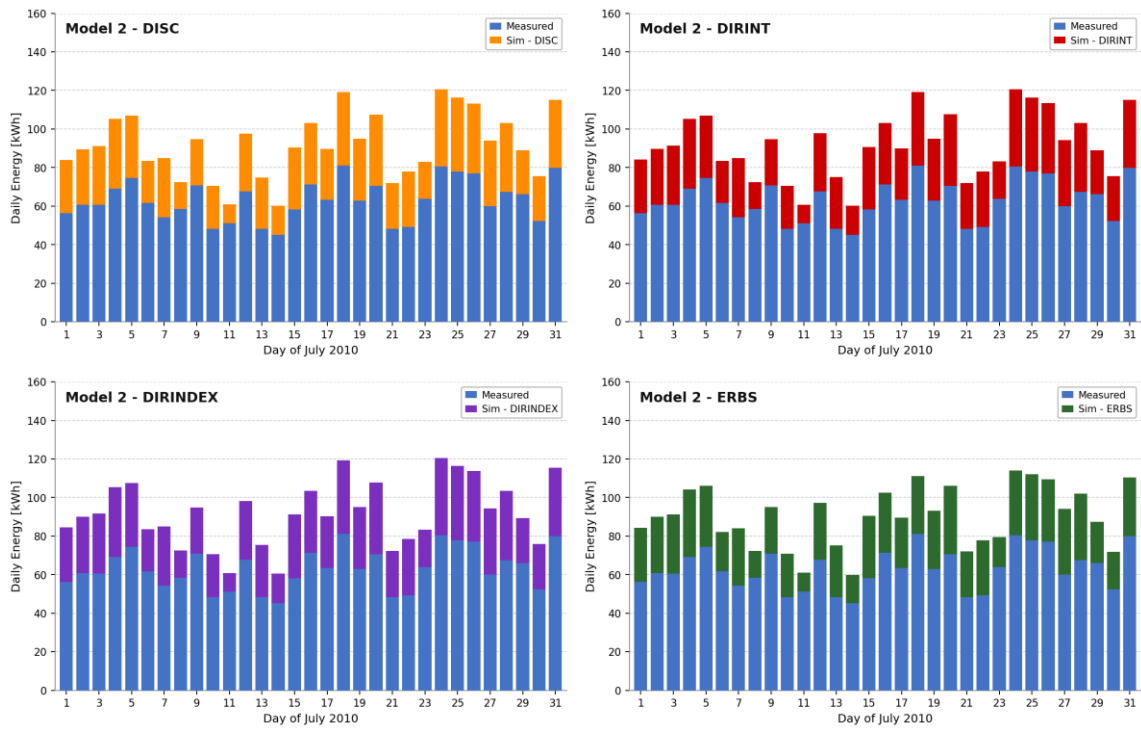


Figure 5.2. Sommelier – daily energy comparison, July 2010, Model 2 variants vs. measured (Polito Weather Station).

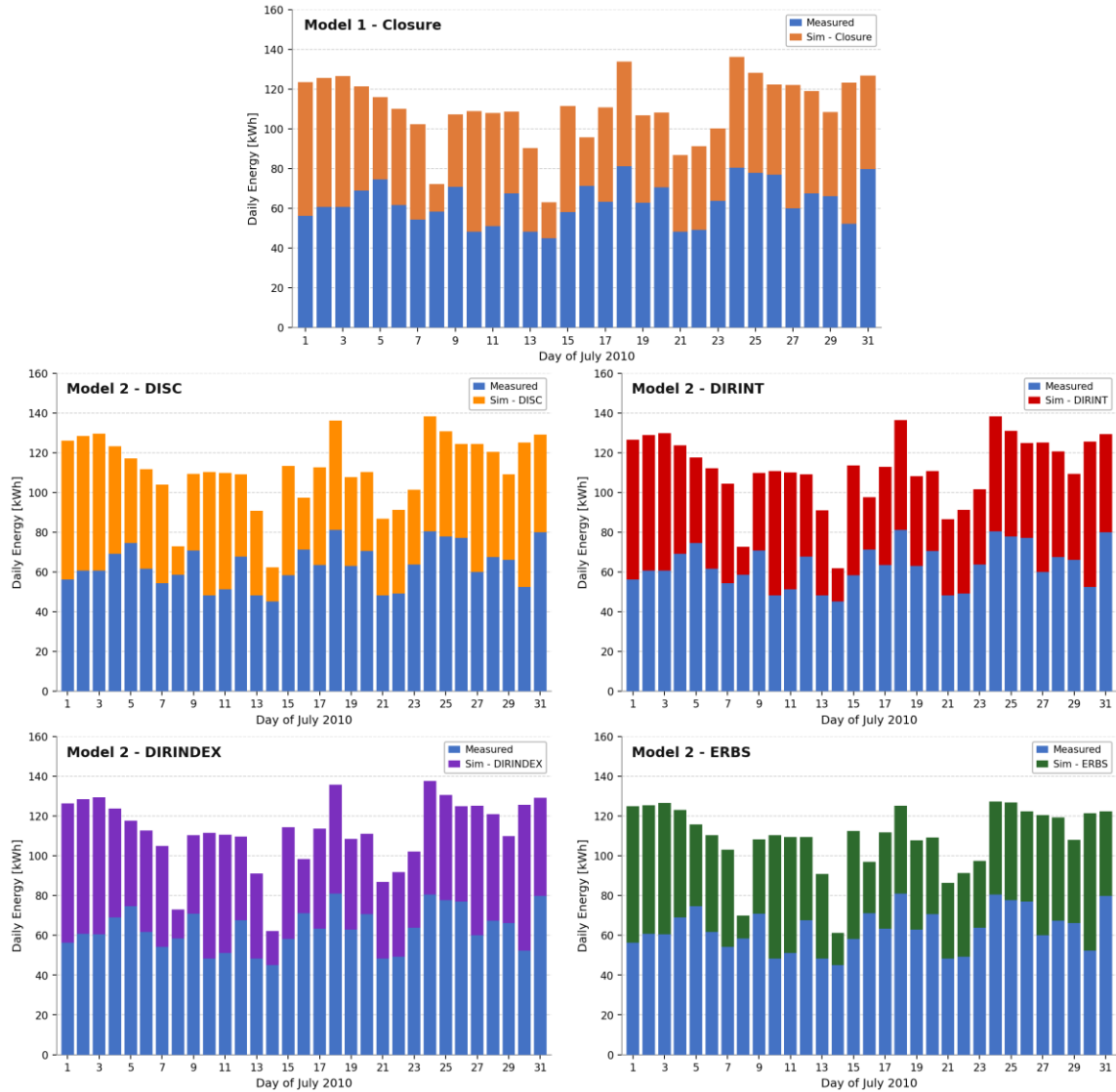


Figure 5.3. Sommelier – daily energy comparison, July 2010, Model 1 & 2 variants vs. measured (Open-Meteo).

5.5 PV System Simulation Validation

This section presents the simulation validation results for each of the four PV systems, using the ERBS decomposition model selected in [Section 5.4](#). For each system, the simulation output is compared against measured generation data at multiple time resolutions, and the behaviour of the simulation under both the Polito and Open-Meteo irradiance sources is assessed. The presentation follows a progression from the two 2010-2011 systems, Sommelier and GalFer, to the two 2020 systems, I3P and Cittadella, which benefit from a more complete dataset and a full seasonal analysis.

5.5.1 Sommelier

The Sommelier PV system was simulated over the full monitoring period from 1 March 2010 to 22 February 2011. The performance indicators for both irradiance sources at 15-minute, daily, and yearly timeframes are reported in **Table 5.4** and **Table 5.5** for the Polito and Open-Meteo sources respectively.

Source	Time-frame	LCE	MAD [%]	MBD [%]	r^2	RMSD [%]	WIA
Polito – ERBS	15 minutes	0.48	45.18	34.84	0.484	64.42	0.903
	Daily	0.41	36.27	36.12	0.406	44.41	0.899
	Yearly	—	37.69	37.69	—	37.69	—

Table 5.4. Sommelier – ERBS simulation performance indicators, Polito Weather Station, March 2010 to February 2011. Daily time-frame results highlighted in bold.

Source	Time-frame	LCE	MAD [%]	MBD [%]	r^2	RMSD [%]	WIA
Polito – ERBS	15 minutes	-0.08	69.98	61.76	0.71	94.53	0.83
	Daily	-0.82	65.05	63.16	0.8	77.32	0.76
	Yearly	—	63.16	63.16	—	63.16	—

Table 5.5. Sommelier – ERBS simulation performance indicators, Open-Meteo, March 2010 to February 2011. Daily time-frame results highlighted in bold.

The results in **Table 5.4** and **Table 5.5** reveal a consistent pattern of overestimation across both irradiance sources and all timeframes, as indicated by the strongly positive MBD values. For the Polito source, the daily MBD reaches 36.12%, while for Open-Meteo it reaches 63.16%. This systematic positive bias is the dominant feature of the Sommelier results and is attributable primarily to the building-integrated nature of the installation. Building-integrated PV systems typically experience higher thermal losses than free-standing or rooftop systems, since the rear face of the modules is insulated by the building structure and cannot dissipate heat through natural convection. This increases the operating cell temperature and reduces the module efficiency relative to the values assumed by the simulation, which models thermal behaviour using standard NOCT parameters. The effect is amplified in summer, when ambient temperatures are highest and irradiance is greatest, leading to a systematic gap between simulated and measured generation that cannot be corrected through decomposition model selection alone.

The comparison between the two irradiance sources is also notable. The Polito source achieves substantially better accuracy than Open-Meteo at all timeframes: at the daily level, RMSD is 44.41% for Polito against 77.32% for Open-Meteo, and MAD is 36.27% against 65.05%. The difference in MBD between the two sources, 36.12% for Polito and 63.16% for Open-Meteo, suggests that Open-Meteo introduces an additional layer of systematic overestimation on top of the building-integration thermal effect, likely due to the coarser spatial and temporal resolution of the reanalysis product and its inability to capture local meteorological conditions in the La Crocetta district with the same fidelity as the on-site pyranometer.

The instantaneous generation profiles for three representative days, namely sunny, cloudy, and rainy, are shown in **Figure 5.4**. The figure is consistent with the overestimation pattern observed in the statistical indicators. On the sunny day, both simulations substantially overestimate the measured peak power, with Open-Meteo producing an even larger excess than Polito. On the cloudy day, the simulation tracks the general shape of the measured profile but consistently overestimates the instantaneous power. On the rainy day, both sources overestimate the measured output, with Open-Meteo again showing the larger divergence. Across all three weather conditions, the Polito simulation follows the measured profile more closely than Open-Meteo, further supporting the superiority of the local irradiance source for this system.

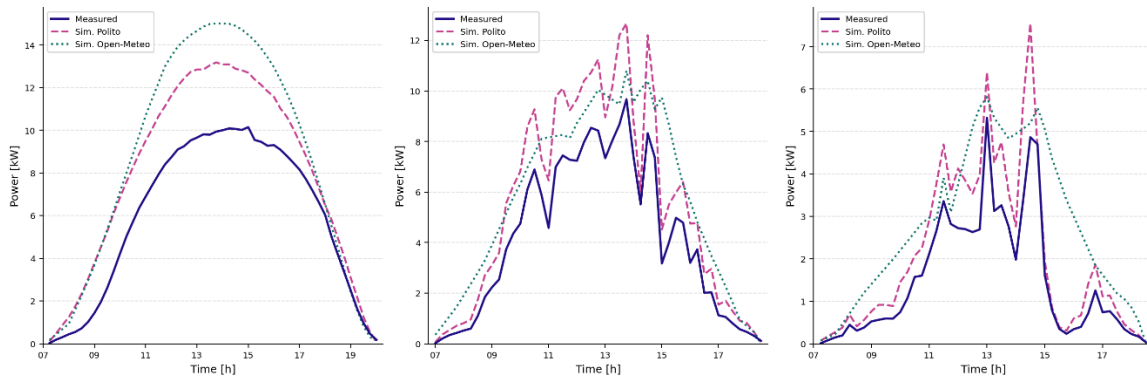


Figure 5.4. Sommelier – simulated vs. measured instantaneous power profiles by weather type.

5.5.2 Galfer

The GalFer PV system was simulated over the same monitoring period as Sommelier, from 1 March 2010 to 22 February 2011. As discussed in **Section 5.2**, a hardware malfunction

was identified in the GalFer system affecting its performance from 15 May 2010 onwards. The results are therefore reported both for the full monitoring period and separately for the periods before and after 15 May 2010, following the approach adopted by Bottaccioli et al. [49]. The performance indicators for both irradiance sources are reported in **Table 5.6** and **Table 5.7**.

Source	Time-frame	LCE	MAD [%]	MBD [%]	Source	Time-frame	LCE
Polito – ERBS	15 min (All)	0.78	26.66	1.26	0.804	41.89	0.946
	Daily (All)	0.93	8.92	2.29	0.938	14.68	0.984
	15 min (Before)	0.74	29.57	-3.81	0.76	47.42	0.933
	Daily (Before)	0.97	6.79	-3.29	0.974	8.98	0.992
	15 min (After)	0.8	25.78	2.8	0.82	39.95	0.95
	Daily (After)	0.93	9.56	3.97	0.932	16.17	0.981

Table 5.6. GalFer – ERBS performance indicators, Polito Weather Station, 2010–2011 (pre/post malfunction)

Source	Time-frame	LCE	MAD [%]	MBD [%]	Source	Time-frame	LCE
Polito – ERBS	15 min (All)	0.58	41.74	4.54	0.629	57.91	0.889
	Daily (All)	0.78	19.72	5.75	0.796	27	0.941
	15 min (Before)	0.6	41.76	3.09	0.63	58.6	0.89
	Daily (Before)	0.58	41.73	4.98	0.629	57.67	0.889
	15 min (After)	0.79	17.88	3.67	0.798	23.46	0.942
	Daily (After)	0.77	20.27	6.37	0.794	28.07	0.941

Table 5.7. GalFer – ERBS performance indicators, Open-Meteo, 2010–2011 (pre/post malfunction)

The results in **Table 5.6** reveal a clear and important contrast between the two monitoring sub-periods under the Polito source. Before the malfunction, the simulation achieves excellent accuracy: daily RMSD is 8.98%, daily MAD is 6.79%, LCE reaches 0.97, and WIA reaches 0.992. These are among the best results observed across the entire study and confirm that the infrastructure produces highly accurate simulations for a free-standing poly-crystalline system under normal operating conditions. After 15 May 2010, accuracy degrades noticeably: daily RMSD increases to 16.17% and MAD to 9.56%, while MBD shifts from -3.29% to 3.97%, indicating a reversal from slight underestimation to systematic overestimation. This degradation is consistent with a hardware malfunction that reduces the actual generation of the system below what the simulation, which models the system as fully operational, would predict.

The behaviour under Open-Meteo is strikingly different and reveals an important limitation of the reanalysis source. As reported in [Table 5.7](#), the before and after periods show comparable accuracy under Open-Meteo, with daily RMSD of 57.67% before and 28.07% after the malfunction. The before period is in fact substantially worse than the after period under Open-Meteo, which is the opposite of what is observed with the Polito source. This means that Open-Meteo introduces its own systematic errors that are large enough to mask the malfunction signal entirely. The instantaneous generation profiles for representative days before and after the malfunction, together with a detailed analysis of the divergence pattern, are presented in [Section 5.7](#).

5.5.3 I3P

The I3P system comprises two independently monitored arrays installed on opposite faces of a dual-pitch rooftop: an East-facing array and a West-facing array. Both arrays were simulated over the full calendar year 2020. As noted in [Section 5.2](#), the West array experienced a sensor outage from approximately 14 June 2020, eliminating most of the summer monitoring period. West array results are therefore reported for completeness but treated as indicative only throughout this section, and summer results are flagged accordingly.

East Array

The seasonal and weather-condition performance indicators for the I3P East array are reported in [Table 5.8](#), which combines results for both the Polito and Open-Meteo sources at the 15-minute timeframe, together with full-year daily and 15-minute aggregates.

Source	Season	Weather	LCE	MAD [%]	MBD [%]	r ²	RMSD [%]	WIA
Polito – ERBS	Winter	Sunny	0.849	24.99	-5.03	0.8	39.12	0.94
	Winter	Cloudy	0.58	111.76	100.58	0.009	174.12	0.764
	Winter	Rainy	0.28	626.66	626.66	-6.305	829.31	0.372
	Spring	Sunny	0.947	10.72	-1.06	0.951	16.59	0.987
	Spring	Cloudy	0.958	12.86	4.55	0.95	21.72	0.987
	Spring	Rainy	0.988	17.94	12.7	0.956	22.65	0.988
	Summer	Sunny	0.924	10.73	4.82	0.93	20.73	0.982
	Summer	Cloudy	0.949	12.68	4.2	0.949	21.68	0.987
	Summer	Rainy	0.995	6.91	4.49	0.989	9.33	0.997
	Autumn	Sunny	0.942	15.01	-11.32	0.936	19.24	0.982
	Autumn	Cloudy	0.947	15.52	-0.10	0.934	29.19	0.981
	Autumn	Rainy	0.991	18.67	17.87	0.95	20.9	0.987
	Full year	15-min	0.932	14.28	0	0.933	23.98	0.983
	Full year	Daily	0.681	7.54	-1.61	0.952	10.69	0.989
Open-Meteo – ERBS	Winter	Sunny	0.747	39.85	-6.67	0.664	54.7	0.895
	Winter	Cloudy	0.619	163.24	140.29	-0.280	205.41	0.722
	Winter	Rainy	—	—	—	—	—	—
	Spring	Sunny	0.796	23.03	-1.03	0.807	34.89	0.949
	Spring	Cloudy	0.468	54.99	9.71	0.37	85.59	0.773
	Spring	Rainy	0.714	77.1	20.24	0.184	110.58	0.675
	Summer	Sunny	0.745	26.19	4.47	0.766	39.69	0.937
	Summer	Cloudy	0.381	59.85	12.85	0.363	85.54	0.785
	Summer	Rainy	—	—	—	—	—	—
	Autumn	Sunny	0.793	26.25	-13.15	0.772	37.38	0.936
	Autumn	Cloudy	0.58	67	23.12	0.43	98.13	0.781
	Autumn	Rainy	0.778	88.17	49.87	-0.417	111.97	0.607
	Full year	15-min	0.759	30.85	0	0.763	45.93	0.934
	Full year	Daily	-0.232	16.41	-1.61	0.816	20.99	0.952

Table 5.8. I3P East – seasonal performance indicators by weather type, 2020
(15-min, Polito & Open-Meteo).

The results for the I3P East array under the Polito source represent the strongest validation outcome across the entire study. The full-year daily RMSD reaches 10.69% with a MAD of 7.54% and a near-zero MBD of -1.61%, indicating that the simulation reproduces the measured generation with high fidelity and negligible systematic bias. The WIA of 0.989 and r² of

0.952 at the daily level confirm an exceptionally strong agreement between simulated and measured output

s. At the 15-minute resolution, the full-year RMSD increases to 23.98%, which is consistent with the expected degradation at finer time scales due to short-term irradiance variability that neither the decomposition model nor the simulation can fully resolve at sub-hourly resolution.

The seasonal breakdown reveals important patterns. Spring and Summer under the Polito source achieve the best accuracy across all weather conditions, with 15-minute RMSD values ranging from 16.59% to 22.65% in Spring and from 9.33% to 21.68% in Summer. Autumn results are also strong, with RMSD values between 19.24% and 29.19%. Winter performance is markedly different. While Winter Sunny days achieve acceptable accuracy (RMSD 39.12%, LCE 0.849), Winter Cloudy and Winter Rainy days produce extreme percentage error values (RMSD 174.12% and 829.31% respectively). These values, however, must be interpreted with caution. In winter, the absolute power output of the East array is very low, particularly on cloudy and rainy days. Under these conditions, small absolute errors in the simulation translate into very large percentage errors relative to the near-zero measured mean, inflating the percentage-based indicators far beyond what the absolute accuracy of the simulation would suggest. This is a known limitation of percentage-based performance metrics when applied to low-power conditions [38].

Under the Open-Meteo source, the East array results show a consistent and substantial degradation relative to Polito across all seasons and weather conditions. The full-year daily RMSD increases from 10.69% to 20.99%, and the 15-minute RMSD increases from 23.98% to 45.93%. The seasonal breakdown reveals that the degradation is most severe under cloudy and rainy conditions: Spring Cloudy RMSD rises from 21.72% to 85.59%, Autumn Rainy RMSD rises from 20.90% to 111.97%, and Spring Rainy RMSD reaches 110.58% under Open-Meteo compared to 22.65% under Polito. These results suggest that the reanalysis product struggles to accurately represent the highly variable irradiance conditions associated with overcast and precipitating skies, while the local pyranometer captures these conditions with much greater fidelity. Missing data for Winter Rainy and Summer Rainy under Open-Meteo further limit its coverage under the most challenging conditions.

The instantaneous generation profiles for representative sunny, cloudy, and rainy days are shown in **Figure 5.5**. On the sunny day, the Polito-ERBS simulation tracks the measured profile almost perfectly across the entire day, while both Open-Meteo variants substantially overestimate the output, with Model 1 Closure producing the largest excess particularly in the morning hours. On the cloudy day, Polito-ERBS follows the measured profile shape closely with a modest underestimation, while Open-Meteo-ERBS overestimates significantly and Open-Meteo-Closure overestimates dramatically around midday. On the rainy day, Polito-ERBS underestimates in the morning but tracks the afternoon generation reasonably, while both Open-Meteo variants overestimate throughout. The figure provides clear visual confirmation of both the superiority of the Polito source over Open-Meteo and the superiority of ERBS decomposition over Model 1 Closure across all weather conditions.

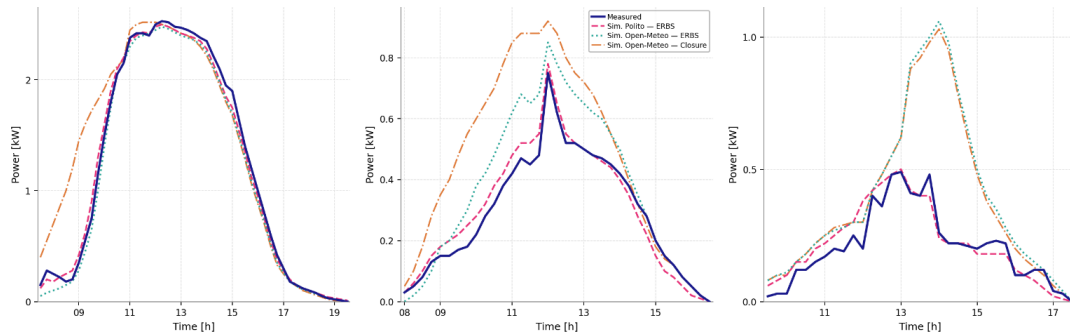


Figure 5.5. I3P East — simulated vs. measured instantaneous power profiles by weather type.

West Array

The performance indicators for the I3P West array are reported in **Table 5.9**, which provides a full seasonal and weather-condition breakdown for both the Polito and Open-Meteo sources, together with full-year aggregates at 15-minute, daily, and yearly timeframes. Summer results are based on limited data following the sensor outage from 14 June 2020 and are treated as indicative only throughout this section.

Source	Season	Weather	LCE	MAD [%]	MBD [%]	r ²	RMSD [%]	WIA
Polito – ERBS	Winter	Sunny	0.801	49.45	-5.14	0.536	66.69	0.814
	Winter	Cloudy	0.854	90.54	10.69	0.435	111.44	0.758
	Winter	Rainy	0.832	319.91	230.47	-0.753	406.14	0.503
	Spring	Sunny	0.34	47.64	-41.97	0.296	66.67	0.717
	Spring	Cloudy	0.586	48.44	-48.03	0.436	68.38	0.785
	Spring	Rainy	0.865	50.24	-48.38	0.473	76.89	0.778
	Summer	Sunny	-0.582	265.8	190.36	-0.700	353.73	0.478
	Summer	Cloudy	0.414	120.33	27.92	0.152	165.81	0.648
	Summer	Rainy	0.728	72.95	-29.81	0.376	98.77	0.699
	Autumn	Sunny	0.488	83.23	15.56	0.119	117.42	0.64
	Autumn	Cloudy	0.278	202.95	112	-1.262	288.16	0.416
	Autumn	Rainy	0.902	48.32	-46.74	0.428	67.69	0.773
	Full year	15-min	0.425	81.74	-0.05	0.287	120.84	0.681
	Full year	Daily	-3.182	48.28	-22.29	0.282	63.24	0.684
Open-Meteo – ERBS	Winter	Sunny	0.779	53.36	-8.24	0.48	74.09	0.765
	Winter	Cloudy	0.831	102.9	7.38	0.35	138.92	0.662
	Winter	Rainy	—	—	—	—	—	—
	Spring	Sunny	0.27	51.52	-44.54	0.208	73.07	0.671
	Spring	Cloudy	0.427	57.62	-47.98	0.199	88.3	0.649
	Spring	Rainy	0.638	71.18	-53.52	0.01	123.43	0.457
	Summer	Sunny	-0.450	264.35	183.65	-0.638	338.83	0.459
	Summer	Cloudy	0.418	102.55	4.22	0.173	139.44	0.57
	Summer	Rainy	—	—	—	—	—	—
	Autumn	Sunny	0.478	91.61	17.66	0.106	125.7	0.597
	Autumn	Cloudy	0.55	130.24	41	-0.431	176.1	0.428
	Autumn	Rainy	0.838	64.93	-40.20	-0.038	96.25	0.509
	Full year	15-min	0.378	87.56	-0.04	0.228	126.93	0.618
	Full year	Daily	-3.585	49.03	-22.54	0.213	66.15	0.635

Table 5.9. I3P West — seasonal performance indicators by weather type, 2020
(15-min, Polito & Open-Meteo).

The West array results are substantially less accurate than those of the East array across all seasons and conditions under both sources. Under Polito, the full-year daily RMSD reaches 63.24% with a daily MAD of 48.28%, compared to 10.69% and 7.54% respectively for the East array. The dominant feature of the West array results is the large and consistent negative MBD observed in Spring across all weather conditions, reaching values between -41.97% and -48.38% under Polito and between -44.54% and -53.52% under Open-Meteo.

This systematic underestimation in Spring suggests that the simulation overestimates the effective irradiance losses on the west-facing surface during this season, likely related to the geometric complexity of the dual-pitch rooftop and limitations in the DSM representation of the horizon profile as seen from the West array. A similar pattern of negative MBD is observed in Autumn Rainy conditions under both sources, reaching -46.74% under Polito and -40.20% under Open-Meteo. The instantaneous generation profiles for representative sunny, cloudy, and rainy days are shown in **Figure 5.6**, which illustrates the systematic underestimation pattern observed across all three weather conditions under the Polito source, and the contrasting behaviour of the Open-Meteo variants.

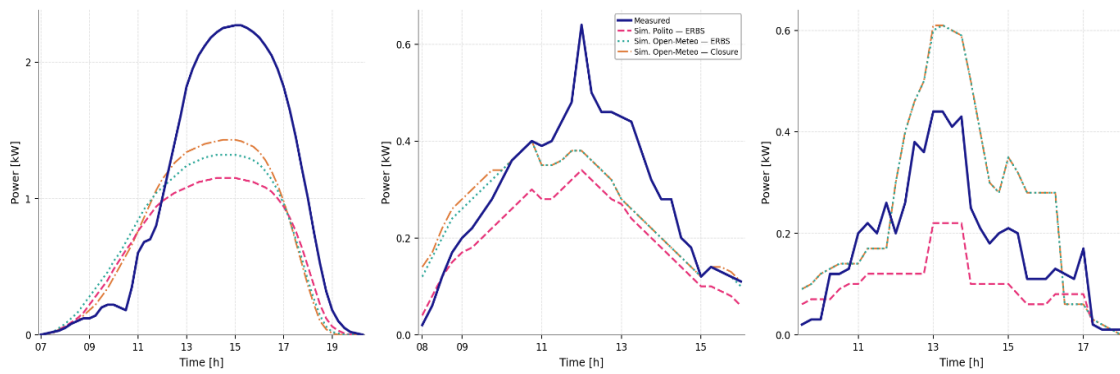


Figure 5.6. I3P West — simulated vs. measured instantaneous power profiles by weather type.

Winter Rainy conditions produce extreme percentage error values under both sources, with RMSD reaching 406.14% under Polito. As discussed for the East array, these extreme values reflect the very low absolute power output of the West array on winter rainy days rather than a fundamental failure of the simulation. Summer results under both sources are dominated by the sensor outage effect and cannot be considered representative. The Summer Sunny RMSD of 353.73% under Polito and 338.83% under Open-Meteo are direct consequences of the severely limited and unrepresentative dataset available for that condition.

Comparing the two irradiance sources for the West array, the differences are smaller than those observed for the East array, though Polito still achieves marginally better full-year accuracy (daily RMSD 63.24% vs 66.15%). This reduced gap between sources for the West array, compared to the pronounced gap for the East array, is consistent with the interpretation that the West array accuracy is dominated by geometric and orientation-related factors that neither source can resolve, rather than by the quality of the irradiance input itself. Under

these conditions, the advantage of the local pyranometer over the reanalysis product becomes less decisive.

It is worth noting that despite the poor absolute accuracy across individual seasons and conditions, the full-year MBD values are near zero for both sources at the 15-minute level (-0.05% for Polito and -0.04% for Open-Meteo), indicating that overestimation and underestimation errors partially cancel across the year. However, the full-year daily MBD of -22.29% under Polito and -22.54% under Open-Meteo reveals a net systematic underestimation at the daily aggregation level, suggesting that the cancellation is incomplete and that the West array simulation tends to underpredict daily energy production when aggregated over the full monitoring period.

5.5.4 Cittadella

The Cittadella installation is the largest system in the study, with a nominal power of 631 kWp distributed across 1,836 mono-crystalline modules on the main rooftop of the Politecnico di Torino university campus. It was simulated over the full calendar year 2020. The seasonal and weather-condition performance indicators for both irradiance sources are reported in **Table 5.10**.

Source	Season	Weather	LCE	MAD [%]	MBD [%]	r ²	RMSD [%]	WIA
Polito – ERBS	Winter	Sunny	-1.832	15.75	10.46	0.914	21.98	0.979
	Winter	Cloudy	-0.319	12	7.66	0.921	24.17	0.983
	Winter	Rainy	0.551	14.57	3.14	0.907	26.35	0.975
	Spring	Sunny	-3.893	16.68	-3.13	0.89	23.07	0.97
	Spring	Cloudy	-1.050	12.18	-3.89	0.943	22.84	0.985
	Spring	Rainy	0.651	10.99	-9.75	0.966	17.23	0.991
	Summer	Sunny	-4.467	18.65	-0.47	0.876	27.34	0.966
	Summer	Cloudy	-1.603	14.54	-1.43	0.93	24.88	0.982
	Summer	Rainy	0.494	9.93	-9.26	0.975	13.75	0.993
	Autumn	Sunny	-1.446	12.95	-0.11	0.939	18.32	0.983
	Autumn	Cloudy	-0.649	11.25	-5.11	0.953	23.48	0.988
	Autumn	Rainy	0.748	10.72	-9.44	0.965	14.78	0.991
	Full year	15-min	-2.41	15.79	0.1	0.918	24.92	0.978
	Full year	Daily	-9.247	6.68	0	0.966	9.46	0.991
Open-Meteo – ERBS	Winter	Sunny	-6.536	25.4	2.39	0.77	36.05	0.937
	Winter	Cloudy	-32.566	86.87	63.42	-1.017	126.17	0.692
	Winter	Rainy	-3.707	66.25	34.92	0.033	89.31	0.736
	Spring	Sunny	-7.842	22.51	-9.57	0.801	31.09	0.943
	Spring	Cloudy	-17.321	47.61	11.54	0.492	69.15	0.851
	Spring	Rainy	-10.610	73.44	27.73	-0.125	103.52	0.668
	Summer	Sunny	-9.562	27.52	-2.38	0.761	38.31	0.928
	Summer	Cloudy	-23.782	56.54	23.7	0.333	78.72	0.803
	Summer	Rainy	-17.325	61.83	27.41	0.088	84.98	0.784
	Autumn	Sunny	-5.311	20.88	-5.08	0.842	29.61	0.954
	Autumn	Cloudy	-24.256	61.19	11.06	0.283	93.42	0.749
	Autumn	Rainy	-30.164	119.65	108.07	-3.265	172.81	0.524
	Full year	15-min	-8.982	29.77	-0.01	0.76	43.09	0.929
	Full year	Daily	-51.052	16.7	0	0.825	21.32	0.947

Table 5.10. Cittadella – seasonal performance indicators by weather type, 2020
(15-min, Polito & Open-Meteo).

The Cittadella results under the Polito source represent the strongest full-year accuracy achieved across the entire study. The full-year daily RMSD reaches 9.46% with a MAD of 6.68% and a MBD of exactly 0%, indicating that the simulation reproduces the annual energy yield of the system with no net systematic bias and with a typical daily error of less than

10%. The WIA of 0.991 and r^2 of 0.966 at the daily level confirm an exceptionally strong agreement between simulated and measured outputs. At the 15-minute resolution, the full-year RMSD increases to 24.92%, again consistent with the pattern observed for I3P East and reflecting the inherent limitations of sub-hourly irradiance decomposition rather than any fundamental weakness of the simulation methodology.

The seasonal breakdown under Polito reveals a broadly consistent accuracy across all four seasons and all three weather conditions, with RMSD values ranging from 13.75% (Summer Rainy) to 27.34% (Summer Sunny) at the 15-minute level. This consistency is notable given the scale and complexity of the installation. Unlike Sommelier, where building integration introduced a strong positive bias, and unlike the I3P West array, where geometric orientation effects produced large seasonal variations, Cittadella shows a balanced pattern of small positive and negative MBD values across conditions, none exceeding 10.46% in absolute terms. The near-zero full-year MBD suggests that these seasonal biases cancel almost completely over the annual cycle. It is worth noting that despite the strong RMSD and WIA values, the LCE indicator takes strongly negative values for most seasonal conditions under Polito, reaching as low as -4.467 for Summer Sunny. Negative LCE values indicate that, for those subsets, the mean squared simulation error remains larger than the variance of the observed series. This behaviour reflects the statistical properties of the LCE formulation when applied to short or highly variable subsets and does not imply poor absolute accuracy. LCE should therefore be interpreted together with RMSD, MAD, r^2 , and WIA, particularly when percentage-normalised metrics and subsets of different variability are being compared. The strong values of RMSD, MAD, and WIA reported in **Table 5.10** are the more reliable indicators of simulation quality in this context [50], [51].

Under the Open-Meteo source, the Cittadella results show a pattern of degradation consistent with that observed for I3P East, but more pronounced in magnitude. The full-year daily RMSD increases from 9.46% to 21.32%, and the 15-minute RMSD increases from 24.92% to 43.09%. The seasonal breakdown reveals that the degradation is most severe under cloudy and rainy conditions: Winter Cloudy RMSD rises from 24.17% to 126.17%, Autumn Rainy RMSD rises from 14.78% to 172.81%, and Spring Rainy RMSD reaches 103.52% under Open-Meteo compared to 17.23% under Polito. Sunny day conditions are also substantially degraded under Open-Meteo, with Summer Sunny RMSD rising from 27.34% to 38.31% and Winter Sunny from 21.98% to 36.05%. The full-year daily MBD remains at 0%

under Open-Meteo as well, indicating that while the reanalysis product introduces much larger instantaneous and daily errors, it does not introduce a net annual bias for this system.

The instantaneous generation profiles for representative sunny, cloudy, and rainy days are shown in **Figure 5.7**. On the sunny day, Polito-ERBS tracks the measured profile closely throughout the day, while Open-Meteo-ERBS and Open-Meteo-Closure both underestimate the output, with Closure producing the largest deficit in the afternoon hours. On the cloudy day, which shows a complex measured profile with sharp drops due to passing cloud cover, Polito-ERBS follows the overall shape of the measured curve reasonably well, capturing both the morning ramp and the afternoon variability, while both Open-Meteo variants substantially overestimate the output across the entire day. On the rainy day, Polito-ERBS slightly underestimates the measured generation, while Open-Meteo variants again overestimate significantly. The figure is consistent with the substantially higher accuracy of the Polito source for Cittadella across all weather conditions and indicates that the advantage of the local pyranometer is particularly pronounced under the variable irradiance conditions of cloudy days, where the coarser spatial and temporal resolution of the reanalysis product cannot capture the rapid irradiance fluctuations recorded by the on-site sensor.

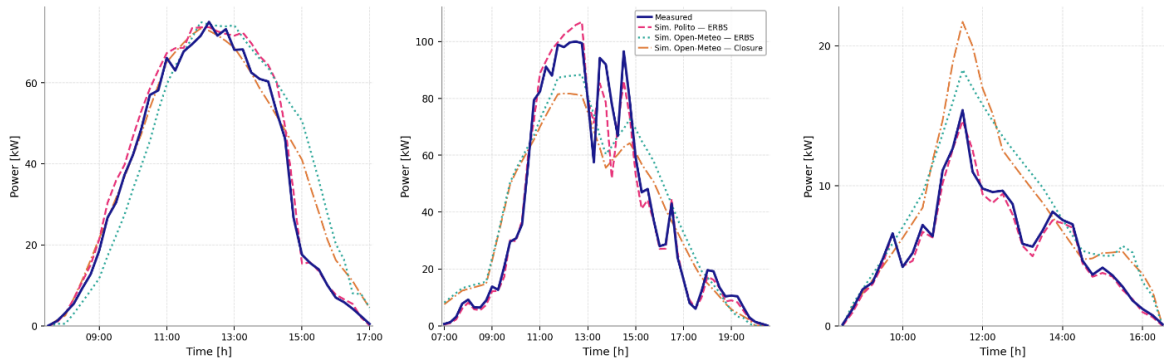


Figure 5.7. Cittadella – simulated vs. measured instantaneous power profiles by weather type.

The Cittadella results also confirm the cross-scale generalisability of the simulation infrastructure. Despite the nominal power of 631 kWp being approximately 32 times larger than Sommelier, 48 times larger than I3P, and representing a fundamentally different installation category in terms of module count, rooftop area, and system complexity, the simulation achieves comparable or superior accuracy to the smaller systems when driven by the local Polito irradiance source. This scale-invariance is an important validation outcome, demon-

strating that the infrastructure performs consistently from small residential-scale arrays to large commercial and institutional installations without requiring system-specific calibration or parameter adjustment.

5.6 Weather Source Comparison: Polito vs Open-Meteo

The results presented in [Section 5.5](#) consistently indicate that the choice of irradiance source has a substantial effect on simulation accuracy, with the local Polito Weather Station outperforming the Open-Meteo reanalysis product across all systems and monitoring periods. This section consolidates the cross-system comparison and examines the magnitude and nature of the differences between the two sources.

The full-year daily performance indicators for both sources across all five validated arrays are summarised in [Table 5.11](#).

System	Weather Source	LCE	MAD [%]	MBD [%]	r ²	RMSD [%]
<i>Daily Time-Frame – Full Year (Sommelier & GalFer: Mar 2010–Feb 2011; I3P & Cittadella: Jan–Dec 2020)</i>						
Sommelier	Polito – ERBS	0.41	36.27	36.12	0.406	44.41
	Open-Meteo – ERBS	–0.82	65.05	63.16	0.8	77.32
GalFer	Polito – ERBS	0.93	8.92	2.29	0.938	14.68
	Open-Meteo – ERBS	0.78	19.72	5.75	0.796	27
I3P East	Polito – ERBS	0.681	7.54	–1.61	0.952	10.69
	Open-Meteo – ERBS	–0.232	16.41	–1.61	0.816	20.99
I3P West	Polito – ERBS	–3.182	48.28	–22.29	0.282	63.24
	Open-Meteo – ERBS	–3.585	49.03	–22.54	0.213	66.15
Cittadella	Polito – ERBS	–9.247	6.68	0	0.966	9.46
	Open-Meteo – ERBS	–51.052	16.7	0	0.825	21.32

Table 5.11. Cross-system ERBS daily performance – Polito vs. Open-Meteo, all four PV systems.

The table indicates that the Polito source achieves consistently lower RMSD and MAD values than Open-Meteo across all systems. However, the magnitude of the improvement varies considerably depending on the system and its characteristics. For GalFer, I3P East, and Cittadella, the Polito source achieves approximately half the RMSD of Open-Meteo, with RMSD ratios of 1.84, 1.96, and 2.25 respectively. For Sommelier, the improvement is also substantial, with RMSD decreasing from 77.32% to 44.41%, though the absolute values re-

main high for both sources due to the building-integration thermal effect discussed in **Section 5.5.1**. For the I3P West array, the difference between sources is minimal (RMSD 63.24% vs 66.15%), suggesting that the accuracy of this array is governed primarily by geometric and orientation-related factors rather than irradiance input quality, as discussed in **Section 5.5.3**.

The MBD values reveal an important distinction between two categories of system behaviour. For Sommelier and GalFer, the MBD differs substantially between sources: Sommelier shows MBD of 36.12% under Polito and 63.16% under Open-Meteo, while GalFer shows 2.29% under Polito and 5.75% under Open-Meteo. This indicates that Open-Meteo introduces an additional positive bias on top of any bias already present in the Polito-driven simulation, suggesting that the reanalysis product systematically overestimates the irradiance reaching these systems during the 2010-2011 monitoring period. For I3P East and Cittadella, by contrast, the MBD is identical between the two sources (-1.61% and 0% respectively), indicating that the bias is determined entirely by the decomposition model and simulation parameters rather than by the irradiance source. In these cases, Open-Meteo degrades accuracy through increased random error (higher RMSD and MAD) without introducing additional systematic bias.

The seasonal analysis presented in **Section 5.5** provides further insight into the mechanisms underlying the source difference. The degradation under Open-Meteo is consistently most severe under cloudy and rainy conditions, where the reanalysis product fails to capture the rapid and localised irradiance fluctuations that the on-site pyranometer records in real time. Under sunny conditions, Open-Meteo achieves much better relative performance, with RMSD values typically 1.5 to 2 times higher than Polito rather than the 3 to 6 times higher values observed under cloudy and rainy conditions. This pattern is consistent across I3P East and Cittadella, which are the two systems for which a full seasonal breakdown is available under both sources.

These findings have important practical implications. For applications requiring accurate sub-daily or daily energy yield estimates, such as grid management, energy trading, or performance monitoring, the use of a local high-quality irradiance measurement is strongly preferable to a gridded reanalysis product. Open-Meteo, despite its convenience and global coverage, introduces errors that are particularly large under the overcast and variable sky

conditions that characterise a significant portion of the annual irradiance budget in Turin. For applications requiring only annual energy yield estimates, where errors tend to cancel over long periods, the MBD values at the yearly level are close to zero for both sources on most systems, suggesting that Open-Meteo may be an acceptable substitute when local measurements are unavailable and annual precision is sufficient.

5.7 PV System Operation Assessment

Beyond its primary role in energy yield estimation and planning, the simulation infrastructure can serve as a diagnostic tool for assessing the operational condition of deployed PV systems. This application is demonstrated through the GalFer case study, in which a hardware malfunction affecting the system from 15 May 2010 was identified through systematic comparison of simulated and measured generation profiles. The analysis presented here follows the approach introduced by Bottaccioli et al. [49], extending it to include the comparison between two irradiance sources and their respective ability to expose the malfunction signal.

As reported in **Section 5.5.2**, the GalFer simulation under the Polito source achieves excellent accuracy before the malfunction date, with a daily RMSD of 8.98% and a daily MAD of 6.79% for the period from 1 March 2010 to 14 May 2010. After 15 May 2010, the accuracy degrades noticeably: daily RMSD increases to 16.17%, MAD rises to 9.56%, and MBD shifts from -3.29% to +3.97%, indicating a reversal from slight underestimation to systematic overestimation. Since the simulation models the GalFer system as fully operational throughout the entire monitoring period using fixed nominal parameters, any systematic divergence between the simulated and measured output that emerges at a specific point in time cannot be attributed to the simulation model itself. Instead, it signals a change in the physical behaviour of the real system, consistent with a hardware malfunction that reduces the actual generation capacity below the nominal value assumed by the simulation.

The instantaneous generation profiles shown in **Figure 5.8** provide direct visual evidence of this divergence. Before 15 May 2010, the Polito-ERBS simulation tracks the measured output closely across all three weather conditions, namely sunny, cloudy, and rainy, with only minor deviations that are consistent with normal modelling uncertainty. After 15 May 2010, a clear and systematic gap opens between the simulated and measured profiles, particularly on sunny days where the reduced peak power of the degraded system becomes most

visible. The measured peak power on a representative sunny day after the malfunction is substantially lower than on a comparable sunny day before the malfunction, while the simulation predicts essentially the same peak power for both days since it uses unchanged system parameters. This divergence pattern is a characteristic signature of a hardware malfunction that reduces system efficiency without completely disabling the installation.

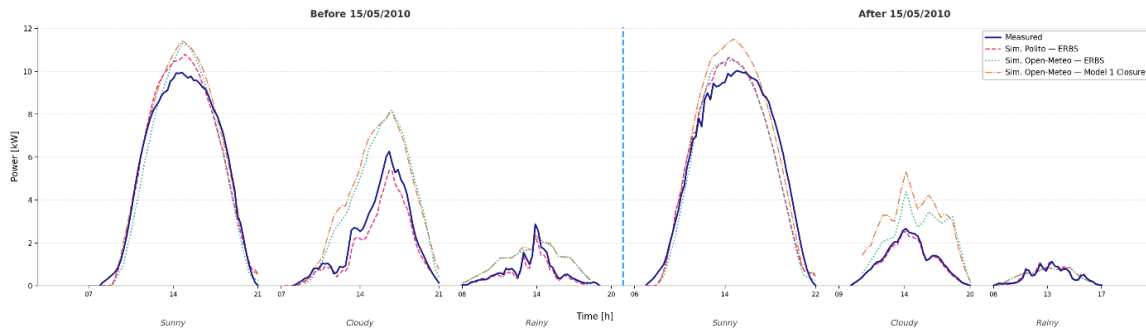


Figure 5.8. GalFer — simulated vs. measured power profiles by weather type, before and after the May 2010 malfunction.

The comparison between the two irradiance sources reveals a critical limitation of the reanalysis product for this application. As shown in [Table 5.7](#), the Open-Meteo source does not reproduce the before/after accuracy contrast observed with the Polito source. Under Open-Meteo, the before period actually yields worse accuracy than the after period, with daily RMSD of 57.67% before the malfunction against 28.07% after. This is the opposite of the pattern observed under Polito, where the before period achieves RMSD of 8.98% and the after period 16.17%. The explanation lies in the magnitude of the errors introduced by the reanalysis product itself. Open-Meteo introduces systematic errors of the order of 50 to 60% RMSD even during the pre-malfunction period when the system is fully operational. These errors are large enough to completely mask the additional divergence caused by the malfunction, which amounts to a shift of approximately 7 percentage points in RMSD under the Polito source. When the baseline error is already 57%, a 7-percentage point change is indistinguishable from normal simulation variability. The Polito source, by contrast, achieves a pre-malfunction baseline RMSD of 8.98%, making the post-malfunction increase to 16.17% a clearly detectable signal with a signal-to-noise ratio of approximately 1.8.

This finding establishes an important principle for the use of PV simulation as a fault detection tool. The sensitivity of the malfunction detection depends directly on the baseline accuracy of the simulation under normal operating conditions. A high-quality local irradi-

ance source is therefore not merely preferable but necessary for this application. A reanalysis product such as Open-Meteo, despite its convenience and global availability, introduces errors that are too large relative to the malfunction signal to enable reliable fault detection. This conclusion reinforces the practical importance of maintaining local meteorological measurement infrastructure in urban areas where PV monitoring and performance assessment are required.

It is also worth noting that the malfunction was not detected in the Sommelier system, which shares the same geographic location, the same installation year, the same module technology, and a similar nominal power as GalFer. The Sommelier simulation under Polito does not show any comparable discontinuity in accuracy across the monitoring period, suggesting that the degradation observed in GalFer is localised to that specific system and is not an artefact of the simulation methodology, the irradiance source, or the monitoring period. This cross-system cross-reference further supports the diagnostic value of the simulation infrastructure and indicates that it can distinguish between a genuine system-level malfunction and a general modelling limitation.

5.8 Discussion

The experimental results presented in this chapter address four interconnected objectives: decomposition model selection, simulation accuracy validation, weather source comparison, and fault detection. Taken together, the findings provide a comprehensive characterisation of the proposed simulation infrastructure across a wide range of system scales, installation types, orientations, and meteorological conditions.

The first and most consistent finding is the superiority of the ERBS decomposition model for the Turin urban area. ERBS was selected as the best performing model across all four systems and both irradiance sources, outperforming not only the other three decomposition variants but also Model 1 Closure, which uses DNI and DHI values supplied directly without decomposition. The failure of Model 1 Closure to match ERBS performance, despite bypassing the decomposition step entirely, is a significant result. It demonstrates that the quality of directly supplied irradiance components from reanalysis products and closure computations is insufficient to compensate for the inherent uncertainty of the decomposition approach when a high-quality GHI measurement is available as the starting point. The ERBS model, which was originally developed for hourly resolution and temperate climates,

proves to be robust and consistent across the Turin dataset spanning two distinct monitoring periods a decade apart, confirming its suitability as the default decomposition model for this geographic and climatic context.

The second major finding concerns simulation accuracy and its dependence on time resolution and weather conditions. Across all systems where the Polito source is used and the system is operating normally, daily RMSD values range from 9.46% for Cittadella to 44.41% for Sommelier. Excluding Sommelier, whose elevated errors are attributable to building-integration thermal effects rather than to the simulation methodology, and excluding the I3P West array, whose results are compromised by the sensor outage, the remaining systems achieve daily RMSD values between 9.46% and 14.68% under Polito. This narrow range across systems spanning nominal powers from 13.20 kWp to 631 kWp demonstrates that the simulation infrastructure maintains consistent accuracy regardless of system scale. The consistent degradation from daily to 15-minute resolution, with RMSD values approximately doubling at the finer time scale, is a known and expected characteristic of sub-hourly PV simulation driven by decomposed irradiance data [34] and does not represent a limitation specific to this infrastructure. Accuracy improves systematically from winter to summer and from rainy to sunny conditions across all systems and sources, with the most extreme percentage errors occurring on winter rainy days where near-zero absolute power output amplifies small absolute errors into large percentage values.

The third finding concerns the systematic superiority of the local Polito Weather Station over the Open-Meteo reanalysis product. The Polito source achieves lower RMSD than Open-Meteo for every system without exception, with improvement factors ranging from approximately 1.05 for the I3P West array to approximately 2.25 for Cittadella. The degradation under Open-Meteo is most severe under cloudy and rainy conditions, where the coarser spatial and temporal resolution of the reanalysis product cannot capture the rapid irradiance fluctuations recorded by the on-site pyranometer. Under sunny conditions, the two sources converge more closely, suggesting that the primary advantage of the local measurement lies in its ability to resolve sub-hourly irradiance variability under variable sky conditions rather than in its absolute accuracy under clear skies. The MBD analysis reveals that Open-Meteo introduces additional systematic positive bias for the 2010-2011 systems but not for the 2020 systems, suggesting that the reanalysis product accuracy may have

improved over time or that the bias characteristics differ between the two monitoring periods. This temporal aspect warrants further investigation in future work.

The fourth finding concerns the application of the simulation infrastructure as a fault detection tool. The GalFer malfunction analysis demonstrates that a simulation driven by a high-quality local irradiance source can detect hardware malfunctions through the emergence of a systematic divergence between simulated and measured generation. The malfunction signal, quantified as an increase in daily RMSD from 8.98% to 16.17% and a reversal in MBD sign, is clearly detectable under the Polito source but completely masked under Open-Meteo, where the baseline simulation errors are an order of magnitude larger than the malfunction signal itself. This finding establishes a minimum accuracy requirement for simulation-based fault detection: the baseline RMSD under normal operating conditions must be substantially smaller than the expected magnitude of the malfunction signal. For the GalFer case, the Polito source satisfies this requirement while Open-Meteo does not.

Several limitations of the present study should be acknowledged. The validation relies on a single local irradiance source, the Polito Weather Station, whose representativeness for the entire La Crocetta district is assumed but not verified through spatial interpolation or multi-station comparison. The I3P West array results are compromised by the sensor outage and cannot be considered a complete validation for that array. The two monitoring periods, 2010-2011 and 2020, are separated by a decade during which both the PV systems and the meteorological conditions may have changed, making direct cross-period comparisons of absolute accuracy values indicative rather than definitive. Finally, the simulation uses standard NOCT-based thermal models with fixed parameters, which may not fully capture the thermal behaviour of building-integrated systems such as Sommelier, contributing to the elevated MBD observed for that system.

Despite these limitations, the overall picture emerging from the validation is one of a robust and scale-invariant simulation infrastructure that achieves high accuracy when driven by quality local irradiance data, degrades gracefully when driven by reanalysis data, and retains sufficient sensitivity to detect hardware malfunctions in deployed systems. These characteristics position the infrastructure as a practical tool for urban PV planning, performance monitoring, and operational assessment across a wide range of application contexts.

5.9 Summary

This chapter presented and discussed the experimental validation of the proposed simulation infrastructure against measured generation data from four PV systems located in the La Crocetta district of Turin, Italy, spanning a nominal power range from 13.20 kWp to 631 kWp and two independent monitoring periods separated by a decade.

The chapter first established that the ERBS decomposition model is the most appropriate choice for the Turin urban area, outperforming three alternative decomposition variants and Model 1 Closure across all systems and both irradiance sources. This result was consistent across all validation cases and confirmed that estimating DNI and DHI from high-quality GHI measurements through the ERBS scheme is preferable to using directly supplied irradiance components from reanalysis products or closure computations.

The simulation validation results demonstrated that the infrastructure achieves high and consistent accuracy across systems of fundamentally different scales, installation types, and orientations when driven by the local Polito Weather Station. Daily RMSD values for normally operating systems ranged from 9.46% for the 631 kWp Cittadella installation to 14.68% for the 13.20 kWp GalFer system, confirming that simulation accuracy is scale-invariant and does not require system-specific calibration. The elevated errors observed for Sommelier were attributed to the building-integration thermal effects not fully captured by the standard NOCT thermal model, while the poor West array results were attributed to the sensor outage and geometric orientation factors rather than to limitations of the simulation methodology itself.

The comparison between the two irradiance sources confirmed that the local Polito Weather Station consistently outperforms the Open-Meteo reanalysis product, with improvement factors in daily RMSD ranging from 1.05 to 2.25 depending on the system. The degradation under Open-Meteo was most pronounced under cloudy and rainy conditions, where the reanalysis product cannot resolve the rapid irradiance fluctuations captured by the on-site pyranometer, and was minimal under sunny conditions. These findings highlight the practical importance of local meteorological measurement infrastructure for high-accuracy PV simulation in urban environments.

Finally, the chapter demonstrated that the simulation infrastructure can serve as a diagnostic tool for detecting hardware malfunctions in deployed PV systems. The GalFer malfunc-

tion analysis showed that a systematic divergence between simulated and measured generation, emerging at a specific point in time, can be reliably identified when the baseline simulation accuracy is sufficiently high. This application requires a high-quality local irradiance source, as the baseline errors introduced by reanalysis products are too large to allow the malfunction signal to be distinguished from normal modelling uncertainty.

Taken together, these results indicate that the proposed infrastructure provides a robust, accurate, and practically applicable framework for urban rooftop PV simulation, with demonstrated capabilities spanning energy yield estimation, cross-scale deployment assessment, weather source evaluation, and operational fault detection.

Conclusion

6.1 Summary of the work

This thesis addressed a methodological gap in urban rooftop photovoltaic assessment. While the existing literature has made substantial contributions to individual components of the problem, including geospatial rooftop representation, GIS-based solar radiation modelling, suitability analysis, and photovoltaic performance estimation, these components are typically addressed in isolation. Studies tend to present rooftop PV assessment either as a geospatial exercise or as a photovoltaic estimation problem, rarely as a coherent, reproducible, multi-stage workflow in which each stage feeds the next through explicit, reusable intermediate products. The gap is therefore methodological as much as analytical: what is missing is not another isolated model, but a structured framework that connects heterogeneous spatial inputs, terrain-derived analysis, solar preprocessing, suitability detection, and weather-aware photovoltaic estimation within a single backend-oriented, reproducible system.

In response to this gap, the thesis proposed, implemented, and validated a service-based computational framework for urban rooftop photovoltaic simulation. The framework integrates GRASS GIS and the `r.sun` radiation model through the Actinia execution environment to perform spatially explicit high-resolution geospatial preprocessing, including Digital Surface Model handling, slope and aspect derivation, horizon computation using `r.horizon`, and rooftop suitability analysis. A defining design decision is that all intermediate outputs are materialised as durable, structured Apache Parquet artefacts rather than treated as transient states. This decouples the computationally intensive geospatial stage from the downstream simulation, enabling different weather scenarios to be evaluated and different simulation branches to be executed without repeating the upstream computation, which improves traceability, reduces unnecessary re-computation, and supports reproducibility across repeated executions.

Two simulation paths are provided within the same architecture. The workflow-based real-sky path scales precomputed clear-sky baselines with observed meteorological data through a direct-to-diffuse decomposition step, then passes the resulting irradiance to a pvlib-based PVWatts power estimation module. The fast horizon-based path offers a computationally lighter alternative, combining locally derived horizon profiles from GRASS *r.horizon* with externally retrieved PVGIS horizon data, after harmonising their azimuth conventions, and performing direct centroid-level photovoltaic estimation for selected suitable areas. The framework also accommodates heterogeneous geospatial input conditions: when a high-resolution DSM is available it is used directly, and when it is not, building footprints and height information are used to construct an approximate rooftop-oriented surface. The entire system is deployed as a *FastAPI* backend with JWT-based authentication, PostgreSQL metadata tracking, and *Valkey* caching, and supports both batch and co-simulation execution models.

6.2 Main findings

The framework was validated against measured generation data from four photovoltaic systems located in the La Crocetta district of Turin, Italy: Sommelier (19.80 kWp, building-integrated), GalFer (13.20 kWp, free-standing), I3P (13.20 kWp, dual-pitch rooftop with East and West arrays), and Cittadella (631 kWp, large-scale flat rooftop). The validation spans two independent monitoring periods a decade apart, covering March 2010 to February 2011 and the full calendar year 2020. Turin's temperate continental climate, characterised by frequent winter overcast periods and variable irradiance conditions, represents a challenging and realistic environment for urban PV simulation.

The first finding concerns irradiance decomposition model selection. The ERBS model was identified as the most appropriate choice for the Turin urban context, consistently outperforming three alternative decomposition variants (DISC, DIRINT, and DIRINDEX) and Model 1 Closure across all four systems and both irradiance sources. The failure of Model 1 Closure to match ERBS performance, despite bypassing the decomposition step entirely, is a significant result: it demonstrates that high-quality GHI measurements decomposed through ERBS yield more accurate irradiance components than DNI and DHI values supplied directly from reanalysis products, whose systematic biases the decomposition approach avoids. The ERBS model proved robust and consistent across two monitoring periods a

decade apart, confirming its suitability as the default decomposition model for this geographic and climatic context.

The second finding concerns simulation accuracy. For normally operating systems driven by the local Polito Weather Station, the framework achieves daily RMSD values between 9.46% for Cittadella (631 kWp) and 14.68% for GalFer (13.20 kWp), with near-zero mean bias. This narrow range across systems spanning nearly fifty times the nominal power demonstrates that the simulation infrastructure maintains scale-invariant accuracy without requiring any system-specific calibration or parameter adjustment. The elevated errors observed for Sommelier (daily RMSD 44.41%) are attributable to building-integration thermal effects not fully captured by the standard NOCT thermal model and are therefore not a limitation of the simulation methodology itself. Similarly, the I3P West array results were affected by a sensor outage and orientation-related uncertainties and cannot be interpreted as a validation failure of the framework. The consistent degradation from daily to 15-minute resolution, with RMSD values approximately doubling at the finer time scale, is a known characteristic of sub-hourly PV simulation driven by decomposed irradiance data and does not represent a weakness specific to this infrastructure.

The third finding concerns the comparison between the two irradiance sources. The local Polito Weather Station consistently outperforms the Open-Meteo reanalysis product across all systems without exception, with daily RMSD improvement factors ranging from 1.05 for the I3P West array to 2.25 for Cittadella. The degradation under Open-Meteo is most pronounced under cloudy and rainy conditions, where the coarser spatial and temporal resolution of the reanalysis product cannot resolve the rapid irradiance fluctuations captured by the on-site pyranometer, while the gap narrows substantially under clear-sky conditions. For applications requiring only annual energy yield estimates, Open-Meteo may be an acceptable substitute when local measurements are unavailable, since annual biases remain small for most systems. For applications requiring sub-daily or daily accuracy, such as grid management, performance monitoring, or energy trading, local high-quality irradiance measurement is strongly preferable.

The fourth finding concerns an additional application of the framework: operational fault detection. The GalFer case study demonstrated that the simulation infrastructure can identify hardware malfunctions in deployed PV systems through the emergence of a systematic

divergence between simulated and measured generation. Before the malfunction of 15 May 2010, the Polito-driven simulation achieves a daily RMSD of 8.98%; after the malfunction, RMSD increases to 16.17% and MBD reverses sign from -3.29% to +3.97%, providing a clear and detectable signal. This signal is completely masked under Open-Meteo, where the pre-malfunction baseline RMSD already reaches 57.67%, an order of magnitude larger than the malfunction-induced divergence. This establishes an important principle: simulation-based fault detection requires that the baseline accuracy under normal operating conditions be substantially smaller than the expected magnitude of the malfunction signal, making a high-quality local irradiance source not merely preferable but necessary for this application.

6.3 Limitations

Several limitations of the present work should be acknowledged. First, the validation relies on a single local irradiance station whose spatial representativeness for the entire **La Crocetta** district is assumed but not verified through multi-station comparison or spatial interpolation. Second, the I3P West array results are substantially affected by a sensor outage from mid-June 2020, eliminating the summer period and making that array's validation incomplete. Third, the two monitoring periods are separated by a decade, during which both the PV systems and the local meteorological conditions may have changed, making direct cross-period comparisons of absolute accuracy values indicative rather than definitive. Finally, the simulation uses standard NOCT-based thermal models with fixed parameters, which do not fully capture the thermal behaviour of building-integrated installations such as *Sommelier*, contributing to the elevated systematic overestimation observed for that system.

6.4 Future directions

Several directions for future work emerge naturally from these findings. First, the integration of multiple local weather stations or spatially interpolated irradiance fields would allow the influence of meteorological-source location to be assessed more rigorously and would strengthen the spatial validity of the simulation results. Second, the refinement of thermal modelling, especially for building-integrated PV systems, could substantially reduce the systematic bias observed for systems such as *Sommelier*, for instance by replacing the standard NOCT model with installation-type-specific or dynamic thermal parameters. Third, further validation across additional cities, climates, rooftop types, and PV technologies would help assess the generalisability of the framework beyond the Turin case study. Fourth, the

fast horizon-based simulation branch offers a promising avenue for computational efficiency at scale but requires dedicated validation against measured data to characterise its accuracy trade-offs relative to the full workflow-based path. Finally, the malfunction-detection capability demonstrated in the GalFer case could be formalised into a systematic diagnostic module for operational PV performance monitoring, extending the practical scope of the framework beyond planning and assessment contexts.

6.5 Concluding remarks

This thesis has shown that rooftop photovoltaic assessment benefits from being treated as a reproducible, staged, and backend-oriented computational workflow rather than as a sequence of isolated modelling tasks. By combining geospatial preprocessing, clear-sky solar radiation modelling, reusable analytical artefacts, weather-aware photovoltaic simulation, and validation against measured production data, the proposed framework provides a practical and scalable infrastructure for urban rooftop PV analysis. The validation results demonstrate that, when supported by high-quality local irradiance data, the framework achieves accurate cross-scale simulation performance, supports meaningful weather-source comparison, and enables operational assessment of deployed PV systems. The thesis therefore contributes both methodologically and practically: methodologically, by structuring rooftop PV assessment as a reusable, traceable, and reproducible computational process; and practically, by demonstrating its applicability to real urban PV systems under realistic weather and operational conditions.

References

- [1] B. Resch, G. Sagl, T. Tornros, A. Bachmaier, J.-B. Eggers, S. Herkel, S. Narmsara, and H. Gundra, "GIS-based planning and modeling for renewable energy: Challenges and future research avenues," *ISPRS International Journal of Geo-Information*, vol. 3, no. 2, pp. 662–692, 2014.
- [2] L. R. Camargo, R. Zink, W. Dorner, and G. Stoeglehner, "Spatio-temporal modeling of rooftop photovoltaic panels for improved technical potential assessment and electricity peak load offsetting at the municipal scale," *Computers, Environment and Urban Systems*, vol. 52, pp. 58–69, 2015.
- [3] J. A. Jakubiec and C. F. Reinhart, "A method for predicting city-wide electricity gains from photovoltaic panels based on lidar and GIS data combined with hourly Daysim simulations," *Solar Energy*, vol. 93, pp. 127–143, 2013.
- [4] N. Lukac, S. Seme, D. Zlaus, G. Stumberger, and B. Zalik, "Buildings roofs photovoltaic potential assessment based on LiDAR (Light Detection and Ranging) data," *Energy*, vol. 66, pp. 598–609, 2014.
- [5] S. Freitas, C. Catita, P. Redweik, and M. Brito, "Modelling solar potential in the urban environment: State-of-the-art review," *Renewable and Sustainable Energy Reviews*, vol. 41, pp. 915–931, 2015.
- [6] X. Li, L. Di, W. Han, P. Zhao, and U. Dadi, "Sharing geoscience algorithms in a web service-oriented environment (GRASS GIS example)," *Computers & Geosciences*, vol. 36, no. 8, pp. 1060–1068, 2010.
- [7] F. Qiu, F. Ni, B. Chastain, H. Huang, P. Zhao, W. Han, and L. Di, "GWASS: GRASS web application software system based on the GeoBrain web service," *Computers & Geosciences*, vol. 47, pp. 143–150, 2012.
- [8] M. Suri, T. Huld, E. Dunlop, and T. Cebecauer, "Geographic aspects of photovoltaics in Europe: Contribution of the PVGIS website," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 1, no. 1, pp. 34–41, 2008.
- [9] B. Marion and M. Anderberg, "PVWatts—an online performance calculator for grid-connected PV systems," in *Proceedings of the Solar Conference*, 2000, pp. 119–124.
- [10] M. Fowler and J. Lewis, "Microservices," 2014.

- [11] United Nations, FCCC, "Adoption of the Paris Agreement. Proposal by the President," 2015.
- [12] L. Dusonchet and E. Telaretti, "Economic analysis of different supporting policies for the production of electrical energy by solar photovoltaics in western European Union countries," *Energy Policy*, vol. 38, no. 7, pp. 3297–3308, 2010.
- [13] International Energy Agency, "Tracking Clean Energy Progress 2016," 2016.
- [14] J. Hofierka and J. Kanuk, "Assessment of photovoltaic potential in urban areas using open-source solar radiation tools," *Renewable Energy*, vol. 34, no. 10, pp. 2206–2214, 2009.
- [15] L. Bergamasco and P. Asinari, "Scalable methodology for the photovoltaic solar energy potential assessment based on available roof surface area: Further improvements by ortho-image analysis and application to Turin (Italy)," *Solar Energy*, vol. 85, no. 11, pp. 2741–2756, 2011.
- [16] M. H. Kabir, W. Endlicher, and J. Jägermeyr, "Calculation of bright rooftops for solar PV applications in Dhaka megacity, Bangladesh," *Renewable Energy*, vol. 35, no. 8, pp. 1760–1764, 2010.
- [17] M. C. Brito, N. Gomes, T. Santos, and J. A. Tenedorio, "Photovoltaic potential in a Lisbon suburb using LiDAR data," *Solar Energy*, vol. 86, no. 1, pp. 283–288, 2012.
- [18] A. Ronzino, A. Osello, E. Patti, L. Bottaccioli, C. Danna, A. M. Lingua, A. Acquaviva, E. Macii, M. Grosso, G. Messina, and G. Rascon, "The energy efficiency management at urban scale by means of integrated modelling," in *Proceedings of SEB-15*. Elsevier.
- [19] GRASS Development Team, *Geographic Resources Analysis Support System (GRASS GIS) Software*, Open-Source Geospatial Foundation, USA, 2015.
- [20] M. Šuri and J. Hofierka, "A new GIS-based solar radiation model and its application to photovoltaic assessments," *Transactions in GIS*, vol. 8, no. 2, pp. 175–190, 2004.
- [21] N. Blair, A. Dobos, J. Freeman, T. Neises, M. Wagner, T. Ferguson, P. Gilman, and S. Janzou, "System Advisor Model, SAM 2014.1.14: General description," 2014.
- [22] L. de Sousa, C. Eykamp, U. Leopold, O. Baume, and C. Braun, "iGUESS: A web-based system integrating urban energy planning and assessment modelling for multi-scale spatial decision making," in *Proceedings of iEMSs 2012*, 2012.
- [23] R. De Amicis, G. Conti, D. Patti, M. Ford, and P. Elisei, *I-SCOPE: Interoperable Smart City Services through an Open Platform for Urban Ecosystems*, 2012.
- [24] M. Brumen, N. Lukac, and B. Zalik, "GIS application for solar potential estimation on buildings roofs," in *Proceedings of the WEB2014 International Conference on Building and Exploring Web Based Environments*, IARIA, 2015.

- [25] PVGIS, "PVGIS web application," available through the Joint Research Centre photovoltaic geographical information system platform.
- [26] Mapdwell Solar System, available online.
- [27] F. Spertino and F. Corona, "Monitoring and checking of performance in photovoltaic plants: A tool for design, installation and maintenance of grid-connected systems," *Renewable Energy*, vol. 60, pp. 722–732, 2013.
- [28] D. Erbs, S. Klein, and J. Duffie, "Estimation of the diffuse radiation fraction for hourly, daily and monthly-average global radiation," *Solar Energy*, vol. 28, no. 4, pp. 293–302, 1982.
- [29] J. Ruiz-Arias, H. Alsamamra, J. Tovar-Pescador, and D. Pozo-Vázquez, "Proposal of a regressive model for the hourly diffuse solar radiation under all sky conditions," *Energy Conversion and Management*, vol. 51, no. 5, pp. 881–893, 2010.
- [30] S. Karatasou, M. Santamouris, and V. Geros, "Analysis of experimental data on diffuse solar radiation in Athens, Greece, for building applications," *International Journal of Sustainable Energy*, vol. 23, no. 1–2, pp. 1–11, 2003.
- [31] D. T. Reindl, W. A. Beckman, and J. A. Duffie, "Diffuse fraction correlations," *Solar Energy*, vol. 45, no. 1, pp. 1–7, 1990.
- [32] A. Skartveit and J. A. Olseth, "A model for the diffuse fraction of hourly global radiation," *Solar Energy*, vol. 38, no. 4, pp. 271–274, 1987.
- [33] P. Ineichen, "Comparison and validation of three global-to-beam irradiance models against ground measurements," *Solar Energy*, vol. 82, no. 6, pp. 501–512, 2008.
- [34] C. A. Gueymard and J. A. Ruiz-Arias, "Extensive worldwide validation and climate sensitivity analysis of direct irradiance predictions from 1-min global irradiance," *Solar Energy*, 2015.
- [35] M. Mattei, G. Notton, C. Cristofari, M. Muselli, and P. Poggi, "Calculation of the polycrystalline PV module temperature using a simple method of energy balance," *Renewable Energy*, vol. 31, no. 4, pp. 553–567, 2006.
- [36] C. Schwingshackl, M. Petitta, J. E. Wagner, G. Belluardo, D. Moser, M. Castelli, M. Zebisch, and A. Tetzlaff, "Wind effect on PV module temperature: Analysis of different techniques for an accurate estimation," *Energy Procedia*, vol. 40, pp. 77–86, 2013.
- [37] S. Younes, R. Claywell, and T. Muneer, "Quality control of solar radiation data: Present status and proposed new approaches," *Energy*, vol. 30, no. 9, pp. 1533–1549, 2005.
- [38] C. A. Gueymard, "A review of validation methodologies and statistical performance indicators for modeled solar radiation data: Towards a better bankability of solar projects," *Renewable and Sustainable Energy Reviews*, vol. 39, pp. 1024–1034, 2014.

- [39] J. A. Duffie and W. A. Beckman, *Solar Engineering of Thermal Processes*. Wiley, 2013.
- [40] N. Engerer, "Minute resolution estimates of the diffuse fraction of global irradiance for southeastern Australia," *Solar Energy*, vol. 116, pp. 215–237, 2015.
- [41] C. Jacovides, F. Tymvios, V. Assimakopoulos, and N. Kaltsounides, "Comparative study of various correlations in estimating hourly diffuse fraction of global solar radiation," *Renewable Energy*, vol. 31, no. 15, pp. 2492–2504, 2006.
- [42] L. Menard, I. Blanc, D. Beloin-Saint-Pierre, B. Gschwind, L. Wald, P. Blanc, T. Ranchin, R. Hirschier, S. Gianfranceschi, S. Smolders et al., "Benefit of GEOSS interoperability in assessment of environmental impacts illustrated by the case of photovoltaic systems," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 6, pp. 1722–1728, 2012.
- [43] S. Newman, *Building Microservices*. O'Reilly Media, Inc., 2015.
- [44] L. Bottaccioli, E. Patti, M. Grosso, G. Rascona, A. Marotta, S. Rinaudo, A. Acquaviva, and E. Macii, "Distributed software infrastructure for evaluating the integration of photovoltaic systems in urban districts," in *Proceedings of the 5th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, 2016, pp. 357–362.
- [45] L. Bottaccioli, A. Estebarsari, E. Patti, E. Pons, and A. Acquaviva, "A novel integrated real-time simulation platform for assessing photovoltaic penetration impacts in smart grids," *Energy Procedia*, vol. 111, pp. 780–789, 2017.
- [46] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology*, vol. 2, no. 2, pp. 115–150, 2002.
- [47] Open Geospatial Consortium, available online.
- [48] L. Bottaccioli, E. Patti, A. Acquaviva, E. Macii, M. Jarre, and M. Noussan, "A toolchain to foster a new business model for photovoltaic systems integration exploiting an energy community approach," in *Proceedings of IEEE ETFA 2015*, 2015.
- [49] L. Bottaccioli, E. Patti, E. Macii, and A. Acquaviva, "GIS-based Software Infrastructure to Model PV Generation in Fine-grained Spatio-temporal Domain."
- [50] Willmott, C. J., "On the validation of models," *Physical Geography*, vol. 2, no. 2, pp. 184–194, 1981.
- [51] Legates, D. R. and McCabe, G. J., "Evaluating the use of goodness-of-fit measures in hydrologic and hydroclimatic model validation," *Water Resources Research*, vol. 35, no. 1, pp. 233–241, 1999.

Appendix: Implementation-oriented UML Diagrams

This appendix provides implementation-oriented UML-style diagrams that document the internal backend organisation of the proposed rooftop photovoltaic simulation framework. These diagrams complement the methodological description presented in Chapter 4 by showing how the implemented backend is organised into API modules, service packages, repositories, ORM models, core utilities, external systems, and domain-specific solar service groups. They are intended as supporting technical documentation rather than as additional methodological workflow figures.

A.1 Core backend service structure

Figure A.1 summarises the main implementation-level organisation of the backend. It shows the separation between API modules, authentication and user services, grouped solar services, ORM models, repositories, core utilities, and external execution or persistence systems. The diagram is intended to provide a compact overview of the principal backend components that support controlled workflow execution, metadata persistence, session context handling, and geospatial command execution.

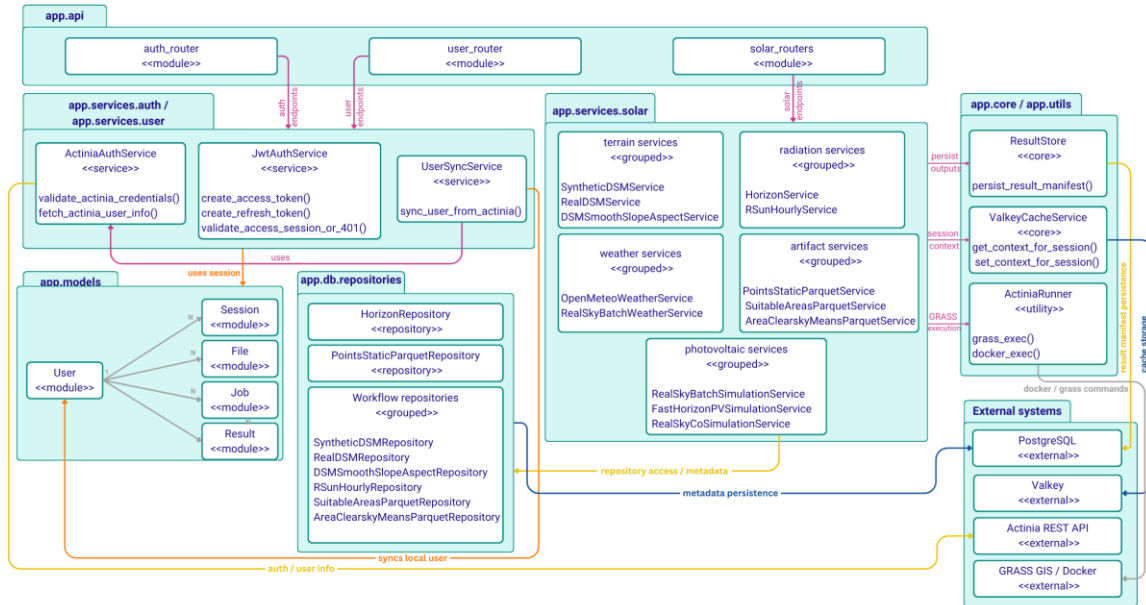


Figure A.1. Core backend service class diagram.

A.2 Solar service package structure

Figure A.2 expands the internal organisation of the *app.services.solar* package. It groups the implemented services into terrain, radiation, artefact, weather, and photovoltaic components, and shows the principal dependency flows between them.

The diagram also highlights the distinction between the workflow-based photovoltaic branch, which relies on area-level clear-sky artefacts, and the fast horizon-based branch, which combines selected suitable areas, local horizon information, prepared weather data, and azimuth-convention harmonisation.

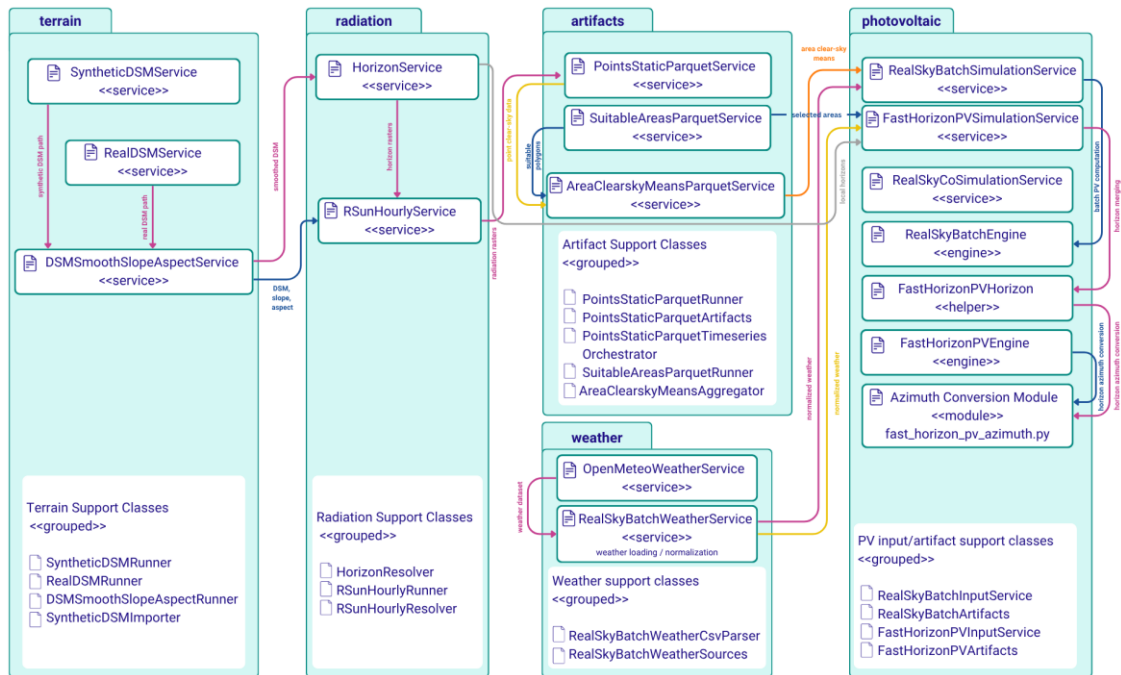


Figure A.2. UML diagram of the solar service package.

A.3 Selected implementation excerpts

The following listings complement the implementation-oriented diagrams presented in the previous appendix sections by showing selected code excerpts from the backend code-base. They are not intended to reproduce complete source files or replace the methodological explanation provided in Chapter 4. Instead, each excerpt documents a representative implementation decision that supports the proposed workflow, including centralised configuration, containerised GRASS execution, artefact persistence, weather acquisition, workflow-based real-sky photovoltaic simulation, and azimuth-convention harmonisation in the fast horizon-based branch. This selection follows the same principle used for the architectural figures in the methodology chapter: the code is shown only where it clarifies how an important methodological concept is realised in the implemented system.

Before considering the execution of individual workflow stages, it is useful to clarify how deployment-specific parameters are separated from the service logic. The framework relies on a central configuration object to define the external services and runtime paths required by the backend. This includes the Actinia connection, host-side and container-side storage

roots, workspace directories, authentication parameters, and database connectivity. The excerpt in **Listing A.1** shows the representative structure of this configuration layer.

```
class Settings (BaseSettings):
    # Actinia / storage
    ACTINIA_BASE_URL: str = Field(...)
    ACTINIA_USERNAME: str | None = os.getenv("ACTINIA_USER")
    ACTINIA_PASSWORD: str | None = os.getenv("ACTINIA_PASSWORD")
    FILE_STORAGE_ROOT: str = Field(...)
    USERDATA_CONTAINER_ROOT: str = Field(...)

    # Workspace execution
    ACTINIA_CONTAINER_NAME: str | None = Field(default=None)
    VALKEY_CONTAINER_NAME: str | None = Field(default=None)
    WORKSPACE_TMP_HOST_ROOT: str = "./data/actinia-data/workspace/tmp"
    WORKSPACE_TMP_CONTAINER_ROOT: str = "/actinia_core/workspace/tmp"

    # JWT / authentication
    JWT_SECRET: str = Field(...)
    JWT_ALG: str = "HS256"
    ACCESS_TTL: int = 86400
    REFRESH_TTL: int = 604800

    SQLALCHEMY_DATABASE_URL: str = Field(...)
```

Listing A.1. Centralized backend configuration and path normalisation.

Listing A.1 illustrates how the backend avoids embedding deployment assumptions directly in the implementation of the workflow services. The same service logic can therefore be executed in different environments by changing configuration values rather than modifying code. This is especially important for the proposed framework because the geospatial runtime depends on the coordination of host storage, container-visible paths, authentication settings, and database persistence. In this sense, the configuration layer supports the reproducibility and deployment portability principles discussed in the methodology chapter.

The geospatial stages of the framework require the backend to execute GRASS GIS modules inside the containerised Actinia environment. Rather than duplicating command-execution logic across individual services, the implementation centralises this responsibility in a shared runner utility. This runner resolves the active GRASS mapset context, constructs

the corresponding *grass ... --exec command*, executes it through the container runtime, and reports diagnostic information when execution fails. Listing A.2 provides a compact excerpt of this execution pattern.

```
exec_part = " ".join(shlex.quote(a) for a in args)
prefix = "GRASS_OVERWRITE=1 " if overwrite else ""
shell_cmd = f"{prefix}grass {shlex.quote(mapset_path)} --exec {exec_part}"

proc = self.docker_exec_sh(
    shell_cmd=shell_cmd,
    check=False,
    error_context="GRASS command failed",
)

if check and proc.returncode != 0:
    raise RuntimeError(
        "GRASS command failed\n"
        f"owner={grass_owner} requested_owner={owner} "
        f"location={location} mapset={mapset}\n"
        f"CMD: docker exec {container} sh -lc {shell_cmd}\n"
        f"STDOUT:\n{proc.stdout}\n"
        f"STDERR:\n{proc.stderr}\n"
    )
return proc
```

Listing A.2. Executing GRASS commands inside the Actinia container.

Listing A.2 shows the bridge between the FastAPI service layer and the GRASS-based geospatial runtime. The individual terrain, horizon, and clear-sky radiation services do not execute Docker commands independently; instead, they delegate command execution to a common runner. This keeps the geospatial runtime behaviour consistent across workflow stages and makes failures easier to trace, since the raised error includes the execution context, command, standard output, and standard error. The listing therefore supports the methodological claim that GRASS-side processing is executed within a controlled and reproducible containerised environment.

A central part of the proposed framework is that intermediate and final outputs are not treated as temporary files only. Instead, generated artefacts are registered as structured result metadata so that they can be traced, reloaded, and reused by later stages of the work-

flow. This is particularly important in a multi-stage pipeline where outputs from one computation, such as Parquet artefacts or simulation results, may become inputs to downstream services. Listing A.3 shows the persistence pattern used to register workflow result manifests in the database.

```
def persist_result_manifest(
    self,
    *,
    db: Session,
    job_id: int,
    user_id: int,
    manifest: Optional [Iterable[Dict[str, Any]]] = None,
) -> int:
    normalised = self.iter_manifest_entries(manifest)
    if not normalised:
        return 0

    existing_file_paths = {
        row[0]
        for row in db.query(ResultModel.file_path)
        .filter(ResultModel.job_id == int(job_id))
        .all()
    }

    created = 0
    for item in normalised:
        if item["file_path"] in existing_file_paths:
            continue

        db.add(ResultModel(
            job_id=int(job_id),
            user_id=int(user_id),
            result_name=item["result_name"],
            result_type=item["result_type"],
            result_format=item["result_format"],
            file_path=item["file_path"],
        ))
        created += 1

    return created
```

Listing A.3. Persisting workflow result manifests.

Listing A.3 illustrates how the backend converts generated outputs into persistent result records. The method first normalises the entries contained in a result manifest, checks whether the same file path has already been registered for the job, and then creates database rows only for new artefacts. This supports the artefact-driven design of the framework because produced files are not only stored on disk but are also linked to jobs and users through structured metadata. As a result, later workflow stages and inspection tools can resolve previous outputs through the database instead of relying on transient API responses or manual path tracking.

Real-sky photovoltaic simulation requires meteorological inputs that are consistent across the different simulation branches. In the implemented framework, weather data may be supplied by the user or retrieved from an external weather service. When the Open-Meteo pathway is used, the backend validates the requested location and date interval and then requests a fixed set of weather and irradiance variables required by the photovoltaic models. **Listing A.4** shows the initial part of this canonical weather retrieval procedure.

```
def run(
    self,
    *,
    latitude: float,
    longitude: float,
    start_date: str,
    end_date: str,
    timezone: str = "UTC",
) -> pd.DataFrame:
    """
    Fetch weather data from Open-Meteo and return canonical columns
    for photovoltaic simulation.
    """
    _ = timezone
    self._validate_query_inputs(
        latitude=latitude,
        longitude=longitude,
        start_date=start_date,
        end_date=end_date,
    )

    hourly = [
```

```

"temperature_2m",
"wind_speed_10m",
"direct_normal_irradiance",
"diffuse_radiation",
"shortwave_radiation",
]

```

Listing A.4. Canonical weather retrieval from Open-Meteo.

Listing A.4 shows how the *Open-Meteo* retrieval path is constrained to the meteorological and irradiance variables required by the photovoltaic simulation stages. The service validates the spatial and temporal query parameters before requesting temperature, wind speed, DNI, diffuse radiation, and shortwave radiation. These variables are later normalised into the canonical weather representation used by the real-sky workflow-based branch and by the fast horizon-based simulation branch.

In the workflow-based photovoltaic simulation branch, the framework does not recompute the full solar-radiation model for every real weather timestamp. Instead, it reuses the pre-computed rooftop-level clear-sky area means and scales them according to the real weather conditions associated with the current timestamp. This mechanism is the operational link between the clear-sky artefact layer and the final real-sky photovoltaic simulation.

Listing A.5 shows the core logic used to compute weather-based scaling factors, resolve the corresponding clear-sky snapshot, and derive the real-sky irradiance on the rooftop surface.

```

# 2) clear-sky reference
cs = self.pv_location.get_clearsky(pd.DatetimeIndex([ts_utc]), model=self.clearsky_model)
dni_clear = float(cs["dni"].iloc[0])
dhi_clear = float(cs["dhi"].iloc[0])

# 3) scaling factors
k_dir = _safe_ratio(step_weather["dni"], dni_clear)
k_diff = _safe_ratio(step_weather["dhi"], dhi_clear)

# 4) map real time -> clear-sky snapshot key
timestamp_clear = _map_timestamp_to_timestamp_clear(timestamp_real=ts_utc.to_pydatetime())

# 5) load hot means
beam_clear_area_mean, diff_clear_area_mean = _load_hot_means_for_area_timestamp(

```

```

hot_root=self.hot_root,
location=self.location_name,
mapset=self.mapset_name,
suitable_area_id=self.suitable_area_id,
timestamp_clear=timestamp_clear,
)

# 6) GTI on area plane
beam_real = float(beam_clear_area_mean) * k_dir
diff_real = float(diff_clear_area_mean) * k_diff
gti_area = beam_real + diff_real

```

Listing A.5. Scaling clear-sky area means with real weather.

Listing A.5 illustrates the central computation performed by the workflow-based real-sky simulation branch. The simulator first computes clear-sky reference irradiance for the real timestamp and derives direct and diffuse scaling factors from the weather input. It then maps the real timestamp to the corresponding precomputed clear-sky snapshot, loads the rooftop-level clear-sky area means, and applies the weather-based scaling factors to obtain real-sky irradiance on the candidate PV surface. This implementation reflects the artefact-driven methodology of the framework, since expensive GRASS-based solar-radiation outputs are reused rather than recomputed for each weather scenario.

In the fast horizon-based simulation branch, directional information is combined from different sources. Local horizon profiles are derived from GRASS-based outputs, while external horizon information and solar-position calculations follow the PVGIS and *pvlb* azimuth convention. Because these systems do not represent azimuth directions in the same way, the implementation must harmonise the directional reference before merging horizon profiles or using surface aspect values in *pvlb*. Listing A.6 shows the conversion helper used to transform GRASS directional angles into the clockwise-from-north convention used by *pvlb* and PVGIS.

```

def grass_angle_to_pvlb_azimuth(angle: float) -> float:
    """
    Convert GRASS horizon/aspect angle to pvlb/PVGIS azimuth.
    GRASS r.horizon directions are counterclockwise from east:
    0=East, 90=North, 180=West, 270=South. The aspect values used by this
    workflow follow the same cardinal convention. pvlb/PVGIS azimuths are

```

```

clockwise from north: 0=North, 90=East, 180=South, 270=West.
"""

return (90.0 - float(angle)) % 360.0

def local_horizon_profile_points_in_pvlib_order(
    *,
    local_horizon_profile: list[float],
    grass_angles_deg: list[int],
) -> list[tuple[float, float, int]]:
    if len(local_horizon_profile) != len(grass_angles_deg):
        raise ValueError(
            "Local horizon profile length does not match GRASS direction grid."
        )
    points = [
        (
            grass_angle_to_pvlib_azimuth(grass_angle),
            float(elevation),
            int(grass_angle),
        )
        for grass_angle, elevation in zip(grass_angles_deg, local_horizon_profile)
    ]

    return sorted(points, key=lambda item: item[0])

```

Listing A.6. Azimuth convention harmonisation for fast horizon PV.

```

def do_step(self, *, timestamp, weather):
    # 1) Require a UTC-aware timestamp
    if timestamp.tzinfo is None:
        raise RealSkySimInputError({"error": "timestamp_not_utc_aware"})
    ts_utc = pd.Timestamp(timestamp).tz_convert("UTC")
    step_weather = self._parse_weather_values(weather)

    # 2) Clear-sky reference (pvlib)
    cs = self.pv_location.get_clearsky(pd.DatetimeIndex([ts_utc]), model=self.clearsky_model)
    dni_clear, dhi_clear = float(cs["dni"].iloc[0]), float(cs["dhi"].iloc[0])

    # 3) Scaling factors from real vs clear-sky irradiance
    k_dir = _safe_ratio(step_weather["dni"], dni_clear)
    k_diff = _safe_ratio(step_weather["dhi"], dhi_clear)

    # 4) Map real time to the nearest clear-sky snapshot and load area means

```

```

timestamp_clear = _map_timestamp_to_timestamp_clear(timestamp_real=ts_utc.to_pydatetime())
beam_clear_area_mean, diff_clear_area_mean = _load_hot_means_for_area_timestamp(
    hot_root=self.hot_root, location=self.location_name, mapset=self.mapset_name,
    suitable_area_id=self.suitable_area_id, timestamp_clear=timestamp_clear)

# 5) Real-sky GTI on the area plane
gti_area = beam_clear_area_mean * k_dir + diff_clear_area_mean * k_diff

# 6) Cell temperature (pvlib SAPM or PVsyst)
if self.temp_model == "sapm":
    temp_cell_c = float(pvlib.temperature.sapm_cell(
        poa_global=gti_area, temp_air=step_weather["temp_air"],
        wind_speed=step_weather["wind_speed"], **self.temp_params))
else: # pvsyst
    temp_cell_c = float(pvlib.temperature.pvsyst_cell(
        poa_global=gti_area, temp_air=step_weather["temp_air"],
        wind_speed=step_weather["wind_speed"], **self.temp_params))

# 7) PVWatts DC power
pdc_w = float(pvlib.pvsystem.pvwatts_dc(
    effective_irradiance=gti_area, temp_cell=temp_cell_c,
    pdc0=self.pdc0_W, gamma_pdc=self.gamma_pdc, temp_ref=25.0))
return {"timestamp": ts_utc, "gti": gti_area,
        "temp_cell": temp_cell_c, "pdc_w": pdc_w}

```

Listing A.7. Per-timestep real-sky photovoltaic computation: clear-sky scaling, cell temperature, and PVWatts DC power.

Listing A.6 documents the directional harmonisation step required by the fast horizon-based branch. The helper converts GRASS horizon and aspect directions into the *pvlib*/*PVGIS* convention using a deterministic transformation, then orders the local horizon profile according to the converted azimuth grid. This prevents local DSM-derived horizons from being merged with *PVGIS* horizons under inconsistent directional references. The same convention alignment also supports the use of representative surface aspect values in *pvlib*-based irradiance calculations.