Research and analysis of ray tracing methods

Chenxi Qu[1]

[1]Department of Computer Science, University of Manchester, Manchester, UK

qu.chenxi@student.manchester.ac.uk

Abstract. In computer graphics, the study of how to build realistic scenes on the computer has been the focus of research. The dominant approach to simulating the correct light source was to use rasterization, but rasterization does not correctly simulate multiple reflections. To make virtual scenes more realistic, ray tracing was proposed to provide better results. With the rapid development of computer hardware, the use of ray tracing for rendering animation or games is becoming more and more common, and many scenes can now be rendered with ray tracing instead of rasterization for better results. This paper introduces and analyses the various methods of ray tracing in chronological order, starting with the original forward ray tracing and backward ray tracing, followed by Whitted algorithms with recursion, and then a series of methods introducing rendering formulas. And then, a few rays tracing algorithms introducing the Metropolis sampling method is mentioned. One can conclude that this method is going to be used for future optimization and development. It is significant for the development of ray tracing.

Keywords: computer graphics, ray tracing, Metropolis sampling method

1. Introduction

Ray tracing is a method of realistically rendering objects, first proposed by Appel in 1968. It has since been introduced by graphics practitioners from the fields of neutron transport, heat transfer, and illumination engineering, and has gone from being an extremely expensive and unproductive rendering method to a mainstream one, even replacing rasterization in some scenes. This paper first introduces the most basic forward ray tracing and backward ray tracing proposed by Apollo in 1986[1-2], followed by a ray tracing method introduced by Whitted in 2005[3] that allows the introduction of global illumination, which gives similar rendering results to rasterization. The results are like those of rasterization. This was followed by distributed ray tracing, introduced by Cook et al [4], which provided better results and used Monte Carlo random sampling for the first time. After this, the rendering formulation and path tracing method proposed by James T. [5] became the dominant ray tracing method. It was a milestone as almost all ray tracing methods since then had adopted the rendering formulation. Most of the recent research has optimized the raytracing rendering formulas using Monte Carlo integration.

This paper organizes these ray-tracing methods in chronological order [6].

2. Forward ray tracing and backward ray tracing

These two methods were the first ray tracing methods, and they both had their serious problems: too much time consumption and too low accuracy. But these two methods presented by Apollo [1-2] were also a milestone as it was the first time ray tracing was invented. This paper will then introduce each of the two methods and Hybrid ray tracing will be mentioned briefly in 2.3.

2.1. Forward ray tracing

This method calculates all the rays emitted from the light source, then calculates the reflected path of all the rays, and finally determines which rays hit the screen by traversing all the pixels on the screen. The only difference is that in the real world are area light sources, whereas in forward ray tracing it is assumed that the light is made up of multiple rays emitted from the light source. With the later introduction of the rendering formula, we can also use a surface light source to reach the propagation of light, but at this stage, we assume that the light is a point light.

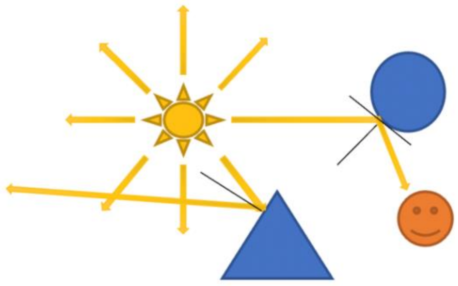

Figure 1. Forward ray tracing.

2.1.1. Advantage of forward ray tracing. Accurate results: This method fits well with real-life light propagation and the results obtained by this method are very accurate.

Easy to code, it is easy to write because of the simplicity of the algorithm.

2.1.2. The drawback of forward ray tracing. Time consumption problem: In this method, most of the rays are not necessary for the calculation. In Figure 1, the calculation of the one ray that reflects off the triangular object, or even all the rays that do not hit the screen, is pointless and wastes a lot of computing time with little impact on the result. This is the biggest problem with this method.

Sampling rate problem: As we mentioned before, this method assumes that the light source is a point light, so it is also a question of how many rays are assumed to be emitted by the light source. For example, in Figure 1, the sun is split into 8 rays. If we divide up many rays, we must render a good result, but we consume an incalculable amount of time, and the screen has a finite number of pixels, and one pixel can only display one single color, so once the number of rays reaches a critical value, we get a result that doesn't change. Conversely, we split the light so little that the result obtained will be distorted. We need to find a suitable way to split the number of rays based on the number of pixels on the screen before rendering.

2.2. Backward ray tracing [2]

In computer graphics we consider the path of light to be reversible. So we can assume that each pixel on the screen emits light and determine whether they are hitting the light source. We start by traversing all the pixels on the screen and calculating whether all the paths will pass through the light source. All pixels that pass through the light source are rendered with the corresponding colors [2, 7].

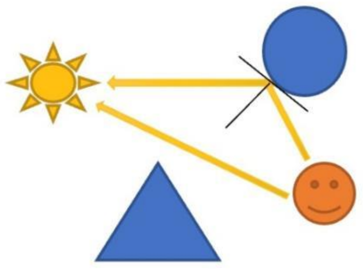

Figure 2. Backward ray tracing.

2.2.1. Advantage of backward ray tracing. Time complexity: This method has a very significant reduction in time complexity and is as easy to write as the forward ray tracing method.Sampling problems solved: We do not need to consider sampling problems in the same way as forward ray tracing, because we do not need to consider how many rays the light source emits.

2.2.2. The drawback of backward ray tracing. Accuracy issues: The results obtained by this model are not necessarily accurate. The downfall of backward ray tracing is that it assumes only the light rays that come through the view plane and on into the eye contribute to the final image of the scene. For example, the light source emits l1, l2, and l3 light onto the view plane as shown in Figure 3.1. This is because the lens is a convex mirror. So, these rays will converge into a single point on plane s2. We are only focused on a reflected ray l7 from this point s2, which passes through plane s1 and is reflected onto the view plane. This is the normal state of reflection of light. However, what happens in backward ray tracing is shown in Figure 3.2. The light received by plane s2 is uniform and not a point. This is because the backward ray tracing method assumes that the light is emitted from the screen and thus ignores the reflection process behind the screen.

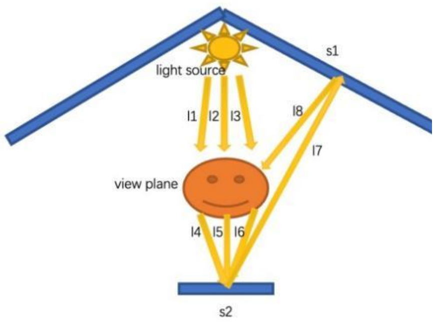

Figure 3 (a). Correct ray tracing

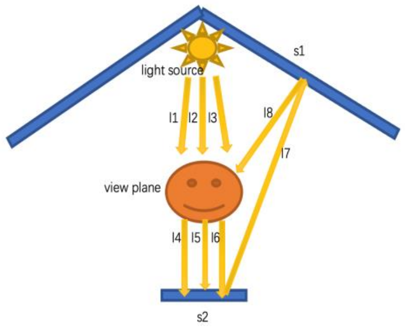

Figure 3 (b). Backward ray tracing.

In this method, we obtain a light path in which almost no light reaches the light source at the end. This is because light from reflection or refraction can come from many unpredictable directions. Unless there is an unobstructed straight line between the light source and the observation point, it may only provide a very weak illumination. The only way to resolve this is for us to sample all points throughout the environment to determine the correct reflection path of the light path. But this would consume a very exaggerated amount of time. So, this method will not be used in complex images.

2.3. Hybrid ray tracing

This method is a generic term for a class of methods that we can call Hybrid ray tracing as long as we apply both backward ray tracing and forward ray tracing. Figure 4 briefly depicts this method.

The basic models mentioned in the appeal, forward and backward, both have serious problems: the time complexity and the accuracy. Since both have drawbacks, we use both methods together, calculating both the light emitted from the screen and the light emitted from the light source to achieve a solution with moderate accuracy and time complexity. This method can also add useful information to the base image, such as reflections and more detailed ambient lighting. For example, Bidirectional Path Tracing is based on hybrid ray tracing, a method proposed by Veach in 1995 [6,8]. This method was proposed by Veach in 1995. It simply connects the reflection and refraction points of the light from front to back and from back to front to express global illumination. As this method is based on path tracing in the first place, we will refer to the details of this method after we have introduced the rendering equation.

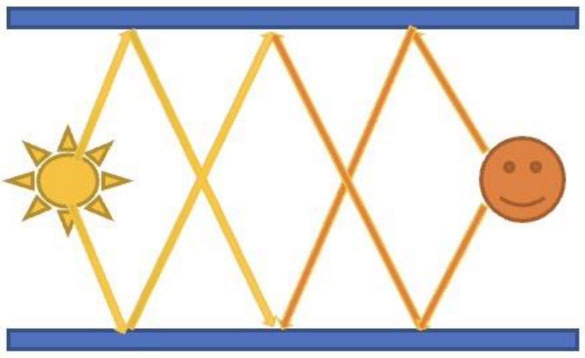
Figure 4. Hybrid ray tracing.

2.4. Whitted ray tracing

In 1979, Turner Whitted [3] developed his own Whitted ray tracing by introducing reflections, refractions, and shadows to prolong the light projection process. This lighting model makes extensive use of techniques previously derived from Phong and Blinn, and it operates recursively to achieve a global illumination effect. It is a landmark model. The light is divided into four types:

- Eye ray. This is the same as the ray emitted from the screen in backward ray tracing.
- Reflected ray. This is the ray that is reflected from the screen after a specular reflection.
- Refracted ray. This is created similarly to a reflected ray, except that it is directed into the object and can eventually exit it. An example is a transparent glass ball.
- Shadow rays. These are calculated by creating shadow rays from the point of intersection to all lights. If a shadow ray intersects an object before it reaches a light, that intersection will be shown shaded from that light.

As above we have given four classifications of rays, the aim being to allow a single incident ray to have multiple emitted rays at the same time. For example, as shown in Figure 5, in the model proposed by Whitted [3] back in the day, when the light hits a glass sphere, the path found by backward ray tracing is not correct because the backward ray cannot be split in two, so this method cannot achieve both refraction and reflection. However, for Whitted ray tracing, it is possible to split the path of the light at this point into a refracted ray and a reflected ray, thus achieving global illumination.

Change the solid ball in Figure 2 to a glass ball in Figure 6. We see that with backward ray tracing, the light does not split into two when it passes over the surface of the glass sphere, and this is where Whitted ray tracing has the advantage of being able to handle global illumination simply.

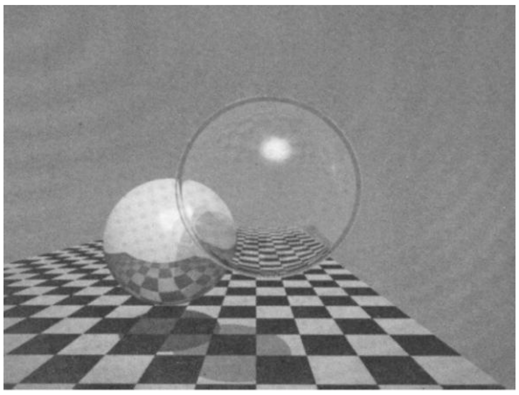

Figure 5. Rendered by whitted [3].

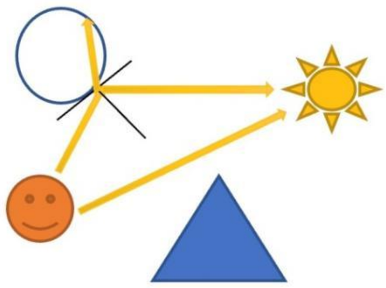

Figure 6. Whitted ray tracing.

2.4.1. Advantage of whitted ray tracing. Global lighting is introduced compared to the previous method. The rendering has been given a milestone boost.

2.4.2. The drawback of whitted ray tracing. Diffuse reflections are provided, but not suitable for diffuse reflections from distributed light sources and specular reflections from surfaces with a low gloss level. For example, if we have an uneven surface on which light is supposed to be diffusely reflected, the reflected light in Whitted ray tracing will not be deflected and will have the effect of specular reflection. There are two ways to deal with this problem. The first is to make multiple recursions of the ray, each of which sets the angle of the normal to the point of reflection at random. The second is distributed ray tracing, which we will describe next.

2.5. DistributedRay tracing

Distributed sampling is a ray tracing algorithm based on the Monte Carlo method proposed after Whitted ray tracing provided by Robert L. Cook [4]. Because it does not introduce a rendering formula, it is grouped with Whitted ray tracing. The specific details of this algorithm are described next.

Distributed ray tracing is a method of ray tracing based on randomly distributed sampling. The main idea of distributed ray tracing is based on the anti-aliasing method. Each pixel is oversampled and can be averaged for anti-aliasing. We can also use the same principle for reflection points, where the reflection point can emit multiple rays per refraction, just like the light source in forward ray tracing, whereas in Whitted ray tracing the reflection point emits only one ray, so it is not possible to Distributed ray tracing generates multiple rays at random to achieve a diffuse reflection that is not possible with Whitted ray tracing. Figure 7 shows what happened in distributed ray tracing.

2.5.1. Advantage of distributed ray tracing. This method is a reasonable solution to the problems of Whitted ray tracing, which cannot handle diffuse reflections and specular reflections on surfaces with low gloss. Distributed ray tracing, on the other hand, uses Monte Carlo random sampling to achieve the effect of diffuse reflections.

2.5.2. The drawback of distributed ray tracing. One of the biggest problems in Distributed ray tracing is that each reflection increases exponentially and has a decreasing global impact. For example, each ray emitted is reflected 10 times. The time complexity is O(n^10), and as more rays are emitted, subsequent rays may not have an appreciable effect on the result but take 10 times more time than the first few reflections. This disadvantage is like the high time consumption of forward ray tracing, both of which are extremely accurate and therefore take a relatively high amount of time. The next section on path tracing will present a solution to this problem.

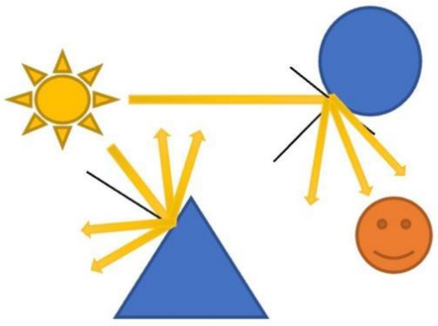

Figure 7. Distributed ray tracing.

3. Pathtracing

Kajiya [5] introduced the rendering equation in 1986, a landmark formula as almost all ray tracing algorithms since then have referred to this formula, which follows the physical propagation of light. In physics, if the spectrum is continuous, it can be solved using definite integrals, and the rendering equation introduces this process into computer graphics. The light of each pixel is made up of light absorbed from the outside world and light emitted by itself, adding them together and solving the ray-tracing problem using definite integrals over this function.

This paper will first introduce the rendering Equation (3.1), then introduce the Monte Carlo integration method for solving the rendering Equation(3.2), and finally, introduce the path tracing algorithm and a simple way of optimizing it (3.3).

3.1. Rendering equation

Kajiya[5] proposed the rendering equation in 1986. He treated the various rendering methods as problems of solving a complex definite integral. As we mentioned, the idea behind the rendering equation is not new. The description of the phenomena modeled by this equation has been well-studied in the radiation heat transfer literature for many years [Siegel and Howell 1981]. However, this approach was first proposed in computer graphics, where all rendering methods are treated as modeling the same physical phenomenon, i.e., the problem of scattering of light on various surfaces. The derivation of the rendering equation involves the action of light with the surface of an object, radiometry, and BRDF. The focus of this paper is to develop a discussion of the advantages and disadvantages of the dominant ray tracing methods, so no derivation of the rendering equations will be made.

3.2. Monte carlo

In the paper where Kajiya [5] presents the rendering formula, he also gives a method for solving It. In summary, by rendering the equations we can see that solving a ray-traced path is essentially a process of solving a difficult definite integral with infinite recursion. Recursion can easily be achieved by for loop iteration. And to solve a definite integral, we can simply solve it by Riemann integration. But as we move from the primitive forward ray tracing to the ray tracing equations we are talking about now that refer to the rendering equation, the primary consideration is the time complexity problem.

If we were to solve the definite integral by Riemann integration, which simply means dividing the definite integral into an infinite number of small rectangular parts and summing their areas to obtain the solution of the definite integral, we would indeed get a very accurate result. However, it would also take a lot of time to compute, so we need another form of solving the definite integral which spend less time. Here, the Monte Carlo integration method is used, which simply means that these small rectangles of the Riemann integral are sampled randomly. The heights of all the rectangles are accumulated to obtain an average height, equivalent to a rectangle for the area, to solve for an approximation of a continuous definite integral. The more rectangles we take, the more accurate the solution of the definite integral will be. We can set a hyperparameter in the algorithm to determine the average of how many rectangles we randomly choose to go to, and this can determine how accurate and how much time Monte Carlo spends solving the definite integral.

3.3. Path tracing

Path tracing is a variant of distributed ray tracing, the essence of which is that we track the rays that have a large impact on the result, effectively. In this case, each point has only one reflected and refracted ray. While this prevents ray proliferation, one ray is too few, and in this case, there is only one reflected and refracted ray for each point where light is reflected and refracted. While this prevents ray proliferation, one ray is insufficient, and the implementation results in a very noisy image. [9-10].

The solution to this problem is to perform the ray reflection operation multiple times and average this out over many iterations. Simply put, we can still perform 100 operations on each ray as we did in distributed ray tracing, but this time instead of revealing all the rays, we average them together to get a single ray, so that there is neither an exponential explosion nor a lot of noise. There is also the problem that we don't set how many times the recursion ends. Here we could use Russian roulette, where each recursion has a probability of terminating with a certain probability.

3.3.1. Advantage of path tracing. Since more visible light is emitted per pixel, camera effects such as depth of field and motion blur can be combined at little additional cost.

Accuracy: This method introduces the rendering formula, and the results he obtains are precise because the rendering formula is derived from the radiometric inversion and corresponds to the physical propagation of light.

3.3.2. The drawback of path tracing. It is more difficult to ensure a good distribution of reflected rays than with distributed ray tracing. In short, distributed ray tracing emits the lightest in the ray tree, while path tracing emits the lightest at the beginning. This is to be expected that any reduction in time spent by the algorithm is accompanied by a reduction in accuracy.

In some cases, bidirectional ray tracing performs very poorly. For example, it has difficult visibility (e.g. a small hole), or concave corners where two surfaces meet (a form of singularity in the integrand). The problem is that bidirectional mutations are relatively large, and so they usually attempt to mutate the path outside the high-contribution region[6].

4. Recently proposed methods

4.1. Bidirectional path tracing

This algorithm was proposed by Eric P LaFortune and Yves S Willems [8]. It is a type of mote Carlo path tracing. We first use path tracing to calculate the light emitted by the light source and the light seen by the observation point. We calculate all the reflection points of the objects passing through these two paths. The reflection points on these two paths are connected. A probabilistic approach is used to take into account different lighting contributions from primary light sources as well as significant secondary, tertiary, and other light sources.

In typical indoor scenes with indirect lighting, the algorithm is better than path tracing. In indoor scenes where indirect lighting is important, this algorithm is better than path tracing. The algorithm gives better results than path tracing in normal indoor scenes, especially in indoor scenes where the effect of indirect lighting is high. The method requires no meshing and thus avoids all the associated problems [8].

4.2. Metropolis path tracing (MLT)

Metropolis et al. propose a new sampling method to solve complex sampling problems in computational physics, originally intended to compute the material properties of fluids. The advantage of the Metropolis sampling method is that it is unbiased [6]. The advantage of the Metropolis method is that it is unbiased because these methods sample the path from the light source to the viewpoint in the scene without taking into account the path distribution characteristics of the scene, using a random sampling method to sample the path and calculate the contribution. If the items in the scene are very specific, the previous sampling method wastes a lot of computing power. To render an image, we generate a sequence of light transport paths by randomly mutating a single current path (e.g. adding a new vertex to the path). Each mutation is accepted or rejected with a carefully chosen probability, to ensure that paths are sampled according to the contribution they make to the ideal image [6].

Markov chains are the necessary formula for metropolis path tracing. A Markov chain simply means that the event that occurs next is only related to the probability of the current event occurring, and not to the probability of the previous or further events occurring. By using Markov chains, this method has similar rendering results for most images with different conditions.

4.2.1. Advantage of metropolis path tracing. The results of Metropolis optical transmission are far better than those of bi-directional path tracing. Bidirectional Path Tracing path tracing is difficult to obtain details such as contact shadows, caustics under the glass teapot, light reflected by the white tiles under the door, and the brighter strip along the back of the floor (due to the narrow gap between the table and the wall).

Figure 8 shows another difficult lighting situation: caustics on the bottom of a small pool, are seen indirectly through the ripples on the water's surface. Path tracing does not work well, because when a path strikes the bottom of the pool, a reflected direction is sampled according to the BRDF. In this case, only a very small number of those paths will contribute because the light source occupies about 1% of the visible hemisphere above the pool. (Bidirectional path tracing does not help for these paths, because they can be generated only starting from the eye.)

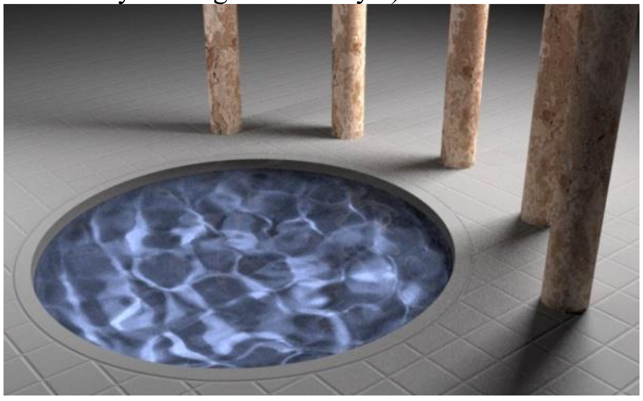

Figure 8. Caustics on the bottom of a small pool [6].

4.2.2. Drawback of metropolis path tracing. MLT can robustly solve most scenarios, but some light paths are very difficult to pick up by random sampling, especially when ray tracing specular reflections, i.e. light paths that are connected by perfectly mirrored or refracted surfaces. This is because for a specular or refractive surface if the direction of incidence is certain, the direction of exit is also uniquely determined, but it is essentially impossible for MLT to obtain a uniquely determined light by random sampling. Therefore, the existence of such an optical path is difficult to solve by unbiased random sampling methods such as MLT.

4.3. Manifold exploration

A persistent problem with unbiased Monte Carlo methods for rendering is that certain difficult types of light transmission paths (such as MLT), especially those involving viewing and illumination along paths containing specular or glossy surfaces, can lead to unusually slow convergence. In 2012, Wenzel Jakob [7] and Steve Marschner gave a new method for dealing with specular paths in rendering. It is based on the idea that the set of paths contributing to an image naturally forms a manifold in path space that can be explored locally by simple equation-solving iterations.

5. Conclusion

This paper presents the advantages and disadvantages of the main methods of ray tracing in chronological order. For the basic forward ray tracing and backward ray tracing, future optimization of these methods is difficult because their models are too simple. However, it is possible to optimize these two methods by combining them, i.e., Hybrid ray tracing. For example, in hybrid ray tracing we can use backward ray tracing for direct light rendering. And we can use forward ray tracing and Russian roulette to simply generate global lighting. On the other hand, for path tracing methods that introduce rendering formulas based on Monte Carlo integration, accuracy is the first and foremost issue, as these methods are random sampling with weighted values and none of them works well for all scenes. Although some unbiased methods have been proposed, they are also not applicable to all scenes. For example, MLT cannot handle perfect specular reflections. So, for these methods, it is an area of future research to develop new random sampling and more comprehensive unbiased ray tracing methods.

References

[1] An introduction to ray tracing. 1989, Morgan Kaufmann. p1-p31.
[2] Arvo J. Backward ray tracing, Developments in Ray Tracing, Computer Graphics, 1986, Proc. of Acm Sig.86 259-263.
[3] Whitted T. An improved illumination model for shaded display, 2005, Acm Sig.4-es.
[4] Cook R L, Porter T, Carpenter L. Distributed ray tracing. 1984, Conf. Com. Gra. Int. Tec. 137-145.
[5] Kajiya J T. The rendering equation. 1986, Conf. Com. Gra. Int. Tec. 143-150.
[6] Veach E, Guibas L J. Metropolis light transport 1997 Conf. Com. Gra. Int. Tec. 65-76.
[7] Jakob W, Marschner S. Manifold exploration: A Markov chain monte Carlo technique for rendering scenes with difficult specular transport. 2012, ACM Trans. Graph. 31 (4): 1-13.
[8] Veach E, Guibas L J. Optimally combining sampling techniques for Monte Carlo rendering 1995 Conf. Com. Gra. Int. Tec. 419-428.
[9] Saito H, Yamamoto T, Nakajima K, et al. Identification of the Infrasound Signals Emitted by Explosive Eruption of Mt. Shinmoedake by Three-dimensional Ray Tracing. 2021 J. Ac. Soc. America, 149 (591): 591-598.
[10] Rf A, Ke B, Sk C, et al. Stray light analysis by ray tracing simulation for the wide-angle multiband camera OROCHI onboard the Martian Moons eXploration (MMX) spacecraft – Science Direct. 2021 Adv. Sp. Res., 69 (2):1236-1248.