Continuous Sky View Factor Calculations Using a Parallel GPU Workflow

Max van der Waal(✉) and Daniela Maiullari

Delft University of Technology, 2628 BL Delft, The Netherlands
m.vanderwaal@tudelft.nl

Abstract. The Sky View Factor (SVF) is a key parameter in urban climate modelling, which quantifies the fraction of the visible sky from a given point and allows for the estimation of incident solar radiation, thermal comfort, and urban heat distribution. High resolution in SVF computation is essential for microclimatic studies in the canopy layer, where detailed representations of urban environments are crucial for understanding variations in heat exposure. Traditional SVF calculation methods often rely on sequential processing of shadow projections, however, these methods are computationally intensive and time-consuming, particularly for urban climate analyses at high resolution.

These computational challenges are further magnified when incorporating complex components such as vegetation, where tree crowns exhibit intricate geometries, partial transparency, and permit sky visibility from beneath the canopy. These properties require detailed modeling to account for light penetration and obstruction. This significantly increases the computational cost of Sky View Factor calculations and extends runtime to hours or even days, depending on scale and resolution.

This study introduces a novel GPU-accelerated ray tracing approach for SVF calculation, designed to address the computational limitations of traditional methods for large-scale analyses. By utilizing NVIDIA GPUs and the CUDA programming framework, the method applies parallel computing to perform ray tracing across the full range of azimuth and altitude angles. It estimates SVFs by systematically weighting blocked rays based on their spatial contributions to the hemisphere.

The accuracy of the developed method is validated through two complementary approaches. First, modelled SVF values are compared against theoretical expectations derived from idealized geometric environments. Additionally, a test case on a neighbourhood in Rotterdam is conducted to compare the results of the developed method against those obtained using an established SVF estimation technique using a serial approach. In addition to accuracy, computational efficiency is evaluated by comparing processing times across different study area extents with those of a CPU-based implementation. The proposed GPU workflow achieves a 99% reduction in processing time compared to traditional shadow casting methods performed on a CPU, while maintaining similarly high resolution and accuracy.

Keywords: Sky View Factor · PyCUDA · GPU computing · Ray Tracing · Urban Climate Modelling · Urban Form

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025, LNCS 15898, pp. 388–398, 2026.
https://doi.org/10.1007/978-3-031-97657-5_24

1 Introduction

Urban form strongly influences urban climate at the micro and local scale [1, 2], by impacting thermal and aerodynamic processes, contributing to the trapping of solar energy and anthropogenic heat, increasing thermal storage, and reducing wind speed and evaporative cooling [3]. These mechanisms explain why air and surface temperatures are higher in cities than in rural areas, a phenomenon known as the Urban Heat Island (UHI) effect.

In observational and modelling climate studies, a conspicuous number of morphological attributes and descriptors have been used to better understand these multiple relationships with energy and heat exchanges. Among these, vertical openness is a form attribute that conveys the degree of the urban fabric's openness to the sky and determines the amount of incoming shortwave radiation as well as the amount of long-wave radiation returned to the sky [4].

The main parameter that describes vertical openness is the Sky View Factor (SVF), defined as the proportion of sky that is visible at a given location. In quantitative terms, SVF captures the three-dimensional structure of the built environment as a two-dimensional metric [5], expressed as a unitless value between 0 (completely obstructed sky) and 1 (fully unobstructed sky). This parameter effectively quantifies how urban morphology modifies incoming radiation fluxes, directly influencing thermal conditions experienced in cities.

In urban climate studies, spatially explicit representations of SVF values across an entire urban area are critical to modelling urban microclimate and to better understand the distribution of heat exposure. These continuous SVFs are calculated for each pixel within the study region, creating a continuous map of values. Unlike discrete point measurements, these maps reveal patterns of sky view obstruction throughout the modeled environment. By capturing this spatial continuity, researchers can identify patterns and relationships between urban form, climate, and sky view obstruction that would not be apparent from isolated point measurements. This comprehensive spatial perspective helps urban planners and designers make informed decisions about urban compositions, street orientation, and public space design to optimize thermal comfort and mitigate urban heat island effects.

However, the generation of continuous SVF datasets at large scales proves to be computationally expensive and remains a critical challenge in urban climatology. Calculating the SVF for a single location is computationally demanding, as it involves aggregating shadow determinations across a range of azimuth and altitude angles to evaluate sky visibility. This demand increases substantially when the process is scaled to every pixel in a study area to generate continuous SVF maps [6]. High-resolution SVF calculations often require hours of processing time due to the large number of computations involved. This issue limits the integration of SVF-driven analyses into iterative urban design workflows and constrains the spatial extent of heat island studies to localized areas rather than city-wide or regional scales [7].

This study addresses the challenge of reducing the processing time of continuous SVF estimations by proposing a novel GPU-accelerated ray tracing approach. The paper begins with a concise review of existing SVF estimation techniques, outlining their

respective benefits and limitations. It then describes the implementation of the proposed GPU-based workflow, along with 3 methods to assess its computational performance. The outcomes of these evaluations are used to demonstrate the method's accuracy and efficiency, followed by a discussion of its practical implications and potential applications.

1.1 Methods for Estimating SVFs

Over the past decades, efforts to quantify the visible sky from urban surfaces have led to a wide array of methods. Traditional techniques such as shadow-casting and ray tracing have laid the groundwork for computational SVF estimation, while newer approaches, such as synthetic fisheye algorithms, offer alternative perspectives that leverage advances in rendering and image-based analysis.

Following the shadow casting method, the SVF is estimated by simulating the obstruction of virtual light sources distributed across a hemispherical dome [8, 9]. Using high-resolution digital elevation models (DEMs), this method projects light rays from predefined azimuth and altitude angles to determine sky visibility. CPU-based implementations are computationally intensive, as they require iterating through thousands of light positions and calculating shadows for each DEM cell. Processing a 1 km2 urban area at 1-m resolution can take hours on standard CPUs, limiting scalability for city-wide analyses. Despite these constraints, shadow casting remains valued for its geometric accuracy and compatibility with coarse-resolution DEMs.

Ray tracing methods estimate SVFs by tracing a multitude of imaginary rays from ground points to the sky dome and calculating the fraction of unobstructed rays. This method, exemplified by tools like HORAYZON [10], processes high-resolution results by iteratively checking intersections between rays and elevations of 3D features. While highly accurate, CPU-based applications struggle with memory management for large datasets, often resorting to terrain simplification or subsampling to reduce computational load [10]. Despite these challenges, ray tracing on CPU achieves sub-0.05 RMSE in validation against fisheye photography [11], making it one of the most reliable methods for estimating continuous SVFs.

Since the introduction of platforms such as Google Street View, the amount of data and access to street-level imagery has grown exponentially. Synthetic fisheye methods generate SVF estimates by creating (artificial) hemispherical views using fisheye photographs, 3D urban models, or satellite data [12]. This approach automates the production of fisheye-like images at arbitrary spatial resolutions (e.g., 5-m grids) and computes SVF through equiangular projection, which corrects distortions inherent in traditional fisheye photography. A key advantage is its reliance on widely available geospatial datasets, eliminating the need for fieldwork. Validation studies comparing synthetic results to ground-truth street view imagery show strong correlations (e.g., $R2 > 0.85$) [13], though vegetation occlusion remains a persistent limitation.

1.2 Study Objective: Parallel Computing Using CUDA

This study aims to develop an approach and related workflow for GPU-accelerated continuous SVF estimation. Parallel computing using GPU acceleration has revolutionized data-intensive fields by leveraging massively parallel architectures to process large datasets faster than traditional CPU-based methods. This approach is particularly interesting for raster data processing and geospatial analyses, where operations on grid-based geospatial arrays benefit from the GPU's ability to execute data-parallel tasks through SIMD/SIMT (single instruction, multiple data/thread) architectures.

CUDA and PyCUDA showed great potential for accelerating raster data processing through data parallelism. CUDA (Compute Unified Device Architecture) is NVIDIA's parallel computing platform that enables developers to harness GPU acceleration for general-purpose processing. PyCUDA allows direct GPU programming within Python environments. This combination enables parallel computations that significantly outperform CPU-based approaches.

To minimize the shared memory contention, data partitioning is applied to match the data to the GPU block sizes. As the capability of GPUs to do conditional logic is limited due to thread divergence, a hybrid CPU-GPU workflow is needed, where logic-heavy tasks should be performed by the CPU, whereas the matrix operations should be offloaded to the GPU for faster processing.

2 Methodology

2.1 Workflow Development

The workflow is structured into four distinct phases, each designed to streamline the process and enhance scalability across large spatial domains.

The first phase involves reading and initializing input parameters. This method relies on ray tracing to assess urban form and vegetation using a set of 2D raster-based elevation models in GeoTIFF format, which are widely available for most locations globally. By utilizing raster elevation data rather than full 3D models, the approach increases both accessibility and computational efficiency.

The workflow is centered on three key elevation datasets:

- Digital Surface Model (DSM): The topographic surface height, including buildings and other anthropogenic structures.
- Canopy Digital Surface Model (CDSM)**:** The height of the vegetation canopy relative to ground level.
- Trunk Digital Surface Model (TDSM): The base height of the vegetation canopy. A point cloud was used to approximate this data. Cloud points resembling vegetation were isolated, clustered, and the 5th percentile of height values within each cluster was selected to represent the canopy base.

Processing parameters such as trace radius and angular interval are defined to balance precision and performance. The trace radius determines the maximum search distance for obstructions, with higher values improving accuracy at the cost of computational time. Similarly, smaller angular intervals produce finer angular resolution but increase

processing time. For this study, the maximum trace radius is set to 400 pixels, with an angular interval of 5 degrees.

In the second phase, the CUDA kernel configuration is established. The block size is set to match the GPU thread block structure, and the grid size is calculated accordingly to enable efficient data partitioning. The input data is then transferred from the CPU to the GPU, and GPU memory is allocated for the output arrays.

In the third phase, the core ray tracing routine is performed within the GPU kernel. For this, two nested loops were used: the outer loop iterates over altitude angles, and the inner loop over azimuth angles. These angles are used as directions for the rays to be traced. For each altitude angle, a corresponding weight is computed to represent its contribution to the hemispherical SVF.

Altitude weights are computed using the following expression:

$$A_{weighted} = \cos(\alpha) \cdot \Delta\alpha \cdot \left[\cos\left(\alpha - \frac{2\Delta}{\alpha}\right) - \cos\left(\alpha + \frac{\Delta\alpha}{2}\right)\right] \tag{1}$$

Here, $\cos(\alpha)$ serves as the altitude weighting, reflecting the influence of radiative flux from different sky regions. The remainder of the expression calculates the surface area on the hemisphere associated with a given altitude band and angular interval.

During each iteration of the azimuth loop, rays are traced from each pixel on the DSM, advancing stepwise in the direction defined by the current azimuth and altitude. At each step, the ray's x, y, and z coordinates are updated based on the azimuth, altitude, and current ray length. The ray height is compared against the elevation values from the DSM, CDSM, and TDSM at the corresponding x,y position to determine if any ray obstruction is encountered. If the ray's z-value at that point does not exceed the height of the elevation model, the ray is considered obstructed. If the ray's z-value is higher, it contributes to the SVF based on its computed weight.

In the fourth phase, the SVF at each pixel is calculated as:

$$SVF = \frac{\sum(unobstructed\ ray\ weights)}{\sum(all\ ray\ weights)} \tag{2}$$

When calculated across all pixels, these values yield a continuous SVF map.

This method improves upon traditional SVF estimation techniques by integrating multiple elevation layers and incorporating vertical vegetation structure. The inclusion of the TDSM enables the calculation of SVF beneath tree canopies, thereby capturing visibility at street level—an aspect often overlooked in standard methods that either assess from treetop level or disregard vegetation entirely.

2.2 Assessment Methods

To prove the use of the novel workflow and assess its usability, three additional assessment methods were employed in this study.

Comparison to Theoretical SVF Values

To evaluate the accuracy of the proposed sky view factor (SVF) estimation method, results were systematically compared to theoretical benchmarks derived from idealized

geometries. Infinitely long street canyons (height-to-width ratios: 1:2, 1:1, 2:1) and circular basins (height-to-radius ratios: 1:2, 1:1, 2:1) were analyzed, with theoretical SVF values calculated using geometric principles. The circular basin represents a circular urban canyon, where obstructions of uniform height are positioned equidistantly at all angles around the midpoint.

Computed SVF values from the proposed method were compared to theoretical benchmarks, with absolute errors quantified as their numerical differences. This analysis provides a controlled framework to verify accuracy, establishing baseline performance metrics for reliable application in real-world environments. The formulae below are used to calculate the SVF value for the mid-points of infinitely long canyons (Eq. 3) and circular basins (Eq. 4), based on their H: W-ratio.

Infinitely long canyons:

$$SVF_{mid-canyon} = \cos\left(\tan^{-1}(2 * \mathrm{H/W})\right) \tag{3}$$

Circular basin:

$$SVF_{mid-\mathrm{basin}} = \cos^2(\beta) \tag{4}$$

Comparison Against UMEP in a Real-world Test Case

The second assessment evaluates the proposed GPU-accelerated workflow against the established Urban Multi-scale Environmental Predictor (UMEP) Processing plugin in QGIS [9], using a 1,650 × 1,400-pixel study area (total of 2,310,000 pixels) in Rotterdam's Blijdorp neighborhood. Two SVF calculations were performed with the two methods using the same input data sources, such as elevation data from the AHN[1] and BGT[2]. The absolute error between the two results was calculated.

Computational Time

To evaluate the computational performance of the method across varying input sizes, nine synthetic test environments were created, ranging from 400 × 400 to 2000 × 2000 pixels. Each environment was composed of repeated 200 × 200-pixel tiles featuring a standard geometric pattern: urban canyons with 20 m-high buildings spaced 100 m apart, and 10 m-high trees placed every 10 m along the streets, with a canopy base height of 2.5 m. For each extent, corresponding DSM, CDSM, and TDSM layers were generated to ensure full algorithmic load.

A continuous SVF was estimated using both the proposed method and the UMEP Processing Plugin for QGIS, a widely used CPU-based tool for SVF calculation. For each raster extent, processing times were recorded, and computational performance was evaluated by calculating the average number of pixels processed per second for each extent. This metric provides a comparative measure of efficiency across different spatial resolutions.

[1] Actueel Hoogtebestand Nederland.

[2] Basisregistratie Grootschalige Topografie.

3 Results

3.1 Comparison to Theoretical SVF Values

The comparison between the SVF generated for generic canyons with the developed and theoretical methods shows absolute errors ranging from 0.000 to 0.003. The script is run with an angular interval of 5.0 degrees and a maximum trace radius of 400 pixels. These minor deviations are likely due to the angular interval settings. Reducing the interval size would improve the accuracy of the obstruction height determination, leading to more precise SVF values (Table 1).

Table 1. Validation of SVF-calculation based on theoretical SVF-values for different test cases.

	Theoretical SVF	Calculated SVF	Absolute Error
Canyon H:W 1:2	0.8944	0.896	0.003
Canyon H:W 1:1	0.4472	0.766	0.001
Canyon H:W 2:1	0.2425	0.244	0.001
Basin H:R 1:2	0.8000	0.800	0.000
Basin H:R 1:1	0.5000	0.500	0.000
Basin H:R 2:1	0.2000	0.201	0.001

3.2 Test Case – Rotterdam Blijdorp

The SVF values calculated for a sample urban area using the developed workflow show strong agreement with those from UMEP, with a mean difference of − 0.0012 and a mean absolute error below 1%. The spatial discrepancy pattern shows a slight SVF underestimation at building roofs and overestimation in densely obstructed areas. This corresponds to anisotropic errors observed in shadow-casting algorithms when resolving complex vertical structures. The minor deviations fall within empirical tolerances established for urban radiative exchange modelshttps://centaur.reading.ac.uk/85498/1/1-s2.0-S2212095519300604-main.pdf [14], suggesting the method's reliability for urban climate studies (Fig. 1).

3.3 Processing Time

When comparing computing time across the geometrically generated test setups with varying spatial extents, the developed workflow significantly outperforms the CPU-based alternative. While the UMEP Processing Plugin's shadow-casting algorithm exhibited exponential time complexity (68.2 s at 400 × 400 vs. 6,000 s at 2000 × 2000), the GPU implementation maintained near-linear scaling (4.6 to 44.3 s), a 135 times speedup at the largest tested extent. At 2000 × 2000 resolution (4 million pixels), the GPU's 44.3-s runtime translates to an effective throughput of 90,293 pixels/second compared

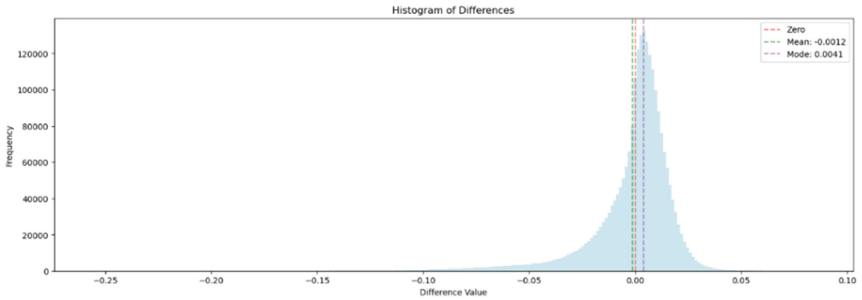

Fig. 1. Differences in SVF values between the CPU-based and proposed SVF estimation method.

to UMEP's 666 pixels/second. For urban climate modeling workflows requiring continuous SVF mapping across city-scale domains (>10 km2), the method eliminates computational bottlenecks inherent to CPU-bound radiative transfer models, enabling simulations previously constrained by hardware limitations (Fig. 2).

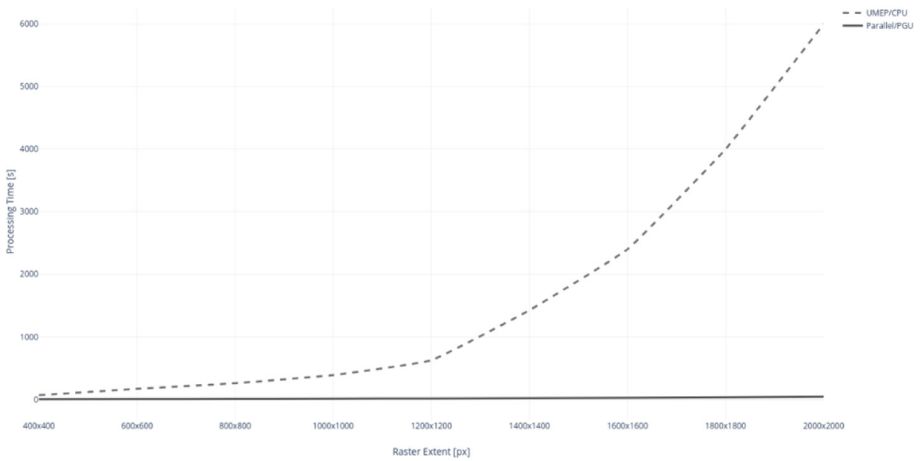

Fig. 2. Processing times of the UMEP Processing plugin for QGIS versus the proposed GPU-accelerated method.

3.4 Workflow Development and Discussions

The accelerated computing capacity enabled by GPU implementation allows for more sophisticated calculations that differentiate between vegetation and building obstructions. This distinction permits more accurate modeling of street-level thermal conditions, particularly in areas with significant tree canopy coverage. The ability to calculate SVF values at the street level underneath tree canopies better represents the actual thermal experience of street users.

The computational efficiency gained through GPU acceleration permits the processing of high-resolution input data (up to resolutions as small as 0.25 m), which was

previously impractical. This advancement is significant because the accuracy of SVF calculations is heavily dependent on the resolution of the underlying elevation data. By processing higher resolution data, GPU-accelerated methods produce more accurate representations of the urban environment, capturing nuanced variations in its surroundings that significantly influence radiation exchanges.

Furthermore, GPU acceleration enables the implementation of ray tracing-based algorithms, which have shown greater efficiency improvements compared to shadow casting-based algorithms. The ability to process larger geographic areas with high-resolution data addresses a fundamental limitation in previous approaches, where SVF applications were often restricted to small urban areas due to computational constraints, facilitating comprehensive urban-scale analyses that were previously unfeasible.

The enhanced computational efficiency of GPU-accelerated SVF calculations fundamentally alters how these climate parameters can be incorporated into urban design and planning workflows. Traditionally, SVF calculations are primarily utilized as post-design assessment tools rather than active design instruments. The extended calculation times meant that SVF analysis was typically performed after design decisions had been finalized, serving as a validation mechanism rather than informing the design process itself. With this workflow, thermal comfort assessments can now be completed within minutes. Further acceleration and parallelization of related processes could enable near-instant thermal comfort simulations, offering the ability to evaluate design options in real time and integrate microclimate performance directly into iterative design decision-making.

Despite the significant advancements in SVF calculation methodologies, several methodological considerations warrant attention. The accuracy of SVF calculations remains dependent on the quality and resolution of input data and processing settings. The resolution and accuracy of the SVFs are dependent on the angular resolution and search radius of the rays. While GPU acceleration enables the processing of higher resolution data, the acquisition of detailed urban geometry information, particularly for vegetation structures, presents an ongoing challenge. Additionally, the GPU memory has practical limitations in handling extremely large raster datasets. Most consumer-grade GPUs are equipped with 8 to 24 GB of VRAM, placing an upper bound on the maximum raster extent that can be processed in a single pass. For high-resolution inputs (e.g., 0.25 m), this generally allows for computation of areas up to km^2. Beyond that, memory overflow or allocation failures may occur, requiring raster tiling.

4 Conclusion

This study demonstrates the reliability and advantages of GPU-accelerated SVF estimation through validation against theoretical models and established computational methods. The proposed workflow achieves high computational efficiency while maintaining the accuracy required for urban climate modeling applications.

The GPU-based method exhibited strong agreement with theoretical SVF values across fictional urban geometries, showing absolute errors ≤ 0.003 when compared to Oke's radiation-based models for infinite canyons and circular basins [5]. The validation against UMEP's shadow-casting algorithm in a real-world urban context (Blijdorp,

Rotterdam) further demonstrated a mean absolute error $< 1\%$ across 2.31 million spatial units. Spatial error patterns aligned with known anisotropic limitations of raster-based SVF estimation, particularly in complex geometries where roof edges and dense vegetation introduce micro-scale uncertainties [9].

The computational performance analysis revealed substantial efficiency improvements with the GPU-accelerated implementation. The GPU processed 4 million pixels in just 44.3 s, while the UMEP required 6,764 s for the same task at 2000×2000 resolution, representing a 135-fold increase in processing speed. The parallelized workflow demonstrates near-linear time complexity, which provides a fundamental advantage over the exponential scaling observed in traditional CPU-based methods. This computational efficiency enables city-scale SVF mapping on standard consumer hardware. The proposed method achieves processing rates of 90,293 pixels per second compared to merely 666 pixels per second with serial CPU processing.

These findings suggest that GPU acceleration effectively removes computational bottlenecks in continuous SVF estimation. This is particularly the case for high-resolution urban climate models requiring frequent radiative exchange updates. The method's validation across theoretical and empirical test cases supports its integration into operational urban planning workflows, where rapid SVF mapping could enhance heat mitigation strategies and microclimate simulations. Future research directions should investigate optimal angular sampling intervals for specific application contexts and extend the parallelization framework to multi-GPU architectures for metropolitan-scale deployments.

Code availability. The source code developed and used in this study is openly available at https://github.com/Maxvdwaal/svf_GPU. The repository includes all scripts necessary to reproduce the main analyses and figures presented in the paper, along with documentation and usage instructions. The code was released under the MIT License, encouraging reuse and adaptation for related research purposes.

Acknowledgements. This work was carried out as part of the "Improving public health through urban greening: A Health Impact Assessment of greening strategies on urban heat stress and heat-related mortality" project, funded by the Resilient Delta Initiative. A portion of the spatial datasets used in this study was kindly provided by the Municipality of Rotterdam. Their contributions are gratefully acknowledged.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Stewart, I.D., Oke, T.R.: Local climate zones for urban temperature studies. Bull. Am. Meteor. Soc. **93**(12), 1879–1900 (2012). https://doi.org/10.1175/bams-d-11-00019.1
2. Middel, A., Häb, K., Brazel, A.J., Martin, C.A., Guhathakurta, S.: Impact of urban form and design on mid-afternoon microclimate in Phoenix Local Climate Zones. Landscape And Urban Planning **122**, 16–28 (2013). https://doi.org/10.1016/j.landurbplan.2013.11.004

3. Middel, A., Lukasczyk, J., Maciejewski, R., Demuzere, M., Roth, M.: Sky View Factor footprints for urban climate modeling. Urban Climate **25**, 120–134 (2018). https://doi.org/10.1016/j.uclim.2018.05.004
4. Maiullari, D.: Urban Form Influence on Microclimate and Building Cooling Demand: An Analytical Framework and Its Application on the Rotterdam Case [Dissertation (TU Delft), Delft University of Technology]. A+BE | Architecture and the Built Environment (2023)
5. Oke, T.R.: Canyon geometry and the nocturnal urban heat island: Comparison of scale model and field observations. J. Climatol. **1**(3), 237–254 (1981). https://doi.org/10.1002/joc.3370010304
6. Li, X., Wang, G.: GPU Accelerated Parallel Computing for Estimating Continuous Sky View Factor Map. Research Square (Research Square) (2021). https://doi.org/10.21203/rs.3.rs-279602/v1
7. Muñoz, D., Beckers, B., Besuievsky, G., Patow, G.: A technique for massive sky view factor calculations in large cities. Int. J. Remote Sens. **39**(12), 4040–4058 (2018). https://doi.org/10.1080/01431161.2018.1452071
8. Ratti, C., Richens, P.: Urban Texture Analysis with Image Processing Techniques. In: Springer eBooks, pp. 49–64 (1999). https://doi.org/10.1007/978-1-4615-5047-1_4
9. Lindberg, F., Grimmond, C.S.B.: The influence of vegetation and building morphology on shadow patterns and mean radiant temperatures in urban areas: model development and evaluation. Theoretical And Applied Climatology **105**(3–4), 311–323 (2011). https://doi.org/10.1007/s00704-010-0382-8
10. Steger, C.R., Steger, B., Schär, C.: HORAYZON v1.2: an efficient and flexible ray-tracing algorithm to compute horizon and sky view factor. Geoscientific Model Development **15**(17), 6817–6840 (2022). https://doi.org/10.5194/gmd-15-6817-2022
11. Bernard, J., Bocher, E., Petit, G., Palominos, S.: Sky view factor calculation in urban context: computational performance and accuracy analysis of two open and free GIS tools. Climate **6**(3), 60 (2018). https://doi.org/10.3390/cli6030060
12. Middel, A., Lukasczyk, J., Maciejewski, R.: Sky view factors from synthetic fisheye photos for thermal comfort routing—a case study in phoenix, Arizona. Urban Planning ***2***(1), 19–30 (2017). https://doi.org/10.17645/up.v2i1.855
13. Gong, F., Zeng, Z., Zhang, F., Li, X., Ng, E., Norford, L.K.: Mapping sky, tree, and building view factors of street canyons in a high-density urban environment. Building And Environment **134**, 155–167 (2018). https://doi.org/10.1016/j.buildenv.2018.02.042
14. Dirksen, M., Ronda, R., Theeuwes, N., Pagani, G.: Sky view factor calculations and their application in urban heat island studies. Urban Climate **30**, 100498 (2019). https://doi.org/10.1016/j.uclim.2019.100498