Received: 20 March 2018 | Accepted: 21 March 2018
DOI: 10.1002/cav.1839

SPECIAL ISSUE PAPER

WILEY

Time-varying light motion in single convergence

Xingyu Lin[1] | Mingxuan Chai[1] | Sheng Li[1,2] | Guoping Wang[1,2]

[1]Graphics and Interaction Laboratory, Peking University, Beijing, China
[2]Beijing Engineering Technology Research Center of Virtual Simulation and Visualization, Peking University, Beijing, China

Correspondence
Sheng Li, Graphics and Interaction Laboratory, Peking University, Beijing, China; or Beijing Engineering Technology Research Center of Virtual Simulation and Visualization, Peking University, Beijing, China.
Email: lisheng@pku.edu.cn

Funding information
National Key Research and Development Program of China, Grant/Award Number: 2017YFB0203002 and 2017YFB1002705; The National Key Technology Research and Development Program of China, Grant/Award Number: 2015BAK01B06; National Natural Science Foundation (NSFC) of China, Grant/Award Number: 61421062 and 61472010

Abstract
As light travels fast in the physical world, it is generally hard to capture the propagation of light with a real camera. However, with the development of photon mapping, it is possible to simulate and visualize such fantastic effect. In this paper, we propose a new algorithm based on progressive photon mapping to render the time-varying light motion. By incorporating our algorithm with participating media, we synthesis the animation of time-varying light beams expanding in space with slow motion. Interesting phenomena can be observed in our experiments through designating either a constant light source or a bullet light that only emits instantaneously. Our algorithm is efficient in the sense that all frames of the animation can be rendered with only one convergence of the progressive photon mapping, showing the advantage of revealing the light propagation process by our algorithm. The experiments demonstrate the effectiveness and efficiency of our approach.

KEYWORDS
motion, participating media, progressive photon mapping, time-varying

1 | INTRODUCTION

Slow motion creates visual art. It is often used for artistic effect, to create a romantic or suspenseful aura or to stress a moment in life. To create the slow motion of lighting would help us see a phenomenon that usually cannot be captured by human eyes and can also help us visualize and analyze the propagation of light. The slow motion of a phenomenon is usually achieved by capturing the object's motion in a much higher rate and replaying it in normal speed. However, to capture light in slow motion in the physical world is not easy, as light travels at a very high speed (3×10^8 m/s) and may require a trillion frame per second (fps) camera to accomplish this task. Only until very recently, scientists are able to capture light with a half trillion fps camera and image synthesis analysis.[1] However, the field of computer graphics has long been rendering photorealistic images with the powerful framework of ray tracing.[2] Previously, other authors[3–5] proposed methods for rendering light shaft in a flash or of real-time rendering. With the recent development of photon mapping algorithms,[6,7] it becomes possible to simulate light traveling and illumination propagation in slow motion.

In this paper, we focus on simulating the magical slow motion lighting phenomenon based on progressive photon mapping (PPM). PPM is an iterative multipass algorithm, with a ray tracing pass to record all hit points for the rays shooting from camera, and multiple photon tracing passes to accumulate the photon energy onto the nearby hit points. To render the light in slow motion, a time tag is well designed when tracing each photon. During the gathering process, the energy of these photons will be accumulated to different frames of the animation based on their time tags associated.

Compared with the traditional frame-by-frame rendering, our algorithm can render multiple frames within a single convergence, with little extra time cost. In order to tackle the limitation caused by the capacity of Graphics Processing Unit (GPU) memory, we propose a simple way to do the trade-off between time and memory consumption. By incorporating the volumetric media and rendering with different types of light sources, we create interesting effects vividly that are not usually seen in the real world.

2 | RELATED WORK

Global illumination is generally the core element of photorealistic rendering. Photon mapping is effective for resolving global illumination with its advantageous rendering of caustic and color bleeding effect. It has been an active field over the past two decades.

Photon Mapping. Classical photon mapping was first presented by Jensen.[6] Basically, it is a two-pass global illumination algorithm that approximately solves the rendering equation. A photon tracing pass to generate photon maps is followed by a ray tracing pass, in which the illumination is reconstructed via the photon map. It is a consistent method, as any desired accuracy can be achieved by increasing the number of photons. However, achieving accuracy demands a huge amount of memory and computing resources.

PPM. PPM was proposed to handle the memory limitation problem of classical photon mapping[7] by multiple photon tracing passes. This method can effectively decrease noise with constrained memory usage. With enough photon passes, the convergence can be guaranteed. Much work has been done to improve the performance, either memory limitation or usage of computing resources.[8–11] These works focus on rendering a single image or frame with decent quality. Considering rendering an animation, such frame-by-frame approach will take a long time to converge. This prevents PPM from being applied to game and more efficiency-required animation.

Volumetric Scattering Rendering. Rendering participating media is important for a number of domains, ranging from commercial applications such as entertainment and virtual reality to simulation systems such as driving, flight, and space simulators. It is popular for the rendering of mediums such as fog,[12] cloud,[13] and ocean.[14] Volumetric photon mapping was first presented by Jensen et al.[15] and then improved by Jarosz et al.[16] to avoid redundant density queries due to ray marching. Jarosz et al. extended PPM to photon beams,[17] which is the basis of our algorithm, because it improves efficiency at the stage of photon gathering and fits well with a ray tracing framework. Zhang et al.[18] incorporated volumetric photon mapping into the precomputed radiance transfer pipeline. All of them contribute to promoting static volumetric rendering result rather than a sequence of animation.

Photon Mapping Animation. Few researchers on photon mapping specifically address animated scenes. Time-dependent photon mapping[19] is the very first work on rendering dynamic scenes, in which the authors explored the distribution of photon in motion sequence and then introduced a fast and concise method to describe and render the motion blur scene with photon mapping. Jonsson et al. recently presented a method for interactive global illumination of time-varying volumetric data based on photon mapping.[20] They explored the correlation between animation frames in order to optimize the performance of photon mapping motion rendering. Jarabo et. al proposed a novel density estimation technique that allows for reusing sampled paths to reconstruct time-resolved radiance and devised new sampling strategies that take into account the distribution of radiance along time in participating media.[21] A transient integral form of the radiative transfer equation was proposed for a new transient method that allows one to significantly mitigate variance.[22] Our work is inspired by these works and focus on the simulation of the transient motion of light.

3 | OUR APPROACH

In order to show the propagation of lighting through space clearly, we create scenes with light beams that travel through space. We implement global illumination by PPM, and the light beams are visualized by rendering with volumetric media that allows for the time-varying scattering of light.

3.1 | Progressive photon mapping

In PPM, the following information is recorded for each hit point of the ray and the region: the hit position x, the direction of the incoming ray ω, Bidirectional Reflectance Distribution Function (BRDF) identifier, surface normal n, current photon radius R, a count of photons N, and accumulated unnormalized flux τ. After the ray tracing pass, there will be several photon tracing passes before convergence. In each pass, a bounded amount of photons are emitted from the light sources and traced in the space, during which the flux of the photon will be accumulated to the hit point recorded in the ray tracing pass that is within its radius R. We show a summary of the structures of the information we stored for each hit point and photon in Table 1.

After each iteration, we update every hit point x. Assume we have N_i photons accumulated within the R_i radius of x after the ith iteration and M new photons in the (i +1)th iteration. The photon counts will be updated as

$$N_{i+1} = N_i + \alpha M,$$

where $\alpha \in (0, 1)$ is a constant. To ensure convergence to the correct solution, the radius for each hit point should converge to zero by updating it with

$$R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M}{N_i + M}}.$$

The total unnormalized flux of the M new photons τ_M will also be calculated, as follows:

$$\tau_M = \sum_{p=1}^{M} f_r(x, \omega, \omega_p) \Delta\Phi_p,$$

where $\Delta\Phi_p$ is the power of one photon, and $f_r(x, \phi, \omega_p)$ is the BRDF at location x in outgoing direction ω and incoming direction ω_p. By assuming that the power density is uniform around each hit point, we update the flux according to the radius each time, as follows:

$$\tau_{i+1} = (\tau_i + \tau_M) \frac{R_{i+1}^2}{R_i^2}.$$

Finally, we estimate the radiance as follows:

$$\begin{aligned} L(x, \omega) &\approx \frac{1}{\Delta A} \sum_{p=1}^{N} f_r(x, \omega, \omega_p) \Delta\Phi_p \\ &\approx \frac{\tau_i}{\pi R_i^2} \frac{1}{N_{\text{emitted},i}}, \end{aligned}$$

where $N_{\text{emitted},i}$ denotes the total number of photons shoot after the ith iteration. The above principle from PPM is the foundation of our approach.

TABLE 1 Structures for each hit point during the ray tracing pass and for each photon during the photon tracing pass

Hit point record	
position x	Hit location
normal n	Surface normal at x
ray direction w	Incoming direction
radius r	Photon collection radius
photon count n	Accumulated photon count
BRDF	Index of BRDF function
flux	Accumulated flux
volumetric radiance[]	Volumetric radiance- for each frame
Photon record	
position x	Photon location
normal n	Normal at x
ray direction w	Ray direction
energy	Photon energy
time	Duration after emission

3.2 | Volumetric media rendering

To show the time-varying slow motion light with beams, instead of assuming vacuum everywhere, we model the photons to be traveling in participating media, where the photons can interact with the media. There are two possible interactions with the participating media: absorption and outscattering.

Photons traveling through the medium may be absorbed by the particles in the medium. This is modeled by an absorption coefficient σ_a, which gives the probability that a photon is absorbed when it travels a unit distance inside the medium. The outscattering means that the photon may collide with small particles and the traveling direction changes. Similarly, the scattering coefficient σ_s of this interaction for one photon traveled in the medium per unit distance. In isotropic medium (which is the general case in our experiment), the direction of outscattering will be sampled equally. Together, we define an extinction coefficient $\sigma_t = \sigma_a + \sigma_s$. A photon that travels a small distance ds will have a probability $\sigma_t * ds$ of being absorbed or outscattered. The scatter albedo $\alpha = \sigma_s/\sigma_t$ gives the probability that an event is subject to an outscatter rather an absorption. Moreover, we define a transmittance function to evaluate the fraction of radiance remains when a photon travels a distance of s through the medium, as follows:

$$Tr(s) = e^{-\sigma_t s}.$$

In the case of photon mapping in vacuum space, the photons will only stop on the surface of some objects. Those photons can be stored in a KD-Tree and collected easily, whereas the volumetric photons here can stay anywhere inside the participating medium. Thus, we need an efficient way to collect photons in a 3D volume.

Ray marching tries to solve this by marching the ray step by step and collect all the volumetric photons along the way. However, ray marching is costly, especially on GPU. We adopt a method known as beam radiance estimation, proposed by Jarosz et al.[16,23] It extends a ray with a radius and turns it into a beam. By intersecting the beam with volumetric photons, the algorithm avoids the inefficient ray marching and can be easily incorporated into the ray tracing framework that has already been properly accelerated. The beam radiance from position x with an outshooting direction ω is estimated by

$$L(x, \omega) = \frac{1}{\pi r^2} \sum_{i \in B} \Phi_i e^{-\sigma_t t_i},$$

where the sum is over all photons that intersect with the beam B, and t_i is the projected distance from x to the photon along the ray and equals to $< \omega, x - p_i >$, where p_i is the position of the photon.

3.3 | Time-varying rendering

In order to reveal the procedure of light in slow motion, we need to simulate a virtual rendering speed that is comparable with the actual speed of light (299,792,458 m/s), where fantastic spectacle may occur. We made two important assumptions: (a) The speed of light is constant. While light speed actually varies according to the properties of different mediums or colors, which could make an uncertain difference under a tiny timescale, we decide to simplify the simulation for our first step. (b) During our animation, while it takes a few frames for the photons to travel from one position to another, these photons can be seen by the camera immediately in the same frame (i.e., we neglect the distance from the location of photons to the camera). Considering the time it takes for a photon to travel to the camera would be another interesting issue.

In the standard PPM algorithm, the radiance for each location x is estimated by $L(x, \omega)$, where time is not involved. Now, that the radiance changes in the same location x should depend on time, we estimate the time-varying radiance in terms of the time variable, as follows:

$$\begin{aligned} L(x, \omega, t) &= \frac{1}{\Delta A} \sum_{p \in N(t)} f_r(x, \omega, \omega_p) \Delta \Phi_p \\ &= \frac{\tau_i(t)}{\pi R_i^2} \frac{1}{N_{\text{emitted},i}(t)}, \end{aligned}$$

where $N(t)$ is the set of photons that have reached to the radius R_i sphere of position x for photon collection, by time t. A straightforward way to extend PPM for our time-based radiance estimation would be to estimate $L(x, \omega, t)$ each time with a different set of $N(t)$. However, this method has two drawbacks: (a) It is slow. The rendering time is linear to the number of time $N(t)$ changes, which equals to the number of frames in our animation. (b) It may cause jittering if $L(x, \omega, t)$ of different

t do not converge to the same point, due to the different starting state of photon tracing sampling. Thus, we propose an incremental strategy that can render all $L(x, \omega, t)$ in a single convergence. The philosophy lies in our observation of the progressive relationship of the radiance estimation, with respect to variable time, as follows:

$$\begin{aligned} L(x, \omega, t + \Delta t) &= \frac{1}{\Delta A} \sum_{p \in N(t+\Delta t)} f_r(x, \omega, \omega_p) \Delta \Phi_p \\ &= \frac{1}{\Delta A} \sum_{p \in N(t) \cup N(\Delta t)} f_r(x, \omega, \omega_p) \Delta \Phi_p \\ &= \frac{1}{\Delta A} \left(\sum_{p \in N(t)} f_r(x, \omega, \omega_p) \Delta \Phi_p + \sum_{p \in N(\Delta t)} f_r(x, \omega, \omega_p) \Delta \Phi_p \right). \end{aligned}$$

Each time when the time increases, instead of estimating the $L(x, \omega, t)$ for each position x from scratch, only the additional photons that arrive during the time period Δt after t are required to estimate. In practice, the radiance of each photon can be directly added into a time buffer that sums up to the final radiance $L(x, \omega, t)$. We create a time buffer and then map a photon to its corresponding position in the buffer, and finally, we accumulate the time buffer value to our radiance estimation. For each photon emitted from the light source, we record the time it has traveled since it left the light source. Given that the light source emitted photons constantly, at time t, we only need to consider those photons that have traveled a distance less than $t * c$, where c denotes the speed of light. For each photon p, the corresponding position in the time buffer where it should store can be calculated by

$$buffer_p = \left\lfloor \frac{t_p * total\_frame}{T} \right\rfloor,$$

where T and $total\_frame$ denote the total time span of the animation and the number of time intervals that we divide the time into, respectively. In this way, we add only one more segment for each photon record and collect the photons for each pixel for all these time intervals at the end. In our experiments, the algorithm can render the light motion in almost the same time as rendering one frame. Moreover, the memory expense does not increase much. The memory cost for photon mapping in a flash mainly consists of two parts: the space for storing all the photons at each pass V_p, and the space for storing each hit point obtained by ray tracing V_t. By analysis, when rendering an animation of light motion with K frames, our algorithm only takes $O(V_p + K * V_t)$ space. For most of the case, V_p will be the main cost, as there can be tens of millions of photons during each pass. On the other hand, V_t is usually determined by the size of the screen and is much smaller.

3.4 | Implementation details

During our implementation of the illumination, the final color of each pixel on the image consists of four parts: direct flux, indirect flux, volumetric radiance, and ambient light. The indirect flux is estimated by collecting the neighboring photon energy using the equation above, except those photons that hit the surface only once. That energy is calculated directly in the direct flux. In this way, we save memory and time for storing these photons, and the volumetric radiance is collected by tracing a special ray that only intersects with these volumetric spheres and collects the energy along the ray.

Another aspect of the implementation is about how to attach a time tag on each process of the radiance estimation. For the volumetric radiance and the indirect flux processing, we can trace the distance each volumetric photon has already traveled in the space. Under the assumption of constant light speed, we can get the accurate time for light traveling. For the part of direct flux, however, as we may not have any traced photons around a hit point, its time tag is directly determined by its projection distance to the light source. The existence of the ambient light is barely to give an overview of the scene, and it will stay unchanged across different frames.

4 | EXPERIMENTS

We implement our algorithm based on the ray tracing framework OptiX 3.9 developed by NVIDIA. All our tests are done on an Intel Core i7 5820K (3.3 GHz) with an NVIDIA GeForce GTX980 Ti GPU at a 768 × 768 resolution with 32 GB memory capacity. We use two scenes for our tests: a church scene (Sibenik) and the Cornell box scene. For the church scene, we set up area lights just outside the church windows, with a 45° tilt, so as to light up the scene and create the light beams through the windows as well. During the ray tracing pass, we sample rays from the camera and maintain all the hit points with a KD-Tree. During each photon tracing pass, we shoot two million photons sampled from a semisphere space and trace them up to three hops. When the photon hits a surface, its flux is accumulated to the nearest hit point, within the radius of 2.0. We use the same radius for the volumetric sphere photons. To render the volumetric effect, we enclose our scene in a large box of participating media, with the absorption and the scattering coefficient being 0.000001 and 0.05, respectively. We assume the medium to be isotropic, which means that the scattering direction will be sampled evenly on a sphere.

4.1 | Rendering results

We render the slow light effect in two ways: (a) The light source constantly emits lights, and (b) the light source only emits for a small period of time. The comparison of rendering using different ways is shown in Figure 1. In the first case, all radiance accumulated before time t_i is added together and displayed in the ith frame. In the second case, for each frame i,

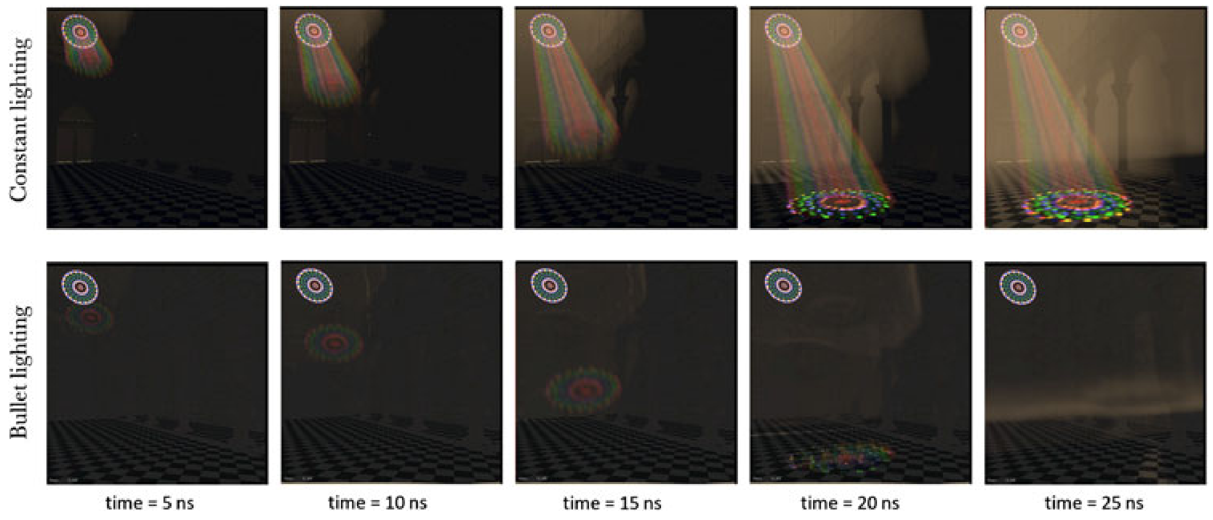

FIGURE 1 We highlight the time-varying slow motion in two styles, under the church scene (Sibenik). For each row from left to right is the listed sequence of the animation according to their timestamps. On the upper row, the light source constantly emits light, whereas on the lower row, the light source only emits light for a very short period of time

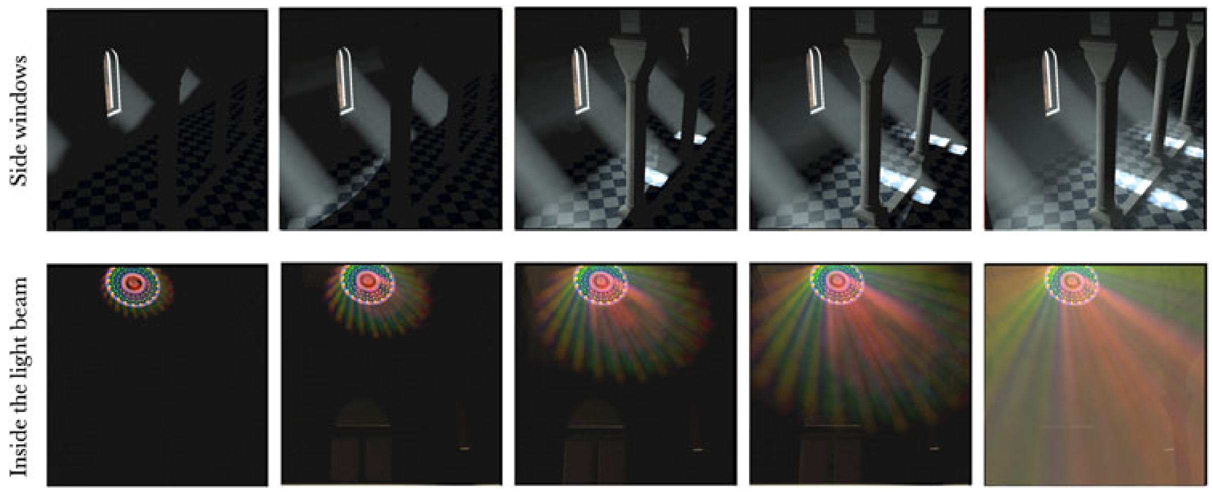

FIGURE 2 Time-varying rendering of the Sibenik scene. For each row from left to right is the listed, different sequence of the animation according to their time. The upper row exhibits the illumination from the side windows of the church, with white constant light source coming from outside. The lower row demonstrates the viewpoint located inside the beam light

only radiance that is collected during a small interval of time around t_i is displayed in the ith frame. In our visualization, we can see the light beams extend gradually in the volume as the light travels through and finally hit the ground with brilliant light spots. On the other hand, if the light emits for a short time, only a thin layer of photons is emitted and can hardly light up the church scene, while we can still see the layer traveling in space. For the church scene, we show more results of different viewpoints, from inside the light beam and from the side wall in Figure 2.

We render another Cornell box scene without the participating medium. Similar to the church scene, we render with one constant light source and one bullet light source. The results are shown in Figures 3 and 4. In Figure 3, we can see that the scene is firstly illuminated by direct lighting, because this is the fastest way. After that, the ceiling is then illuminated through the first bounce of indirect light, followed by the second bounce of indirect light, and thus, it becomes brighter. The process of color bleeding can also be seen clearly, as only in the last image can we see the bleeding effect. In Figure 4, however, we can only see a layer of thin edges moving along the surfaces, much like how colorful ink would blend in the

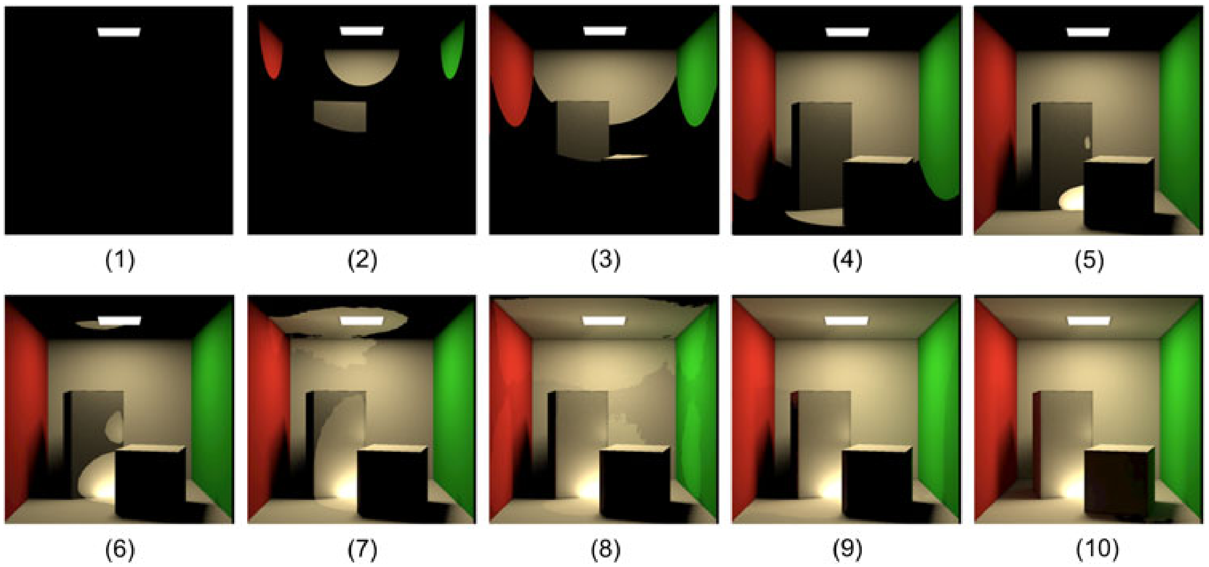

FIGURE 3 Time-varying illumination of the the Cornell box scene, with a constant light source. We highlight the image sequences to show the variation of illumination along with time variation

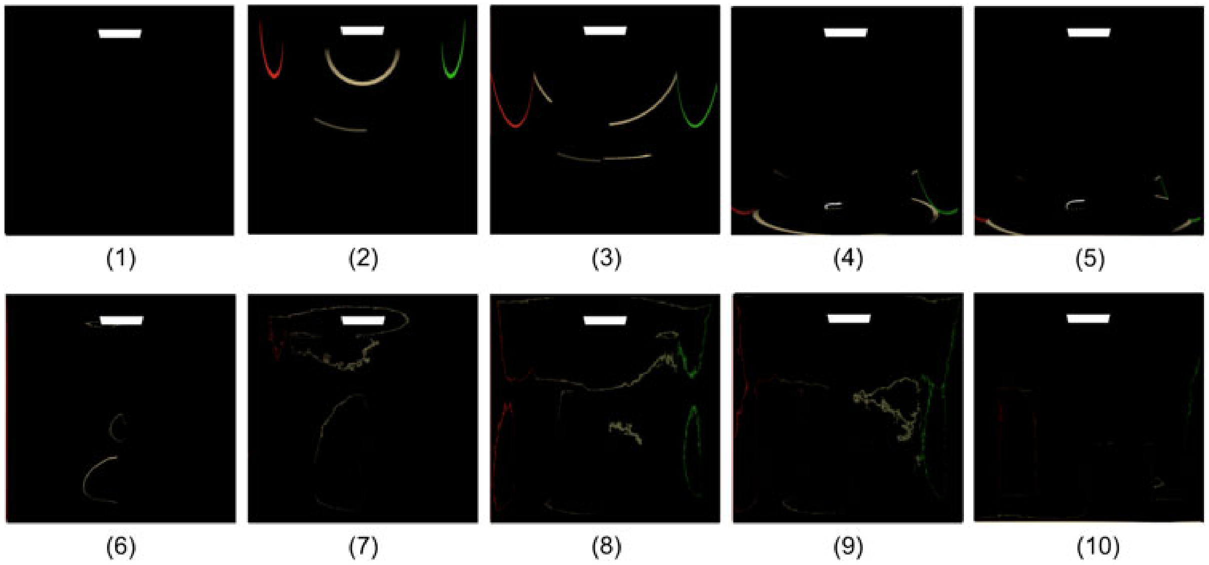

FIGURE 4 Time-varying illumination of the Cornell box scene, with a bullet light source. The image sequences show the variation of illumination along with the time variation

water. As a few photons can only contain limited energy, the bullet light source scene looks much dimmer than the same scene rendered with a constant emission light.

We also show more high-resolution animations of Sibenik in Figure 5 and scenes without stained glass in Figure 6, respectively, to present a more bright atmosphere using different parameters. These demonstrate the effectiveness and diversity of our approach.

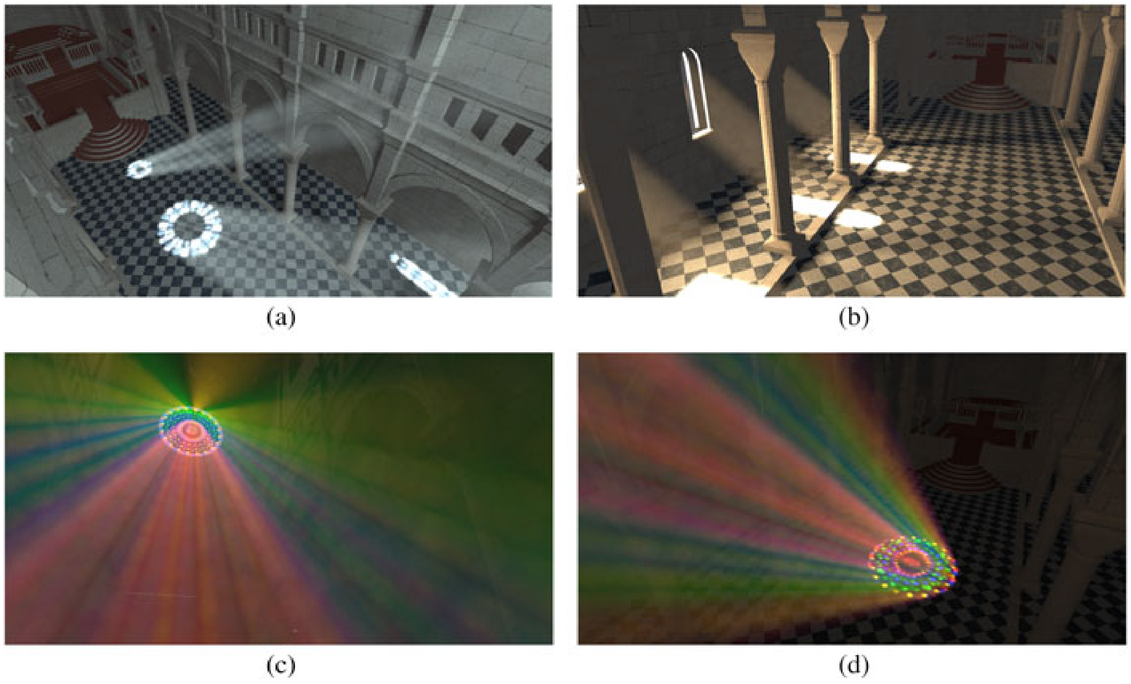

FIGURE 5 Time-varying light beams animation, under different lighting conditions with different views. (a) Morning. (b) Afternoon. (c) Inside of the light beam. (d) Outside of the light beam

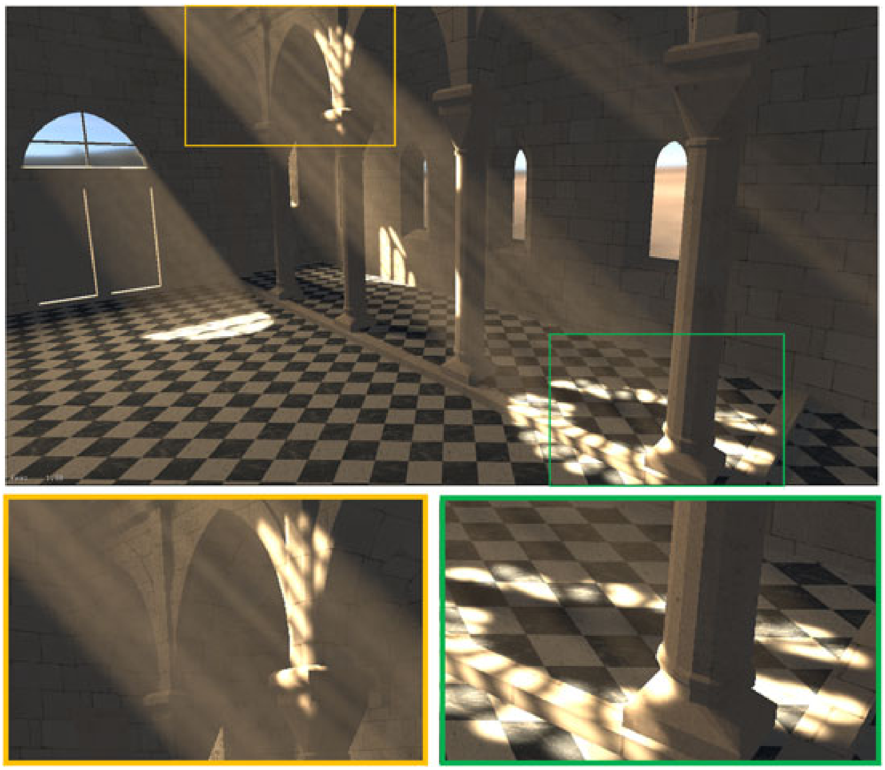

FIGURE 6 We highlight the rendering effects using the scenes without stained glass, with close-up view

TABLE 2 Performance comparison on simulating the same animation with different methods

Frame Number	Ours (s)	Frame by frame (s)
1	79.11	**74.11**
4	**78.06**	297.52
8	**81.92**	698.04
16	**83.59**	1, 193.08
32	**88.19**	2, 383.16
48	**98.75**	3, 571.24

Note. The left column shows the number of frames in an animation. The middle and the right columns show the time cost of each method for the animation, in seconds.

4.2 | Memory/time trade-off

To test the efficiency of our method, we compare our method with the frame-by-frame algorithm. We render animations with same number of frames, for both methods. The performances of both methods are shown in Table 2. We can see that, when only one frame is required to render, our algorithm is slightly slower than the frame-by-frame algorithm, due to the maintenance of all the extra information about the hit point being recorded into the photon structure. However, as the frame numbers increase, the time consumption for the frame-by-frame algorithm increases linearly, which is intolerable when the number of frame becomes larger and larger to some extent. On the other hand, our algorithm only needs to gather the extra time information induced by the new frames and thus only requires a few more seconds and is considerably more efficient than the frame-by-frame algorithm.

In an animation clip as shown in the Supporting Information, at least 196 frames are used to create a smooth transition of the light motion. However, the limitation of the GPU memory prevents us from buffering all the frames at once. As such, by dividing all the frames into K sections, we can save the usage of the GPU memory by a scale factor of K, and at the same time, we need K convergences of the PPM algorithm. In our experiment, $K = 4$ is adopted for most of the cases.

5 | CONCLUSION

In this work, we show that by extending the PPM algorithm with a time tag, magical slow light motion can be captured and visualized. By incorporating scattering medium in the scenes, interesting lighting effect can be synthesized. We show that our algorithm is efficient in the sense that different frames of the animation are rendered within single convergence of the PPM. Our future work will be focused on developing a slow light animation system, in which complex light motions under more natural scene containing specular objects and transparent objects with various material will be simulated. Our method could be used as a tool and be helpful for analyzing and for the design of the magic light beams.

ACKNOWLEDGEMENTS

We acknowledge the anonymous reviewers for their comments and suggestions. This work is supported by National Key Research and Development Program of China under Grants 2017YFB0203002 and 2017YFB1002705, by the National Key Technology Research and Development Program of China under Grant 2015BAK01B06, and by National Natural Science Foundation (NSFC) of China under Grants 61421062 and 61472010.

ORCID

Sheng Li http://orcid.org/0000-0002-8901-2184

REFERENCES

1. Velten A, Raskar R, Bawendi M. Picosecond camera for time-of-flight imaging. Imaging and Applied Optics; 2011; Toronto, Canada. Washington, DC: Optical Society of America; 2011; p. IMB4.
2. Whitted T. An improved illumination model for shaded display. Paper presented at: SIGGRAPH '05 ACM SIGGRAPH 2005 Courses; 2005; Los Angeles, CA. New York, NY:ACM; 2005. p. 4.

3. Dobashi Y, Yamamoto T, Nishita T. Interactive rendering method for displaying shafts of light. Proceedings the Eighth Pacific Conference on Computer Graphics and Applications; 2000; Hong Kong, China. New York, NY: IEEE; 2000.31–435.
4. Shin T, Olano M. Rendering particles in a shaft of light; 2013.
5. Li S, Wang G, Wu E. A new approach for construction and rendering of dynamic light shaft. Comput Graph. 2008;32(6):660–668.
6. Jensen HW. Global illumination using photon maps. Rendering techniques '96. Vienna, Austria: Springer, 1996; p. 21–30.
7. Hachisuka T, Ogaki S, Jensen HW. Progressive photon mapping. ACM Trans Graph. 2008;27(5):130.
8. Hachisuka T, Jensen HW. Stochastic progressive photon mapping. SIGGRAPH Asia '09 ACM SIGGRAPH Asia 2009 Papers; 2009; Yokohama, Japan. New York, NY: ACM; 2009.141:1–141:8.
9. Hachisuka T, Jensen HW. Robust adaptive photon tracing using photon path visibility. ACM Trans Graph. 2011;30(5):114.
10. Kaplanyan AS, Dachsbacher C. Adaptive progressive photon mapping. ACM Trans Graph. 2013;32(2):16.
11. Knaus C, Zwicker M. Progressive photon mapping: A probabilistic approach. ACM Trans Graph. 2011;30(3):25.
12. Sanz-Pastor N, Barcena LA. Volumetric three-dimensional fog rendering technique. United States patent 6268861. 2001.
13. Harris MJ, Lastra A. Real-time cloud rendering. Comput Graph Forum. 2001;20:76–85.
14. Gutierrez D, Seron FJ, Munoz A, Anson O. Visualizing underwater ocean optics. Comput Graph Forum. 2008;27:547–556.
15. Jensen HW, Christensen PH. Efficient simulation of light transport in scenes with participating media using photon maps. Paper presented at: SIGGRAPH '98. Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques; 1998; Orlando, FL. New York, NY: ACM; 1998. p. 311–320.
16. Jarosz W, Zwicker M, Jensen HW. The beam radiance estimate for volumetric photon mapping. Paper presented at: SIGGRAPH '08 ACM SIGGRAPH 2008 Classes; 2008; Los Angeles, CA. New York, NY: ACM; 2008. p. 3.
17. Jarosz W, Nowrouzezahrai D, Thomas R, Sloan P-P, Zwicker M. Progressive photon beams. ACM Trans Graph. 2011;30(6):181.
18. Zhang Y, Dong Z, Ma K-L. Real-time volume rendering in dynamic lighting environments using precomputed photon mapping. IEEE Trans Vis Comput Graph. 2013;19(8):1317–1330.
19. Cammarano M, Jensen HW. Time dependent photon mapping. Proceedings of the 13th Eurographics Workshop on Rendering; 2002; Pisa, Italy.135–144.
20. Jonsson D, Ynnerman A. Correlated photon mapping for interactive global illumination of time-varying volumetric data. IEEE Trans Vis Comput Graph. 2016;23(1):901–910.
21. Jarabo A, Marco J, Muñoz A, Buisan R, Jarosz W, Gutierrez D. A framework for transient rendering. ACM Trans Graph. 2014;33(6):177:1–177:10.
22. Marco J, Jarosz W, Gutierrez D, Jarabo A. Transient photon beams. Paper presented at: SIGGRAPH '17 ACM SIGGRAPH 2017 Posters; 2017; Los Angeles, CA. New York, NY: ACM; 2017. p. 52.
23. Jarosz W, Nowrouzezahrai D, Sadeghi I, Jensen HW. A comprehensive theory of volumetric radiance estimation using photon points and beams. ACM Trans Graph. 2011;30(1):5.

Xingyu Lin received his B.S. in the school of EECS at Peking University in 2017. He is now a Ph.D. candidate in the Robotics Institute at Carnegie Mellon University. His research interest lies in the intersection of robotics, computer vision and reinforcement learning. Particularly, he is interested in developing more generalized robotic manipulation skills through model-free methods and imitation learning.

Mingxuan Chai received his B.S. degree in the school of EECS at Peking University in 2017. He is now a Master candidate at Carnegie Mellon University. His research focuses on the attention-based deep neural network including image fine-grained classification and speech emotion classification. Graphics is also his research interest. He is located in Silicon Valley now and expects to explore more of industrial computer science.

Sheng Li received both B.S. and M.S. in computer science from Shandong University in 1997 and 2000 respectively, and received his Ph.D. in 2005 from the Institute of Software, Chinese Academy of Science (CAS). Currently, he is an Associate Professor of the School of Electronics Engineering and Computer Sciences, Peking University. He works as a member of the Graphics and Interaction Lab. and publishes over 30 refereed papers in prestigious international conferences and journals. His research interests include computer graphics, virtual reality, physical simulation and animation. He is a member of ACM and IEEE.

Guoping Wang, Professor of Computer Science, Peking University, Associate Director of Institute of Software, and Director of Graphics & Interactive Technology Laboratory, Peking University. He got Bachelor's and Master's degree from Dept. of Mathematics, Harbin Institute of Technology in 1987 and 1990 respectively, and got Ph.D from Institute of Mathematics, Fudan University in1997. He achieved the National Science Fund for Distinguished Young Scholars in 2009. His research interests include Virtual Reality, Computer Graphics, Human-Computer Interaction, and Multimedia.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of the article.

How to cite this article: Lin X, Chai M, Li S, Wang G. Time-varying light motion in single convergence. *Comput Anim Virtual Worlds*. 2018;29:e1839. https://doi.org/10.1002/cav.1839