

Methods for Calculating and Visualizing the Spatial Distribution of Illumination in Three-dimensional Models of Optically Complex Scenes

Igor Kinev¹, Dmitry Zhdanov¹, Andrey Zhdanov¹ and Igor Potemin¹

¹ ITMO University, Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia

Abstract

In this paper, we consider the problem of calculating the effective protection of an image from spurious radiation, one of the causes of which is the scattering of light on non-imaging elements of an optical device. The purpose of this work is to improve the means of searching and visualizing the sources of parasitic illumination of an image for designing and analyzing the light protection of optical devices. A visualization of the spurious radiation illumination distribution on the surfaces of an optical device is proposed. The global illumination calculation is based on an advanced bi-directional stochastic ray tracing method with backward photon maps. This method is highly efficient, since it simultaneously uses direct and backward photon maps, which accumulate the distributions of the areas of light emission and the visibility of the scattering elements of the optical device, respectively. The method allows representing the average illumination of the image, accumulated on the diffuse objects of the optical device that form the given illumination. By visualizing it, you can more effectively and clearly find the elements of the optical device that create spurious illumination in the image and degrade the image quality. With this information, the designer of an optical device can develop special means of light protection or use special light-absorbing coatings for such elements. The proposed algorithms were implemented in the Lumicept software package, with the help of which the scattered light in the lens objective was calculated and the result of the stray illumination visualization was presented.

Keywords

Virtual prototyping, light scattering, optical device, visualization, photon mapping, realistic rendering, ray tracing.

1. Introduction

Nowadays, photorealistic computer graphics uses global illumination calculation methods for modeling physically correct images, which allow taking into account various physical phenomena such as light refraction, reflection, scattering, dispersion, interference, caustics, etc.

These methods solve the rendering equation [1], which describes how light propagates in the scene depending on the properties of the surfaces of each object. Since the rendering equation is infinitely complex and is calculated by recursively summing the luminance integrals over the hemisphere [2, 3], then Monte Carlo methods [4] are used to solve it, which allow us to interrupt the calculations and estimate the value of the illumination or luminance integral after a finite time.

Modern software solutions for prototyping optical devices use these methods for efficient and physically correct calculation and visualization of light propagation in optical systems. However, there is a serious problem in calculating the effective protection of the image from background illumination. One of the reasons for the occurrence of background illumination is the scattering of light on non-imaging elements of the optical device. Therefore, methods and algorithms for visualizing the sources of scattered illumination in optical devices are of particular relevance.

GraphiCon 2022: 32nd International Conference on Computer Graphics and Vision, September 19-22, 2022,

Ryazan State Radio Engineering University named after V.F. Utkin, Ryazan, Russia

EMAIL: igorkinevitmo@gmail.com (I. Kinev); ddzhdanov@mail.ru (D. Zhdanov); andrew.gtx@gmail.com (A. Zhdanov); ispotemin@yandex.ru (I. Potemin)

ORCID: 0000-0003-2929-1203 (I. Kinev); 0000-0001-7346-8155 (D. Zhdanov); 0000-0002-2569-1982 (A. Zhdanov); 0000-0002-5785-7465 (I. Potemin)



© 2022 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

When analyzing scattered light, it is necessary to determine the characteristics of light (illuminance, luminance, intensity) at the illumination receiver and understand the root cause of the scattered light. The first case is handled well by optical prototyping methods, which present the results in the form of graphs and tables, but they cannot answer what is the cause of the backlight on the receiver. To find the root cause, ray imaging techniques are used that can visualize the propagation of the light in an optical system. However, such methods have a significant drawback, which lies in the fact that when modeling a large number of rays (i.e., billions), they are superimposed on each other, thereby complicating the process of scattered light analysis.

2. Related works

Designing the light protection of optical devices from stray light illumination that enters the illumination receiver and degrades the image quality is one of the most important tasks in the process of designing optical devices. Parasitic illumination occurs due to light scattering on non-imaging elements of an optical device (lens body, lens frames, diaphragms, lens ends). A new approach to solving the problems of computer simulation of the propagation of scattered light in optical devices is the use of realistic rendering methods, which make it possible to synthesize and visualize images formed by optically complex scenes containing a large number of objects.

There are two main tasks to be solved for modeling the scattered light. At first, determine the sources of scattered light, which are the root cause of the occurrence of stray light on the receiver. Secondly, calculate the illumination at the illumination receiver. If the scene contains surfaces or optical properties that make it difficult to calculate the luminance of global illumination (multidimensional scattering function) or very complex ray paths are formed (multiple reflection, refraction, complex caustics, glare), then the calculation process becomes laborious and requires the use of special physically correct and efficient methods of stochastic ray tracing [5]. However, these methods require generating a large number of Monte Carlo samples to achieve an acceptable accuracy, which slows down the calculation and increases the amount of memory used. To solve this problem, Metropolis methods were developed [6, 7]. They are better at rendering complex indirect illumination but have poor performance when calculating caustic lighting. Photon mapping methods [8, 9] allow efficient calculation of caustic and indirect illumination. Their advantage lies in the fact that they allow you to calculate complex scenes with global illumination, including caustic illumination, and quickly estimate the illumination of the image. The disadvantage is that these methods are biased, but the correct choice of the integration sphere radius and the indirect illumination collection point makes it possible to reduce the bias error of this method to a minimum.

In most cases, two methods are used to obtain the result of stray light simulation. The first one is to calculate the distribution of light characteristics at the receiver by using virtual light propagation prototyping in an optical device and present results in the form of graphs and tables. The second method is to visualize the ray paths [10]. The first method produces a more accurate calculation, but its disadvantage is that even though it allows you to see the result, however, it does not show the cause of the stray illumination. For example, we know the amount of background light, but we cannot determine the cause of its occurrence, perhaps it is lens flare, aperture flare, etc. The second method clearly shows information about how the rays propagate in the scene, which makes it possible to understand the root cause of the occurrence of stray illumination, but when there are many such rays (e.g., a million, a billion), they start overlapping each other and make analysis difficult.

To solve these problems, it is proposed to visualize the distribution of stray light illumination on the surfaces of an optical device. For efficient calculation of global illumination, the method of progressive bidirectional ray tracing with backward photon mapping [11] was chosen, which allows choosing the optimal radius of the integration sphere, due to which the calculation speed is significantly increased. In addition, this method builds a water content map, which allows you to reduce the amount of memory required for the calculation.

It is also proposed to visualize ray traces, namely indirect illumination representing the average illumination of the image accumulated on the diffuse objects of the optical device that form this illumination. The accumulation and visualization of this illumination occur on diffuse objects of the scene, which allows the designer of optical systems to visually analyze the scene faster and more clearly

understand which elements in the device create stray light illumination on the image sensor. Knowing this information, the designer can change the properties of the element or the design of the optical device and improve the quality of light protection. Also, these methods will be useful to software developers for debugging and optimizing rendering algorithms, helping to determine areas in which a high ray density is formed.

3. Light simulation

As mentioned above, in this paper authors propose not only to visualize the ray paths according to some criteria but also to visualize the sources of the ray scatterings causing caustic and indirect image illumination. While passing through the scene, the ray collides with various surfaces which can be sources of indirect and caustic image illumination. And information about sources of the illumination sometimes is more important than rendered image. This is a case of stray light analysis in the optical systems. There are two problems: the first is to find the sources and estimate how the source influences on stray light illumination of the image, and the second is to visualize the sources. Also, we should understand that stray light is a low efficient illumination from the ray tracing method point of view. If the coefficient of the optical system transparency for image rays is about 0.9, then for the stray light rays the coefficient is about from 10^{-6} to 10^{-7} . So, the stray light calculations are very inefficient and require special solutions for image rendering and calculation of the stray light sources. Taking into account all these problems it was proposed to use the solution based on bidirectional ray tracing with backward photon mapping. So, the proposed algorithm consists of the following main steps:

1. Create a three-dimensional grid in the scene (the grid will accumulate an illumination that is created on the image);
2. Emit rays from the camera from each image pixel in the random direction to the exit aperture (for the stray light analysis in the lens device the camera image coincides with the lens image);
3. Create the view map;
4. Emit rays from light sources (for stray light analysis the light sources are out of the field of view ones which create parasitic image illumination);
5. Use the calculated backward photon maps to calculate the indirect and caustic illumination;
6. Accumulation of the caustic and indirect illuminance on the cells in the cells of the three-dimensional grid created in step 1;
7. If calculations should be continued (e.g., the required rendering accuracy is not achieved), then go to step 2;
8. Normalize the illuminance accumulated in the three-dimensional grid cells;
9. Visualize values accumulated in the three-dimensional grid.

Steps 2-7 are standard steps of the bidirectional ray tracing with the backward photon mapping algorithm and are repeated until the required simulation accuracy is achieved or simulation time is exceeded. The rays are traced in the scene, on the scattering surfaces the maps are created, and when a light ray hits the map the corresponding (indirect or caustic) luminance component is calculated. If the camera has a nonzero aperture the luminance can be recalculated to the image illuminance which is stored in the three-dimensional grid of the scene (step 6). It should be noted that the solution with photon maps is biased which can result in a corresponding calculation error. On the other hand, the radii of the backward photons are usually very small and depend on the visible pixel size and the size of the observed primitive. Moreover, the error is reduced if the goal is an estimation of the influence of the scene element on the average image illumination.

Consider how the influence of the scene element on the average stray light image illumination is estimated. Light rays which hit integration spheres of backward photon map create luminance visible from the image. Taking into account the lens aperture the visible luminance can be recalculated to the local image illuminance (and then after the luminance is integrated, we will see the whole image illuminance). The local luminance can be accumulated both on the image and on the scene element (the cell of the three-dimensional grid) which creates the illuminance. Considering that one cell of the three-dimensional grid can accumulate illuminances corresponding to the different points of the image it is very expensive to store where (image pixel) the ray scattering adds local illuminance in the image. So,

the grid cell will contain the average illumination of the image. The process of the illuminance accumulation is illustrated in Figure 1, local illuminances of the different image pixels E1 and E2 are accumulated in one cell of the three-dimensional grid. It should also be noted that to reduce the memory required to store the three-dimensional grid it can be reorganized to use some acceleration structure (e.g., hash tables, k-d trees [12], etc.).

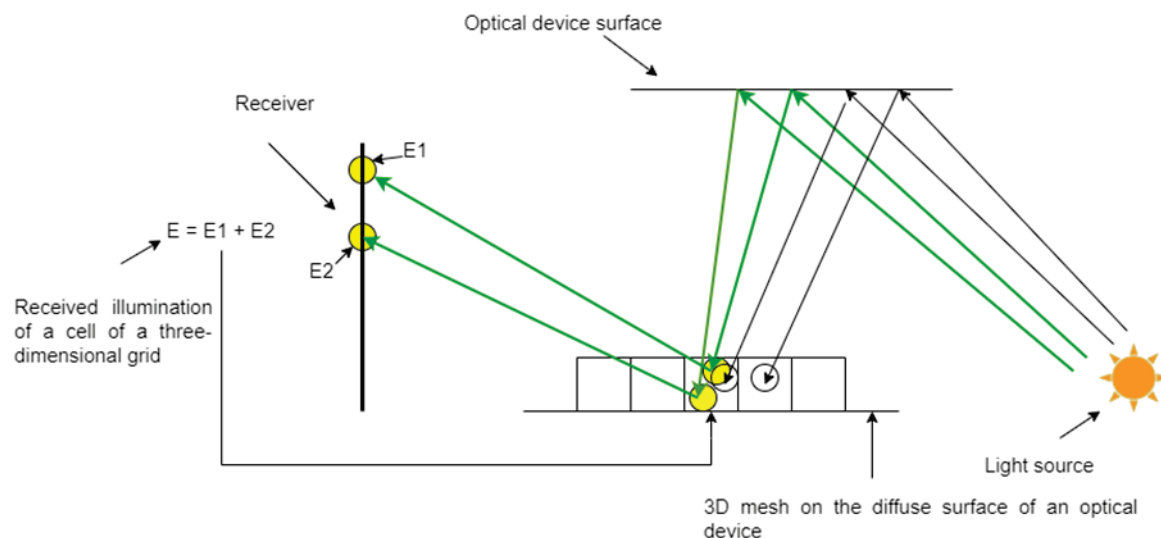


Figure 1: Accumulation of the illuminance in the three-dimensional grid cells

The photon mapping algorithm can separate different components of the illumination (direct view of the light sources, direct illumination, caustic illumination, and indirect illumination). From the viewpoint of the stray light analysis, it is important to separate different kinds of illumination, so the three-dimensional grid has a corresponding separation of the illuminances: one for the caustic illuminance, and another one for indirect illuminance. Moreover, a lens designer wants to see through the lens from the image side and understand which elements of the lens device he can see. It is important to understand that elements can be potential sources of the indirect and caustic illumination of the image, so the three-dimensional grid has an additional layer to store which lens device element the lens image sees.

Figure 2 (left) shows the process of three-dimensional grid creation. The main problem is the grid size. The grid size can be taken from the scene domain (minimum and maximum point coordinates of the scene). It should be noted that such a grid definition is not always optimal, since the scene size can be many times larger than the rendered part of the scene. This situation arises, for example, when the out-of-the-field-of-view light source is a complex scene object and the lens designer has no need to see how the indirect and caustic illumination components are distributed inside this complex light source. Moreover, the large scene size may require a high resolution of the three-dimensional grid leading to the allocation of additional memory to store the illuminance data. To avoid this, it is proposed to manually set the minimum and maximum points on which the three-dimensional grid will be built. Another solution that allows for determining the grid size is to make small pre-calculation of backward photon maps. The minimal and maximal coordinates of the map points will correspond to the size of the three-dimensional grid. Of course, the last solution is applicable for scenes with simple models of light sources without internal interreflection. Additionally, the resolution of the three-dimensional grid should be set manually.

After the formation of a three-dimensional grid, the algorithm goes through all the rays stored in the photon maps and links the photons to the cells of the three-dimensional grid, Figure 2 (middle) illustrates it. Depending on what we want to accumulate in a cell (the number of camera rays, caustic illumination, or indirect illumination), this information is calculated or taken from the events of the ray-map interaction and is summed up in a cell.

As was pointed out above, to be able to see the individual components of the scene lighting, it is possible to generate three maps: caustic illumination, indirect illumination, and a visibility map. The visibility map forms the visibility of the scene. It saves all the rays saved in the backward photon maps. Each cell accumulates the number of rays and the total area so that the visibility map provides information about areas with the highest and lowest spatial concentrations of rays. The caustic and indirect maps accumulate the illumination generated by the caustic and indirect components, respectively. To do this, after the formation of a three-dimensional grid the backward rays are traced, and the backward photon map is created and linked with the cells of the three-dimensional grid. When a light ray hits an integration sphere of the map the image illuminance is calculated and stored in the image. Additionally, the image is stored in the three-dimensional grid cell linked with the map point. Figure 2 (right) illustrates the process. The calculation is done both for indirect and caustic illumination components separately, so the user can investigate the illumination components separately.

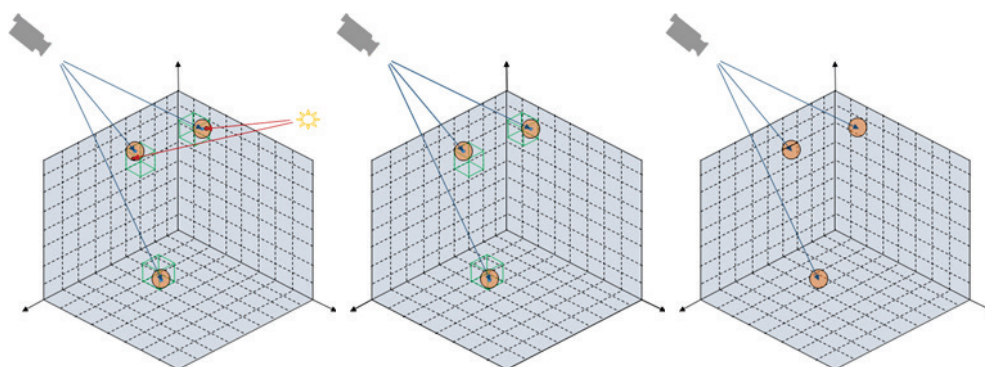


Figure 2: Generated 3D mesh consisting of cells (left); cells hit by the ray (middle, highlighted in green); search for intersections of a ray from a light source and a sphere area on a surface (right)

The last problem is the normalization of the results in the grid. Each of the three-dimensional grid cells is normalized similarly for the indirect or caustic luminance. That is the cell value is multiplied by total scene flux and divided by mixed ray counter (sum of multiplications of forward rays number to backward ray number of each render phase). This leads to an only restriction which is that the adaptive number of backward rays per pixel is not allowed, all pixel emits the same fixed number of rays at the rendering phase.

4. Visualization

The simulation method described in Chapter 3 allows for the calculation of the distribution of some physical values (number of backward rays, average indirect and caustic image illuminance) on the scene in the three-dimensional grid. Another problem is visualizing the three-dimensional grid. The simulation was fulfilled by a special mode of rendering (bidirectional ray tracing with backward photon mapping) for a special camera position (observer corresponds to the lens camera image and see sees the scene through the lens system limited by the exit aperture). The next stage is the visualization of the simulated results. Considering that the three-dimensional grid contains view independent values: the illuminance (not luminance) or the number of rays, the grid can be visualized from another camera position which can be more convenient from the scene analysis point of view, for example, to find the position of the sources of the stray light image illumination. So, the decision was made to display an image of the three-dimensional grid over the scene, for example, the lens device image, which can be visualized by any tool, e.g., OpenGL in shading or wire mode.

There are many ways to visualize the resulting three-dimensional grid with accumulated characteristics. For example, project points on a Z-buffer [13], or use OpenGL or DirectX technologies to project points onto the screen. These technologies create images very quickly and can be successfully applied in this algorithm, but since the software package has already implemented spatial ray tracing

methods, it was decided to use them for map visualization to facilitate the implementation of this algorithm. For this, the modification of Bresenham's algorithm [14] was proposed.

If the rays are emitted from the same perspective scene camera, then all rays will have the same initial coordinates, but their directions will be different following the location of the pixel into which the ray was emitted. If clipping planes were used, then the rays will be emitted from the nearest plane, and, accordingly, the origin coordinates will be different.

After the ray vectors have been found, it is checked whether the vector start point is inside the scene definition area. If it is not inside, then the first intersection with the scene's three-dimensional grid domain is searched, otherwise, the ray is not taken into account. Next, the value of the cell corresponding to the found point is checked to see if the corresponding cell value (illuminance or number of rays) is equal to zero. If the value is not equal to zero, then the value of this cell is written to the pixel, and tracing is stopped. Thus, the visualization allows us to see only the nearest cell of the grid. Otherwise, if the value of the cell is zero, then the ray is traced further from the given point and the next inner side of the cell's cuboid is searched. This is done to prevent ray tracing from looping in one cell. Once an intersection was found, the cell index is updated according to the face that the ray intersected (the ray can intersect with up to three faces of the cube) and its direction, i.e., the index can decrease or increase depending on the direction in which the beam moves from the camera. The update takes place along the X, Y, and Z axes of the generated map. These steps are repeated until a cell with a value greater than zero is ever found, or the ray goes beyond the three-dimensional grid domain or clipping plane (if it is set) is found (Figure 3).

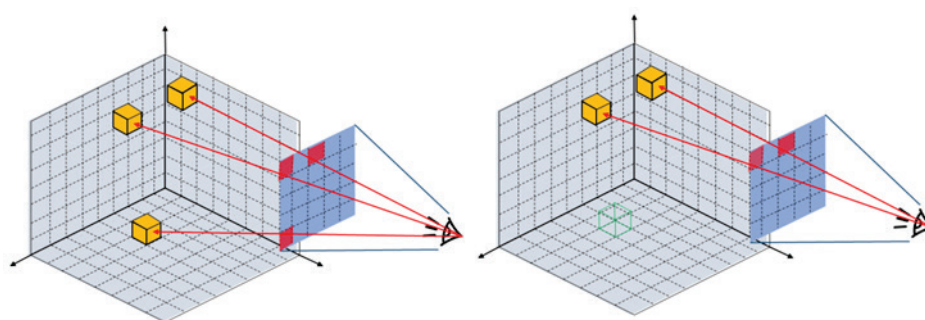


Figure 3: Map rendering process. Visibility maps (right); caustic or diffuse illumination map (left)

After knowing which cells correspond to the pixels on the screen, tone mapping is applied. In the proposed visualization algorithm, a certain color is overlaid under the value of the accumulated characteristic in the cell. With a large value of the characteristic (relative to all values calculated in a three-dimensional grid), the color of the pixel becomes brighter, and with a small value, it becomes darker. You can also manually change the brightness of dots on the screen by changing the tone mapping settings. The rendered color is multiplied by a factor set by the user in the program interface, which changes the brightness of the pixels of the 3D grid on the screen.

5. Removing outliers

During the accumulation of the light characteristics, outliers may appear in the three-dimensional grid, which represent values significantly different from most of the values obtained during the entire calculation. This is due to the use of stochastic integration methods. Figure 4 shows an example of a histogram of all values without outliers (left histogram), and with outliers that take on large values (right histogram)

To show the contribution of a single cell in the image, a coefficient is calculated that determines the brightness of a pixel depending on the value of the cell value, and then this coefficient is multiplied by the color of the pixel. Thus, cells with a large value have a brighter pixel color, while cells with a low value have a darker one. The coefficient is 0 at the smallest value among all cells and 1 at the maximum.

Very large outliers can cause most of the values to become dark or completely black since the coefficient is determined based on the maximum and minimum values. Values that are close to the

maximum overshoot will be bright, while most values much less than the maximum overshoot will be dark. The opposite is true for spikes with very small values, most of the pixels will become bright, and spikes will be dark.

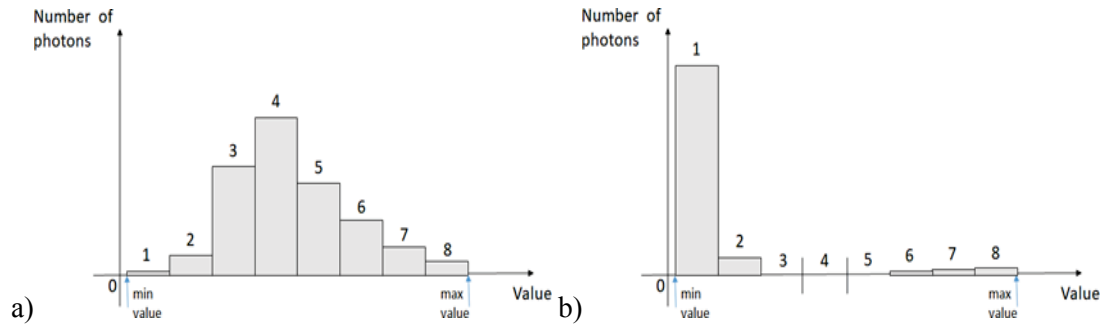


Figure 4: Histogram without outliers (left) and histogram with outliers taking large values (right)

To avoid this, a histogram of values is built from the minimum to the maximum values with a given number of divisions. After that, a binary search checks in which interval the value of each cell is located. Next, it is determined how many values fell into a particular interval. The interval with the largest number of values is the median, based on which a linear function is built that determines the brightness of a pixel depending on its value. An example of this straight line is shown in Figure 5.

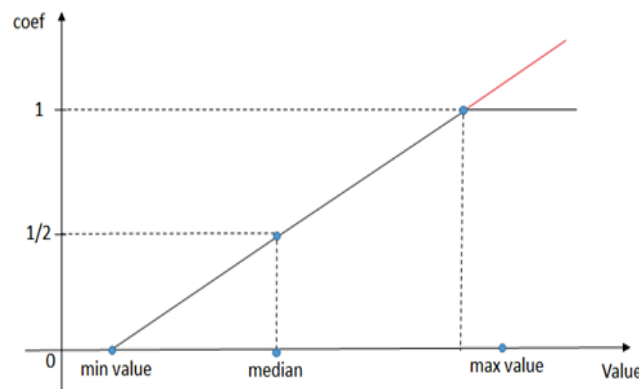


Figure 5: Linear function by which the coefficient is calculated

On the horizontal X -axis, the minimum and maximum values in cells are plotted, and on the Y -axis, the coefficient is from 0 to infinity. At the median point on the X -axis, Y is set to 0.5. That is, if the cell value is close to the median, then the coefficient will be about 0.5, with decreasing the value it will also be decreased, and with increasing the value, respectively, increased. But the coefficient can become greater than 1, in which case its value is clipped to 1 so that it is in the range from 0 to 1.

The formula for calculating the coefficient is given below:

$$coef = \frac{1}{2} \cdot \frac{\alpha \cdot (V - V_{min})}{(M - V_{min})} \quad (1)$$

where α is the tonal mapping coefficient that affects the brightness of the displayed 3D grid cells, V is the cell value, V_{min} is the minimum value among all cells, M is the median of all 3D grid cell values.

If the outlier value differs by an order of magnitude from the median, then the range of histogram segments becomes too large, since the number of histogram bins is not enough to correctly find the median. Most of the histogram will be empty, outliers will remain at the edges of the histogram, and all the main values can be near zero with a large outlier value or near 1 with a small outlier value (Figure 6, left) since they all fall into the same range at the edge of the histogram.

To fix this problem, a certain percentage of clipping values from the total number of cells on the left and right sides of the histogram is set. After clipping, a new histogram without outliers is built and a

linear function is built based on it and a coefficient is calculated by which the color of the pixel is multiplied.

An example of cutting off 10 percent of the maximum values is shown in Figure 6, after which a new histogram is formed from the remaining 90 percent (Figure 6, right).

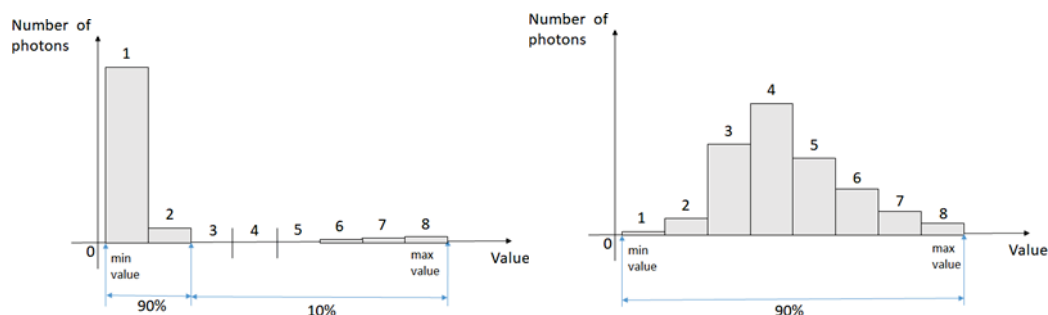


Figure 6: Clipping 10 percent of maximum values (left) and the new histogram formed from the remaining 90 percent of the values (right)

6. Results

This chapter presents the results of testing the developed methods for calculating and visualizing scattered light in optical systems. For this, two models of optical devices created in the software package were used. The first device is a six-lens telephoto lens with a focal length of 457 mm ($f/1.8$) and a field of view with an angle of $\pm 3.5^\circ$. An extended light source in the form of the sun, located outside the field of view, illuminates the lens at an angle $\omega_{Sun} = 5^\circ$. In this case, the illumination receiver does not see the light source directly because the field of view angle ($\pm 3.5^\circ$) is less than the angle of incidence of the rays of the light source ω_{Sun} . The computer model of the lens scene is shown in Figure 7 (left).

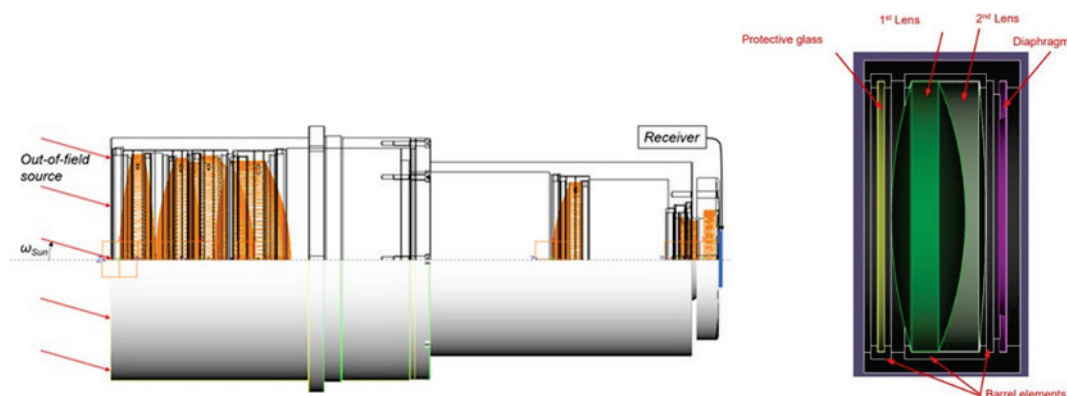


Figure 7: Scheme of the 6-lens telephoto lens (left) and projection of the lens (right)

In this case, only the rays that are scattered by any element of the given optical system can be accumulated by the illumination receiver. This means that when modeling the propagation of light in a given optical system, only scattered light on the diffuse surfaces of the components of the lens system is taken into account, e.g., scattered at lens mounts and rough end surfaces of lenses. To avoid reflection on transparent lens surfaces, it is assumed that all optical surfaces that form images are coated with a good anti-reflective layer with a reflectance coefficient equal to 0.01.

Figure 8 (left) shows the result of calculating the distribution of indirect and caustic illumination formed by the scattering of light emitted by an out-of-the-lens-field-of-view light source on the lens mounts of the optical system. These mounts and lens barrels are set to a reflectivity of 10 percent of the Lambertian scatter plot. Figure 8 (right) shows the same simulation, but with modified properties of the objects in the scene. All mechanical lens elements are set to 5 percent with a Gaussian scatter pattern and a 5° width at half maximum. All calculations time was the same and was 1 hour, on the workstation

with two AMD EPYC 7281 16-core 2.10 GHz processors and 256GB RAM. High image noise is caused by the low transmittance of the lens device for stray light.

The results are very different for different types of scattering properties of lens mounts. First, this manifests itself in the spatial distribution of scattered light in the plane of the illumination receiver, and secondly, in the amount of illumination accumulated at the receiver and caused by this scattered light. The result of the simulation with Gaussian reflectivity shows a lower stray light illumination of the illumination receiver than with Lambertian.

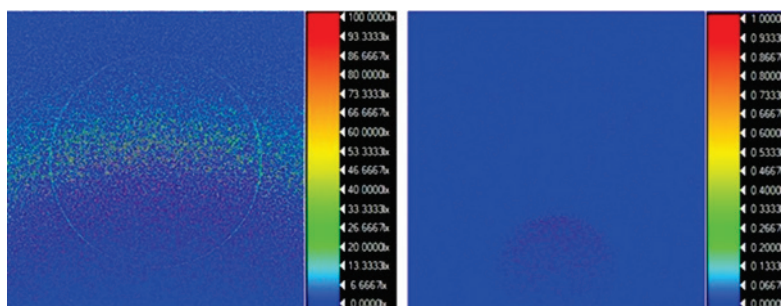


Figure 8: Illumination distribution of scattered light on the detector with Lambertian scattering on the lens mount (left) and Gaussian scattering on the lens mount (right)

Figure 9 and Figure 10 provide the results of calculation and visualization of the accumulated visible illumination of the image, on the surfaces of the scene of the optical system.

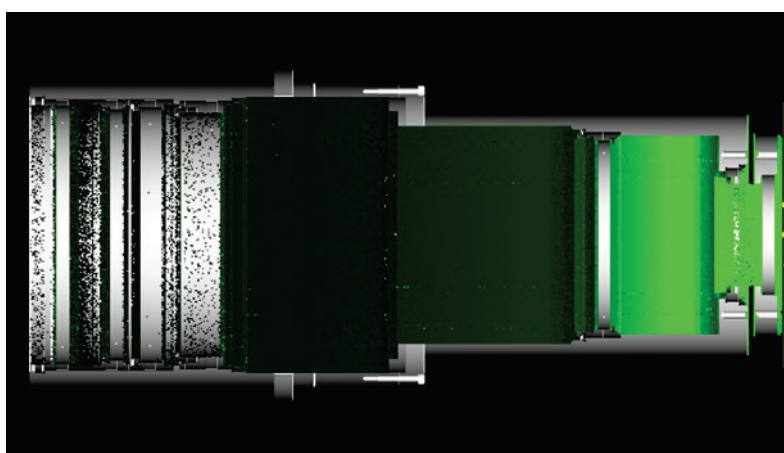


Figure 9: The distribution of image illumination over the surfaces of an optical device

At first, the lens system is visualized using the virtual prototyping software built-in functions. Clipping planes were used during rendering, so it is possible to see the cross-section of the objective in the picture. The white part of the image represents the construction of the lens. Next, rays are launched from the camera pixels into the scene and the first intersection of the ray with a non-empty cell of the three-dimensional grid is checked. After that, depending on the value in the cell, the color of the pixel changes to green (dark green for a small cell value, bright green for a large cell value). Thus, the illumination of the image, accumulated at the camera sensor is visualized on the scene elements and is represented as green cells superimposed on the image of the optical device scheme. The greater the value of the density and brightness of the cells the stronger the effect on the amount of stray light illumination at the illumination receiver from the element. As it was mentioned above, this apparent illumination is not tied to the direction of observation, and therefore it can be calculated once and then viewed from different positions, at different angles, and with different clipping, while choosing the best angle for parasitic illumination analysis, see Figure 10.

To test the proposed algorithm for correct operation, it was decided to use the software package, which allows physically correct modeling of light propagation. In this software, a simple scene was created, which is a two-lens lens. This is a projection lens with a simplified lens structure, which consists of protective glass, two glued lenses, a diaphragm, and housing elements. The lenses have a transmittance of 99 percent and are assumed to be coated with an anti-reflective coating. Also, all structural elements have the property of a diffuse BRDF (10 percent of reflection of the "Ward" type). The lens layout is shown in Figure 7 (right). The goal of the experiment is to show that it is possible to find the source of the stray light in the lens system and the stray light can be eliminated when its source is properly eliminated.



Figure 10: Distribution of image illumination over the surfaces of an optical device at a different camera position

Testing took place as follows. First, the propagation of light in the scene was simulated without the parasitic out-of-field illumination (Figure 11, left). Since the diaphragm has a 10 percent reflection of the "Ward" type when the rays emitted from a light source placed outside of the field of view are scattered on the diaphragm they create a parasitic illumination, which leads to an overexposure of the image and reduces the image contrast (Figure 11, middle). The calculation time was 10 minutes for each case at the workstation with two AMD EPYC 7281 16-core 2.10 GHz processors and 256GB RAM.



Figure 11: Result without stray light (left); result with stray light (middle); the resulting image at an aperture reflectance of 0.01 (right)

Further, using the developed methods, a caustic illumination map was formed and visualized on the image of a computer model of the lens. In the image, the area that can be a source of stray light illumination becomes visible. Figure 12 shows this area.

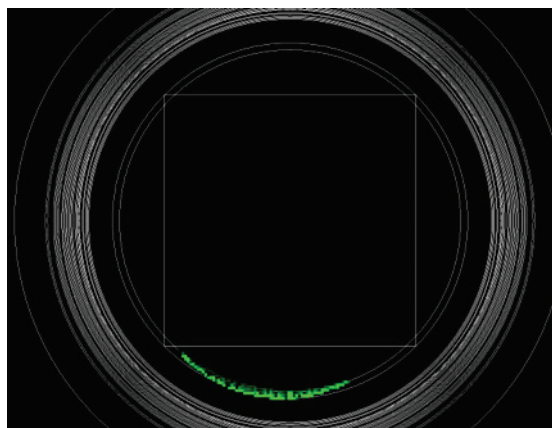


Figure 12: The area that forms the stray light

In the image, the area on the diaphragm that creates stray light is highlighted in green. Thus, it can be understood that because the diaphragm reflects light from a light source outside of the field of view onto the illumination receiver, so parasitic illumination is formed on it.

To remove it, it is necessary to change the optical properties of the diaphragm. Instead of 10 percent Ward reflectivity, a near-zero value (1 percent) was set to eliminate stray light reflection. Then simulation with new optical properties was carried out. As it can be seen from the simulation results, the stray light illumination at the illumination receiver has almost disappeared (Figure 11, right), and the image becomes close to the initial image without stray light (Figure 11, left) Thus it proves that the source of the stray light illumination was properly found and corrected.

These image flare calculation methods were implemented in progressive bidirectional ray tracing algorithms with backward photon mapping and integrated into the physically correct virtual prototyping system, where they are effectively used to analyze scattered light in scenes, containing many sources of caustic illumination.

7. Conclusion

Existing computer graphics methods provide physically correct and efficient solutions to the problems of computational optics and problems associated with modeling complex scenes containing many objects, complex physical phenomena, e.g., a large number of caustics and interreflections, etc. In this work, a new solution was proposed that allows analyzing and modeling scattered light in optical devices that form an image. The developed method of bidirectional stochastic ray tracing with backward photon mapping makes it possible to effectively calculate the stray light illumination appearing due to the indirect and caustic scattering on the structural elements of an optical device and the scattering surfaces of optical elements, as well as to visualize places of the optical device elements which are sources of the apparent illumination. This allows the optical system designed to understand the cause of the appearance of stray light illumination and eliminate it, which is necessary during the optical devices design process. At the same time, the method has several disadvantages: it consumes a lot of memory and has a slow visualization. In further work, it is supposed to overcome these problems by using hash tables and more efficient parallelization of the visualization process.

8. Acknowledgements

This work was supported by the Russian Science Foundation, project no. 22-11-00145.

9. References

- [1] J. T. Kajiya, The rendering equation, in: In Proceedings of the 13th annual conference on Computer graphics and interactive techniques, volume 20 of SIGGRAPH '86, ACM SIGGRAPH Computer Graphics, 1986, pp. 143—150. doi:10.1145/15886.15902.
- [2] R. L. Cook, K. E. Torrance, A reflectance model for computer graphics, *ACM Trans. Graph.* 1 (1982) 7–24. URL: <https://doi.org/10.1145/357290.357293>. doi:10.1145/357290.357293.
- [3] J. Hughes, A. van Dam, M. McGuire, D. Sklar, J. Foley, S. Feiner, K. Akeley, *Computer Graphics: Principles and Practice*. Third Edition., 2013.
- [4] E. Veach, *Robust Monte Carlo Methods for Light Transport Simulation*, Ph.D. thesis, Stanford, CA, USA, 1998. AAI9837162.
- [5] S. Chattopadhyay, A. Fujimoto, Bi-directional ray tracing, in: T. L. Kunii (Ed.), *Computer Graphics 1987*, Springer Japan, Tokyo, 1987, pp. 335–343.
- [6] B. Bitterli, W. Jakob, J. Novák, W. Jarosz, Reversible jump metropolis light transport using inverse mappings, 2017. arXiv:1704.06835.
- [7] A. Gruson, R. West, T. Hachisuka, Stratified Markov Chain Monte Carlo Light Transport, *Computer Graphics Forum* (2020). doi:10.1111/cgf.13935.
- [8] H. W. Jensen, Global illumination using photon maps, in: X. Pueyo, P. Schröder (Eds.), *Rendering Techniques '96*, Springer Vienna, Vienna, 1996, pp. 21–30.
- [9] V. Havran, R. Herzog, H.-P. Seidel, Fast final gathering via reverse photon mapping, *Computer Graphics Forum* 24 (2005) 323–332. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2005.00857.x>. doi:<https://doi.org/10.1111/j.1467-8659.2005.00857.x>. arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2005.00857.x>.
- [10] D. D. Zhdanov, I. S. Potemin, A. D. Zhdanov, A. G. Voloboy, Use of computer graphics methods for efficient stray light analysis in optical design, in: L. Mazuray, R. Wartmann, A. P. Wood (Eds.), *Optical Design and Engineering VII*, volume 10690, International Society for Optics and Photonics, SPIE, 2018, pp. 523 – 534. URL: <https://doi.org/10.1117/12.2312429>. doi:10.1117/12.2312429.
- [11] E. Frest, *Realistic image synthesis using photon mapping*, A K Peters / CRC Press, 2001.
- [12] J. L. Bentley, J. H. Friedman, Data structures for range searching, *ACM Comput. Surv.* 11 (1979) 397–409. URL: <https://doi.org/10.1145/356789.356797>. doi:10.1145/356789.356797.
- [13] T. Theoharis, G. Papaioannou, E. Karabassi, The magic of the z-buffer: A survey, in: *The 9-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2001*, WSCG 2001, University of West Bohemia, Campus Bory, Plzen- Bory, Czech Republic, February 5-9, 2001, 2001, pp. 379–386. URL: http://wscg.zcu.cz/wscg2001/Papers_2001/R12.pdf.
- [14] A. Zingl, *A rasterizing algorithm for drawing curves*, 2012.