RECTILINEAR BSP TREES FOR PREFERRED RAY SETS

Vlastimil Havran
Jiří Bittner

Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University, Karlovo nám. 13
121 35 Prague
Czech Republic
{havran,bittner}@fel.cvut.cz*

Abstract

Rectilinear *Binary Space Partitioning* (BSP) trees are often used for solving various types of range searching problems including ray shooting. We propose a novel method for construction of rectilinear BSP trees for a preferred set of ray shooting queries. Particularly, we study ray sets formed by fixing either the direction or the origin of rays. We analyse and discuss the properties of constructed trees. Theoretical considerations are followed by results obtained from the practical implementation.

Keywords: ray shooting, ray casting, spatial data structures, BSP tree, clipping.

1 INTRODUCTION

Visibility determination is a fundamental problem of computer graphics. One rather simple but very important task for global illumination is ray shooting [Watt92]. The problem is stated as follows: given a ray find its nearest intersection with objects in the scene. The trivial solution would test every object for intersection with the ray in $\Theta(n)$ time. For complex scenes this time complexity is very restrictive since a large amount of rays is needed to synthesise an image.

In our contribution we study the possibility of constructing a rectilinear BSP tree with improved time complexity for preferred set of ray shooting queries. We do not try to bound the worst case complexity. Instead, we exploit a heuristic approach to improve the average query time [Arvo89a].

Firstly, we select a preferred direction of rays, for which the ray shooting should be as much as efficient. This set corresponds to rays involved in the parallel projection. Secondly, we form the preferred ray set by fixing the origin of rays. This ray set is related to the perspective projection. An important aspect of our approach is that the constructed BSP trees can be used for other than preferred ray sets as well. Nevertheless, in the later case the expected time complexity of the visibility query is higher.

The paper is organised as follows: Section 2 outlines some preliminaries regarding to the construction and traversal of BSP trees. In Section 3 the construction of BSP trees for preferred sets of rays is introduced. Section 4 presents results and the statistical evaluation of the proposed method. Finally, Section 5 concludes the paper outlining several directions for future work.

2 BSP TREES

A rectilinear BSP tree ($BSPT$[1]) is a higher dimensional analogy to the binary search tree. A $BSPT$ for a set $\mathcal{S}$ of objects is defined as follows: Each node v in $BSPT$ represents a non–empty axis–aligned box (cell) B_v. The cell associated with the root of the tree is the bounding box of all objects from $\mathcal{S}$. Each interior node v of $BSPT$ is assigned a cutting plane H_v, that divides B_v into two cells. Let H_v^+ be the positive halfspace and H_v^- the negative halfspace bounded by H_v. The cells associated with the left and the right children of v are $B_v \cap H_v^+$ and $B_v \cap H_v^-$, respectively. The left subtree of v is a $BSPT$ for the set of objects $S_v^- = \{s \cap H_v^- \neq \emptyset | s \in S_v\}$, the right subtree is defined similarly. Each leaf node l contains a list of objects S_l that intersect B_l. The leaves of the $BSPT$ are either occupied by objects or vacant. A two dimensional example of $BSPT$ is depicted in Fig. 1.

The fact that the cutting planes of $BSPT$ are orthogonal significantly simplifies the computation of ray shooting queries.

A $BSPT$ is constructed hierarchically. At the current leaf l a cutting plane is selected that subdivides R_l into two boxes. A leaf then becomes an interior node with two new leaves. The objects S_l are distributed into the new descendants of l. The process is repeated recursively until certain termination criteria are reached.

*This research was partially supported by Grant Agency of the Ministry of Education of the Czech Republic number 1252/1998 and by Internal Grant Agency of the Czech Technical University in Prague number 309810103.

[1] Rectilinear $BSPT$s are also called kD-trees [Berg97].

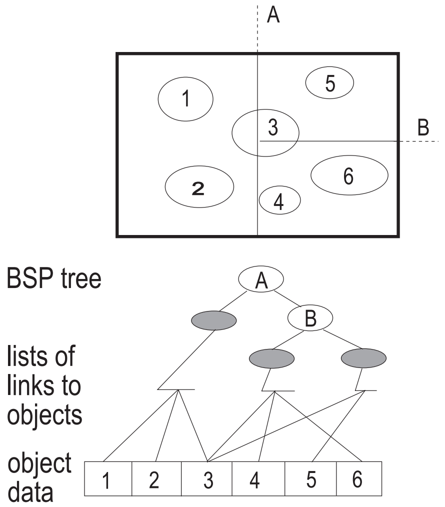


Figure 1: A two dimensional example of $BSPT$.

An important feature of the rectilinear $BSPT$ is its adaptability to the scene geometry, that can be significantly influenced by positioning of the cutting plane. Traditionally, a cutting plane is positioned in the mid–point of the chosen axis, and the order of axes is regularly changed [Kapla85]. In the following text we recall a more elaborate approach for constructing $BSPT$ using a surface area heuristics.

2.1 Surface Area Heuristics

The *surface area heuristics* for construction of BSP trees was introduced in [MacDo90a]. Since this approach is essential for our novel technique we describe the method in more detail. Related work concerning surface area heuristics in computer graphics includes the work of Weghorst et al for selection of bounding volumes [Wegho84] and Iones for collision detection [Iones98].

Using surface area heuristics the position of the cutting plane is determined by minimising a *cost function*. The cost function accounts for the probability $p(B|A)$ that a ray hits an object lying inside a certain volume once it passes through that volume (see Fig. 2.1).

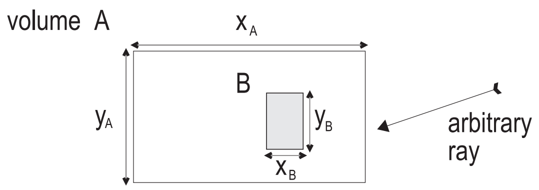


Figure 2: Geometry involved in determination of the probability $p(B|A)$.

Suppose that both object B and volume A are convex boxes. Then the conditional probability $p(B|A)$ can be expressed as a ratio of the surface area of the object B to the surface area of the volume A (see [Arvo89a]):

$$\begin{aligned} p(B|A) &= \frac{S(B)}{S(A)} \\ &= \frac{2.(x_B.y_B + x_B.z_B + y_B.z_B)}{2.(x_A.y_A + x_A.z_A + y_A.z_A)} \end{aligned} \tag{1}$$

Let us assume the situation at the beginning of the $BSPT$ construction. The root node contains n objects. All of them would have to be tested for intersection with a ray passing through the scene. Assume that the intersection test for i–th object takes computational time T_i. The cost for such node is expressed as:

$$C = \sum_{i=1}^{n} T_i \tag{2}$$

The further subdivision decreases the number of intersection tests, but increases the number of interior nodes (see Figure 3).

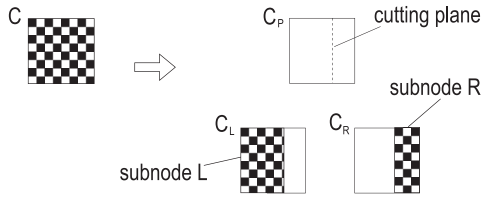


Figure 3: New costs after one cutting.

The node on the left side in Fig. 3 has been replaced by a tree shown on the right side of the figure. The original cost C has changed to C_{new} given as the sum of three terms - C_P, C_L, and C_R. C_P is the cost of traversing the parent node and it does not incorporate any ray–object intersection tests. Costs for left and right child nodes, C_L and C_R, depend on the conditional probability that a ray hits the node L or R once it visits the parent node P. Assuming both children will not be further subdivided, the new cost C_{new} is given as follows:

$$\begin{aligned} C_{new} &= C_P + C_L + C_R \\ &= T_P + p_L.\sum_{j=1}^{n_L} T_j + p_R.\sum_{k=1}^{n_R} T_k + T_T \end{aligned} \tag{3}$$

$$p_L = \frac{S(C_L)}{S(C)} \qquad p_R = \frac{S(C_R)}{S(C)} \tag{4}$$

, where

T_j, T_k	is the time for intersection test with j-th and k-th object respectively
T_T	is the time of traversal step
T_P	is the time of decision step in interior node
p_L, p_R	is the probability that a ray will intersect the left and right sub–cell respectively
S_L, S_R	is a surface area of left sub–cell and right sub–cell respectively
S	is the surface area of the node to be subdivided
n_L, n_R	is the number of objects intersecting the left and right sub–cell respectively

Note, that the Eq. 3 represents the case when the ray visits both left and right sub–cells.

The goal is to build a $BSPT$ with minimised global cost. The cost function is evaluated for all possible positions of splitting plane in all three axes that are induces by the boundaries of axes aligned boxes of all objects. We focus on a greedy heuristics that minimises the cost function by selecting the cutting plane which results in a new minimum cost.

3 RAY SHOOTING FOR PREFERRED SETS OF RAYS

In the previous section we have described the construction of $BSPT$ using surface area heuristics. The main idea of this paper concerns a modification of the surface area heuristics in order to decrease the visibility query time for certain preferred set of rays.

The surface area heuristics is based on the assumption that rays are distributed uniformly in ray space. We break this assumption and change the above presented equations to consider only certain ray set.

In the following text we focus on three types of ray sets that correspond to parallel, perspective, and spherical projection.

3.1 Parallel Projection

Probably the most intuitive set of rays is formed by fixing their direction. The rays of the same direction are involved in the parallel projection of the scene [Watt92].

In such a case rays are perpendicular to the *projection plane*. Let us suppose that the distribution of rays on the projection plane is uniform. Further, we can restrict the preferred set of rays to certain *viewport*. The probability that a ray hits the box B representing a node of $BSPT$ can then be expressed using a surface area of the projection of the box B clipped to the viewport. The corresponding geometry is depicted in Fig 4.

Let viewport W_R be a rectangular window on the projection plane P_P. Let $\mathcal{R}_{PAR_{W_R}}$ be the set of rays perpendicular to P_P and intersecting the W_R.

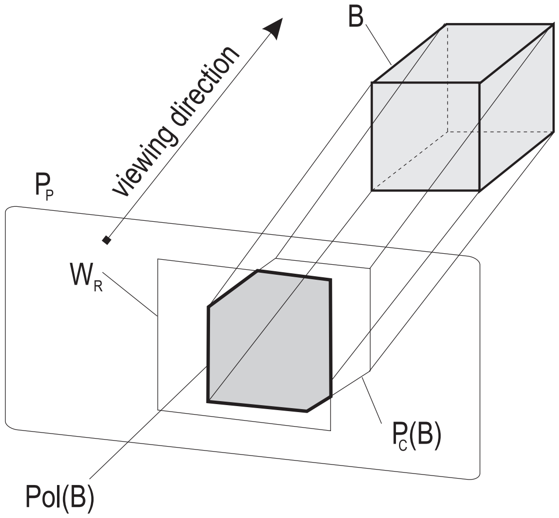


Figure 4: Parallel projection of box B

The silhouette edges of B projected onto the P_P form a convex polygon $P_C(B)$. Optionally, we can define $P_C(B)$ as a convex hull of all the vertices of B projected to P_P. To determine the polygon $Pol(B) = W_R \cap P_C(B)$ lying on P_P a clipping algorithm must be applied [Watt92]. Let $SA^{PAR}(B)$ be the surface area of the polygon $Pol(B)$ and let B_{SC} be the bounding box containing the whole scene. Similarly to Eq. 1 we can express the probability, that a ray from $\mathcal{R}_{PAR_{W_R}}$ hits the box B once it passes through B_{SC} as follows:

$$p(B) = \frac{SA^{PAR}(B)}{SA^{PAR}(B_{SC})} \quad (5)$$

The Eq. 5 can be used to replace directly the Eq. 4 for both left and right sub–cells.

3.2 Perspective Projection

We form another preferred ray set by fixing the origin of rays. Similarly as before we focus only on rays passing through certain viewport. The origin (viewpoint) and the viewport define a viewing frustum (see Fig. 5). Assuming rays are uniformly distributed on the viewport, the corresponding ray set $\mathcal{R}_{PER_W}$ contains rays involved in the commonly used perspective projection.

We compute the projected area $SA^{PER}(B)$ of the box B clipped to the viewing frustum. Again, the conditional probability, that a ray from $\mathcal{R}_{PER_W}$ hits the box B once it passes through B_{SC} can be expressed as:

$$p(B) = \frac{SA^{PER}(B)}{SA^{PER}(B_{SC})} \quad (6)$$

The clipping mentioned previously must be applied for each evaluation of the estimated cost. Hence the speed of the clipping is crucial for the construction process. In the following text, we outline the algorithms of clipping and computing the surface area of the projected polygon.

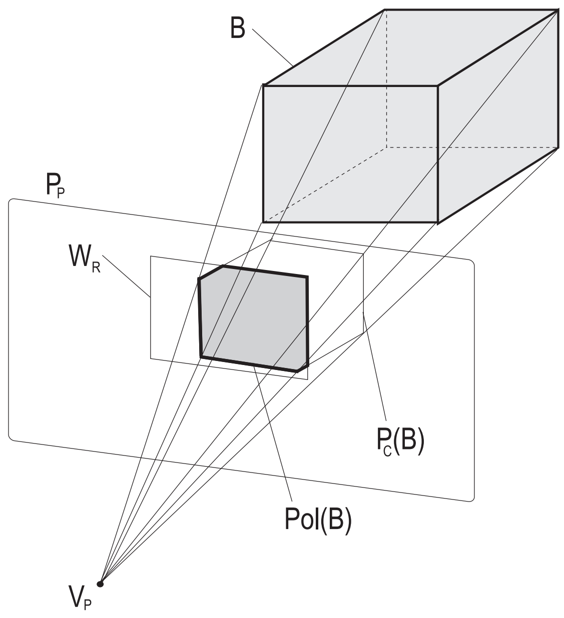


Figure 5: The perspective projection of box B.

Viewing Frustum Clipping

A general algorithm determining surface area of the projection of a rectilinear box can be outlined as follows: Project all the vertices of the bounding box to the projection plane and construct a convex hull of the projected vertices. This convex hull corresponds to a certain polygon. Then clip the polygon with respect to the viewport and compute its surface area.

Obviously, the projection of all the eight vertices and construction of the convex hull of the rectilinear box for a specific viewpoint V_P is useless. We need construct only the projection of the silhouette of the box. In such a case the maximum number of projected points is six, the minimum is four. According to the mutual position of the viewpoint and the box we can identify twenty–seven regions, for which the projected box has the same sequence of silhouette edges. These regions are induced by six planes bounding the box. Given a viewpoint the corresponding sequence of silhouette edges can be determined by table lookup.

The sequence of silhouette edges is clipped in object space by Sutherland–Hodgman algorithm [Watt92] using the four planes that bound the viewing frustum. The clipping forms a sequence of connected edges with at most ten vertices. The vertices are projected to the projection plane forming a convex polygon. Finally, the surface area of the polygon is computed. A special case occurs when the viewpoint lies inside the box. Naturally, every ray originating at the viewpoint must intersect the box. Thus the conditional probability used in Eq. 6 equals 1.0.

3.3 Spherical Projection

Another interesting set of rays is induced by those involved in the *spherical projection*. As with the perspective projection the origin of rays is fixed. Nevertheless, in the spherical projection directions of rays are uniformly distributed.

A data structure built according to such as set of rays could be used to accelerate shadow–ray visibility queries for point light sources. Note, that in this case clipping phase can be omitted.

Assuming the distribution of rays is uniform the task is to compute a solid angle induced by the frustum enclosing a given box. This task is similar to the computation of the point-to-polygon form factor [Glass95]. Nevertheless, in this case it involves determination of an area bounded by Jordan's curve on a unit sphere. The solid angle induces by a given box can be computed as the sum of all solid angles for visible faces from a given viewpoint. The solid angle Ω of a rectangle with length c and width d positioned with one corner at the origin and perpendicular to z–axis can be computed in the following way (see Fig. 6):

$$l = \sqrt{x^2 + y^2 + h^2}$$

$$\cos\beta = \frac{h}{l}$$

$$dA = dx.dy$$

$$d\Omega = \frac{dA.\cos\beta}{l^2}$$

$$d\Omega = \frac{h.dx.dy}{\sqrt{(x^2 + y^2 + h^2)^3}}$$

Solid angle is then computed by integration:

$$\Omega[sr] = \int_{x=0}^{c}\int_{y=0}^{d} \frac{h.dx.dy}{\sqrt{(x^2 + y^2 + h^2)^3}} \tag{7}$$

Solving this integral with the substitution $u = \frac{x}{h}$ and $v = \frac{y}{h}$ results in:

$$\Omega[sr] = \arctan\frac{c.d}{h.\sqrt{h^2 + c^2 + d^2}} \tag{8}$$

The computation of solid angle for a rectangle with one corner not positioned in the origin is computed by more general solution of the equation above.

It is obvious that for a viewpoint located inside a given box the corresponding solid angle is $\Omega = 4.\pi$.

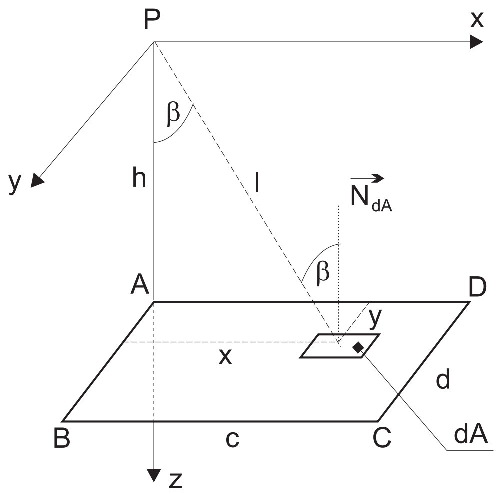


Figure 6: Computation of spherical projection for rectangle

4 IMPLEMENTATION AND EXPERIMENTAL RESULTS

We have implemented the construction of $BSPT$s within the GOLEM rendering system. In our experiments we used three SPD scenes [Haine87a] and one depicting the simulation of a liquid flow. These are depicted in Fig. 10. We have evaluated the $BSPT$ construction for preferred ray sets induced by the parallel (PAR), perspective (PER), and spherical (SPH) projections. The ordinary surface area heuristics (SAH) was used as a reference. Table 1 gives basic description of the scenes.

Scene	fluid	lattice	rings	tree
#objects	2515	8281	8401	8191
#spheres	2514	2197	4200	4095
#polygons	1	-	1	1
#cones	-	-	-	4095
#cylinders	-	6084	4200	-

Table 1: Testing scenes

The scenes were rendered in 513 × 513 resolution. The maximum depth of ray–tracing recursion was set to 1 to test the preferred ray sets only. All $BSPT$s were constructed with following termination criteria: maximum depth was 16 and the number of objects for a node to become a leaf was 2. A hierarchical traversal algorithm [Havra98b] for $BSPT$ was used. Number of rays cast for each test was $513 \times 513 = 263169$.

Table 2 shows the results for $BSPT$ constructed for parallel projection with no clipping performed. The number of ray object intersection tests for parallel projection is decreased by 16 % and the number of traversal steps by 24 % on average.

Table 3 shows the results for $BSPT$s obtained for perspective projection using viewing frustum clipping in object space. The number of intersection tests for $BSPT$ decreased by 29 % and the number of traversal steps by 16 % on average.

Scene	fluid	lattice	rings	tree
ScnCov[%]	100	99.24	100	64.6
viewplane	-0.40	0.0	0.0	-0.99
normal =	-0.50	-0.71	1.0	-0.09
$(x, \vec{y}, z)$	-0.86	-0.71	0.0	-0.11
	# leaves of $BSPT$			
SAH	1532	15722	11845	2750
PAR	1803	14611	12947	3156
	# intersection tests per ray			
SAH	8.15	47.3	8.12	5.63
PAR	6.13	37.7	7.97	4.55
	# traversal steps per ray			
SAH	18.7	71.0	31.2	19.4
PAR	18.1	30.1	26.1	15.3

Table 2: $BSPT$ for parallel projection

Scene	fluid	lattice	rings	tree
ScnCov[%]	100	99.3	100	64.6
	# leaves of $BSPT$			
SAH	1532	15722	11845	2750
PER	3132	13247	16322	3992
	# intersection tests per ray			
SAH	8.79	11.9	11.1	10.1
PER	5.32	9.13	9.15	6.39
	# traversal steps per ray			
SAH	19.6	44.9	40.9	23.1
PER	18.4	29.7	38.1	19.0

Table 3: $BSPT$ for perspective projection

Scene	fluid	lattice	rings	tree
ScnCov[%]	74.4	99.3	96.3	64.2
	# leaves of $BSPT$			
SAH	1532	15722	11845	2750
SPH	2236	12827	13735	3231
	# intersection tests per ray			
SAH	8.72	11.8	11.1	9.92
SPH	6.59	10.8	8.38	6.61
	# traversal steps per ray			
SAH	19.4	44.8	40.3	22.5
SPH	18.4	33.7	38.1	14.4

Table 4: $BSPT$ for spherical projection

The results for spherical projection are very similar to perspective projection (see Table 4).

The preprocessing times for constructing $BSPT$s with SAH and PER do not differ significantly. It is due to the fact that the clipping algorithm is performed in incremental way, that utilises the coherence of clipping for subsequent cutting planes along the tested axis. The preprocessing times for SAH and PAR without clipping are quite comparable, since parallel projection corresponds to multiplying the surface area of box faces with elements of the projection vector defining preferred set of rays. The overhead of preprocessing time for the SPH with regard to SAH is also very small when some approximation for $\arctan$ function is used [Capel91].

4.1 Impact on Traversal Algorithms

We have experimented with two different traversal algorithms for $BSPT$s. The first one [Havra98b] is based on the hierarchical traversal algorithm presented in [Sung92]. The second one uses a *rope technique* for BSP trees [Havra98a] that eliminates traversal of interior nodes of the tree. We have observed that the proposed modifications have similar impact on both traversal algorithms.

4.2 Parallel Projection in Close–Up

We have experimented with the sensitivity of construction of $BSPT$ to the direction of ray shooting queries. We used scene *tree* with the following initial projection settings; $viewpoint = (4.5, 0, 1.5)$, $lookat = (0, 0, 1.5)$, $upvector = (0, 0, 1)$. This corresponds to the normal of the projection plane $\vec{N} = (1, 0, 0)$ and $azimuth = 0^o$. In the measurements the observer moved around the tree. That is, $lookat$ and $upvector$ remained constant and both $viewpoint$ and $\vec{N}$ have changed according to the $azimuth$ in the interval $< 0..90 >$ ($azimuth = 90^o$ corresponds to $viewpoint = (0, 4.5, 1.5)$ and $\vec{N} = (0, 1, 0)$).

First, we compared the $BSPT$s built using SAH and PAR. The $BSPT$ for PAR was constructed for the parallel projection corresponding to the azimuth ($\vec{N}$). The Fig. 7(a) shows the rendering time including shading. The Fig. 7(b) depicts the number of ray object intersection tests. The Fig. 8(a) shows the number of traversal steps per ray. The curves for PAR on figures are referenced as PAR-A.

Second, we tested the sensitivity of $BSPT$ constructed for a fixed $azimuth = 0.5^o$ for rendering with other azimuth angles as well. Fig 8(b) shows the rendering times for views specified by changing azimuth. The curve is referenced as PAR-B. We can observe that the efficiency of ray shooting with $BSPT$ constructed with PAR is restricted to angular divergence of 40^o from the one used for construction. The difference between $BSPT$ constructed using SAH and PAR is illustrated in Fig. 9.

4.3 Discussion

We made observation, that $BSPT$ constructed for PAR with normal $\vec{N} = (a, b, c), |a| = |b| = |c|$ corresponds to $BSPT$ constructed for SAH. There are 8 vectors on a unit sphere with these properties. Hence, $BSPT$ constructed for PAR with other vectors than these 8 vectors can be potentially improved. The efficiency improvements are rather significant, both the number intersection tests and traversal steps are reduced by more than one half for tested scene *tree*. The tree constructed for parallel projection for $\vec{N} = (1, 0, 0)$ degenerates, because one principal axis is not taken into account at all. It is the reason why we selected for Fig. 8(b) as the reference the $BSPT$ constructed for $azimuth = 0.5^o$.

There is a little problem with comparison of $BSPT$s [Havra97a]. Although the termination criteria were the same, SAH, PAR, PER, and SPH methods constructed $BSPT$ with different number of leaves. This makes more difficult to compare their performances. Nevertheless, we verified that the achieved significant speedup for parallel projection cannot be obtained by SAH using other termination criteria settings.

The second potential problem arises with the clipping of the projected box, when the built up $BSPT$ is used for rays outside the viewing frustum. Since the probability using Eq. 5 is zero, the constructed $BSPT$ can degenerate and query time as well. There are two solutions to this problem. The first one detects the situation and then use SAH for any nodes of $BSPT$ that are at least partially outside the viewing frustum. Then even the nodes on the boundary of viewing frustum will be constructed using SAH. The second solution is simpler; the perspective projection is replaced by spherical projection, since the distribution of rays are very similar and no clipping is performed. We verified that the speedup of ray shooting for perspective projection with $BSPT$ built up for spherical projection is almost the same.

5 CONCLUSION AND FUTURE WORK

In this paper we have shown that it is possible to improve the efficiency of rectilinear $BSPT$s by preferring a given set of rays using modified surface area heuristics. If the ray set is induced by perspective or spherical projection, both the number of intersection tests and number of traversal steps are decreased. The time of ray shooting is decreased by 20 % on average in practical application and is scene dependent.

The reduction of time complexity obtained for set of rays induced by parallel projection can be quite remarkable and can reach one half approximately. The efficiency improvement depends on the scene and the setting of preferred direction.

The technique developed is applicable to ray casting for CSG primitives and implicit surfaces, that cannot be efficiently rendered using image space rendering techniques. It can be used for primary rays only, when the observer does not move. In the case the observer moves only slightly, then for the computation of primary rays for several subsequent images can be performed using the same $BSPT$.

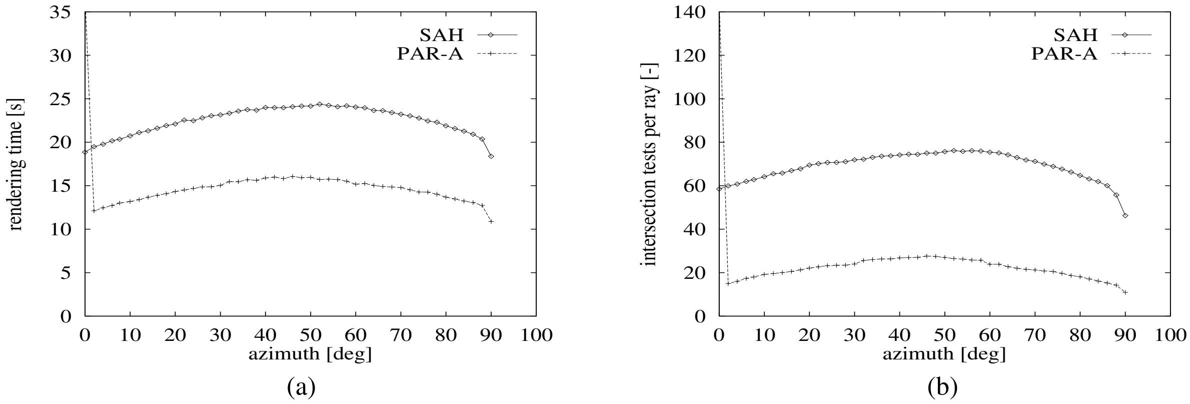


Figure 7: (a) Rendering times including shading for SAH and PAR-A (b) Number of intersection tests per ray

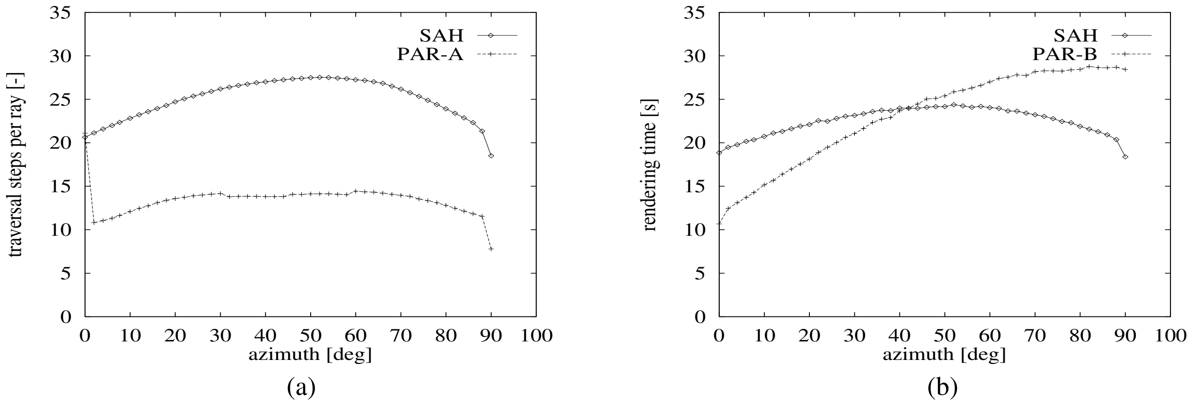


Figure 8: (a) Number of traversal steps per ray (b) Rendering times including shading for SAH and PAR-B

We are going to implement ray tracing with multiple $BSPT$s constructed for parallel projections. For a given ray we will select the most advantageous instance of $BSPT$. Further, we would like to improve the performance of $BSPT$s by considering a *blocking factor*. Mostly, sub–cells of a node have the overlap in the projection, that should be taken into account in construction of $BSPT$s.

References

[Arvo89a] Arvo, J., Kirk, D.: A survey of ray tracing acceleration techniques, In *An Introduction to ray tracing*, pp. 201–262, Academic Press, 1989.

[Berg97] de Berg, M., et al:*Computational Geometry: Algorithms and Applications*, Springer–Verlag, 1997.

[Capel91] Capelli, R: Fast approximation to the arctangent, in *Graphics Gems II*, pp. 389–391, 1991.

[Fuchs80] Fuchs, H., Kedem, M.Z. Naylor, B.: On Visible Surface Generation by A Priori Tree Structures, *Proceedings of SIGGRAPH'80*, pp. 124–133, 1980.

[Glass89] Glassner, A., S.:*An Introduction to Ray Tracing*, Academic Press, London, 1989.

[Glass95] Glassner, A. S.:*Principles of Digital Image Synthesis*, Morgan Kaufmann, San Francisco, 1995.

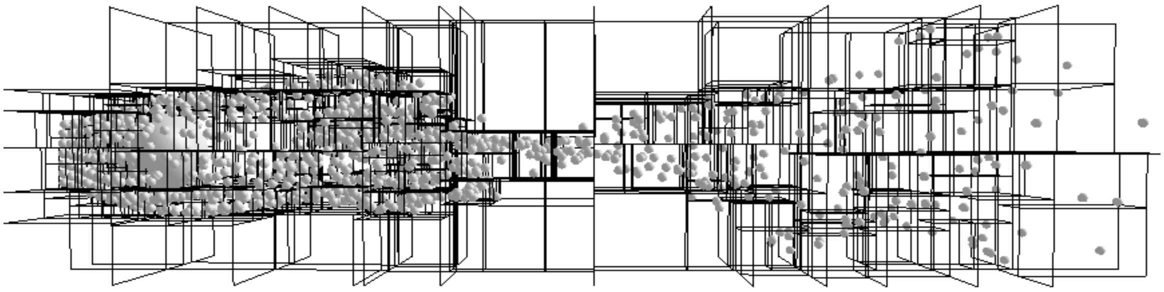
[Haine87a] Haines, E.: A proposal for standard graphics environments, In *IEEE Computer Graphics and Applications*, Vol. 7, No. 11, pp. 3–5, 1987.

[Havra97a] Havran, V., Žára, J.: Evaluation of BSP properties for ray-tracing, in *Proceedings of Spring Conference On Computer Graphics*, Budmerice in Slovakia, pp. 155–162, 1997.

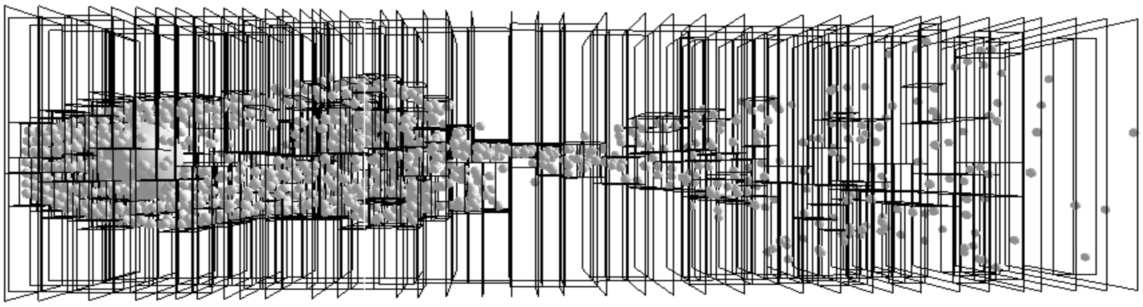
[Havra98a] Havran, V., Bittner, J., Žára, J.: Ray Tracing with Rope Trees, in *Proceedings of 13th Spring Conference On Computer Graphics*, Budmerice in Slovakia, pp. 130-139, 1998.

[Havra98b] Havran, V., Kopal, T., Bittner, J., Žára, J.: Fast Robust BSP Tree Traversal Algorithm for Ray Tracing, in *Journal of Graphics Tools*, AK Peters Ltd., Vol 2, No 4, pp. 15–24, 1998.

[Iones98] Iones, A., Zhukov, S., Krupkin, A.: On Optimality of OBBS for Visibility Tests for Frustrum Culling, Ray Shooting, and Collision Detection., in *Computer Graphics International'98*, published by IEEE New York, Hannover, Germany, pp. 256–263, 1998.



(a)



(b)

Figure 9: Visualisation of the $BSPT$. Fig. (a) depicts a $BSPT$ built using the ordinary surface area heuristics (SAH). Fig.(b) show a $BSPT$ constructed for parallel projection(PAR). For the sake of visual clarity the maximum depth of the tree was set to 10.

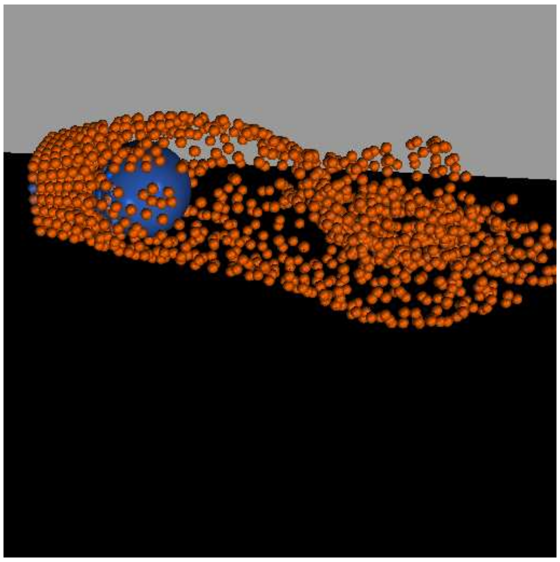
[Kapla85] Kaplan, M.: The Use of Spatial Coherence in Ray Tracing, *ACM SIGGRAPH'85 Course Notes 11*, pp. 22–26, July 1985.

[MacDo90a] MacDonald, J., D., Booth, K.,S.: Heuristics for ray tracing using space subdivision, *The Visual Computer*, Vol. 6, No. 3, pp. 153–166, 1990.

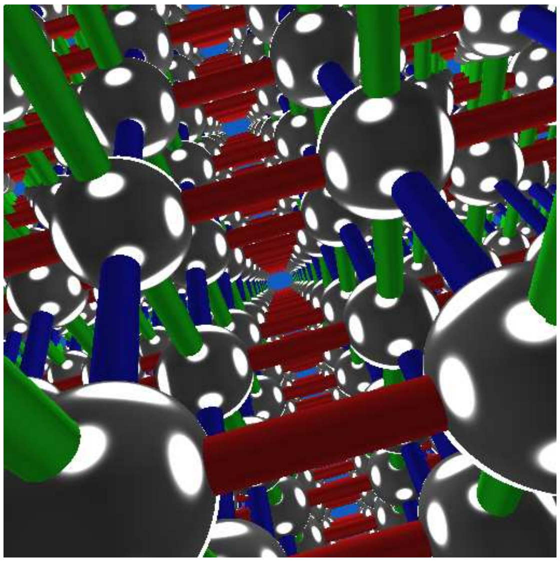
[Sung92] Sung, K., Shirley, P.: Ray Tracing with BSP Tree, *Graphics Gems III*, pp. 271–274, 1992.

[Watt92] Watt, A., Watt, M.:*Advanced Animation and Rendering Techniques*, ACM–PRESS, Addison–Wesley, 1992.

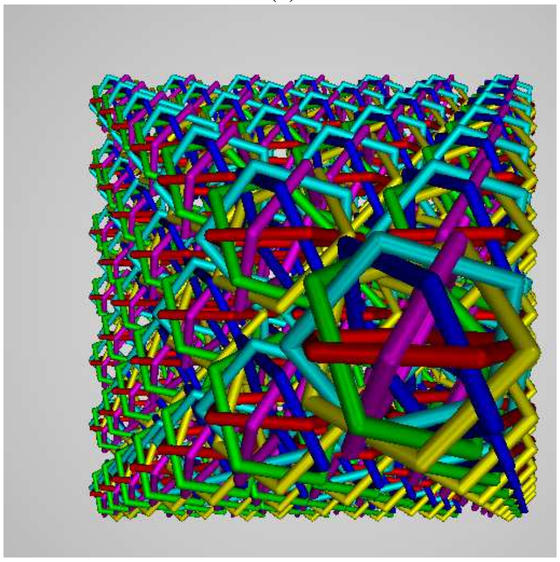
[Wegho84] Weghorst, H., Hooper, G., Greenberg, D.P.: Improved Computational Methods for Ray Tracing, in *ACM Transactions on Graphics*, Vol. 3, No. 1, pp. 52–69, 1984.



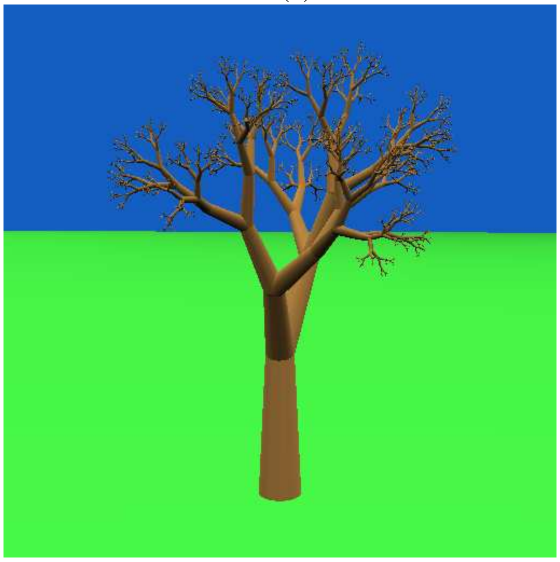
(a)



(b)



(c)



(d)

Figure 10: Test scenes: (a) fluid, (b) lattice, (c) rings, (d) tree