Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Theorie der hybriden Systeme

3D Solar Irradiance Profiling of Buildings Using LiDAR and LOD Dat

3D-Solarstrahlungsprofilierung von Gebäuden mithilfe von LiDAR- und LOD-Daten

Bachelorarbeit
Informatik

September 22, 2025

Vorgelegt von Presented by	Markus Ehring Matrikelnummer: 377820 markus.ehring@rwth-aachen.de
Erstprüfer First examiner	Prof. Dr. rer. nat. Erika Ábrahám Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University
Zweitprüfer Second examiner	Prof. Dr. rer. nat. Jürgen Giesl Lehr- und Forschungsgebiet: Programmiersprachen und Verifikation RWTH Aachen University
Betreuer Supervisor	Dr. rer. nat. Pascal Richter Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University

Contents

1 Introduction

1.1 Motivation

The adoption and deployment of solar panels continues increasing, extending beyond large-scale installations to private buildings, commercial buildings, and various other structures such as balconies. This widespread adoption is driven by the need for renewable energy sources to combat climate change and reduce the dependence on fossil energies such as oil and gas. Solar radiation is globally available for free with zero carbon emissions. As solar panel and energy storage solution become more and more adopted, their prices decrease substantially due to mass production and technological advancements. The successful installation of solar panel and energy storage systems rely on several critical factors from the environment that influence their efficiency and overall energetic and financial performance. A fundamental aspect in determining the viability of solar panel installations is the general availability of solar radiation in a specific area, which directly impacts the energy output of panels. The orientation of rooftops is crucial to maximizing sunlight capture, as they determine the angle at which solar rays shine on the panels. In addition, obstacles such as chimneys, satellite dishes, nearby buildings, trees, or other urban structures can cast shadows on solar panels, leading to significant reductions in energy production. Since some solar panels experience substantial performance losses even when only a single cell is shaded, the correct placement of the panels has to be carefully planned to avoid decreasing the usefulness of the system. In order to ensure the efficient planning of solar energy, accurate and accessible environmental profiles are needed. These profiles need to represent the environment of the target installation site in all aspects that can influence the efficiency and effectiveness of a solar panel and energy storage system. Solar panel and energy storage systems on residential buildings can be categorized into three major groups:

Rooftop photovoltaic with battery storage: Rooftop photovoltaic panels are large silicon panels that convert solar irradiance directly into electricity by the photovoltaic effect. The generated electricity can be consumed by the residents, sold into the energy grid or be stored in a battery. Today most photovoltaic panels are installed with a large battery that enables the system to store electricity that is not directly consumed instead of putting it in the electricity grid. This enables the system to store access energy during the day and make it accessible during the night. A battery can also bridge weather events that decrease the panels output.

Rooftop flat-plate collectors with thermal storage: Rooftop flat-plate collectors are large heat exchangers that collect the thermal energy of the sun. The collected heat gets transferred into a warm water storage installation. The heated water enables the system to reduce the need of heating water with oil, gas or electricity.

Balcony power plant: Balcony power plants are smaller scale photovoltaic installa-

tions. They are also based on silicon panels that convert solar irradiance directly into electricity. The key difference to rooftop photovoltaic systems is that they are not mounted on the roof of a building but mostly on balconies, around windows or walls. Their panels are usually smaller and can be installed more easily. Balcony power plants are also available with battery storage.

Existing solutions like Google Solar API[9] do provide complete solutions for profiling the environment and suggesting optimal irradiance capture systems based on personal energy usage and location. As they are focused on rooftop mounted installations they do not provide a true three dimensional environmental profile. Instead they rely on two dimensional elevation maps and top down rasterized two dimensional irradiance profiles.
In order to provide environmental data that is suitable for all types of irradiance capture systems, a fully three dimensional profile, that contains all environmental data points, is needed. Especially balcony power plants are in need of 3D profiles as their mounting locations are not always visible from above. The profile also has to be flexible enough to enable manual or automated changes to the dataset.

1.2 Related Work

This thesis is based preliminary on three projects that developed tools we were able to implement in our process of creating a 3D environmental profile or were used as an inspiration for our workflow:

PCSRT: The Point Cloud Solar Radiation Tool (PCSRT) [14] is an open source MIT licensed voxel based irradiance simulator. We integrate its irradiance simulation into our data.

Image Detection: A tool developed by Maqsood in his master thesis [12] enables to label potential obstacles on roof surfaces on aerial and satellite imagery. We build upon this tool to enhance our environmental profile.

Foundational Influence: Rastoder's Master thesis [15] offers a framework for rooftop solar potential assessment using voxel-based ray tracing on LiDAR and LOD2 CityJSON models, motivating our environmental profiling approach. We extend this approach by developing more advanced data merging methods and create a complete environmental profile using additional methods and data sources.

1.3 Contribution

This thesis contributes to the field of 3D environment models in the following points:

Data merging: We gather and combine a multitude of data sets into an accurate 3D representation of the environment around a structure.

Simulation workflow: We develop a modular and efficient process that is capable of gathering environmental data around a target, simulating it all with an optimal setup regarding accuracy and efficiency.

3D environment profile: We develop a complete 3D profile that encompasses all environmental data that influences the optimal placing and quantity of potential solar panels and storage installations.

1.4 Outline of This Work

In Section 2 all data sources for the simulations are explained and an overview over the software is given. In Section 3 the individual components and steps of the software are explained in detail. Section 4 focuses on the evaluation of the quality and performance of the resulting profile by evaluating the effect of each parameter going into our process. In Section 5 we summarize the key findings by presenting the optimal parameters for our process and suggest applications of our for future work.

2 Data and Software Overview

2.1 Data

Our profiling process relies on real world data for accurate results. For geometry information we use public datasets made available by the german government. How the data is provided is organized by each federal state. Most states have a public and freely available geoportal that provide many environmental datasets for public and private use. Availability of the datasets varies by state and location.

2.1.1 LoD Data

Our software and any potential software using the generated profile of a building needs a ground truth representation of the target building. That ground truth representation is considered as the actual buildings shape and location in 3D space. Only when there is a common understanding of the targets structure, reasonable data can be extracted and simulated on the building and its environment. A potential panel and storage system optimizer can use the same established ground truth to reconstruct the 3D environment. It is needed for optimal panel placement and conformity with regulations. As the datatype of the 3D structure of a we use LoD data. Level of detail data (LoD) is a common format to distribute 3D building data of different levels of detail over large areas. LoD1 data is the lowest resolution where only the floor outline is represented in 3D space with a rectangular box as an estimation of the buildings height. LoD2 data has the same floor outline as LoD1 but the buildings shape is represented by sets of polygons. Each polygon is labeled by its type, for example wall surface or roof surface. LoD3 data has additional detail by adding elements like chimneys, balconies

and window placements. The most commonly provided type is LoD2 where the data is mostly generated from the governments floor plans and 3D terrain data[2]. The roof shapes are best matched using 3D point cloud data from LiDAR scans. Special buildings and landmarks are manually updated over time. LoD1 data is generated as a lower detailed version from the LoD2 data. If LoD1 and LoD2 data is available it is distributed by the federal states in 1 km by 1 km sections. North Rhine-Westphalia publicly offers LoD2 data for most of its area[2]. The accuracy of the vertices is approximately 10 centimeters and roof surfaces are accurate enough to match the shape of all major surfaces. The data format used to store LoD data is cityGML.

2.1.2 LiDAR Data

For accurate irradiance simulations, high-detailed environment data is needed that are not included in the LoD2 data. Small details on roof surfaces can block potential solar panels for both irradiance and placement. In addition, information on the direct environment around a building is crucial to an accurate irradiance profile. Detailed spatial information can be stored in high-resolution point clouds representing objects in 3D space. LiDAR, or Light Detection and Ranging, uses lasers to measure distance to objects. By emitting a pulsed laser that is reflected by the object, the return time can be measured and the distance to the object can be calculated. With airborne laser scanning (ALS) a point cloud around an object is created by taking many measurements and combining them into a dataset. In Germany LiDAR data for environments is provided by the federal states in 1 km by 1 km sections. North Rhine-Westphalia publicly provides 3D point clouds for most of its area with an accuracy of approximately 10 centimeters[1].

2.1.3 Aerial Imagery

Aerial Imagery is used in our software to enhance the LOD2 data. LOD2 data only contains the main roof surfaces of a building. Details like chimneys, windows, satellite dishes or other fixtures are not included. Almost all buildings have some sort of fixture on its roof that can obstruct the placement of solar panels or cast shadows on potential solar panels and therefore decrease its efficiency. To gain that critical information we can look at aerial imagery of our target building and extract potential obstacles visually. For that we use the ”InVeKoS Digitale Orthophotos” dataset. This set of photos is a collection of different aerial photography flights where high resolution images of the ground were captured from the ground. The dataset has a resolution of approximately 20 centimeters per pixel and is a distortion free, true-to-scale photographic representation of the ground surface. The dataset is divided into 1 kilometer by 1 kilometer squares. Availability varies state by state, for the state of North Rhine-Westphalia most of the area is included in the dataset and is updated every year since 2023[3].

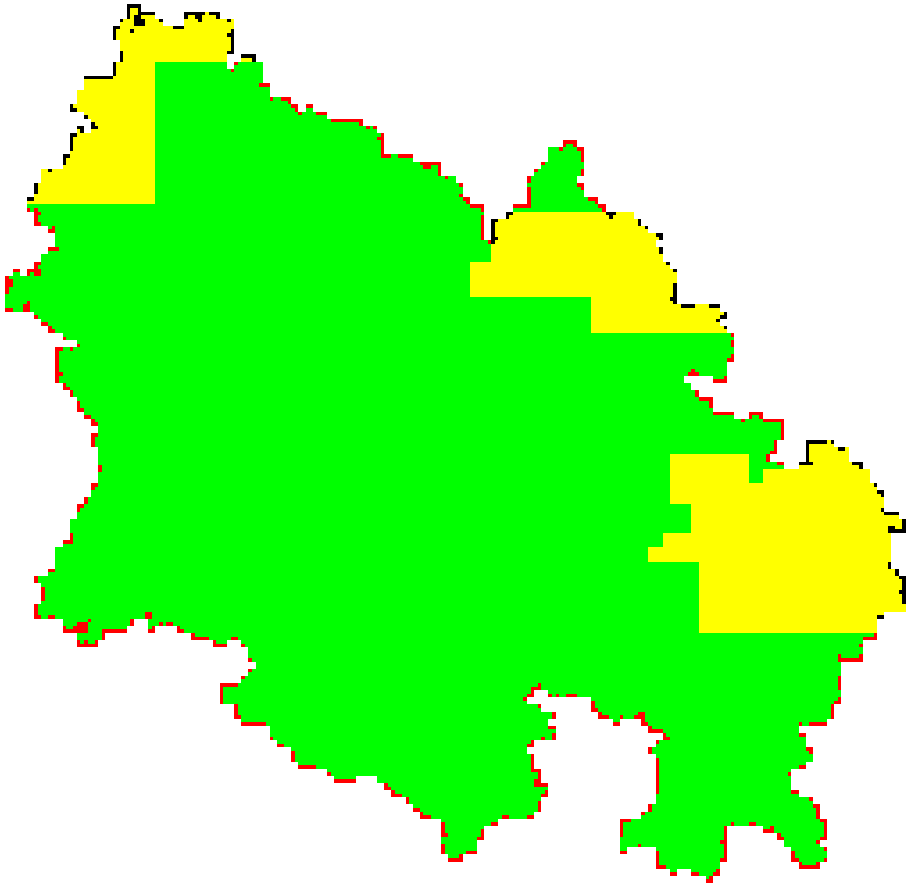

Figure 1: Grid map illustrating the availability of data types in North Rhine Westphalia in 1 km by 1 km sections: green=LoD2+LiDAR+Aerial, yellow=LoD2+LiDAR, red=LiDAR+Aerial, black=LiDAR [10]

Figure 1 displays the state of North Rhine Westphalia (NRW) in 1 km by 1 km large section. Each section is labeled with a color representing the availability of LoD2, LiDAR and Aerial imagery. Sections in green provide all three datasets, sections in yellow only provide LoD2 and LiDAR data, red sections only provide LiDAR and Aerial image data and black sections only provide LiDAR data.

2.1.4 Weather Data

The irradiance simulation tool we use is a clear sky simulator. Weather events like clouds, rain and fog are not reflected in the irradiance simulation. Since the weather has a major impact on how much irradiance takes place on the ground we have to gather weather information to adjust our simulation results. As the source for the weather data we use the Photovoltaic Geographical Information System (PVGIS)[7]. PVGIS is a tool developed by the European Commission's Joint Research Center (JRC) that provides worldwide information about solar irradiance and environmental factors. The data set we use is the Typical Meteorological Year (TMY) data for our target location.

TMY data represents a typical year regarding solar irradiance and temperature for each hour of a year. The TMY dataset is created by averaging data points from the years 2005 to 2023. The bases of the collected data points are the measurements taken by satellites in geostationary orbit around the earth [8]. PVGIS provides an open access web API to download their datasets based on location. The two data points we are interested in are Global Horizontal Irradiance (GHI) data, and temperature data. GHI data represents how much solar irradiance hits a horizontal surface on the ground for a given hour in the year and location. Included in that value are effects by weather that mitigate the irradiance on the ground. We can use that data to adjust our clear sky simulation for weather events. Temperature data has two different use cases for our environmental profile. Firstly, the temperature of a solar panel effects how efficient it is in converting solar irradiance into useful energy. Including the aerial temperature of the target structure enables to extract the panel temperature from the environmental profile based on aerial temperature, solar irradiance, panel type and usage.
For flat-plate collectors in combination with a warm water storage system are also effected in their efficiency by the ambient temperature. The temperature profile can also be used to estimate the energy requirements for heating a structure and help optimize the flat-plate and storage requirements.

2.2 Software Architecture

For the Software architecture a modular structure was chosen to offer an adaptable and clear code base. As the main programming language we chose Python. Python offers a very human readable syntax for large software projects. Pythons ecosystem offers a vast amount of libraries that enable to perform more complex solutions in an efficient manner. The computationally intensive part in our software is done by the given PCSRT. This tool was implemented in Rust and is therefore computationally efficient.

2.2.1 Voxel-Based Irradiance Simulator

For the generation of our environment profile we need to combine multiple data sources that come in a multitude of data formats. A voxel (volume pixel) based simulation is a good tradeoff to combine our LoD2 data (polygons) and LiDAR data (3D point cloud) into a detailed representation of the environments shape without sacrificing any detail. A voxel-based simulation interprets 3D point clouds as a collection of cubical volumes with equal-length sides arranged on a regular grid in 3D space. All data formats get converted into their representation of a 3D point cloud. The final representation of our environment is the merged collection of all point cloud representations.
For the irradiance simulation we use the Point Cloud Solar Radiation Tool (PCSRT) [14]. PCSRT is a voxel-based irradiance simulator based on the clear sky model of the European Solar Radiation Atlas (ESRA). With a 3D point cloud as its input the tool simulates the irradiance of every voxel for a given time period and location. It was implemented in the Rust programming language emphasizing speed and efficiency.

It relies on parallelization to simulate multiple steps simultaneously on the CPU and maximize hardware utilization. Large simulations for environments greater than the memory capacity can be split up into blocks to be processed in sequence.

In order to generate close to real world irradiance values the tool accepts a multitude of parameters that determine physical properties of the environment and computational requirements for the simulation:

Centroid The Centroid, consisting of Latitude, longitude and elevation, represents the central location of the target area in the LiDAR point cloud. Using the centroid the tool can calculate the earth's and sun's positions and rotations for a specific time. From that it can calculate the angle of the sun's rays to the ground. With the sun's position and the targets location period during the night can be skipped by calculating whether the sun is above or below the horizon which eliminates around 50% of the computational load.

Voxel Size In the simulation tool the 3D point cloud gets subdivided into cubical voxels. The voxel-size describes the side length of a voxel and determines how many LiDAR points get bundled into a voxel. All LiDAR points in a voxel are bundled together and will have the same irradiance after the simulation. A voxel containing LiDAR points casts a shadow on other voxels and deflects some amount of diffuse irradiance. Voxels are not represented by a cubical shape but have a normal vector that effects how the voxel interacts with light. A voxel must contain at least four points to determine a normal vector on its own. If there are less than four points, surrounding voxels are searched for points with a distance of up to five voxels. If not enough points are in the area the voxel orientation is considered horizontal. A smaller voxel size means a finer simulation which takes longer to simulate. [14] A tradeoff of voxel based simulations is that flat surfaces are not easily representable in voxel space. Accurate representation require high resolution simulations.

Segment Size The PCSRT simulation tool simulates over given 3D point cloud and a defined time period. The time period gets divided in multiple time segments of equal duration. The duration of those segments is defined by the segment size which represents the temporal resolution of the simulation. What segment size is used has a major impact on the accuracy and meaningfulness of the simulation result. Generally smaller segment durations result in more iterations and therefore a more accurate result. Too large segment durations are not only inaccurate but do not really represent the simulated time period at all. For example a simulation over the duration of a year with a segment size of 24 hours will only simulate a single step per day at midnight. This would result in all voxels to have close to zero irradiance despite the time period being an entire year.

Linke-Turbidity factor The Link-Turbidity factor is a parameter that reflects how much of the sunlight gets absorbed or scattered by the atmosphere. Factors like pol-

lution and water particles in the air decrease the amount of radiation that reaches the ground.

The direct normal irradiance at the ground I_{bn} is given by:

$$I_{bn} = I_0 \cdot \exp(-0.8662\, T_L\, m\, \delta)$$

I_0 represents the extraterrestrial solar irradiance corrected for Earth-Sun distance, T_L is the linke turbidity factor, m the relative optical air mass and δ the altitude dependent correction. [16] The formula shows that higher values for T_L, which represents the amount of particles in the air, lead to higher attenuation and therefore a lower irradiance at the ground.

The PCSRT tool enables to define the linke-turbidity factor for either the entire simulation or in monthly intervals.

2.3 Libraries Used

In our software we took advantage of multiple python libraries the following listing represents the most relevant ones used:

laspy: Laspy[11] is an open source python library for reading and writing LiDAR datasets in the LAS/LAZ file format. LAS files allow to store 3D point clouds in combination with custom data fields for points (for example type, building index, surface index). LAZ files are the lossless compressed version of LAS files.

numpy: Numpy[13] is an open source python library for scientific computing. We take advantage of its advanced array capabilities that enables to handle 3D point clouds as array objects.

shapely: Shapely[17] is an open source python library for handling planar geometric objects. We use its vectorized functionality to perform geometric test on many points.

3 Methodology and Implementation

3.1 Point cloud Generation from LoD2 data

LiDAR data has many advantages as it contains a lot of real world detail with a usual resolution of 10 cm. But it has some limitations due to how the data is collected. As environmental LiDAR data is collected by airborne laser scanners, there are parts of the environment that are not visible from the air, due to the planes altitude or flight plan. Partial LiDAR representations cause inaccurate irradiance simulations. An ideal representation of a 3D objects as a point cloud, is a regular spaced sampling of the entire surface. A voxel-based simulator can not distinguish between the outside and inside of a surface represented by a point cloud. That means that a LiDAR scan of a

building with missing elements can allow light to shine inside of the object and make the simulator register wrong irradiance events. In order to avoid incomplete surface sampling from LiDAR scans, we can rely more on LoD2 data to generate our own representation of the objects surface. This can be archived by either replacing all LiDAR points on surfaces or detecting surfaces with insufficient LiDAR sampling.
LiDAR scans have a fixed resolution that might not be optimal for every use case. Accurate placement for solar panels might require higher precision than the given LiDAR data can provide. If the goal is to simulate over large areas computational cost has to be considered, which depends strongly on the resolution of the LiDAR data. Filtering and interpolation can mitigate these problems in some cases but the control over the result and the usefulness of the new data is limited. Since the LoD2 data also contains a type classification for each surface, a mix of real and generated data can be created. Such a mixed approach might be generated points for all roof surfaces, since the placement of the solar panels depend on the LoD2 roof shapes, and real LiDAR points around to get the accuracy of real LiDAR data for shadows on potential panel places. The result is a 3D point representation of the environment with exactly the accuracy as needed that can be used in the voxel-based irradiance simulator.

3.2 LoD3 from Aerial Imagery

In previous work Maqsood developed a tool to detect obstacles on roof surfaces using satellite images[12]. The tool uses a neural network that classifies potential obstacles on roof surfaces. We can use this capability and combine it with geometry information to elevate the LoD2 roof surface information to a higher degree. We refine the two dimensional output in these two steps:

Filtering: Detected obstacles are checked against the LoD2 geometry, in order to remove false positives not on actual roof surfaces.

3D Projection: The remaining 2D regions are projected onto the corresponding roof surfaces of the LoD2 geometry. Their height is estimated from LiDAR points located above the region.

The result is a first step in the direction of LoD3 data by analyzing LoD2 and aerial images using image detection.

3.3 Simulating an Environment

In this section we describe the process creating a 3D solar environmental profile. Each section represents a key step in the tools workflow. For each step we will describe what data is used and how it is transformed.

3.3.1 Setup

For the input a coordinate in latitude and longitude format is given as the target buildings location. The area around that location will be the used to create an environmental profile. In this example we assume that LiDAR, LoD2 and Aerial images are all available. Since those datasets are not available for every federal states and there are sections where not all three types are present the full scope of this process can not always be achieved.

Data type	**When not available**
LoD2	No profiling possible
LiDAR	Simulation missing detail and obstacles from the environment
Aerial images	Simulation missing some detail on roof surfaces and obstacle classification

Table 1: Impact of missing data sources on simulation quality.

Table 1 displays the impact of missing datasets on our process. If LoD2 data is not available, no ground truth for the shape of the structure can be established. Therefore, no profiling of the environment can take place and no profile can be created. Missing LiDAR data has a major impact on the level of realism to the irradiance simulation as all objects in direct proximity to potential panel positions are missing and can therefore cast no shadows. If no aerial images are available, a higher level of understanding of the environment than LoD2 can not take place. In most cases this will limit the effectiveness of excluding obstructed sections on roof surfaces from the set of possible panel placements.

3.3.2 Geometry Data Download and Conversion

Firstly we have to gather all datasets representing our target structure and its environment in 3D space. With the input coordinates the tool converts the Lat, Lon coordinates format into the UTM format used in the environments datasets using the pyproj library, an open source MIT licensed python library for cartographic projections and coordinate transformations [6]. All three data types (LiDAR, LoD2, Aerial Images) are, when available aligned in 1 km by 1 km squares. The tool selects the area that includes the target building. LoD2 data is provided in the cityGML dataformat. CityGML is an XML based file format that stores the geometry information of the LoD2 data. The data is structured by building, where each building has a set of surfaces. Each surface is labeled with its type (roof surface, wall, ground). A surface points to its vertices on a global list of vertices present in the file. We convert the cityGML file into CityJSON format using CityGMLTools, an open source Java tool for processing cityGML files [4]. CityJSON is a more modern version of the cityGML file format. It is easier to work with and very flexible to be extended for our special use cases. For all three types it has to be checked if the target lies close or directly on the line between multiple segments. Is that the case, neighboring datasets have to be downloaded as well and be merged together.

3.3.3 Filter Data

To reduce the computational resources of our simulation we define a radius d that represents the maximal distance of an object n the x-y direction can have to be included in the simulation. For LoD2 data we check for every surface if it is within the defined radius. If a building has at least one surface included then all of its surfaces are included. For LiDAR data we calculate the distance to the center for every point and filter out all those with distances greater than d. We avoid per-point loops by utilizing NumPy's vectorized operations and masks. We also calculate the minimum z-coordinate of all surfaces in the target building. This gives us the needed elevation in the centroids coordinate.

3.3.4 Classify LiDAR Points

Ideally we want to know what type of environment each LiDAR-point represents, for example roof, wall, tree, or obstacle. Since the LiDAR datasets do not contain such information, we have to derive an estimation of that classification from the position of a LiDAR point relative to the 3D geometry of the LoD2 data. The LoD2 data is based partially on the LiDAR data to approximate the best shape of the roof surface, together with other information of the building and manual edits. Therefore, the distance of a LiDAR point to a roof surface is a good metric to classify its type. We want three types of classification: roof surface, obstacle and environment. Roof surface points are points that lie directly on the roof surface and contain the irradiance data needed for panel placement. Obstacle points are points that can be an obstacle to panel placement (chimney, satellite dish, etc.). Environment points are points that are around the building and can therefore cast shadows on roof surfaces.

We classify all points closer than a threshold of 20 cm to a roof surface as roof points. Points that are further away but that lie on top of roof surfaces are obstacle points. We determine those by projecting all non roof points into roof surfaces on the (0,0,-1) vector. All points that are not roof or obstacle points are environment points. 20 cm is used as the threshold as the LoD2 dataset is build by fitting an optimal shape to the LiDAR dataset. 20 cm is a typical mounting distance for solar panels on roof surfaces, as thresholds greater than 20 cm would remove valuable detail from the dataset by labeling potential obstacles as roof surface. Thresholds smaller than 20 cm are too close to the margin of error withing LiDAR data sets which have a resolution of at most 10 cm.

Figure 2: Point classification example: green=roof surface, red=obstacle, yellow=environment

Figure 2 displays an example of the LiDAR point classification based on LoD2 geometry. Points displayed in green represent points directly on roof surfaces. Red points are obstacle points that can obstruct the placement of panels. Yellow points are environmental points representing objects that might cast shadows on panels.

3.3.5 Generate Surfaces

For every surface of all buildings we generate a 3D point cloud representation by spanning a grid over its polygon. Given a polygon p with vertices v_i we calculate the normal vector n. Then we transform the vertices so that they lie flat on the x-y plane directly next to the origin point in positive x and y direction. Within the bounding box a regular grid of points is generated with a grid spacing of g. For all points, it is checked if it lies within the polygon or outside. All outside of it are deleted. For this inside-outside check we utilize Shapely's[17] vectorized functions for performing this test. Using vectorized functions instead of looping over all points allows us to

handle much larger or detailed surfaces, with potentially millions of points. The points withing are transformed in the inverse of the initial transformation of the vertices. The result is a 3D set of points in the shape of p located at its position with resolution g.

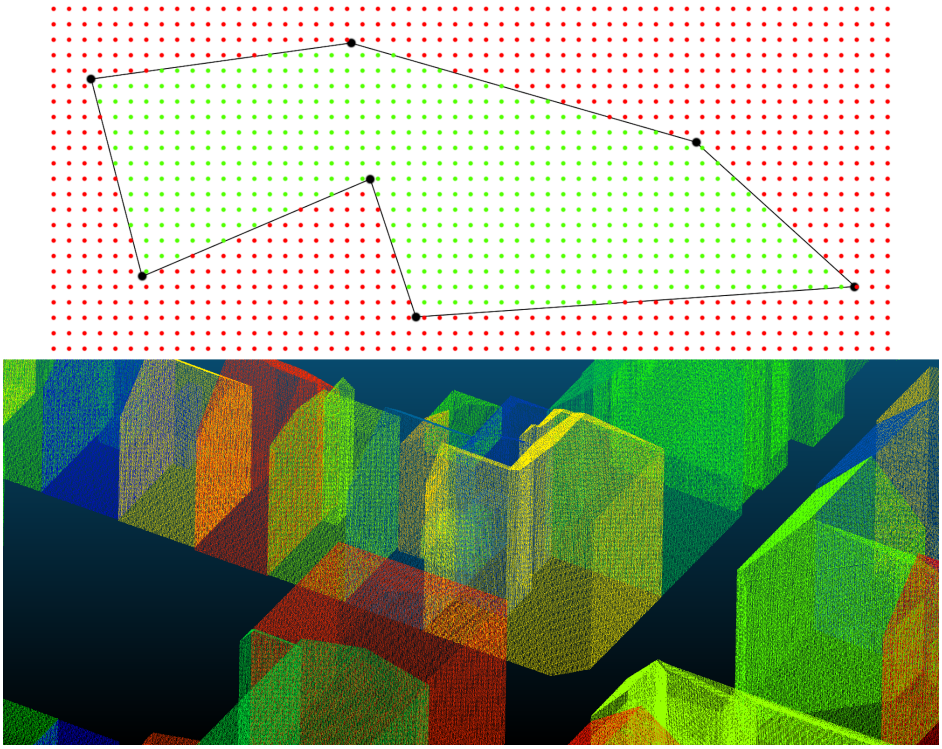

Figure 3: (Top) point sampling selection example. (Bottom) generated environment visualization

Figure 3 displays the surface point generation visually. The top half displays an example polygon with vertices represented as black dots. The edges of the polygon are represented by black lines. Red dots represent generated points outside of the polygon, green dots points inside the polygon. The bottom half of the figure displays the combined point cloud of all generated surface points, visualized in the software CloudCompare[5]. CloudCompare is a free and open source tool to visualize and interact 3D point cloud data. The coloring of a point indicates its building association. At this stage it is also possible to insert any manual requirements to the simulation. For example to analyze the potential of a balcony power plant, one would insert a surface representing its mounting position into the set of surfaces and label it accordingly.

3.3.6 Detect Obstacles

We use an image detection tool developed in Maqsood master thesis [12] to classify potential obstacles on roof surfaces. The tools input is a 250 by 250 resolution image. We crop the download aerial image based on the targets coordinates so it is centered in the input data. The tool uses a neural network that can classify each pixel as

either roof or non-roof. The tool then detects non-roof regions withing roof sections and approximates each with an axis aligned rectangle. Each rectangle is a potential obstacle on the roof surface (for example chimney window, satellite dish).

Figure 4: Image-based obstacle detection example: Aerial image with detected obstacles marked as red rectangles

Figure 4 visualizes the output of the image detection tool from the NRW aerial imagery dataset[3]. On the right side it shows the cropped section of the aerial image at the target location. Overlaid in red are the 2D rectangular bounding boxes of potential obstacles on roof surfaces.

We use that 2D information and enhance and filter it using LoD2 and LiDAR data. Firstly we transform each 2D rectangle into its corresponding position in 3D so that it lies above all buildings. Then we project the rectangles center point onto all roof surfaces in the area facing $(0, 0, -1)$ direction. If a projection exists we take that point as the new center of the rectangle in 3D space. If none exists we discard the rectangle and assume it was detected incorrectly. For all LiDAR points we take the subset that is directly above the rectangle. From that subset we calculate the maximum z-coordinate z_{max} and calculate the distance h between the rectangles center and z_{max}. The rectangle

gets extended by the height h forming a box. We now have an approximation of the area of an obstacle on the targets roof surfaces.

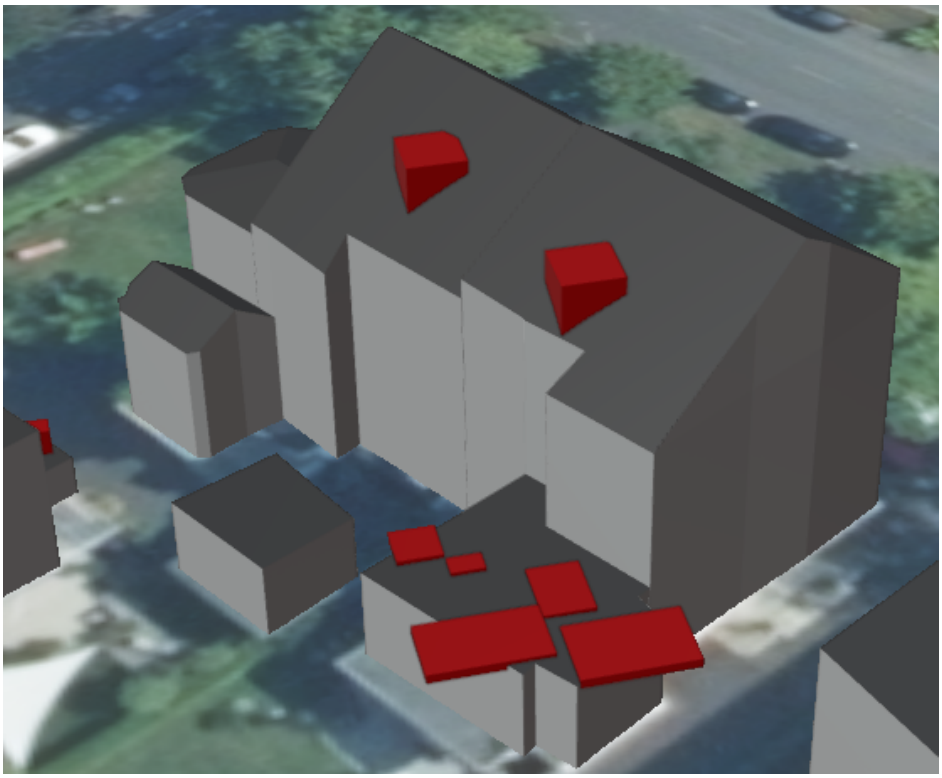

Figure 5: Detected obstacles filtering and enhancement example: 3D display of LoD2 data and obstacle location

Figure 5 shows a 3D representation of the enhanced obstacle detection method. In the scene the ground is representing the reference section of the aerial image. On top is a 3D model generated from the LoD2 dataset, with roof surfaces colored in dark gray. The red boxes on roof surfaces are the filtered rectangles in their enhanced placement in 3D.

In scenarios where no LiDAR data is available we can use the detected obstacles to generate details on roof surfaces instead. Each surface of the detected box gets added to the list of surfaces to generate a 3D point cloud representation.

3.3.7 Merge Data

In this step we merge all point clouds into a single file that will be used for simulation. In the best case we select all point clouds that are not roof surfaces from the LiDAR data. This dataset gives us high resolution information about obstacles directly on roof surfaces and a detailed representation of the environment that can reflect sunlight or cast shadows on potential panel areas.
From the generated surface data we include every surface. We need data as close

as possible to our ground truth representation in LoD2 as possible to analyze the simulation results. Generated surfaces are more accurate in placement and resolution than real LiDAR data.

3.3.8 Linke-Turbidity Factor

The linke-turbidity factor has no impact on performance but is nevertheless an important factor for close to real world accuracy. As this setting represents a data point based on real world data we can simply choose the linke-turbidity factor for each month from an available dataset. As the source data we use the European Solar radiation Data (SoDa) dataset that provides linked turbidity factors worldwide in monthly and yearly intervals.[18]

3.3.9 Temperature Data

Temperature data is an important data point to determine a solar panel and storage combinations efficiency. To include the hourly aerial temperature data in our profile we download the TMY data set for our location from the PVGIS website using their API. The data set is available for download in JSON format and can be easily stored in the cityJSON file.

3.3.10 Simulate Yearly Irradiance

The yearly irradiance profile of a building can be used to determine the best placement for solar panels with the highest irradiance capture per panel. Using the merged 3D point cloud we simulate the total irradiance of the area over the time period of 1 year using the PCSRT simulation tool. From the simulation result we can extract the $W.h/m^2$ per point in our point cloud. Since the dataset includes classification for building and surface index we can immediately assign each data point to a specific surface at a specific location in our LoD2 data.

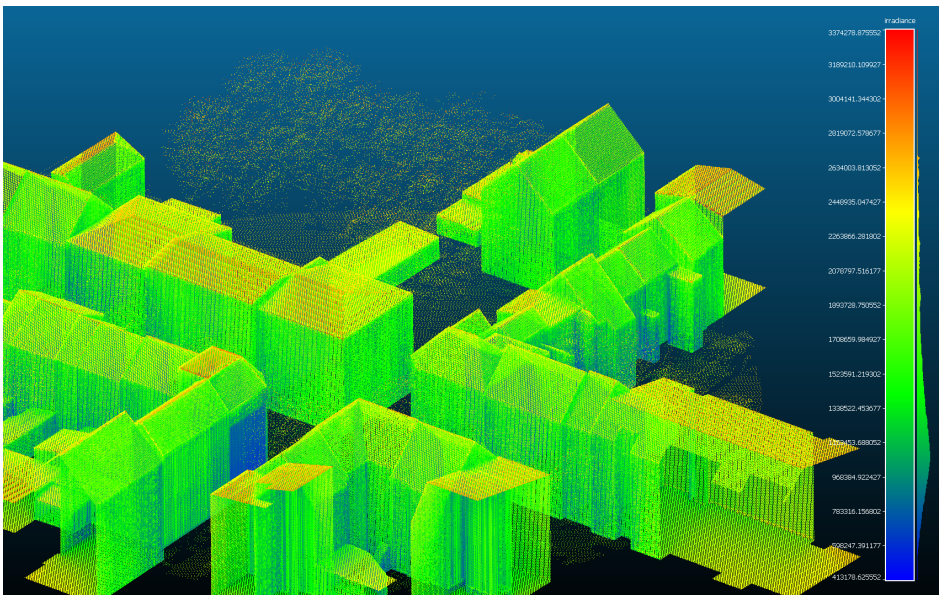

Figure 6: Yearly irradiance on a target structure visualized in CloudCompare[5]

Figure 6 displays an exemplary yearly irradiance simulation result displayed in CloudCompare[5]. The graphic shows the mixed environment based on LiDAR for the environment and generated data for surfaces. The color of a point represents its yearly irradiance in Wh/m^2 according to the range on the right. This example is not adjusted for weather data and therefore not representative of real-world irradiance values, but the irradiance values relative to each other are correct.

3.3.11 Simulate Hourly Irradiance

On the same merged dataset as for the yearly simulation we run multiple simulations representing 1 hour long intervals over the period of 1 year. The result is a list of irradiance values for each point in our data set.

3.3.12 Adjusting Clear Sky Data

To get irradiance data closer to real world we adjust our clear sky simulation results using Global Horizontal irradiation (GHI) data. GHI is the total irradiance of a horizontal plane at a given location over a time period measured with the influence of real world weather events. For our target's location we download the TMY data from PVGIS using their API service. From the available data we can read out the GHI and clear sky GHI values. Then we generate a new 3D point cloud environment in the same location as our target but only with a single horizontal surface. No other points but that one surface are present. Then we simulate the irradiance for that surface in the same intervals as the yearly and hourly simulations. From the resulting irradiance simulations we calculate the mean of all irradiance values over all points for each time interval t. That irradiance value $GHI_{cs}(t)$ represents the clear sky GHI value for our

simulation over the time period t.
The normalization factor $N(t)$ at time t is calculated as the ratio of the real-world Global Horizontal Irradiance ($GHI_{real}(t)$) from the dataset to the clear sky Global Horizontal Irradiance ($GHI_{cs}(t)$) from our simulations:

$$N(t) = \frac{GHI_{real}(t)}{GHI_{cs}(t)} \tag{1}$$

The adjusted irradiance at a point p at time t, denoted $I_p^{adj}(t)$, is computed by scaling the original irradiance $I_p(t)$ by the normalization factor $N(t)$:

$$I_p^{adj}(t) = I_p(t) \cdot N(t) \tag{2}$$

In our simulation results we scale every points irradiance data in the time period t with the normalization factor $N(t)$.

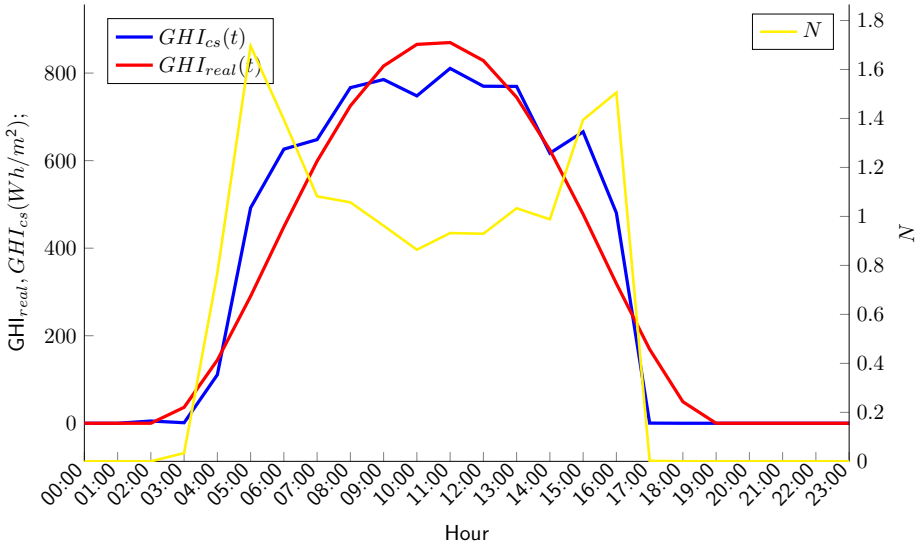

Figure 7: Hourly values of $GHI_{real}(t)$, $GHI_{cs}(t)$ and normalization factor N of a sample location over the period of a day

Figure 7 displays the hourly $GHI_{real}(t)$ and $GHI_{cs}(t)$ and the normalization factor of a sample location over the course of a 24 hour time period. The x-axis represents the times of the hourly simulations ranging from $00:00$ to $24:00$. The left y-axis represents the irradiance at the location in Wh/m^2. The right y-axis represents the normalization factor N. The $GHI_{real}(t)$ represented by the red line has the shape of a bell curve, peaking at $11:00$ with an irradiance of $810 Wh/m^2$ and 0 irradiance before $2:00$ and after $17:00$. The $GHI_{cs}(t)$ represented by the blue line has a shape approximating a bell curve, peaking at $11:00$ with and irradiance of $870 Wh/m^2$. The normalization factor represented by the yellow line is at 0 for the time periods $00:00 - 02:00$ and $18:00 - 23:00$. In the time period of $07:00 - 14:00$ it lies

around 1 and around 1.5 in the periods of $05:00-06:00$ and $15:00-16:00$. At $04:00$ it shows a normalization factor of 0.76. Using the normalization values as represented by an example time period and location in Figure 7 all points will have their irradiance values adjusted to account for weather influences.

3.3.13 Result

All of the downloaded, generated and simulated data gets inserted into a CityJSON file. The file contain the following data point:

LoD2: All buildings in the filter radius around the target are stored as 3D geometry with labeled surface types.

LiDAR obstacles: All obstacle points, which are points from the LiDAR dataset that could potentially obstruct the placement of a solar panel, are stored for each surface they are obstructing.

Detected obstacles: The 3D enhanced obstacle detection results based on image detection get stored as their position and dimension, linked to their assigned surface.

3D roof surface points: All points generated on surfaces get stored for each surface in the CityJSON file with their irradiance values for yearly and hourly simulations. Each point P is stored as a tuple in this format:

$$P = ((x, y, z), i_y, [i_t], [t])$$

where (x, y, z) are the 3D coordinates, i_y is the yearly irradiance (in $\mathrm{Wh/m^2}$), $[i_t]$ is the list of irradiance values for time segment t in $\mathrm{Wh/m^2}$, $[t]$ is the list of time segments.

Temperature profile: For each hour of a year the aerial temperature is listed.

Metadata: Information on location, date of creation and the processes parameters are stored.

4 Evaluation and Discussion

In this section the accuracy and performance of the described simulation workflow are systematically evaluated. The voxel based simulation tool and the individual steps in the workflow offer several configuration parameters. The goal is to determine the optimal settings that offer the most accurate irradiance values with the highest computational efficiency. The key settings that determine the quality and computational requirements are voxel size, generated LiDAR resolution, filter radius, segment size, weather data and normalization. All tests were performed on a system with a AMD 7950x CPU and a memory capacity of 32 Gigabytes. As memory usage was for all runs was always below 200 MB we only need to focus on computational performance represented by the total simulation time.

4.1 Voxel Size

The voxel size is the most significant parameter in our simulation workflow as it represents the spacial resolution of the simulation. The simulation is the most computationally intensive part of the software. In this test we measured how different voxel sizes impact the simulation duration compared against loss of accuracy.
The simulation period spans one year with a segment size of 30 minutes. The chosen range for the resolution is 10 cm to 1 m. 10 cm represents the highest accuracy in the LiDAR datasets and is also the resolution of the dataset used for this test. 1 m is a clear upper bound as a spacial accuracy of 1 meter is not practical anymore for our application. The duration of the simulation was measured in seconds, and the accuracy of the result was determined using the Root Mean Square Error (RMSE). For each simulated voxel size the irradiance of all points was compared against the reference simulation with the highest resolution (10 cm). For every point p in the point cloud and every voxel size the RMSE was calculated as

$$\mathrm{RMSE}_p = \sqrt{(x_p - y_p)^2}, \tag{3}$$

where y_p is the irradiance of point p in the reference simulation (voxel size of 10 cm) and x_p is the irradiance of point p in the coarser simulation. Since the $RMSE_p$ values are in the unit of Wh/m^2 over a one year time period, the absolute $RMSE_p$ values were normalized per point by their corresponding reference value, resulting in the relative Root Mean Square Error $rRMSE$,

$$\mathrm{rRMSE}_p = \frac{\mathrm{RMSE}_p}{y_p}. \tag{4}$$

As a total measure of accuracy the median $rRMSE_p$ of all points in a simulation $mRMSE$ was calculated. The median was chosen to exclude outliers with extremely low or high irradiation values.

For this test we expected higher accuracy with higher resolution simulations and shorter simulation durations with lower resolutions.

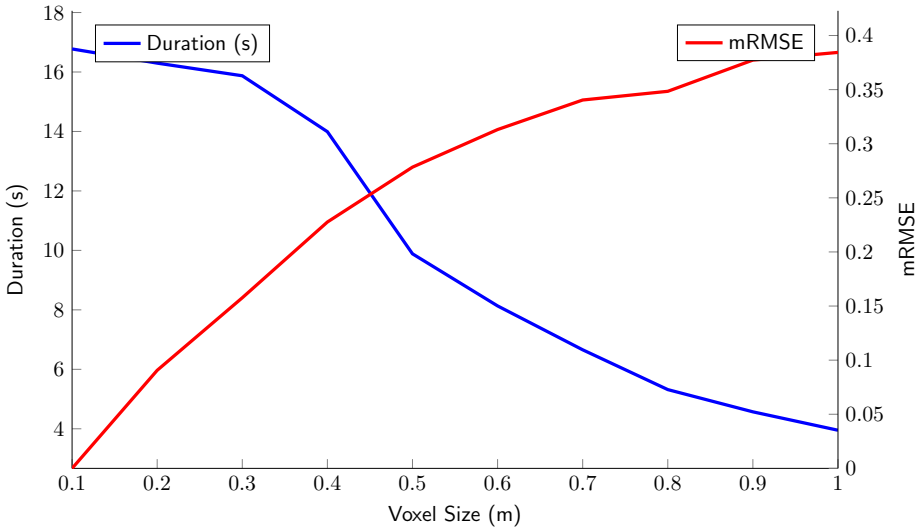

Figure 8: Plot of voxel size analysis

Figure 8 illustrates the results of our test simulations with the x-axis represents the voxel size from 10 cm to 1 meter. The left y-axis represents the duration of the simulation in seconds for a given voxel size corresponding with the blue graph "duration". It is monotonously falling with a double-curved shape from 17 seconds for the 10 cm simulation to just under 4 seconds for the 1 meter simulation. From 10 cm to 30 cm the graph is almost horizontal with simulation durations around 16 seconds. The right y-axis represents the $mRMSE$ for a given simulation corresponding to the red graph "mRMSE". The graph is monotonously increasing with higher voxel sizes and concave-up shape from 0 at voxel size 10 cm and 0.38 at 1 meter.

Our expectations for this test were met as the results clearly demonstrate a higher degree of accuracy (lower $mRMSE$) for smaller voxel sizes and a shorter simulation time for larger voxel sizes. This confirms the anticipated trade-off between accuracy and computational efficiency regarding spatial resolution in the simulation.

4.2 Generated Point Cloud Resolution

How we generate point cloud data from LoD2 data has an impact on how many points are present in the simulation. In this test, we evaluate how the number of points impacts the duration of the simulation based on the grid spacing for generated point data. As the basis for this test an environment with multiple buildings was chosen that represents a typical target for a solar radiation simulation. Only generated points were included in the environment of the simulation and no LiDAR data was used. For each grid distance from 10 cm to 1 meter, a point cloud representing LoD2 surface data was simulated over a time period of one year with a fixed voxel size of 30 cm. The simulation duration was measured in seconds.

For this test we expect longer resolution durations with smaller grid spacings.

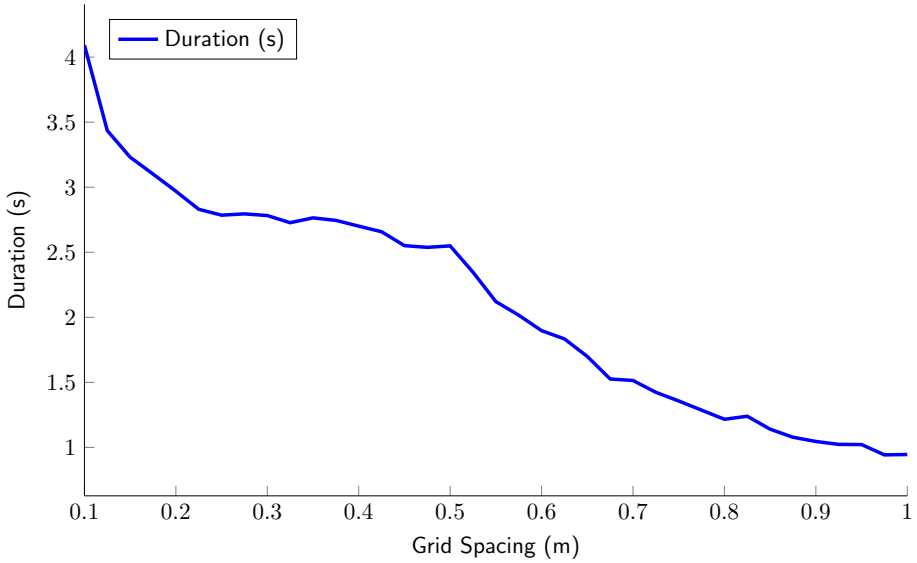

Figure 9: Plot of duration analysis for range of grid spacings

The graph in Figure 9 shows the results of the test with the x-axis representing the grid-spacing in centimeters used to generate the point clouds and the y-axis representing the duration of each simulation. The graph is falling almost monotonously with 4 seconds simulation time at 10 cm resolution and just under 1 second duration at 1 meter resolution. It follows a linear trend with a slight upwards curve around 50 cm. The results align with our expectations, demonstrating shorter simulation durations for lower resolution generated point cloud environments.

4.3 Filter Radius

The filter radius is used in the workflow purely as a means to increase the performance of the simulation. As the datasets are very large compared to the target structures we want to only simulate areas that are necessary for high accuracy. In this test we evaluate the impact of the filter radius on the accuracy of the simulation by simulating the same environment with multiple filter radii and comparing the resulting irradiance values of all roof surfaces of our target structure. Only the roof surfaces of our target were chosen as they lie in the center of the filtered area and surrounding points on the edge of the radius are much more influenced and would degrade the results. As the target building a urban environment was chosen that has many objects and building surrounding the structure. For the point cloud data only LiDAR data was included. LoD2 data was used to classify LiDAR data as roof surfaces. For each filter radius from 50 meters to 5 meters a simulation with voxel size of 30 cm over the period of one year.

The accuracy for all roof surface points was calculated using $RMSE_p$ and $mRMSE$ in the same manner as for the voxel size test but instead of calculating it for all points in the simulation only roof surface points were included as not all points are present

in all simulations.

For this test we expect the accuracy to decrease with smaller filter radii as more objects in the environment are not included in the point cloud.

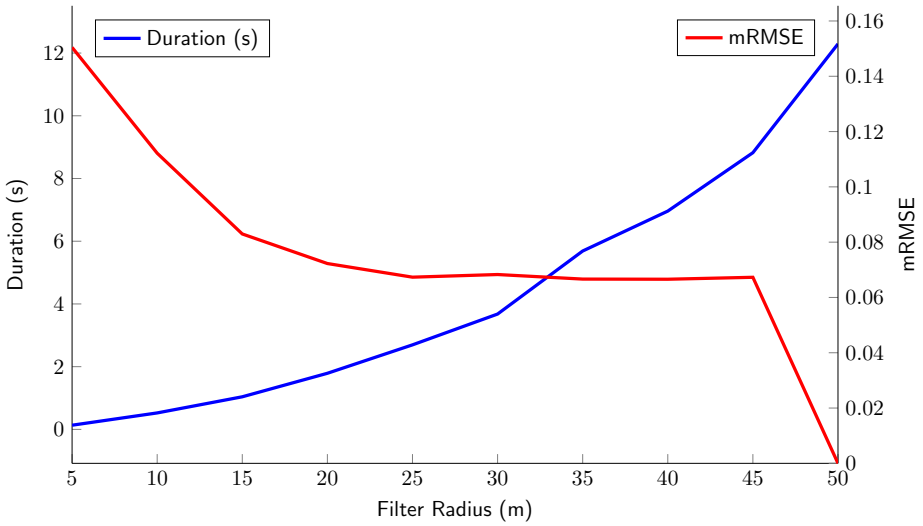

Figure 10: Plot of filter radius analysis

Figure 10 displays the results of this test with the x-axis representing the filter radius in meters from 5 to 50 meters and the left y-axis representing the $mRMSE$ value corresponding with the red line labeled "mRMSE". It is monotonously decreasing from 0.15 at 5 meters to 0 at 50 meters with a mostly flat section between 15 and 45 meters around the 0.05 error mark. At both ends it has a steep divergence from that line with the smallest filter radii having much higher and the highest filter radii having much lower $mRMSE$ values. The right y-axis represents the simulation duration in seconds corresponding with the blue line labeled "duration". It is monotonously increasing from 0.13 seconds at 5 meters to 12.3 seconds at 50 meters. It has a roughly linear shape with a slight downward curve around the 30 meters radius.

Our expectations for this test were met but with some unexpected details. The test confirmed that with smaller filter radii the accuracy decreases and the simulation duration gets shorter. Unexpected was that the $mRMSE$ value is almost constant for filter radii greater than 25 meters.

4.4 Segment Size

The segment size represents the temporal resolution for our simulations. This test was chosen to determine how the segment duration impacts the duration and accuracy of the simulation. For the test simulations we used point cloud data from LiDAR data and simulated them over the time period of one year with a voxel size of 30 cm within a filter radius of 50 meters. The segment durations ranged from 1 minute to 120 minutes.

As the measurement for accuracy the same $mRMSE$ method as in the voxel size test were chosen.

For the results we expect that smaller segment duration results in higher accuracy and longer simulation durations.

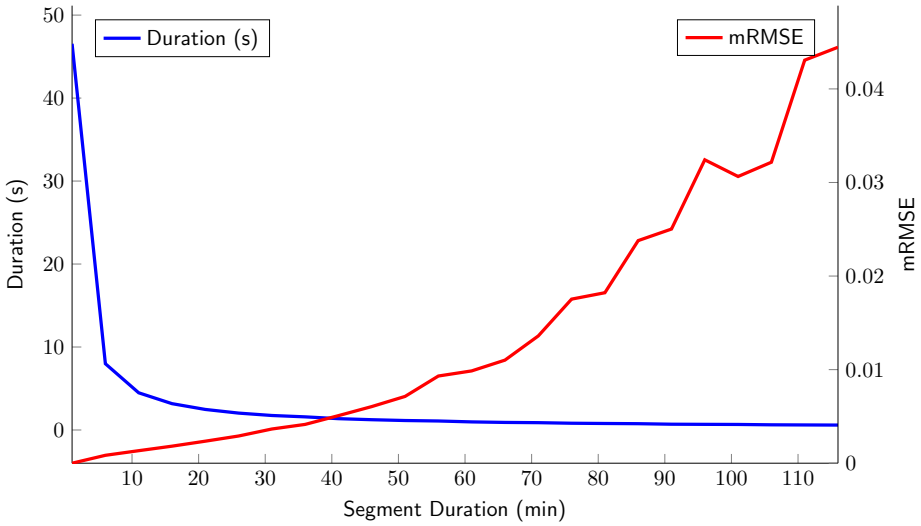

Figure 11: Plot of segment duration analysis

Figure 11 displays the results of this test with the x-axis representing the segment duration in minutes ranging from 1 minute to 120 minutes. The left-y axis represents the duration of the simulation in seconds corresponding with the blue graph labeled "duration". The duration has monotonously decreasing hyperbolic shape with a duration of 46 seconds at the segment size of 1 minute and a duration of 0.6 seconds at 120 minutes segment duration. The graph is almost linear for segment durations larger than 20 minutes. The right y-axis represents the $mRMSE$ value ranging from 0 at 1 minute segment size to 0.044 at 120 minutes segment size. The line is not monotonous but generally follows a slightly downwards facing curve. Especially for segment sizes greater than 70 minutes the fluctuations in the $mRMSE$ are significant. The result generally confirm our expectations but show some unexpected results regarding the simulation duration. Instead of following a more linear trend line the duration has extremely high simulation durations for segment durations shorter than 10 minutes.

5 Conclusion

5.1 Summary of Key Findings and Optimal Data Usage

In this section, we will briefly summarize the tests in evaluation and reason over their meaning and explain potential anomalies. The goal is to make a creditable suggestion for the optimal settings of our workflow.

5.1.1 Voxel Size

The results of the voxel-size test confirmed our expectation that larger voxel sizes correlate with lower accuracy of the irradiance and shorter simulation duration. The result of 0 $mRMSE$ at voxel size 10 cm indicated that our testing method works as intended as it calculate no error for identical sets. From a computational standpoint lower voxel sizes cause that more voxels are present in the environment which leads to more steps the simulation tool has to compute. Since the amount of voxels increases cubically with the side length of a voxel a steeper increase of the simulation duration was expected with shrinking voxel sizes. The test clearly shows that voxel sizes smaller than 40 cm do not cause a steep increase in simulation duration but only roughly linear. The cause for this trend is that as the voxel size approaches the resolution of the 3D point cloud, the maximum number of points contained in all voxels approaches zero. Voxels containing no points are considered empty space and are not computationally intensive. This means that the number of voxels that have to be simulated converges to the number of points in the point cloud. Simulating with smaller voxel sizes as the resolution of the point cloud would also not increase the number of simulated voxels, but would cause surfaces to leak light as neighboring points are represented by voxels smaller than their distance.
Given that the simulation duration between 10 cm and 30 cm voxel size remains fairly horizontal at around 16 seconds and that the error in that range spans from 0 to 0.15, we can determine that this tests indicates a optimal voxel size between 10 cm and 30 cm.

5.1.2 Generated Point Cloud Resolution

The Generated Point Cloud Resolution test confirmed our expectation that higher grid spacing results in longer simulation durations. The density N of points arranged in a regular on a surface is defined by

$$N = \frac{1}{d^2}] \tag{5}$$

where d is the grid spacing distance. The roughly linear relationship between grid spacing and simulation duration despite the inverse-square relationship between the number of points and the grid spacing suggests that there are little tradeoffs for denser surfaces where the generated points are grouped close together.

From the inner workings of the PCSRT simulation tool we know that a voxel needs at least four points in it to calculate an accurate normal vector. Voxels containing fewer points rely on neighboring voxels points to calculate their normal vector [14]. The normal vector determines what angle the surface has to the light source and concluding the amount of solar radiation hitting the surface. The surface irradiance E depending on the angle of the surface to the light source is given by

$$E = E_0 \cdot \cos(\theta) \tag{6}$$

. Here E_0 is the maximum possible irradiance where the surface is perpendicular to the light source and θ is the angle of the surface to the light source. The direct dependence on irradiance leads us to the conclusion that we want the maximum amount of accuracy regarding the angle of potential panel surfaces. In order to guarantee that every voxel with a voxel size v_d contains at least 4 points we can use this formula that calculates the required grid spacing:

$$d \leq \frac{v_d}{2}. \tag{7}$$

Solving for d we can determine that a grid spacing smaller than $v_d/2$ is required to always have optimal normal calculation on a surface. The conclusion from this test is that the grid spacing distance has less impact on simulation time an can be determined as a dependency of the voxel size.

5.1.3 Filter Radius

The results of the Filter Radius did confirm our expectation that higher filter radii correlate with higher accuracy and longer simulation times. The $mRMSE$ value of 0 at 50 meters filter radius indicate that our test setup is working as intended. The almost unchanged $mRMSE$ values for filter radii greater than 25 meters reveal that the irradiance of a buildings roof surfaces is independent from objects further away than 25 meters. Since the simulation duration increases faster than linear with greater filter radii we can conclude that a filter radius of 25 meters around the target is the best choice for an efficient and accurate simulation in our workflow.

5.1.4 Segment Size

The Segment Size test confirmed our expectation that smaller segment sizes correlate with a higher degree of accuracy. The test case where we compared the reference simulation to itself resulted in a $mRMSE$ of 0 which indicates that our test setup is correct. From a simulation standpoint it makes sense that smaller temporal resolutions generate more accurate results. The irregularities in the graph of the $mRMSE$ result for segment durations longer than 70 minutes can be explained by how shadows move. As the temporal resolution decreases small shadow events where a shadow is cast on a group of points for a short period of time are missed. As direct irradiation is effectively binary missing hose events causes irregular noise in the result.
The simulation duration result show large outliers for segment durations shorter than 10 minutes. Since the PCSRT tool is built to process individual time steps of the simulation in parallel, there is a point where the maximum amount of parallelization on a CPU is reached. In this case it seems to be the case for time segments longer than 10 minutes can be processed almost in parallel with little impact on the simulation time. Segment sizes smaller than 10 minutes cause the CPU to be at its capacity for parallel computing and therefore increase the simulation time significantly.
From this test we can conclude that simulation times should not be shorter than 10 minutes. Segment sizes of up to 60 minutes offer an $mRMSE$ value smaller than 0.01 which represents a less than 1% deviation from the reference simulation.

5.1.5 Optimal Settings

All test parameters indicated clear trends to their accuracy and performance which enables us to create the optimal settings for our workflow.

Parameter	Recommended value / range
Voxel size	10–30 cm
Grid resolution	$d \leq \frac{v_d}{2}$
Filter radius	25 m
Segment size	10 minutes

Table 2: Summary of tested parameters and recommended settings.

Table 2 shows the key information we were able to extract from our tests and their analysis. The voxel size has a clear range between 10 cm and 30 cm, where the simulation is very accurate but also computationally efficient. The test data combined with insight of how the PCSRT simulation tool works we were able to make the Grid Size dependent on the voxel size and eliminate a variable. The filter radius had a very clear test result where the optimal filter radius is around 25 meters. Results and analysis of the Segment Size test revealed a greater dependency on parallel CPU compute with a clear optimal segment duration of 10 minutes. This setting is also easily adaptable to different CPU's by running the test again and identifying the obvious outliers where the parallel computing power is saturated.
Since our profile contains yearly and hourly simulations we can differentiate the settings between them based on their use case. The yearly simulation is needed to determine the optimal placement of the panels. Installed panels have to be places optimally for the panels to be as useful as possible in an environment. For a 1 year simulation only one simulation has to be performed so simulation time can be neglected for such small environments. For that reason the smallest voxel size in the range of 10 cm should be used to create a high detail irradiance map for optimal placement. Hourly simulations are required to optimize the storage size of the PV system. A potential energy storage optimizer would need to analyze the energy usage and production during each day and find an optimal solution for the entire year. Each hour has to be simulated individually, causing a large overhead of read and write operations. This results in major inefficiencies and much longer total simulation times. Since the storage optimizer relies on accurate placements of the panels based on the detailed yearly irradiance data, the voxel size for hourly simulations can be chosen from the upper end of the range. 30 cm voxel size will reduce the total simulation time for hourly irradiance as much as possible but will result in accurate enough irradiance for a storage optimization.

Parameter	Yearly simulation	Hourly simulation
Voxel size	10 cm	30 cm
Grid resolution	5 cm	15 cm
Filter radius	25 m	25 m
Segment size	10 minutes	10 minutes

Table 3: Recommended parameters for yearly and hourly simulations.

Table 3 summarizes the optimal settings we were able to determine based on our test cases for each parameter. The optimal parameters for our workflow for yearly simulations have a voxel size of 10 cm, a grid resolution for generated points of 5 cm, a filter radius around the target of 25 meters and a segment size of 10 minutes. The optimal parameters for the hourly simulations have a voxel size of 30 cm, a grid resolution for generated points of 15 cm, a filter radius around the target of 25 meter and a segment size of 10 minutes.

5.2 Suggestions for Future Work

By implementing the 3D solar radiation profile workflow, we were able to gain some significant insights into multiple avenues of tools and datasets. In this section, we will reflect on what we learned and suggest improvements and possible use cases directly related to our work.

5.2.1 Alternative data sources

LoD2, LiDAR and aerial imagery are detailed and widely available data sets. But they are not available everywhere and do not reflect the environment perfectly. Enabling the use of other data formats such as photogrammetry and architectural formats can improve this thesis's process in detail and data source dependency. Photogrammetry is a technique that generates 3D point clouds of an environment by analyzing a large set of photos of the target. These photos can be captured using mobile phones or drones. Architects use 3D software to plan the layout of a building. These files are much more detailed than the available LoD2 data, which would result in more accurate results. It would also enable a building to be profiled before its construction, enabling the structure to be adjusted for optimal energy efficiency.

5.2.2 Improved Image-Based Detection

Image-based obstacle detection has the potential to assess all environmental datasets in high detail. Improving the demonstrated use case of rectangular two-dimensional detection by enabling obstacle classification and high detail shape matching can enhance the available LoD2 datasets greatly.

5.2.3 PCSRT Performance Optimization

The PCSRT tool was not built to provide segmented irradiance intervals. Removing the overhead of running a separate simulation for each hour in a year would decrease the simulation duration by a lot. Cutting down the computation time from potentially hours to minutes or seconds.

References

[1] Bezirksregierung Köln, Geobasis NRW. Nutzerinformationen für die 3D-Messdaten aus dem Laserscanning für NRW. Informationsblatt Stand: 02/2020, Bezirksregierung Köln, Geobasis NRW, February 2020. URL https://www.bezreg-koeln.nrw.de/system/files/media/document/file/geobasis_hm_3dm_nutzerinfo_3d-messdaten_aus_dem_laserscanning.pdf.

[2] Bezirksregierung Köln, Geobasis NRW. Nutzerinformationen zur 3D-Gebäudemodelle-Übersicht für NRW. Informationsblatt Stand: 06/2021, Bezirksregierung Köln, Geobasis NRW, June 2021. URL https://www.bezreg-koeln.nrw.de/system/files/media/document/file/geobasis_3dg_nutzerinfo_3d-gm-uebersicht.pdf.

[3] Bezirksregierung Köln, Geobasis NRW. InVeKoS Digitale Orthophotos (2-fache Kompression) – Paketierung: Einzelkacheln. Datenbeschreibung Stand: 2024, Bezirksregierung Köln, Geobasis NRW, 2024. URL https://www.opengeodata.nrw.de/produkte/geobasis/lusat/akt/idop/idop_jp2_f2/.

[4] citygml4j Contributors. citygml-tools: Collection of tools for processing citygml files. https://github.com/citygml4j/citygml-tools, September 2025. Version 2.4.0.

[5] CloudCompare Development Team. CloudCompare: 3D point cloud and mesh processing software, 2025. URL https://www.cloudcompare.org/. Open Source Software, Accessed: 2025-09-21.

[6] PyProj Developers. pyproj: Python interface to proj (cartographic projections and coordinate transformations library), September 2025. URL https://doi.org/10.5281/zenodo.2592232. Version 3.7.2.

[7] European Commission Joint Research Centre. Photovoltaic geographical information system (pvgis). https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis_en, 2025. Accessed: September 20, 2025.

[8] European Commission Joint Research Centre. Sarah-2 solar radiation data. https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis/pvgis-data-download/sarah-2-solar-radiation-data_en, 2025. Accessed: September 20, 2025.

[9] Google. *Solar API Documentation*, 2025. URL https://developers.google.com/maps/documentation/solar. Accessed: September 21, 2025.

[10] Landesamt für Natur, Umwelt und Verbraucherschutz Nordrhein-Westfalen. Geobasis nrw - geoportal database, 2025. URL https://www.opengeodata.nrw.de/produkte/geobasis/. Accessed: September 21, 2025, 04:01 PM CEST.

[11] laspy Developers. *laspy: Python library for lidar LAS/LAZ IO*, 2025. URL https://laspy.readthedocs.io/en/latest/index.html. Accessed: September 21, 2025, 05:43 PM CEST.

[12] Muhammad Hassan Maqsood. Detection of roof shapes and obstacles using stereosatellite imagery. Master's thesis, RWTH Aachen University, 2025. In progress; supervised by Erika Ábrahám and Pascal Richter.

[13] NumPy Developers. Numpy: The fundamental package for scientific computing with python, 2025. URL https://numpy.org/. Accessed: 2025-09-21.

[14] Filip Pružinec and Renata Ďuračiová. A point-cloud solar radiation tool. *Energies*, 15(19), 2022. ISSN 1996-1073. doi: 10.3390/en15197018. URL https://www.mdpi.com/1996-1073/15/19/7018.

[15] Erdzan Rastoder. Optimizing solar panel installation and assessing rooftop solar energy potential using machine learning and big data analysis. Master's thesis, RWTH Aachen University, 2024. URL https://ths.rwth-aachen.de/theses/. Accessed: September 22, 2025.

[16] Christelle Rigollier, Olivier Bauer, and Lucien Wald. On the clear sky model of the esra — european solar radiation atlas — with respect to the heliosat method. *Solar Energy*, 68(1):33–48, 2000. ISSN 0038-092X. doi: https://doi.org/10.1016/S0038-092X(99)00055-9. URL https://www.sciencedirect.com/science/article/pii/S0038092X99000559.

[17] Shapely Developers. Shapely: Manipulation and analysis of geometric objects, 2025. URL https://pypi.org/project/shapely/. Version 2.1.1, Accessed: 2025-09-21.

[18] SoDa Project. Monthly linke turbidity factor maps (2003). Dataset, 2003. URL https://www.soda-pro.com/help/general-knowledge/linke-turbidity-factor. Accessed via https://www.soda-pro.com/help/general-knowledge/linke-turbidity-factor on September 19, 2025.