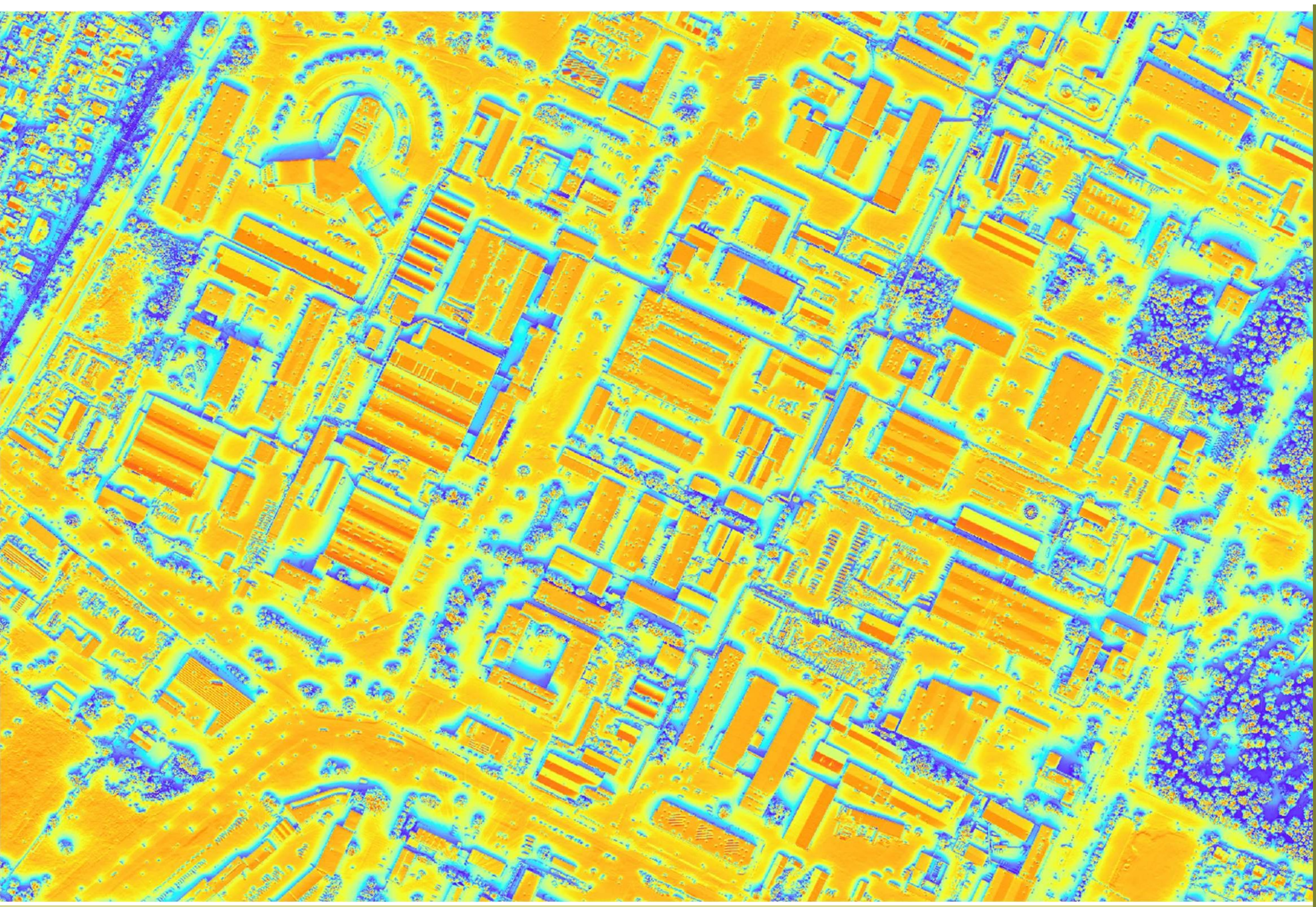
EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung



caREL - Solar potential analysis

Documentation – v1.0. 09/11/22

caREL – Computer-Aided Renewable Energy Language
Frankfurt University of Applied Sciences - Laboratory for Geoinformation

TABLE OF CONTENTS

1. INTRODUCTION

The presented algorithm can be used to identify the solar potential of areas. LiDAR point clouds from which digital surface models are calculated serve as the basis for the analysis. Based on these models, the incoming global radiation is calculated for each grid cell. The LiDAR data are first processed for a job database using LASTools. Then a Python script processes these jobs and triggers the necessary calculations, carried out with ArcGIS Pro. Global radiation rasters are created for the individual jobs, which are then merged into a raster for the entire analysis area. In addition, the algorithm calculates alignment and slope grids. In the course of the caREL project, this calculation was carried out in five pilot states in 11 analysis areas. This documentation describes the necessary software components, the functioning of the algorithm and the data basis.

2. SOFTWARE COMPONENTS

In the solar potential analysis, different software components are used for different purposes. This chapter gives an overview of the software used and its field of application.

Python is a portable, object-oriented programming language with a very clear syntax and interfaces to many system calls and libraries. Python can be run on Linux, MacOS and Windows systems, among others. Accordingly, Python is excellently suited as an extension language for a wide range of applications. All Python versions are open source, usually also GPL (General Public License) compatible and are administered and further developed by the Python Software Foundation (PSF). For the solar potential analysis, a Python script based on version 3.7.10 is used to automate the solar potential analysis.

PostgreSQL is an object-relational open-source database system, developed and managed by the PostgreSQL Global Development Group. Various extensions are available for PostgreSQL, including PostGIS, which enables PostgreSQL to be used as a spatial database for geographical information systems. For the solar potential analysis, a PostgreSQL database in version 10.19, with the PostGIS extension in version 2.4.3, is used for the job management. The graphical open source administration tool pgAdmin 4 in version 4.19 is used for database access. The use of newer versions should not be a problem.

For communication between the database and the Python script, the extension psycopg is used. Psycopq is a PostgreSQL adapter released under the GNU Lesser General Public License and allows the use of PostgreSQL features within the Python script.

Both QGIS and ArcGIS Pro are used as geographic information systems. ArcGIS Pro, in its version 2.6.2, is used for the calculation of solar irradiance, alignment and tilt. In addition, the ArcGIS site package ArcPy is used. ArcPy provides native Python functions and allows geographic data analysis to be performed using Python. QGIS is a free open-source alternative to the various ESRI ArcGIS products. QGIS, version 3.22.2, is used to transfer the extent of individual LiDAR datasets to the PostgreSQL database via the database management function.

For the analysis and processing of the LiDAR data, the paid software LAStools from the company rappidlasso is used. The LAStools software suite offers a collection of highly efficient, batch-scriptable, multi-core command line tools, among others for classifying, tiling, converting and polygonising LiDAR data. In addition, the various tools can be used via a native GUI or as a toolbox in ArcGIS and QGIS. In the solar potential analysis, the LAStools were used at various points for calculating the boundaries of LiDAR datasets, calculating digital surface models and analysing and merging multiple LiDAR datasets, among others. The following tools were used:

- lasinfo, for the analysis of LiDAR datasets.
- lasboundary, for calculating the extent of single LiDAR datasets.
- lasmerge, for merging multiple datasets.
- las2dem, for calculating digital surface models.

The LAStools process LiDAR data in .LAS format as well as data in the compressed .LAZ format. Due to the lower memory requirements of the .LAZ format, this is used for solar potential analysis.

3. PROJECT AREAS

Five areas in the following EU countries were selected for the analysis of solar potential areas:

- Belgium
- Denmark
- Estonia
- Slovenia
- Spain

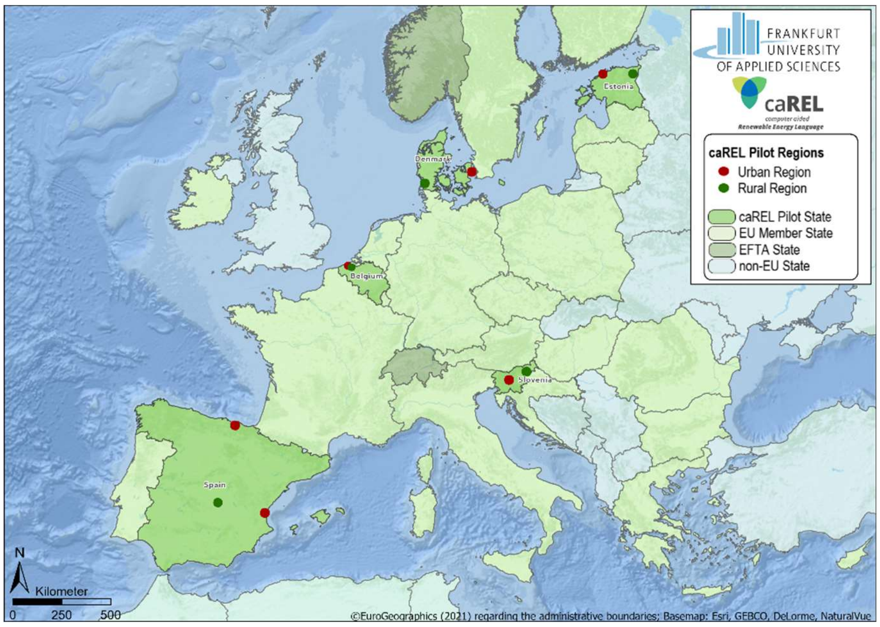


Figure 1 – Pilot States

These are the same five states in which wind potential analyses were carried out. Due to the larger land area of Spain compared to the other states, the calculation was carried out in three regions to highlight the differences between the northern and southern parts of Spain. Figure 1 shows a graphical overview of the selected project areas. For these test areas, the necessary baseline data for an analysis are available as open data. If an analysis is to be carried out in other states, it must first be checked whether the necessary data are available.

4. CALCULATION MODEL

Figure 2 shows the calculation model for the solar potential areas. It shows the various input and result data (in green) and the individual processing steps (in orange) and accruing intermediate results (in yellow).

The LiDAR data of the respective areas to be calculated serve as a starting point. Their availability is explained in more detail in chapter 5. Alternatively, the calculation could also be carried out on the basis of digital surface models (e.g. from INSPIRE). For this, the Python script would have to be adapted accordingly. This was not done in this project. If necessary, the LiDAR data must be processed before the calculation. For example, it is advisable to bring the source data into a regular grid (e.g. 1 km²), even if the algorithm can work with irregular grids.

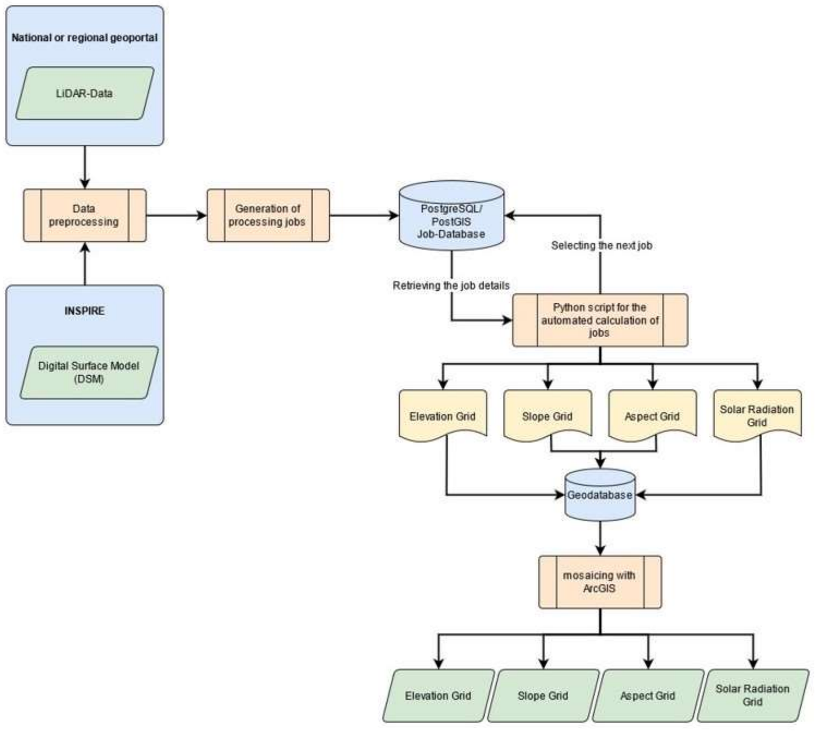


Figure 2 - Model overview solar potential algorithm

4.1 DATABASE AND JOBS

Before the solar potentials can be calculated using the Python script, the individual jobs must be created in the database. Each job corresponds to a single polygon, derived from the LiDAR data. For each of these jobs, a separate solar potential analysis is performed using the Python script. The jobs must cover not only the actual analysis area, but also an area around the analysis area necessary for shading calculation. LAStools is used to create the jobs. For the solar potential analysis, a polygon size of 1 km² was selected for caREL. This ensures that the individual calculation processes do not take too long and the loss of time in the event of errors remains small. Theoretically, larger jobs can also be processed. Grids with a lower resolution, for example 500 m², can be merged into a larger raster using the lasmerge tool of LAStools. Larger data sets without a regular grid or a grid with more than 1 km² resolution can be processed with the lastile function. This function tiles a set of LiDAR points from one or more source files into square, non-overlapping tiles of a given size, in this case 1 km. If the data is available in the desired grid, the extent of the individual polygons can be calculated with the lasboundary tool and output as a shapefile. To do this, navigate to the storage location of the LAZ files via the console and execute the following code.

*lasboundary -i *.laz -oshp -use_bb -o bounds.shp -overview -labels*

For each individual polygon, information on the number of laser scan points, the extent of the polygon and the minimum and maximum height is stored. With the help of the QGIS database management, the shapefile can be exported to the PostgreSQL database. Within the database, further adjustments must be made to the job table. The additionally added columns are used by the Python script to obtain certain information or to store results in the database. An added job ID column is used to uniquely identify the jobs. Within the database, individual jobs must be blocked if they have already been calculated or belong to the shading area at the border. Another table column is added for this purpose. To block the individual jobs, any text can be entered in this column. The script skips these columns during the calculation. In addition, further columns are added for time information and error messages, which are automatically filled by the script if necessary. Finally, the coordinate reference system of the source data must be specified. This is done via the PostGIS function UpdateGeometrySRID. The complete SQL code used for the database adjustments and the job tables is as follows:

```
ALTER TABLE jobs_solar
      DROP COLUMN id,
      ADD COLUMN job_id serial primary key,
      ADD COLUMN pcname text,
      ADD COLUMN start_time timestamp,
      ADD COLUMN finish_merge_time timestamp,
      ADD COLUMN finish_dom_time timestamp,
      ADD COLUMN finish_solar_time timestamp,
      ADD COLUMN finish_total_time timestamp,
      ADD COLUMN result_size double precision,
      ADD COLUMN error boolean DEFAULT false,
      ADD COLUMN error_cause text ;
select UpdateGeometrySRID('public', 'Table_Name', 'geom', 'SRID_Number');
```

Once these steps have been completed, the preparations for the solar potential analysis are finished.

4.2 SOLAR POTENTIAL ANALYSIS

A Python script is used for the solar potential analysis, which automatically processes the individual jobs stored in the database one after the other. It can be divided into two sections:

- Preparation and settings.
- Execution of the various software functions and storage of the results.

The script is started via a .bat file. It should be noted that the script used only uses a single processor core. Accordingly, on a system with a multi-core processor, the script can be executed several times simultaneously without loss of performance. The corresponding code for executing the script twice is shown below. If the script is to be run more than once, the corresponding lines of code simply need to be repeated.

```
start "" "Drive:\PathtoArcPy\arcgispro-py3\python.exe" "Drive:\PathtoScript\solar.py"

ping localhost -n 10

start "" "Drive:\PathtoArcPy\arcgispro-py3\python.exe" "Drive:\PathtoScript\solar.py"
```

The first section is executed once at start-up, while the second is in a loop and runs until all jobs are completed. The two sections are presented in more detail below. The script was developed by Philipp Winkemann, laboratory engineer in the Laboratory for Geoinformation at the Frankfurt University of Applied Sciences. Previously, it had already been used in several other projects, for example a solar potential calculation for the city of Darmstadt.
The Python script starts by importing various modules that will be used subsequently. This provides access to the code of these modules. In this way, for example, the functions of psycopg and ArcPy can be used. Subsequently, various variables are defined that serve as settings for the software components used. First, the connection parameters of the database are specified in this way. Then the file paths of the file geodatabases are defined, which are used to save the results. In addition, the file paths of the output data and the LAStools are specified. Then the parameters for the raster are defined. This concerns the overlap necessary for shading, the buffer radius and the raster resolution. For the areas calculated in caREL, an overlap of 200 m, a buffer radius of 250 m and a raster resolution of 0.5 m were selected. A more detailed analysis of different grid resolutions and their effect on the computation time of the solar potential analysis can be found in section 4.4. In the next part of the code, the parameters for the solar irradiance calculation are defined. The following values were defined:

- skySize: 200
- timeConfig: TimeMultipleDays (2022, 1, 365)
- dayInterval: 14
- hourInterval: 2
- calcDirections: 32
- zenithDivisions: 8

- azimuthDivisions: 8
- diffuseProp: 0,3
- transmittivity: 0,5
- zFactor: 1
- project_latitude: 50

Most of these specifications are in accordance with the ESRI recommendations for the use of the solar irradiance algorithm. The latitude is recalculated separately for each job in the second section of the script. After these definitions, some further settings follow. For example, a bounding box object is defined, which will be used later, and it is checked whether the necessary tools of the LAStools are available.
After defining these settings, the second section of the script follows, in which the calculations take place within a loop. The structure within the loop is shown in Figure 3. The processes are repeated until no more open jobs are available in the database and the script ends.

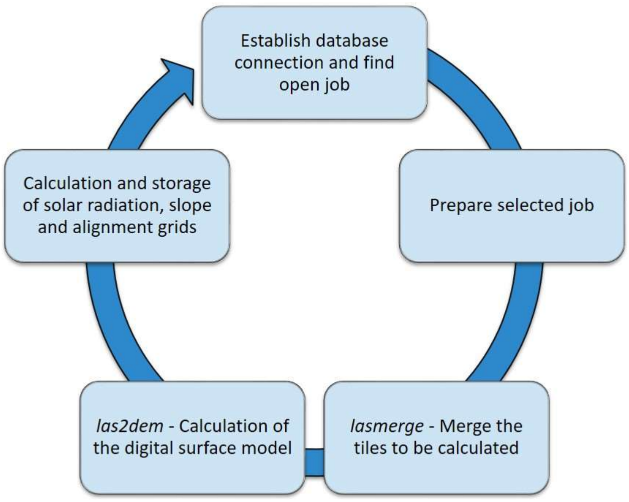

Establish database connection and find open job
Prepare selected job
lasmerge - Merge the tiles to be calculated
las2dem - Calculation of the digital surface model
Calculation and storage of solar radiation, slope and alignment grids

Figure 3 - Loop within the script

In the first step, the script uses psycopg functions to establish a connection to the PostgreSQL database. Within the database table, it searches for the first column with an open job, recognisable by a missing entry in the pcname column, writes the PC name of the executing computer into the corresponding data field and saves the job ID. As a result, this field is no longer empty and is skipped when searching for the next open job. If the script does not find a database entry with an empty pcname data field, it terminates automatically at this point. Using the stored job ID, the script can clearly identify which job and thus which LiDAR data tile it is currently processing.
In the second step, the processing of the selected job is prepared. First, the mean latitude of the tile to be processed is calculated and saved. PostGIS functions are used for this. Then a job box (bounding box of the job), a processing box (bounding box including overlap) and a clip box (bounding box including buffer radius) are calculated from the extent of the job specified in the database.
In the next step, the individual data sets of the job tile and the eight adjacent tiles are identified. To do this, a PostGIS function is used to check which data sets overlap with the job data set when overlap and buffer radius are added to the extent of the job data set. The LAStools tool lasmerge is then started and the individual data sets are merged. lasmerge receives the corner coordinates of

the clipbox as extension parameters and cuts the result to these coordinates. The resulting .LAZ file is saved in a temporary folder so that it can be deleted after the calculation is finished.

In the fourth step, the script uses the las2dem tool to calculate a digital elevation model, in this case a surface model. The LiDAR data set assembled in the previous step serves as the initial data for this. The extent of the result grid is that of the clipbox and is thus larger than that of the original job tile by the buffer radius. This guarantees that the shading at the edge of the job tile is correctly included in the solar radiation calculation in the next step.

In the final loop step, solar irradiance, orientation and slope are calculated and stored based on the digital surface model. First, it is checked whether a valid ArcGIS Pro licence is available so that the ArcPy functions can be used. Then the previously defined processing box is set as the extent. In the next step, the solar irradiance calculation is started with the parameters defined in the first section of the script and the newly calculated mean latitude. The surface model calculated in the fourth step serves as the basis for the calculation. This is also used as the basis for the slope and alignment grids calculated at this point. These are calculated with the ArcGIS functions Slope and Aspect and cut to the extent of the job tile. The calculated solar radiation raster is not cut to the size of the job tile at this point, as this would lead to errors when all job tiles are later merged into one result raster. All result grids calculated in this step are stored directly in a corresponding file geodatabase.

This described loop is run until there are no more open jobs. In this case the script terminates automatically. If the area to be calculated consists of five 1 km² tiles, for example, the result will be five surface models, slope grids, alignment grids and solar radiation grids. These four different result grids are stored separately in four file geodatabases.

The solar potential analysis is not yet complete; the following steps are described in the next section.

4.3 FINALISING THE RESULTS

As described in Section 4.2, the result of the Python script is one surface, slope, alignment and solar radiation grid per calculated job tile. These individual result grids must then be combined to obtain one complete grid each (surface, slope, orientation and solar) for the study area. The mosaic dataset function of ArcGIS Pro is used for this purpose. The result grids overlap in their boundary areas due to the overlap used. When creating a mosaic grid with ArcGIS Pro there are several possibilities to consider the overlapping areas of individual grids. The cell values can be merged using a weighting-based algorithm or using their mean, maximum or minimum value. For the merging of the elevation, alignment and slope grids, the mean value operator is used. Due to shading and its dependence on the direction of radiation and existing shadow-casting objects, the mean value cannot be used for the solar radiation grids. Instead, the minimum operator is used so that the maximum shading in the overlapping grids is always taken into account. In addition to the operator, the appropriate pixel type, in this case 32bit float, and the correct number of bands (1) must be selected. After the mosaic has been calculated, it can be exported as one contiguous data set. For this purpose, lossless data compression and a suitable data format, for example GeoTIFF, should be used if possible. This step concludes the solar potential analysis.

4.4 RUNTIME

The runtime is the period of time between the start and end of exactly one script run. This corresponds to the time required for the calculation of a job. The aim of this analysis was to find out which parameters are decisive for the duration of a script run. A Windows PC with Intel Core i9-9900 CPU (3.10 GHz) and 32 GB RAM was used as the test computer. The test results are shown in Figure 4. The total computing time increases with higher resolution. Here, the computation time of dataset B, with significantly lower point density, is slightly faster at resolutions of 5 m to 0.5 m than for dataset A. When the resolution is increased from 0.5 m to 0.25 m, the runtime in dataset A increases by a factor of 5.8, while it only doubles for dataset B. This shows that the computation time of dataset A increases by a factor of 5.8, while that of dataset B only doubles. This shows that the computing time at high resolutions is much more strongly influenced by the point density than at low resolutions.

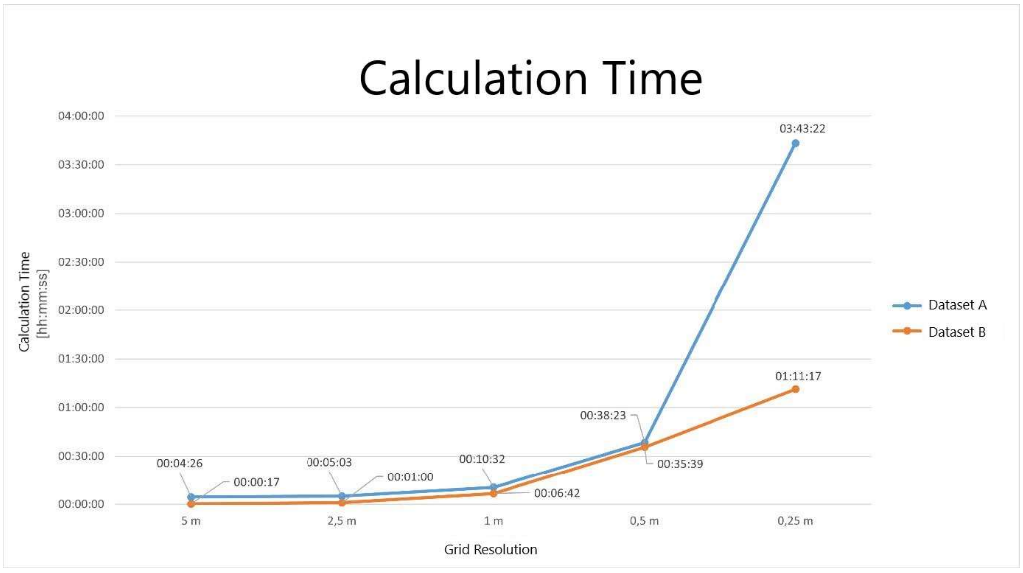


Figure 4 - Runtime calculations

In addition to the calculation time, different resolutions were compared in preparation for the solar potential analysis. Map series A (Figure 5) shows the result of the solar radiation calculation in different resolution levels at a scale of 1:3000 and map series B at a scale of 1:1000. The resolutions of 5 m and 2.5 m are too blurred to make a precise statement about the potentials. At 1 m grid resolution, the results are clearly more informative, which is why this resolution can be considered as minimally necessary. The resolutions of 0.5 m and 0.25 m offer an even higher information density. The difference between these is particularly noticeable in the level of detail of the vegetation. A grid resolution of 0.5 m offers a sensible middle ground between detail and computation time.

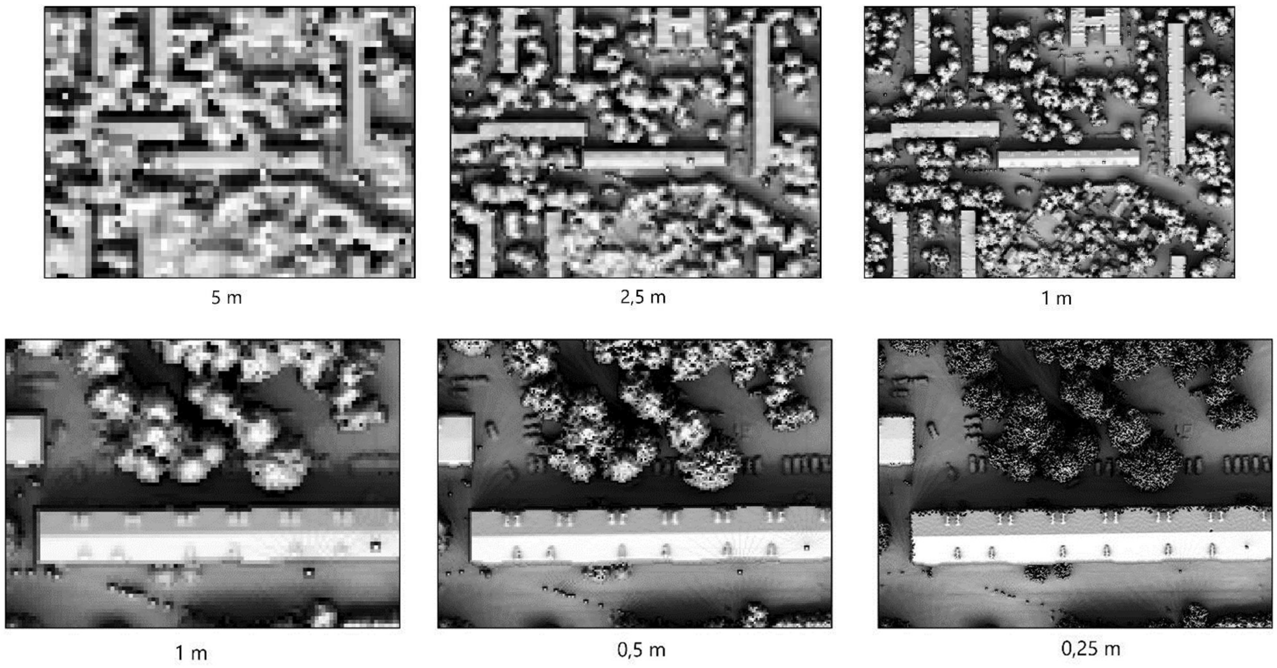


Abbildung 1 - Different resolution levels of the solar potential analysis

HESSEN

digitales.hes
DISTR@L

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

caREL
computer aided
Renewable Energy Language

EUROPÄISCHE UNION:
Investition in Ihre Zukunft
Europäischer Fonds für regionale Entwicklung

Project participants:

- Prof. Dr. Robert Seuß - Project Management
- Prof. Dr. Martina Klärle - Project Management
- Prof. Dr. -Ing. Tine Köhler - Project Management
- Prof. Dr. -Ing. Thomas Hollstein - Project Management
- Dipl. -Ing Ute Langendörfer - External Employee
- M.Eng. Mariam Hussain - Scientific Employee
- M.Eng. Nicolas Diedrich - Scientific Employee
- M.Eng. Julia Anderie - Scientific Employee

Frankfurt University of Applied Sciences
Nibelungenplatz 1
60318 Frankfurt am Main

carel@fb1.fra-uas.de
Tel. +49 (0)69 1533-3696
www.carel-energy.eu