Efficient Visualization of Light Pollution for the Night Sky

YOSHINORI DOBASHI, Hokkaido University, Prometech CG Research, Japan
NAOTO ISHIKAWA, Hokkaido University, Japan
KEI IWASAKI, Saitama University, Prometech CG Research, Japan

Artificial light sources make our daily life convenient, but cause a severe problem called *light pollution*. We propose a novel system for efficient visualization of light pollution in the night sky. Numerous methods have been proposed for rendering the sky, but most of these focus on rendering of the daytime or the sunset sky where the sun is the only, or dominant light source. For the visualization of the light pollution, however, we must consider many city light sources on the ground, resulting in excessive computational cost. We address this problem by precomputing a set of intensity distributions for the sky illuminated by city light at various locations and with different atmospheric conditions. We apply a principal component analysis and fast Fourier transform to the precomputed distributions, allowing us to efficiently visualize the extent of the light pollution. Using this method, we can achieve one to two orders of magnitudes faster computation compared to a naive approach that simply accumulates the scattered intensity for each viewing ray. Furthermore, the fast computation allows us to interactively solve the inverse problem that determines the city light intensity needed to reduce light pollution. Our system provides the user with both a forward and inverse investigation tool for the study and minimization of light pollution.

CCS Concepts: • **Computing methodologies → Rendering**.

Additional Key Words and Phrases: light pollution, the night sky, principal component analysis, fast Fourier transform

ACM Reference Format:
Yoshinori Dobashi, Naoto Ishikawa, and Kei Iwasaki. 2023. Efficient Visualization of Light Pollution for the Night Sky. *ACM Trans. Graph.* 42, 6, Article 219 (December 2023), 11 pages. https://doi.org/10.1145/3618337

1 INTRODUCTION

The sky is complex and beautiful. Many computer graphics researchers have tried to reproduce the beauty of the sky under different lighting and atmospheric conditions. For the familiar blue and the sunset sky, the sun is the primary light source that determines the appearance of the sky. After sunset, the sky exhibits a completely different appearance, together with the moon and the stars. However, there is another source of light at night, artificial light sources in the city, that often ruins the beauty of the night sky. Whilst artificial light sources make our daily life convenient and comfortable during the night, they also cause a severe problem called *light pollution*. Inappropriate or excessive artificial light sources can have serious negative impacts on our health, ecosystems, and of course on the aesthetic environment [Gaston et al. 2013]. In an analysis of the influence of the artificial light sources on these factors, the visualization of the light pollution plays a significant role, which we focus on in this paper.

Many methods have been developed for the efficient rendering of the daytime or the sunset sky where the sun is the only light source (e.g., [Preetham et al. 1999]). For the visualization of the light pollution for the night sky, however, we must consider many artificial light sources on the ground, resulting in increased computational cost. Our goal is fast visualization of the light pollution that allows the user to interactively investigate the influence of city lights on the night sky. We want to allow the user to interactively investigate the influence of the city light on the sky with a forward and an inverse approach. With the forward approach, the light pollution should be visualized in real-time for an arbitrary observer location under different atmospheric conditions. The inverse approach should then allow the user to interactively control the city light to investigate how we can reduce light pollution. To achieve these functions, we propose a fast method for computing the intensity of the sky, or *skyglow*, given a density distribution of light sources on the ground.

We employ a precomputation-based approach to achieve this goal. Our method precomputes the intensity of the sky under different lighting and atmospheric conditions. To save on the storage cost for the precomputed data, we compress the data using principal component analysis (PCA) followed by Fast Fourier Transform (FFT). This also allows us to achieve a fast intensity calculation. Note that our primary focus is on the fast computation of skyglow due to the presence of city lights consisting of a large number of artificial light sources. Our method is independent of how the scattered intensity is computed; we can use any computational models for the light scattering. In this paper, we primarily employ a single scattering model for computing the skyglow, while also demonstrating that our method can accommodate multiple scattering.

This paper is organized as follows. Section 2 discusses the previous work on rendering the sky. Section 3 proposes the fast computation method for the skyglow. Next, Section 4 describes our inverse investigation tools. Section 5 shows experimental results to investigate the effectiveness and usefulness of the proposed method. Section 6 discusses the limitations of our method and future research directions. Finally, Section 7 concludes this paper.

2 RELATED WORK

Research on the realistic rendering of the sky has a long history. The first research was done by Klassen [1987] who tried to display the sky color by considering the spectral distribution of the light scattered by particles in the atmosphere. However, this model represented the atmosphere as multiple plane-parallel layers and was not appropriate for rendering realistic images. Kaneda et al. [1991] then proposed an improved and more realistic model in which they

Authors' addresses: Yoshinori Dobashi, Hokkaido University, Prometech CG Research, Kita 14, Nishi 9, Sapporo, 060-0814, Japan, doba@ime.ist.hokudai.ac.jp; Naoto Ishikawa, Hokkaido University, Kita 14, Nishi 9, Sapporo, 060-0814, Japan, ishikawa@ime.ist.hokudai.ac.jp; Kei Iwasaki, Saitama University, Prometech CG Research, –, Saitama, –, Japan, kiwasaki@mail.saitama-u.ac.jp.

employed a spherical-shell representation for the atmosphere and the density distributions of both air molecules and aerosols were assumed to vary exponentially with altitude. Nishita et al. [1993] further extended the model to the rendering of the earth viewed from space. These early researchers provide the fundamentals for a realistic rendering of the sky.

Following on from those fundamental papers, several real-time methods have been investigated for rendering the sky by using advanced graphics hardware or GPUs [Dobashi et al. 2002][O'Neil 2007][Wenzel 2007]. An analytical method for single scattering has also been proposed [Sun et al. 2005]; however, the method's main objective is not related to rendering the sky. Methods considering multiple scattering of light have also been developed to improve the realism of the synthetic sky [Bruneton and Neyret 2008][Hillaire 2020]. Since the computation of the multiple scattering is usually expensive, these methods use lookup tables for acceleration.

Another research direction is a data-driven approach to derive a compact and easy-to-use representation for computing the intensity of the sky under the different sun directions and different atmospheric conditions. Pioneering work in this direction was done by Preetham et al. [1999], in which an analytical model was derived by fitting basis function coefficients to simulated intensity distributions of the sky. Hosek et al. [2012] improved this model to accurately render sunsets and high atmospheric turbidity scenes. Wilkie et al. [2021] made further improvements by using the atmospheric measurement data. A machine-learning-based approach was also proposed to compactly represent the sky illumination from both existing analytic sky models and captured environmental maps [Satilmis et al. 2017].

All the above methods, however, focus on the rendering of the daytime or the sunset sky, so they are not suitable for the visualization of the light pollution for the night sky. Jensen et al. [2001] were the first to render the night sky in a physically-based manner. This method uses a path tracer to synthesize realistic images of the night sky including the moon and the stars, considering the multiple scattering of light. Minor et al. [2016] extended this method to consider artificial light sources. However, this method is computationally expensive since it is also based on path tracing.

Researchers in the field of light pollution developed several analytical models for computing the intensity of the sky illuminated by artificial light sources [Solano Lamphar 2018]. Such models consider the atmospheric scattering of light radiated from the city. Amongst these, the model proposed by Garstang [1986] is widely used by many researchers, and many improved models have been developed based on this work. Kocifaj [2007] extended this model to consider a cloudy sky, which was used in a light pollution visualization system [Kocifaj and Kundracik 2016]. An open-source tool, Skyglow Estimation Toolbox, has been developed by NASA and can also be used to visualize light pollution [Avery et al. 2017]. Further extension has been made to include different physical processes such as multiple scattering, the Earth's curvature and scattering phase functions [Cinzano and Falchi 2012; Kocifaj 2018]. However, these methods are not fast enough for real-time visualization of the skyglow viewed from an arbitrary location. A method of expanding the skyglow distribution into basis functions was proposed for efficient evaluation of the skyglow [Bará et al. 2015]. However, the computation is still too slow for our purpose since we must consider many light sources in the city. For fast computation of the light pollution map, which records the total light flux at every location due to the surrounding city light sources, several researchers have developed point spread functions that enable fast computation of the map using fast Fourier transform methods [Bará et al. 2020; Falchi and Bará 2021; Simoneau et al. 2021]. However, these methods are not applicable to the computation of the skyglow distribution observed at an arbitrary location. Furthermore, no methods have been developed for an efficient and interactive solution of the inverse problems of light pollution.

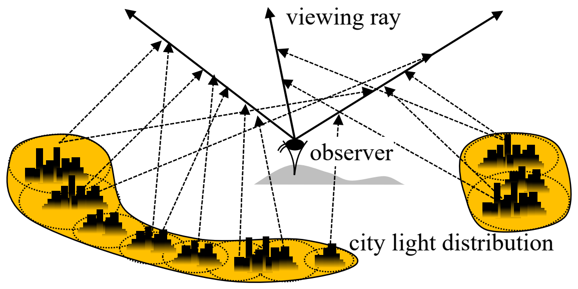


Fig. 1. Overview of the light pollution computation.

3 FAST COMPUTATION OF SKYGLOW

This section describes our method for the fast computation of the skyglow. The input to our system is the terrain geometry and the density distribution of light sources on the ground. The skyglow images are rendered in real-time for arbitrary locations of the observer. Note that our method does not consider the occlusion of light by the terrain. We first explain an overview of the light pollution computation in Section 3.1. Section 3.2 describes the computation of the skyglow due to a single infinitesimal area light source. Next, the formulation for our fast computation method is provided in Section 3.3. The implementation details of the method are described in Section 3.4.

3.1 Light Pollution Computation

When we want to visualize the light pollution, or the skyglow, over the whole sky, we need to calculate the intensity of the sky for every viewing direction considering all the surrounding city light sources as shown in Fig. 1. For each point on each viewing ray, we need to accumulate the contribution from those city light sources. A city light at several tens of kilometers from the observer could have visible contributions to the intensity of the sky. This is particularly true when the observer is located at a higher location, such as the one in a mountain, where the light pollution is often most problematic. Thus, the distribution of the city light needs to be considered and should cover a wide range of regions, resulting in extremely expensive computational cost.

Since modeling individual light sources is impractical, we represent the city lights with a density distribution of light sources

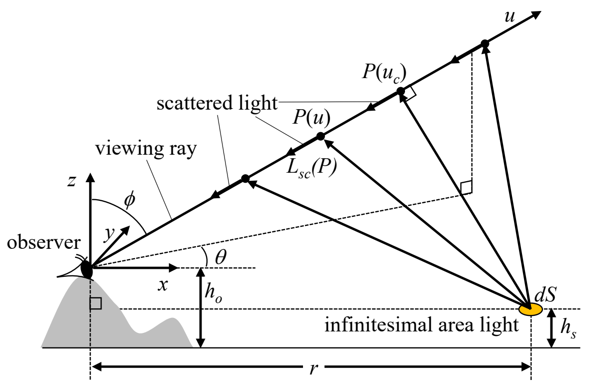


Fig. 2. Computation of skyglow.

on the ground. The density at each point corresponds to the emitted light flux per unit area from that point. We assume that the light is emitted to the zenith direction. The skyglow is rendered by accumulating the contribution over the distribution. In the computation of the skyglow, we do not consider the occlusion of the light by the terrain (we will discuss viable solutions to this limitation in Section 6). We employ a precomputation approach for efficient visualization. In the preprocess, we calculate a set of skyglow distributions due to infinitesimal area light sources located at various distances and altitudes, each with unit light flux. Our method then applies the principal component analysis to them, followed by the Fourier transform, for faster computation of the runtime process and for reducing the storage cost. In the following subsections, we start the explanation of our method with the computation of the skyglow due to an infinitesimal area light source.

3.2 Skyglow by a Single Light Source

The skyglow intensity observed at a certain location is determined by accumulating the scattered intensities on a viewing ray. The intensity depends on the altitude of the light source and the observer as well as the atmospheric condition. For the atmospheric condition, we use a single parameter called turbidity that determines the reduction in the transparency of air [Preetham et al. 1999]. The skyglow due to an infinitesimal area light source is explained in the following.

As shown in Fig. 2, we assume that the observer is at the origin and we represent the locations of the light source with a polar coordinate. That is, the location of the light source is represented with the distance from the observer, the azimuth angle from x axis, and the altitude. Let us consider the differential distribution of the skyglow, db $[W/sr]$, due to the infinitesimal area light source with unit light flux, located at distance r along x axis. Then, db is expressed by the following equation.

$$db = b(\theta, \phi, r, h_c, h_o, \tau, \lambda)dS, \tag{1}$$

where dS is the area of the light source. b is the accumulated intensity of the scattered light reaching from a point on the light source, as expressed by:

$$b(\theta, \phi, r, h_c, h_o, \tau, \lambda) = \int_0^{\infty} L_{sc}(P(u), r, \theta, h_c, h_o, \tau, \lambda)du, \tag{2}$$

where (θ, ϕ) represents the direction of the viewing ray, h_c is the altitude of the light source, h_o the altitude of the observer, u the distance from a point on the viewing ray to the viewpoint, τ the turbidity, and λ the wavelength. L_{sc} is a function that returns the intensity of light scattered at point $P(u)$ on the viewing ray. In our current approach, our method cannot consider occlusion by the terrain as we mentioned in the previous section.

One important feature of our method to be noted here is that our method does not assume any model for L_{sc}; we can use any model from analytical to numerical ones. In this paper, we primarily use a physically-based numerical model, originally developed by Nishita et al. [1993], that is one of the popular models in computer graphics for rendering the realistic daytime sky. We utilize a GPU-accelerated version of the method to render a skyglow image. However, this method requires two integrals to compute the scattered intensity for each point on the viewing ray; one is for the optical depth from the viewpoint and the other for that from the light source. Fortunately, we can precompute the optical depth from the light source to be stored in a two-dimensional table. For more details, please refer to the Appendix A provided as a supplemental document.

The intensity of light reaching the viewpoint is then obtained by accumulating the intensities of scattered light at the sample points generated on the viewing ray. However, the contribution of the light source to the scattered intensity is limited to the region near the light source. So, we reduce the computational cost of the skyglow by the following heuristic approach. The intensity of light incident on a point along the viewing ray from the light source decreases in proportion to the square of the distance between them. When we ignore the attenuation caused by the atmospheric particles, the intensity of the incident light is highest at the point nearest to the light source. We use this point as the starting point for the integral. We denote the nearest point by $P(u_c)$ (see Fig. 2). Then, we split the integration process into two. One is to accumulate the scattered intensity for points with $u > u_c$ and the other is for points with $u \leq u_c$. We terminate the integration process when the intensity of the incident light becomes smaller than a specified threshold. The sampling interval for u is important for accurate computation. We have determined this by trying different sampling intervals. We find that 10m is a good compromise to balance the accuracy and the computational cost, which is used for our examples shown in Section 5.

3.3 Skyglow by Many Light Sources

As we mentioned before, our method takes the density distribution of light sources as input. Let us denote the distribution as L_c $[W/m^2]$. The wavelength λ is sampled at those corresponding to RGB color channels and our method is applied to each of the colors independently. So, we omit λ for simplicity. Then, the intensity of the skyglow, L_s $[W/sr]$, for a given viewing direction is obtained by accumulating b (Eq.1) over the distribution. Since we use the polar coordinate with its origin at the observer location, L_s can be expressed by:

$$L_s(\theta,\phi,h_o,\tau) = \int_0^R \int_0^{2\pi} L_c(r_c,\theta_c) b(\theta-\theta_c,\phi,r_c,h_c,h_o,\tau) r_c d\theta_c dr_c, \quad (3)$$

where (θ,ϕ) is the viewing direction, (r_c,θ_c) and h_c are respectively the horizontal location and the altitude of the light source. R is the maximum distance from the observer to the light source that contributes to the skyglow. Note that h_c is a function of (r_c,θ_c), though we do not explicitly indicate it in the above equation.

Our goal is the fast computation of the above equation, given the distribution of the light sources (L_c) and the turbidity of the atmosphere (τ). This computation includes triple integrals since b is computed by accumulating the scattered intensity along the viewing ray (see Eq.1). Thus, we must compute the triple integrals for every viewing direction (θ,ϕ), which is extremely time-consuming.

Let us now explain our formulation for the fast computation of Eq. 3. Our method is independent of the turbidity, so we omit τ in the following formulation for the simplicity of the explanation. We first apply PCA to b to obtain the basis functions, denoted by $B_k (k = 0, 1, \cdots, M-1)$. The number of the basis functions, M, is determined so that the cumulative contribution rate is larger than a user-specified threshold. Then, b is represented by the following equation.

$$b(\theta,\phi,r,h_c,h_o) \approx \sum_{k=0}^{M-1} c_k(r,h_c,h_o) B_k(\theta,\phi,h_o), \quad (4)$$

where c_k is the coefficient for the basis functions. c_k is obtained by computing the dot product between b and B_k. By putting the above equation into Eq. 3, we obtain:

$$L_s(\theta,\phi,h_o) = \sum_{k=0}^{M-1} L_k(\theta,\phi,h_o), \quad (5)$$

where,

$$L_k(\theta,\phi,h_o) = \int_0^R \int_0^{2\pi} L_c(r_c,\theta_c) c_k(r_c,h_c,h_o) B_k(\theta-\theta_c,\phi,h_o) r_c d\theta_c dr_c. \quad (6)$$

L_k represents the basis distribution of the skyglow. Since B_k does not depend on r_c, we can further rewrite the above equation as:

$$L_k(\theta,\phi,h_o) = \int_0^{2\pi} H_k(\theta_c,h_o) B_k(\theta-\theta_c,\phi,h_o) d\theta_c, \quad (7)$$

where,

$$H_k(\theta_c,h_o) = \int_0^R L_c(r_c,\theta_c) c_k(r_c,h_c,h_o) r_c dr_c. \quad (8)$$

Note that H_k is a function of θ_c and h_o only, since h_c is a function of (r_c,θ_c). Since H_k and B_k are both periodic functions with respect to the azimuth angles (θ and θ_c), Eq. 7 indicates that L_k is obtained by the convolution of H_k and B_k, which can be computed very efficiently using a Fast Fourier Transform (FFT), that is,

$$L_k(\theta,\phi,h_o) = \mathcal{F}_\theta^{-1}\left[\mathcal{F}_\theta[H_k] \times \mathcal{F}_\theta[B_k]\right](\theta,\phi,h_o), \quad (9)$$

where $\mathcal{F}_\theta[\cdot]$ and $\mathcal{F}_\theta^{-1}[\cdot]$ represent the FFT and the inverse FFT operators with respect to the azimuth angle θ. We precompute c_k and $\mathcal{F}_\theta[B_k]$, and use Eq. 9 for fast computation at runtime. Note that we need to repeat the precomputation when the observer altitude (h_o) changes. We thus sample the observer altitude at a regular interval and precompute c_k and $\mathcal{F}_\theta[B_k]$ for each of the sampled values (see Section 3.4). The precomputation could become long, but we emphasize that the precomputed data is independent of the terrain geometry and can be used for any terrain data.

Although the above approach based on basis function expansion is popular in the graphics community, no methods are suitable for the situation treated in this paper. The acceleration by our method comes from not only the basis function expansion but also the careful derivation of the formulation. At high level, PCA removes the redundancy among the skyglow distributions using different city light sources. This is responsible for reducing the storage cost but not for acceleration. One of the important keys to the acceleration in our method lies in Eqs. 6 through 8. The naive method needs to evaluate Eqs. 1 and 3, so it requires a triple integral for each viewing ray. Our method requires a single integral (Eq. 8) for each azimuth direction θ_c to sample the density distribution of light sources to compute H_k, followed by FFT convolution (Eq. 9). This is achieved by decoupling $b(\theta,\phi,r,h_c,h_o)$ into the product of functions of (r,h_c,h_o) and (θ,ϕ,h_o) as in Eq. 4. This seems a simple and straightforward formulation, but we have tried different approaches before reaching the current solution that makes the computation simplest and most efficient. This approach could be extended to similar rendering problems, such as rendering participating media illuminated by many light sources.

The computational cost and the accuracy of our method for the skyglow are discussed in Appendix B in the supplemental document. We have a trade-off between these factors and the discussion should help the user to balance them.

3.4 Implementation

We discretize (θ,ϕ,r,h_c,h_o) at regular intervals, denoted by $(\Delta_\theta, \Delta_\phi, \Delta_r, \Delta_{h_c}, \Delta_{h_o})$, respectively. The density distribution of light sources $L_c(r_c,\theta_c)$ is also represented with the same discretization. Let us denote the indices of the discretized values of these variables by $(i_\theta, i_\phi, i_r, i_{h_c}, i_{h_o})$, where $0 \le i_\theta < N_\theta, 0 \le i_\phi < N_\phi, 0 \le i_r < N_r$, $0 \le i_{h_c} < N_{h_c}$, and $0 \le i_{h_o} < N_{h_o}$. The turbidity of the atmosphere, τ, is also discretized at a regular interval, and the related data is precomputed for each of the sampled turbidity values. When using our system, the user selects one of the sampled turbidity values. While it is possible to interpolate the precomputed data across various turbidity values, our current implementation does not employ this approach.

We apply the computation process illustrated by Fig. 3 to each of the sampled values of the observer altitude. We precompute two 3D textures for each sampled observer altitude to store the basis images and the coefficients for the use on the GPU at runtime process. Therefore, in total, we use $N_{h_o} \times 2$ textures. When the user specifies the observer location at runtime, the skyglow is rendered by using the precomputed textures corresponding to the observer altitude.

In the following explanation of the precomputation and the runtime processes, we do not explicitly denote the dependence of the observer altitude.

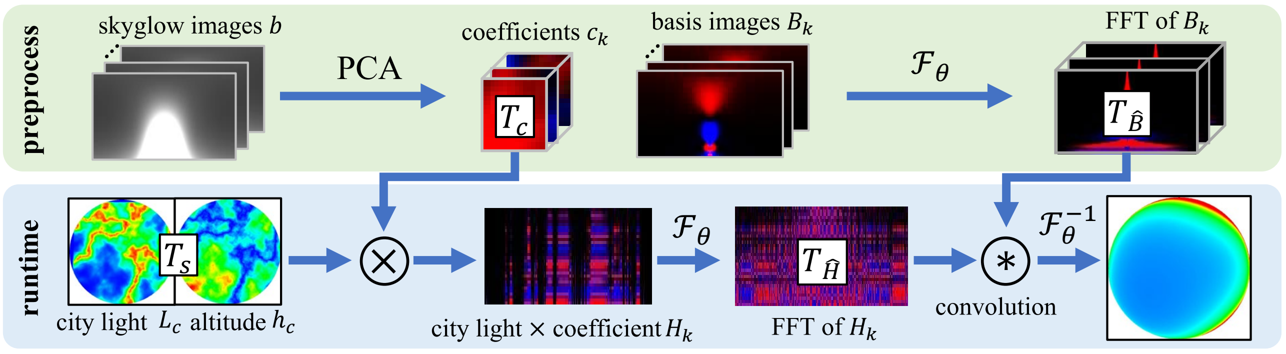


Fig. 3. Implementation details.

3.4.1 *Precomputation.* In the precomputation, we first render a set of images of the skyglow, b, using Eq. 1 for different discrete values of $(r, h_c) = (i_r\Delta_r, i_{h_c}\Delta_{h_c})$. Next, PCA is applied to the rendered images to compute the images of the basis functions, $B_k(\theta, \phi)(k = 0, 1, \cdots, M-1)$. We use the Eigen library [Guennebaud et al. 2010] for the PCA computation. The rendered images are projected onto the basis space to obtain the coefficients $c_k(r, h_c, h_o)$. Next, the FFT operator $\mathcal{F}_\theta$ is applied to the basis images. These computations take from several minutes to a few hours depending on the resolution of the discretization.

As shown in Fig. 3, we denote the texture for the Fourier-transformed basis images as $T_{\hat{B}}$ with the size of $N_\theta \times N_\phi \times M$. Each texel at (i_θ, i_ϕ, k) corresponds to $B_k(i_\theta\Delta_\theta, i_\phi\Delta_\phi)$. The texture for the coefficients are denoted by T_c of size $N_r \times N_{h_c} \times M$ that stores $c_k(i_r\Delta_r, i_{h_c}\Delta_{h_c})$ at (i_r, i_h, k) texel.

3.4.2 *Runtime Process.* The runtime process first extracts a terrain map including the density distribution of the city light sources around the observer from a given database. Usually, the distribution is given with the Cartesian coordinate, so our system converts it into the polar coordinate with the observer at the origin. The converted distribution is stored in the form of a 2D texture denoted by T_s of size $N_\theta \times N_r$. Each texel (i_θ, i_r) stores the light flux L_c and the altitude h_c at $(i_r\Delta_r, i_\theta\Delta_\theta)$.

Next, $H_k(\theta_c)$ is computed according to Eq. 8. The computation is done in parallel on the GPU. To compute H_k, we need to accumulate the light flux L_c multiplied with coefficient c_k along the distance r_c. We first fetch the light flux L_c and the altitude h_c at $(i_\theta\Delta_\theta, i_r\Delta_r)$ from T_s. Next, the coefficient $c_k(r_c, h_c)$ is extracted from texture T_c. These texel values are multiplied and accumulated to obtain $H_k(i_\theta\Delta_\theta)$. FFT operator $\mathcal{F}_\theta$ is applied to the resulting texture, denoted by $T_{\hat{H}}$ whose size is $N_\theta \times M$.

The convolution in the frequency space, $\mathcal{F}_\theta[H_k] \times \mathcal{F}_\theta[B_k]$ is also computed on the GPU. Both of the Fourier transforms of H_k and B_k are already on the GPU memory as textures $T_{\hat{H}}$ and $T_{\hat{B}}$, so we simply multiply the corresponding texels in parallel. The inverse FFT operator $\mathcal{F}_\theta^{-1}$ is then applied to obtain the final image, representing the distribution of the skyglow. Both of $\mathcal{F}_\theta$ and $\mathcal{F}_\theta^{-1}$ are also computed on the GPU.

4 INVERSE INVESTIGATION

This section proposes our interactive investigation tools. That is, inversely to the forward rendering method described in the previous section, the method proposed in this section can efficiently identify the city light sources that affect the light pollution or the skyglow observed at a specified location. The results are visualized in real-time. The user can then adjust the intensity of those city light sources so that the light pollution or the skyglow is weakened to the desired level. The method is based on the PCA-based representation of the skyglow explained in the previous section. We have developed two types of tools. The first one considers the level of the light pollution at a specified location on the terrain (Section 4.1). The second one is for the skyglow intensity observed at a specified location (Section 4.2).

In the following, we assume that the terrain is represented with a regular grid and the location of the center of i-th grid cell is represented with its horizontal location $\mathbf{p}_i$ and altitude h_i. The light flux of the city light located at i-th grid cell is denoted by $L_{c,i}$.

4.1 Inverse Investigation of Light Pollution

We estimate the level of the light pollution with the squared sum of the skyglow intensities over the whole sky, which can be computed approximately by the sum of the PCA coefficients, c_k (see Eq. 4). That is, the light pollution level due to a light source with unit area and unit light flux is estimated by:

$$Q(r, h_c, h_o) = \sqrt{\sum_{k=0}^{M-1} c_k^2(r, h_c, h_o)}, \tag{10}$$

where r is the distance from the observer. h_c and h_o are respectively the altitudes of the light source and the observer. Let us now assume that the observer is located at $\mathbf{p}_o$ (its altitude is h_o). The light pollution level, Q_i $[W]$, due to a light source located at i-th grid cell is then expressed by:

$$Q_i = L_{c,i} Q(|\mathbf{p}_o - \mathbf{p}_i|, h_i, h_o), \tag{11}$$

where $L_{c,i}$ is the light flux of the light source. Eq. 11 indicates the degree of the light pollution caused by i-th light source. Our system visualizes this quantity with a pseudo color for all the light sources on the terrain map, as shown in Fig. 4. We use the GPU for real-time visualization. $Q(r, h_c, h_o)$ is precomputed and used as a two

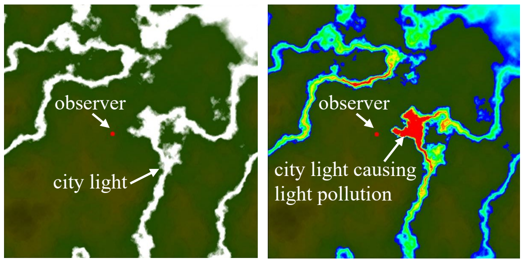


Fig. 4. Inverse visualization of light pollution. The left is a normal view showing a terrain map with city light distribution overlaid. The right image shows the influence of the light sources on the light pollution at the observer location.

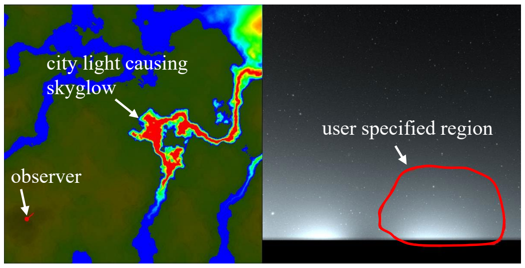


Fig. 5. Inverse visualization of skyglow. The left image shows the influence of the light sources on the skyglow indicated by the red curve specified by the user shown in the right image.

dimensional texture. Then, Eq. 11 is computed in parallel on the GPU. The user can move the observer location interactively to investigate the influences of the surrounding light sources on the light pollution at the observer.

Our system allows the user to reduce the city light intensity to investigate how city light affects the light pollution. We employ the following strategy. We first calculate the reduction rate γ_i for each of the light sources with the following equation:

$$\gamma_i = \frac{Q_i - Q_a}{Q_b - Q_a}\gamma_{usr}, \tag{12}$$

where Q_a, Q_b, and γ_{usr} are specified by the user. Note that both γ_i and γ_{usr} are limited to a value between 0 and 1. Then, the intensity of the light source is reduced by: $L_{c,i} \leftarrow L_{c,i} - \gamma_i L_{c,i}$. This strategy is based on a policy that the reduction rate should be proportional to the level of the light pollution; the rate for the light source with high influence is larger than that with low influence. Q_a and Q_b control the allowable range of the light pollution caused by each light source. When γ_{usr} is one, the light sources with $Q_i > Q_b$ are eliminated. The light sources with $Q_i < Q_a$ are on the other hand unchanged with any value of γ_{usr}.

4.2 Inverse Investigation of Skyglow

The method proposed in this section allows the user to identify the light sources that affect the skyglow displayed on the screen. The user can specify the region on the sky dome by drawing a closed curve (see Fig. 5). The system then computes the influence of the surrounding light sources on the skyglow inside the closed curve.

Let us denote the set of viewing directions for points inside the curve by $(\theta_l, \phi_l)(l = 0, 1, \cdots, N_d)$, where N_d is the number of directions. The contribution of light source at i-th grid cell to direction (θ_l, ϕ_l) can be computed by:

$$s_i(\theta_l, \phi_l) = L_{c,i} b(\theta_l - \theta_i, \phi_l, r_i, h_i, h_o), \tag{13}$$

where θ_i is the azimuth direction of the light source viewed from the observer. r_i is the distance from the observer to the light source, and h_i is its altitude. s_i can be computed by using the PCA basis functions (see Eq. 4). The influence of i-th light source on the region inside the curve is measured by the following equation:

$$S_i = \sum_l s_i(\theta_l, \phi_l) \tag{14}$$

Our system visualizes S_i with a pseudo color, as shown in Fig. 5. We again use the GPU for real-time visualization.

In the similar way to Section 4.1, we allow the user to weaken the skyglow by reducing the intensities of the surrounding light sources. We employ the same strategy as the one described in the last paragraph of Section 4.1, but we use S_i instead of Q_i.

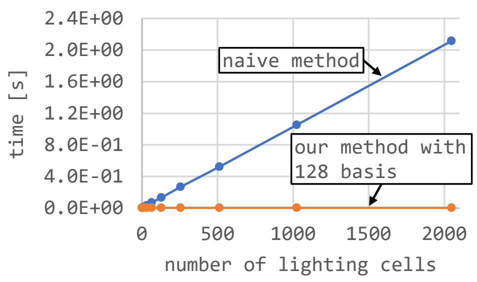
5 RESULTS

In this section, we demonstrate the effectiveness of our method and show some application results. For the results shown in this section, we use a desktop PC with an Intel Core i9 3.00 GHz (CPU) and an NVIDIA Quadro RTX 6000 (GPU). The memory capacities of the CPU and the GPU are 128 GB and 24 GB, respectively. We use a virtual terrain model for the following example. The size of the terrain is 10km×10km×1.0km. For the discretization, we choose the following settings: $N_\theta = 256$, $N_\phi = 64$, $N_r = 128$, $N_{h_s} = 16$, and $N_{h_o} = 16$. The sampling intervals are :$\Delta_\theta = 2\pi/256$ radian, $\Delta_\phi = \pi/128$ radian, $\Delta_r = 78.125$ m, and $\Delta_{h_s} = \Delta_{h_o} = 62.5$ m. The precomputation took about 4 - 6 hours for each sampled turbidity. The time changes depending on the number of basis functions. Note that this precomputation data can be used for any terrain data as long as the size is the same. For the runtime process, we render a panorama image of the skyglow corresponding to the upper half of the sky and use it as an environment map. The resolution of the map is $N_\theta \times N_\phi = 256 \times 64$. Using this resolution, the size of each pixel corresponds to 1.4 degrees. The following examples use the precomputed data for a small turbidity value, 1.3, that corresponds to a clear sky. Comparison of images with different turbidity values can be found in the supplemental document (Appendix C). We use a star map (Deep Star Maps 2020) for rendering the stars, which has been made publicly available by NASA.[1].

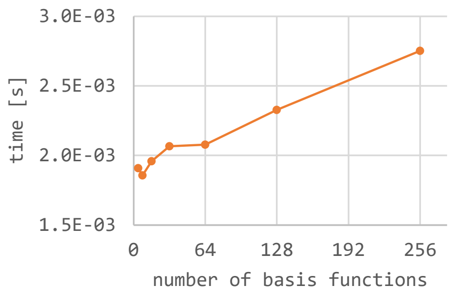
5.1 Performance Evaluation

We investigate the efficiency and the accuracy of our method. We measured the computation times for the runtime process (the second row of Fig. 3) and the average relative errors to reference solutions. The results are shown in Figs. 6 and 7. In these experiments, the

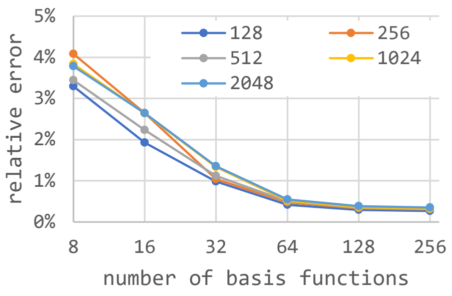
[1]NASA/Goddard Space Flight Center Scientific Visualization Studio. Gaia DR2: ESA/Gaia/DPAC. Constellation figures based on those developed for the IAU by Alan MacRobert of Sky and Telescope magazine (Roger Sinnott and Rick Fienberg).



(a) computation time and number of lighting cells.



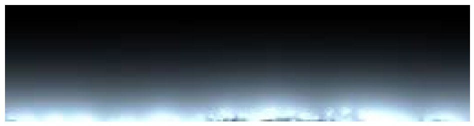
(b) computation time and number of basis functions.

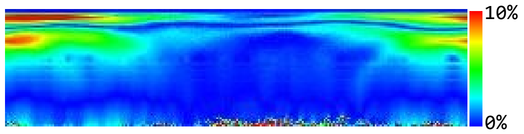


(c) average relative error and number of basis functions.

Fig. 6. Computation time and accuracy.

(a) naive method.


(b) our method with 32 basis functions.



(c) relative error.

Fig. 7. Comparison of skyglow images with and without our method for 1,024 lighting cells.

reference solutions are calculated by a naive method that directly integrates the scattered intensity (Eq. 3). For the reference solution, only cells with a positive density of light sources are taken into consideration. We refer to them as "lighting cells" in the following. Note that the details of the measured timing and errors can be found in the supplemental document (Appendix D).

Fig. 6(a) shows the computation times of our method and the naive method. For our method, we use 128 basis functions. We measured the computation times for different numbers of lighting cells, that is, 1, 2, 4, $\cdots$, 2048. The locations and the light fluxes of the lighting cells are randomly determined. We computed the skyglow images a hundred times and Fig. 6(a) shows the average of them. As indicated by the figure, the computation time of the naive method is linearly proportional to the number of the lighting cells. The figure also demonstrates that our method is almost independent of the number of the lighting cells. As the number increases, our method becomes efficient by a factor of one to two orders of magnitudes. In the case of 2,048 lighting cells, our method achieves 910 times faster computation. This property is preferable for visualizing the light pollution since we must consider many city light sources.

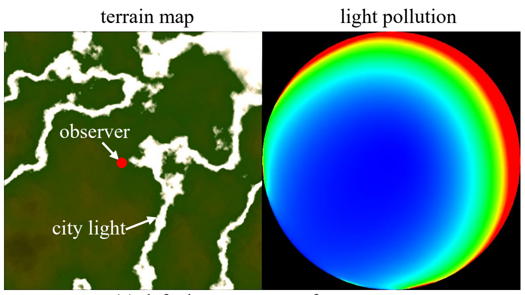
Next, Fig. 6(b) shows the computation times for the runtime process of our method with different numbers of basis functions, that is, 8, 16, $\cdots$, 256. The number of the lighting cells is fixed at 2,048. The time increases linearly when the number of the basis functions is greater than 64. When the number is smaller, the relationship between the time and the number of the basis functions seems nonlinear. We have not investigated the reason, but we assume that we cannot fully exploit the parallel computation power of the GPU for such a smaller number of basis functions. However, even for 256 basis functions, the computation time is less than 3ms which is sufficient for real-time visualization.

Fig. 6(c) shows the average relative errors of our method with different numbers of basis functions and the lighting cells. We measured the errors for 128, 256, 512, 1024, and 2048 lighting cells; each plot in this figure corresponds to each of these. The horizontal axis corresponds to the number of basis functions. For all the cases, the error is less than 5%.

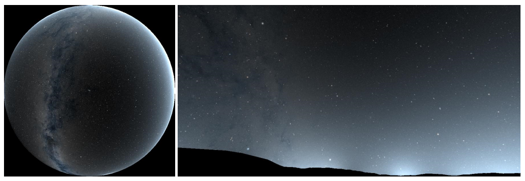
Finally, Fig. 7 compares panorama images of the skyglow with and without using our method. Figs. 7(a) and (b) were respectively generated by the naive method and our method. Our method uses 32 basis functions. The storage size for the precomputed data in this case is 108 MB. The original storage size of the skyglow distributions is 2,048 MB. The storage size is thus reduced to 1/19 by our method. Fig. 7(c) shows the relative error between them. Although the relative error is high in the upper region, the intensity of the skyglow there is small as shown in Figs. 7(a) and (b).

5.2 Applications

We developed a simple interactive visualization system using our method. Although we use a white color for the light source, the color of the skyglow becomes bluish due to the atmospheric scattering. The user can optionally change the color of the light sources.

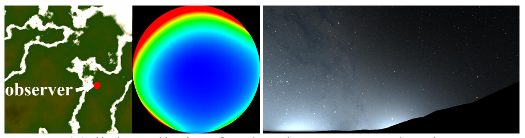


(a) default appearance of our system.

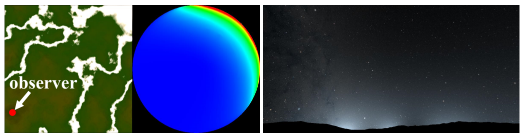


(b) realistic display. (c) view from the observer.

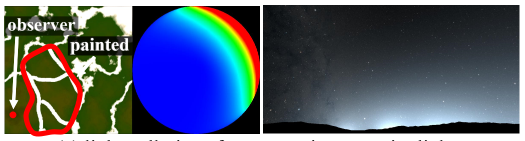
Fig. 8. Our light pollution visualization system.



(a) light pollution for the observer near the city.



(b) light pollution for the observer far from the city.



(c) light pollution after generating new city lights.

Fig. 9. Forward investigation.

Fig. 8(a) shows the appearance of our system. On the left, the terrain is displayed with a pseudo-color representing the altitude of every location. The city light distribution is overlaid with the color of the light. The user can interactively specify the location of the observer from which she/he wants to visualize the light pollution. The location of the observer is displayed with a small red dot. The light pollution is visualized in real-time on the right with a pseudo-color representing the intensity of the skyglow. A realistic display of the light pollution is also possible with the stars overlayed as shown in Fig. 8(b). The user can also switch to the perspective visualization of the sky viewed from the observer location as shown in Fig. 8(c). Our system can be used for different purposes in the context of light pollution. The following examples demonstrate some application cases of our system.

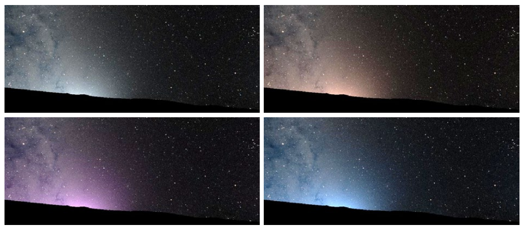
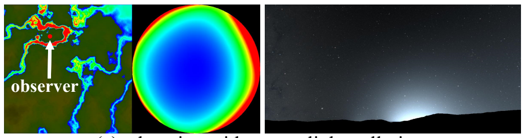


Fig. 10. Effects of the color of light sources on the skyglow.

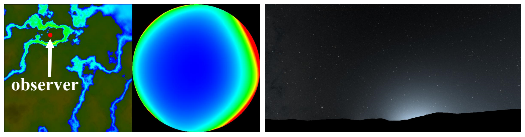
A direct application of our system is astronomical observation. Astronomers can use the system to find a dark place for a better observatory location. Fig. 9 shows such an example. The user can explore different places interactively to find a suitable place for the observation. Figs. 9(a) and (b) show visualization results for two observer locations. For the location near the city as in Fig. 9(a), the sky is severely polluted, so this is not a good place. The location far from the city shown in Fig. 9(b) is better since the light pollution is reduced. Next, in Fig. 9(c), the user investigates how the light pollution increases when new city light sources are generated. Our system allows the user to increase/decrease the intensity of the city light with a painting operation. This function is useful for city planning. We would like to additionally mention that the system can be helpful not only to find better places for fixed observatories, but also for ’on purpose’ observations where one can travel to see an object optimally. As an example, a comet that appears in a particular azimuth direction can be observed optimally if one finds a dark sky in that direction.

Next, our system is useful for the city light planning as well. In Fig. 10, the user investigates the influence of the color of light sources on the skyglow. Our system allows the user to choose an arbitrary color for the light sources and the images are rendered in real-time. In this example, four different colors, white, orange, pink, and blue, are chosen but the luminance is the same for all these colors. We see the appearance of the nigh sky changes depending on the color of the light sources. This function allows the city light planners to choose appropriate spectral distribution of the light sources to minimize the influence of light pollution on the visual quality of the night sky.

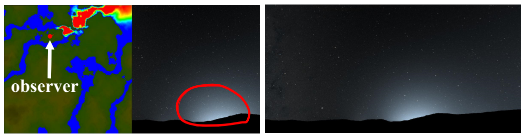
Fig. 11 shows another application case for the city light planners using our inverse tools. The city light planners can benefit from the tools to optimize and design lighting layouts, such as streetlamps, to minimize light pollution and enhance the overall quality of urban lighting. Fig. 11(a) shows a location with severe light pollution. The left shows the light sources causing the light pollution with the pseudo color. The right image shows a panorama view from that location. In Fig. 11(b), the user reduces the intensity of the light sources by using our method described in Section 4.1. However, the annoying skyglow is still visible as shown in the right image. In Fig 11(c), the user specifies the region of the annoying skyglow with the red curve. Then the system displays the light sources causing the skyglow with the pseudo color as shown in the left image. In Fig. 11(d), the user reduces the intensities of those light sources and the skyglow is reduced.



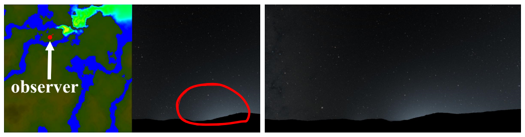
(a) a location with severe light pollution.



(b) light sources with large influences are weakened.



(c) light sources with large influence on the skyglow.



(d) the skyglow is fairly reduced.

Fig. 11. Inverse investigation.

All the images shown in this section are displayed in real-time according to the user's operation. Please refer to the supplemental video for the demonstration of our system working in real-time.

5.3 Experiments with Multiple Scattering

We have so far used the single scattering model to evaluate our method. However, our method is not limited to specific scattering models and can actually work with more advanced ones. This section demonstrates this by presenting experiments involving multiple scattering.

We utilize the physically-based path tracer, mitsuba3, to compute the multiple scattering for the skyglow [Jakob et al. 2022]. Our preprocess then starts by computing a set of skyglow images using this path tracer. Due to the computationally expensive nature of multiple scattering, we reduced the discretization setting for (N_r, N_{h_s}, N_{h_o}) to $(32, 16, 16)$, resulting in the precomputation of 8,192 images. Without this adjustment, the precomputation time would become impractically long. We rendered images with 4,096 samples per pixel; however the images still exhibited noise. To address this, we applied a Gaussian blur filter with the kernel size of five pixels. PCA is subsequently applied to compute the basis images. The precomputation took 34 hours, mostly spent on the computation of the skyglow images using the path tracer, while PCA took only 44 minutes. After the precomputation, our method can render the skyglow in real-time and the rendering speed is the same as that shown in the previous section.

Fig. 12 shows comparisons between the panorama images rendered using our method and those generated by the path tracer. We selected three observer locations, (a), (b), and (c), for these comparisons, as indicated in the left most image. The images in the

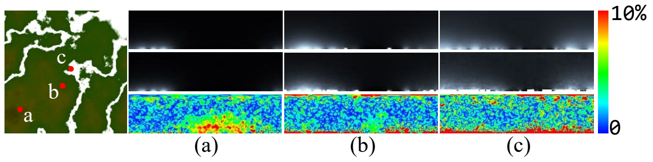


Fig. 12. Skyglow images with multiple scattering. The leftmost image shows the three locations (a), (b), and (c), respectively for computing the skyglow images on the right. The images in the top row are generated by our method. The second row shows the reference images computed by Mitsuba3. The bottom row shows the difference images presented with the pseudo color scheme.

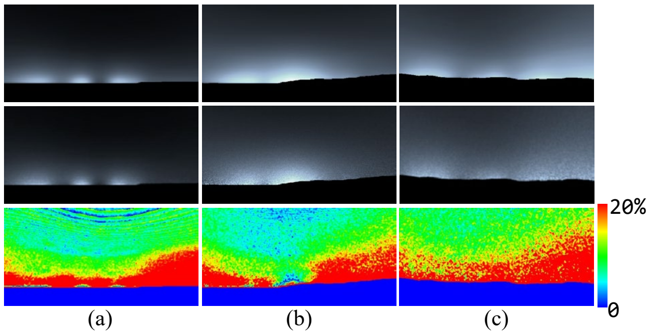


Fig. 13. Skyglow images as viewed from the three observer locations. The images in the top row are generated by our method. The second row shows the reference images computed by Mitsuba3. The bottom row shows the relative error presented with the pseudo color scheme.

top and second rows were respectively generated by our method and the path tracer. The bottom images show the relative errors presented using a pseudo color scheme. We use 32 basis functions for our method. Note that the occlusion by the terrain is not taken into account in these comparisons. Since the images by the path tracer exhibited severe noise even with 4,096 samples per pixel, we rendered each image four times and computed an average of them. The Gaussian filter was then applied to the result. Although the noise cannot be completely removed, we can confirm that our method produces accurate images as shown in the error images. The path tracer took 130 seconds on average to create a single image. Our method achieves more than 10^4 faster computation for these particular examples.

Fig. 13 shows a comparison of images as observed from the viewer's perspective. The observer locations for (a), (b), and (c) are the same as those in Fig. 12. In this comparison, however, the images generated by the path tracer consider the occlusion caused by the terrain, while our method does not account for it. While the rendered images appear similar as shown in the top and second rows, there are noticeable relative errors near the horizon as shown in the bottom row. This is one of the limitations of our method; we discuss possible solutions to this problem in Section 6.

Next, we further investigate the capability of our method through an additional experiment similar to Fig. 7, but this time taking into account the multiple scattering in the computation of the skyglow.

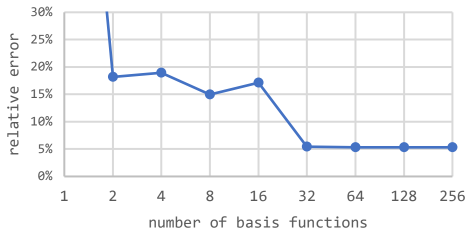


Fig. 14. Average relative error and number of basis functions for skyglow computed with multiple scattering.

We assume a flat ground and generate light sources randomly. We compute panorama images of the skyglow with different number of basis functions and compare them with the reference image computed by the path tracer. The results are summarized in Fig. 14, demonstrating that our method is accurate when utilizing more than 32 basis functions. Note that the average relative error with 8 basis functions was accidentally smaller than that with 16 basis functions, but severe artifacts appeared in the skyglow image due to the small number of the basis functions.

6 DISCUSSION

While the examples in the previous section focused on the investigation of light pollution, it is worth noting that our method can offer several applications in the context of light pollution. A good example is a flight simulator. Given the rapid rendering capabilities of our method, it can be seamlessly integrated with real-time flight simulators, which enables a visual assessment of how city lights impact flights. In addition to such a practical application, we could use our system for a more artistic purpose; we could use our system to design the appearance of the night sky. For example, we can extend the method in Section 4.2 to inversely determine the color distribution of light sources to allow the user to paint the sky with arbitrary colors. Another practical application is the design of lens filters for capturing the beauty of starry photographs. Common city lighting fixtures emit light with characteristic spectral distributions. Filters that selectively block light within specific spectral bands are often utilized for photographers of the night sky. Our system can be used for evaluating their performance during the design process. However, in our current implementation, we can compute the skyglow for three color channels (RGB) only. The computation could become excessive when we consider more spectral bands. We may be able to reduce the computational cost by projecting the spectral distribution of the light sources onto some basis functions [Peercy 1993].

One limitation of our method is that it does not account for light occlusion caused by terrain. An approximate solution to this limitation involves multiplying the ratio of the occluded region to the entire sky. However, precise solutions for this issue remain a subject of our future research. We may draw insights from the existing literature on precomputed radiance transfer methods [Ramamoorthi 2009].

Another limitation is the long precomputation time, particularly when using fine sampling intervals for the distance and the altitude. When considering the single scattering only, PCA consumes a significant portion of the time. To address this problem, we plan to use faster methods like power iteration. However, when considering multiple scattering, the computation time for rendering skyglow images also increases significantly. We can speed up this computation by taking advantage of the fact that the skyglow distribution is symmetric around the light source. However, this is part of our future research.

One might think that we could use common basis functions like spherical harmonics or wavelets. Then PCA process would be no longer needed. However, we found that PCA offers more accurate representations of skyglow. Since the skyglow due to a single light source has a sharp peak around the light source, the spherical harmonics fails to approximate it accurately. Similarly, for the wavelets, we experimented with Haar wavelets, but encountered blockwise artifacts appeared unless we used a large number of basis functions. In contrast, PCA resolved all these problems since PCA provides better basis functions that are well-suitable for the given dataset. Additionally, we would like to emphasize that our PCA basis functions are not terrain-dependent and can be applied to any terrain map. Although PCA takes longer for precomputation, the benefits it offers outweigh this drawback.

An important research direction is using our system to design the city lighting devices, such as streetlamps. In this context, three factors need to be considered: brightness, light distribution curve, and spectral distribution. These factors collectively influence the appearance of the night sky. Although we have already demonstrated examples for the influence of the brightness and the spectral distribution (color) of light sources in Section 5.2, the light distribution curves are not considered. We would like to extend our method to the light distribution curves in the future.

Another interesting research direction is the detailed modeling of the city light sources. Currently, each city light is approximated with a distribution of light sources. Although we can analyze the overall property of the light pollution with this approximation, in reality city lights are much more complicated, containing potentially millions of different lighting devices. Modeling of a realistic distribution considering individual lighting devices in the city is an interesting future problem.

7 CONCLUSIONS

We have proposed and implemented a fast visualization method for the night sky considering the light pollution. Our method precomputes the skyglow distributions illuminated by an infinitesimal area light located at different locations. The precomputed distributions are then projected onto the PCA basis functions, followed by the FFT operator. This allows us to efficiently compute the intensity of the skyglow illuminated by many city light sources. The inverse investigation method has also been proposed based on our fast computation method of the skyglow. We have developed an interactive system based on our method that visualizes the light pollution in real-time for arbitrary observer locations. The user can also add/delete the city light sources interactively.

As for the future work, we are planning to use a public database [Elvidge et al. 2021] for the distribution of the city light sources[2]. This database stores the average light flux of night cities measured by satellites over the world. By combining this database and an open mapping platform (like Google map), we wish to develop a web service that visualizes the light pollution at any location on the Earth.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their constructive comments to improve our paper. We also thank Dr. Fabio Falchi, Dr. Miroslav Kocifaj, and Dr. Nobuaki Ochi for their helpful discussion on the application of our method to light pollution research. This work was supported by JSPS KAKENHI Grant Numbers JP20H05954 and JP21H03571.

REFERENCES

Ryan Avery, Stanley Yu, Kenton Ross, Veronica Warda, and Steven Chao. 2017. *Skyglow Estimation Toolbox — SET 0.0.1 Documentation.* https://nasa-develop.github.io/SET/

Salvador Bará, Fabio Falchi, Riccardo Furgoni, and Raul C. Lima. 2020. Fast Fourier-transform calculation of artificial night sky brightness maps. *Journal of Quantitative Spectroscopy and Radiative Transfer* 240 (2020), 106658. https://doi.org/10.1016/j.jqsrt.2019.106658

Salvador Bará, Salvador J Ribas, and Miroslav Kocifaj. 2015. Modal evaluation of the anthropogenic night sky brightness at arbitrary distances from a light source. *Journal of Optics* 17, 10 (aug 2015), 105607. https://doi.org/10.1088/2040-8978/17/10/105607

Eric Bruneton and Fabrice Neyret. 2008. Precomputed Atmospheric Scattering. *Computer Graphics Forum* 27, 4 (2008), 1079–1086. https://doi.org/10.1111/j.1467-8659.2008.01245.x

P. Cinzano and F. Falchi. 2012. The propagation of light pollution in the atmosphere. *Monthly Notices of the Royal Astronomical Society* 427, 4 (2012), 3337–3357. https://doi.org/10.1111/j.1365-2966.2012.21884.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2966.2012.21884.x

Yoshinori Dobashi, Tomoyuki Nishita, and Tsuyoshi Yamamoto. 2002. Interactive rendering of atmospheric scattering effects using graphics hardware. In *Proceedings of Graphics Hardware.* 99 – 108.

Christopher D. Elvidge, Mikhail Zhizhin, Tilottama Ghosh, Feng-Chi Hsu, and Jay Taneja. 2021. Annual Time Series of Global VIIRS Nighttime Lights Derived from Monthly Averages: 2012 to 2019. *Remote Sensing* 13, 5 (2021), 922. Issue 5. https://doi.org/10.3390/rs13050922

Fabio Falchi and Salvador Bará. 2021. Computing light pollution indicators for environmental assessment. *Natural Sciences* 1, 2 (2021), e10019. https://doi.org/10.1002/ntls.10019 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ntls.10019

R. H. Garstang. 1986. Model for Artificial Night-Sky Illumination. *Publications of the Astronomical Society of the Pacific* 98 (1986), 364. https://doi.org/10.1086/131768

Kevin J. Gaston, Jonathan Bennie, Thomas W. Davies, and John Hopkins. 2013. The Ecological Impacts of Nighttime Light Pollution: A Mechanistic Appraisal. *Biological Reviews* 88, 4 (2013), 912–927. https://doi.org/10.1111/brv.12036

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Sébastien Hillaire. 2020. A Scalable and Production Ready Sky and Atmosphere Rendering Technique. *Computer Graphics Forum* 39, 4 (2020), 13–22.

Lukas Hosek and Alexander Wilkie. 2012. An Analytic Model for Full Spectral Sky-Dome Radiance. *ACM Trans. Graph.* 31, 4, Article 95 (jul 2012), 9 pages. https://doi.org/10.1145/2185520.2185591

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022. *Mitsuba 3 renderer.* https://mitsuba-renderer.org.

Henrik Wann Jensen, Frédo Durand, Julie Dorsey, Michael M. Stark, Peter Shirley, and Simon Premože. 2001. A Physically-Based Night Sky Model. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '01* (2001). ACM Press, 399–408. https://doi.org/10.1145/383259.383306

K. Kaneda, T. Okamoto, E. Nakamae, and T. Nishita. 1991. Photorealistic Image Synthesis for Outdoor Scenery under Various Atmospheric Conditions. *The Visual Computer* 7, 5-6 (1991), 247–258.

R. V. Klassen. 1987. Modeling the effect of the atmosphere on light. *ACM Transactions on Graphics* 6, 3 (1987), 215–237.

M. Kocifaj. 2007. Light-Pollution Model for Cloudy and Cloudless Night Skies with Ground-Based Light Sources. *Applied optics* (2007). https://doi.org/10.1364/AO.46.003013

Miroslav Kocifaj. 2018. Multiple scattering contribution to the diffuse light of a night sky: A model which embraces all orders of scattering. *Journal of Quantitative Spectroscopy and Radiative Transfer* 206 (2018), 260–272. https://doi.org/10.1016/j.jqsrt.2017.11.020

Miroslav Kocifaj and František Kundracik. 2016. Modeling the Night Sky Brightness Distribution via New SkyGlow Simulator. In *2016 IEEE Lighting Conference of the Visegrad Countries (Lumen V4)* (2016-09). 1–3. https://doi.org/10.1109/LUMENV.2016.7745553

Tom Minor, Robert R. Poncelet, and Eike Falk Anderson. 2016. Skyglow: Towards a Night-time Illumination Model for Urban Environments. In *EG 2016 - Posters*, Luis Gonzaga Magalhaes and Rafal Mantiuk (Eds.). The Eurographics Association. https://doi.org/10.2312/egp.20161055

Tomoyuki Nishita, Takao Sirai, Katsumi Tadamura, and Eihachiro Nakamae. 1993. Display of the Earth Taking into Account Atmospheric Scattering. In *Proceedings of ACM SIGGRAPH 1993.* 175–182.

Sean O'Neil. 2007. *Accurate atmospheric scattering.* Addison-Wesley Professional.

Mark S. Peercy. 1993. Linear Color Representations for Full Speed Spectral Rendering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA) *(SIGGRAPH '93)*. Association for Computing Machinery, New York, NY, USA, 191–198. https://doi.org/10.1145/166117.166142

A. J. Preetham, Peter Shirley, and Brian Smits. 1999. A Practical Analytic Model for Daylight. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 91–100. https://doi.org/10.1145/311535.311545

Ravi Ramamoorthi. 2009. *Precomputation-Based Rendering.* NOW Publishers Inc.

Pinar Satilmis, Thomas Bashford-Rogers, Alan Chalmers, and Kurt Debattista. 2017. A Machine-Learning-Driven Sky Model. *IEEE Computer Graphics and Applications* 37, 1 (2017), 80–91. https://doi.org/10.1109/MCG.2016.67

Alexandre Simoneau, Martin Aubé, Jérôme Leblanc, Rémi Boucher, Johanne Roby, and Florence Lacharité. 2021. Point spread functions for mapping artificial night sky luminance over large territories. *Monthly Notices of the Royal Astronomical Society* 504, 1 (04 2021), 951–963. https://doi.org/10.1093/mnras/stab681 arXiv:https://academic.oup.com/mnras/article-pdf/504/1/951/37251093/stab681.pdf

Héctor Antonio Solano Lamphar. 2018. The Emission Function of Ground-Based Light Sources: State of the Art and Research Challenges. *Journal of Quantitative Spectroscopy and Radiative Transfer* 211 (2018), 35–43.

Bo Sun, Ravi Ramamoorthi, Srinivasa G. Narasimhan, and Shree K. Nayar. 2005. A practical analytic singlescattering model for real time rendering. *ACM Transactions on Graphics* 24, 3 (2005), 1040–1049. https://doi.org/10.1145/1073204.1073309

Carsten Wenzel. 2007. Real time atmospheric effects in games revisited. In *Game Developers Conference.*

Alexander Wilkie, Petr Vevoda, Thomas Bashford-Rogers, Lukáš Hošek, Tomáš Iser, Monika Kolářová, Tobias Rittig, and Jaroslav Křivánek. 2021. A Fitted Radiance and Attenuation Model for Realistic Atmospheres. *ACM Trans. Graph.* 40, 4, Article 135 (jul 2021), 14 pages. https://doi.org/10.1145/3450626.3459758

[2]The database is available at https://eogdata.mines.edu/products/vnl/