The Development of Ray Tracing and Its Future

Zijin Wang

Denison University, Ohio, USA
wang_c2@denison.edu

Abstract
Ray tracing algorithm is a computer 3D graphics rendering algorithm based on real light path simulation. Compared with most other rendering algorithms, the ray tracing algorithm can provide a more realistic light and shadow effect. This algorithm was initially proposed by Appel in 1968, and improved into a recursive algorithm, and proposed a global illumination model by Whitted in 1980. To this day, ray tracing algorithms are still a hot topic in graphics, and many improvements are being made. Based on the study of natural light paths, ray tracing uses the backward calculation of light paths to restore true colors. The tracing process covers the reflection, refraction, absorption, and other properties of light (accurate calculations), supplemented by other important rendering ideas (further simulation).

Keywords
Ray Tracing

1. Introduction

Ray tracing is an important concept in computer graphics. To achieve a better explanation of ray tracing, we might want to first compare it to another concept called rasterization. Rasterization is a basic technique in rendering. A complicated rendering task could be separated into different sub-tasks by different objects. Then we separate each object into different polygons and project them onto the screen. Finally, we render each pixel that covered by polygons. This process is called rasterization. Compared to that, ray tracing is a different technique that shoots rays from the camera and calculates the light based on the objects they meet.

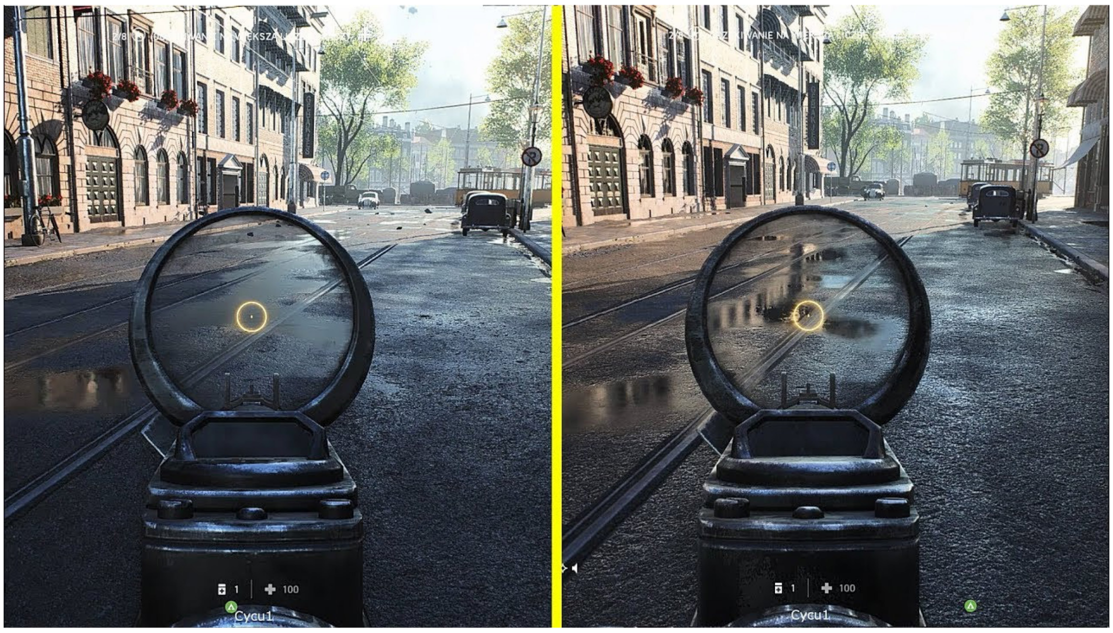

Figure 1: Comparison between Ray tracing on and off

The Fig. 1 was the comparison when ray tracing effect is off and on. Notice that the scene contains more details and the water surface reflects more details when ray tracing effect is on.

2. History

2.1. Ray casting

The first Ray casting algorithm that used for rendering was first introduced by Arthur Appel in 1968. Ray casting renders the scene by emitting a ray from the observation point to each pixel and finding the closest object that blocks the light path in the world scene. Only two rays were involved in ray casting. The first emitted by the eye to find the intersection point, the other was sent from the intersection point to the light, to see if the ray itself is in the shadow. The Fig. 2 provides an intuition of the algorithm.

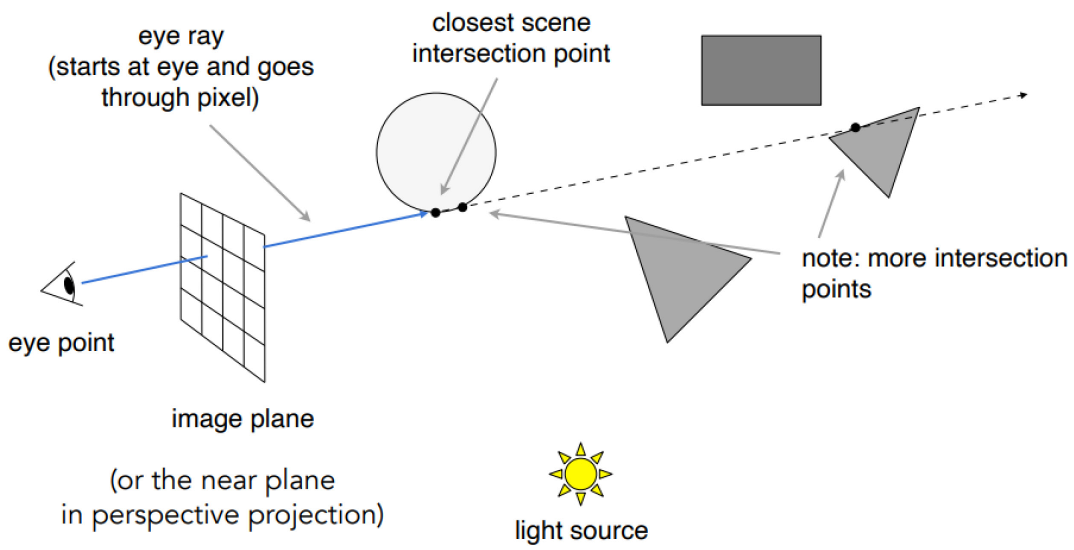

Figure 2: Intuition of Ray casting

2.2. Whitted Ray

This is a similar algorithm that was introduced by Turner Whitted in 1979. He made a progress based on traditional ray casting. He posts four kinds of rays in the scene. The eye ray was cast from the eye which is similar to that in ray casting. Then we have a reflection ray that reflects towards the surface mirror reflection direction. The third ray is called a refraction ray which could go into the object from an angle. The final ray is a shadow ray that is calculated by the shadow produced by the light source and object. This article describes a shading model that calculates intensities using global data. Extensions to a ray tracing visible surface algorithm are then given in order to enable this shader. Classical ray optics provides a basic model for light reflection from completely smooth surfaces. For example, in Fig.1, the light intensity, I, composed mostly by specular reflection, S, transmission, T, show viewer from a single point from surface.

Algorithm 1 Whitted Algorithm
```
for each pixel P do
      Create ray R through P
      Make NT to Infinity
      Make NP to NULL
      Color C = TraceRay(R,SceneS, NT)
      Shade P with C
end for
for every primitive in S do
      if ray intersects this primitive and if intersection point
t is less than NT then
```

```
          Set NT to t of the intersection
          Set NP to this primitive
          end if
     end for

     if NP is NULL then
          output background color
     else
          C is black
          make ray to each light and check if there's shadow
          if surface is reflective, make a reflection ray R1 then
               C+ = TraceRay(R1, S NT)
          end if
          if surface is transparent, make a reflection ray R2 then
               C+ = TraceRay(R2, S NT)
          end if
          C+ = Shading(NP, NT)
          output C
     end if
```

Taking into account that the surfaces shown are not always flawlessly glossy, a word is needed to model the diffuse component as well. It would be ideal to include components due to reflections from surrounding objects and sources of light, but the amount of processing needed to represent a spread light source is prohibitive.

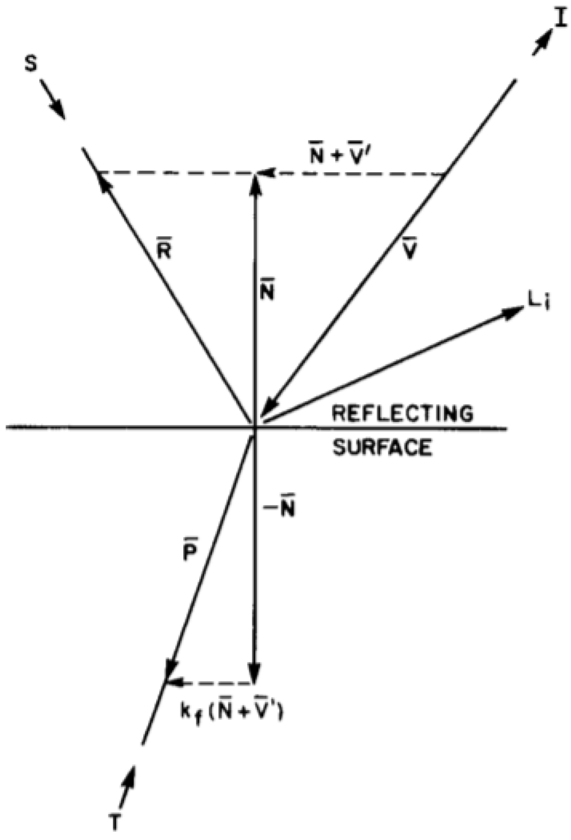

Figure 3: Reflecting surface

In order to achieve a better comparison, the Fig. 4 shows whitted ray tracing algorithm.

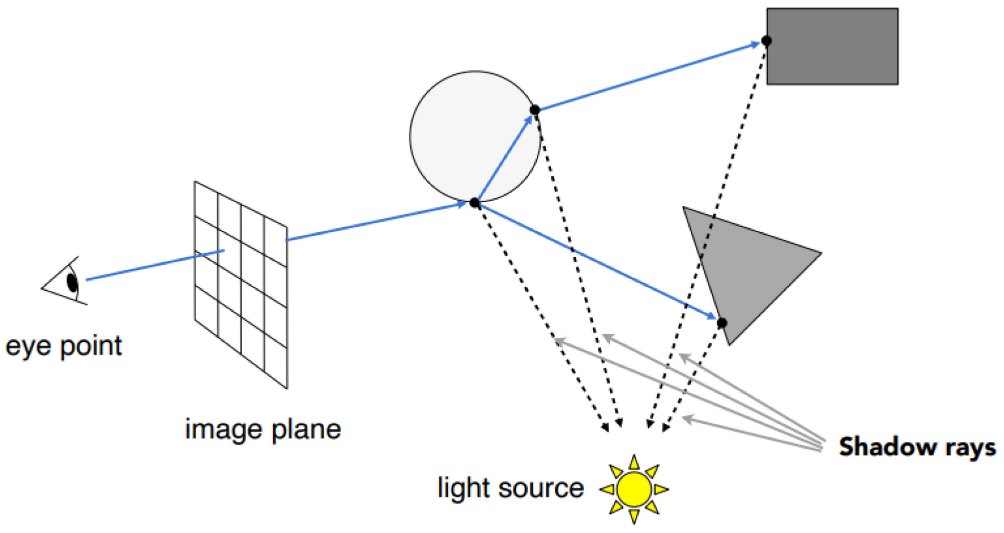

Figure 4: Intuition of Whitted Ray casting

2.3. Directions of ray tracing

Forward ray tracing follows the rule that photons from the source to the object. Even though forward lighting is the most accurate way to determine the color of each object, it is very inefficient. This is because a lot of light from a light source shall never pass through the plane of the eye(camera) and enters the eye(camera). Tracking every light that comes from a light source means that a lot of light will be wasted because it's never seen by the eye(camera). Here's the graph of forward ray tracing.

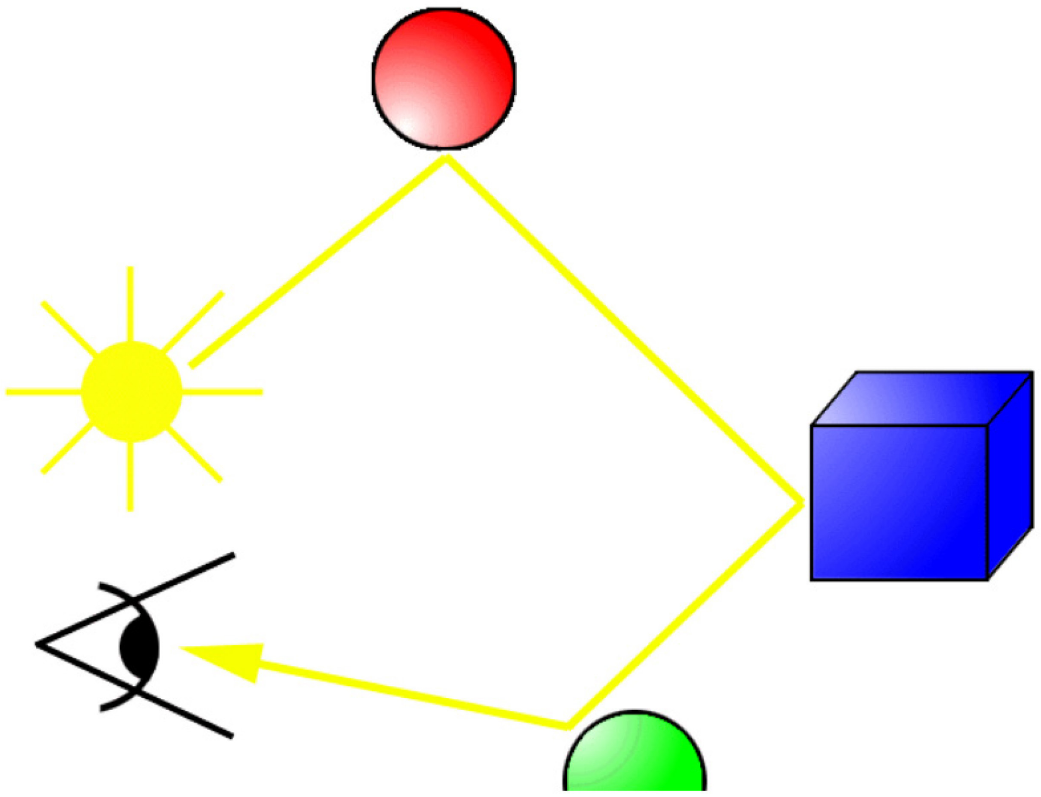

Figure 5: Forward Ray Tracing

To make ray tracing more effective, a backward tracing method was introduced. In this method, a ray was produced from the camera. It comes from the camera and enters the visual world. The first object that the ray hits is the object that is visible from that point in the plane of view. The disadvantage of backward tracing is that it assumes that only light that passes through the plane of view and into the eye contributes to the final image. In some cases, this assumption is flawed. As mentioned above, one of the most efficient and easiest ways to achieve performance optimization is to emit light backward from the eye, rather than from the light emission line of the source. In this way, no computing power is wasted trying to get light that never hits the model or the camera.

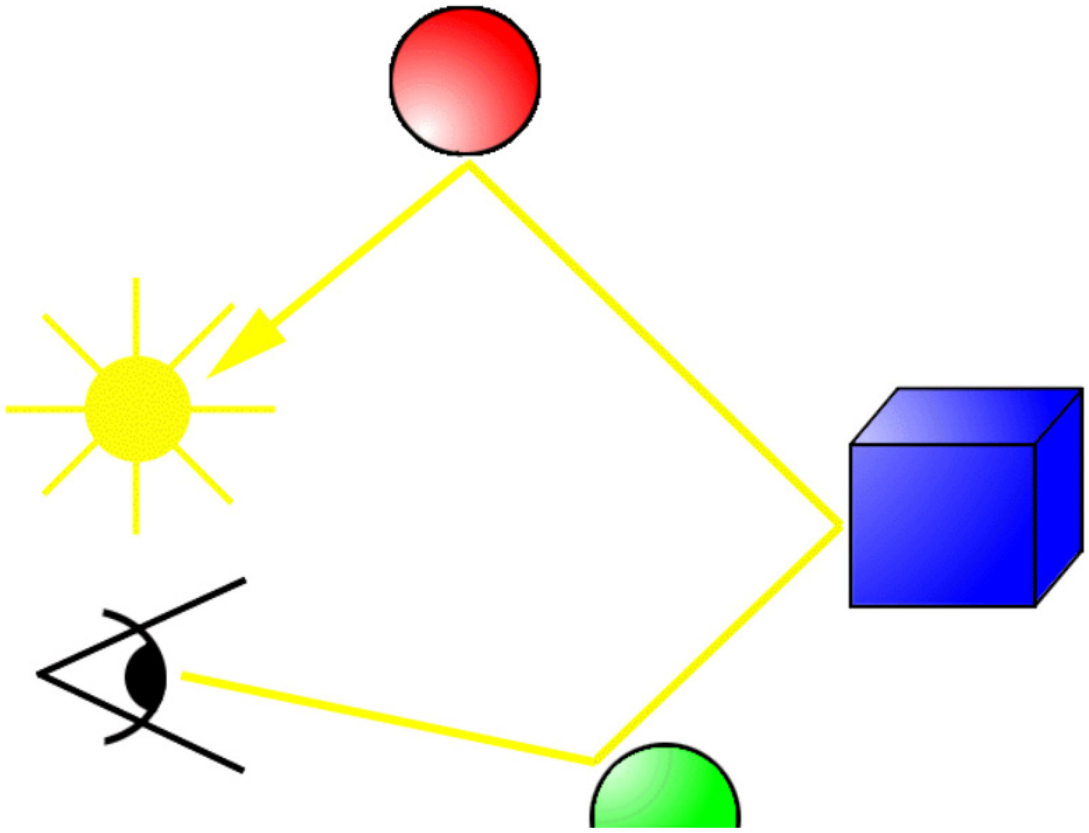

Figure 6: Backward Ray Tracing

Due to the disadvantages of both forward and backward ray tracing, recent studies have attempted to develop hybrid solutions that affect speed and accuracy. In these hybrid solutions, only certain levels of forwarding rays are performed. The algorithm records the data and then continues to perform backward ray tracing. The final coloring of the scene takes into account both backward and forward lighting calculations.

3. Further research on ray tracing

3.1. Acceleration structure

Whitted ray tracing was a considerable step in ray tracing. However, in terms of time complexity, there are still problems. Consider that since each ray need to connect to each polygon to make a judgment, then the complexity will be *number of pixels*number of polygons*reflection times.*

So consider that our goal is to find the closest triangle that the ray passes through.Although there are massive triangles on the screen, we only need to detect all triangles that are traversed by one ray at a time when ignoring the "recent" condition. Unless all of the triangles are in a neat row, there is no need to examine them all. In other words, most of the triangles on the screen can be ignored according to their position in space. By using this method, we can greatly reduce the time complexity. We will talk about two common structures here.

The first one was called the k-d tree. k-d Tree can be regarded as a special case of BSP tree, which is a tree structure extended to multidimensional space. BSP tree is a kind of spatial segmentation technology, which has been applied in many fields. It was introduced into various research fields and applications of computer graphics in the 1990s. The principle is to take the whole scene as a tree and divide the current tree into two Spaces to get two subtrees by dividing the plane. These two subtrees are then divided by their respective segmentation planes to get smaller subtrees until the depth of the tree reaches a predetermined threshold or the number of scene facets contained in the node is less than the predetermined threshold. Each node of the tree represents a subspace, including all facets contained in the represented space, and the root node of the tree represents the entire landscape space. The Fig. 7 gives a basic intuition about this method.

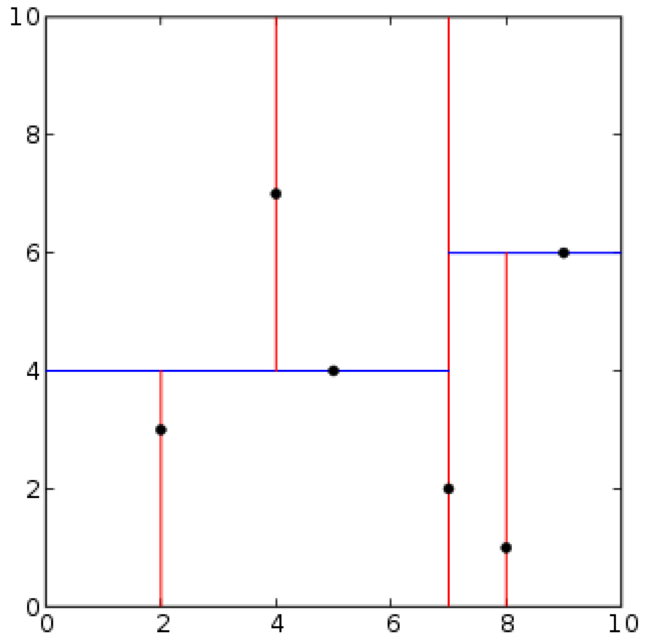

Figure 7: Intuition of K-D Tree

The bounding box hierarchy is the most representative of the hierarchical partitioning methods. The basic idea of the hierarchical bounding box algorithm is as follows: surround the faces in the scene with simple bounding boxes (such as spheres, cuboids, etc.). The adjacent bounding boxes are contained in a larger bounding box, which is expanded step by step to generate a hierarchical structure. Before the intersecting test of the light and the object, the test of the light and the bounding box should be carried out first. If the light intersects the bounding box, the intersecting test of the object surface it contains should be carried out to improve efficiency.By organizing the bounding box according to the effective hierarchical structure, the number of intersecting tests can be reduced, the complexity can be reduced, and the efficiency can be further improved.In the selection of the bounding box, on the one hand, the bounding box with a simple shape should be selected to reduce the cost of the intersection test between the light and the bounding box.The AABBs of nodes at the same level of a BVH may overlap, which is a fundamental distinction between it and a kd-tree. A basic bottom-up method was used in the early BVH creation algorithms. A firstlevel clustering builds a collection of leaf nodes from primitives. The nodes are then grouped together level by level to build a hierarchy.

On the other hand, the bounding box that can closely surround the surface of the object should be selected to improve the effectiveness of intersecting test between the light and the bounding box.In practice, the most commonly used is an Axis Aligned Bounding Box (AABB).AABB axisymmetric bounding box is simple, easy to store, easy to calculate, good robustness, is a good compromise between the simplicity of intersection test and the compactness of the surrounding objects, high efficiency.Compared with the partitioning method based on spatial segmentation, the hierarchical partitioning method has more advantages in dynamic scenes.Because in a dynamic scene, the organizational structure of the scene has to change and every frame has to be rebuilt, which is undoubtedly a huge overhead, resulting in a surge in the amount of computation.In a dynamic scenario, the data structure formed by the hierarchical partitioning method can update only the relevant information about the bounding box (such as its position and size).That is to say, BVH only needs to refresh the data structure, but does not need to rebuild, so there is a significant improvement inefficiency.

Algorithm BVH Construction Algorithm
```
procedure buildBVH  (node N, primitive set S)
      if S is small enough then
            N be leaf node
            assign S to n
            compute the bounding box of N
      end if
      make variable bestCut infinity
      Cut S into groups evenly along axes
      Split S into S+ and S- according to bestCut
      buildBVH(new node left, S+)
      buildBVH(new node right, S)
end procedure
```

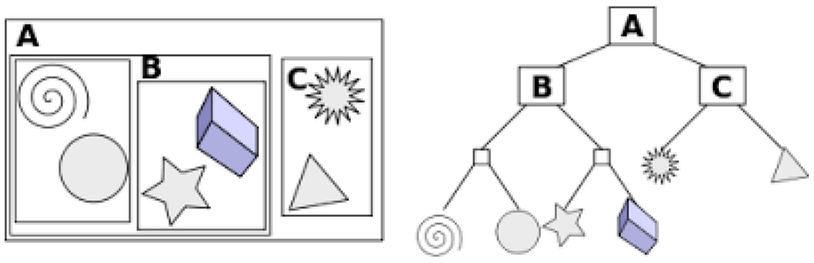

Figure 8: BVH

3.2. sawtooth

The appearance of sawtooth was from regular sampling. Since the ray tracing algorithm, for each pixel, creates only one ray, samples only one point in the scene and color. But, for a pixel, it's likely to contain a lot of different points, especially in the case of dealing with object edges. Thus, the different colors in one pixel lead to this situation.

Monte Carlo sampling is one of the effective attempts to solve this problem. The program starts by emitting a fixed number of lights and comparing their colors.If the colors are similar, the program assumes that the pixels are looking at the same object and then the average light is calculated as the color of the pixel.If the light color is different, the difference was determined by a certain threshold. Then we think this pixel is special and needs to be examined further.In this case, the pixel is subdivided into smaller areas, and each new area is treated as a complete pixel.The process begins again, and the same pattern will loop.

3.3. Distributed ray tracing

Now we finally look into the modern ray tracing method that uses Monte Carlo Method.Distributed ray tracing is a ray-tracing method based on randomly distributed sampling to reduce artifacts in rendered images. Cook once thought that the process of rendering using ray tracing is indeed the process of solving a nested integral. However, it couldn't be solved in finite running time. Thus, Monte Carlo is a common method to deal with this sort of problem. Distributed ray tracing has several properties. It uses jittered sampling and uses noise to supplement the situation of distortion. Some special effects were also offered by the algorithm.

3.4. Dithering

There are many types of dithering techniques. This section mainly introduces the dithering technique of the specification grid. This technique can produce good experimental results and is very suitable for image rendering algorithms. The specific principle is to divide each pixel and add a random offset to the central area of each block to ensure that the offset is in the same block of pixels. Of course, this sampling method can also be used in the sampling of area light.

Dithering can reduce the high-frequency signal, but the energy in the reduced high-frequency signal will appear in the noise and will not disappear, so the basic spectrum combination has not changed. Compared with the pure-bred Poisson disc distribution technique, this technique may cause more noise and may leave some jaggies.

An interesting side effect of distributed ray tracing random sampling modes is that they inject noise into the solution (slightly coarser images). This kind of noise is more acceptable than a distorted image.

Think of a pixel as a grid or a large grid composed of multiple subpixel grids, which is a two-dimensional jitter. Noise is randomly added to the position in the X-direction or the position in the Y direction. The X and Y directions are independent of each other, which is equivalent to two one-dimensional jitters. It is required that each sampling point occurs in a certain pixel grid range. At random locations within. If it is known which sampling points are visible, the values of those sampling points are processed through the reconstruction filter.

The implementation method of the reconstruction filter is an open question. The simplest reconstruction filter is the Box Filter: take the average of multiple sampling points. A weighted reconstruction filter can also be used. In this case, the filter is a weighted value related to the surrounding pixels at the sampling position. Each pixel is the sum of the value of the nearby sampling points multiplied by the weighted value. These filters can be calculated in advance and stored in a lookup table.

4. Conclusion

Ray tracing algorithms have been developed for decades. From the early ray casting algorithm to nowadays newborn methods, this algorithm becomes more efficient and useful. It's not yet the time that ray tracing is widely used. However, ray tracing will be a more important role in computer graphics.

5. References

[1] “Ray Tracing (Graphics).” Wikipedia, Wikimedia Foundation, 13 June 2021, en.wikipedia.org/wiki/Ray tracing (graphics)

[2] Haines, Eric, and Akenine-Moller Tomas. *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs* . Apress, 2019.

[3] “Ray Tracing Essentials Part 1: Basics of Ray Tracing.” NVIDIA Developer Blog , 18 Jan. 2020, developer.nvidia.com/blog/ray-tracing-essentials-part-1-basics-of-ray-tracing/.

[4] Scratchapixel. *Introduction to Ray Tracing: a Simple Method for Creating 3D Images (Raytracing Algorithm in a Nutshell) ,* 30 Oct. 2014, www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/raytracing-algorithm-in-a-nutshell.

[5] Scratchapixel. Introduction to Ray Tracing: a Simple Method for Creating 3D Images (Implementing the Raytracing Algorithm), 30 Oct. 2014, www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/implementing-the-raytracing-algorithm.

[6] Tomas Nikodym (June 2010). "Ray Tracing Algorithm For Interactive Applications" . Czech Technical University, FEE.

[7] Whitted, T. (1979). "An Improved Illumination Model for Shaded Display". Proceedings of the 6th annual conference on Computer graphics and interactive techniques. CiteSeerX 10.1.1.156.1534. ISBN 0-89791-004-4.

[8] Eric P. Lafortune and Yves D. Willems (December 1993). "Bi-Directional Path Tracing". Proceedings of Compugraphics '93: 145–153.
[9] Georg Rainer Hofmann (1990). "Who invented ray tracing?". The Visual Computer. 6 (3): 120–124.
[10] “Intro of Modern Computer Graphics.” Cnblogs, www.cnblogs.com/czy-stu/p/14445740.html.