

REDUCING RENDER TIME IN RAY TRACING BY PIXEL AVERAGING

Ali Asghar Behmanesh¹, Shahin pourbahrami², Behrouz Gholizadeh³

¹Computer Department, Avecina University, Hamedan-Iran
aa.behmanesh@gmail.com

²Computer Department, Al Zahra University, Tehran-Iran
Sh_porbahrami@yahoo.com

³Computer Department, Al Zahra University, Tehran-Iran
gholizad@alzahra.ac.ir

ABSTRACT

Many computer graphics rendering algorithms and techniques use ray tracing for generating natural and photo-realistic images. Ray tracing is a method to convert 3D-models into 2D-high quality images by complex computation. The millions of rays must be simulated and traced to create realistic image. A method for reducing render time is acceleration structures. The kd-tree is the most commonly used in accelerating ray tracing algorithms. This paper has focused on reducing render time. We propose two methods that are combined with ray tracing method for obtaining some pixel colour of image plane. Our methods are linear algorithm and are faster than ray tracing method. Our results show that our methods are at least 50% faster than spatial median approach for reasonably complex scenes with around 70k polygons and about 0.2% quality degradation. In this article, we show that these proposal methods can be combined with other ray tracing methods such as SAH to reduce render time.

KEYWORDS

Ray tracing, Ray shooting, kd-tree, structures acceleration

1. INTRODUCTION

Almost everyone in the field of 3D computer graphics is familiar with ray tracing that is a very popular rendering method for generating and synthesizing photo realistic images. The simplicity of the algorithm makes it very attractive. Many global illumination algorithms such as ray tracing use a discrete sampling of visibility, carried out by a ray shooting algorithm. Huge number of ray shooting queries ($\approx 10^6 - 10^9$) are performed in order to compute one image of a commonly used resolution such as 640*480[1]. Ray tracing algorithm was introduced by Whitted in 1979 for the first time. This algorithm has more capability and ability than Rasterization.[2]

In computer graphics, ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects. The technique is capable of producing a very high degree of visual realism, usually higher than that of typical scanline rendering methods, but at a greater computational cost. This makes ray tracing best suited for applications where the image can be rendered slowly ahead of time, such as in still images and film and television visual effects, and more poorly suited for real-time applications like video games where speed is critical. (Figure 1).

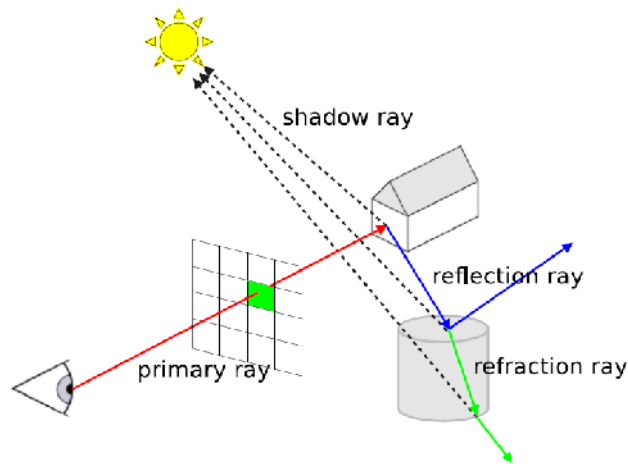


Figure 1. Image of a sphere is created on the image plane by tracing rays

Typically, each ray must be tested for intersection with some subset of all the objects in the scene. Once the nearest object has been identified, the algorithm will estimate the incoming light at the point of intersection, examine the material properties of the object, and combine this information to calculate the final colour of the pixel. It is focused on the kd-tree construction in this paper.

The ray shooting is a difficult problem in spite of its simple definition: given a ray find its nearest intersection with objects in the scene if such an object exists. However, it has very high computational demands and a lot of researches have been done for the last couple of decades in order to increase the performance of ray tracing algorithm. Since the time of developing of ray tracing scientists have been captivated to expedite the tracing of rays, based on combing kinds of approximation, other methods and technologies. Making interactive images in many applications is more significant than its accuracy or quality. Ray tracing is employed in the non-interactive applications.

A typical approach to reduce the running time is to use the acceleration constructions. These constructions are the same hierarchical constructions (e.g. queue) that make it possible to use liner algorithm (liner searching). Running time can be reduced by acceleration constructions from $O(n)$ to $O(\log(n))$. Ray tracing needs to traverse the whole scene space, so choosing and building efficient spatial presentation is the key factor to accelerate ray tracing. Many methods have been proposed, such as octree, kd-tree, BVH, SAH, and so on [3]. Though all these techniques have their advantages, kd-tree has received lots of attentions and has been widely used in large applications, for its good spatial adaptability and high efficiency. Next section describes ray tracing in brief.

1.1. THE SPATIAL MEDIAN METHOD

Kd-tree is a binary acceleration construction which divides the space into two separated subspaces. This division is recursive and includes two balanced cut plane along one of the Cartesian axes.

Cut planes are chosen arbitrary. This algorithm implies a global scene on the recursive construction kd-tree (Algorithm1). Selecting cut planes is done according to different methods:

1. Spatial median: The cut planes divide the space into two identical sub spaces.
2. Object median: The cut planes are located in the space, where the number of elements of the sub surfaces would be the same.
3. SAH (surface area heuristic): The cut planes are selected based on a cost function.

The median spatial method simply makes the tree fastest than two approaches, so it is used in this work. The tree is a binary which covers the confined elements of the space by its own nodes.

Algorithm 1: Recursive kd-tree construction

```

function PARTITION(triangles  $T$ , voxel  $V$ ) returns node
  if Terminate( $T; V$ ) then
    return new leaf node( $T$ )
   $p = \text{FindPlane}(T; V)$  /* find a plane  $p$  to split  $V$  */
  ( $V_L, V_R$ ) = Split  $V$  with  $p$ 
   $T_L = \{t \in T \mid (t \cap V_L) \neq \emptyset\}$ 
   $T_R = \{t \in T \mid (t \cap V_R) \neq \emptyset\}$ 
  return new node( $p$ , Partition( $T_L; V_L$ ), Partition( $T_R; V_R$ ))
function BUILDKDTREE(triangles[]  $T$ ) returns root node
   $V = \text{AABB}(T)$  /* start with the full scene */
  return Build( $T; V$ )
    
```

The off springs of the tree, halve the space of ancestor nodes, into two identical sub space which have not any cross over. These sub spaces divide into smaller sub spaces to construct the tree.

The leaves of the tree include the objects of the scene which covered by the node space. For a detailed description on building kd-trees see [4] by Wald and Havran. Although they propose an $O(n \log n)$ algorithm – the theoretical lower bound for comparison-based sorting algorithms – they did not apply low-level optimization techniques or other approaches to speed-up the method.

1.2. PREVIOUS WORKS

A problem of ray tracing in last two decades is complex computation. The different methods like object –fast ray intersections, kd-tree, bounding volume hierarchies(BVH), octrees and different constructions of grid like uniform, non-uniform recursive and hierarchical [5, 6, 7] are introduced. One of the practical and well adaptive constructions in ray tracing is kd-tree. Wald [8] and Havran [5] showed that this kd-tree has better than others methods in results. Access to today hardware causes to get real time by an ordinary personal computer.

Such works have been done by Wald & Havran [9] in the field of acceleration constructions and new approaches rendering in kd-tree. Shevtson [10] has introduced an algorithm called min-max binning. This algorithm pierces the scene and prepares the cutting surface for evaluating the least cost function of the kd-tree consequently. Having precise numbers in both sides of cutting surface is difficult so getting a min value in cost function is for away.

Laaszlo [11] and Matt [12] proposed an algorithm based on complexity value of $O(n \log 2n)$ which was not enough fast as expected. To get more details on kd-tree construction and its algorithm is suitable to refer to [10]. They introduced an algorithm based on ordering algorithm using $O(n \log n)$. They did not used low level optimization methods or kinds of methods are used to accelerate. Havran [13] recently proposed a method based on hierarchical acceleration structures

in kd-tree. Moreover he introduced a SAH based on accelerate combined methods; using specification total 3D space will be formed. Hunt [14] also evaluated the kd-tree in fast construction issue; he anticipated the cost function of SAH by means of subsampling and piecewise function (in order two). The other experts marginally concentrated on constructing the kd-tree. Benthin [15] successfully implemented an expedite construction the kd-tree based on a SIMD construction and showed.

2. PROPOSAL METHODS

The most of render time is belonged to the intersection of ray and objects in scene, so the acceleration structures may increase the speed of rendering because the numbers of interactions are decreased. Decreasing the number of rays is another method to reduce the number of collisions. In this paper we propose two hybrid methods that combine averaging method with ray tracing method:

2.1. PIXEL DIFFERENTIAL METHOD

The First method that we proposed is hybrid method; we can reduce the number of traced rays. Pixels of odd rows are computed by ray tracing method (figure 2). The other pixels are obtained with the averaging pixel colour of the upper and lower if the upper and lower pixel colour difference is small; otherwise, the colour pixels are obtained by ray tracing method. If we have 28 values for red, green or blue for RGB colour images, the colour difference we used in the code would be minimum 0 and maximum 256. The colour difference between the upper and lower pixels is showed with the parameter K. If K is close to zero averaging method runs less than the ray tracing method. Therefore, created image will be better quality. Otherwise, if the K is closer to 256 then averaging method can be implemented more than the ray tracing method. So the resulting image quality will be very low. In this paper, because image quality is not very low the K value has been considered in 10, 20 and 30. This algorithm is composed of two parts: one ray tracing method and the other proposed method (Algorithm 2). In this method, the pixel odd rows are scanning through image plane and other colours of pixels obtained by the averaging method according to neighbouring pixels. Thus less than of half the pixels are obtained by averaging method that its computation complexity is linear and the calculation of this method is less than ray its tracing method.

Algorithm 2: **PIXEL DIFFERENTIAL METHOD**

```

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
  }
}

```

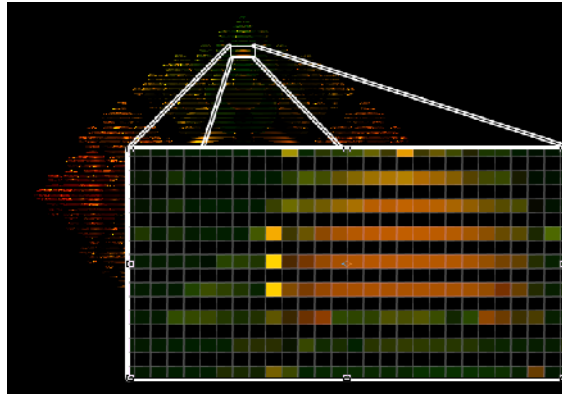


Figure 2. Obtained pixels in odd rows by ray tracing

2.2. PIXEL AVERAGING METHOD

In the second method, we refer to an algorithm that output image plane rendered combined with Ray tracing method and the average between neighbouring pixels ,Since that time is much shorter than the previous methods and Image quality has changed very little (about 0/2).

The algorithm is used for fast rendering on the image plane and finally the end is shown the final colour of the pixel and the ray tracing method and the parameters of the average colour difference between pixels. In this method a couple of rows through are calculated by the function Render Ray and rows that are not calculated by the function ray tracing are calculated using the average. The algorithm starts the first two rows obtained with function ray tracing and two rows are stored in a two-Dimensional array (The purpose of the definition array is contains two rows of pixel values stored in the array was already, A dimension to store the Length image and a dimension to store RGB) Third row, are obtained with averaging between two rows of the two previous row (figure 2).

Algorithm 3: PIXEL AVERAGING METHOD

```

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
    }
}
    
```

The remaining rows without colour are also detected with Averaging method and difference colour between the top and bottom rows of colour pixel by pixel rendering by function ray tracing and the final output is obtained which reduces the computing and time of slow ray tracing rendering method (Algorithm 3).

3. RESULT AND DISCUSSION

We experiment on system with Intel 2.20GHz core 2 Duo CPU T7500 3GB RAM and Windows 7. We implement all methods on different scenes that are rendered in 800*600 pixels. We execute the methods (spatial, Pixel differential and Pixel averaging methods) for five scenes (Figure 3). The numbers of triangles (primitives) that build each scene are showed in table2.

Table 2. Scene's objects

BUNNY	VENUS	ELEPHAM	ATENEAM	TEAPOT	SCENES
69451	43357	39292	15014	6220	Number of triangles

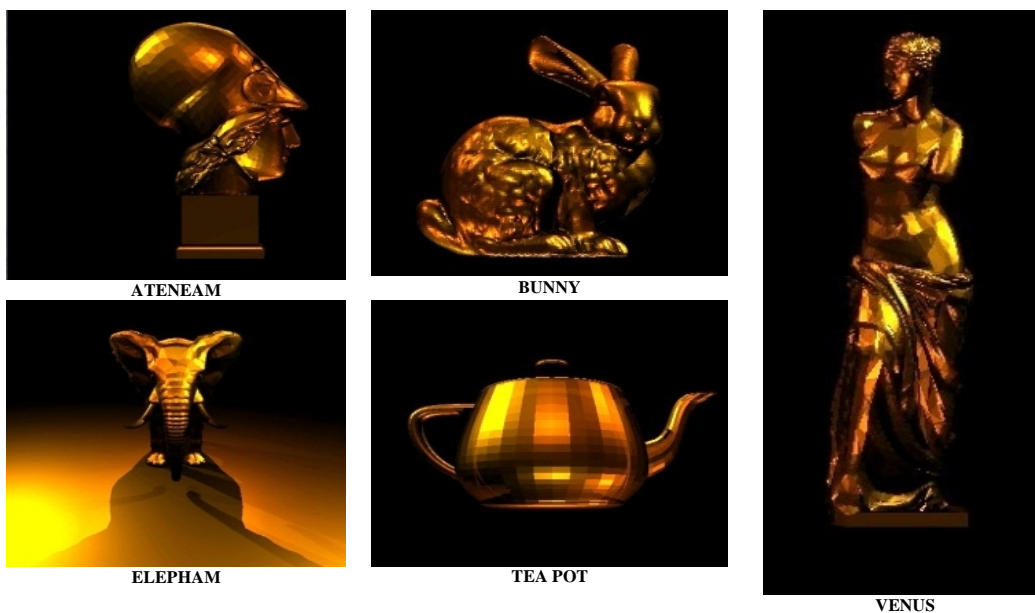


Figure 3. The different scenes(3D models) is used in this paper

Figure 4 shows the results of the spatial, Pixel differential and Pixel averaging methods for all scenes when $k = 10$.in this case, the Pixel averaging method is the best and the Pixel differential method is better than the spatial method.

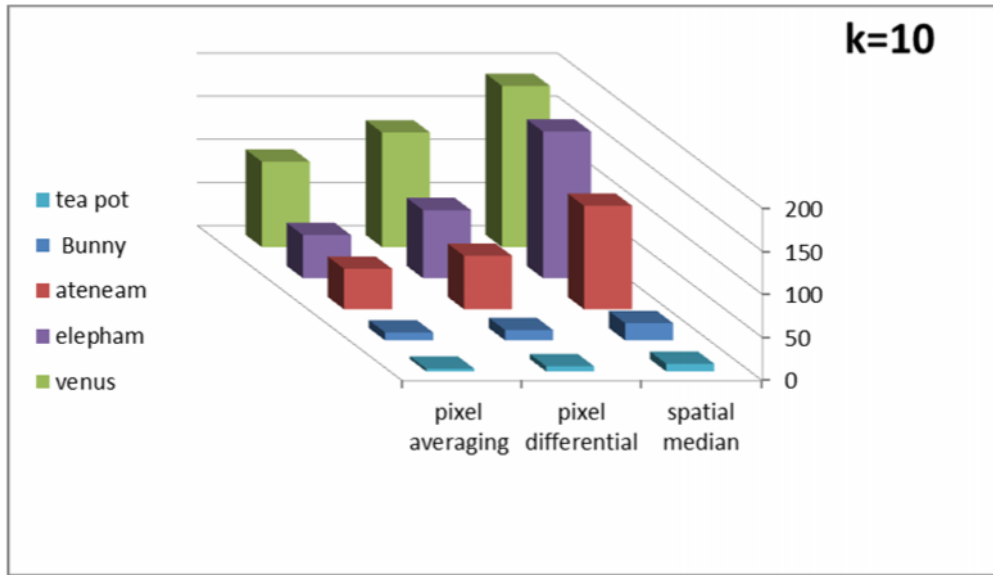


Figure 4. The Differentscenes when k=10

Figure 5 shows the results of the spatial, Pixel differential and Pixel averaging methods for all scenes if k = 20. Result in this case similar to previous case.

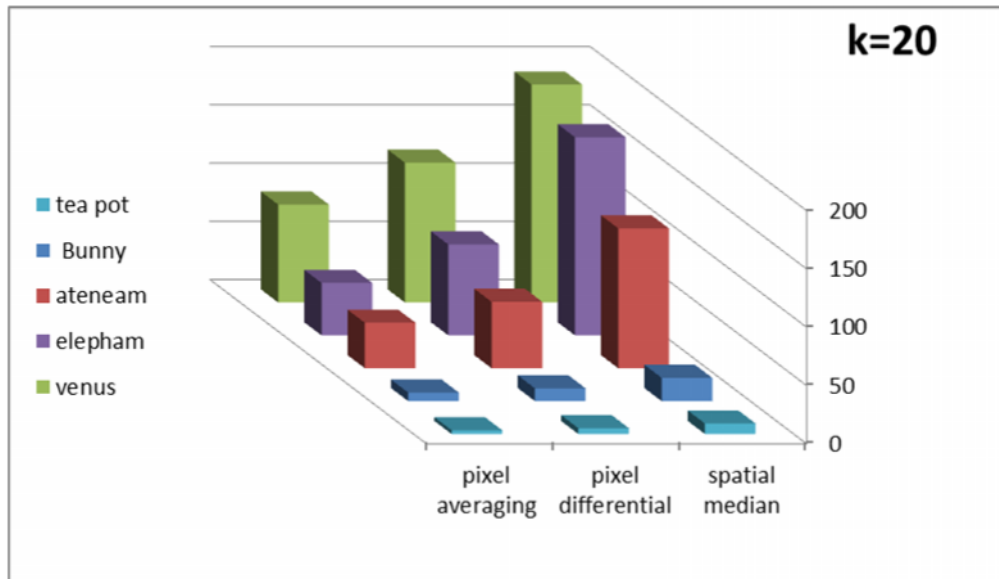


Figure 5. The Differentscenes when k=20

Figure 6 shows the results of the spatial, Pixel differential and Pixel averaging methods for all scenes if k = 30.

The three graphs show the render time of the proposed method is better than the spatial median method. Render time of all scenes in the third method has fallen more than 50%. As indicated in Table3. This amount of decreasing the time or increasing the speed will loss very little of image quality.

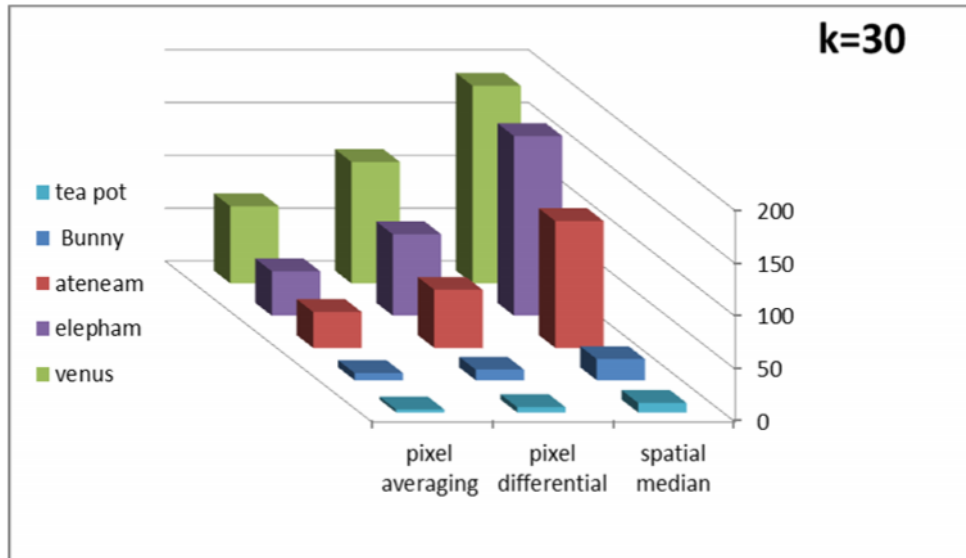


Figure 6. The Differentscenes when k=30

Table 3. Comparing images quality for three value of K

K	scenes	Quality comparisons between	
		spatial median and pixel differential method	spatial median and pixel averaging method
10	Bunny	0.99942	0.99568
	ateneam	0.99965	0.99732
	venus	0.99945	0.99635
	elepham	0.99977	0.99796
	Tea pot	0.99990	0.99850
20	Bunny	0.99886	0.99451
	ateneam	0.99941	0.99680
	venus	0.99906	0.99581
	elepham	0.99958	0.99769
	Tea pot	0.99976	0.99822
30	Bunny	0.99839	0.99361
	ateneam	0.99910	0.99608
	venus	0.99967	0.99789
	elepham	0.99944	0.99749
	Tea pot	0.99967	0.99789

In order to observe the effect of k on render time we have implemented the method for three different K (figure 7). (Comparing the quality of these methods was made with SSIM method in MATLAB software). The results show render time is reduced with increasing K. But this reducing time associated with the loss of image quality which is shown in Table 1. According to

the results, image quality is reduced with increasing k . All graphs have a linear behaviour, thus K has a direct impact on rendering time. But this quality difference is very little. This technique is used where the speed is more important than quality. There are three sections in Table 3. Each section was tested for different $k(10, 20, 30)$. This table shows pixel differential method is better than the pixel averaging method for equal k . differences are very small.

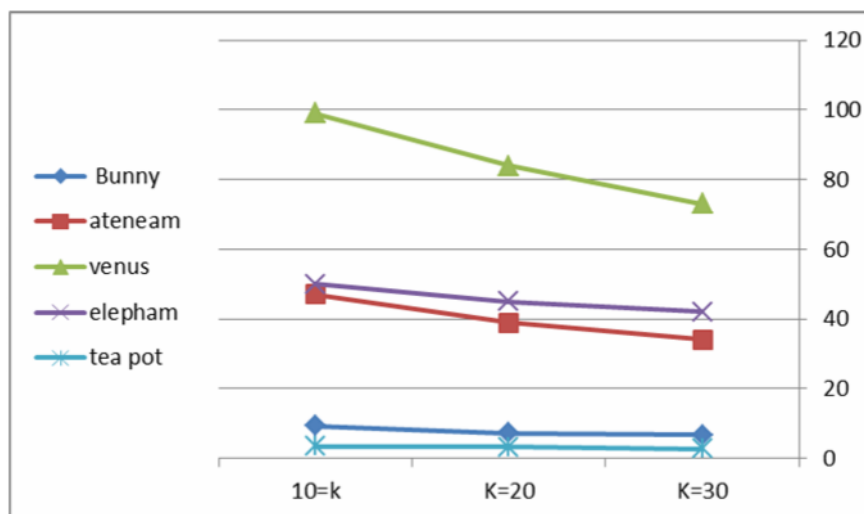


Figure 7. The comparing the render time for three values of K the pixel averaging method

4. CONCLUSION

Ray tracing is a high computational cost, we can reduce ray tracing computation by decreasing the number of rays which should be traced. We propose two hybrid methods to reduce the number of rays. In our methods, some pixels passed in image plane can be computed by ray tracing algorithm or are computed without ray tracing base on using the averaging the neighbour pixels. In this paper, the results show that we can make the image two times faster by reducing the number of rays and the quality of image may be reduced a few hundredths of per cent (about 0.02%). So in this paper, we show that by reducing computational of ray tracing we can make images whit same quality. For having the best rendering motor, we can combine the proposed methods with other methods such SAH that is the highest speed method for building kd-tree.

REFERENCES

- [1] V. Havran and J .Bittner, "On improving kd tree for any shouting", the WSSCG'2002 Conference, pages 209–216, 2002.
- [2] Whitted T. (1979) An improved illumination model for shaded display. Proceedings of the 6th annual conference on Computer graphics and interactive techniques.
- [3] A .Glassner, "An Introduction to Ray Tracing", San Francisco, USA:Morgan Kaufmann, 1989.
- [4] I.Wald and V. Havran. On building fast kd-trees for ray tracing, and on doing that in $O(N \log N)$. In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, Sept. 2006.
- [5]. Havran, V.: Heuristic Ray Shooting Algorithm. PhD thesis, Czech Technical University, Prague (2001)

- [6]. Stoll, G.: Part I: Introduction to Realtime Ray Tracing. SIGGRAPH 2005 Course on Interactive Ray Tracing (2005)
- [7]. Zara, J.: Speeding Up Ray Tracing - SW and HW Approaches. In: Proceedings of 1th Spring Conference on Computer Graphics (SSCG 1995), Bratislava, Slovakia, pp. 1–16 (May 1995)
- [8]. Wald, I.: Realtime Ray Tracing and Interactive Global Illumination. PhD thesis, Computer Graphics Group, Saarland University, Saarbrücken, Germany
- [9] Wald, I. and Havran, V. 2006. On building fast Kd-trees for ray tracing, and on doing that in $O(N \log N)$. In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, pages 61–69.
- [10] Shevtsov, M., Soupikov, A., and Kapustin, A. 2007. Fast and scalable Kd-tree construction for interactively ray tracing dynamic scenes. Computer Graphics Forum 26, 3. (Proceedings of Eurographics).
- [11] Szecsi, L. 2003. An Effective Implementation of the Kd-tree, Graphics Programming Methods, pp 315-326.
- [12] Pharr, M. and Humphreys, G. 2004. Physically Based Rendering : Form Theory to Implementation, Morgan Kaufman.
- [13] V. Havran, R. Herzog, and H.-P. Seidel. On fast construction of spatial hierarchies for ray tracing. In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, Sept. 2006.
- [14]. Hunt, W., Mark, W., Stoll, G.: Fast kd-tree construction with an adaptive error-bounded heuristic. In: Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, pp. 81–88 (September 2006)
- [15]. Benthin, C. 2006. Realtime Raytracing on Current CPU Architectures, Ph.D Dissertation, Saarland University.