1 Introduction

Rendering technique for producing realistic images that simulates well specular surfaces often use discrete sampling of space. The main drawback of these techniques is its rather big computational complexity of this discrete sampling, that disallows its interactive use.

The principal expense of ray tracing and other global illumination methods is the determination of the closest ray–object intersection for a given ray and a set of objects. This problem is known as ray–shooting. The naive algorithm solves the ray–shooting problem by testing all objects for intersection with a given ray in $O(N)$ time. Another related problem is determining if two points in a scene are visible. Visibility of two points is used to determine shadows cast by point light sources.

We deal with an unusual method of ray–shooting acceleration based on BSP trees using modified surface area heuristics.

Related Work

The ray–shooting problem has been dealt by the researches in the field of computer graphics and computational geometry. Techniques developed by these two groups differ in approach used.

The first community mentioned is oriented to improve the average complexity of ray–shooting. The algorithms developed are intended to be used in practice, even for large scale scenes. They are mostly based on the observations and heuristics. Recent comprehensive overview is presented in [Sim95].

The computational geometry community solves the problem more theoretically focusing on worst–case complexity. In order to obtain mathematically provable results the scene description is usually restricted to polygons. A recent
algorithm published [dB93] reaches the time complexity $O(\log n)$ with $O(n^{4+\epsilon})$ preprocessing time and $O(n^{4+\epsilon})$ storage, where $\epsilon$ is an arbitrarily small positive constant, and $n$ is a number of polygons. The preprocessing and storage complexities restrict their practical use.

2 Binary Space Partitioning

A Binary Space Partitioning tree (abbr. BSP tree or BSPT) is a variant of binary search tree, but it organizes $n$–dimensional data ($n > 1$). A BSPT for a set $S$ of objects in $\mathbb{R}^n$ is a binary tree defined as follows: Each node $v$ in BSPT represents a non–empty box (rectangular parallelepiped) $R_v$ and set of objects $S_v$ that intersects $R_v$. Leaf is such a node for which the number of objects $|S_v|$ belonging to $v$ is smaller than a specific constant or the depth of $v$ in BSPT is equal to maximal depth allowed. The box associated with the root of BSPT is $\mathbb{R}^n$ itself. Each interior node of BSPT is assigned cutting plane $H_v$, that intersects $R_v$ into two boxes. If we let $H_v^+$ be the positive halfspace and $H_v^-$ the negative halfspace bounded by $H_v$, the boxes associated with the left and right children of $v$ are $R_v \cap H_v^+$ and $R_v \cap H_v^-$ respectively. The left subtree of $v$ is a BSPT for the set of objects $S_v^- = \{s \cap H_v^- | s \in S_v\}$, right subtree is defined similarly. Let size of BSPT be the number of interior nodes and the number of references to objects in all its leaves.

A BSPT is constructed hierarchically step by step until termination criteria given for leaf are reached. The cutting plane $H$ is for ease of computing range search queries including ray–shooting perpendicular to one of coordinate axes (orthogonal cutting). The leaves of the BSPT are either occupied by objects or vacant. The example of BSPT for two–dimensional space is depicted in Fig. 1.
3 Surface Area Heuristics

It is advantageous to devote a greater effort to create an efficient BSPT, under the assumption, that the extra time would be recovered during ray–traversal.

In [MB90] a simple heuristics for finding the optimal position of a splitting plane is used. The plane position is determined by minimizing a cost function. The cost function is based on the probability that a ray hits an object placed inside a certain volume once it passes through that volume as shown in Fig. 2. Suppose that both object $B$ and a enclosing object $A$ are of convex shape. Then the conditional probability $Pr(B|A)$ is expressed as a ratio of the surface area of the object $B$ to the surface area of the volume $A$ (see [AK89]):

$$Pr(B|A) = \frac{S_B}{S_A} = \frac{2(x_B y_B + x_B z_B + y_B z_B)}{2(x_A y_A + x_A z_A + y_A z_A)}$$

Figure 2: Surface area heuristics

During the building of a BSPT the cost function helps to decide when and where to split a certain cell, i.e., to replace a leaf node by a new interior node with two children (sub–cells).

Let us assume the situation at the beginning of a tree construction. One node contains $n$ objects, the intersection test for $i$–th object takes computation time $T_i$. The cost for such non–subdivided node is $f_M = \sum_{i=1}^{n} T_i$.

Let $A_v$, $A_v^-$, and $A_v^+$ is the surface area of $R_v$, $R_v^-$, and $R_v^+$ respectively. The selection of cutting plane for BSPT proposed in [MB90] is proceeded by maximizing the measure $f_M = A_v^- / A_v^- |S_v^-| + A_v^+ / A_v^+ |S_v^+|$ for each constructed cutting plane $H$.

The recursive ray traversal algorithm through BSPT deserves a special attention, but it is out of scope of this paper.

4 Modified Surface Area Heuristics

It is also possible to modify the surface area heuristics for a restricted set of rays. This restriction fixes either origin of rays (perspective projection) or direction of rays (parallel projection). The above formula are modified in the probability term, which is computed as the projected surface of child node to the surface area of the node being subdivided.

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 $B$ clipped to the viewport. The corresponding geometry is depicted in Fig 4.

We compute the projected area $S^P_E R(B)$ of the box $B$ clipped to the viewing frustum. Again, the conditional probability, that a ray from $R_{PERW}$...
Figure 3: Parallel projection of box $B$

hits the box $B$ once it passes through $B_{SC}$ can be expressed as:

$$p(B) = \frac{SA_{PER}(B)}{SA_{PER}(B_{SC})}$$  \hspace{1cm} (1)

5 Results

The tests were performed by rendering Standard Procedural Database scenes introduced by Haines in [Hai87]. We used the experimental measure $<\Lambda, \Delta>$ to compare the efficiency of spatial subdivisions given in [H97b] for presentation of our results, that are published in [H98b].

All images were rendered in 513×513 resolution. The maximal ray–recursion depth was set to 4. All BSP trees were constructed with following termination criteria: maximal depth was 18 and the number of primitives for a node to become a leaf was 2.

In order to decrease further the time complexity of ray-shooting the concept of mailboxes was used for testing of all acceleration methods.

The results show that the BSPT built up using the surface area heuristics is always quicker than Kaplan’s BSPT (constructed by splitting in the spatial median and with regular change of splitting planes orientation). This difference can be very significant, when the objects are non–uniformly distributed in the scene (scene balls and tree) (up to 97% of rendering time can be saved for scene tree).

We have evaluated the BSPT construction for preferred ray sets induced by the parallel projection (PAR) and the perspective projection (PER). The ordinary surface area heuristics (SAH) was used as a reference. Table 1 gives several results using modified surface area heuristics.

Fig. 5 depicts visualisation of BSP trees for normal and modified surface area heuristics.
6 Conclusion and Future Work

The combination of normal construction techniques using surface area heuristics presented here decreases computational complexity by 20-38% in comparison with [MB90]. The size of BSPT is also reduced in dependence of input scene. Modified surface area heuristics also decreases computation cost up to 50%. The robust optimal traversal algorithm [H98b] keeps number of intersection tests, but traversal of data structure is decreased also by 30–50%. Combinining techniques, the reduction of computation time is thus significant.

It remains open problem how to estimate the total cost $f_C(S_v)$ of node $R_v$ containing $N$ objects assuming $R_v$ is to be refined by constructing its $BSP$. Solving the problem should further improve performance of $BSP$ for ray–tracing.

References

<table>
<thead>
<tr>
<th>Scene</th>
<th>fluid</th>
<th>lattice</th>
<th>rings</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScnCov[%]</td>
<td>100</td>
<td>99.24</td>
<td>100</td>
<td>64.6</td>
</tr>
<tr>
<td>#rays[×10⁴]</td>
<td>26.3</td>
<td>26.3</td>
<td>26.3</td>
<td>26.3</td>
</tr>
<tr>
<td># leaves of BSPT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAH</td>
<td>1532</td>
<td>15722</td>
<td>11845</td>
<td>2750</td>
</tr>
<tr>
<td>PER</td>
<td>3132</td>
<td>13247</td>
<td>16322</td>
<td>3992</td>
</tr>
<tr>
<td># intersection tests per ray</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAH</td>
<td>8.79</td>
<td>11.9</td>
<td>11.1</td>
<td>10.1</td>
</tr>
<tr>
<td>PER</td>
<td>5.32</td>
<td>9.13</td>
<td>9.15</td>
<td>6.39</td>
</tr>
<tr>
<td># traversal steps per ray</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAH</td>
<td>19.6</td>
<td>44.9</td>
<td>40.9</td>
<td>23.1</td>
</tr>
<tr>
<td>PER</td>
<td>18.4</td>
<td>29.7</td>
<td>38.1</td>
<td>19.0</td>
</tr>
</tbody>
</table>

Table 1: BSPT for perspective projection

SIGGRAPH ’87, ’88, ’89 Introduction to Ray Tracing course notes, code available via FTP from princeton.edu:/pub/Graphics.


Figure 5: Visualization of the BSPT. Fig. (a) depicts a BSPT built using the ordinary surface area heuristics (SAH). Fig. (b) shows a BSPT constructed for parallel projection (PAR). For sake of the visual clarity the maximum depth of the tree was set to 10.