The present work was submitted to the LuFG Theory of Hybrid Systems

A Novel Approach to Scale Shadow Flicker Simulations for Wind Farms Using Approximations

Ein Neuer Ansatz zur Skalierung von Schattenwurf-Simulationen für Windparks unter Verwendung von Näherungswerten

Bachelor of Science Thesis

August 26, 2024

Presented by Vorgelegt von	Patrick Erik Marcel De Smet Matrikelnummer: 435011 patrick.de.smet@rwth-aachen.de
First examiner Erstprüfer	Univ.-Prof. Dr. rer. nat. Erika Ábrahám Lehr- und Forschungsgebiet Theorie Hybrider Systeme RWTH Aachen University
Second examiner Zweitprüfer	Prof. Dr. Ralf Schelenz Center for Wind Power Drives RWTH Aachen University
Supervisor Betreuer	Nicolai Radke, M.Sc. Theory of Hybrid Systems RWTH Aachen University

Abstract

As wind farms expand near urban areas to meet the growing demand for renewable energy, evaluating their potential adverse effects on nearby buildings, particularly shadow flickering exposure, becomes crucial. Calculating the annual and daily duration of shadow flicker in residential areas, aligning with siting restrictions, is computationally expensive. To address this challenge, we propose a framework to estimate the exposure by leveraging pre-computed three-dimensional lookup tables and utilizing tri-linear interpolation, allowing us to efficiently compute the exposure for a sizeable set of receivers. Our approach yields a negligible method error with accuracy mainly depending on the resolution of the lookup table. We apply spline interpolation and high-degree polynomials to model the patterns within the lookup tables between different latitudes. Further, we explore various approaches to compress and generate shadow maps more efficiently.

Acknowledgments

I would like to give my warmest thanks to my supervisor, Nicolai Radke, for his helpful guidance and advice throughout all stages of this work. His ideas and impulses helped me develop and refine the concepts within this work. Moreover, I would like to thank Anja Noß-De Smet for proofreading and providing insightful suggestions to express this work's ideas as lucidly as possible. Finally, I would like to thank Prof. Dr. Erika Ábrahám at the Theory of Hybrid Systems chair and Prof. Dr. Ralf Schelenz at the Center for Wind Power Drives for the opportunity to work on this exciting topic and their ideas for further developments.

Contents

Introduction

In 2023, Germany expanded its onshore wind energy capacity by adding 745 new wind turbines with a total of 3,567 MW, marking a 48% increase in expansion compared to the previous year. The decommissioning of 423 wind turbines offsets this growth with a capacity of 534 MW [10]. The growing demand for renewable energy and wind power, coupled with spatial constraints in densely populated regions, has led to a trend of expanding wind farms closer to residential areas. The increased proximity of wind turbines to residential buildings is under criticism for potentially resulting in adverse effects on residents, such as shadow flicker and noise emissions. Current scientific studies, however, suggest no negative implications of shadow flicker on human health even though frequencies must remain lower than 2.5 Hertz to avoid nuisance [20, 26, 46]. Due to complaints related to shadow flicker and further justifications such as species conservation, many wind projects in Germany are currently subject to legal proceedings. The German state of Bavaria, in particular, stands out with wind projects constituting 42% of their total power output sued in 2019, followed by the states of Hessia and Baden-Wuerttemberg [36]. Legal regulations designed to mitigate shadow flicker effects require us to account for their impact in the evaluation of wind farm layouts, aligning with siting restrictions.

In this work, we are focusing on shadow flicker (”SF”) emissions. A position on the ground experiences shadow flicker when the shadows of the moving blades of a wind turbine cause a flickering effect on their surrounding areas as they rotate. Germany and many other European countries impose annual and daily limitations on the maximum duration of exposure for residential buildings and other types of buildings. For residential buildings, the annual limit is 30 hours with a maximum of 30 minutes per day [6]. A turbine must be shut down whenever it exceeds these limitations, potentially affecting efficiency. With the current implementation of the shadow cast module of the site-planning project WindFarm3D at RWTH Aachen, determining SF exposure is computationally expensive. We aim to reduce the computational effort, providing a fast and reliable method for approximation.

▶ Related Work As wind energy projects expand closer to urban areas, frictions with residents increasingly threaten to hinder the advancement of renewable energies, encouraging research into optimizing wind farm sites. Several papers have explored different approaches to optimization, such as genetic algorithms, greedy algorithms, particle swarm optimization, and more [1, 2, 3, 17, 33, 41, 42, 44]. In the following, we examine related work that incorporates SF emissions in the optimization.

The calculations to analyze SF exposure on surrounding areas are well-established. Approaches essentially differ in their method of including SF exposure in the optimization process. One approach, presented by Roscher in [38], includes the direct assessment of SF emissions in the optimization loop. This straightforward method induces time-consuming computations in each iteration, significantly slowing the optimization. Schellong et al. excluded turbine positions based on their distance to buildings [40]. Further, Dobrić et al. explored the calculation of priority values for SF exposure and reused them when computing the objective in each optimization step [12]. There exist multiple commercial tools such as windPRO and WindFarm that can compute SF emissions for an existing wind farm layout but do not incorporate SF in the optimization. To the best of our knowledge, no existing approaches aim to reduce the running time of SF computations.

▶ Contribution This work introduces a novel method for efficiently approximating SF in settings with multiple wind turbines. Our findings indicate that the error margin of this method remains minimal in most settings when utilizing high-resolution lookup tables. We present several approaches to model shadow maps using functions dependent on latitude. We also explore alternative methods to compute shadow maps more efficiently and reduce their memory requirements.

▶ Outline In Section 1, we cover the fundamentals of computing SF emissions, interpolating within three-dimensional lookup tables, and constructing two-dimensional matrices using spline interpolation and generative neural networks. Section 2 details the computation of shadow maps for efficient SF approximation and the lookup and interpolation method given a wind farm layout with multiple turbines. Section 3 focuses on reducing memory requirements and generating shadow maps using alternative computation methods. Further, Section 4 presents our implementation of these techniques within our shadow cast tool. Finally, we validate our methods in Section 5, demonstrating low error margins in most realistic scenarios.

1 Preliminaries

In this section, we provide an overview of computing SF exposure through ray backtracing and estimating the duration of exposure using uniformly sampled solar positions. We elucidate tri-linear interpolation within a three-dimensional cuboid, and spline interpolation in a two-dimensional setting. Additionally, we introduce the fundamentals of Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) in the context of processing and generating image data.

1.1 Shadow Flicker Computations

To evaluate the effect of SF on residential buildings, we compute the duration of exposure throughout the year. We follow the common assumption of a worst-case setting: a constantly clear sky, perfect turbine alignment towards the sun, and continuous operation of the turbine. The shadow cast by a wind turbine's rotor depends on the ground position relative to the rotor center, the rotor diameter, the turbine alignment, and the sun's position.

Two angles, the azimuth ϕ_s, and elevation θ_s, define the observer-centric solar position. The azimuth angle represents the sun's direction along the horizon, while the elevation angle describes the sun's height above the horizon. We generally write both angles in degrees but use radians for SF calculations.

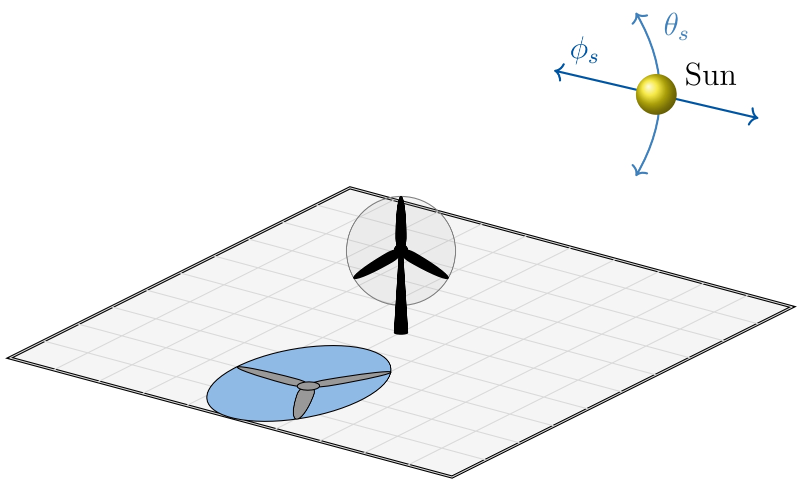


Figure 1: Projected shadow ellipse of a wind turbine on the ground, determined by solar angles: azimuth ϕ_s and elevation θ_s.

We refer to the wind turbine as the emitter, while the position on the ground or building is the receiver. We do not take into account the existence or lack of windows of residential buildings in our calculation and, therefore, model them with the greenhouse mode. This approach accounts for SF flicker exposure from all cardinal directions, even if no windows face the wind turbine and, consequently, no SF could be perceived by residents

[23]. Additionally, we do not account for sun-blocking objects such as hills, houses, or trees that might obstruct SF emissions from reaching the receiver.

1.1.1 Exposure Approximation

We aim to determine the SF exposure for a single receiver over an entire year and for each individual day. Using ray backtracing, as described in Section 1.1.2, we assess whether a specific ground position is exposed to SF at any given time t. To estimate the duration of SF over a time interval, we sample solar positions uniformly throughout that interval, check for SF at each sample point, and approximate the SF duration by assuming constant exposure between consecutive samples. With N uniformly sampled solar positions (ϕ_i, θ_i, t_i), where each sample is separated by the sampling resolution Δt, we describe the existence or lack of SF using the following characteristic function:

$$\mathbb{1}_{SF}(i) = \begin{cases} 1 & \text{if SF occurs at time } t_i \\ 0 & \text{otherwise} \end{cases}$$

Further, we approximate the duration of SF exposure for the time interval that we cover with our discrete samples by multiplying all actual instances by the sampling distance:

$$T_{\text{approx}} = \Delta t \cdot \sum_{i=1}^{N} \mathbb{1}_{SF}(i)$$

To compute the annual SF exposure, we utilize samples of the solar position covering the entire year. For the maximum daily SF, we calculate the SF exposure duration for each day and then determine the maximum value across all days. The accuracy of this approach is merely dependent on the sampling resolution Δt, and it converges to 100% as $\Delta t \to 0$. This is because our method approximates the true SF exposure as a Riemann sum[1] over the sampled interval $[a, b]$:

$$T = \int_a^b f_{SF}(t)dt = \lim_{||\Delta t|| \to 0} \sum_{i=1}^{N} \mathbb{1}_{SF}(i) \cdot \Delta t$$

In this expression, $f_{SF}(t)$ represents the function for SF at time t, similar to $\mathbb{1}_{SF}(i)$, which equals 1 if SF occurs and 0 otherwise. We can construct $\mathbb{1}_{SF}(i) = f_{SF}(t_i)$. The integral determines the total SF duration over the specified time period, while the Riemann sum approaches the integral as the sampling interval Δt decreases. Our function $f_{SF}(t)$ is Riemann integrable since it is non-negative by definition, bounded on the interval $[a, b]$ and has a set of discontinuities with Lebesgue measure zero since only countably many continuous instances of shadowing and thus discontinuities where shadowing begins or ends may occur [32].

[1] A Riemann sum is a method of approximating an integral using a finite sum.

1.1.2 Shadow Ray-Backtracing

Using ray backtracing[2], we can accurately determine if a ground position experiences SF at a given time. We trace a vector from the ground in the direction and with the inclination of the solar rays to observe whether it intersects the turbine rotor. If true, the sun's rays would pass through the rotor, causing a shadow whenever the blades rotate.

We require the azimuth ($\phi_{s,t}$) and elevation angle ($\theta_{s,t}$) of the sun at time t, the position $p_{\text{rotor}} = (x_r, y_r, z_r)^T$, hub height h_r, and rotor diameter d_r of the wind turbine, and the position of a receiver as a point the ground $p_{\text{ground}} = (x_g, y_g, z_g)^T$ to perform the necessary ray backtracing computations.

The vector $\vec{v}_{\text{sun}}$ denotes the direction of the solar rays from the sun at its current position $(\phi_{s,t}, \theta_{s,t})$ to the ground, assuming the rays are parallel. We can make this assumption due to the Earth's large distance from the sun. The normal vector $\vec{n}_{\text{rotor}}$, that together with the position of the wind turbine's nose, defines a plane spanned by the turbine's rotor. This plane is perpendicular to the sun's azimuth angle whenever we compute the worst-case setting. Thus, we assume the rotor always faces the sun. In a statistically realistic setting, when the turbine's rotor is directed towards the opposite of the wind direction, we can replace the sun's azimuth with the angle of the wind direction at time t.

$$\vec{v}_{\text{sun}} = \begin{pmatrix} \cos(\theta_{s,t}) \cdot \sin(\phi_{s,t}) \\ \cos(\theta_{s,t}) \cdot \cos(\phi_{s,t}) \\ \sin(\theta_{s,t}) \end{pmatrix}, \quad \vec{n}_{\text{rotor}} = \begin{pmatrix} \sin(\phi_{s,t}) \\ \cos(\phi_{s,t}) \\ 0 \end{pmatrix}$$

We utilize both vectors to determine the intersection point of the sun vector with the rotor plane. Subsequently, we compute the Euclidean distance of the intersection point to the turbine center on the plane. This tells us how far the solar rays intersect the plane away from the turbine center.

$$p_{\text{intersect}} = p_{\text{ground}} + \left(\frac{-\vec{n}_{\text{rotor}} \cdot (p_{\text{ground}} - p_{\text{rotor}})}{\vec{n}_{\text{rotor}} \cdot \vec{v}_{\text{sun}}} \right) \cdot \vec{v}_{\text{sun}}$$

$$d_{\text{intersect}} = \|p_{\text{intersect}} - p_{\text{rotor}}\|_2$$

If this distance is less than or equal to the rotor radius, the ground position is considered to experience shadow flicker. For a single receiver, we utilize the computations above for each sample of the solar position and determine the daily and annual SF duration as described in Section 1.1.1.

[2]Ray backtracing involves projecting rays from a source to determine if they intersect with an object.

1.2 Tri-linear Interpolation

Tri-linear interpolation is a method of interpolating values within a three-dimensional space using known values at the vertices of a cube.

We aim to interpolate the value of some arbitrary point $(x, y, z) \in \mathbb{R}^3$ within the boundaries of a three-dimensional grid of uniformly sampled points. We interpolate using the values of the eight nearest points within the grid, which form a three-dimensional cuboid.

Let the coordinates of the eight vertices of the cube surrounding the point (x, y, z) be denoted as c_{abc}, with $abc \in \{0, 1\}^3$ and corresponding values $\phi(abc)$.

Further, let (x_0, y_0, z_0) and (x_1, y_1, z_1) be the coordinates of the two opposite vertices c_{000} and c_{111}. We define the normalized relative coordinates of our point (x, y, z) within the cube as following:

$$u = \frac{x - x_0}{x_1 - x_0}, \quad v = \frac{y - y_0}{y_1 - y_0}, \quad w = \frac{z - z_0}{z_1 - z_0}$$

We interpolate using the formula for tri-linear interpolation:

$$\begin{aligned}
f(u, v, w) = {} & \phi(000)(1-u)(1-v)(1-w) + \phi(100)u(1-v)(1-w) \\
& + \phi(010)(1-u)v(1-w) + \phi(110)uv(1-w) \\
& + \phi(001)(1-u)(1-v)w + \phi(101)u(1-v)w \\
& + \phi(011)(1-u)vw + \phi(111)uvw
\end{aligned}$$

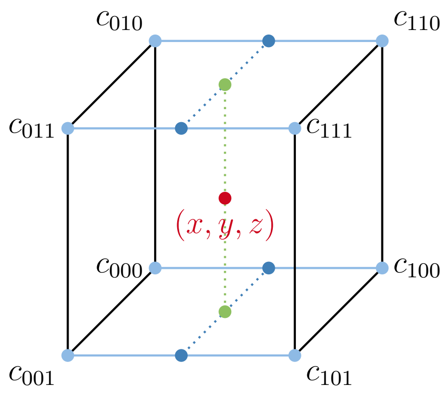


Figure 2: Illustration of trilinear interpolation within a 3D cube.

In a more intuitive geometric notion, as seen in Figure 2, we first interpolate along the x-axis to find intermediate values at the faces of the cube, reducing the linear interpolation to two dimensions. Next, we interpolate these intermediate values along the z-axis to finally obtain our result using one-dimensional linear interpolation [7].

1.3 Spline Interpolation

With spline interpolation in a two-dimensional setting, we interpolate between a set of known data points using a piecewise polynomial function[3] with a fixed degree. Compared to high-degree polynomials, spline interpolation minimizes oscillations and avoids the Runge's phenomenon.

Given a set of data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where x_i are the input values and y_i are the corresponding output values, we aim to find a smooth function $S(x)$ that interpolates between these points. We construct a function $S(x)$ such that each polynomial segment $S_i(x)$ interpolates between two consecutive points (x_i, y_i) and (x_{i+1}, y_{i+1}). E.g. for cubic splines, each segment $S_i(x)$ is a cubic polynomial [48]:

$$S_i(x) = a_i(x - x_i) + b_i(x - x_i)^2 + c_i(x - x_i)^3 + d_i$$

To ensure the smoothness of $S(x)$, we impose a set of continuity conditions for the splines S_i at each inner point x_i. The spline function $S(x)$ must pass through each data point and the first derivative $S_i'(x)$ and the second derivative $S_i''(x)$ must be continuous at each inner point. Using these conditions and the equation for the spline, we can formulate a system of linear equations for the coefficients a_i, b_i, c_i, and d_i [31]:

$$\begin{aligned}
S_i(x_i) &= y_i, \\
S_i(x_i) &= d_i, \\
S_i(x_i) &= S_{i-1}(x_i), \\
S_i'(x_i) &= S_{i-1}'(x_i), \\
S_i''(x_i) &= S_{i+1}''(x_i) \\
S_i''(x_i) &= S_{i+1}''(x_i)
\end{aligned}$$

With the resulting spline function $S(x)$, we can reliably interpolate any x within the range of the given data. The function $S(x)$ is continuously differentiable. Furthermore, we can choose other splines depending on the desired complexity of our function. For example, we can utilize quadratic splines, where each segment is a quadratic polynomial, or higher-order splines for more complex approximations. In each case, the degree of the polynomial and the corresponding continuity conditions must be adjusted accordingly.

Furthermore, we can smooth the spline representation by some smoothing factor s, allowing for a trade-off between the adherence to the data points and the smoothness of the resulting function. This is particularly useful when data sets are noisy data and an exact interpolation might overfit. Thus, a smoothed spline often provides a more generalized and robust approximation.

[3]A piecewise polynomial is a function composed of multiple polynomial segments.

1.4 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are often used for image processing and classification tasks. CNNs can learn to abstract and interpret patterns in images. Their architecture, as introduced by LeCun et al. in [4], consists of input, convolutional, pooling, fully connected, and output layers, as shown in Figure 4. Each type of layer serves a unique purpose in the network's ability to process and comprehend visual data.

The convolutional and pooling layers augment the typical artificial neural network (ANN), comprising the input, output, and fully connected layers. This allows for better abstraction of image data. In general, neural networks are systems that adapt their internal configuration in relation to an objective function [18]. Given training samples encompassing input and output variables, we can train the network to learn the underlying function that generated the samples. We call this method of training supervised. Generic neural networks work particularly well with numerical data but fail to learn image data's structural and spatial relations.

With convolutional layers, we extract meaningful information such as corners, edges, shapes, and intensity patterns to evaluate them further using the fully connected layers [43]. Convolutional layers employ multiple kernels, also known as filters, which are small matrices with learnable parameters. Each kernel slides over the input image, performing a convolution operation at every position. This linear operation generates a feature map, where each pixel value in the feature map represents the kernel's response to a specific image region, as shown in Figure 3. Using multiple kernels, a convolutional layer detects various patterns and structures. Typically, multiple successive convolutional layers progressively detect increasingly complex features, allowing the network to build detailed and hierarchical image representations.

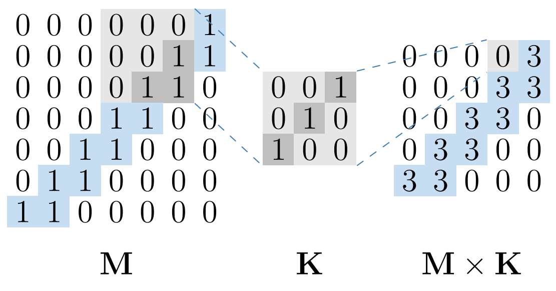


Figure 3: Visualization of convolution operation: initial matrix **M**, kernel **K**, and the resulting matrix **M** × **K**.

After the convolutional layers, we apply pooling layers, also known as down-sampling layers. By utilizing them, we aim to reduce the spatial dimensions of the feature maps. With smaller feature maps, further computations are more feasible, and the number of parameters in the network decreases. This helps the CNN prevent overfitting. In max pooling, the maximum value is taken from a fixed-size window of cells from the previous layer, while in average pooling, the average value is taken. This process reduces the size of the feature maps and retains the most important features detected by the convolutional layers.

After several convolutional and pooling layers, the network flattens the extracted information and passes them to one or more fully connected layers[4]. These learnable layers interpret the high-level features to perform the final classification. They combine the features in complex ways to determine the class of the input image, outputting a probability distribution over the possible classes [34].

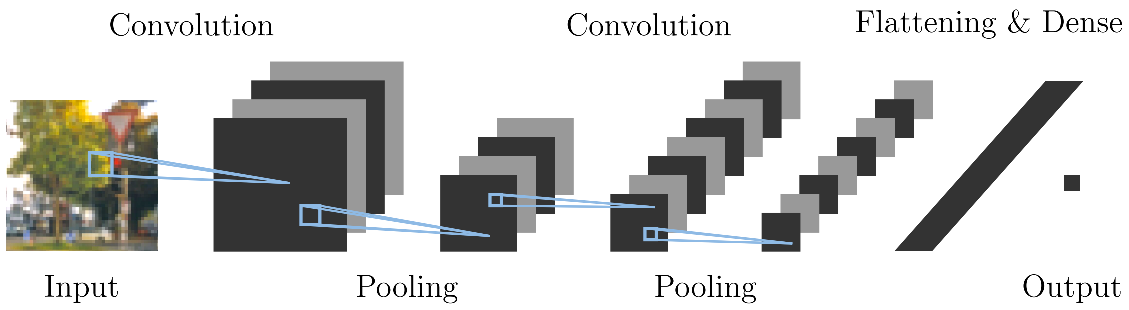


Figure 4: Exemplary visualization of a CNN. A first convolutional layer captures the high-level features of the input image. Furthermore, a pooling layer downsamples the information, and another convolutional layer extracts the final features, which are flattened and passed to the dense layer that provides a classification result.

[4]In fully connected layers, each neuron is connected to every neuron in the previous inner layer. They are typically used to interpret features and return final predictions or classifications.

1.5 Generative Adversarial Networks (GANs)

On the contrary, to generate images, another type of neural network called Generative Adversarial Networks (GANs) is often used. GANs consist of a generator and a discriminator component. The generator applies transpose convolutions or ”Deconvolutions” to upsample numerical data into a structured image. The discriminator, on the other hand, is a CNN that attempts to distinguish between real images from the training set and forged images produced by the generator.

During training, the generator and discriminator compete against each other, as shown in Figure 5. The generator aims to produce increasingly realistic images to trick the discriminator. The discriminator, conversely, gets better at identifying real versus forged images. By relying solely on the discriminator’s feedback, the generator learns how to imitate the original images without access to the training data.

GANs are typically very difficult to train. The training process is often unstable which results in problems such as vanishing gradients, where the discriminator becomes too good at identifying the forged samples and the generator fails to learn [25]. To prevent this phenomenon, we must carefully tune the hyperparameters or adjust the architecture. Often, the learning rate of the generator is set higher than that of the discriminator to ensure that both networks learn at a consistent pace and to prevent the discriminator from overpowering the generator too quickly [16].

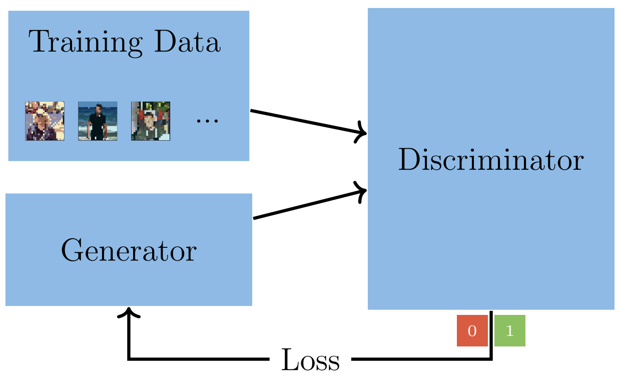


Figure 5: The generator creates images that are fed to the discriminator alongside real samples. The discriminator classifies these as real or fake and provides loss feedback to the generator. This iterative process trains the generator to produce realistic images and the discriminator’s accuracy over time.

Additionally, we can constrain the generation process using additional information such as class labels or continuous inputs, allowing for more control over the output. These models are called Conditional GANs (CGANs). We pass the additional information to the generator and the discriminator, ensuring that the discriminator gives appropriate feedback considering the additional variable.

2 Shadow Approximation

Computing the SF duration for multiple buildings using discrete solar position samples is computationally expensive. The complexity depends on the number of buildings, turbines, and the step size of the solar samples, as described in paragraph 2.1. We present a method to approximate SF efficiently for multiple turbines and receivers as we necessitate more feasible calculations to include SF as an objective within the optimization of wind farm layouts. We utilize pre-computed lookup tables and tri-linear interpolation. In the following we refer to the lookup tables as shadow maps.

A shadow map is a three-dimensional lookup table that captures the annual and maximum daily SF durations for various positions on the ground, assuming a wind turbine located at the center of the map.

2.1 Shadow Map Architecture

To compute a shadow map, we initialize a uniform three-dimensional coordinate grid, as shown in Figure 6. We set the size of both horizontal dimensions to twice the assumed maximum shadow distance. As the intensity of SF decreases the farther one moves away from the turbine, the intensity is diffuse enough at 15 times the rotor diameter such that we do not have to consider distances beyond [20]. These large dimensions are necessary to account for long shadows in all cardinal directions from our turbine in the middle of the lookup table. In our case, we choose a standard maximum distance of 1500 meters for a turbine with a hub height of 100 meters and a rotor diameter of 80 meters. Furthermore, shadows beyond this distance typically occur only during periods of low sun elevation, particularly dawn and dusk. German law allows us to disregard SF for solar elevations lower than 3 degrees [27].

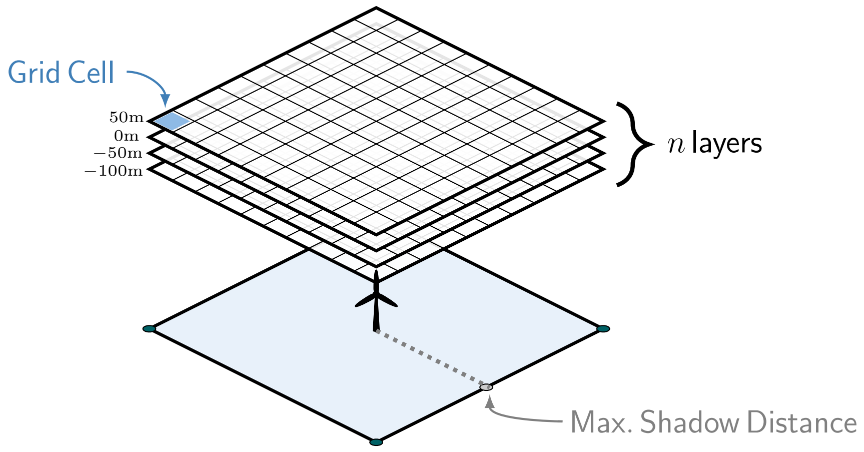


Figure 6: Three-dimensional shadow map comprising n layers from 50m to −100m with a wind turbine in the center.

For the z-dimension, we set the maximum height to the hub height subtracted by the radius of the rotor. The hub height of a wind turbine is the distance from the ground to the center of the turbine's rotor, where the blades are attached. Higher layers are most likely not useful since elevations reaching the height of the turbine's rotor blade, causing the turbine to stand in a valley, significantly affect the turbine's efficiency and, therefore, represent an unlikely scenario. Moreover, the elevation of the lowest layer is site-dependent due to differing terrain roughness[5]. If the turbine stands on a higher elevation in the terrain, we can increase the minimum height to account for lower buildings. When computing a shadow map, we assume flat terrain for each layer. Only later, with the interpolation between the layers, do we look at terrain information.

For each point in the grid, we require the relative x, y, z coordinates to the lowest leftmost point within the grid in meters. This information is necessary to interpolate within three-dimensional cuboids in our lookup table using relative positions in meters. Further, we compute the annual and daily SF duration for every grid point, considering the wind turbine in the center of the map.

▶ Shadow Map Parameters Our computed shadow map depends on the latitude of our wind farm, the hub height and rotor diameter of the assumed wind turbine, and the dimensions and resolution of the grid. We differentiate between the horizontal resolution in grid points per meters on the x- and y-axis, and the number of layers, allowing us to find a better trade-off between spatial complexity and accuracy, as further described in paragraph 2.1. We do not consider the longitude as a relevant parameter since it only influences the timing of shadow occurences and not the duration when ignoring earths rotation around the sun. Thus, it has no significant impact on the mere duration of SF.

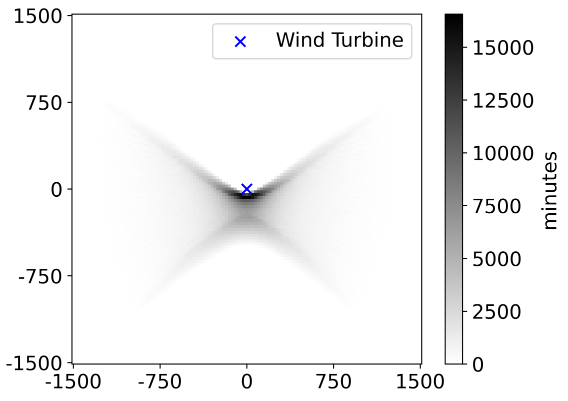


Figure 7: Annual shadow map for a wind turbine at the geodetic latitude 51°N, with a hub height of 100 m and a rotor diameter of 80 m.

[5]Terrain roughness refers to the intensity of variations in ground elevation.

▶ Computational Complexity Let us consider a shadow map with a maximum shadow distance of 1500 meters, a horizontal resolution of n simulation points per meter, and l vertical layers. The shadow map comprises a total of $l \cdot (3000 \cdot n + 1)^2$ grid points. Further, we generate m samples of the solar position per minute throughout the entire year, resulting in a list containing $m \cdot 525600$ samples. The higher the resolution parameters n, l, m get, the larger our dimensions and amount of samples grow. For every grid point and each sample of the solar position, we compute whether the position experiences SF, as shown in Section 1.1.1. We assume the computation for a single solar sample has complexity $\mathcal{O}(1)$ and does not depend on the size of our input values. Subsequently, the computation has the following asymptotic complexity:

$$\mathcal{O}\left(m \cdot 525600 \cdot l \cdot (3000 \cdot n + 1)^2\right) = \mathcal{O}\left(mln^2\right)$$

The computational complexity grows polynomially concerning the number of solar samples, vertical layers, and horizontal resolution. Specifically, the complexity scales quadratically with the horizontal resolution n.

We differentiate between the number of layers and the horizontal resolution since the areas experiencing SF vary strongly in size between different elevations. The shaded area gets increasingly smaller for lower relative distances between the ground and the rotor center on the vertical axis, especially for positions horizontally distant from the wind turbine. Furthermore, we need to cover a significantly smaller dimension of merely a few hundred meters on the vertical axis, from the hub height, including the rotor diameter, to the minimum layer height. Consequently, we aim to find a good trade-off between horizontal and vertical resolution and accuracy.

In practice, the computation of a shadow map with a reasonable choice of hyperparameters can require up to multiple hours on a standard issue multi-core CPU computer, even though we already take into account JIT compilation[6] and parallelization to speed up calculations, as further described in Section 4. However, we can utilize shadow maps any number of times once pre-computed.

[6]JIT (Just-In-Time) compilation accelerates code execution by compiling code during runtime. In commonly interpreted languages such as Python, code segments can be accelerated using JIT compilation with special libraries.

2.2 Shadow Flicker Lookup

We utilize the pre-computed lookup tables, our shadow maps, to efficiently estimate the SF exposure for a wind farm layout with n wind turbines and m receivers. We look up the annual and daily SF exposure in minutes for each receiver based on its relative position to every turbine. In the following we elaborate on the interpolation approach for one turbine and receiver and further with multiple emitters.

▶ Emitter-Receiver Interpolation Given a wind turbine and one single receiver, we look up the annual and maximum daily SF exposure based on the receiver's relative position to the turbine. Since this position likely does not correspond exactly to any single grid point within our shadow map, we use tri-linear interpolation, finding the SF values at the eight nearest grid points on the shadow map, as detailed in Section 1.2. We consider the wind turbine to be at the center of our shadow map with $x = 0, y = 0, z = 0$. We further compute the receiver's relative coordinates by moving in the direction specified by the normalized relative position of the receiver, as illustrated in Figure 8. In simple terms, we locate the receiver within our shadow map with the turbine at the center, find the closest grid points, and apply tri-linear interpolation to estimate the SF exposure at the receiver's position.

Suppose the actual turbine's hub height differs from the height used in the shadow map. In that case, we adjust the z-coordinate of the receiver's relative position: subtracting the height difference if the turbine is lower and adding it if the turbine is higher. All other parameters, such as rotor diameter, must be identical between the actual turbine and the shadow map parameters. We then use tri-linear interpolation with the adjusted relative position to determine the annual and daily SF exposure.

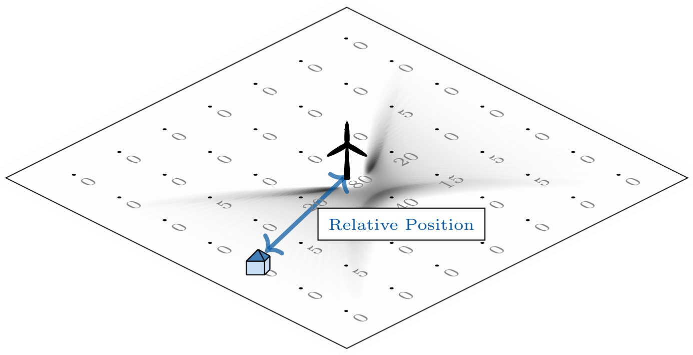
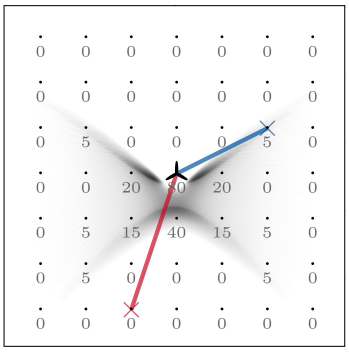


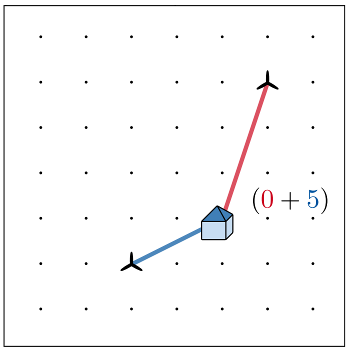
Figure 8: Interpolation of SF duration for one emitter and receiver. The figure depicts a shadow map centered on the wind turbine, showing shadow flicker exposure in minutes at various grid points. The normalized relative position of the receiver to the wind turbine is highlighted.

▶ Exception Handling If a receiver's relative position extends beyond the maximum shadow distance on one of the horizontal axes, we assign a SF value of zero since the intensity of emissions is low, as outlined in Section 2. For positions that fall outside the highest or lowest vertical layers, we use bi-linear interpolation[7] on the nearest horizontal layer. In such situations, the assumption that the SF intensity is diffuse enough may not hold true, given that the lowest considered layer might be only a few hundred meters from the receiver, depending on vertical dimensions of the shadow map. Subsequently, this approach improves accuracy compared to simply assuming no SF exposure. In our implementation, we issue a warning when an exception occurs. Generally, if such exceptions occur frequently, we recommend reconsidering computing a shadow map with adjusted dimensions.

▶ Multiple Emitters Given a wind farm with multiple turbines, we estimate SF for a single receiver by considering each turbine independently, as detailed in paragraph 2.2. We determine the receiver's position relative to each turbine, perform tri-linear interpolation to compute the SF exposure for each turbine-receiver pair, and then sum the interpolated values from all turbines to obtain the total SF for the layout. Regulations mandate evaluating SF from the receiver's perspective, as multiple turbines can collectively exceed allowable SF limits. If SF limits are exceeded, all turbines that further cause SF for the receiver must be shut down.



(a) Two-dimensional shadow map with SF at various points for a turbine in the center. The figure indicates the relative positions of both turbines to the receiver.



(b) The figure depicts one receiver and two turbines with relative positions. Both interpolation results are aggregated to the overall SF exposure.

Figure 9: Assessment of SF exposure for a receiver relative to two wind turbines: we determine the receiver's position relative to each turbine, reference the shadow map to find the SF exposure along the direction of this relative position from the shadow map's center, and aggregate the exposure results for both turbines.

[7]Bi-linear interpolation, the two-dimensional counterpart to tri-linear interpolation, applies linear interpolation sequentially in one direction and then in the perpendicular direction. [13]

If wind turbines in a wind farm layout differ only in their hub height, the same shadow map can be used for all lookups. We adjust for hub height differences by modifying the vertical coordinate as described in the emitter-receiver interpolation in paragraph 2.2. However, if turbines differ in other parameters, such as rotor diameter, separate shadow maps must be assigned.

▶ Additive Method Error When simulating SF using discrete solar positions, as detailed in Section 1.1.2, a ground position is marked as experiencing SF if at least one turbine casts a flickering shadow on it. Hence, when simulating SF for one receiver and multiple turbines, we consider a receiver to experience SF regardless of whether multiple turbines are responsible for the exposure. However, with our additive approach of estimating SF, as outlined in paragraph 2.2, whenever n turbines simultaneously cast flickering shadows on a single receiver, we count these instances n times since we aggregate the respective SF exposure of all turbines. Consequently, we overestimate the SF exposure. This error source becomes more significant when the wind turbines are closely spaced and the sun is at a low elevation, leading to extended shadows. In these scenarios it is more likely that multiple turbines cast shadows on one receiver simultaneously. We analyze the severity of this error in Section 5.

2.3 Daily Shadow Flicker Map

With shadow maps we can effectively estimate the annual duration of SF with a manageable error, as detailed in Section 5. However, using the same method to calculate the maximum daily SF exposure can, in some cases, introduce significant inaccuracies. If we look up and interpolate the maximum daily SF for each turbine relative to a single receiver with the method in paragraph 2.2, we do not account for the exact days when the daily maximum is reached for each turbine. For example, one turbine might cause 30 minutes of SF on January 1st, while another turbine casts no shadow on that day but reaches a daily maximum of 25 minutes of SF on a different day in September. Since we store only the 30 and 25 minutes of maximum daily SF for the relative positions within our shadow maps without regarding any temporal information when these maxima are reached, we would sum these values to 55 minutes, even though the actual daily maximum is merely 30 minutes.

To compensate for this error and provide a considerably more precise method for determining the maximum daily SF, we can compute a shadow map with an alternative structure that integrates the temporal component of the occurrences of SF exposure. For every grid point, we additionally store the SF duration for each day, totaling 365 attributes. Figure 10 displays the temporal alternative axis of the structure.

Daily shadow maps require significantly more space. However, by utilizing unsigned 8-bit integers to store the daily duration of exposure, with 255 minutes as a reasonable maximum value for a single grid point, we compensate for the additional space to some extent. Additionally, when using a compression algorithm, we can further reduce the space required for the SF data, as the values of the individual grid points are multiples of the sampling distance for the solar position. Moreover, many grid points are zero. This results in less unique values which can be compressed efficiently.

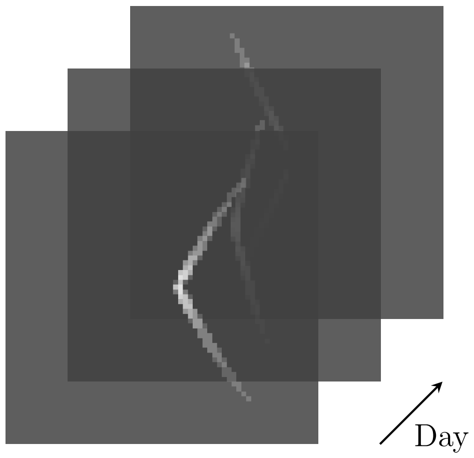


Figure 10: Visualization of the temporal axis of daily shadow maps for a single two-dimensional layer. We invert the colors of the respective layer for better visibility of the SF exposure.

▶ Interpolation Approach The method for calculating annual shadow flicker SF follows a process similar to the one described in paragraph 2.2. We use tri-linear interpolation to compute the SF for each day individually and aggregate these daily values to obtain an annual total. This annual total can be combined with results from other turbines as outlined in paragraph 2.2. In simpler terms, we utilize the approach in paragraph 2.2 for each temporal layer of our daily shadow map and aggregate the results. However, to determine the maximum daily SF for n turbines and one receiver, we retain temporal data from our daily shadow maps when aggregating with multiple turbines. By estimating the SF emissions for each turbine on each day of the year, we create n arrays of daily emissions, one for each turbine. We then sum these arrays element-wise and find the maximum value from the resulting array to approximate the maximum daily SF for all turbines. Consequently, we determine the SF exposure for each day regarding all turbines and then find the maximum value instead of the other way around.

▶ Compatibility In Section 3, we explore approaches to generate shadow maps more efficiently without the need to compute the SF exposure for each grid point using the method in Section 1.1.1. We aim to reduce the computational complexity, as detailed in paragraph 2.1. Further, we attempt to reduce the storage requirements of our shadow maps by exploiting attributes such as approximate symmetry along the north-south axis. Many of these methods are compatible with the structure of daily shadow maps or can be adapted easily. However, some approaches, such as equatorial symmetry, are incompatible due to the additional temporal information.

3 Map Interpolation & Storage Reduction

As shown in paragraph 2.1, computing shadow maps is computationally expensive. While it may seem counter-intuitive to use them due to the added inaccuracies when interpolating, they become efficient when repeatedly evaluating SF for different wind farm layouts at the same geographic coordinates, as the pre-computed shadow map can be reused. This is particularly useful when incorporating SF in the objective of an optimization algorithm. Nevertheless, this method does not yet achieve the required computational efficiency to significantly accelerate SF simulations since we have to pre-compute the shadow map first, causing a large computational overhead.

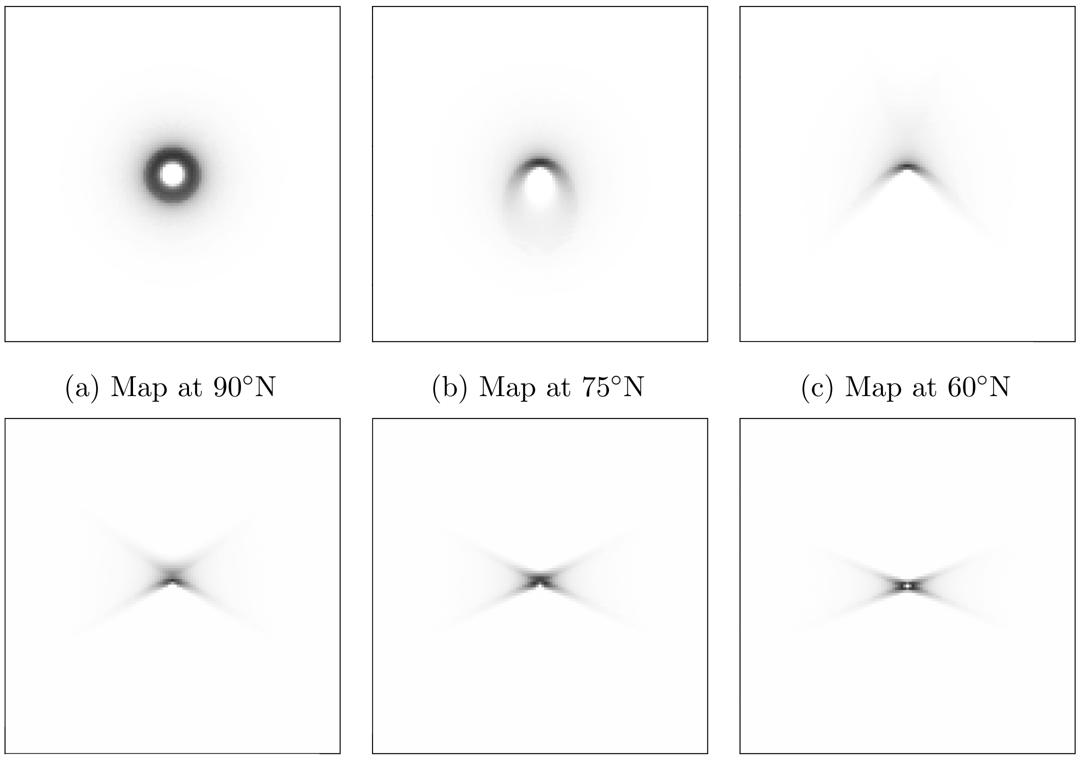
In this chapter, we explore alternative methods for efficiently inferring shadow maps based on latitude. We aim to reduce the computational effort required to generate valid shadow maps. Further, we want to reduce their spatial complexity by exploiting attributes such as approximate symmetry along their north-south axis and the equator.

3.1 Geographical Dependence

The duration of SF cast by a wind turbine on the ground, disregarding the time of occurrence throughout the day, is predominantly determined by the turbine's latitude, as described in Section 1.1.2. At the poles, shadowing occurs in all cardinal directions. For several months, the solar elevation angle remains high, allowing for shadowing in all directions. As we move closer to the equator, the range of cardinal directions where shadowing is possible becomes more restricted to the east and west. Graphically, this transition from the poles to the equator exhibits a highly uniform and predictable pattern. We can observe this pattern in Figure 11.

▶ Weighted Mean Given a latitude lat and two samples of shadow maps for latitudes lat_{lower} and lat_{upper} with $lat_{\text{lower}} \leq lat \leq lat_{\text{upper}}$, we aim to determine the shadow map corresponding to lat. A simple method is to calculate the weighted mean of the matrices, multiplying each map by its respective weight and adding them cell-wise. This approach works well for small latitude differences but requires storing entire shadow map samples, as shown in Figure 11. To be able to determine shadow maps for all possible latitudes, we require a large number of samples with a small step size.

▶ Optical Flow Algorithms Given the predictable movement of the shadowed parts of the lookup table, we employed more complex methods than merely the weighted mean to reduce the added inaccuracy when interpolating between samples of shadow maps. Optical flow is the apparent speed and direction of brightness changes in an image. Visual computing algorithms can compute the optical flow to determine the direction and speed of movement for all pixels between two or more frames. We use this information and apply the computed movement with a weight factor, similar to a weighted mean. This method additionally captures the spatial movement between the shadow maps.



(a) Map at 90°N (b) Map at 75°N (c) Map at 60°N

(d) Map at 45°N (e) Map at 30°N (f) Map at 0°N

Figure 11: Shadow Maps showing the predictable transition of SF duration by a wind turbine from 90°N to 0°N. The shadow patterns shift from all cardinal directions at the poles to predominantly east-west near the equator.

3.2 Shadow Map Symmetries

We aim to exploit potential approximate symmetries in our shadow maps to reduce storage requirements and computational effort by avoiding the calculation and storage of redundant information. We examine symmetries along the north-south axis and the equator, mirroring shadow maps between the northern and southern hemispheres, and further evaluate their suitability in Section 5.5.

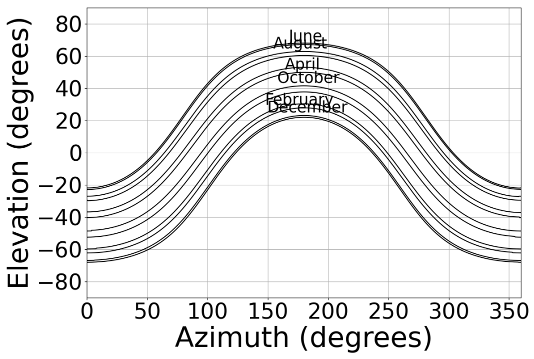
▶ East-West Symmetry We intend to reduce computational and storage requirements by exploiting an assumed symmetry along the north-south axis, mirroring the SF pattern within the individual shadow maps. We assume this symmetry since, for many locations and times of the year, the angle of sunlight at sunrise has a corresponding mirrored azimuth angle with a similar elevation at sunset, though this is not exact due to the Earth's tilt and orbital variations. In Figure 11 we can observe that the two-dimensional shadow maps seem mirrored along an axis from lower center to the upper center. By utilizing this assumed symmetry and storing as well as computing only one half of the shadow map, we halve the storage and computational complexity.

However, if we compute the shadow map using discrete samples of the solar position, as detailed in Section 1.1.1, we project many ellipses on the ground which may be slightly shifted and asymmetrical regarding both mirrored sides due the approximate symmetry of the azimuth and elevation not matching with the frequency of the solar samples. This discrepancy can introduce minor inaccuracies but remains manageable within the broader context of our storage optimization strategy. We validate the accuracy of exploiting this symmetry in Section 5.5.

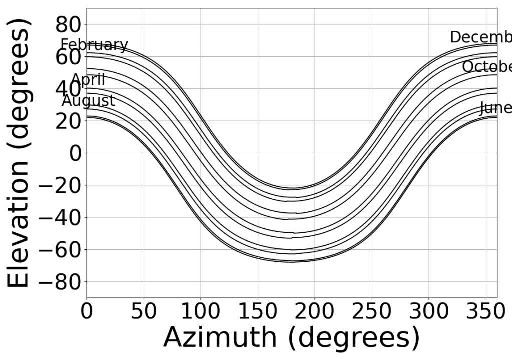
▶ Equatorial Symmetry To infer shadow maps for all latitudes, we need to store multiple samples ranging from 90°N to 90°S. We aim to reduce the range of samples required to infer all possible shadow maps by exploring an assumed approximate symmetry along the equatorial plane[8]. Hence, we investigated whether shadow maps at x°N are approximately equivalent to those at x°S when mirrored. If we exploit this assumed symmetry, we can store only half the amount of samples, potentially without losing the ability to infer shadow maps for all latitudes.

The solar path is approximately mirrored along the equator, as shown in Figure 12, when ignoring seasonal shifts. For instance, at latitude 45°N, the solar elevation peaks in the south, while at 45°S, it peaks in the north. Since the computation of SF is dependent on the solar position and we aggregate the results of the entire year and can thus ignore the seasonal shift, we can derive the symmetry within our shadow maps from that. We assume that mirroring a shadow map for latitude x°N along the east-west axis provides an estimate for the shadow map at x°S, and vice versa.

[8]The equatorial plane divides the Earth into the northern and southern hemispheres.



(a) Solar Path at 45°N



(b) Solar Path at 45°S

Figure 12: Solar paths for latitudes 45°N and 45°S, showing the relation between azimuth and elevation for the first day of each month in 2024. The north is azimuth 0°, the south is 180°.

In Section 5.5, we evaluate the error of exploiting the potential equatorial symmetry. The assumed symmetry of the solar path is not perfect due to variations caused by Earth's axial tilt and orbital eccentricity.

▶ Orbital Eccentricity Earth's orbit around the Sun is not a perfect circle but an ellipse. Subsequently, the distance between the Earth and the Sun varies throughout the year, with approximately 1.67% difference between the aphelion[9] and the perihelion[10] [8]. This variation primarily affects the length of the seasons [35].

▶ Earth's Axial Tilt The axis of Earth's rotation lacks 23.5 degrees from being fully perpendicular to the plane of the elliptic, which is an imaginary plane defined by the Earth's movement round the sun [5]. In combination with the orbital eccentricity, this tilt is the primary reason for the different seasons. The Earth orbits around the Sun and different parts of the planet are exposed to dissimilar amounts of sunlight throughout the year. When the northern hemisphere is tilted toward the Sun, it experiences summer, as sunlight reaches this region more directly. Conversely, when the southern hemisphere is more directly tilted toward the Sun, the northern hemisphere experiences winter.

▶ Solar Path Asymmetry As the length of the seasons varies due to Earth's orbital eccentricity, the topocentric solar path, which is dependent on the Earth's axial tilt relative to the movement around the sun, becomes asymmetrical regarding the hemispheres. Consequentially, the shadow maps at x°N and x°S are not perfectly symmetric.

[9]The point in Earth's orbit when it is at its greatest distance from the Sun.
[10]At the perihelion, the Earth's orbit is closest to the Sun.

3.3 Alternative Generative Methods

We aim to infer shadow maps for all latitudes without the need to save samples and interpolate between them. With the methods in Section 3.1, utilizing samples of high-resolution shadow maps, we require a substantial amount of storage. Further, transferring multiple shadow maps between computers seems impractical. In the following, we intend to combine the storage and the ability to determine shadow maps for all latitudes in one efficient data structure.

3.3.1 Polynomial Fit

We compute a polynomial for each grid point that models SF duration for that specific position as a function of latitude. To infer a shadow map for some latitude, we compute the function values for each grid point. For ease of notation, we summarize the individual polynomials $p_{xyz} : [-90, 90] \to \mathbb{R}$ for the grid points xyz as a function generating the entire shadow map, $f : [-90, 90] \to \mathbb{R}^{n \times n \times l}$, where n is the dimension of the horizontal axes and l is the number of layers, as described paragraph 2.1. With this approach, there is no need to store individual samples of shadow maps. Instead, we store the polynomials for the grid points and employ a data structure similar to the initial shadow map. Each grid point holds the polynomial coefficients $a_{xyz,k}$. When using NumPy arrays, as detailed in Section 4, all data points are required to have the same dimension. Thus, we choose the degree of our polynomials before training.

After choosing the fixed degree d as a hyperparameter, we aim to minimize the squared error of the function value $p_{xyz}(lat)$ and true SF duration $T_{lat,xyz}$ for some sampled latitudes lat, given by the error function E. This process generates the polynomial $p_{xyz}(x)$ for each grid point xyz.

$$E = \sum_{lat \in [-90,\ldots,90]} |p_{xyz}(lat) - T_{lat,xyz}|^2, \quad p_{xyz}(lat) = \sum_{k=0}^{d} a_{k,xyz} \cdot lat^k \tag{3.1}$$

In practice, we utilize the discrete SF T_{approx} instead of the true SF duration T, as detailed in Section 1.1.1. Throughout the different latitudes, the accuracy of T_{approx} might vary, depending on the sampling distance of the solar position, which potentially leads to oscillations in the training data. Since we flatten those inaccuracies, if our polynomial does not overfit, it may not perfectly match the data used for training.

▶ Runge's Phenomenon Typically, we assume that a higher degree polynomial reduces inaccuracies and allows us to get a better representation of the SF exposure. However, increasing the polynomial degree can lead to Runge's phenomenon, where the polynomial oscillates significantly between the sample points, especially near the edges of the training interval. Even though the representation of SF in the central region, near the equator, improves, the result close to the endpoints, at the poles, potentially oscillates [24]. Since we require an accurate representation of the SF and cannot alter our training interval $[90, -90]$, we aim to mitigate this potential error in the following.

3.3.2 Spline Interpolation

Using spline interpolation, rather than fitting high-degree polynomials, enables us to achieve a low interpolation error margin even though the splines themselves are based on low-degree polynomials [21]. We elucidate the concept of spline interpolation in Section 1.3. Employing natural cubic spline interpolation circumvents Runge's phenomenon. We explored this method since it typically leads to more stable and reliable results, especially at the extremities of the training interval, such as the latitudinal regions near the poles. This stability is crucial in applications where precision at the boundaries is as important as it is within the central regions [24].

Instead of the coefficients of a high-degree polynomial, as described in Section 3.3.1, each grid point holds the vector of knots, the B-spline coefficients, and the degree of the spline, adding one additional dimension to the architecture. To avoid saving all data points as knots with splines in-between, we generate a smooth representation of the SF duration for the grid point by applying the smoothing algorithm described by P. Dierckx [11]. Nevertheless, storing a reasonably precise spline representation of our function typically requires more storage than fitting a high-degree polynomial.

3.3.3 Generative Adverserial Method

When applying the methods described in Section 3.3.1 and Section 3.3.2, we must evaluate large polynomials or spline representations with many spline segments for each grid point. Both methods merely focus on each position's separate SF exposure without considering spatial information such as their surrounding values. They are computationally more feasible than calculating the SF from scratch, but we aim to reduce the necessary computations further. In the following, we introduce a method that is able to generate shadow maps without the need to evaluate function values for each grid point.

By considering shadow maps as gray-scale images where each grid point has a normalized value in the interval $[-1, 1]$, we can train a conditional generative adversarial network (CGAN), as described in Section 1.5, to learn a representation of the maps' structure given some latitude. We train a network for each layer of our shadow map separately to reduce the generation process to two-dimensional images and thus decrease the amount of data points we require for training. Nonetheless, CGANs still require a substantial amount of data points surpassing the number of samples we can compute in a reasonable time. We generated additional samples using weighted mean or optical flow to fill the spaces between samples to overcome this issue.

▶ Model Architecture The generator network takes a continuous label input, the latitude, transforms it through a series of dense layers, and reshapes it into a 15x15x256 feature map. Several transposed convolutional layers then process this feature map to increase the spatial resolution. Finally, the network produces an output through a convolutional layer to generate a single-channel image. As a demonstration of feasibility, we generate two-dimensional shadow maps with dimensions 60x60. The discriminator network, designed to distinguish between real and generated images, begins with convolutional and pooling layers that downsample the input image. It flattens the features and concatenates them with a dense representation of the continuous input. The combined features pass through several dense layers, before a final dense layer outputs the classification result. The discriminator learns to determine whether a shadow map is valid or forged and associated with the correct latitude.

The CGAN combines the generator and discriminator, feeding the generator's output along with the continuous input into the discriminator. The discriminator is set to non-trainable whenever we update the generator's weights during the CGAN training process. This architecture allows the generator to learn to produce realistic shadow maps conditioned on the label input.

▶ Training Process We train the discriminator and generator separately in each epoch, with a batch size of 48. The discriminator's loss is the sum of the loss when predicting the forged versus real samples. The generator's loss quantifies how well the discriminator can distinguish real images from fakes. In the first epochs, the generator learns that shadow maps have higher values in the center of the map while the outer edges experience none or less SF. The generator learns general structures and patterns within Shadow Maps in the following epochs. However, the model generates only a reduced set of possible shadow map patterns with insufficient quality due to mode collapse. This phenomenon is due to limited control over the discriminator. The current research presents two main approaches to solving this problem: using multiple generators or using a better and more precise metric to differentiate fake from authentic images [28].

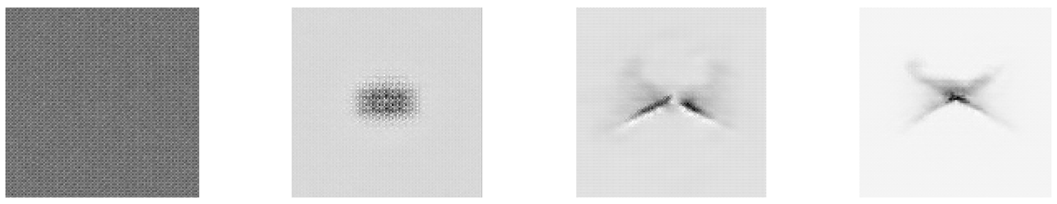

(a) Epoch 1 (b) Epoch 50 (c) Epoch 230 (d) Epoch 400

Figure 13: Evolution of generated shadow maps over training epochs for latitude $45°N$.

▶ WCGANs & Gradient Penalty Wasserstein-GANs (WGANs) are a variation of GANs that aim to improve training stability and mitigate issues such as mode collapse. Instead of using a discriminator that outputs whether an input image is real or faked, WGANs use a critic that scores the realness of the input. The loss function is based on the Wasserstein distance, providing smoother gradients and more reliable convergence. This modification allows the generator to receive more informative feedback, leading to a new improved objective function for the critic [14]. The objective encompasses a gradient penalty term. Essentially, this term encourages the critic to assign intermediate scores to samples between real and generated data.

We utilize a conditional WGAN with gradient penalty, passing latitude as input for the generator and additional input for the critic. Given this variation of the objective with the previous overall architecture of the generator and critic, we obtain higher-quality results and the generator returns maps for all latitudes, learning the association between patterns within shadow maps and the latitude.

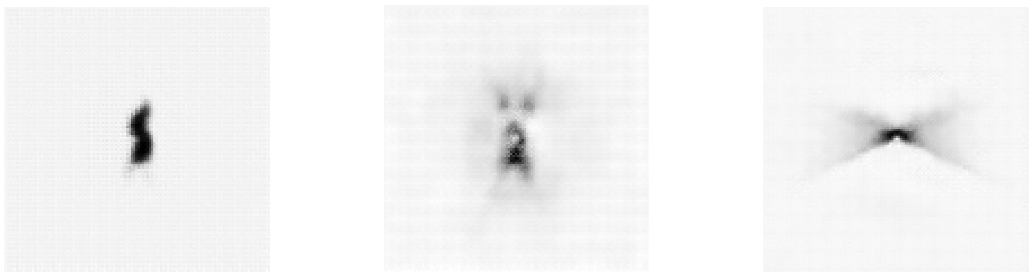

(a) Epoch 200 (b) Epoch 350 (c) Epoch 950

Figure 14: Evolution of generated shadow maps over training epochs using the WCGAN.

Once the training process is successfully completed and convergence is acceptably stable, our generator can approximate shadow maps using a single forward pass with the given latitude, allowing us to infer them with a significant speed-up.

We can discard the discriminator and retain only the generator with its weights. Based on the architecture we used for demonstration, the generator requires approximately 38 megabytes of memory. However, employing more lightweight generators can further reduce this requirement. For exemplary purposes, we trained a model that infers two-dimensional shadow maps, thus we need to train a model for each layer of our three-dimensional lookup tables. By incorporating additional parameters, such as ground height or using three-dimensional models, we can generate entire shadow maps in merely one forward-pass. Moreover, it is possible to include parameters such as different rotor diameters, allowing for further abstraction.

Despite computational limitations in generating large amounts of samples for training the GANs, we are able to produce exemplary shadow maps that closely resemble actual shadow maps. While they are not yet fully optimized for practical use, they demonstrate promising potential for further development.

3.4 Continuous Shadow Flicker

As a more precise alternative to the discrete compution of SF, as described in Section 1.1.1, we propose further research to adopt a continuous methodology. In the following, we broach our proposed approach.

Given two continuous functions, ϕ_{s} and θ_{s}, which return the solar azimuth and elevation angles, respectively, based on the hour h since the beginning of the year and geographical coordinates, latitude and longitude, we can integrate them into the definition of the sun vector and the turbine's rotor normal vector in Section 1.1.2. Subsequently, we can compute the distance between the intersection point and the turbine center for the given rotor and position of the receiver. Overall, this results in a continuous function $d : \mathbb{R}^9 \to \mathbb{R}$ with nine parameters.

We can fix the geographic coordinates, the positions of the turbine rotor, and the receiver as point on the ground, subtract the rotor radius from our constructed function, and find all roots of the to determine when shadowing either commences or ends. Visually, we identify exactly those instances where the intersection point of the traced vector with the turbine plane enters or exits the turbine rotor, touching its edges.

$$S = \{h \in \mathbb{R} \mid d(h) - r_{\mathrm{rotor}} = 0\}$$

Following from the geometric nature of the shadow's projection and the continuous movement of the sun across the sky, there must always be an even number of roots. We evaluate the function on the interval of each day of the year individually. Whenever we find no roots, we expect no shadowing and return a daily SF duration of zero. Otherwise, we add the distances between all pairs of roots $(h_i, h_{i+1}) \forall i \in \{0, 2, n-1\}$ and multiply with 60 minutes to get the duration of SF in minutes:

$$\sum_{i \in \{0,2,\ldots,n-1\}} (h_{i+1} - h_i) \cdot 60$$

Finding the roots of $d(h)$ analytically is infeasible. Instead, we propose the use of numerical solvers to approximate the solutions with arbitrary precision. We suggest the acceleration of function queries by compiling d with its nine parameters into efficient callable function code and fixing all but h for the solver. Additionally, we propose the use of domain knowledge about the earliest time of sunrise and latest time of sunset to limit the interval for the root solver.

In a typical setting, where only two or no roots lie within a single day, the worst-case number of roots to find is $365 \cdot 2$. Depending on the numerical root solver, we require an approximately constant number of function queries to find a single root with acceptable precision. Consequently, we decouple the computational complexity of the SF calculations from the sampling distance of solar sample.

4 Implementation

The content of this work is implemented in Python as part of an optimization software for wind farm layouts. By integrating our new methods, we aim to accelerate the computation of SF and, thus, evaluate the effect of turbine shutdowns in the optimization.

This section presents the details of implementing this work's data structures and algorithms. We elaborate the role of external tools and Python libraries required for running and accelerating our implementation. Even though our algorithms can be implemented more efficiently in low-level languages such as C++, we choose Python due to the large number of libraries facilitating computations such as the solar position.

4.1 Project Architecture

WindFarm3D is a wind farm project planning tool designed to improve siting layouts and is currently in development at our chair. It encompasses many modules, e.g., examining general site suitability, noise propagation, and SF emissions. Our implementation only concerns the project's SF module, which is mainly involved in the optimization process, the computation of energy loss due to turbine shutdowns, and the subsequent validation of the resulting layout to meet siting restrictions and regulatory constraints.

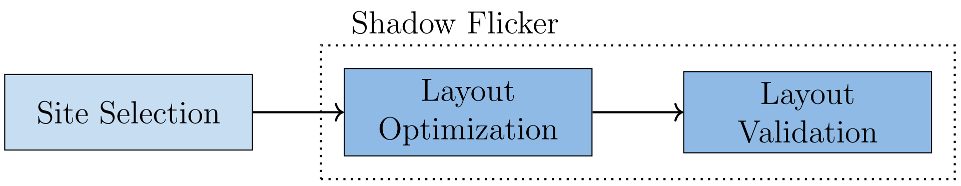


Figure 15: Relevant parts of the project architecture. The shadow flicker module concerns both the optimization and subsequent validation of wind farm layouts.

The program's workflow is divided into several logical steps. Initially, the user selects a target region for which they want to find an optimal wind farm layout. The system then automatically retrieves the geographical data for the selected region and discards unsuitable areas, e.g., those that do not meet the minimum required distance from residential buildings. Subsequently, we employ an optimization algorithm to generate a wind farm layout. The algorithm incorporates SF and noise propagation using pre-computed priority masks derived from shadow maps and lookup tables for noise propagation [37]. Further, the algorithm considers wake decay[11] using the Jensen model and minimum spacings between turbines. The project offers multiple optimization algorithms.

[11]Wake decay is the gradual reduction in turbulence intensity behind a wind turbine as the disturbed air stream returns to a more uniform state.

Finally, we validate the generated layout to ensure that it meets regulatory requirements, including maximum annual and daily SF exposure regarding all receiver positions in the vicinity. Using the results of the shadow cast simulation, we calculate the energy loss due to SF whenever a turbine must be shut down.

4.2 External Tools & Libraries

We use various external tools to compute shadow flicker. Most importantly, we utilize NumPy arrays for computing and storing shadow maps, allowing us to address sets of grid points more precisely with slicing, dimensional, and integer array indexing, returning subarrays without copying the underlying data [22]. This is particularly useful for implementing the tri-linear interpolation and, e.g., exploiting the symmetry of shadow maps. The arrays store the relative coordinates and SF duration using 32-bit floating points. However, for the spline interpolations, we store pickled tuples that hold the list of knots and coefficients due to different amounts of knots when applying smoothing. We store shadow maps using NumPy's npy-format or the corresponding compressed version.

▶ JIT & Parallelization To enhance the efficiency of iterative computations with our NumPy arrays, such as the ray backtracing with the solar samples, we employ JIT-compilation and parallelization with the Numba library. Numba is a just-in-time compiler for array-oriented computations that accelerates program execution compared to interpreted Python code. Utilizing Numba allows us to achieve performance comparable to that of compiled code [29]. Additionally, when computing the continuous shadow cast, we utilize JIT compilation to create a fast callable function for the intersection distance, thus decreasing the overhead of the numerical root solver queries.

▶ Curve Fitting We use several methods to fit smooth functions on data points within this work. To generate polynomials describing the SF duration as a function of latitude, we use NumPy's polyfit function. We call this function on a one-dimensional array holding the data points and receive all polynomial coefficients as return values. Subsequently, we utilize the poly1d method to evaluate the function values. We apply the splrep and splev methods from the SciPy library for spline interpolations. SciPy contains a broad pallet of algorithms for scientific computing [47]. Similarly to polyfit and poly1d, splrep generates a spline representation given a list of data points and some smoothing factor s. Subsequently, splev evaluates the returned spline representation for some x. Both NumPy and SciPy use Python wrappers for C-compiled code, enabling faster computation than possible with mere Python-interpreted code.

▶ Map Architecture We utilize NumPy arrays to store the spline representations of our grid points, as described in Section 3.3.2, requiring all fields to have the same dimension.

The smoothing algorithm within SciPy's splrep returns spline representations with different amounts of knots, depending on the complexity of the underlying function. For instance, a grid point with a value of zero for all latitudes would result in a small spline representation. Consequently, we store the tuples containing the knots, coefficients and degree as objects within our NumPy array.

► Optical Flow To determine the optical flow between two Shadow Maps, we utilize the OpenCV library. It contains a broad range of algorithms for visual computing tasks and provides an interface for the underlying C++ implementations [9]. Specifically, we utilize the calcOpticalFlowFarneback method that implements Gunnar Farneback's algorithm for dense optical flow of two single-channel images. The algorithm approximates the neighborhood of the two images with quadratic polynomials and examines how these polynomials change under translation. By observing the polynomial transformations, the algorithm estimates the displacement fields, which represent the motion of each pixel between the two frames [15].

► GAN For the Generative Adverserial Network (GAN), comprising the generator and discriminator, or alternatively critic for the WGAN, we utilize the Keras interface to define the architecture and loss functions. The flexibility of the Keras interface enables us to perform experiments with different layer configurations, activation functions, and optimization strategies. In the backend, we use Tensorflow, providing efficient data structures, fast implementations of various optimizers and GPU-support [30].

► Continuous Shadow Flicker We use the computer algebra system SageMath, built on top of Python, to generate the continuous shadow cast function from the various components [39]. We define the individual functions using symbolic variables and join them to the overall function d. SageMath simplifies the function and outputs the result as a string. We could use SageMath's very powerful numerical root solvers. However, evaluating each function query with symbolic variables causes an immense computational overhead. Hence, we utilize the returned function string, manually change some minor aspects of notation, and insert it into the source code to create an executable Python function. We JIT-compile this code to generate a fast callable function.

5 Validation & Case Study

In this section, we validate the results of our approximation method against the already existing implementation of the SF simulation. We assume the correctness of the project's implementation of the ray backtracing in Section 1.1.2 and the estimation of SF duration, as described in Section 1.1.1. We looked at the possible sources of inaccuracies that might cause us to over- or underestimate the SF exposure for a single receiver. In particular, we examined the error of interpolating SF duration within shadow maps, the aggregation of SF duration when considering multiple turbines, the estimation of the maximum daily SF for multiple turbines, and the suitability of the alternative methods to generate shadow maps more efficiently.

We utilized a Vestas V80-2.0MW wind turbine with a hub height of 100m and rotor diameter of 80m, similar to the validation approach by Rácz [37]. Further, aligning with German regulations, we disregarded SF exposure for instances with the solar elevation angle lower than 3 degrees, as outlined in Section 3.

5.1 Test Sites

Depending on the phenomena studied, we used terrains with varying properties, particularly the intensity of variations in surface elevation. For a single turbine, we position it at the center of the terrain segment. When simulating multiple turbines, we select terrain dimensions such that each turbine is at least half the maximum shadow distance, as defined in Section 2.1, far from the boundaries. We utilized terrain segments based on geographical data at the geodetic coordinates 50.86187°N, 6.085465°E.

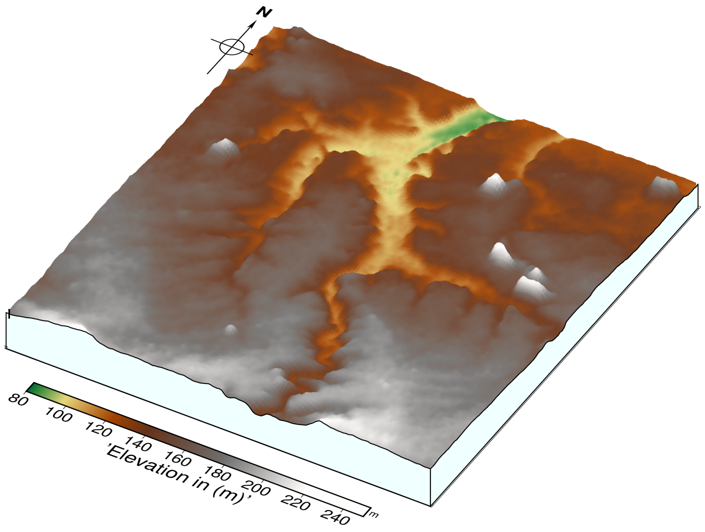
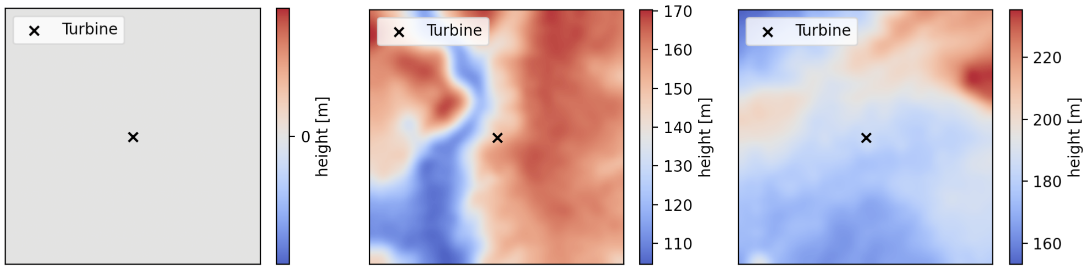


Figure 16: Surface contour of elevations within the coordinates 50.80721°N to 50.91653°N and 6.0065°E to 6.16443°E, encompassing the city Herzogenrath. The plot was generated using [45]. Dimensions are approximately $11 - 12$km.

To examine errors caused by height differences between emitter and receiver, we utilize the rough terrains in Figure 17. Both are sampled from the area in Figure 16 to represent a typical surface structure. The first terrain exhibits deviations of ±65 meters in elevation. The second terrain has a rougher surface structure with deviations of ±80 meters in elevation. Whenever the elevation is irrelevant, or we compare the magnitude of the error when including or disregarding the ground elevation, we utilize a flat terrain with a constant elevation of 0 meters at all positions.



(a) Flat Terrain (b) Realistic Terrain 1 (c) Realistic Terrain 2

Figure 17: Terrain profiles used for validation. The realistic terrain maps are both sampled from the geographic area shown in Figure 16. Dimensions are 3 kilometers on both axes.

If not stated otherwise, we validate using a shadow map with a horizontal resolution of 0.1 simulation points per meter on the horizontal axis and 0.1 simulation points per meter vertically. The highest layer is approximately 100 meters above ground height, and the lowest layer is approximately 100 meters below the ground.

5.2 Discretization Error

When we utilize a shadow map to estimate the SF exposure of a receiver relative to a single wind turbine, as described in Section 3, the projected position of the receiver within the map is likely not within the discrete set of grid points[12]. With our approach, we look up the nearest grid points and perform a tri-linear interpolation. This causes an error dependent on the resolution of the shadow map and the abruptness of changes in SF exposure for adjacent positions.

To evaluate the severity of this error, we ran two validation approaches. First, we utilized a three-dimensional grid with the same dimensions as our shadow map. For every cuboid formed by eight adjacent grid points of the test grid, we randomly chose a position within that cuboid. We used tri-linear interpolation to determine the SF for that position and, conversely, computed the actual SF exposure using ray backtracing. This ensures that we account for variations across the entire map, including extreme differences in elevation. We determined the average error for various resolutions of shadow

[12]In a dense coordinate space, the probability of randomly choosing a point that is within a discrete set of points is zero because they occupy no measurable area.

maps, as shown in Table 1. With a higher resolution, the average error considering all grid cells decreases. Furthermore, the error is highest in the vicinity of the turbine where we usually find no residential buildings, as indicated in Figure 18.

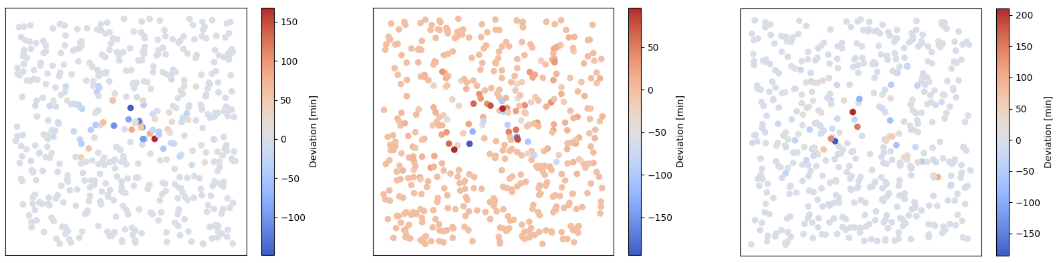
As a second approach, we used the terrains shown in Figure 17. We randomly selected points on the horizontal plane and determined their elevation from the terrain map. Similar to the first approach, we approximated the SF using tri-linear interpolation and compared it to the actual SF obtained via ray backtracing. We calculated the relative error by finding the absolute difference between the estimated and actual SF durations, summed over all points, and divided by the total actual SF.

Table 1: Average relative and absolute errors for annual SF approximation using different grid resolutions. Resolutions are given in simulation points per meter.

Vertical & Horizontal Res. $(\frac{1}{m})$	Avg. Rel. Error (%)	Avg. Error (min)
1/15	4.6728064	14.0679
1/10	3.0273383	8.9183
1/8	2.2420904	7.2906
1/7	2.1577294	6.4439

Table 2: Average relative and absolute errors for annual SF approximation regarding the terrains in Figure 17.

Error Metric	Flat Terrain	Realistic Terrain 1	Realistic Terrain 2
Avg. Abs. Error (min)	8.1399	7.6006	6.5744
Avg. Rel. Error (%)	3.2139	2.5405	2.4181



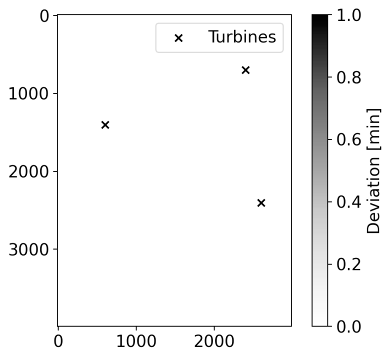
(a) Flat Terrain (b) Realistic Terrain 1 (c) Realistic Terrain 2

Figure 18: Absolute deviation (min) of approximated annual SF from actual SF exposure regarding different terrains from Figure 17.

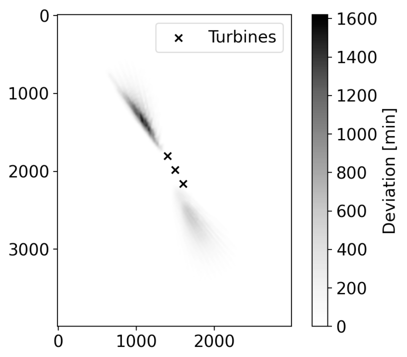
5.3 Additive Method Error

Potential inaccuracies arise when we aggregate the interpolated SF regarding a single receiver and multiple turbines, as described in Section 3. For a specific solar position, a receiver is either affected by SF or not, irrespective of the number of turbines contributing to the effect. Our method assumes that a receiver only experiences SF from one turbine simultaneously. Whenever this assumption is violated, an error occurs. This error is most likely to occur when turbines are positioned very closely together, and the sun is low in the sky, resulting in extended shadows. In practice, turbine placements are usually three to nine times the rotor diameter apart [19]. Under such conditions, the error of the annual SF duration is minimal. Regarding only this error, our method overapproximates the duration of SF.

We utilized two exemplary wind farm layouts to demonstrate the error. We aligned our shadow map with the positions where we examined the SF duration to avoid an additional interpolation error, as explored in Section 5.2.



(a) Scenario 1 with distances of approx. 1600 to 2000 meters.



(b) Scenario 2 with distances of approx. 200 to 400 meters.

Figure 19: Additive method error of annual SF under different turbine spacing scenarios. Scenario one (a) and Scenario two (b) demonstrate the small error, as described in Table 3.

Table 3: Average absolute and relative deviation caused by the additive method error for turbine layout scenarios in Figure 19.

Error Metric	Scenario 1 (realistic)	Scenario 2 (narrow)
Average Absolute Error (min)	0	16.88
Average Relative Error (%)	0.00	2.5272

We utilize two exemplary wind farm layouts to demonstrate the error. We align our shadow map with the positions where we examine the SF duration to avoid an additional interpolation error, as explored in Section 5.2.

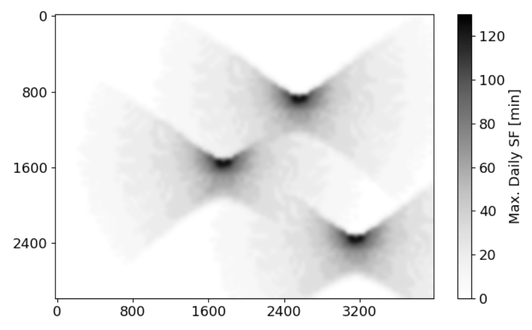
If two turbines are placed next to each other, the method error can emerge along the axis that directly connects the two turbines. This stands because when the solar azimuth aligns with this axis, the shadow from one turbine potentially overlaps and intersects with the shadow from the other turbine. Further, there is a margin around this hypothetical axis that is determined by the distance between the turbines and the potential length of the shadows in that direction, which occur when the sun is low in the sky.

Our findings indicate that the error of aggregating the annual SF for multiple turbines is not existent or negligible in realistic scenarios. In scenarios where wind turbines are positioned very closely, as shown in Figure 19b, the error is manageable with 2.5272%. Regarding the maximum daily SF duration, the error should not differ from the annual SF. However, when aggregating the maximum daily SF duration as a single value, we introduce large inaccuracies, as described in Section 5.4.

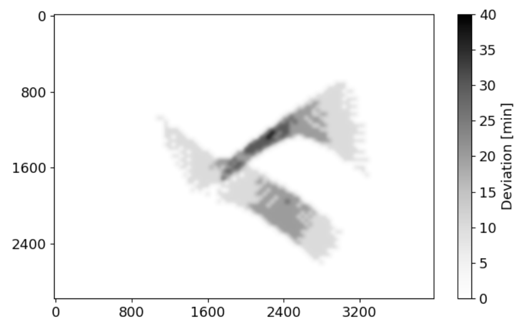
5.4 Maximum Daily Aggregation

In Section 2.3, we describe the concept of daily shadow maps that encompass an additional temporal axis, allowing us to obtain more precise results when computing the maximum daily SF for a receiver regarding multiple turbines.

We illustrate the error using an example wind farm layout with reasonably spaced turbines. The area significantly impacted by maximum daily SF is much larger in comparison to the annual SF. This leads to more intersections and thus widespread errors. Maximum daily SF affects a larger area since it doesn't vary much between frequently and infrequently shadowed areas throughout the year. Figure 20 displays the area covered by maximum daily SF and the corresponding error. We utilize the same definition of the relative error as in Section 5.2.



(a) Accurate maximum daily SF for three turbines.



(b) Difference of actual max. daily SF and aggregated values.

Figure 20: The aggregation of the maximum daily SF causes an avg. relative error of 9.9033% and avg. absolute error of 2.112 minutes. Dimensions are in meters.

Due to the high additional relative error of 9.9033%, as shown in Figure 20, we do not recommend to aggregate maximum daily SF values of multiple turbines. However, the error vanishes with our alternative map architecture, as outlined in Section 2.3. With the alternative architecture, the maximum SF exposure can still be affected by the errors described in Section 5.2 and Section 5.3.

5.5 Map Interpolation & Symmetries

We introduced several methods to efficiently derive shadow maps without computing the SF for each grid cell, as described in Section 3. Our goal is to evaluate how much these inferred shadow maps deviate from the actual ones to determine which approaches are suitable in realistic scenarios.

Weighted Mean

To estimate shadow maps between latitude steps, we used a weighted mean for each grid point independently, as detailed in Section 3.1. This simple method doesn't account for spatial relationships like adjacent grid points or changes in SF patterns across latitudes. To validate, we utilized the shadow maps from 90°N to 90°S in integer steps and determined the map at x using weighted mean with the maps from $x-1$ and $x+1$.

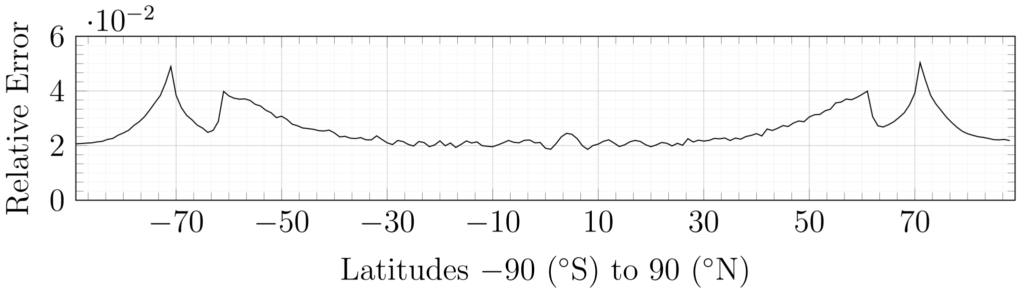


Figure 21: Relative error in annual SF when utilizing weighted mean between maps at latitude $x-1$ and $x+1$ to compute shadow map for latitude x.

We computed the relative error by subtracting the generated map from the actual map, summing the absolute differences, and dividing by the sum of the actual map at latitude x. We can significantly reduce the error by choosing a smaller step size between the shadow map samples. In practice, we can increase the sampling distance for intervals of latitudes with greater relevance.

In Figure 21, we can see that the error differs between distinct intervals of latitude. In proximity to the poles, the error is low. Around latitudes 70°N and 70°S, the error peaks and decreases when moving closer to the equator. Overall, the error ranges from approximately 2% up to 5%. If we require precise shadow maps for latitudes higher than 50°N and 50°S, we can increase the sampling distance for these regions to decrease the error when interpolating.

Polynomial Fitting & Spline Interpolation

In Section 3.3.1, we fitted high-degree polynomials to model the SF exposure of each grid point as a function of latitude. Alternatively, in Section 3.3.2, we utilized spline interpolation to avoid Runge's phenomenon. With these two approaches, we intended to minimize storage by avoiding the need to store shadow maps for multiple latitude steps. However, neither approach can represent the shadow maps used as training data without introducing inaccuracies based on the degree of the polynomials or the smoothing factor for the spline representations. To validate the severity of the added inaccuracies, we applied both methods with shadow map samples at even latitude steps between 90°N and 90°S as training data. Further, we derived shadow maps at all integer latitude steps and compared them with the actual shadow maps for the respective latitudes. We utilized different degrees and smoothing factors to demonstrate the development of the errors and the trade-off between accuracy and dimension, as shown in Table 5 and Table 6.

This trade-off between the dimension of the map of polynomials and the accuracy of the derived shadow maps, as shown in Table 5, is characterized by the degree and the error metrics. Storing a shadow map that utilizes polynomials with degree d to describe SF as a function of latitude for each grid point is comparable to storing shadow maps for d latitudes when comparing the spatial requirements. We cannot describe the relationship as easily for the spline interpolation since the returned spline representation depends on the complexity of the underlying function in combination with the smoothing factor. However, a significant advantage is that we describe cells that constantly experience no SF with very simple spline interpolations. Contrarily, we would fit a high-degree polynomial for these cases, irrespective of the function's complexity.

The error in Table 5 grows smaller as the degree increases. However, this trend gets increasingly slower considering the average and the maximum error. Overall, we do not recommend using polynomials to derive shadow maps. To reduce the error to a reasonable level, we require polynomials with a substantially high degree, defeating the purpose of reducing storage. On the other hand, spline interpolation seems to be a more suitable alternative. The error is generally much lower in comparison to the polynomial approach, as shown in Table 6. Further, the error only marginally increases as we choose a higher smoothing factor.

Table 4: Computing time of a shadow map with horizontal dimensions of 3km and −100m to 100m vertically.

Resolution $(\frac{1}{m})$	**Computation** [sec]	**Polynomial Evaluation** [sec]
1/25	5781	2.434
1/50	849	0.871

Table 5: Average and maximum relative error percentages for polynomial fitting with varying degrees. Higher-degree polynomials provide a closer fit to the shadow map data, reducing both the average and maximum relative errors. However, the rate of error reduction diminishes at higher degrees.

	Relative Error (%)		
Degree	**Average**	**Maximum**	**Minimum**
5	28.9697	35.1406	11.3409
8	23.5409	30.0814	7.5750
10	19.4858	23.1586	7.5575
11	18.5317	22.6815	4.6813
14	14.9567	18.7149	4.8915
15	14.4683	18.5205	4.8581
25	9.0379	11.7616	3.1140
30	7.5702	10.2662	2.6355
35	6.2874	8.4338	1.8340

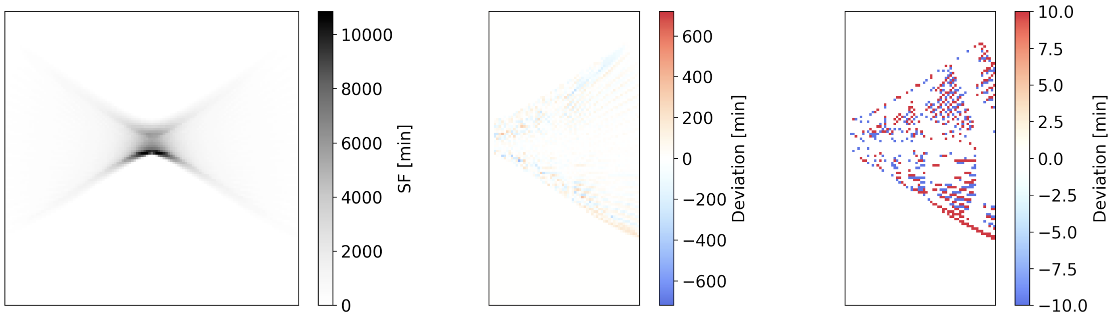
Table 6: Relative error metrics for spline interpolation with varying smoothing factors. Mere interpolation has the smallest error. The error decreases as the smoothing factor increases up to a certain point.

	Relative Error (%)		
Smoothing Factor	**Average**	**Maximum**	**Minimum**
0 (Interpolation)	1.0645	2.1847	0.0000
50	2.6269	8.1133	0.0801
75	1.1738	2.4465	0.0935
100	1.1356	2.2193	0.1056
150	1.1384	2.2044	0.1266
200	1.1481	2.2069	0.1423

Accuracy of Symmetries

In Section 3.2, we examined assumed symmetries within shadow maps or their samples across latitudes. These symmetries, while useful for reducing computations and storage, introduce errors due to their imperfectness. We aim to quantify the difference between the reconstructed and actual shadow maps.

▶ East-West Symmetry For the east-west symmetry, we assumed a mirrored SF pattern along the north-south axis in the middle of our shadow maps. As shown in Figure 22, we computed the difference between both halves of a shadow map considering the annual and maximum daily SF.



(a) Annual SF for 45°S (b) Annual Difference (c) Max. Daily Difference

Figure 22: Reducing spatial complexity in shadow maps by exploiting sunrise-sunset symmetry halves storage needs while preserving essential data. Relative errors are 2.562% annually and 4.585% daily, given a map for 45°S with a resolution of three meters and time steps of 10 minutes.

To compute the error between the constructed shadow map, utilizing symmetry, and the original map, we first subtract the original map from the constructed map and take the absolute value. We then sum the absolute differences across all cells and divide by the sum of the original matrix, yielding a relative error of 2.562% for the annual SF and 4.585% for the maximum daily SF, as show in Figure 22. Alternatively, by dividing the sum of absolute differences by the number of cells, we obtain the average deviation of 0.3592 minutes for the maximum daily shadow flicker and 6.5854 minutes annually.

▶ Equatorial Symmetry We assumed a symmetry between samples of shadow maps along the equator such that shadow maps at x°N are identical to those mirrored at x°S. We show that this assumed equatorial symmetry in Section 3.2 is not fit for use. Given samples of shadow maps for latitude steps of one, we compared the differences between shadow maps at $x°N$ and $x°S$ for all integer steps of latitude. We used maps with a resolution of approximately 0.07 simulation points meters on the horizontal axes. Furthermore, we validated the annual SF at ground height with 0 meters.

To measure the relative difference, we subtracted both maps cell-wise, took the sum of the absolute values, and divided them by sum of the SF values from the respective map in the northern hemisphere. For all samples, the deviation ranges from 6.4319% to 7.9536% with an average of 7.1598%.

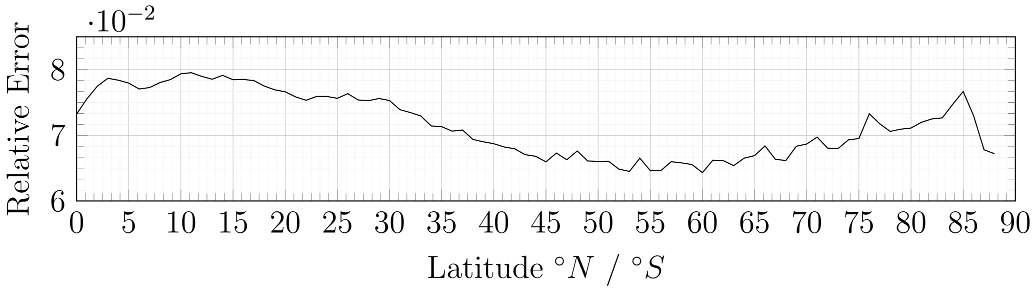
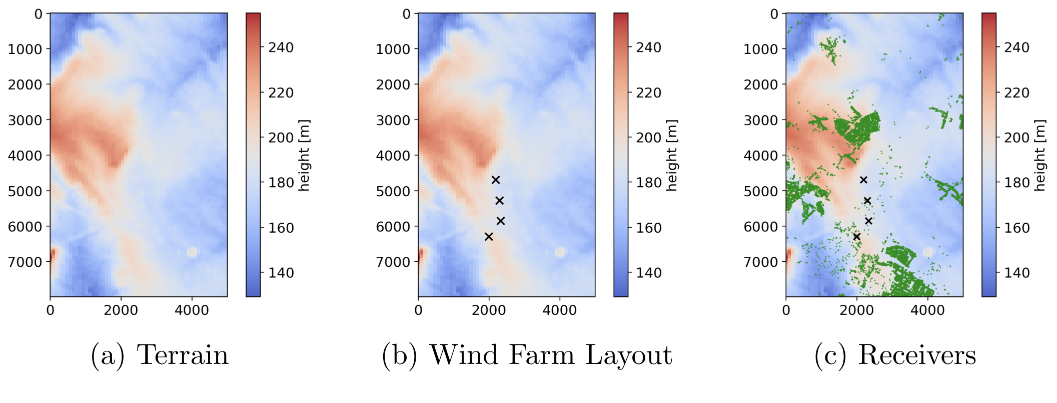


Figure 23: Comparing hemispheric differences under equatorial symmetry assumptions.

In paragraph 3.2, we elaborated on a possible reason for the imperfectness of this assumed symmetry, causing the high error. Consequently, we do not recommend to use this symmetry to reduce the dimension of shadow maps.

5.6 Case Study Herzogenrath

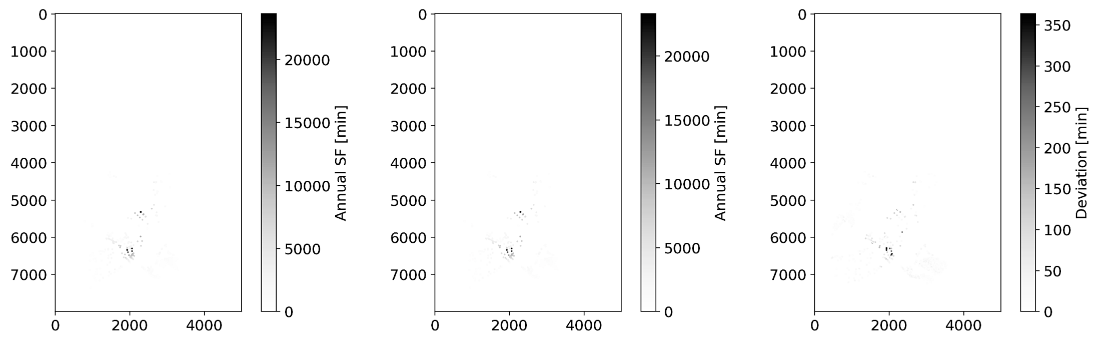
To illustrate the error of our approximation approach in a realistic setting, we demonstrate the computation of SF for a wind farm layout and multiple receivers using the geographical information of the German city Herzogenrath. We simulate the SF regarding four wind turbines and 10236 receivers. We presume a wind farm layout, as shown in Figure 24b, and compute the SF exposure for the receivers marked in Figure 24c. We compute the SF using a shadow map with $\frac{1}{7}$ grid points per meter on all axes, conversely, ray backtracing for the individual positions with the exact solar position.



(a) Terrain (b) Wind Farm Layout (c) Receivers

Figure 24: Realistic wind farm layout showing terrain and building data for the German city Herzogenrath. Dimensions are in meters.

In Figure 25, we show that we can approximate the SF for a large set of receivers with only a negligible additional error in a realistic setting. In our example, we get an average error of approximately 2.2467% regarding the total exposure of SF for all receivers.



(a) Approximation (b) Actual SF (c) Difference

Figure 25: Approximated and actual duration of SF exposure for the wind farm layout in Figure 24. The additional relative error is 2.2467%.

▶ Performance On a 3.2Ghz ARM-based CPU with eight cores, the approximation of the annual SF regarding the four turbines and 10236 receivers takes approximately 1.72 seconds. If we compute the actual SF with a sampling frequency of 10 minutes for the solar position, we require approximately 694.50 seconds on the same system with our current implementation. Hence, our approach accelerates the SF computations significantly.

We assume a similar computational performance of our approach with the maximum daily SF. We can disregard the error of aggregating maximum daily SF as single values if we utilize the shadow map architecture in Section 2.3. Furthermore, there are no differences in the likelihood of discretization and method error. Subsequently, we assume a similar error for the maximum daily SF.

Conclusion and Outlook

As this work's principal achievement, we introduced a new algorithm for estimating SF exposure using three-dimensional lookup tables in Section 2. We can estimate SF for a receiver regarding a single turbine or a wind farm encompassing multiple turbines. The novel approach enables us to, e.g., include SF more efficiently in optimizing and generating wind farm layouts. We have shown in Section 5.2 that our approach yields no or negligible method errors in realistic layouts. Further, the discretization error can be reduced using shadow maps with higher resolutions. Due to the high computational efficiency of our approach, we can compute the SF exposure for a large number of receivers instead of a reduced set of representative receivers. Moreover, we have pursued multiple approaches in Section 3 that allow us to infer shadow maps for arbitrary latitudes without the need to compute SF exposure for each grid point individually. This makes the estimation of SF even further feasible and prevents extensive computational overhead. Additionally, we examined the spatial properties of our shadow maps and deemed an approximate symmetry along the north-south axis suitable to reduce the memory requirements of shadow map samples.

Future Work

More research needs to be done concerning different methods to incorporate SF in the optimization and, consequently, the trade-off between losses in AEP[13] and keeping SF emissions as low as possible. To achieve that, we require fast computations of SF, both the annual and the maximum daily exposure. In the following, we broach relevant fields where we deem a need for further research.

We utilize the method in Section 1.1.2 to compute a shadow map. Hence, for each grid cell, we check whether SF occurs for every solar position sample. Given the solar position, we could significantly accelerate the computations by selecting cells as candidates when they lie in an area that qualifies for shadowing. After selecting the candidates, we could precisely check for SF using ray backtracing. We suggest further research incorporating this candidate preselection but also regards differences in elevation and subsequently does not introduce a new source of inaccuracy.

In Section 2.3, we describe an alternative architecture for shadow maps that incorporate an additional temporal axis to avoid errors when aggregating maximum daily SF for multiple turbines. However, this method significantly expands the dimensions of our shadow map. We suggest further research to find a better trade-off between accuracy, spatial, and computational complexity of estimating the maximum daily SF.

Due to the high computational complexity of precomputing shadow maps, we were merely able to generate shadow maps with a resolution of $\frac{1}{7}$ grid points per meter in all

[13]Annual Energy Production (AEP) is the total amount of produced energy of a wind farm.

dimensions. Consequently, we propose further simulations to determine the discretization error of high-resolution shadow maps.

Computing the exact SF exposure remains computationally inefficient, especially with a high sampling resolution of the solar position. Conversely, small deviations due to the discreteness can accumulate over a year with a lower sampling resolution, leading to potential misestimations about whether a turbine exceeds SF limits. As a potential solution to that problem, we have broached a continuous approach to compute SF in Section 3.4 that is independent of the sampling resolution. We suggest more research in that direction to make SF computations more efficient and precise at the same time.

We utilized different approaches, such as polynomial curve fitting and spline interpolation on the individual grid points, and generating an entire shadow map using a generative neural network to make the computation of shadow maps more feasible and avoid a significant computational overhead. For the neural network, we suggest hyperparameter optimization and experimentation with different architectures and parameters to reach the full potential of this method. Additionally, we propose further research on alternative methods that might capture the spatial transformation of the SF patterns between different latitudes in a better and more direct manner.

References

[1] Acero, J. F. H. et al. “A Review of Methodological Approaches for the Design and Optimization of Wind Farms”. In: *Energies* 7.11 (2014), pp. 6930–7016.

[2] Azlan, F. et al. “Review on optimisation methods of wind farm array under three classical wind condition problems”. In: *Renewable and Sustainable Energy Reviews* 135 (2021), p. 110047.

[3] Baños, R. et al. “Optimization methods applied to renewable and sustainable energy: A review”. In: *Renewable and Sustainable Energy Reviews* 15.4 (2011), pp. 1753–1766.

[4] Bengio, Y. and Lecun, Y. *Convolutional Networks for Images, Speech, and Time-Series*. 1997.

[5] Boon, J. D. “The Tilt Of The Earth’s Axis And The Consequences Thereof”. In: *Field and Laboratory* 13.1 (1945). URL: `https://scholar.smu.edu/fieldandlab/vol13/iss1/2`.

[6] Bund-/Länder-Arbeitsgemeinschaft für Immissionsschutz (LAI). *Hinweise zur Ermittlung und Beurteilung der optischen Immissionen von Windkraftanlagen - Aktualisierung 2019*. 2020.

[7] Canelhas, D. R., Stoyanov, T., and Lilienthal, A. J. “A Survey of Voxel Interpolation Methods and an Evaluation of Their Impact on Volumetric Map-Based Visual Odometry”. In: *Journal of Computer Vision and Image Processing* 12.3 (2024). UnivrsesAB, Sweden; Center of Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden, pp. 123–145.

[8] Chiang, J. C. H. and Broccoli, A. J. “Orbital eccentricity and Earth’s seasonal cycle”. In: *PLOS Climate* 3.7 (2024), pp. 1–4. URL: `https://doi.org/10.1371/journal.pclm.0000436`.

[9] Culjak, I. et al. “A brief introduction to OpenCV”. In: *2012 Proceedings of the 35th International Convention MIPRO*. 2012, pp. 1725–1730.

[10] Deutsche WindGuard GmbH. *Status of Onshore Wind Energy Development in Germany*. 2023.

[11] Dierckx, P. “An algorithm for smoothing, differentiation and integration of experimental data using spline functions”. In: *Journal of Computational and Applied Mathematics* 1.3 (1975), pp. 165–184.

[12] Dobrić, G. and urišić, Ž. “Double-stage genetic algorithm for wind farm layout optimization on complex terrains”. In: *Journal of Renewable and Sustainable Energy* 6.3 (2014), p. 033127.

[13] Dos Passos, W. *Numerical methods, algorithms and tools in C*. 2016. URL: `https://doi.org/10.1201/9781420007602`.

[14] Fabbri, C. “Conditional Wasserstein Generative Adversarial Networks”. In: *University of Minnesota* (2024).

[15] Farnebäck, G. *Two-Frame Motion Estimation Based on Polynomial Expansion.* 2003.

[16] Ghayoumi, M. *Generative Adversarial Networks in Practice.* 1st. New York, 2023, p. 670. URL: `https://doi.org/10.1201/9781003281344`.

[17] González, J. S. et al. "A review and recent developments in the optimal wind-turbine micro-siting problem". In: *Renewable and Sustainable Energy Reviews* 30 (2014), pp. 133–144.

[18] Grossi, E. and Buscema, M. "Introduction to artificial neural networks". In: *European journal of gastroenterology hepatology* 19 (2008), pp. 1046–54.

[19] Gupta, N. "A review on the inclusion of wind generation in power system studies (Elsevier- Impact Factor- 10.556)". In: *Renewable and Sustainable Energy Reviews* 59 (2016), pp. 530–543.

[20] Haac, R. et al. "In the shadow of wind energy: Predicting community exposure and annoyance to wind turbine shadow flicker in the United States". In: *Energy Research Social Science* 87 (2022), p. 102471.

[21] Hall, C. A. and Meyer, W. W. "Optimal error bounds for cubic spline interpolation". In: *Journal of Approximation Theory* 16.2 (1976), pp. 105–122.

[22] Harris, C. R. et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362. URL: `http://dx.doi.org/10.1038/s41586-020-2649-2`.

[23] Jay Haley, P. *Pronghorn Flats Wind Farm Shadow Flicker Analysis Banner and Kimball Counties, NE.* Technical Report. 3100 DeMers Ave., Grand Forks, ND 58201: EAPC Wind Energy, 2020. URL: `https://www.eapc.net`.

[24] Jianan, Z. et al. *A case study of the Lunger phenomenon based on multiple algorithms.* 2023. arXiv: `2311.11253 [math.NA]`.

[25] Khanuja, S. S. and Khanuja, H. K. "GAN Challenges and Optimal Solutions". In: *International Research Journal of Engineering and Technology (IRJET)* 8.10 (2021). Sanjeet Singh Khanuja is a Software Cloud Architect, Pune, India and Harmeet Kaur Khanuja is an Associate Professor, Department of Computer Engineering, MMCOE, Pune, India. URL: `http://www.irjet.net`.

[26] Knopper, L. D. et al. "Wind turbines and human health". In: *Front. Public Health* 2 (2014), p. 63.

[27] Koppen, E., Gunuru, M., and Chester, A. "International Legislation and Regulations for Wind Turbine Shadow Flicker Impact". In: *Proceedings of the 7th International Conference on Wind Turbine Noise.* Arcadis Netherlands (formerly), Arcadis United Kingdom. International Conference on Wind Turbine Noise. Rotterdam.

[28] Kushwaha, V., Nandi, G., et al. "Study of prevention of mode collapse in generative adversarial network (GAN)". In: *2020 IEEE 4th Conference on Information & Communication Technology (CICT).* IEEE. 2020, pp. 1–6.

[29] Lam, S. K., Pitrou, A., and Seibert, S. "Numba: a LLVM-based Python JIT compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM '15. Austin, Texas, 2015. URL: `https://doi.org/10.1145/2833157.2833162`.

[30] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[31] McKinley, S. and Levine, M. "Cubic spline interpolation". In: *College of the Redwoods* 45.1 (1998), pp. 1049–1060.

[32] Metzler, R. C. "On Riemann Integrability". In: *The American Mathematical Monthly* 78.10 (1971), pp. 1129–1131. URL: `https://doi.org/10.1080/00029890.1971.11992961`.

[33] Ning, A., Dykes, K., and Quick, J. *Chapter 7: Systems Engineering and Optimization of Wind Turbines and Power Plants.* Tech. rep. NREL/CH-5000-73325. Stevenage, UK: Institution of Engineering and Technology, 2019.

[34] O'Shea, K. and Nash, R. *An Introduction to Convolutional Neural Networks.* 2015. arXiv: `1511.08458 [cs.NE]`.

[35] Oostra, B. "Introducing Earth's Orbital Eccentricity". In: *The Physics Teacher* 53 (2015), pp. 554–556.

[36] Quentin, J. *Hemmnisse beim Ausbau der Windenergie in Deutschland.* 2019.

[37] Rácz, F. T. B. *Wind Farm Layout Optimization near Urban Areas Using Approximated Shadow Flicker and Noise Emissions.* Bachelor of Science Thesis. 2023.

[38] Roscher, B., Schelenz, R., and Schröder, K.-U. "Multi-dimensionale Windparkoptimierung in der Planungsphase". PhD thesis. Verlagsgruppe Mainz, 2020.

[39] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.4).* `https://www.sagemath.org`. 2024.

[40] Schellong, W. and Weimbs, M. *Site-planning and -optimizing of wind farms.* Tech. rep. Tech. rep., 2009.

[41] Shakoor, R. et al. "Wake effect modeling: A review of wind farm layout optimization using Jensen's model". In: *Renewable and Sustainable Energy Reviews* 58 (2016), pp. 1048–1059.

[42] Shourangiz-Haghighi, A. et al. "State of the Art in the Optimisation of Wind Turbine Performance Using CFD". In: *Archives of Computational Methods in Engineering* 27.2 (2020), pp. 413–431.

[43] Tareen, S. K. and Khan Tareen, F. *Convolutional Neural Networks for Beginners.* 2023.

[44] Tesauro, A., Réthoré, P.-E., and Larsen, G. C. "State of the Art of Wind Farm Optimization". In: *European Wind Energy Conference and Exhibition 2012 Proceedings.* 2012.

[45] Tian, D. et al. *PyGMT: A Python interface for the Generic Mapping Tools.* Version 0.12.0. 2024. URL: `https://doi.org/10.5281/zenodo.11062720`.

[46] Verkuijlen, E. and Westra, C. “Shadow hindrance by wind turbines”. In: *Verkuijlen, CA Westra: Proceedings of the European wind Energy Conference (October 1984), Hamburg, Germany.* 1984.

[47] Virtanen, P. et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272.

[48] Xu, Y. and Xu, R. *Research on Interpolation and Data Fitting: Basis and Applications.* 2022. arXiv: `2208.11825`.