Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2014, Article ID 308629, 13 pages
http://dx.doi.org/10.1155/2014/308629

Research Article

Foggy Scene Rendering Based on Transmission Map Estimation

Fan Guo, Jin Tang, and Xiaoming Xiao

School of Information Science and Engineering, Central South University, Changsha, Hunan 410083, China

Correspondence should be addressed to Xiaoming Xiao; xiaoxiaomingcsu@163.com

Received 4 May 2014; Revised 3 September 2014; Accepted 29 September 2014; Published 19 October 2014

Academic Editor: Mark Green

Realistic rendering of foggy scene is important in game development and virtual reality. Traditional methods have many parameters to control or require a long time to compute, and they are usually limited to depicting a homogeneous fog without considering the foggy scene with heterogeneous fog. In this paper, a new rendering method based on transmission map estimation is proposed. We first generate perlin noise image as the density distribution texture of heterogeneous fog. Then we estimate the transmission map using the Markov random field (MRF) model and the bilateral filter. Finally, virtual foggy scene is realistically rendered with the generated perlin noise image and the transmission map according to the atmospheric scattering model. Experimental results show that the rendered results of our approach are quite satisfactory.

1. Introduction

Visualization of atmospheric phenomena is an important use of computer graphics, and among of these, fog is one of the most common scenes in game scene and virtual reality. A realistic fog will greatly improve the reality of virtual scenes and add attractiveness to the generated scenes when created. Thus, many applications of the simulated fog in computer games, virtual reality, special effects in movies and TV, and so forth can be found.

The simulation in real-time is usually limited to depicting a homogeneous fog with constant density distribution. However, in the natural world, the phenomenon is more complex. It is a participating medium made of many non-homogenous layers moving with the wind and undergoing turbulent motion. Thus, this paper discusses the problem of fog simulation and visualization, with special emphasis on generating heterogeneous fog in real-time using image processing method.

The organization of this paper is as follows. We begin by reviewing existing works on the foggy scene simulation. In Section 3 we introduce the atmospheric scattering model and the Perlin noise. In Section 4 we propose the foggy scene rendering algorithm based on the estimated transmission map. In Section 5 we present some experimental results. Finally, in Section 6 we give some including remarks.

2. Related Works

There are several related works concerning the simulation of foggy scene. These existing methods can be divided into two categories: software-based methods and model-based methods. The software-based methods generate rendering effects using software tools. For example, the software tools, such as Photoshop, 3ds max, and OpenGL, can be used to simulate foggy scenes. However, although these tools can convert an input no-fog image to a virtual foggy image, the whole procedure has many steps and each step needs user involvement, which is very hard to control and requires good skills.

A more feasible and effective rendering method is the model-based method. Modeling the physical mechanism of fog makes it possible to simulate foggy conditions. Max [3] introduced a single scattering model for light diffusion to generate haze in the atmosphere. Yamamoto et al. [4] proposed a fast rendering method for atmospheric scattering effects by using graphics hardware. Jackèl and Walter [5] simulated the Mie Scattering in 1997, and Nishita et al. [6] used an improved sky illumination to render the fog scene. But this approach of calculating the scattering effect for air particles is quite time-consuming. Sun et al. [1] proposed a practical analytic method to render the single scattering in fog, mist, and haze in real-time. Wang et al. [7] proposed

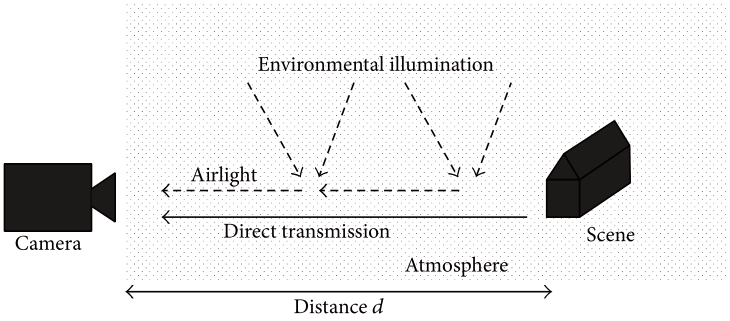


FIGURE 1: Scattering of light by atmospheric particles.

a sky light model considering atmospheric scattering and refraction. Based on the model, various sky scenes, including foggy day and other conditions, are realistically rendered in real-time. Cao et al. [2] present a new atmospheric point spread function to simulate foggy scene for natural images. However, the above methods have many parameters to control owing to the complex physical models they used or require a number of advanced computational steps and a long time to compute. To reduce the algorithm parameters and computational complexity, a simple and elegant model proposed by Koschmieder et al. [8, 9] is applied to solve the foggy scene simulation problem. Zdrojewska [10] used the Koschmieder model (also called atmospheric scattering model) to synthesize the fog phenomenon by using hardware acceleration in the form of a GPU. Dong et al. [11] proposed an algorithm to simulate the foggy scene for a real image. The method first segments the input image into several semantic regions and then estimates depths for these regions and synthesizes virtual fog using the Koschmieder model.

For the rendering of foggy image based on the Koschmieder model, the most important step is to estimate the transmission map of the model. Besides the color, texture, and geometric position used in Dong's method [11], there are many other features that can be taken as depth cues for the transmission map estimation. These features include focus/defocus, motion parallax, relative height/size, and texture gradient. For example, Hoiem et al. [12] generated maps using machine learning algorithms. The transmission values of scene and object are assigned according to the trained classes. Park and Kim [13] determined the distance from the focal plane by using the blurriness of low depth-of-field in optical physics. Jung et al. [14] assigned transmission maps by using edge information and prior depth knowledge. Han and Hong [15] generated the transmission map employing both vanishing points and super pixels as geometric and texture cues. However, the above methods are unreliable when the selected cues are weak in an input image. Yang et al. [16] generated a depth map using the local depth hypothesis based on the structural information of the input image and salient regions, but user interaction is required for this method.

Generating a transmission map from single image is an ill-posed problem. Not all the depth cues can be retrieved from an image. To overcome the challenge, in this work, we present a novel method to automatically generate a pseudo transmission map which reflects the depth information in the scene of the input image. Based on the estimated map, a virtual foggy scene can be simulated for both game scene and natural images. To improve the realism of simulated phenomenon, Perlin noise is also introduced in our method to generate heterogeneous fog. Experimental results indicate that the proposed algorithm may generate realistic rendered results.

3. Physical Characters of Foggy Scenes

To render the foggy scene, we first analyze the physical characters of a foggy image, including the model that describes the formation of a foggy image and the noise used for generating heterogeneous fog.

3.1. Atmospheric Scattering Model. Light passing through a scattering medium is attenuated and distributed to other directions. This can happen anywhere along the path and leads to a combination of radiances incident towards the camera, as shown in Figure 1. The atmospheric scattering model widely used to describe the formation of the foggy image is as follows [8, 9]:

$$I(\mathbf{x}) = J(\mathbf{x})\, t(\mathbf{x}) + A\,(1 - t(\mathbf{x}))\,, \tag{1}$$

where this equation is defined on the three RGB color channels. I stands for the observed image, A is the airlight color vector, and J is the surface radiance vector at the interaction point of the scene and the real world ray corresponding to the pixel $\mathbf{x} = (x, y)$. $t(\mathbf{x})$ is called the transmission map and expresses the relative portion of light that manages to survive the entire path between the observer and a surface point in the scene without being scatted. Theoretically, the goal of fog scene rendering is to calculate J from the no-fog image I, the estimated transmission t, and the airlight A.

3.2. Perlin Noise for Generating Heterogeneous Fog. The natural phenomena usually do not change in regular ways but are characterized by large degree of randomness. Such feature is also present in fog, which is a volume object of variable density, taking irregular shapes due to wind and air turbulence. Because of that, random noise function seems like a good candidate to help simulate it. An argument to

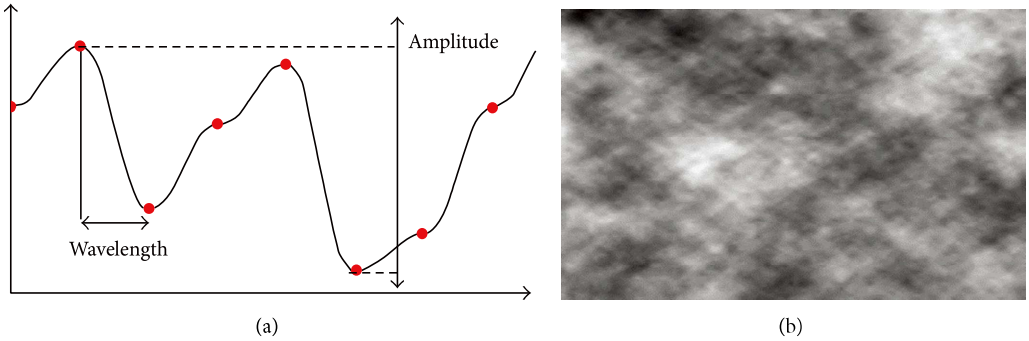


FIGURE 2: Perlin noise for generating foggy scene. (a) Amplitude and frequency of the noise function. (b) Perlin's turbulence texture.

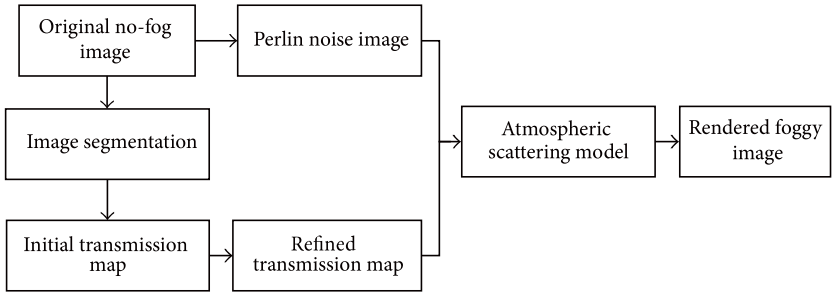


FIGURE 3: Flowchart of the algorithm.

the function is two- or three-dimensional coordinates of a point in space, and the result is a pseudorandom value of fog's density at these coordinates. Noise generated by such function has a high frequency and hence displays rapid changes between contrasting values, which is not typical for fog density distribution. Therefore, it should be rather modeled with smooth noise created by interpolation of random value samples sequence. The method of generating such noise was proposed by Perlin in [17].

The noise function is characterized by two properties: amplitude and frequency. As seen in Figure 2(a), amplitude is the difference between maximum and minimum noise value. Frequency is reciprocal of the distance between the noise samples. The main ideas behind simulating the fog phenomenon are to adjust the atmospheric scattering coefficient of the transmission map in the atmospheric scattering model and add several noises with various frequencies and amplitudes together using Perlin's turbulence texture (see Figure 2(b)).

4. Rendering of Foggy Scenes

4.1. Algorithm Flowchart. Specifically, the proposed algorithm has three steps to render a foggy scene: the first one is generating Perlin noise image as the density distribution texture of heterogeneous fog. The second step is computing the transmission map with the Markov random field (MRF) model and the bilateral filter. The goal of this step is assigning the accurate pixel label using the graph-cut based α-expansion and removing the redundant details using the bilateral filter. Finally, with the generated Perlin noise image and transmission map, the foggy scene can be rendered according to the atmospheric scattering model. The flowchart of the proposed method is depicted in Figure 3.

4.2. Transmission Map Estimation. The estimation of transmission map is the most important step for foggy scene rendering and consists in image segmentation, initial map estimation based on MRF, and refined map estimation using bilateral filter.

4.2.1. Image Segmentation. The transmission map describes the portion of the light that is not scattered and reaches the camera. Since the map is a continuous function of depth, it thus reflects the depth information in scene. Koenderink [18] experimentally measured the human's ability to infer depth from an image, which shows that people cannot determine the relative depth of two points unless there is some visible and monotonic surface that connects them. Therefore, image segmentation technique is used here for estimating the transmission map and enhancing our knowledge of the image structure. The advantage of this technique is that it can often group large homogeneous regions of the image together while dividing heterogeneous regions into many smaller parts.

Mean-shift (MS) algorithm as a classical image segmentation technique is a robust feature-space analysis approach. It can significantly reduce the number of basic image entities, and due to the good discontinuity preserving filtering characteristic, the salient features of the overall image are retained. Besides, it is particularly important in the partitioning of images, in which only several distinct regions are used in representing different scenes such as sky, mountain, building,

(a) (b)

FIGURE 4: Image segmentation result. (a) Original images. (b) Its corresponding segmentation result.

lake, and animal, whereas other information within a region is often less important and can be neglected [19]. All these features are very useful for acquiring the relative depth information of scene objects. Thus, MS method is used as the first step to estimate the transmission map. Figure 4 shows an example of image segmentation result obtained by MS method.

4.2.2. Initial Map Estimation Using MRF. After obtaining the segmentation result, we use the graph-cut based α-expansion method to estimate the map $t(\mathbf{x})$, as it is able to handle regularization and optimization problem and has a good track record with vision-specific energy function [20]. Specifically, each element t_i of the transmission map is associated with a label x_i, where the set of labels $L = \{0, 1, 2, \ldots, l\}$ represents the transmission values $\{0, 1/l, 2/l, \ldots, 1\}$. Before labeling, we first convert input RGB image to gray-level image. Thus, the number of labels is 32 since the labeling unit of pixel value is set to be 8 and $l = 31$. The most probable labeling x^* minimizes the associated energy function:

$$E(x) = \sum_{i \in P} E_i(x_i) + \sum_{(i,j) \in N} E_{ij}(x_i, x_j), \tag{2}$$

where P is the set of pixels in the unknown transmission t and N is the set of pairs of pixels defined over the standard four-connect neighborhood. The unary function $E_i(x_i)$ is the data term representing the possibility of pixel i having transmission t_i associated with label x_i. The smooth term $E_{ij}(x_i, x_j)$ encodes the possibility where neighboring pixels should have similar depth.

For data function $E_i(x_i)$, which represents the possibility of pixel i having transmission t_i associated with label x_i, we first convert the input RGB image I_i to gray-level image I_i' and then compute the absolute differences between each pixel value and the label value. The process can be written as

$$E_i(x_i) = \left| I_i' \times \omega - L(x_i) \right|. \tag{3}$$

In (3), I_i' is the intensity of pixel in the gray-level image ($0 \le I_i' \le 1$). $L(x_i)$ is each element in the set of labels $L = \{0, 1/l, 2/l, \ldots, 1\}$. The parameter ω is introduced to ensure that I_i' and $L(x_i)$ have the same order of magnitude.

The smooth function $E_{ij}(x_i, x_j)$ encodes the possibility where neighboring pixels should have similar depth. Inspired by the work [21], we use the linear cost function, which is solved by α-expansion:

$$E_{ij}(x_i, x_j) = g \left| x_i - x_j \right|. \tag{4}$$

According to our daily experience, we know that objects which appear closer to the top of the image are usually further away. Thus, if we consider two pixels i and j, where j is directly above i, we have $d_j > d_i$. Thus, we can deduce that the transmission t_j of pixel j must be less than or equal to the transmission t_i of pixel i, that is, $x_j \le x_i$. For any pair of labels which violate this trend, a cost $s > 0$ can be assigned to punish this pattern. Thus, the smooth function in (4) can be written as

$$E_{ij}(x_i, x_j) = \begin{cases} s & \text{if } x_i < x_j, \\ g \left| x_i - x_j \right| & \text{otherwise.} \end{cases} \tag{5}$$

The parameters g and s are used to control the aspect of rendering effect. The value of g controls the strength of the image details display and is usually set 0.01. The cost s controls the strength of the image color display and is usually set 100. Besides, the weights associated with the graph edge should be determined. If the subtraction of the two neighboring pixel values in the input image is less than 15, then the two pixels may have high possibility of having the same transmission value. Thus, the cost of the labeling is increased by 15x to minimize the artifacts due to the depth discontinuities in this case. Taking the data function and the smooth function into the energy function equation (2), the pixel label of transmission map can be estimated by using the graph-cut based α-expansion. In our method, the gco-v3.0 library [22] developed by O. Veksler and A. Delong is adopted for optimizing multilabel energies via the α-expansion. It supports energies with any combination of unary, pairwise, and label cost terms [20, 23]. Thus, we use the library to estimate each pixel label in initial transmission map. The pseudocode of the estimation process using the gco-v3.0 library is presented in Pseudocode 1.

In Pseudocode 1, M and N are the height and width of the input image. By using the functions (e.g., GCO_SetDataCost, GCO_SetSmoothCost, and GCO_GetLabeling) defined in the optimization library, we can obtain each pixel label x_i. Then, a proper intensity value of the initial transmission map can

(a) (b)

Figure 5: Synthetic example. (a) Input gray-level image. (b) Output multilabel image.

```
Input: Input image I.    Output: Each pixel label x_i in initial transmission map
  Step 1. // Create new object
    Set NumSites = M × N and NumLabels = 32; h = GCO_Create(NumSites, NumLabels);
  Step 2. // Compute data term
    (2.1) Convert color image I into a gray image and express it in one-dimensional array Data;
    (2.2) for i = NumLabels
              for j = NumSites
                  DataTerm(i, j) = |Data(j) * ω − ((i)/NumLables) * 255|;
              end
          end
    (2.3) GCO_SetDataCost(h, DataTerm);
  Step 3. // Compute smooth term
    (3.1) for i = NumLabels
              for j = NumLabels
                  if i ≥ j
                      SmoothTerm(i, j) = g * |i − j|;
                  else
                      SmoothTerm(i, j) = s;
                  end
              end
          end
    (3.2) GCO_SetSmoothCost(h, SmoothTerm);
  Step 4. // Compute the cost of the labeling
    (4.1) for i = NumSites − 1
              if |Data(i + 1) − Data(i)| < 15
                  NeighborCost(i, i + 1) = 15;
              end
          end
    (4.2) GCO_SetNeighborCost(h, NeighborCost);
  Step 5. // Compute optimal labeling via α-expansion
    (5.1) GCO_Expansion(h);
    (5.2) Label = GCO_GetLabeling(h);
    (5.2) Covert Label which is a one-dimensional array into a M × N array, that is the output pixel label x_i.
```

Pseudocode 1: The pseudocode of the label assignment using gco-v3.0 library.

be assigned to each image pixel. Specifically, for each label x_i, we have

$$t_{\text{ini}}(\mathbf{x}) = 255 - (x_i - 1) \times 8. \tag{6}$$

In Figure 5, we show a synthetic example where the image consists of 5 gray-level regions. The image can be accurately segmented to be 5 label regions by using the proposed MRF method, and the 5 labels are represented by 5 intensity values, whose results are shown in Figure 5.

The MRF-based algorithm can also be applied for estimating the initial transmission map for real world images. An illustrative example is shown in Figure 6. In the figure, Figure 6(b) shows the initial transmission map estimated using the algorithm presented above; its corresponding rendered result is shown in Figure 6(c). One can clearly see that the fog-like feeling given by the simulated foggy image is not very strong due to the redundant image details.

4.2.3. Refined Map Estimation Using Bilateral Filter. As shown in Figure 6, there is obvious deficiency in the rendered image in the discontinuities of the transmission map obtained by MRF model. For example, the trees or mountains

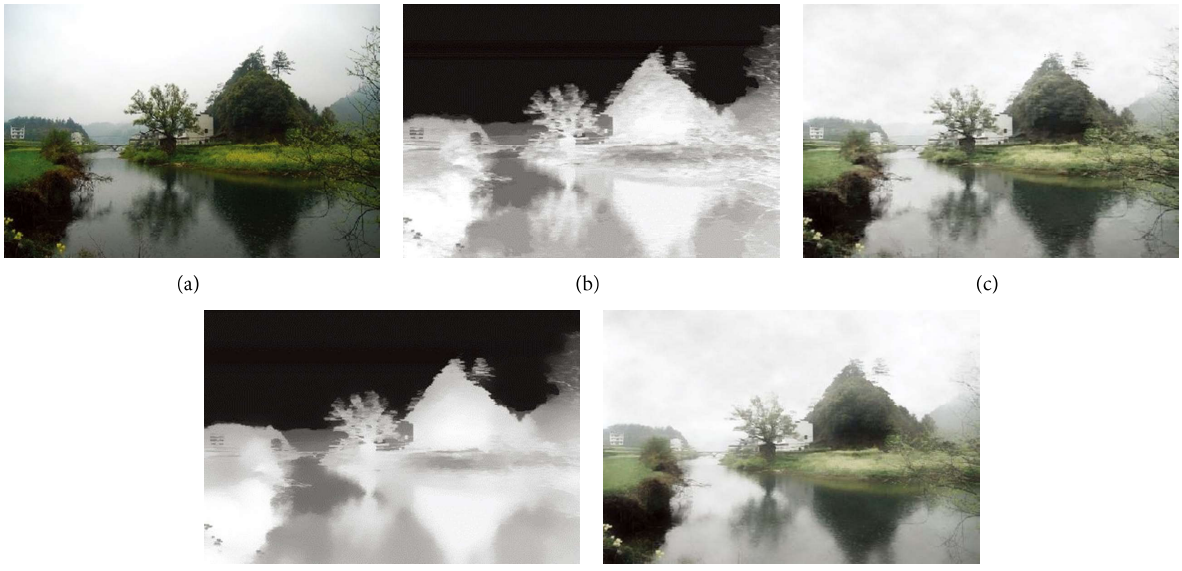


FIGURE 6: Example of real world image. (a) Input image. (b) Initial transmission map. (c) Rendered result obtained using (b). (d) Bilateral filter to (b). (e) Rendered result obtained using (d).

should have the same depth values. However, as shown in Figure 6(b), one can clearly see different intensity layers of trees or mountains in the transmission map estimated by the MRF-based algorithm. In order to handle these discontinuities, we apply the bilateral filter to our algorithm since the filter can smooth images while preserving edges [24]. Thus, the redundant details of the transmission map t_{ini} estimated by the algorithm presented above can be effectively removed, which improves the rendered result with better fog-like feelings. This process can be written as

$$t(\mathbf{u}) = \frac{\sum_{\mathbf{p}\in N(\mathbf{u})} W_c(\|\mathbf{p}-\mathbf{u}\|)\, W_s\left(\left|t_{\text{ini}}(\mathbf{u}) - t_{\text{ini}}(\mathbf{p})\right|\right) t_{\text{ini}}(\mathbf{p})}{\sum_{\mathbf{p}\in N(\mathbf{u})} W_c(\|\mathbf{p}-\mathbf{u}\|)\, W_s\left(\left|t_{\text{ini}}(\mathbf{u}) - t_{\text{ini}}(\mathbf{p})\right|\right)}, \tag{7}$$

where $t_{\text{ini}}(\mathbf{u})$ is the initial transmission map corresponding to the pixel $\mathbf{u} = (x, y)$ and $N(\mathbf{u})$ is the neighbor of $\mathbf{u}$. The spatial domain similarity function $W_c(x)$ is a Gaussian filter with the standard deviation σ_c : $W_c(x) = e^{-x^2/2\sigma_c^2}$, and the intensity similarity function $W_s(x)$ is a Gaussian filter with the standard deviation σ_s; it can be defined as $W_s(x) = e^{-x^2/2\sigma_s^2}$. In our experiments, the value of σ_c and σ_s is set as 3 and 0.4, respectively. Thus, we can obtain the final refined transmission map, as shown in Figures 6(d) and 6(e), which is the rendered result obtained using the refined map. From Figure 6(e), one can see that the rendered result obtained using the bilateral filter has better fog-like feelings compared with the result (see Figure 6(c)) obtained without using the filter. However, the refined transmission map is in fact a pseudo depth map instead of a recovery of real depth information. It reflects the relative positions between scene objects and their neighboring regions. In our experiment we find that the pseudo depth map can produce realistic foggy scenes without creating significant errors in the rendered image.

4.3. Fog Scene Synthesis. Since now we already know the input no-fog image $I(\mathbf{x})$, and the final refined transmission map $t(\mathbf{x})$, and the airlight A is set to be 255, we can obtain the simulated foggy image $J(\mathbf{x})$ according to the atmospheric scattering model (see (1)). The simulation result $J(\mathbf{x})$ is calculated by

$$J(\mathbf{x}) = \frac{I(\mathbf{x}) - A}{\max\left(t(\mathbf{x})^{\lambda}, t_0\right)} + A. \tag{8}$$

In (8), the value of t_0 is set to be 0.1, and the transmission map $t(\mathbf{x})$ combines the geometric distance $d(\mathbf{x})$ and the medium extinction coefficient β (the net loss from scattering and absorption) into a single variable [25]:

$$t(\mathbf{x}) = e^{-\beta d(\mathbf{x})}. \tag{9}$$

Experimental results show that the simulated foggy scenes with different fog density can be created by adjusting the coefficient β with the parameter λ in (8), that is, $t^{\lambda} = (e^{-\beta d})^{\lambda} = e^{-(\lambda\beta)d}$. However, using atmospheric scattering model on the input no-fog image, we can only generate rendered result with a homogenous fog added. However, fog is not always perfectly homogeneous in real situations. Therefore, Perlin noise is also introduced in our method to generate heterogeneous fog, which seems more natural to human visual perception. This process can be written as

$$R(\mathbf{x}) = J(\mathbf{x}) + k * n(\mathbf{x}). \tag{10}$$

In (10), J is the rendered result obtained by (8), n is Perlin's noise, and R is the simulated foggy scene with heterogeneous

Table 1: International visibility grades with their medium extinction coefficients.

Grade	Weather	Visibility distance	Extinction coefficient
0	Dense fog	<50 m	>78.2
1	Thick fog	50–200 m	78.2–19.6
2	Moderate fog	200–500 m	19.6–7.82
3	Light fog	500 m–1 km	7.82–3.91
4	Thin fog	1 km-2 km	3.91–1.96
5	Haze	2 km–4 km	1.960–0.954
6	Light haze	4 km–10 km	0.954–0.391
7	Clear	10 km–20 km	0.391–0.196
8	Very clear	20 km–50 km	0.196–0.078
9	Extremely clear	>50 km	0.0141

fog. The parameter k is used to control the appearance of Perlin's turbulence texture in our rendered result. We fixed it to 0.15 for all the results reported in this paper. An illustrative example is shown in Figure 7. In this figure, Figure 7(a) is the input no-fog image, Figure 7(b) is the transmission map estimated by the proposed method, Figure 7(c) is the simulated foggy image with uniform fog, and Figure 7(d) is the rendered heterogeneous fog result. One can clearly see that the simulated result in Figure 7(d) is more natural.

5. Experimental Results

Our proposed rendering method can work well for a wide variety of game scenes and real world images. In the experiments, we performed the algorithm by executing Matlab on a PC with 3.00 GHz Intel Pentium Dual-Core Processor.

5.1. Parameter Adjustment for Different Foggy Conditions. Since the parameter λ used to change the value of medium extinction coefficient β controls the fog density of the simulated scene, we can thus obtain our ideal rendered foggy image by tuning this parameter. Figure 8 shows some fog simulation results with different λ. One can clearly see that the larger value of λ is, the denser fog will be, as shown in Figure 8.

Therefore, by changing the value of β using the fog's density factor λ, it is possible to control the visibility distance or the global field of view, which decreases when λ rises, and vice versa. Table 1 gives the ten levels of the visibility grades and their corresponding extinction coefficient values widely used in meteorology [26].

From Table 1, one can clearly see that different foggy weather conditions are related to different visibility distances and extinction coefficients. Thus, we can deduce that the value of parameter λ can be selected according to the extinction coefficient value of the ten visibility grades to produce different foggy effects. To verify the effectiveness of the proposed method, a virtual scene in clear weather (see Figure 9(a)) is used as a benchmark image, which contains some targets at different distances, such as 33 m, 80 m, and 200 m. As can be seen in Figure 9, these targets gradually become blur as the value of parameter λ increases from 0.4 to 20. Since different visibility distances correspond to different foggy weather according to the international visibility grades shown in Table 1, we can thus simulate various fog effects by selecting proper parameter values. Table 2 gives the parameter values and the corresponding weather conditions for Figure 9. Note that the intrinsic and extrinsic parameters of the camera that creates the virtual scene are unknown for the transmission map generation, so the distance value of the targets shown in Figure 9 only reflects the relative relationship of the visibility. Thus, the weather type of the simulated scene is mainly determined by our subjective judgment.

To further validate the proposed technique for foggy weather simulation, we apply it to several synthetic images. An illustrative example is shown in Figure 10, and its corresponding parameter value and weather condition are given in Table 3. One can clearly see that the simulation results are consistent with human visual perception by setting the λ value according to the extinction coefficient value in visibility grades. This confirms our observations in Figures 9 and 10.

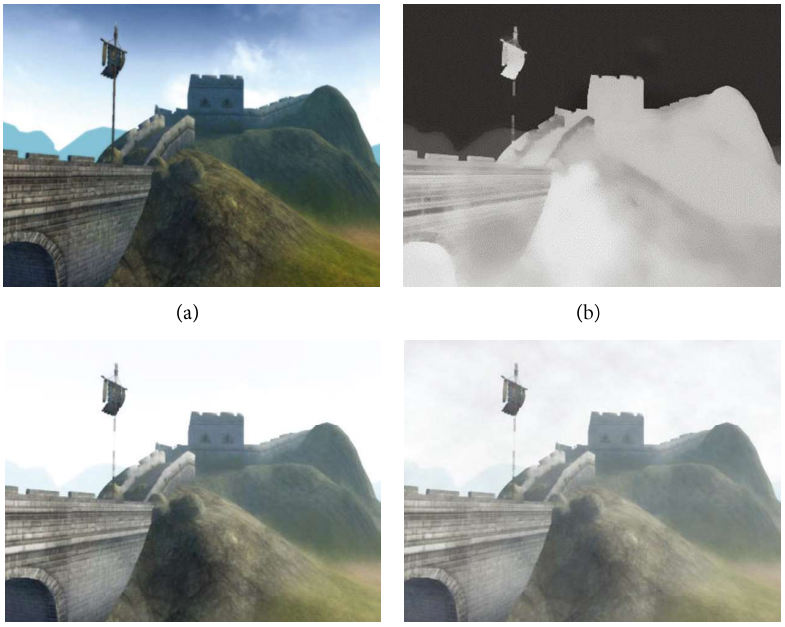
5.2. Visual Quality Comparison

5.2.1. Image Test Results. The algorithm proposed in this paper works well for a wide variety of synthetic images and real captured images. Figure 11 shows some examples of the rendering effects for game scene images. The parameter values for the simulated haze or thin fog scene are $\lambda = 0.8$ and $\lambda = 2.0$, respectively. The transmission maps estimated from the input original images are shown in Figure 11(b). Note that the maps do not recover the real depth information, but the generated foggy scenes still consist with our daily life experience. From the figures, we can see that the generated foggy scenes (see Figure 11(c)) look natural due to the differentiation of the fog's density.

We also apply our method to real world images. The results for real captured image are shown in Figure 12. The heterogeneous fog effect for the scenes is presented in the figure and the λ coefficient is, respectively, set to 1 and 0.5 to create the haze effect. From these results, we can get that our simulated results are realistic.

5.2.2. Qualitative Comparison. To verify the effectiveness of the proposed algorithm, we evaluate the visual quality of our rendered image by comparison with the results obtained by the commercial software and the state of the art rendering methods.

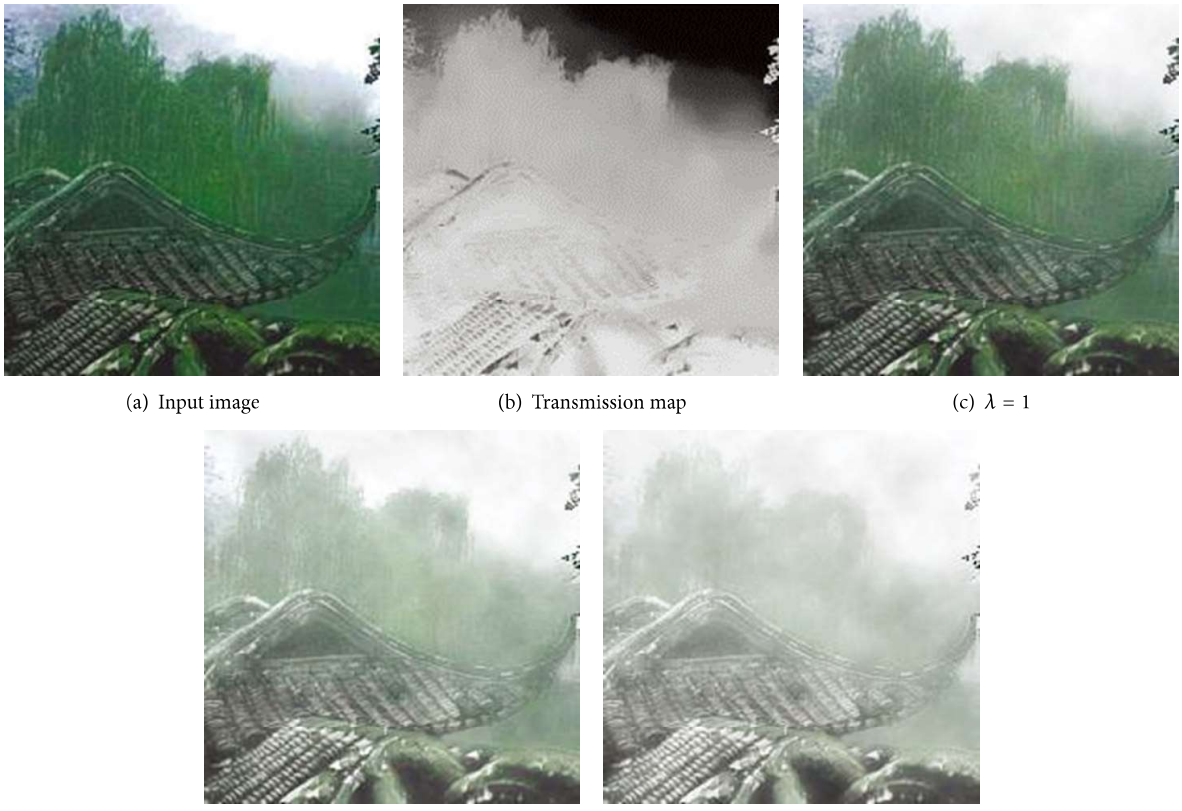
Figure 13 shows a comparison between our rendered foggy scene and real captured foggy image. It can be seen from the figure that there are some differences between our simulation results and real captured image, especially in the image details. Therefore, neither of the simulated foggy scenes (homogenous fog or heterogeneous fog) is visually close to the real foggy situation. That is because the reason for the change of real fog's shape, density, and color is very complicated and irregular. Thus, the fog density and color at each pixel of an image are hard to be described by some regular formulas. What we can do is adding more descriptions of fog detail transformation in the rendering



(a) (b)

(c) (d)

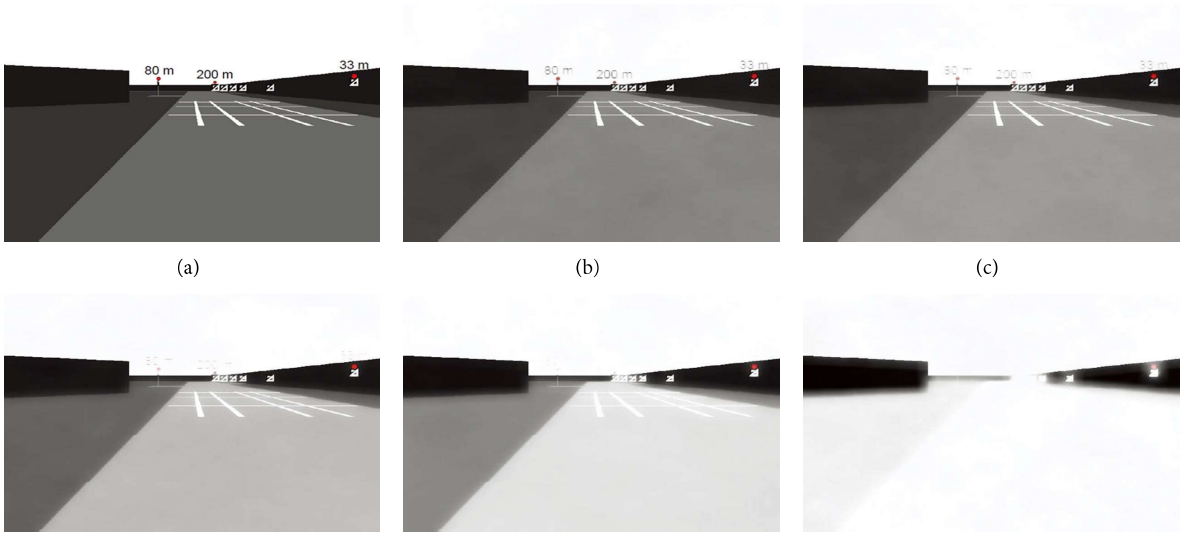
FIGURE 7: Example of fog scene synthesis. (a) Input no-fog image. (b) Our refined transmission map. (c) Generated foggy scene with homogeneous fog. (d) Generated foggy scene with heterogeneous fog.



(a) Input image (b) Transmission map (c) $\lambda = 1$

(d) $\lambda = 3$ (e) $\lambda = 5$

FIGURE 8: Simulated foggy images with different λ.



(a) (b) (c)

(d) (e) (f)

Figure 9: Fog simulation for different relative visibility distance.

(a) (b) (c)

(d) (e) (f)

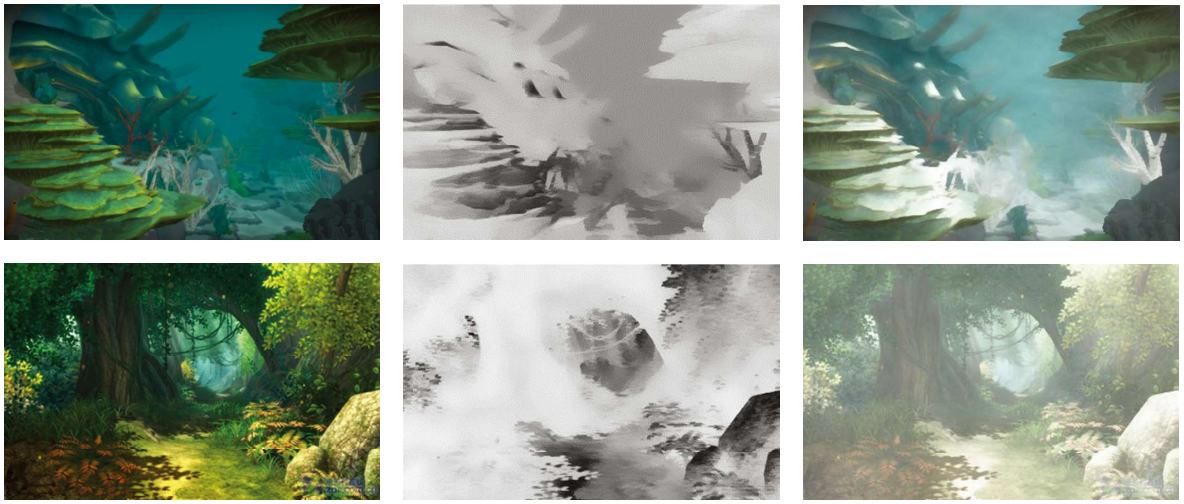
Figure 10: Different foggy weather simulations.

Table 2: The parameter values and the corresponding weather conditions for Figure 9.

Image	Figure 9(b)	Figure 9(c)	Figure 9(d)	Figure 9(e)	Figure 9(f)
Parameter λ	0.4	1.0	2.0	5.0	20.0
Weather condition	Light haze	Haze	Thin fog	Light fog	Thick fog

Table 3: The parameter values and corresponding weather conditions for Figure 10.

Image	Figure 10(b)	Figure 10(c)	Figure 10(d)	Figure 10(e)	Figure 10(f)
Parameter λ	0.6	1.8	4.0	15.0	25.0
Weather condition	Light haze	Haze	Light fog	Moderate fog	Thick fog



(a) Input image (b) Transmission map (c) Rendered results

FIGURE 11: Rendered results for game scenes.

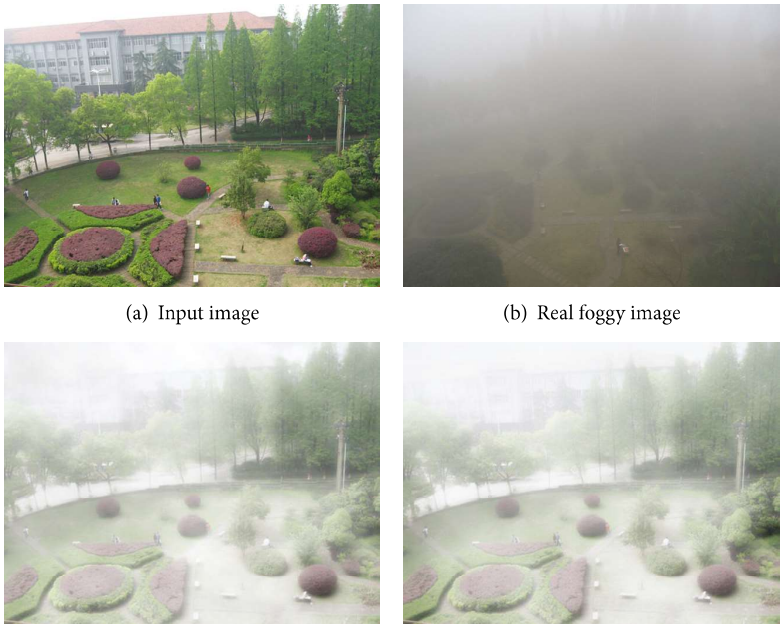
(a) Input image (b) Transmission map (c) Rendered results

FIGURE 12: Rendered results for real world images.

algorithm to provide viewers with more realistic sense of fog in the simulated scenes.

The comparison between our rendered foggy scenes and the 3ds max result is shown in Figure 14. One can clearly see that the fog effect created by 3ds max (see Figure 14(b)) looks good but not natural enough. That is because the fog density produced by the software is constant over the whole scene, which is not always true in real situations. A more natural way is to let scene objects gradually disappear in the distance fog; thus the atmospheric scattering model is used here for generating homogeneous fog effect. Thanks to the depth information provided by the estimated transmission map, the realism of simulated phenomenon is improved by the distribution of the fog, as shown in Figure 14(c). Compared with the homogenous foggy scene, much more realistic fog effect can be produced by using the Perlin noise and turbulence. As can be seen in Figure 14(d), the visibility is lower at the point being far away from the camera and the fog's density is differentiated by using the noise textures, just like in real world.

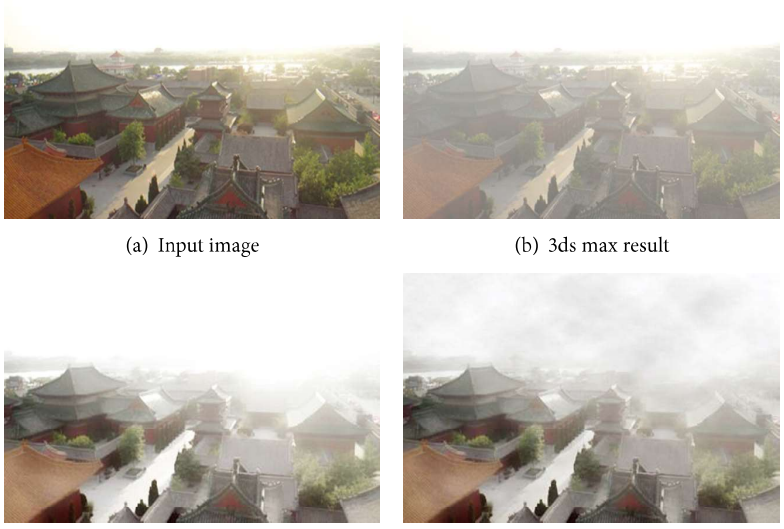
Figures 15 and 16 allow the comparison of our results with three state-of-the-art rendering methods: OpenGL, which simply blends the fog color with the object color based on the distance of the viewer; Sun's work [1], which is based on a single scattering light transport equation; Cao's work [2], which is based on an atmosphere point spread function (APSF). Notice that the object outline of OpenGL results seems still clear after rendering original scene with the built-in OpenGL functions, which is not always true for natural foggy image. Both Sun's work and Cao's work can produce



(a) Input image (b) Real foggy image

(c) Homogenous fog (d) Heterogeneous fog

FIGURE 13: Comparison between our rendered foggy scenes and real foggy image.



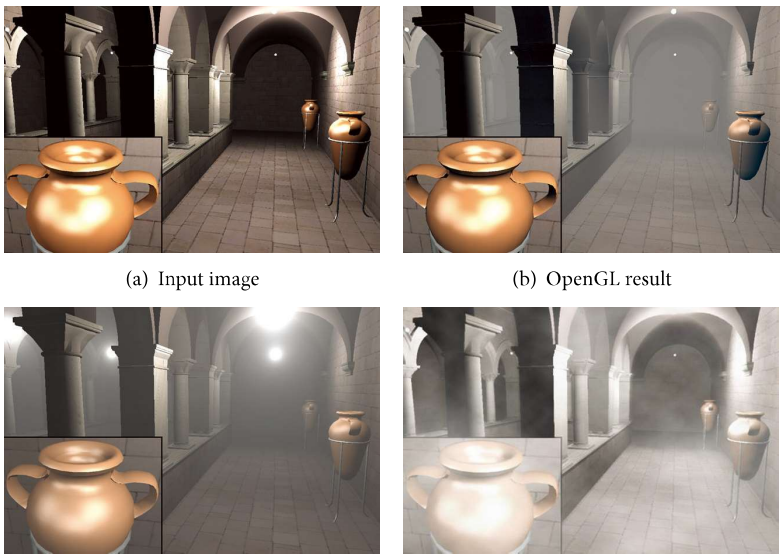
(a) Input image (b) 3ds max result

(c) Homogenous fog (d) Heterogeneous fog

FIGURE 14: Comparison between our rendered foggy scenes and 3ds max result.

impressive rendering results; especially the glows around light sources in Sun's result and the blurriness caused by contrast reduction in Cao's result are all visually compelling. However, many parameters are involved in these methods, which leads to more user interaction and makes the final simulation effects hard to control as well. Compared with these existing methods, the proposed algorithm can produce comparable result with much less parameter and user involvement.

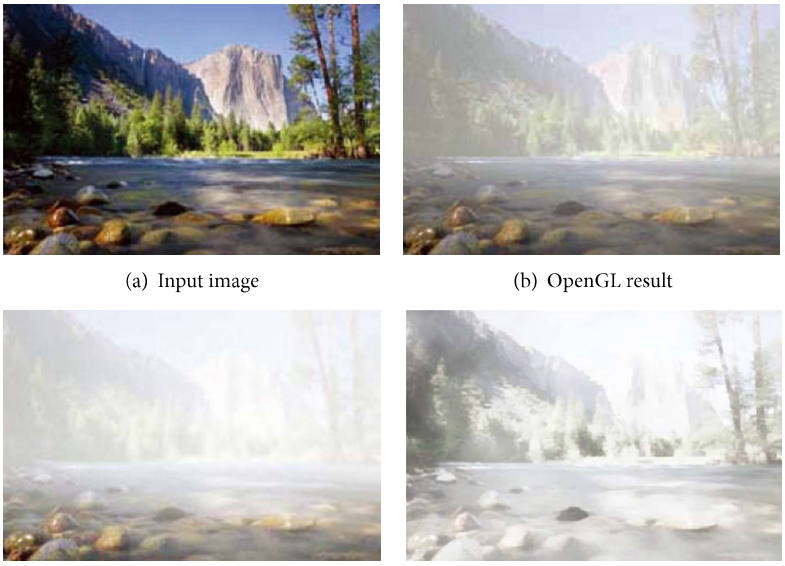
5.3. Computational Time. For computational efficiency, the complexity of the OpenGL is $O(s_x s_y)$ for an image of size $s_x \times s_y$, which implies the complexity of the OpenGL method is a linear function of the number of input image pixels due to the point-to-point operations of the built-in OpenGL functions. Thanks to programmable graphics hardware, Sun's method precomputes the numerical functions and stores them as 2D texture maps. Then, the entire analytic computation and table lookups are implemented in simple pixel or vertex



(a) Input image (b) OpenGL result

(c) Sun's result (d) Our result

Figure 15: Comparison with OpenGL result and Sun's work [1].



(a) Input image (b) OpenGL result

(c) Cao's result (d) Our result

Figure 16: Comparison with OpenGL result and Cao's work [2].

shaders in the graphics hardware [1]. Thus, the computing speed of Sun's method is very fast to achieve real-time rates. The computational time of Cao's method is much longer than Sun's method, since the APSF used in Cao's algorithm needs the convolution operation, so its computing speed is associated with the product of convolution mask size and input image size, whose time complexity is relatively high. Our proposed method has a relatively faster speed compared with Cao's method, and less time is needed to obtain a comparable rendered result of the same size in Matlab environment. Table 4 shows the computing speeds of the three methods with an image of size 640×480.

Table 4: Running time of different rendering methods for an image of size 640×480.

Method	Sun's method	Cao's method	Our method
Running time	0.05 s	130 s	18 s

6. Conclusions and Future Work

We propose a new algorithm to simulate foggy scenes for input no-fog images. Firstly, we generate Perlin noise image as the density distribution texture of heterogeneous fog. Then, we estimate the transmission map using the MRF model and

the bilateral filter. Finally, virtual foggy scene is realistically rendered with the generated Perlin noise image and transmission map according to the atmospheric scattering model. The proposed method provides a new way to simulate foggy scene with less algorithm parameter and user interaction compared with existing techniques. In the future, we intend to further accelerate the rendering speed by exploiting the many techniques of GPU.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (91220301), the China Postdoctoral Science Foundation (no. 2014M552154), the Hunan Planned Projects for Postdoctoral Research Funds (no. 2014RS4026), and the Postdoctoral Science Foundation of Central South University (no. 126648).

References

[1] B. Sun, R. Ramamoorthi, S. G. Narsinhan, and K. Nayar, "A practical analytic single scattering model for real time rendering," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1040–1049, 2005.

[2] Y. Cao, W. Shi, and S. Fang, "Simulation of high realism foggy image for natural scene," *Journal of System Simulation*, vol. 24, no. 6, pp. 1247–1253, 2012 (Chinese).

[3] N. Max, "Atmospheric illumination and shadows," *Computer Graphics*, vol. 20, no. 4, pp. 117–124, 1986.

[4] T. Yamamoto, Y. Dobashi, and T. Nishita, "Interactive rendering method for displaying shafts of light," in *Proceedings of the 8th Pacific Conference on Computer Graphics and Applications (PG '00)*, pp. 31–37, IEEE Press, January 2000.

[5] D. Jackèl and B. Walter, "Modeling and rendering of the atmosphere using Mie-scattering," *Computer Graphics Forum*, vol. 16, no. 4, pp. 201–210, 1997.

[6] T. Nishita, Y. Dobashi, and E. Nakamae, "Display of clouds taking into account multiple anisotropic scattering and sky light," in *Proceedings of the Computer Graphics Conference (SIGGRAPH '96)*, pp. 379–386, August 1996.

[7] C. B. Wang, Z. Y. Wang, and Q. S. Peng, "Real-time rendering of sky scene considering scattering and refraction," *Computer Animation and Virtual Worlds*, vol. 18, no. 4-5, pp. 539–548, 2007.

[8] H. Koschmieder, "Theorie der horizontalen sichtweite," *Beiträge zur Physik der freien Atmosphäre*, pp. 171–181, 1924.

[9] S. G. Narasimhan and S. K. Nayar, "Vision and the atmosphere," *International Journal of Computer Vision*, vol. 48, no. 3, pp. 233–254, 2002.

[10] D. Zdrojewska, "Real time rendering of heterogenous fog based on the graphics hardware acceleration," in *Proceedings of the Central European Seminar on Computer Graphics for Students*, 2004.

[11] J. Dong, K. Liu, and J. Wang, "Simulation of the foggy scene under outdoor natural scenes," *Journal of Computer-Aided Design and Computer Graphics*, vol. 25, no. 3, pp. 397–409, 2013 (Chinese).

[12] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 577–584, 2005.

[13] J. Park and C. Kim, "Extracting focused object from low depth-of-field image sequences," in *The International Society for Optical Engineering*, vol. 6077 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2006.

[14] Y. J. Jung, A. Baik, J. Kim, and D. Park, "A novel 2D-to-3D conversion technique based on relative height depth cue," in *Stereoscopic Displays and Applications XX*, vol. 7237 of *Proceedings of SPIE*, p. 8, The International Society for Optical Engineering, San Jose, Calif, USA, January 2009.

[15] K. Han and K. Hong, "Geometric and texture cue based depth-map estimation for 2D to 3D image conversion," in *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE '11)*, pp. 651–652, IEEE, January 2011.

[16] N. E. Yang, J. W. Lee, and R. H. Park, "Depth map generation from a single image using local depth hypothesis," in *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE '12)*, pp. 311–312, IEEE Press, January 2012.

[17] K. Perlin, "An image synthesizer," *Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[18] J. J. Koenderink, "Pictorial relief," *The Royal Society of London Philosophical Transactions Series A: Mathematical, Physical and Engineering Sciences*, vol. 356, no. 1740, pp. 1071–1086, 1998.

[19] W. B. Tao, H. Jin, and Y. M. Zhang, "Color image segmentation based on mean shift and normalized cuts," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 5, pp. 1382–1389, 2007.

[20] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[21] R. T. Tan, "Visibility in bad weather from a single image," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, IEEE Press, June 2008.

[22] "The gco-v3.0 library (gco-v3.0)," http://vision.csd.uwo.ca/code/.

[23] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," *International Journal of Computer Vision*, vol. 96, no. 1, pp. 1–27, 2012.

[24] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE 6th International Conference on Computer Vision*, pp. 839–846, IEEE Press, January 1998.

[25] M. C. W. van Rossum and T. M. Nieuwenhuizen, "Multiple scattering of classical waves: microscopy, mesoscopy, and diffusion," *Reviews of Modern Physics*, vol. 71, no. 1, pp. 313–371, 1999.

[26] C. H. Yang, *Handbook of Laser and Infrared Technology*, National Defense Industry Press, 1990, (Chinese).