International Journal of Advance Engineering and Research Development (IJAERD)
Special Issue, Volume 1,Issue 4, April 2014, e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

Reducing Computational Time in Ray Tracing

Darshit Patoliya[1],Bhargav Ghodasara[2], Vatsal Shah[3]

[1]*Information and Technology Department, BVM Engineering College, darshit7@gmail.com*
[2]*Information and Technology Department, BVM Engineering College,b.ghodasara1994@gmail.com*
[3]*Assist. Prof., Information and Technology Department, BVM Engg. College, vatsal.shah@bvmengineering.ac.in*

Abstract—Many computer graphics rendering algorithms use ray tracing to generate realistic and high quality image. It is a method to convert 3D images in to the high quality 2D realistic image. In original ray tracing algorithm we need millions of rays to produce the render image, by calculating intersection of ray with image plane and characteristic of surface, color of pixel is determined. So this process is time consuming and complex. In this paper we study some algorithms which reduce the computational time of ray tracing.

Keywords- Ray Tracing, kd-tree, Spatial Median Method

I. INTRODUCTION

Everyone in the field of the computer graphics is familiar with the ray tracing algorithm. Because of the simplicity of the algorithm it is easy to implement and understand it. Ray tracing algorithm is discrete sampling of visibility, carried out by a ray shooting algorithm. Huge number of ray shooting queries (≈106–109) are performed in order to compute one image of a commonly used resolution such as 640*480[1]. The more compatible ray tracing algorithm was first introduced by Whitted in 1979[2].

Ray tracing is basically a technique in which image is generated by path of the light passing through pixel in image plane. Ray tracing technique generates image with high power of quality at very high computational cost. So where time is critical than quality of image, ray tracing algorithm is used to render image such as Television, animated films and video games.

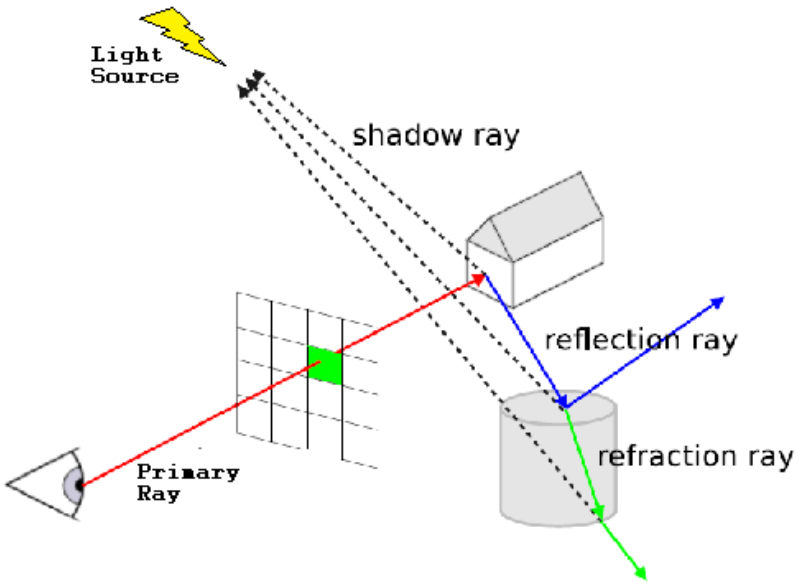

Figure 1.Image of sphere is createdon image plane by ray tracing rays [3]

In ray tracing basically all rays are tested for intersection with some subset of all models in the scene. When nearest object is identified in the scene based on the properties of the object, color of pixel determined. Lot of research has been done for the last couple of decades in order to increase the performance of the ray tracing algorithm. Making interactive images in many applications such as video games is more significant than its accuracy and quality. Ray tracing needs traversing whole image, so choosing an effective algorithm is the key factor to improve computation time in ray tracing. There are many method proposed such as kd-tree, SAH, BVH [4]. But one method which received lot of attention from researchers is kd-tree, because of its good spatial adaptability and high efficiency.

1.1 Spatial Media Method

Kd-tree is binary acceleration method in which plane is divided into two recursive parts. This division includes two cut planes on Cartesian axes. Spatial method simply makes faster traversing of kd-tree. See [5] recursive kd-tree algorithm introduce by Wald and Havran.

Algorithm: recursive kd-tree construction

```
function Partition(triangles T, voxel V) return node
    if Terminate(T,V) then
    return  new leaf node(T)
    p=FindPlane(T;V) /*find a plane p to split V*/
    (VL,VR)=Split V with p
    TL={t∈  T | (t∩ VL) ≠∅  }
    TR={t ∈  T | (t ∩VR) ≠∅  }
    return new node(p, Partition (TL;VL), Partition(TR;VR))
    function BuildKDTREE(triangles[] T) returns root node
    V = AABB(T) /* start with the scene*/
        return Build(T;V)
```

II. PRIVIOUS INRODUCED METHOS

There are many methods introduced for reducing the ray tracing time. See [3] by Ali and Shahin for some of the method described here based on the pixel.

2.1 Pixel Differential Method

In this method number of traced ray are decreased by using the differential relationship of pixel. The problem with conventional ray tracing technique is complex computation. The most of the render time is used to find out intersection of ray and objects in scene. So if we want to increase the speed of algorithm, intersection point should be decreased. Another method is reducing the number of rays where, the rays reduced are more compared to conventional method. Pixel of odd rows are computed as ray tracing method but even rows pixel are derived from the average of the upper and lower rows pixel color.

If the difference of upper and lower pixel color is small then, it is derived by taking their average, but if difference is greater, ray tracing technique is used. If we have 28 values for red, green and blue (RGB images), then color difference should be minimum 0 and maximum 256. Let K be a parameter as a difference of color between upper and lower rows. If K is close to zero then time consume is less than the ray tracing method. Therefore created image have high quality. Otherwise if

K is close to the 256 then averaging method can be implemented more than ray tracing method. So the image quality will be very low.

Algorithm 2: Pixel Differential Method [3]

```
Procedure Render()
{
        FOR(each pixel in row)
        {
                If(row is odd )
                        ColorPixel = RayTrace(ray)
                //Send ray from origin through pixel , trace ray and intersect with scene
                Else if (row is even)
                {
                        If(|ColorPixel.up – ColorPixel.down| < K)
                                ColorPixel=( ColorPixel.up – ColorPixel.down)/2
                        Else
                                ColorPixel =RayTrace(ray)
                //Send ray from origin through pixel , trace ray and intersect with scene
                }
        Show(colorpixel) //show pixels from image plane
        }
}
```

2.1 Pixel Averaging Method

This method combines both ray tracing and average of nearer pixels. So time is shorter than ray tracing method but quality of image is quite low. This algorithm is used when fast rendering is required but quality of image is reasonable. In this technique couples of rows are calculated through ray tracing method and row other than that is calculated using average method. Algorithm stores values of first two rows in two dimensional arrays. Third row is obtained by averaging two rows of the previous calculation.

Algorithm 3: Pixel Averaging Method [3]

```
Procedure Render( )
{
        FOR(each pixel in scene)
        {
                If(row is even)
                        ColorPixel = RayTrace(ray)
                //Send ray from origin through pixel , trace ray and intersect with scene
                Else if (row is odd)
                {
                        If((ColorPixel.up – ColorPixel.down) < K )
                                ColorPixel=((ColorPixel.up – ColorPixel.down)/2 &&(
                                arr[row-2][rgb]+arr[row-1][rgb])/2)
```



```
                    Else
                        ColorPixel = RayTrace(ray)
            //Send ray from origin through pixel , trace ray and intersect with scene

                }
                Show(colorpixel) //show pixels from image plane
            }
    }
```

III. PROPOSED METHOD

3.1 Advanced Pixel Averaging Method

In previous pixel differential method we calculate the difference of upper and lower pixel if difference is less than K then color of pixel is determined by those values and if value of K is near to 256 then ray tracing method is used to determine the color of pixel for each odd row. But in even row for each pixel we have to use ray tracing. In this method we introduce some technique to avoid using ray tracing for each pixel in even row. For even row first we calculate each and every even pixel using ray tracing and for determining the color of the odd pixel we use the difference of left and right pixel. This method increases the speed of algorithm but image quality is going to be low. So when rendering time is critical than quality of image then this method is used. For example video game.

After calculating every even pixel in even row using ray tracing for odd pixel we check the difference of left and right pixel. If difference K is near to 0 then using pixel differential method color of pixel determined but if K is near to 256 then color of the pixel determined by the ray tracing algorithm.

Algorithm 4: Advanced Pixel Differential Method

```
Procedure Render()
{
        For (each even row)
        {
                For(each even pixel)
                        ColorPixel[row][PixelNumber]=RayTrace(ray)
        }
        FOR (each pixel in row)
        {
        If (row is even)
        {
                If (Pixel is odd)
                {
                        If((ColorPixel[row][PixelNumber-1]-
                                ColorPixel[row][Pixelnumber])<K)


                        {
```



```
                                    ColorPixel=(ColorPixel[row][PixelNumber-1]-
                                    ColorPixel[row][Pixelnumber])/2
                            }
                    }
            }

            Else if (row is odd)
            {
                    If((ColorPixel.up – ColorPixel.down) < K)
                                    ColorPixel=( ColorPixel.up – ColorPixel.down)/2
                    Else
                                    ColorPixel =RayTrace(ray)
                    //Send ray from origin through pixel , trace ray and intersect with scene
            }
            Show(colorpixel) //show pixels from image plane
    }
```

A main advantage of this algorithm is less time complexity compared to ray tracing algorithm but at a high risk of the image quality.

IV. CONCLUSION

Ray tracing algorithm has a high computational cost, we reduce the cost of computation by decreasing the number of ray. In pixel averaging and differential method we calculate color of some pixel using ray tracing and for rest of the pixel color is determined by the difference of the neighboring pixel. In advance pixel averaging method, odd row even pixel is calculated using ray tracing and odd row odd pixel is calculated by average of left and right pixel.

REFERENCE

[1] V.Havran and J.Bittner, "On improving kdtree for any shouting", the WSSCG'2002 Conference, pages 209-216, 2002.
[2] Whitted T., "An improved illumination model for shaded display". Proceeding of the 6th annual conference on Computer graphics and interactive techniques 1979.
[3] Ali Asghar, Shahin Pourbahrami, "Reducing Render Time In Ray Tracing By Pixel Averaging", "International Journal of Computer Graphics & Animation", Vol.2, No: 4, October 2012.
[4] A Glassner, "An Introduction to Ray Tracing", San Francisco, USA: Mogran Kaufmann, 1989.
[5] I. Wald and V. Harvran, "On building fast kd-tree for tracing, and on doing that in O(N log N)", In proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, Sept. 2006
.

