GPU Accelerated Parallel Computing for Estimating Continuous Sky view Factor Map

xiaojiang li (lixiaojiang.gis@gmail.com)
Temple University https://orcid.org/0000-0002-4208-1641
Guoqing Wang

Research Article

Keywords: Sky view factor (SVF), urban form, GPU parallel computing, PyCUDA

Posted Date: March 11th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-279602/v1

Abstract

The sky view factor (SVF) that represents the fraction of visible sky on a hemisphere or the percentage of radiation reaching the planar ground in the entire hemisphere's input radiation is an important parameter for urban climate studies. However, the estimation of a continuous SVF map is very time-consuming, which limits the applications of SVF to small geographical areas. This study proposed to use *g*raphics processing unit (GPU) parallel computing to accelerate the computing of SVF in the city of Philadelphia, Pennsylvania, USA. This study implemented and compared both the GPU-accelerated version and regular CPU version of two major methods for estimating continuous SVF maps, ray tracing-based algorithm and shadow casting-based algorithm based on the high-resolution building height model. Results show that the GPU-accelerated algorithms can reduce the time consumption dramatically and estimate the SVF map for the city of Philadelphia in less than 20 minutes on a personal computer with one NVIDIA GPU. The ray tracing-based algorithm has a much more efficiency increase than the shadow casting-based algorithm on GPU. The proposed method makes it possible to generate large-scale continuous SVF maps using regular personal computers with GPU. The proposed GPU-accelerated SVF estimation methods would benefit urban climate studies.

1. Introduction

As an important parameter of the urban geometry, the sky view factor (SVF) is a dimensionless parameter that describes the fraction of visible sky on a hemisphere (Hämmerle et al., 2011; Oke, 1981). The SVF also represents the proportion of the incoming solar radiation without being obstructed from the entire upper hemisphere (Li et al., 2018; Oke, 1981; Watson and Johnson, 1987). The value of SVF ranges from 0 to 1, which indicates the sky is totally obstructed and unobstructed, respectively. The SVF has been widely applied in urban microclimate modeling, urban forestry, urban morphology, and urban air pollution studies (Dirksen et al., 2019; Eeftens et al., 2013; Li et al., 2018a; Li et al., 2017; Li and Ratti, 2019; Gong et al., 2018).

The hemispherical image based photographic method is one of the standard methods for SVF estimation (Anderson, 1964; Steyn, 1980). In the photographic method, the hemispherical image is used to represent the projected hemisphere on a two-dimensional flat plane, and the SVF is estimated by dividing the hemispherical image in to annuli and calculating the sum of all sky fractions in all annuli (Steyn, 1980; Li et al., 2018). Since the hemispherical images cover all kinds of obstructions in the street canyons, such as buildings, tree canopies, and other streetscape features, an accurate SVF estimation can be achieved (Li et al., 2018). What is more, the hemispherical images usually need to be taken in time-consuming and labor-intensive field surveys, which limits the application of photographic method to small geographical areas.

Recently, the massively collected geo-tagged street-level images have been introduced in mapping the SVF based on image transformation and segmentation (Carrasco-Hernandez et al., 2015; Gong et al., 2018; Li et al., 2017; Li et al., 2018; Liang et al., 2016; Middel et al., 2017; Middel et al., 2018). In the

proposed methods, the street-level images were geometrically transformed to hemispherical images, which were then used to calculate the SVF. In addition, the different types of streetscape features covered in street-level images were able to be more objectively identified in street canyons. The recent applications of deep learning algorithms to street-level images made it possible to precisely estimate SVF by accurately recognizing different streetscape features (Gong et al., 2018; Li et al., 2018b; Middel et al., 2019). This street-level image-based method is generalizable and scalable for generating the street-level SVF map because of the globally available and publicly accessible street-level images, such as those provided by Google, Baidu, Mapillary, etc. Although the street-level image-based method makes the automatic calculation of the SVF possible, the method is only suitable for mapping the SVF for street centerlines, where the street-level images were collected. However, the street-level image-based method is not suitable to generate the SVF map for those areas with no street-level images available. In addition, the image segmentation in the street-level image-based method is time-consuming for large scale SVF mapping.

The simulation method based on the building height model provides a general and automatic tool to estimate continuous SVF maps (Ratti and Richens, 1999; Ratti and Richens, 2004; Lindberg and Grimmond, 2010). The ray tracing-based algorithm is the most straightforward and accurate method to compute the SVF based on the building height model. By radiating light from each pixel to the surrounding environment and examining the obstructions in searching range, the ray tracing-based algorithm can accurately simulate all obstructions around each pixel. However, the ray tracing-based algorithm is time-consuming to calculate SVF for a large number of pixels. Practically the ray tracing-based algorithm is impossible to generate the continuous SVF map for a relatively large area (Ratti and Richens, 1999). Ratti and Richens (1999) proposed a shadow casting-based method, which is much more efficient than the ray tracing-based method. Different from the pixel-wise ray tracing-based algorithm, the shadow casting-based method is based on the simulation of sun positions in the upper hemisphere, and the SVF is estimated by accumulating the shadow casting maps for different sun positions. Lindberg and Grimmond (2010) further modified the shadow casting-based algorithm by reducing the number of simulated sun positions to increase the efficiency of SVF mapping. The modified shadow casting-based algorithm increases the efficiency and makes city-scale analyses possible (Lindberg et al., 2018). Although the shadow casting-based method increases the efficiency significantly, it stills takes days or even longer for high performance computers to generate the SVF map for a relatively large geographic area. To solve this problem, cloud computing has been applied for SVF mapping for large geographic areas (Dirksen et al., 2019), but it is still time-consuming and expensive for many users.

In this study, we propose to use GPU parallel computing to generate the SVF map based on the high-resolution digital surface model. Since most operations in computing SVF are parallelable, it is possible to use the graphics processing unit (GPU) that has a parallel structure and a large number of cores to compute the SVF parallelly and efficiently. NVIDIA's PyCUDA computing framework was used to access the GPU CUDA (Compute Unified Device Architecture) by combining Python and C programming to accelerate the computation of SVF. As GPUs become available for more and more personal computers,

this proposed method will greatly benefit users in generating SVF maps efficiently on personal computers instead of relying on high performance computing.

2. Methodology

2.1. Study area and data preparation

The study area is the City of Philadelphia (Pennsylvania, USA) with a total land area of 369.59 km^2 (**Fig. 1**). To simulate the obstructions of building blocks in computing the SVF, a building height model was created by overlaying the building footprint map and the normalized digital surface model (DSM). The building footprint map was extracted from the most recent land cover map (**Fig. 1**(a, c)), which is derived from multispectral aerial images and LiDAR data with the spatial resolution of 1 meter and provided on Pennsylvania Spatial Data Access Portal (https://www.pasda.psu.edu/). The normalized DSM was generated from the LiDAR data cloud point that is in form of pre-processed x, y, z points cloud files in the study area (**Fig. 1**(b)).

2.2. Algorithms for computing the sky view factor

The sky view factor (SVF) was calculated in two different ways: the ray tracing and the shadow casting-based algorithms (**Fig. 2**).

In the ray tracing-based algorithm (**Fig. 2b**), for each pixel, the largest obstruction angles at different azimuth directions (β_α) from 0 to 359° in the surrounding pixels was calculated as,

$$\beta_\alpha = max_i \left\{ \text{atan}\left(\frac{H_{\alpha,i}}{D_{\alpha,i}}\right) \right\}, for\ i = 1, \ldots, n \qquad (1)$$

where $H_{\alpha,i}$ is the height of ith pixel from the origin pixel in the building height model at the azimuth direction of α, $D_{\alpha,i}$ is the distance to the origin pixel, and n is the searching radius. Then the SVF value for the pixel can be calculated as (Gal et al., 2009):

$$SVF = 1 - \frac{1}{360}\sum_{\alpha=0}^{359} sin^2\beta_\alpha \qquad (2)$$

Different from the pixel-wise ray tracing-based algorithm, the shadow casting-based algorithm calculates the shadow distributions cast by building blocks at a number of random sun positions (sun elevation and sun azimuth angles) that simulate the solar radiation from the upper hemisphere (Ratti and Richens, 1999, 2004). A continuous SVF map was then generated by averaging all those shadow maps in a cosine-weighted manner. **Fig. 2** (c) shows the process of shadow casting-based algorithm method for computing the SVF. Following Lindberg and Grimmond (2010), in this study, the sun positions were

uniformly spread to reduce the number of sun positions in the shadow casting method, which were proved to generate the same level of accuracy and increase the efficiency of the shadow casting-based SVF estimation algorithm.

2.3. GPU-accelerated algorithms for sky view factor mapping

Taking advantage of PyCUDA's high efficiency in GPU-accelerated parallel algorithm development, the GPU-accelerated ray tracing-based algorithm and the shadow casting-based algorithm were implemented in the PyCUDA framework using Python and C programming languages with Python script used to manipulate the geospatial information and the C code used to implement the pixel level operations. As a comparison, the two SVF algorithms were also implemented and run in Python script with regular CPU. **Fig. 3** shows GPU-based and CPU-based computational models of the two SVF algorithms.

The building height model for the whole study area is too large to compute SVF map at once. Therefore, the building height model of the study area was split into smaller tiles. The SVF computing algorithms were then run on those smaller tiles. To solve the manual shadow effects on the edges of tiles due to abrupt change on borders caused by splitting, a buffer zone with the buffer distance of 150m was added to the top, left, right and down directions for each tile (**Fig. 4**). Those buffered zones were removed in later mosaicking to generate the SVF map for the whole study area.

3. Experiments And Results

This study verified the efficiencies of different algorithms for computing SVF on an Ubuntu Linux computer with 16G memory, Intel Core (TM) i3-4170 CPU, and an NVIDIA GeForce GTX 1060 6GB GPU. The building height model of the study area was split into tiles of different sizes, 1000×1000, 2000×2000, 4000×4000, and 6000×6000 to examine the performances of different algorithms on different tile sizes. In comparison, the GPU-based shadow casting and ray tracing algorithms need much less time to generate the continuous SVF map than the CPU-based shadow casting algorithm with tiles of 1000×1000 and 2000×2000 (**Fig. 5**). The CPU-based ray tracing algorithm took too much time to be plotted at the same scale with other three algorithms in **Fig. 5**. The CPU-based shadow casting algorithm runs out of memory for tiles of 4000×4000 and 6000×6000 (**Table 1**), and the calculating time is supposed to be much longer than the GPU-based shadow casting and ray tracing algorithms.

Table 1 presents the detailed time consumption for different algorithms on different tile sizes. On average, the CPU-based ray tracing algorithm needs 0.38 seconds for one pixel. Therefore, for a tile of 1000×1000, the total time needed to generate a continuous SVF map of 1000×1000 is estimated to be 105.58 hours, and the time consumption for larger tiles increases. The GPU-based ray tracing algorithm is more than 400,000 times faster than the CPU based ray tracing algorithm, and the GPU-based shadow casting algorithm is more than 100 times faster than the CPU-based shadow casting algorithm. Although the CPU-based shadow casting algorithm is much more efficient than the CPU-based ray tracing algorithm, using GPU parallel computing, the ray tracing-based algorithm becomes 3 times faster than the shadow casting-based algorithm.

This study also compared the total time consumption to generate the SVF map of the study area using different algorithms. To increase the computational efficiency, this study only run algorithms on those tiles that intersect the study area. **Table 2** presents the minimum tile numbers to cover the whole study area at different tile sizes. The total time consumption of different algorithms for generating the SVF map of the whole study area is the multiplication of the tile numbers and the time consumption for each tile. **Table 2** lists the total time consumption of different algorithms for creating the continuous SVF map of Philadelphia. By setting the tile size as 2000×2000, the total time needed to generate the SVF map of the study area is as low as 0.32 hours (19.2 minutes). The total time consumption of SVF mapping in the study area generally decreases as tiles size increases, which means increasing the tile size helps to reduce the time consumption, while larger tiles also need more memory. The time consumption is lowest when the tile size was set as 2000×2000, because of the relatively small number of required tiles to cover the whole study area when the tile size is 2000×2000, while the number of required tiles is impacted by the combination of tiles size and the shape of the study area.

Fig. 6 shows the spatial distributions of the SVF at the pixel level (**Fig. 6**(a)) and the aggregated census tract level by the mean value (**Fig. 6**(b)) in the study area. Since only the building height model was used to generate the SVF map, therefore, the SVF maps indicate the obstruction of building blocks only. It can be seen clearly that the downtown area with most of the high-rise buildings in Philadelphia has the lowest SVF values, while the periphery parts of the study areas have larger SVF values. For those areas with no building blocks, the SVF values are close to 1, which indicates no obstruction by building blocks.

4. Discussion

A fine level sky view factor (SVF) map is important for us to model the solar radiation fluxes, understand the impacts of urban form on the urban microclimate, study the spatial pattern of urban heat islands, and investigate human heat exposure and public health (Gong et al., 2018; Li et al., 2018a; Dirksen et al., 2019; Yuan and Chen, 2011). However, the SVF mapping is usually very time-consuming, which blocks large scale urban environmental analyses. This study proposed to use graphics processing unit (GPU) accelerated parallel computing to generate continuous SVF maps based on a high spatial resolution building height model generated from multispectral remotely sensed imageries and LiDAR data. The results show that the GPU parallel computing increase the computational efficiency for SVF computing dramatically, which makes the continuous SVF mapping possible at city scale on personal computers. Compared with previous algorithms that need days or longer on high performance computers, the GPU-accelerated algorithm only needs less than 20 minutes to estimate the SVF map for Philadelphia on a personal computer with a GPU device. Since the SVF is an important parameter of urban form, the proposed GPU-accelerated algorithms for the SVF estimation would benefit all those studies related to SVF and scale up the case studies to a larger geographic scale. With the high-resolution building height model becomes more accessible, the proposed method is generalizable for other study areas, which would benefit fine-scale urban climate modeling and the urban form quantification.

This study compared the efficiencies of different GPU-accelerated algorithms for SVF mapping. Although the shadow casting-based algorithm has been proved much faster than the ray tracing-based algorithm for SVF mapping using CPU-based computing framework, in GPU computing framework, GPU-accelerated ray tracing algorithm is about 3 times faster than the shadow casting algorithm. This is because the ray-tracing algorithm is more parallelable compared with the shadow casting-based method in terms of pixel-level operations, which makes the computational capacity of GPU been fully utilized in the GPU-based ray-tracing algorithm.

This study also investigated the impacts of splitting tile sizes on the computational efficiencies of different algorithms. Generally, increasing the tile size would help to increase the efficiency, while a larger tile size would also require larger memory. The shape of the study would also impact the efficiency because the shape of the study area would impact the number of minimum tiles to cover the whole study area. In addition, the GPU cores and the number of GPUs would also impact the optimized configuration for computing SVF.

Although this study showed the efficiency improvement of GPU-based parallel computing in estimating SVF. There are still some limitations that need to be resolved in future studies. Firstly, this study used the building height model to examine the computational efficiency increase using the GPU based method, without taking into consideration of the impact of tree canopies. Thus, the SVF map generated in this study only represents the enclosure of urban space caused by building blocks. In cities, trees are also an important obstruction of solar radiation and impact the SVF on the ground. Therefore, future studies should also incorporate the tree canopy height model in estimating the SVF. Since the proposed GPU-accelerated algorithm is scalable and generalizable, new SVF maps with considering the tree canopy map can be generated with the same level of efficiency.

In this study, only one GPU was used on a regular personal computer with a memory of 16G to compute the SVF map. The efficiency increase is also impacted by the total number of tiles and the GPU configuration. This study shows that even a lower-level GPU device would increase the efficiency dramatically, which makes the large scale SVF mapping possible on personal computers. Future studies should also explore to optimize the configuration in order to achieve maximum efficiency with consideration of the number of GPU cores, the memory limit, and tile size and numbers.

In addition, fine level urban microclimate modeling requires a lot of highly parallelable pixel-level operations. Therefore, other than mapping the SVF, future studies should explore how to accelerate other procedures in the urban microclimate modeling in order to better model the dynamic urban thermal comfort.

5. Conclusion

This study proposed to use GPU-based parallel computing for estimating the sky view factor map (SVF) in Philadelphia. Compared with the CPU-based algorithms for SVF mapping that needs days or longer, the proposed method needs less than 20 minutes for mapping the SVF for the city of Philadelphia on a

personal computer with only one GPU. The GPU-accelerated parallel computing increases the computational efficiency dramatically compared with the traditional CPU based algorithms for SVF mapping. In the GPU computational framework, the more straightforward ray tracing-based algorithm is three times faster than the shadow casting-based algorithm for estimating SVF. The proposed method makes it possible to do large scale urban form analysis on a personal computer with GPU devices without relying on high performance computers, which would benefit urban microclimate studies at large scales significantly.

Declarations

Conflict of Interest

No conflict of Interest

Funding Statement

There is no funding support for this research.

Author's Contribution

XL developed the idea, implemented the algorithm, and wrote the paper.

GW revised the paper.

Availability of data and material

The dataset used in this study is publicly accessible from https://www.pasda.psu.edu/

Code availability

The code will be available on GitHub after publication.

Ethics approval

The Author warrants that the Work has not been published before in any form except as a preprint, that the Work is not being concurrently submitted to and is not under consideration by another publisher, that the persons listed above are listed in the proper order and that no author entitled to credit has been omitted, and generally that the Author has the right to make the grants made to the Publisher complete and unencumbered. The Author also warrants that the Work does not libel anyone, infringe anyone's copyright, or otherwise violate anyone's statutory or common law rights.

Consent to participate

There are no human participants in the experiment.

Consent for publication

The Author hereby transfers to the Publisher the copyright of the Work. As a result, the Publisher shall have the exclusive and unlimited right to publish the Work wholly or in part throughout the World in all languages and all media for all applicable terms of copyright.

References

Anderson, M. C. (1964). Studies of the woodland light climate: I. The photographic computation of light conditions. *The Journal of Ecology*, 27-41.

Carrasco-Hernandez, R., Smedley, A. R., & Webb, A. R. (2015). Using urban canyon geometries obtained from Google Street View for atmospheric studies: Potential applications in the calculation of street level total shortwave irradiances. *Energy and Buildings*, 86, 340-348.

Dirksen, M., Ronda, R. J., Theeuwes, N. E., & Pagani, G. A. (2019). Sky view factor calculations and its application in urban heat island studies. *Urban Climate,* 30, 100498.

Eeftens, M., Beekhuizen, J., Beelen, R., Wang, M., Vermeulen, R., Brunekreef, B., ... & Hoek, G. (2013). Quantifying urban street configuration for improvements in air pollution models. *Atmospheric Environment*, 72, 1-9.

Gal, T., Lindberg, F., & Unger, J. (2009). Computing continuous sky view factors using 3D urban raster and vector databases: comparison and application to urban climate. *Theoretical and Applied Climatology*, 95(1), 111-123.

Gong, F. Y., Zeng, Z. C., Zhang, F., Li, X., Ng, E., & Norford, L. K. (2018). Mapping sky, tree, and building view factors of street canyons in a high-density urban environment. *Building and Environment*, 134, 155-167.

Hämmerle, M., Gál, T., Unger, J., & Matzarakis, A. (2011). Comparison of models calculating the sky view factor used for urban climate investigations. *Theoretical and Applied Climatology*, 105(3), 521-527.

Li, X., Ratti, C., & Seiferling, I. (2017, July). Mapping urban landscapes along streets using google street view. In International cartographic conference (pp. 341-356). Springer, Cham.

Li, X., Ratti, C., & Seiferling, I. (2018a). Quantifying the shade provision of street trees in urban landscape: A case study in Boston, USA, using Google Street View. *Landscape and Urban Planning*, 169, 81-91.

Li, X., Cai, B. Y., & Ratti, C. (2018b). Using street-level images and deep learning for urban landscape studies. Landscape Architecture Frontiers, 6(2), 20-30.

Li, X., & Ratti, C. (2019). Mapping the spatio-temporal distribution of solar radiation within street canyons of Boston using Google Street View panoramas and building height model. *Landscape and Urban Planning*, 191, 103387.

Liang, J., Gong, J., Sun, J., Zhou, J., Li, W., Li, Y., ... & Shen, S. (2017). Automatic sky view factor estimation from street view photographs—A big data approach. *Remote Sensing*, 9(5), 411.

Lindberg, F., & Grimmond, C. S. B. (2010). Continuous sky view factor maps from high resolution urban digital elevation models. *Climate Research*, 42(3), 177-183.

Lindberg, F., Grimmond, C. S. B., Gabey, A., Huang, B., Kent, C. W., Sun, T., ... & Zhang, Z. (2018). Urban Multi-scale Environmental Predictor (UMEP): An integrated tool for city-based climate services. Environmental Modelling & Software, 99, 70-87.

Middel, A., Lukasczyk, J., & Maciejewski, R. (2017). Sky view factors from synthetic fisheye photos for thermal comfort routing—a case study in Phoenix, Arizona. *Urban Planning*, 2(1), 19-30.

Middel, A., Lukasczyk, J., Maciejewski, R., Demuzere, M., & Roth, M. (2018). Sky View Factor footprints for urban climate modeling. *Urban Climate*, 25, 120-134.

Middel, A., Lukasczyk, J., Zakrzewski, S., Arnold, M., & Maciejewski, R. (2019). Urban form and composition of street canyons: A human-centric big data and deep learning approach. *Landscape and Urban Planning*, 183, 122-132.

Oke, T. R. (1981). Canyon geometry and the nocturnal urban heat island: comparison of scale model and field observations. *Journal of Climatology*, 1(3), 237-254.

Ratti CF, Richens P (1999) Urban texture analysis with image processing techniques Proc CAADFutures99, Atlanta, GA.

Ratti, C., & Richens, P. (2004). Raster analysis of urban form. Environment and Planning B: Planning and Design, 31(2), 297-309.

Steyn, D. G. (1980). The calculation of view factors from fisheye-lens photographs, *Atmosphere-Ocean*, 18 (1980), pp. 254-258

Watson, I. D., & Johnson, G. T. (1987). Graphical estimation of sky view-factors in urban

environments. *Journal of Climatology*, 7(2), 193–197.

Yuan, C., & Chen, L. (2011). Mitigating urban heat island effects in high-density cities based on sky view factor and urban morphological understanding: a study of Hong Kong. Architectural Science Review, 54(4), 305-315.

Tables

Table 1. The time consumptions of different algorithms for generating continuous sky view factor maps for different tile sizes.

Algorithms	Time consumption			
Tiles	1000×1000 (0.6×0.6 km^2)	2000×2000 (1.2×1.2 km^2)	4000×4000 (2.4×2.4 km^2)	6000×6000 (3.6×3.6 km^2)
CPU ray tracing	105.6 *h*	422.3 *h*	1689.3 *h*	3801.0 *h*
CPU shadow casting	321.90 *s*	1543.17 *s*	Out of memory	Out of memory
GPU ray tracing	1.12 *s*	3.85 *s*	16.72 s	36.92 *s*
GPU shadow casting	3.03 *s*	13.19 *s*	55.56 s	128.20 *s*

Table 2. Total time consumptions of different algorithms for generating a continuous SVF map of Philadelphia

Algorithms	Time consumption			
Tiles numbers	1076 (1000×100)	297 (2000×2000)	84 (4000×4000)	32 (6000×6000)
CPU ray tracing	4734.4 days	5226.0 days	5912.5 days	5068 days
CPU shadow casting	96 h	127.31 h	Out of memory	Out of memory
GPU ray tracing	0.54 h	0.32 h	0.39 h	0.33 h
GPU shadow casting	1.46 h	1.08 h	1.30 h	1.14 h

Figures

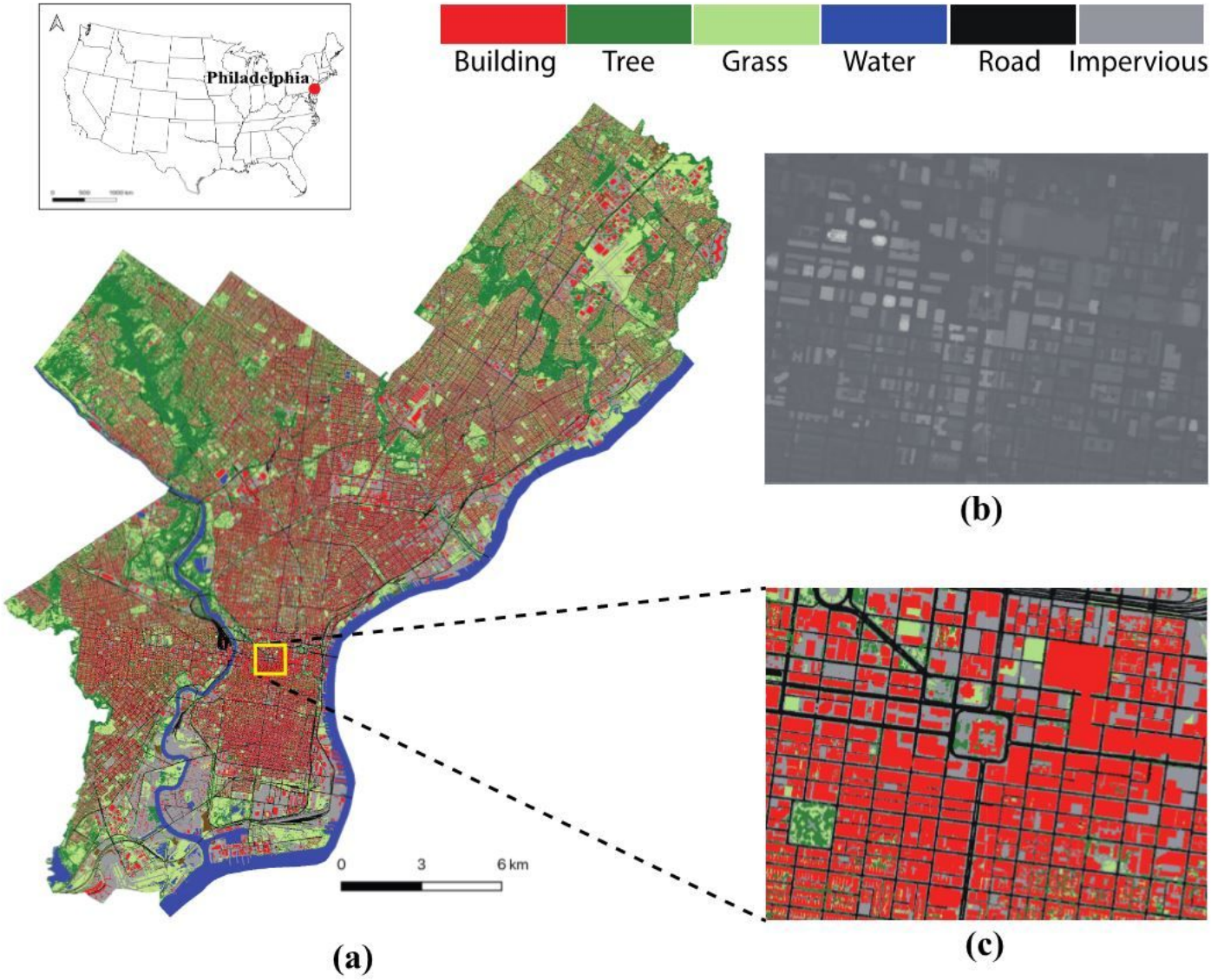

Figure 1

The location and datasets of the study area, (a) the location and the land cover map of Philadelphia, (b) the normalized digital surface model of the downtown Philadelphia, (c) the land cover in downtown Philadelphia. Note: The designations employed and the presentation of the material on this map do not imply the expression of any opinion whatsoever on the part of Research Square concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries. This map has been provided by the authors.

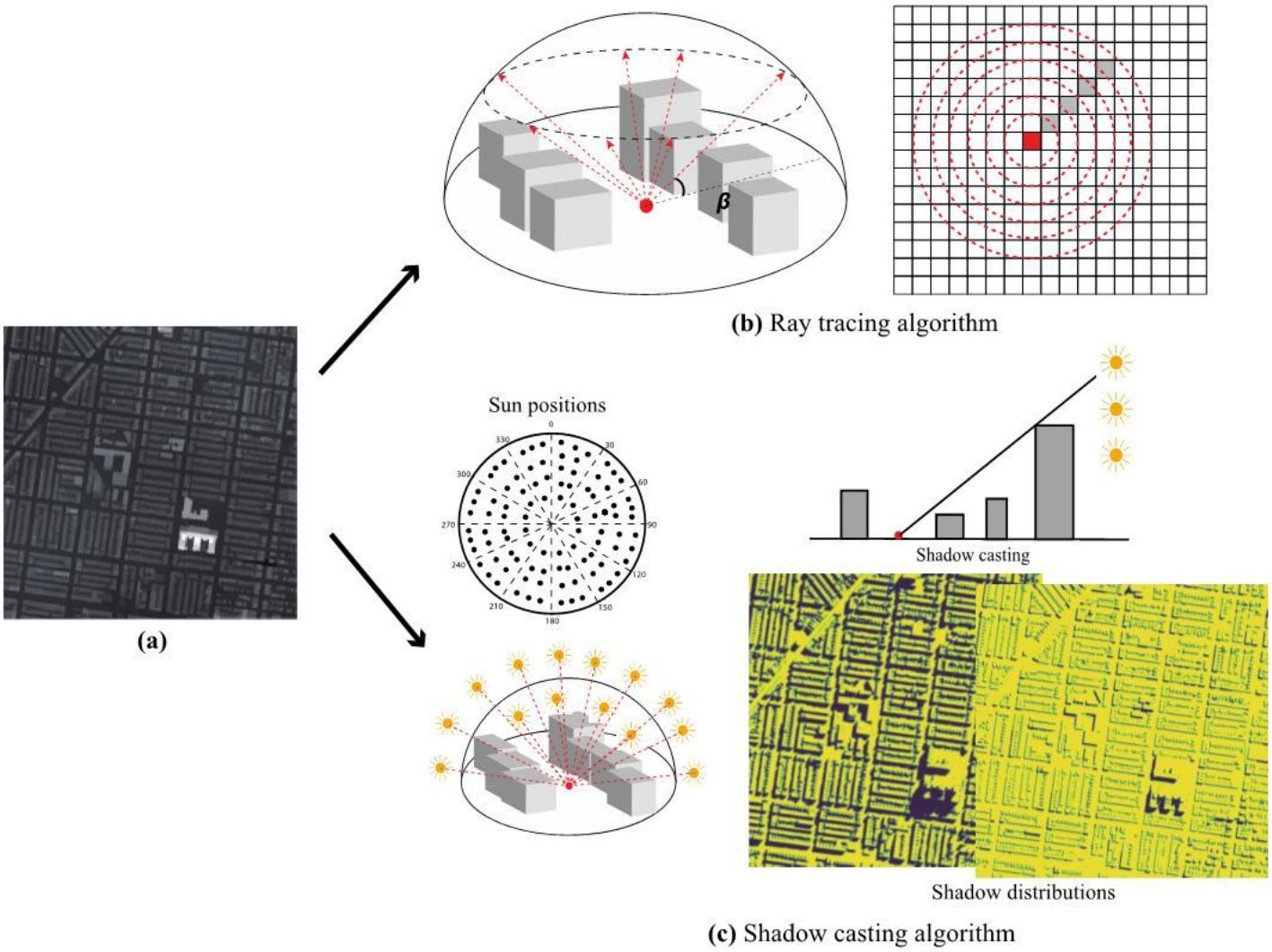

Figure 2

The ray tracing and the shadow casting algorithms for estimation of the sky view factor (SVF), (a) the input building height model, (b) the ray tracing-based algorithm for SVF estimation, (c) the shadow casting-based algorithm for SVF estimation.

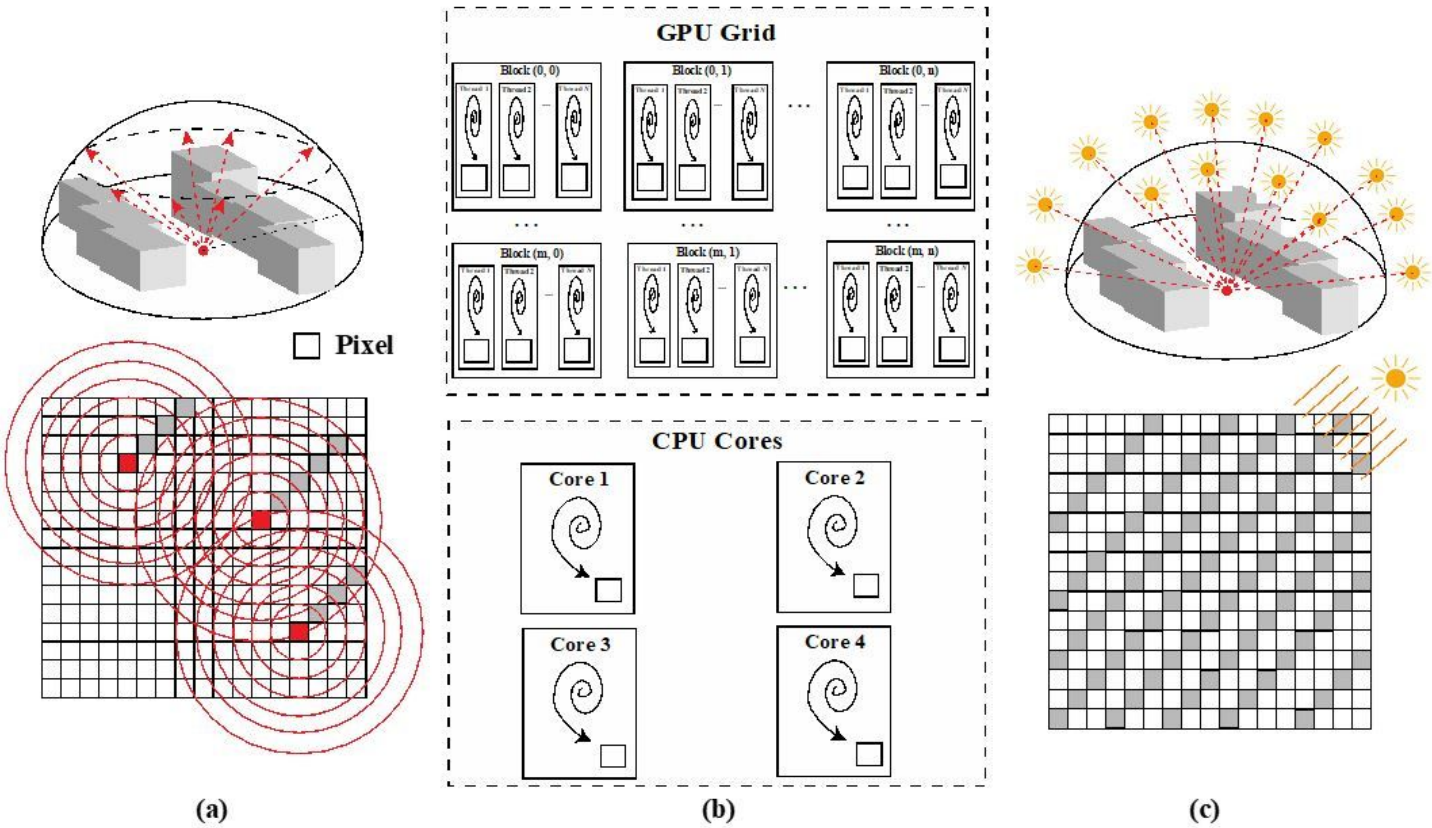

Figure 3

The CPU and GPU based computational models for computing the sky view factor (SVF) maps, (a) the ray tracing-based algorithm for SVF estimation, (b) GPU and CPU models for pixel level operations, (c) the shadow casting-based algorithm for SVF estimation.

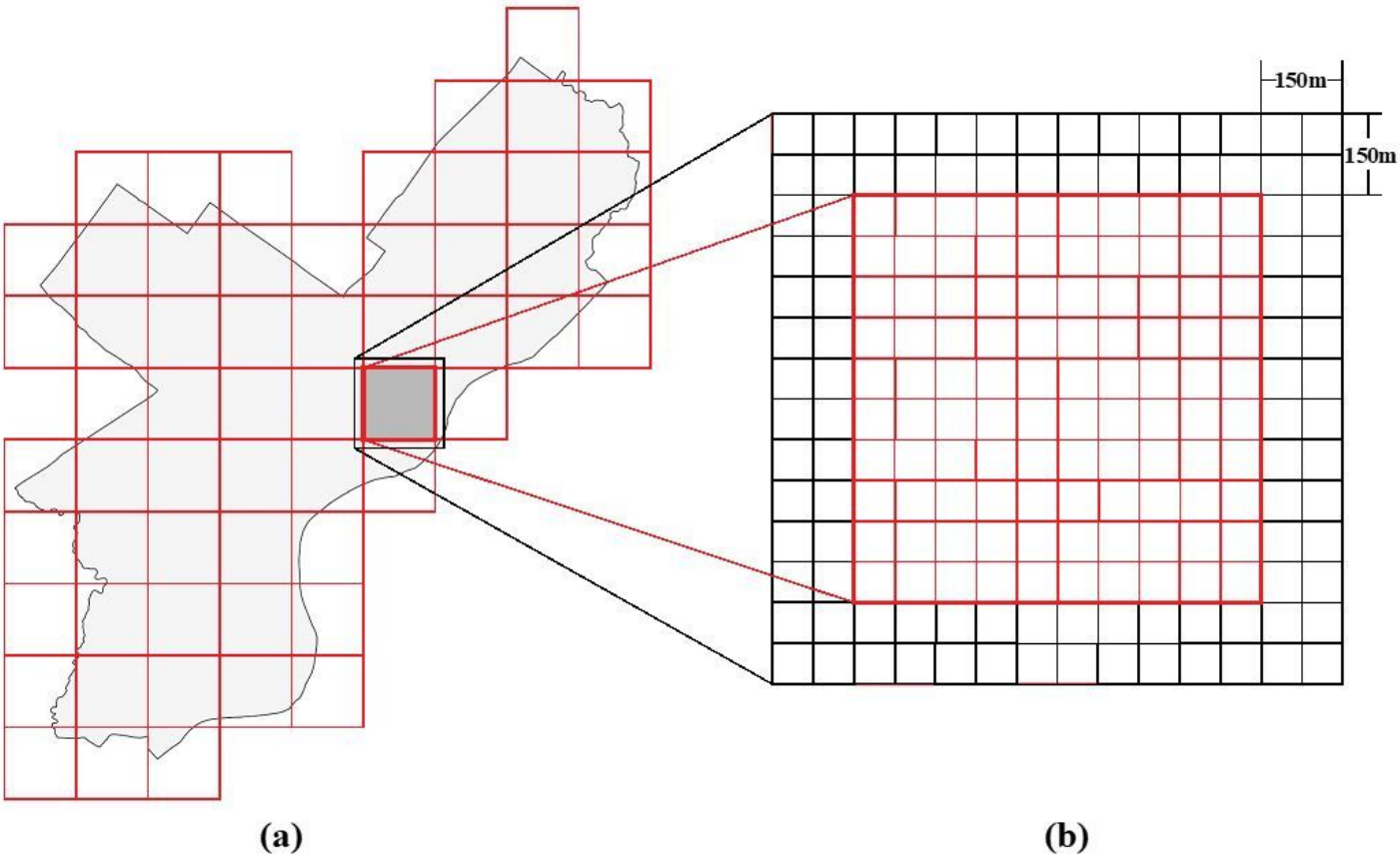

Figure 4

Splitting building height model of Philadelphia in tiles and adding a buffer of 150 m on left, right, top and bottom to each tile avoiding mutual shadowing effects on the edge.

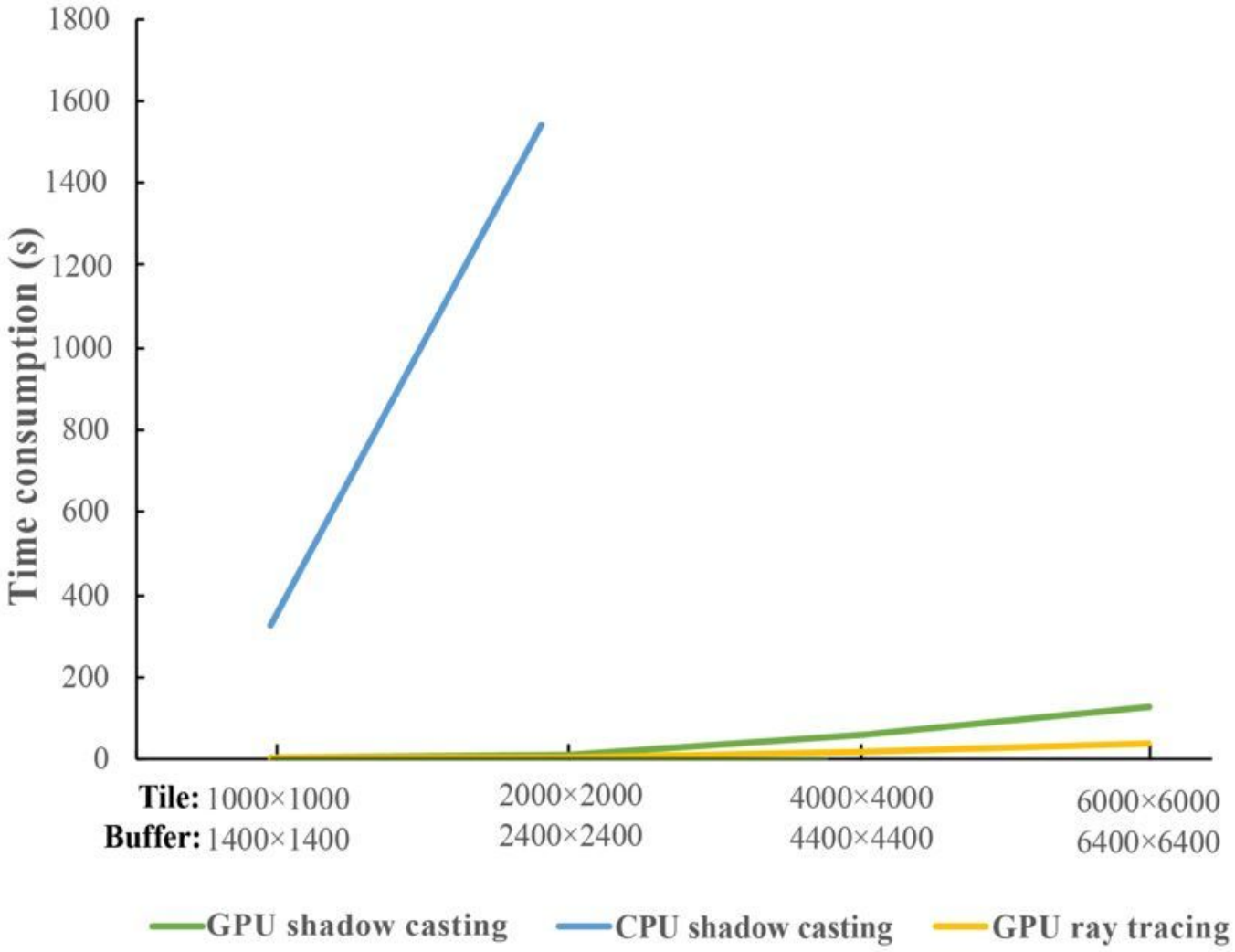

Figure 5

The time consumptions of the SVF estimation using CPU and GPU versions of ray tracing algorithm and shadow casting algorithm for different sizes of building height model tiles.

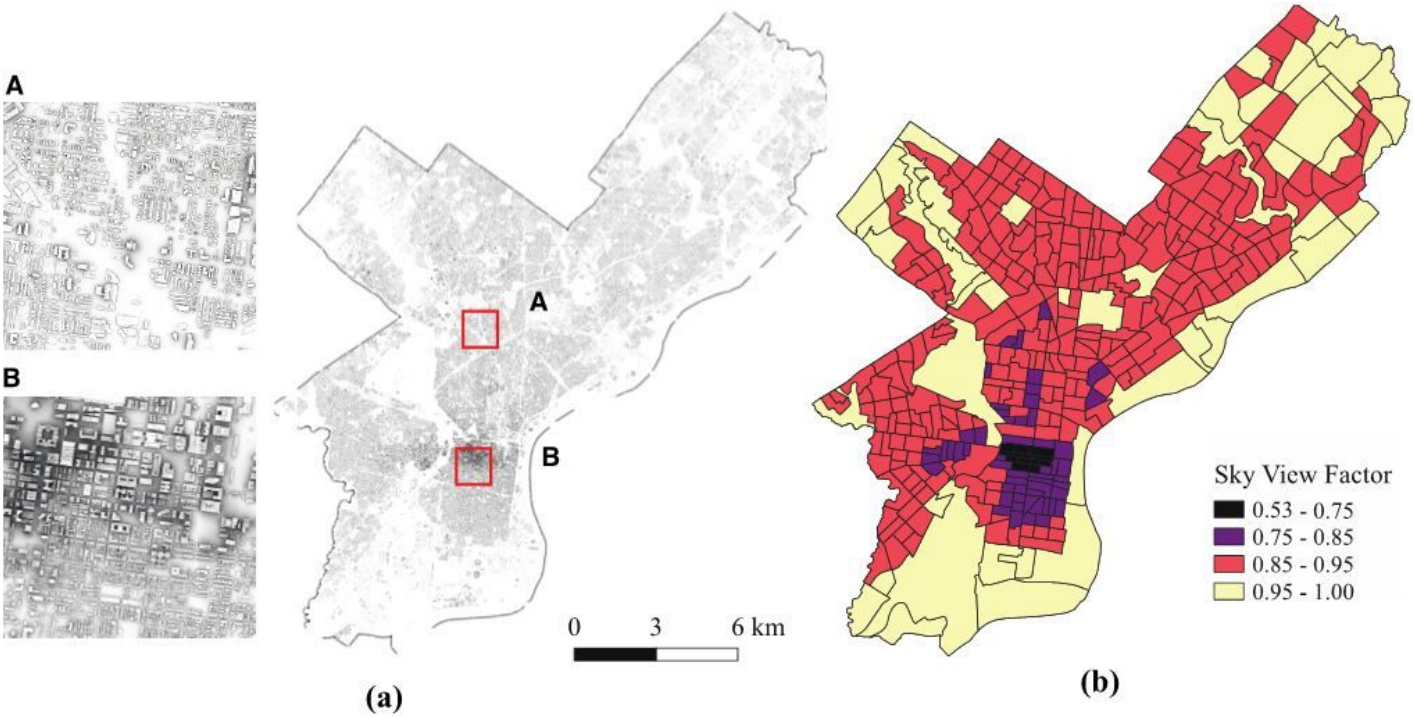

Figure 6

The spatial distributions of the SVF in Philadelphia at the pixel level (a) and the aggregated by mean value at the census tract level (b). Note: The designations employed and the presentation of the material on this map do not imply the expression of any opinion whatsoever on the part of Research Square concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries. This map has been provided by the authors.