Cost Analysis of a Ray Tracing Algorithm

Bruce Walter
Cornell University

Peter Shirley
Cornell University

July 23, 1997

Abstract

Wait and see.

CR Keywords and Categories: F.2 Analysis of Algorithms and Problem Complexity; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling. I.3.0 [Computer Graphics]: General; I.3.6 [Computer Graphics]: Methodology and Techniques. **Additional Keywords:** Geometric Probability, Ray Tracing.

1 Introduction

Ray casting has become a fundamental step in many graphics algorithms especially in the area of global illumination. Examples include using rays to calculate form factors for traditional radiosity, track energy in particle tracing, and trace possible transport paths in path tracing. The appeal of ray casting is that it offers a flexible and intuitive way to query visibility within a scene and we expect that its popularity will continue to grow. However ray casting can be an expensive operation and it is often the single most expensive part of an algorithm. Thus it is important to study the costs of ray casting both to find ways to minimize its cost and to enable the cost analysis of algorithms which utilize ray casting.

Unfortunately the complexity of ray casting is not very well understood. A variety of acceleration schemes have been proposed by researchers, but there has been relatively little analysis of these schemes. Currently we have no "a priori" assurance about what the actual cost of these schemes will be in real world use. Ray casting studies have provided useful heuristics about the costs of various schemes, however to our knowledge all fall short of completeness in one of the following ways:

- Analyzed impractical methods. Only methods which are used and are competetive are of interest.
- Used assumptions which are usually violated in practice. They may be good approximations but we cannot be sure.

$A \oplus B$	Minkowski Sum: $A \oplus B = \{x \mid x = a + b, a \in A, b \in B\}$.
$A \ominus B$	Shorthand for $A \oplus (-B)$.
$\mathcal{C}$	Set of subdivision cells.
D(A)	The average length of A's projection onto a random line
$E(x)$	The expected value of x.
L(A)	The length of a 1-D point set A
M_i	The multiplicity of o_i.
M	The total of the multiplicities of all objects.
N	The number of objects in $\mathcal{O}$.
N_r	The number of rays.
N_c	The number of cells in $\mathcal{C}$.
$\mathcal{O}$	The set of all objects.
$O(f(N))$	The set of functions of N that are bounded by $kf(N)$ for some $k > 0$ and for all $N > N_0$, where N_0 is a some positive integer.
$p(c_i)$	The probability that a random ray will strike cell i
p_{avg}	The average probability of a cell
p_{tot}	The total probability of all cells. Also the average number of cells struck by a ray.
$p(r)$	The probability density function for rays
$\mathrm{P}_B(A)$	The projection of A onto B
$\mathcal{R}_\mathcal{Z}$	the set of all rays that hit point set $\mathcal{Z}$
SA(A)	The surface area of a 3-D point set A
T_p	The preprocessing time to build efficiency structure.
T_0	The time to initialize a ray
T_s	The time used for ray to step from one cell to another
T_h	The time used to check if a ray intersects an object
V(A)	The the volume of a point set A
c_i	The ith cell in $\mathcal{C}$
o_i	The ith object in $\mathcal{O}$
l	The side length of a cubical cell
$\mathcal{V}$	The bounding volume of subdivision structure
$\delta(A, B)$	Intersect function: one if the two point sets A and B contain at least one point in common and zero otherwise.
κ	The collusion factor between the distributions of objects and rays
$\Theta(c_i)$	Occupancy or the number of objects which intersect cell i
Θ_{avg}	The average occupancy per cell
Θ_{tot}	The total occupancy of all cells

Table 1: Symbols and Terms

- Consisted of empirical studies and thus give no guarantee of future performance.

Our goal in this paper is to provide a more rigorous analysis of a particular scheme, the uniform spatial subdivision, but we emphasize that much more work needs to be done in this area.

After some background on ray casting analysis in section 2, we define the collusion factor and find a high level cost equation in section 3. In sections 4 and **??** we show how to approximate some of the quantities in our high level cost equation. Section ??? demonstrates how compute the optimal level of subdivision and cost in a particular case. We then use our cost equation and some upper bounds in section 8 to derive some complexity order results when the collusion factor is bounded. We include an empirical study in section ??? to illustrate our analysis and to study the collusion factor.

2 Background

Ray casting[1] can be define as, "Given a ray, find the object(s) which it intersects"[2]. The easiest and most naive solution is to simply test the ray against every object in the scene. This is unnecessarily expensive and in practice a variety of schemes are used to reduce the cost (see [1] for a survey). Unfortunately their costs are not well understood, its still a matter of debate as to which scheme is the fastest.

One useful way of comparing algorithms is by expressing the complexity in "big-oh"[3] notation. For instance an algorithm whose cost is linear or proportional to the size of its input is $O(N)$. When using this notation though, we need to remember though that the complexity order is only an upper bound (i.e. any algorithm which is $O(\log N)$ is also $O(N)$) and is asymptopic (i.e. only relevant for sufficiently large problems).

For our analysis let's define N to be the number of objects or primitives (e.g. polygons, spline patches, spheres) in a scene. Its easy to show that the naive ray casting method is $O(N)$. In the rendering literature there is good empirical evidence that at least some of the acceleration schemes achieve sublinear time complexity (i.e. better than $O(N)$), but there is a lack of proofs to show what complexity they actually achieve and under what conditions. In the computational geometry literature there is a method which has been proven to be $O(\log N)$ by de Berg[**?**], but unfortunately its extreme storage requirements make it impractical. Thus to the best of our knowledge, the practical complexity

[1] Some authors make a distinction between *ray casting* and *ray tracing*, where the former is a simple visibility test and the latter involves tracing a ray of light through multiple bounces. To avoid confusion, we will use the restrictive term *ray casting*. In the computational geometry literature the terms *ray shooting* and *stabbing line query* are also used.

[2] Some applications only need the first intersection and others need all. We will try to make our analysis general enough to handle both cases.

[3] $O(f(N))$ is the set of all functions of N bounded above (with equality allowed) by the function $kf(N)$ for some constant k and all $N \geq N'$. So if for some k and N', $g(N) < kf(N)$ for all $N \geq N'$ then $g(N) \in O(f(N))$.

of ray casting is unknown. It is believed to be sub-linear and often assumed to be $O(\log N)$, but except in very specialized cases, the analysis needed to confirm this is lacking.

Of all the acceleration schemes, we have choosen to study the uniform spatial subdivision scheme both because it is amenable to simple analysis techniques and because some researchers have reported it to be the fastest method in practice [10, 12, 5]. It is also probably the best analyzed of all the practical acceleration schemes. Both Cleary and Wyvill [3] and Devillers [4] have derived cost equations under the assumptions that rays are uniformly distributed and that objects are randomly placed. Unfortunately these assumption usually are not valid in practice. Our goal is to show how a cost analysis can be done using much weaker assumptions.

Most current algorithms introduce randomness into the generation of rays to reduce aliasing and bias. Hence in principle the exact rays to be cast and the runtime are non-deterministic. Given this, an expected cost analysis is appealing both for its relative simplicity and because it is a good fit for many modern algorithms. This is the approach used by Cleary and Wyvill [3], Devillers [4], and here. The difficulty is that the most straightforward analysis would require knowledge of the exact probability distribution function for the rays. Unfortunately in practice this can be an extremely difficult function to compute. Previous studies have simply assumed the relatively simple uniform distribution. Our approach is to define a quantity called the collusion factor which encapsolates the relevant details of the ray distribution function. We show that the collusion factor and a few easy to compute statistics are sufficient to bound the cost. Unfortunately computing the exact the collusion factor is still a difficult problem, but we argue that in practice the collusion factor is frequently well approximated as a small constant.

One of the difficulties in trying to formulate a general analysis of ray casting is that there is such a wide variety of possible situations. It is easy to construct adversarial cases which will defeat a particular acceleration scheme. However most of these cases are not typical of the cases usually encountered in practice. Thus its important to find measures which distinguish the adversarial cases. In our case we use the collusion factor and some geometric information about the primitives to this. Thus our cost equation will be given in terms of the total volume, surface area, and average diameter of the primitives and the collusion factor. *Mention something about adversarial cases and the need to have measures to separate them from most typical scenes*

3 Expected Time for Ray Casting

We want to find a general expression for the cost for ray casting using a uniform spatial subdivision. To keep the analysis simple we will only analysze it in its simplest form. Most of the additional optimizations which have been proposed do not change the basic structure of the analysis.

In the naive method, ray casting is accomplished by simply checking the ray

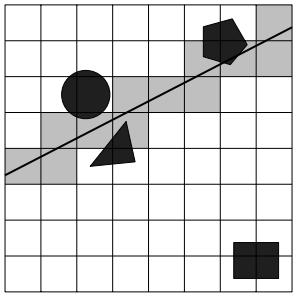

Figure 1: The line steps through twelve cells and six intersection tests are performed (two each for the the circle, triangle, and pentagon).

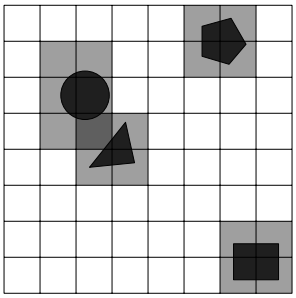

Figure 2: The circle occupies six cells so its multiplicity is six. The other three objects each have a multiplicity of four, so the total multiplicity is eighteen.

for intersection with every object. The most effective acceleration schemes work by finding ways to reduce the number of explicit ray-object intersection tests. The uniform spatial subdivision does this by breaking space up into a regular array of "cells". For each cell we keep a list of all the objects which intersect that cell. When casting a ray we determine which cells the ray passes through and only check the ray against objects associated with those cells. No intersection tests are done for objects which do not share a cell with the ray, but multiple intersection tests may be performed if an object and cell share multiple cells.

In order to formalize this cost we need to introduce a few definitions. Let $\mathcal{O} = \{o_1, o_2, \ldots, o_N\}$ be the set of objects which rays might intersect, $\mathcal{V}$ be a bounding volume of the objects, and $\mathcal{C} = \{c_1, c_2, \ldots, c_{N_c}\}$ be the set of cells which partition $\mathcal{V}$. Then we can write pseudocode for the casting a ray as:

```
Find first cell c_i hit by r
while (c_i is valid and r has not terminated)
    check for intersections of r with objects in c_i
    find next c_i hit by r
```

Let T_0 be the time to initialize a ray, T_h be the time to test an object for intersection with the ray, and T_s the time to step to the next cell. Define the set intersection function $\delta(A, B)$ to be 1 if point sets A and B share at least one point in common and 0 otherwise. Then we can write the cost of a ray as:

$$cost(r) = T_0 + \sum_i^{N_c} \delta(r, c_i) \left(T_s + T_h \sum_j^{N} \delta(o_j, c_i) \right) \tag{1}$$

We can simplify this expression a little by defining $\Theta(c_i)$, the *occupancy* of a cell, to be the number of objects which intersect it.

$$\Theta(c_i) = \sum_j^{N} \delta(o_j, c_i) \tag{2}$$

We take the expected value of the cost by noting that the expected value of $\delta(r, c_i)$ is just $p(c_i)$, the probability that c_i will be struck by a random ray. (An example of the formally correct way to do this is given in section???.) Thus we can write the expected cost per ray as:

$$E(cost(r)) = T_0 + \sum_i^{N_c} p(c_i) T_s + \sum_i^{N_c} p(c_i) \Theta(c_i) T_h \tag{3}$$

Note that the first term is a constant, the second is the cost of stepping through cells, and the third represents the cost of explicit intersection tests. Any cost savings will come from decreases in the third term as we increase the number of cells. However it is possible to create situations where the third term will not decrease significantly and our cost saving will not materialize. We need to make

a few more definitions to help discuss ways in which the acceleration scheme can fail.

We define the total occupancy and total cell probabilities to be the sum of all the individual cell occupancies and probabilities, and define their averages in the usual way.

$$\Theta_{tot} = \sum_{i}^{N_c} \Theta(c_i) \tag{4}$$

$$\Theta_{avg} = \frac{\Theta_{tot}}{N_c} \tag{5}$$

$$p_{tot} = \sum_{i}^{N_c} p(c_i) \tag{6}$$

$$p_{avg} = \frac{p_{tot}}{N_c} \tag{7}$$

We define κ, the collusion factor between the cell occupancies and cell probabilities as:

$$\kappa = \frac{1}{N_c} \sum_{i}^{N_c} \frac{p(c_i)}{p_{avg}} \frac{\Theta(c_i)}{\Theta_{avg}} \tag{8}$$

By noting that all these quantities are positive and that $\Theta(c_i) \leq \Theta_{tot}$ we easily show the bounds:

$$0 \leq \kappa \leq N_c \tag{9}$$

We can now rewite our expected cost as:

$$E(cost(r)) = T_0 + p_{tot} T_s + p_{avg} \Theta_{tot} \kappa T_h \tag{10}$$

We now have the cost expressed in terms of three unknown quantities; the total occupancy, the total cell probability, and the collusion factor. To complete our analysis we need to find both good approximations and worst case upper bounds for these quantities. In later sections we will show how to do this for the total occupancy and cell probability. Unfortunately calculating the collusion factor is a more difficult problem and usually requires detailed information about the exact scene and algorithm being used. In this paper we will argue that in practice the collusion factor is roughly constant and support this contention with some empirical studies. Better theoretical calculations of the collusion factor are left as future work.

3.1 Total Cost

To evaluate how quickly the third term decreases we need to be able to evaluate, total occupancy, average cell probabilities, and the collusion factor. In the next two sections we will show how to calculate good approximations for Θ_{tot} and p_{avg}. In general the analysis of κ, the collusion factor, is more difficult and requires knowledge of the specific scene and application being used. However

we believe that it is usually well behaved in practice and often can be well approximated as a constant. We will show that this is at least true for some simple cases and in our examples.

Any complete analysis of the benefits of the uniform spatial subdivision must include T_p , the preprocessing time necessary to build it. Thus if we cast N_r rays our total cost will be:

$$totalcost = T_p + N_r E(cost(r)) \tag{11}$$

For most of our analysis we will assume that N_r is large enough that the preprocessing step is not the dominant term in total cost equation. However this needs to be verified when analyzing a particular application.

4 Occupancy and Multiplicity

To help us understand occupancy we need to introduce the related concept of *multiplicity*. The multiplicity of an object is the number of cells which it intersects given by:

$$M_j = \sum_{i}^{N_c} \delta(c_i, o_j) \tag{12}$$

The total multiplicity is then the sum of the multiplicities of all objects in a scene. It is easy to show that the total occupancy is the same as the total multiplicity.

$$M = \sum_{j}^{N} M_j = \sum_{i}^{N_c} \sum_{j}^{N} \delta(c_i, o_j) = \Theta_{tot} \tag{13}$$

Ideally each object would be contained in exactly one cell (i.e. $M_j = 1$) and our total occupancy would be just N. Unfortunately objects may lie across cell boundaries and intersect multiple cells, especially if we use small cells. In this section we show two ways to calculate the size of this effect. First we find the expected value for Θ_{tot} assuming our objects are randomly placed and oriented. This result has been used in previous studies and is usually a good approximation. However real models are usually constructed in a deliberate and non-random manner, thus the assumption that objects are randomly positioned is usually a false one. Thus we derive a new result, by finding a bound on the total occupancy assuming worst case positioning of the objects.

4.1 Minkowski Sums

We can use Minkowski Sums [4] to calculate the average multiplicity of an object. Given two point sets A and B, we define the Minkowski Sum $\oplus$ by:

$$A \oplus B = \{x | x = a + b, a \in A, b \in B\} \tag{14}$$

[4] Minkowski Sums are also referred to as augmented volumes, mixed volumes, mixed sets, and vector sums (e.g. Cleary and Wyvill [3], Benson [2], Santaló [8], and Latombe [7]).

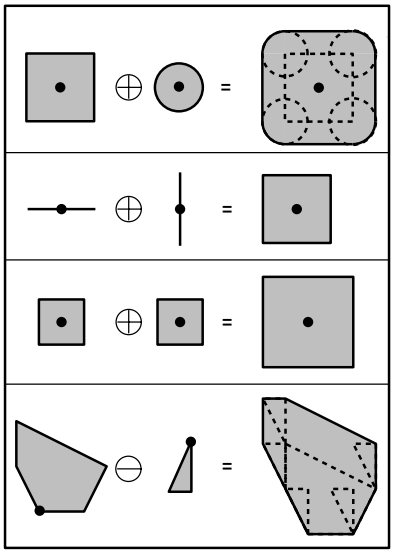

Figure 3: Some Minkowski Sums

where $a + b$ denotes the standard component-wise addition of points as vectors in $\Re^n$. The result of $A \oplus B$ can be thought of as placing a copy of B at every point in A (see figure 3). Minkowski Sums have the following useful properties:

- Minkowski Sums are commutative and associative.
- If A and B are convex then $A \oplus B$ is also convex.
- If A' is a translation of A then $A' \oplus B$ is a translation of $A \oplus B$ by the same amount, and if P is a point then $A \oplus P$ is a translation of A by P.

Minkowski Sums have been used previously in motion planning (e.g. [7]) and ray casting analysis (e.g. [3]), because they transform the difficult problem of determining if two point sets intersect into the easier problem of determining if a point is in a point set. It follows directly from the definition that A intersects B if and only if the sum[5] $A \ominus B$ contains the origin, and that for any point P, A intersects $B \oplus P$ if and only if $P \in (A \ominus B)$. Thus we can interpret $A \ominus B$ as the set of all translations of B such that it intersects A.

4.2 The Expected Case Multiplicity

Let o_j be a randomly positioned object such that at least part of o_j is contained in the bounding volume $\mathcal{V}$. This is equivalent to saying that $o_j = A_j \oplus P$ where A_j has the same shape as o_j and $P \in (\mathcal{V} \ominus A_j)$. If we assume that all such

[5] $A \ominus B$ is shorthand for $A \oplus (-B)$.

values of P are equally likely, and use volume as a measure for 3-D point sets, then the probability that o_j intersects cell c_i is:

$$E(\delta(c_i, o_j)) = \frac{\mathrm{V}(c_i \ominus o_j)}{\mathrm{V}(\mathcal{V} \ominus o_j)} \tag{15}$$

If we assume that o_j is small compared to $\mathcal{V}$ (i.e. $\mathrm{V}(o_j) \ll \mathrm{V}(\mathcal{V})$) then $\mathrm{V}(\mathcal{V} \ominus o_j) \approx \mathrm{V}(\mathcal{V})$. Summing over all $N_c = \mathrm{V}(\mathcal{V})/\mathrm{V}(c_i)$ cells we get:

$$E(M_j) \approx \frac{\mathrm{V}(c_i \ominus o_j)}{\mathrm{V}(c_i)} \tag{16}$$

Note that most definitions of random placement would require that o_j be contained in $\mathcal{V}$, not just intersect it. However under our assumption that o_j is small compared to $\mathcal{V}$, the discrepancy between these two definitions will be small.

4.3 The Volume of a Minkowski Sum

To evaluate $E(M_j)$ we must find the volume of a Minkowski Sum. To make this tractable we will assume that the objects are either convex or that we use their convex hulls to estimate their occupancy. If we assume that our cells are axis aligned boxes with side lengths ℓ_x , ℓ_y , and ℓ_z , then we can simplify the problem by noting that a box can be written as the Minkowski Sum of three lines parallel to the each of the axes with lengths ℓ_x , ℓ_y , and ℓ_z .

$$c_i = \ell_x \oplus \ell_y \oplus \ell_x \tag{17}$$

We can use associativity to write:

$$\mathrm{V}(c_i \ominus o_j) = \mathrm{V}(\ell_x \oplus (\ell_y \oplus (\ell_z \ominus o_j))) \tag{18}$$

where we now only need to compute the Minkowski Sum of a line segment and a general convex set.

Define $\mathrm{P}_{\ell_a}(A)$ to be the projection of A onto the line ℓ_a, and $\mathrm{P}_{\sigma_a}(A)$ to be the projection of A onto the plane σ_a. Let ℓ_a , ℓ_b , and ℓ_c be three mutually perpendicular lines, σ_b and σ_b be planes perpendicular to ℓ_a and ℓ_b respectively, A be a line segment parallel to ℓ_a , and B be a convex bounded set. Then using geometric constructions the following equations can be derived.

$$\begin{aligned} \mathrm{V}(A \oplus B) &= \mathrm{L}(A)\mathrm{A}(\mathrm{P}_{\sigma_a}(B)) + \mathrm{V}(B) \\ \mathrm{A}(\mathrm{P}_{\sigma_b}(A \oplus B)) &= \mathrm{L}(A)\mathrm{L}(\mathrm{P}_{\ell_c}(B)) + \mathrm{A}(\mathrm{P}_{\sigma_b}(B)) \end{aligned}$$

Applying this to Equation 18 and writing $\mathrm{P}_x(A)$ and $\mathrm{P}_{xy}(A)$ when we mean projection onto a line parallel to the x axis and a plane parallel to the x-y plane, we get:

$$\begin{aligned} V(c_i \ominus o_j) = \quad & \ell_x\ell_y\ell_z + \ell_x\ell_y\mathrm{L}(\mathrm{P}_z(o_j)) + \ell_x\mathrm{L}(\mathrm{P}_y(o_j))\ell_z \\ & + \mathrm{L}(\mathrm{P}_x(o_j))\ell_y\ell_z + \ell_x\mathrm{A}(\mathrm{P}_{yz}(o_j)) \\ & + \ell_y\mathrm{A}(\mathrm{P}_{xz}(o_j)) + \ell_z\mathrm{A}(\mathrm{P}_{xy}(o_j)) + \mathrm{V}(o_j). \end{aligned} \tag{19}$$

One interesting question that we can consider here is what is the ideal shape to make our cells. If we assume our objects have no preferred axis, or in otherwords we expect their projections onto the x, y, and z axes to be the same on average, then we can easily show that we minimize the total multiplicity by using cubical cells.

We can write an simplified equation by assume that the objects are randomly oriented and our cells are cubes with length l (i.e. $l = \ell_x = \ell_y = \ell_z$). Let $\mathrm{D}(B)$ be the expected length of B's projection onto a random line, $\mathrm{SA}(B)$ be the surface area of B, and note that expected area of a convex set B projected onto a random plane is $\mathrm{SA}(B)/4$. The expected value for the multiplicity will be

$$E(M_j) = 1 + \frac{3\mathrm{D}(o_j)}{l} + \frac{3\mathrm{SA}(o_j)}{4l^2} + \frac{\mathrm{V}(o_j)}{l^3}$$

Since the set of cells partition $\mathcal{V}$, we know that the sum of the volume of the cells equals the volume of $\mathcal{V}$. Thus if our cells are cubes with volume l^3 we can easily calculate:

$$l = \sqrt[3]{\frac{\mathrm{V}(\mathcal{V})}{N_c}} \tag{20}$$

Using this and summing over all objects we get:

$$E(\Theta_{tot}) = E(M) = N + \frac{3\mathrm{D}(\mathcal{O}){N_c}^{\frac{1}{3}}}{\mathrm{V}(\mathcal{V})^{\frac{1}{3}}} + \frac{3\mathrm{SA}(\mathcal{O}){N_c}^{\frac{2}{3}}}{4\mathrm{V}(\mathcal{V})^{\frac{2}{3}}} + \frac{\mathrm{V}(\mathcal{O})N_c}{\mathrm{V}(\mathcal{V})} \tag{21}$$

4.4 The Worst Case Multiplicity

In real scenes objects are usually organized in a highly non-random manner, so the analysis of the expected case does not directly apply. Fortunately we can calculate a bound on the multiplicity even assuming worst case positioning of the objects. Let the cells be axis aligned boxes with side lengths ℓ_x, ℓ_y, and ℓ_z, and o_j be a randomly placed object as above. Choose a point P' such that $P' \oplus o_j$ has worst case occupancy. Let $A_{box} = \{(x, y, x) | 0 \leq x \leq \ell_x, 0 \leq y \leq \ell_y, 0 \leq z \leq \ell_z\}$. By the symmetry in regular layout of cells there are many such P' values to choose from, and moreover there exist at least one point P' such that $P' \in A_{box}$.[6] Since $P' \oplus o_j \subset A_{box} \oplus o_j$, the maximum occupancy of o_j is bounded by the minimum occupancy of $A_{box} \oplus o_j$. Thus we have:

$$\max(M_j) \leq E(M(A_{box} \oplus o_j)) \leq \frac{V(c_i \oplus c_i \oplus o_j)}{V(c_i)},$$

where we used the fact that A_{box} has the same shape as c_i. The Minkowski Sum of c_i with itself will be another axis aligned box with side lengths $2\ell_x$, $2\ell_y$, and $2\ell_z$. Worst case orientation can also be accounted for by noting that for any

[6] For simplicity we are here approximating $\mathcal{V}$ as being infinite, since this will not change the worst case multiplicity

convex set A, $\mathrm{L}(\mathrm{P}_\ell(A)) \leq \frac{\pi}{2}\mathrm{D}(A)$ and $\mathrm{A}(\mathrm{P}_\sigma(A)) \leq \frac{1}{2}\mathrm{SA}(A)$ for any line ℓ and plane σ. Following Equation 19 we can write this for cubical cells as::

$$\max(M_j) \leq 8 + \frac{6\pi \mathrm{D}(o_j)}{l} + \frac{3\mathrm{SA}(o_j)}{l^2} + \frac{\mathrm{V}(o_j)}{l^3},$$

And in general we have:

$$\max(M) \leq 8\, E(M)$$

Thus worst case positioning is no more than a factor of 8 worse than random positioning, and thus will have the same complexity order as the expected occupancy.

5 Cell Probability

The total cell probability, p_{tot}, is a simple abstract quantity that tell us how often cells are accessed. In fact $p_{tot}$is exactly the number of cells intersected by a random ray on average. We can calculate p_{tot}without knowing anything about the precise distribution of the rays because we have hidden this detailed information inside the collusion factor. Here we will take two approaches to estimating and bounding p_{tot}, by approximating rays as uniformly distributed lines and by using our occupancy results.

5.1 Uniform Lines

We can turn a ray into a line by ignoring its starting point (and ending point if it has one). Its clear that any cell intersected by a ray will also be intersected by its corresponding ray. Thus we can use lines to get a usually conservative estimate for the probability of a cell. One especially simple method is to approximate our rays as uniformly distributed lines. Here uniform distribution means that the probability of a line does not depend on the position or orientation of the line. We can then calculate the probability that line will strike a particular cell given that it strikes the bounding volume $\mathcal{V}$ (e.g. [**?**]).

$$p(c_i) = \frac{\mathrm{SA}(p(c_i))}{\mathrm{SA}(\mathcal{V})} \tag{22}$$

Since all cells are identical, this means that the probability of all cells are the same. This is a somewhat surprising result, but it follows directly from the symmetries of the uniform distribution. It is also interesting to note that this equation indicates that we want to minimize the surface area of our cells which usually means we want them to be cubes. We can sum over all cells to find the total probability.

$$p_{tot} = \frac{6\mathrm{V}(\mathcal{V})^{\frac{2}{3}} {N_c}^{\frac{1}{3}}}{\mathrm{SA}(\mathcal{V})} \tag{23}$$

where we have used Equation 20 for the side length of a cubical cell. This provides a simple estimate without knowing anything about the particular application. An upper bound can also easily be derived by noting that the cells

are formed from the bounding volume by a set of cutting planes. A ray can only enter a new cell by crossing a cutting plane and it can only cross each cutting plane once. However for many applications this way of calculating $p_{tot}$is too conservative, especially if we only want the first intersection for each ray.

5.2 Average Length of a Ray

Cleary and Wyvill [3] noted that average number of cells intersected by a ray is just a multplicity problem. Thus we can use Equation??? to find the average number cells a ray intersects. Note that a ray of length s has average diameter of $s/2$, zero surface area, and zero volume.

$$E(p_{tot}) = 1 + \frac{3L_r N_c^{\frac{1}{3}}}{2\mathrm{V}(\mathcal{V})^{\frac{1}{3}}} \tag{24}$$

where $L_r$is the average length of a ray and we have used Equation 20 for cubical cells.

We can easily extend this to a worst case value by replacing the average ray length with the maximum ray length and by using the fact that the worst case multiplicity is at most eight times the expected multiplicity (Equation??).

6 Collusion Factor

The collusion factor, κ, is the last piece of information we need to complete the cost analysis and unfortunately its also the most difficult to calculate. It requires detailed information about the precise distribution of rays and objects. Except in special cases (e.g. uniformly distributed rays) calculating the collusion factor is prohibitively difficult. We can easily find it "post facto" by keeping some statistics, but this is of little help trying to predict costs.

Our way out of this dilemma is to argue that in practice the collusion factor is usually a small constant. We will try to back up this assertion in two ways. Later in this paper we do an empirical study to find the collusion factor for a few example scenes and algorithms. This will show that its well behaved at least for a few reasonable complex examples. Also we will discuss under what conditions the collusion factor will blow up. This should help the reader decide if the collusion factor is likely to be a problem for their particular application.

6.1 Relation between Collusion and Correlation

show the the collusion is

$$\kappa = \frac{\sigma_p}{p_{avg}} \frac{\sigma_o}{\Theta_{avg}} r \tag{25}$$

and this means it will only be large if both the rays and the objects are non-uniformly distributed and there is a strong correlation between their distributions.

7 Completed Cost Equation

Now we can use our estimates for p_{tot} and Θ_{tot} in our cost equation. By combining Equations 10 21 24, we get:

$$E(T(r)) = T_0 + \left[1 + \frac{3L_r N_c^{\frac{1}{3}}}{2\mathrm{V}(\mathcal{V})^{\frac{1}{3}}}\right] T_s + \left[1 + \frac{3L_r N_c^{\frac{1}{3}}}{2\mathrm{V}(\mathcal{V})^{\frac{1}{3}}}\right] \frac{\kappa T_h}{N_c} \left[N + \frac{3\mathrm{D}(\mathcal{O}) N_c^{\frac{1}{3}}}{\mathrm{V}(\mathcal{V})^{\frac{1}{3}}} + \frac{3\mathrm{SA}(\mathcal{O}) N_c^{\frac{2}{3}}}{4\mathrm{V}(\mathcal{V})^{\frac{2}{3}}} + \frac{\mathrm{V}(\mathcal{O}) N_c}{\mathrm{V}(\mathcal{V})}\right] \quad (26)$$

If you assume that κ is some small constant then you can use this equation to solve for the value of N_c which minimizes the cost. One way to do this is to approximate N_c as a continuous variable, differentiate the cost with respect to N_c, and set the derivative equal to zero. The optimal discrete value of N_c will be close to its optimal continuous value. In the general case this involves solving for the roots of a fourth order polynomial. This can be done analytically, but in practice we may not actually want to use the analytic minimum. The cost function is usually quite flat near the minimum and we would prefer to use fewer cells in order to reduce the memory usage and corresponding cache misses. Thus you may want to use an iterative solver to find smallest value for N_c which gets you sufficiently close to the analytic minimum cost.

8 Order Statistics

The biggest advantage our analysis has over previous ones is that we have some worst case bounds which we can use to derive complexity order statistics. Previously we showed that the worst case values for Θ_{tot} and p_{tot} are at most eight times the expected values given in Equations 21 and 24. The only quantity which we don't have a worst case bound for is κ, but if we assume that the collusion factor is roughly constant then we can bound the expected cost and compute the complexity statistics for the expected cost of ray casting.

Unfortunately the complexity of ray casting with a uniform spatial subdivision depends on how the scene geometry scales as we move to more complex scenes. To quantify this we introduce values e_D, e_{SA}, e_V, and $e_L$to measure how the total diameter, total surface area, total volume, and average ray length scale with increasing complexity. We define these as follows:

$$\frac{\mathrm{D}(\mathcal{V})}{\mathrm{V}(\mathcal{V})^{\frac{1}{3}}} \in O(N^{e_D})$$

$$\frac{\mathrm{SA}(\mathcal{V})}{\mathrm{V}(\mathcal{V})^{\frac{2}{3}}} \in O(N^{e_{SA}})$$

$$\frac{\mathrm{V}(\mathcal{V})}{\mathrm{V}(\mathcal{V})} \in O(N^{e_V})$$

$$\frac{L_r}{\mathrm{V}(\mathcal{V})^{\frac{1}{3}}} \in O(N^{e_L})$$

Now using the assumption that κ is $O(1)$ and the worst case equations for Θ_{tot} and p_{tot}, we find the cost is:

$$E(T(r)) \in O(1+N_c{}^{\frac{1}{3}}N^{e_L})O(1+N_c{}^{-1}N+N_c{}^{\frac{-2}{3}}N^{e_D}+N_c{}^{\frac{-1}{3}}N^{e_{SA}}+N^{e_V}) \quad (27)$$

We can solve this for the order for $N_c$which minimizes the cost and the corresponding course by a tedious case analysis of possible dominant terms. There turn out to be twelve different possible pairs of dominant terms. The results of the case analysis are listed in Table 2. Note that we have added a thirteenth case for conditions where our analysis indicates that the naive method will outperform the uniform spatial subdivision.

8.1 How general is this result?

The fundamental assumption underlying this result is that the collusion factor, κ, can be well approximated as a constant[7]. We have shown that $\kappa = 1$ for uniformly distributed lines. It can easily be shown that $\kappa = 1$ if the objects are uniformly distributed. But even for more typical scenes with non-uniform distributions of rays and objects, κ will still be approximately constant as long as there is not a strong correlation between the two distributions.

We cannot prove that this will always be the case as it is possible to construct scenarios where κ is not constant. One easy way to do this is to construct a scene where the objects are aligned along a line and then arrange the rays so that most of them hit most of the objects. However it is our contention that these are pathological cases, and that most scenes encountered in practice are well behaved with κ approximately constant. We hope that our examples later in the paper show that this is at least a plausible claim. However its left as future work to show that its validity.

8.2 Storage

We can also analyze the storage used by the uniform spatial subdivision. For each cell we need to store a value to show if the cell contains any objects and for each cell which contains objects we need to store a list of objects whose length is given by the occupancy of the cell. Thus the storage required for the uniform spatial subdivision is $O(N_c + \Theta_{tot})$. We can expand this using the notation in this section to get:

$$storage \in O(N_c + N + N_c{}^{\frac{1}{3}}N^{e_D} + N_c{}^{\frac{2}{3}}N^{e_{SA}} + N_cN^{e_V}) \quad (28)$$

This can then be evaluated using the same case analysis technique and the results are listed in Table 2.

[7]The formal requirement is that κ must be $O(1)$ with respect to N and N_c or in other words, κmust be bounded above by some constant.

case	N_c	$E(T(r))$	storage	conditions
1	$O(N)$	$O(N^{\frac{1}{3}+e_L})$	$O(N)$	$e_D \leq \frac{2}{3}$, $e_{SA} \leq \frac{1}{3}$, $e_V \leq 0$, $e_L \geq \frac{-1}{3}$, $e_L \leq \frac{2}{3}$
2	$O(N)$	$O(1)$	$O(N)$	$e_D \leq \frac{2}{3}$, $e_{SA} \leq \frac{1}{3}$, $e_V \leq 0$, $e_L \leq \frac{-1}{3}$
3	$O(N^{\frac{3}{2}e_D})$	$O(N^{\frac{1}{2}e_D+e_L})$	$O(N^{\frac{3}{2}e_D})$	$e_D \geq \frac{2}{3}$, $e_{SA} \leq \frac{1}{2}e_D$, $e_V \leq 0$, $e_L \geq \frac{-1}{2}e_D$, $e_L \leq 1-\frac{1}{2}e_D$
4	$O(N^{\frac{3}{2}e_D})$	$O(1)$	$O(N^{\frac{3}{2}e_D})$	$e_D \geq \frac{2}{3}$, $e_{SA} \leq \frac{1}{2}e_D$, $e_V \leq 0$, $e_L \leq \frac{-1}{2}e_D$
5	$O(N^{1-e_V})$	$O(N^{\frac{1+2e_V}{3}+e_L})$	O(N)	$e_D \leq \frac{2+e_V}{3}$, $e_{SA} \leq \frac{1+2e_V}{3}$, $e_V \geq 0$, $e_L \geq -\frac{1-e_V}{3}$, $e_L \leq \frac{2(1-e_V)}{3}$
6	$O(N^{1-e_V})$	$O(N^{e_V})$	O(N)	$e_D \leq \frac{2+e_V}{3}$, $e_{SA} \leq \frac{1+2e_V}{3}$, $e_V \geq 0$, $e_L \leq -\frac{1-e_V}{3}$, $e_V \leq 1$
7	$O(N^{\frac{3}{2}(e_D-e_V)})$	$O(N^{\frac{e_D+e_V}{2}+e_L})$	$O(N^{\frac{3e_D-e_V}{2}})$	$e_D \geq \frac{2+e_V}{3}$, $e_{SA} \leq \frac{e_D+e_V}{2}$, $e_V \geq 0$, $e_L \geq -\frac{e_D-e_V}{2}$, $e_L \leq \frac{2-e_D-e_V}{2}$
8	$O(N^{\frac{3}{2}(e_D-e_V)})$	$O(N^{e_V})$	$O(N^{\frac{3e_D-e_V}{2}})$	$e_D \geq \frac{2+e_V}{3}$, $e_{SA} \leq \frac{e_D+e_V}{2}$, $e_V \geq 0$, $e_L \leq -\frac{e_D-e_V}{2}$, $e_V \leq 1$
9	$O(N^{\frac{3}{2}(1-e_{SA})})$	$O(N^{e_{SA}})$	$O(N)$	$e_D \leq \frac{1+e_{SA}}{2}$, $e_{SA} \geq \frac{1}{3}$, $e_V \leq \frac{3e_{SA}-1}{2}$, $e_L \geq \frac{e_{SA}-1}{2}$, $e_L \leq 1-e_{SA}$
10	$O(N^{3e_{SA}})$	$O(N)$	$O(N^{3e_{SA}})$	$e_D \leq 2e_{SA}$, $e_{SA} \geq \frac{1}{3}$, $e_V \leq 0$, $e_L \leq -e_{SA}$
11	$O(N^{3(e_D-e_{SA})})$	$O(N^{e_{SA}+e_L})$	$O(N^{2e_D-e_{SA}})$	$e_D \geq \frac{1+e_{SA}}{2}$, $e_{SA} \geq \frac{1}{2}e_D$, $e_V \leq 2e_{SA}-e_D$, $e_L \geq e_{SA}-e_D$, $e_L \leq 1-e_{SA}$
12	$O(N^{3(e_{SA}-e_V)})$	$O(N^{e_V})$	$O(N^{3e_{SA}-e_V})$	$e_D \leq 2e_{SA}-e_V$, $e_{SA} \geq \frac{1+2e_V}{3}$, $e_V \geq 0$, $e_L \leq e_V-e_{SA}$, $e_V \leq 1$
13	$O(1)$	$O(N)$	$O(1)$	$e_V \geq 1$ or $e_L \geq \frac{2}{3}$ or $e_D+e_L \geq 2$ or $e_{SA}+e_L \geq 1$ or $2e_V+3e_L \geq 2$ or $e_D+e_V+2e_L \geq 2$

Table 2: Order Results for Ray Casting

8.3 Preprocessing Time

When analyzing an algorithm which uses a uniform spatial subdivision, we need to include T_p, the preprocess time to build the subdivision. This involves creating and initializing the cells and then for each object finding the cells it intersects and adding it to the associated cell lists. Thus we have $T_p \in O(N_c + \Theta_{tot})$. This is the same order as the storage which we already analyzed in the previous section. To complete the analysis we need to analze the order of N_r for the particular algorithm. Then we can plug these results into Equation??? to find the total complexity of ray casting for a particular algorithm.

9 Example Scene Scalings

Finding the appropiate values for e_D, e_{SA}, and e_V will depend on the particular application which you are analyzing. To give a flavor of how this can be done, we will work through two illustrative examples: tesselation with triangles and non-intersecting spheres.

9.1 Tesselation

Suppose an environment of piece-wise smooth surfaces with total area A is approximated using a set $\mathcal{O}$ of N convex polygons. If the vertices of each polygon lie on a single such surface, then the total area of the polygons will be bounded by A. If we assume that the interior angles of the polygons are restricted to be above some threshold then we find that $\mathrm{D}(o_j) = O(\sqrt{N})$ by applying Lagrange multipliers subject to the constraint on total area. Since polygons have zero volume we know that $e_D \leq \frac{1}{2}$, $e_{SA} = 0$, and $e_V = 0$. In addition let's assume that the average length of a ray does not change (i.e. $e_L = 0$). We can look this up in Table 2 and see that this falls into case 1. This we can cast rays in $O(N^{\frac{1}{3}})$ time using $O(N)$ space.

9.2 Non-intersecting spheres

put a discussion of non-intersection spheres and k-fat objects here.

10 Experimental results

put total cost analysis expression here? do experimental analysis of κ and p_{avg} and Θ_{tot} and time and storage.

References

[1] J. Arvo and D. Kirk. A survey of ray tracing acceleration techniques. In A. S. Glassner, editor, *An Introduction to Ray Tracing*. Academic Press, San Diego, CA, 1989.

[2] R. V. Benson. *Euclidean Geometry and Convexity*. McGraw-Hill, New York, NY, 1966.

[3] J. Cleary and G. Wyvill. An analysis of an algorithm for ray tracing using uniform space subdivision. *The Visual Computer*, 4:65–83, 1988.

[4] O. Devillers. Tools to study the efficiency of space subdivision structures for ray tracing. In *Second Annual Conference on Computer Graphics in Paris*, pages 467–481, September 1989.

[5] F. W. Jansen. Comparision of ray traversal methods. *Ray Tracing News*, 7(2), February 1994.

[6] J. T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986. ACM Siggraph '86 Conference Proceedings.

[7] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic, Boston, MA, 1991.

[8] L. A. Santaló. *Integral Geometry and Geometric Probability*. Encyclopedia of Mathematics and its Applications. Addison-Wesley, Reading, MA, 1976.

[9] B. E. Smits, J. R. Arvo, and D. P. Greenberg. A clustering algorithm for radiosity in complex environments. *Computer Graphics*, 28(3):435–442, July 1994. ACM Siggraph '94 Conference Proceedings.

[10] J. M. Snyder and A. H. Barr. Ray tracing complex models containing surface tessellations. *Computer Graphics*, 21(4):119–128, August 1987. ACM Siggraph '87 Conference Proceedings.

[11] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

[12] A. Woo. Ray tracing polygons using spatial subdivision. In *Proceedings of Graphics Interface '92*, pages 184–191, June 1992.