情報処理学会論文誌 Vol.55 No.12 2559–2568 (Dec. 2014)

レイサンプリングによる効率的な分割統治法を用いたレイトレーシング

名畑 豪祐[1,a)] 岩崎 慶[1] 土橋 宜典[2] 西田 友是[3]

受付日 2014年4月8日, 採録日 2014年9月12日

概要：本論文では，分割統治法を用いた効率的なレイトレーシングを提案する．Divide-And-Conquer Ray Tracing（DACRT）は，大量のレイと幾何プリミティブ間の交差判定問題を，容易に解くことができるまで交差判定問題を分割することで解く手法である．先行研究では，プリミティブの分布のみに基づき問題を分割しており，レイの分布は考慮していなかった．そのため，追跡するレイの数が多い高解像度画像のレンダリングにおいて非効率な場合がある．そこで，本研究ではレイの分布に基づき問題の分割，つまり高速化データ構造の構築，探索順序の決定を行う．また，レイトラバーサルを効率的に行うために，新たなコストメトリックを提案する．これにより，レイの数について問題サイズを十分に削減できない場合の非効率な問題分割を回避できる．提案するコストメトリックは計算，実装が容易に行える．本研究により，あらゆる種類のレイで高速化を実現し，先行研究と比較して最大約 2 倍の高速化を達成した．さらに，レイとバウンディングボリュームの交差判定回数を削減できることを示した．

キーワード：レイトレーシング，分割統治法，レイサンプリング

An Acceleration Method for Divide-And-Conquer Ray Tracing Using Ray Sampling

Kosuke Nabata[1,a)] Kei Iwasaki[1] Yoshinori Dobashi[2] Tomoyuki Nishita[3]

Received: April 8, 2014, Accepted: September 12, 2014

Abstract: In this paper, we propose an efficient ray tracing method based on divide-and-conquer approach. Previous divide-and-conquer ray tracing (DACRT) methods subdivide intersection problems between large numbers of rays and primitives, by recursively subdividing the problem size based on the distribution of primitives only, until it can be easily solved. To subdivide the intersection problems, the lack of considering the distribution of rays can decrease the rendering performance, especially for high resolution images with antialiasing. To address this problem, our method exploits the distribution of rays for subdivision of intersection problems, construction of acceleration data structure, and determination of ray traversal order. To accelerate ray traversals, we have derived a new cost metric which is used to avoid inefficient subdivision of the intersection problem where the number of rays is not sufficiently reduced at the cost of expensive intersection tests between rays and bounding volumes. The cost metric is easy to calculate and hence easy to implement. Our method can accelerate the tracing of many types of rays by a factor of up to two. In addition, we show that our method can reduce the number of intersection tests between rays and bounding volumes compared to the state-of-the-art DACRT method.

Keywords: ray tracing, divide-and-conquer algorithm, ray sampling

[1] 和歌山大学
Wakayama University, Wakayama 640–8510, Japan
[2] 北海道大学
Hokkaido University, Sapporo, Hokkaido 060–0808, Japan
[3] 広島修道大学/UEI リサーチ
Hiroshima Shudo University/UEI Research, Hiroshima 731–3195, Japan
[a)] s141044@center.wakayama-u.ac.jp

1. はじめに

近年のレイトレーシングに関する研究により，高速化データ構造（格子 [1]，kd 木 [2]，階層バウンディングボリューム（Bounding Volume Hierarchy，BVH）[3] など）を用いることで静的なシーン，動的なシーンの両方でイン

タラクティブなレンダリングが可能となっている．これらの手法では，レイトレーシングを行う前にあらかじめ高速化データ構造を構築しておき，レイトレーシング時には，保持しておいた高速化データ構造を参照し，効率的にレイとプリミティブの交差判定を行う．

近年，高速化データ構造の構築とレイトレーシングを同時に行うことで，高速化データ構造のための追加のメモリが不要な DACRT（Divide-And-Conquer Ray Tracing）が提案された [4], [5], [6], [7]．この手法では，大量のレイとプリミティブの交差判定問題を分割統治法を用いて解いている．分割統治法とは，そのままでは解けないような問題を小さな問題へ分割し，分割された問題を解くことで元の問題を解く方法である．DACRT において解く問題とは，すべてのレイに対して最近傍の物体との交点を求めることである．また，問題サイズはレイの数とプリミティブの数にあたる．先行研究 [4], [5], [6], [7] では，交差判定問題をプリミティブの分布にのみ基づき分割しており，レイの分布を考慮していなかった．そのため，レイの数について問題サイズの大きな削減を行えないことがあり，高解像度画像のように大量のレイが必要な場合では非効率である．近年，高解像度画像の需要は高まってきており，レイの分布を考慮しレイの数について問題を効率的に分割する方法が必要である．

本研究では，レイの集合からレイを標本抽出し，抽出したレイからレイの分布を推定する．これをレイサンプリングと呼び，レイの分布を基に，高速化データ構造を構築することで，プリミティブの数，レイの数について問題サイズを大きく削減することができる．しかし，レイの分布を考慮した高速化データ構造を使用した場合でもレイの数について問題サイズを大きく削減できない場合がある．レイの数について問題を分割するには，分割されたプリミティブ集合を包含するバウンディングボリュームとレイ集合の交差判定が必要であるため，この場合，不要な交差判定が多数行われることになる．この問題を解決するために，新たなコストメトリックを導入し，不要な交差判定が行われるのを避ける方法を提案する．また，レイサンプリングにより得た情報を使用し，レイ全体に適した高速化データ構造の探索順序の決定を行う方法を提案する．提案法では，コヒーレンスの高いプライマリレイ（視点からのレイ），コヒーレンスの低いセカンダリレイ（視点からのレイが物体表面で反射したレイ），コヒーレンスがほとんど存在しないランダムレイ（複数回の反射を経たレイ，ランダムな方向に反射したレイ）で高速化を実現し，最大約 2 倍の高速化を達成した．なお，本論文は，同著者らの論文 [8] に，レイのサンプリング方法についての実験を追加し，効率的なレイのサンプリング方法を提案したものである．さらに，レイとバウンディングボリュームの交差判定回数，レイと三角形の交差判定回数について計測し，提案法の問題分割手法が従来の問題分割手法よりも優れていることを定量的に示した．

2. 先行研究

一般的なレイトレーシングでは，レイトレーシングを行う前に高速化データ構造（格子 [9], kd 木 [10], 階層バウンディングボリューム（Bounding Volume Hierarchy，BVH）[11] など）を構築し，レイトレーシング時に，保持しておいた高速化データ構造を参照し，交差判定を効率的に行う．静的なシーンでは，高速化データ構造は 1 度だけ構築していればよいため，時間をかけて高品質な高速化データ構造を構築するべきである．一方，動的なシーンでは，毎フレーム高速化データ構造の再構築が必要となるため，高速化データ構造の品質を多少落としても，高速に構築する必要がある．そのため，高品質かつ高速な高速化データ構造の構築方法の研究が多く行われている [2], [3], [12], [13], [14]．

近年，分割統治法を用いたレイトレーシング（Divide-And-Conquer Ray Tracing, DACRT）が提案された [4], [5]．DACRT では，高速化データ構造の構築とレイトレーシングを同時に行うことにより，使用メモリ量をレイの数とプリミティブの数から計算することができる．そのため，メモリ使用量に制限があるハードウェアでは非常に有用である．また，レンダリング速度に関しても，従来法と同程度の速度である．Mora は，円錐状のコーンパケットを使用することで，プライマリレイに対して最適化された DACRT を提案した [4]．しかし，パケットを用いているため，コヒーレンスの低いセカンダリレイ，ランダムレイなどでは適用が困難である．Afra は CPU の SIMD 拡張命令セットの SSE（Streaming SIMD Extensions），AVX（Advanced Vector Extensions）を使用し，コヒーレンスの低いレイに対して最適化した DACRT を提案した [6]．Ravichandran らは，DACRT の GPU による実装を行った [7]．この手法では，GPU による並列処理のためにレイの複製を生成する必要があり，メモリ使用量が決定論的という DACRT の利点が失われてしまっている．DACRT は，高速化データ構造の構築とレイトレーシングを同時に行うため，高速化データ構造の構築時にレイの情報を持っている．しかしながら，これらの手法では，問題の分割にプリミティブの分布のみを考慮しており，レイの分布を考慮していなかった．提案法では，レイサンプリングにより，レイの分布を考慮した問題の分割を行う．

レイの分布，交差判定結果を利用し，高速化データ構造の構築，レイの追跡を行う研究は多く行われている．Bittner らは，レイの分布を考慮した高速化データ構造の構築方法を提案した [15]．この手法では，階層的データ構造の分割位置の決定を行う際にレイの分布を考慮したヒューリスティックを用いている．しかしながら，この手法ではレイトレーシングの高速化率の大きな向上はなかった．Feltman

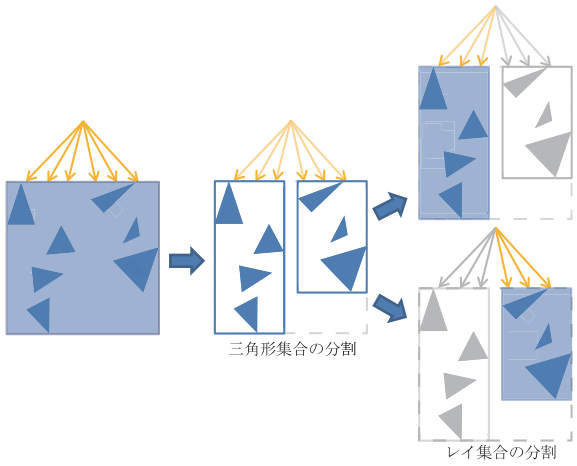


図 1 問題分割の流れ．三角形集合を二分し，分割された三角形集合を包含するバウンディングボリュームとレイ集合で交差判定を行い，交差するレイを計算することで問題を分割する

Fig. 1 A flow of a single problem partitioning step (partitioning a set of triangles, intersection tests between a set of rays and bounding volumes of the partitioned set of triangles, filter rays that intersect each bounding volume).

らは，シャドウレイの分布を考慮した BVH の構築と，探索方法を提案した [16]．しかしながら，シャドウレイに最適化された BVH のための構築時間，メモリが必要である．

3. Divide-And-Conquer Ray Tracing

DACRT は，分割統治法を用いてレイ集合とプリミティブ集合間の交差判定問題を解く方法である．DACRT では，任意の幾何プリミティブ，木構造の高速化データ構造を用いることができるが，本研究では，幾何プリミティブとして三角形，高速化データ構造として BVH を使用する．そのため，以降では，幾何プリミティブを三角形，高速化データ構造を BVH として説明する．DACRT では，与えられたレイ集合と三角形集合を再帰的に分割し，どちらかの集合の要素数が十分に小さくなれば，集合内のすべてのレイと三角形で交差判定を行う．すべての再帰処理が終了したとき，すべてのレイの最近傍の三角形との交点が計算される．ここで，集合の分割を行うには，まず，三角形集合を二分し，分割された三角形集合を包含するバウンディングボリューム（Bounding Volume，BV）とレイ集合で交差判定を行い，交差するレイを計算すればよい．問題の分割の流れを図 1 に示す．

三角形集合の分割は，レイ集合の分割の効率に大きく影響する．分割された三角形集合を包含する BV が大きいままでは，ほとんどのレイが BV と交差し，レイの数を大きく削減できないためである．三角形集合の良い分割を行うには，以下のコスト関数 C に従い分割すればよい．

$$C(V \to V_L, V_R) = C_T + C_I(p_L N_L + p_R N_R) \qquad (1)$$

Algorithm 1 提案法のアルゴリズム．R はレイ集合，T は三角形集合，cb は T の各三角形の AABB の重心を包含する BV である．δ は再帰停止条件の閾値，V は BV，N_s はサンプルレイの数である．NaiveRT は与えられたレイ集合と三角形集合のすべての組合せについて交差判定を行う関数である．

```
1: procedure DACRT(R, T, cb)
2:     if |R| < δ_R or |T| < δ_T then
3:         return NaiveRT(R, T)
4:     end if
5:     K−1 個の分割候補の計算 T_{L,j}, T_{R,j}, V_{L,j}, V_{R,j}, cb_{L,j}, cb_{R,j}
6:     カウンタを初期化 c_L, c_R, n_L, n_R ← 0
7:     for each サンプルレイ r do                ▷ レイサンプリング
8:         Intersect(r, V_L, V_R, c_L, c_R, n_L, n_R)
9:     end for
10:    C_min ← ∞, j_min ← 1
11:    for j = 1 to K − 1 do                     ▷ 分割位置の計算
12:        α_{L,j} ← c_{L,j}/N_s, α_{R,j} ← c_{R,j}/N_s
13:        式 (2) を使いコスト C を計算
14:        if C ≤ C_min then
15:            j_min ← j, C_min ← C
16:        end if
17:    end for
18:    if n_{L,j_min} ≥ n_{R,j_min} then          ▷ 探索順序の決定
19:        (α0,T0,V0,cb0) ← (α_{L,j_min},T_{L,j_min},V_{L,j_min},cb_{L,j_min})
20:        (α1,T1,V1,cb1) ← (α_{R,j_min},T_{R,j_min},V_{R,j_min},cb_{R,j_min})
21:    else
22:        (α0,T0,V0,cb0) ← (α_{R,j_min},T_{R,j_min},V_{R,j_min},cb_{R,j_min})
23:        (α1,T1,V1,cb1) ← (α_{L,j_min},T_{L,j_min},V_{L,j_min},cb_{L,j_min})
24:    end if
25:    if α0 > C_bv/(n_child C_child) then
26:        DACRT(R, T0, cb0)                     ▷ レイ集合分割を省略
27:    else
28:        R0 ← R ∩ V0 DACRT(R0, T0, cb0)        ▷ レイ集合を分割
29:    end if
30:    if α1 > C_bv/(n_child C_child) then
31:        DACRT(R, T1, cb1)                     ▷ レイ集合分割を省略
32:    else
33:        R1 ← R ∩ V1 DACRT(R1, T1, cb1)        ▷ レイ集合を分割
34:    end if
35: end procedure
```

ここで，V_L，V_R は BV である V の子ノードの BV，C_T はレイと BV の交差判定コスト，C_I はレイと三角形の交差判定コスト，p_L，p_R は V_L，V_R とレイの交差確率，N_L，N_R は V_L，V_R 内に含まれる三角形の数である．式 (1) を評価するには，V_L，V_R との交差確率 p_L，p_R が必要であるが，通常のレイトレーシングでは BVH の構築時にはレイの分布が未知なため p_L，p_R は未知である．そのため，レイの分布は一様であると仮定し，p_L，p_R を，V_L，V_R の表面積と，V の表面積の割合で近似した SAH（Surface Area Heuristic）コスト関数が一般的に使用されている [17]．

4. 提案法

アルゴリズム 1 に，提案法のアルゴリズムの擬似コードを示す．アルゴリズム 1 は，レイ集合 R，三角形集合 T，三

角形集合の各三角形を包含する軸平行境界ボックス（Axis Aligned Bounding Box，AABB）の重心を包含するBVであるcbを入力とし，RとT間の交点を計算する．アルゴリズム1では，最初にレイの数か三角形の数が閾値以下かどうか判定し，現在のノードが葉ノードかどうかを判定する．閾値δ_R，δ_Tはともに8程度にすれば良い結果が得られている．もし現在のノードが葉ノードである場合は，レイ集合と三角形集合で交差判定を行い，葉ノードでない場合は，問題の分割処理を行う．問題の分割処理では，まず三角形集合の分割を行う．三角形集合の分割にはWaldの提案したbinningベースの手法を用いる[3]．binとは，BVをある軸に垂直な面で等分割したときの各区間のことを指す．この手法により，$K-1$個の分割候補の三角形集合$T_{L,j}$，$T_{R,j}$と，$T_{L,j}$，$T_{R,j}$を包含するBVである$V_{L,j}$，$V_{R,j}$，重心を包含するBVである$cb_{L,j}$，$cb_{R,j}$が得られる．ここで，Kはbinの数，jは分割候補のインデックスを表し$1 \leq j \leq K-1$である．binの数Kは32程度にすれば良い結果が得られている．

アルゴリズム1において，7行目以降が提案法の処理である．提案法では，最初にレイサンプリングを行い，レイの分布を計算する．つまり，これまでの処理で計算した$K-1$個の分割候補のBVの組$V_{L,j}$，$V_{R,j}$に対して，レイ集合から取り出した少量のレイで交差判定を行い，レイとBV間の情報を記録する．そして，この情報を基に分割候補でコスト関数を評価し，コストが最小となる候補を選択する．次に，レイサンプリングで得た情報を基に，分割されたノードのどちらを先に探索するかを決定する（V_0，V_1の順で探索する）．最後に，ノードとレイ集合で交差判定を行うか判定を行い，レイの数について問題サイズを大きく削減できない場合は，レイ集合の分割を行わずに再帰処理を続け，それ以外は，通常どおりレイ集合の分割を行い再帰処理を続ける．

4.1 レイサンプリング

レイサンプリングは，レイの分布を計算するために行う．レイ集合から取り出した少量のレイで，各分割候補のBVの組に対して交差判定を行うことで，レイとBV間の情報を計算する．ここで，情報を記録するための変数として要素数が$K-1$の配列c_L，c_R，n_L，n_Rを用意する．$c_{L,j}$，$c_{R,j}$は$V_{L,j}$，$V_{R,j}$と交差したサンプルレイの数，$n_{L,j}$，$n_{R,j}$は$V_{L,j}$と$V_{R,j}$でサンプルレイがどちらにより近いかを記録するためのものである．レイサンプリングにはコストがかかるため，レイの数が1,000本以上のときのみ，100本のレイでレイサンプリングを行い，1,000本未満のときは，Afraの提案したDACRTを行っている．また，サンプリング方法については，レイ集合を格納している配列を一定間隔でサンプリングする方法を用いている．なお，サンプリング数とサンプリング方法を変えて比較実験を行ったが，

Algorithm 2 サンプルレイrとBVである$V_{L,j}, V_{R,j}$で交差判定を行う．INTERSECTP(V, r, t_n, t_f)は一般的なレイとBVの交差判定を行う関数であり[18]，rとVが交差するならば$[t_n, t_f]$に交差している間の区間を代入して真を返す．

```
1: procedure INTERSECT(r, V_L, V_R, c_L, c_R, n_L, n_R)
2:     for j = 1 to K − 1 do
3:         d_{L,j}, d_{R,j} ← ∞
4:         if INTERSECTP(V_{L,j}, r, t_n, t_f) then
5:             c_{L,j} ← c_{L,j} + 1, d_{L,j} ← t_n
6:         end if
7:         if INTERSECTP(V_{R,j}, r, t_n, t_f) then
8:             c_{R,j} ← c_{R,j} + 1, d_{R,j} ← t_n
9:         end if
10:        if d_{L,j} < d_{R,j} then
11:            n_{L,j} ← n_{L,j} + 1
12:        else
13:            n_{R,j} ← n_{R,j} + 1
14:        end if
15:    end for
16: end procedure
```

上記のパターンを用いれば良好な結果が得られている．レイサンプリングの処理の擬似コードをアルゴリズム2に示す．

アルゴリズム2では，サンプリングされたレイrと$V_{L,j}$，$V_{R,j}$で交差判定を行い，交差している場合は$c_{L,j}$，$c_{R,j}$を1増分する．レイとBVの交差判定には一般的に使用されている方法[18]を使用する．この方法では交差判定の過程で，レイの始点からレイとBVの交点までの距離t_nを計算するため，この距離を使用して$V_{L,j}$と$V_{R,j}$でどちらがよりサンプルレイに近いかを判定する．そして，$V_{L,j}$が近い場合は$n_{L,j}$，$V_{R,j}$が近い場合は$n_{R,j}$を1増分する．

4.2 コスト関数

Afraはコスト関数としてSAHコスト関数を使用している[6]．SAHコスト関数は，BVとレイの交差確率を，そのノードのBVの表面積と，親ノードのBVの表面積の割合で近似している．レイがシーンにわたって一様に分布している場合は良い近似であるが，分布が偏っている場合では，良い近似とはいえない．これは高速化データ構造とレイトレーシングを独立に行っている場合は避けられないことだが，DACRTではこれらを同時に行うためこの問題は解決することができる．レイサンプリングでBVと交差するサンプルレイの数$c_{L,j}$，$c_{R,j}$を計算しており，この値からBVとの交差確率$\alpha_{L,j}$，$\alpha_{R,j}$を計算できる．交差確率$\alpha_{L,j}$，$\alpha_{R,j}$を使用したコスト関数は以下の式となる．

$$C(V \to V_{L,j}, V_{R,j}) = C_T + C_I(\alpha_{L,j} N_{L,j} + \alpha_{R,j} N_{R,j}) \tag{2}$$

ここで，$N_{L,j}$，$N_{R,j}$は$V_{L,j}$，$V_{R,j}$に含まれる三角形の数である．これにより，実際のレイの分布を考慮したBVH

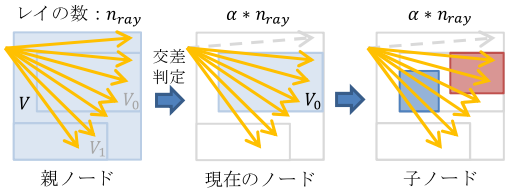


図 2 レイ集合の分割が非効率な場合．レイ集合の大部分がバウンディングボリューム V_0 と交差するため，レイ集合の分割による，問題サイズの大きな削減ができない．レイ集合と現在のノードのバウンディングボリュームの交差確率 α より，レイ集合の数は $\alpha * n_{ray}$ に削減されると推測され，さらなる問題の分割が行われる

Fig. 2 An inefficient case of partitioning set of rays. In this case, since most of rays intersect the bounding volume V_0, the number of rays is not sufficiently reduced by subdividing the set of rays. The number of rays intersecting V_0 is estimated by $\alpha * n_{ray}$, where α is the intersection ratio.

の構築を行うことができる．

4.3 探索順序

効率的なレイトラバーサルを行ううえで，ノードの探索順序は重要な要素の 1 つである．レイの始点からレイと三角形の交点までの距離が，レイの始点からレイと BV の交点までの距離より短い場合，そのノードにはレイの始点により近いレイと三角形の交点は存在しないため，そのノードの探索が不要となるからである．そのため，より近いノードから探索を行うことは，計算時間の短縮につながる．Afra は探索順序を 1 本のレイを使用して決定していた [6]．レイのコヒーレンスが非常に小さい場合はこれでも問題ないが，セカンダリレイのように，コヒーレンスが多少存在しているようなレイの場合は不十分であり，レイ全体では最適でないノードを先に探索すれば大きな遅延が生じてしまう．そこで，レイサンプリング時に計算しておいた $n_{L,j}$，$n_{R,j}$ で，より値の大きい方のノードを先に探索する．これにより，より精度の良い探索順序の決定を行うことができる．

4.4 レイ集合分割の省略

レイ集合を分割するには BV との交差判定を行い，BV と交差するレイを計算する必要がある．しかし，**図 2** に示すように，レイの分布によってはほとんどのレイが BV と交差している場合がある．この場合レイ集合の大きな分割は行えず，レイと BV の交差判定のための大きな計算時間が費やされるだけである．よってこのような場合は，交差判定を行わずにすべてのレイが交差していると見なし，次の処理へ移る方が効率的となる．交差判定を省略した場合の処理の流れを**図 3** に示す．提案法では，レイサンプリングで得た交差確率 α を使用しこの問題を解決する．以降で

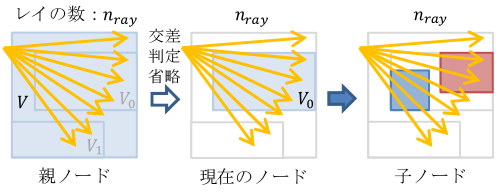


図 3 レイ集合の分割を省略した場合．レイ集合の分割を省略するため，レイ集合の数は n_{ray} のまま，さらなる問題の分割が行われる

Fig. 3 A case of skipping partitioning set of rays. In this case, the number of rays is not reduced, and n_{ray} rays are used to perform intersection tests between rays and bounding volumes of child nodes.

は，レイと BV の交差判定を行う場合のコストと，レイと BV の交差判定を行わない場合のコストについて説明し，最適な選択を行う条件式を導出する．

レイと BV の交差判定を行う場合のコストについて説明する．BV と交差判定を行うレイの数を n_{ray}，レイと BV の交差判定コストを C_{bv} とすると，n_{ray} のレイと BV で交差判定を行うコストは $n_{ray} * C_{bv}$ となる．ここで，BV と交差するレイの数は，レイサンプリングで計算した交差確率 α から $\alpha * n_{ray}$ になると予想される．次に，交差したレイは今後，子ノードとも交差判定を行う必要があるため，C_{child} をレイと子ノードとの交差判定コスト，n_{child} を子ノードの数とすると，$\alpha * n_{ray}$ のレイと n_{child} の子ノードで交差判定を行う場合のコストは $(\alpha * n_{ray} * C_{child}) * n_{child}$ となる．よって，レイと BV の交差判定を行う場合のコスト C_{int} は以下の式で計算される．

$$C_{int} = n_{ray} * C_{bv} + (\alpha * n_{ray} * C_{child}) * n_{child} \qquad (3)$$

次に，交差判定を省略する場合のコストについて説明する．BV と交差判定を行わないので，以後のステップでのレイの数は n_{ray} のままであるが，交差判定コストは 0 である．続いて，n_{ray} のレイと n_{child} の子ノードで交差判定を行う場合のコストは $(n_{ray} * C_{child}) * n_{child}$ である．よって，交差判定を省略する場合のコスト C_{skip} は以下の式で計算される．

$$C_{skip} = 0 + (n_{ray} * C_{child}) * n_{child} \qquad (4)$$

式 (3)，(4) より，レイと BV の交差判定を行う場合のコスト C_{int} が，レイと BV の交差判定を行わない場合のコスト C_{skip} より大きくなる交差割合 α は以下の条件式で表される．

$$\alpha > 1 - \frac{C_{bv}}{C_{child} * n_{child}} \qquad (5)$$

交差割合 α が式 (5) の条件を満たす場合は，レイと BV の交差判定を行わずに，すべてのレイが交差していると見なす．条件を満たさない場合は，レイと BV の交差判定を行う．この条件式に従いレイ集合の分割を行うことで，非

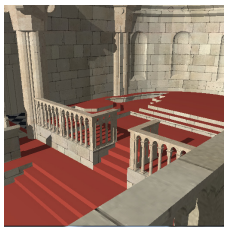
Sibenik			
画像サイズ	(a) 点光源	(b) 点光源	(c) 面光源
512^2	415 ms/295 ms（1.41）	281 ms/183 ms（1.54）	374 ms/255 ms（1.47）
$1{,}024^2$	1,520 ms/1,003 ms（1.52）	1,130 ms/700 ms（1.61）	1,430 ms/860 ms（1.66）
$2{,}048^2$	5,992 ms/3,759 ms（1.59）	4,656 ms/2,647 ms（1.76）	5,730 ms/3,164 ms（1.81）
$4{,}096^2$	27,300 ms/14,700 ms（1.86）	19,772 ms/11,330 ms（1.75）	22,269 ms/11,491 ms（1.94）

図 4 Sibenik シーン（三角形数 75K）での解像度別の Afra の手法と提案法の計算時間の比較．括弧内の数字は提案法による高速化率を表す

Fig. 4 Comparisons of Afra's method with our method for different image sizes of the Sibenik scene (75K triangles). The number in the parentheses is the acceleration ratio using our method.

効率なレイ集合の分割を避けることができる．ここで，BV が葉ノードである場合は，子ノードの数 n_{child} はノードに含まれる三角形となり．C_{child} は三角形とレイの交差判定コストとなる．BV が葉ノードでない場合は，子ノードとレイとの交差判定コスト C_{child} は C_{bv} となり，BVH が 2 分木の場合 n_{child} は 2 となるので，式 (5) は以下の式に簡略化される．

$$\alpha > 0.5 \tag{6}$$

5. 結果

提案法の実行結果を**図 4**，**図 5**，**図 6**，**図 7** に示す．実行環境は，CPU が Intel Core i7 2.67 GHz，メモリ 6.0 GB RAM の PC で，1 スレッドのみで実行している．なお，計測時間はすべてのレイが最近傍の三角形を計算するまでの時間で，レイの生成，シェーディングの時間は含めていない．比較対象は Afra の手法 [6] である．提案法のアルゴリズムは，SIMD 命令を使用して実装しており，レイ，三角形のデータ構造は Afra の手法と同じ SIMD 命令に最適化されたものである．また，Afra の手法と同様に，レイ集合内のレイの数と三角形集合内の三角形の数の割合が閾値（現在の実装では 1.5）を超える場合は，レイサンプリングによる BV の分割を行い（レイの数が 1,000 本未満の場合は SAH による分割），それ以外では，BV を幅が最大の軸の中点で分割する方法をとっている．この場合ではレイサンプリングにより，探索順序，レイ集合分割の省略の決定のみ行っている．

図 4 に Sibenik シーンのレンダリング結果を示す．モデルの床には鏡面反射材質を設定している．また，図 4 (a) (b) は点光源，図 4 (c) は面光源を設置している．図 4 より，提案法では平均 1.85 倍，最大 2 倍程度の高速化を達成している．セカンダリレイを扱うには，プライマリレイとシーンの交点から反射レイを追跡すればよいが，DACRT では BVH を保持しないため，プライマリレイ集合の交点計算時とセカンダリレイ集合の交点計算時にそれぞれ BVH を再構築する必要がある．そこで，Sibenik シーンにおいて，プライマリレイ集合の交点計算時に構築した BVH を保持し，セカンダリレイ集合の交点計算時にその BVH を再利用した場合の実験を行ったが，BVH を保持せずにセカンダリレイ集合の交点計算時に最初から BVH を再構築する場合と比較し，レンダリング速度は変わらなかった．これは，Sibenik モデルではレイの追跡処理の計算時間が支配的であるためと思われる．

図 4 (a) 解像度 $4{,}096^2$ において，本論文で提案したコスト関数，探索順序の決定方法，レイ集合分割の省略それぞれ個別の高速化率は，1.24 ×（27.3 s/22.0 s，42%），1.05 ×（27.3 s/26.0 s，10%），1.28 ×（27.3 s/21.3 s，48%）となった．括弧内は各手法を無効，有効時の計算時間，計算時間削減への寄与率である．この結果より，実際のレイの分布を使用したコスト関数，レイ集合分割の省略が計算時間削減の大きな割合を占めていることが分かる．

図 5 (a) に提案法が非効率となった場合の例を示す．この場合では，解像度 $1{,}024^2$ において，提案法が Afra の手法より劣る結果となった．しかしながら，両者の差はわずかであり，解像度が高くなれば提案法の方がより高速に計算を行えている．図 5 (b) に環境遮蔽，図 5 (c) に Sanmiguel シーンでの被写界深度（DOF）を適用した 3 回反射までのパストレーシングのレンダリング結果を示す．なお，環境遮蔽は，レイと物体表面の交点から半球上のランダムな方向に短いレイを飛ばし遮蔽率を調べる手法である．パストレーシングは，レイと物体表面の交点から半球上のランダムな方向にレイを飛ばし間接照明の寄与を計算する手法である．図 6 に Conference シーン，図 7 に Sponza シーンのレンダリング結果を示す．点光源の場合と比較してパ

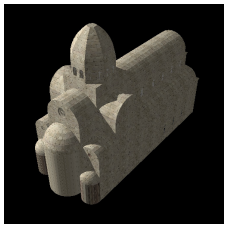
			
画像サイズ	(a) 点光源	(b) 環境遮蔽	(c) パストレーシングと DOF
512^2	110 ms/110 ms（1.00）	446 ms/365 ms（1.22）	17,319 ms/16,876 ms（1.03）
$1{,}024^2$	331 ms/334 ms（0.99）	1,716 ms/1,217 ms（1.41）	27,725 ms/26,435 ms（1.05）
$2{,}048^2$	1,236 ms/1,130 ms（1.09）	6,948 ms/4,359 ms（1.59）	67,253 ms/59,893 ms（1.12）
$4{,}096^2$	5,075 ms/4,219 ms（1.20）	29,439 ms/19,300 ms（1.53）	215,805 ms/173,018 ms（1.25）

図 5 Sibenik シーンと Sanmiguel シーン（三角形数 7.9M）での解像度別の Afra の手法と提案法の計算時間の比較．括弧内の数字は提案法による高速化率を表す

Fig. 5 Comparisons of Afra's method with our method for different image sizes of the Sibenik scene and the Sanmiguel scene (7.9M triangles). The number in the parentheses is the acceleration ratio using our method.

Conference			
画像サイズ	(a) 点光源	(b) 点光源	(c) パストレーシング
512^2	234 ms/189 ms（1.24）	273 ms/189 ms（1.44）	1,208 ms/1,071 ms（1.13）
$1{,}024^2$	803 ms/602 ms（1.34）	1,030 ms/671 ms（1.54）	3,936 ms/3,279 ms（1.20）
$2{,}048^2$	3,113 ms/2,182 ms（1.43）	5,992 ms/2,636 ms（1.60）	15,533 ms/12,316 ms（1.26）
$4{,}096^2$	12,784 ms/8,930 ms（1.43）	17,987 ms/10,618 ms（1.69）	65,117 ms/47,761 ms（1.36）

図 6 Conference シーン（三角形数 331K）での解像度別の Afra の手法と提案法の計算時間の比較．括弧内の数字は提案法による高速化率を表す

Fig. 6 Comparisons of Afra's method with our method for different image sizes of the Conference scene (331K triangles). The number in the parentheses is the acceleration ratio using our method.

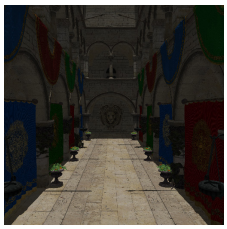
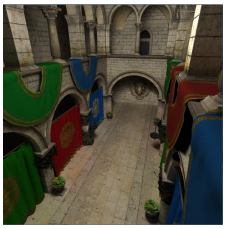
Sponza			
画像サイズ	(a) 点光源	(b) 点光源	(c) パストレーシング
512^2	661 ms/478 ms（1.38）	599 ms/448 ms（1.52）	2,853 ms/2,331 ms（1.22）
$1{,}024^2$	2,298 ms/1,428 ms（1.61）	2,213 ms/1,471 ms（1.50）	8,288 ms/7,066 ms（1.17）
$2{,}048^2$	8,801 ms/5,056 ms（1.74）	8,774 ms/5,363 ms（1.64）	31,960 ms/23,779 ms（1.34）
$4{,}096^2$	35,397 ms/19,657 ms（1.80）	35,643 ms/20,548 ms（1.73）	136,327 ms/98,228 ms（1.39）

図 7 Sponza シーン（三角形数 262K）での解像度別の Afra の手法と提案法の計算時間の比較．括弧内の数字は提案法による高速化率を表す

Fig. 7 Comparisons of Afra's method with our method for different image sizes of the Sponza scene (262K triangles). The number in the parentheses is the acceleration ratio using our method.

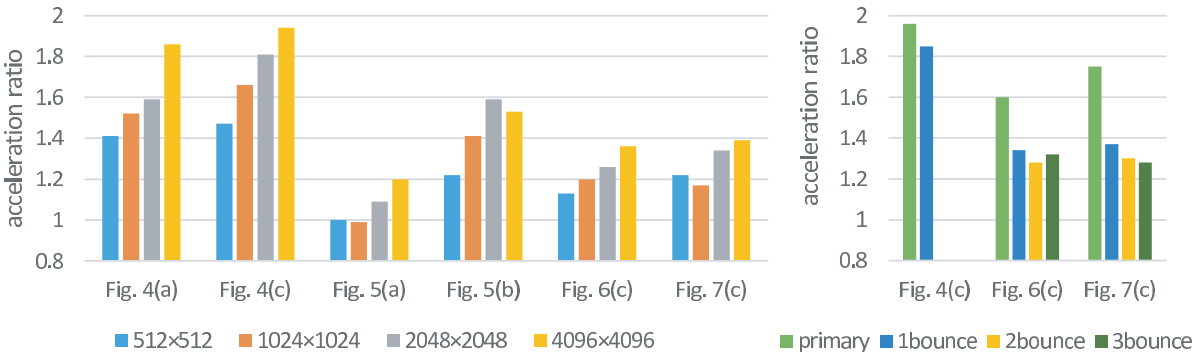


図 8 (a) 解像度別の高速化率と (b) 反射回数別の高速化率

Fig. 8 Acceleration ratios for (a) different image sizes and (b) different bounce numbers of rays.

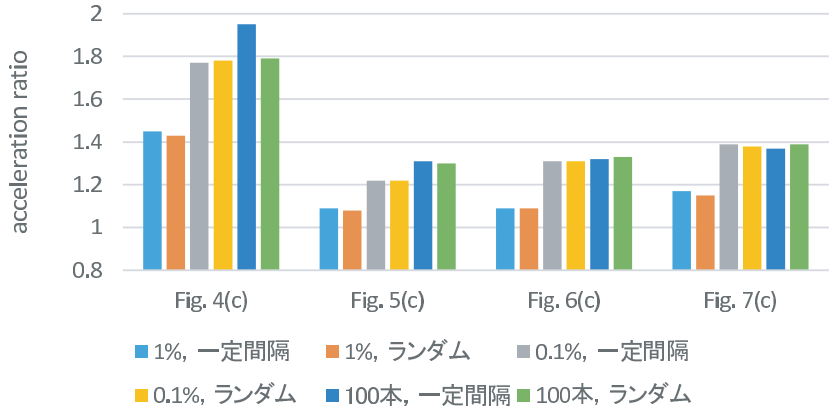


図 9 レイのサンプリング方法別の高速化率

Fig. 9 Acceleration ratios for different ray sampling methods.

ストレーシングでは，高速化率が低くなっているが，パストレーシングではレイの分布が均一に近くなるため，SAH コスト関数が式 (1) を精度良く近似できているためと思われる．

図 8 に解像度別の高速化率，反射回数別の高速化率を示す．図 8 (a) より，提案法ではレイ集合の大きさが大きいほど高速化率が高く，高解像度画像のレンダリングや，1 ピクセルに複数のレイを飛ばすスーパサンプリングによるアンチエイリアス処理に適していることが分かる．また，図 8 (b) より，提案法ではプライマリレイだけでなく複数回反射したレイについても高速化を達成していることが分かる．

図 9 に解像度 $4{,}096^2$ の画像生成時の，レイのサンプリング方法別の高速化率を示す．レイ集合を格納している配列を一定間隔で 1%サンプリング，ランダムに 1%サンプリング，一定間隔で 0.1%サンプリング，ランダムに 0.1%サンプリング，レイの数によらず 100 本一定間隔でサンプリング，ランダムに 100 本サンプリングして Afra の手法に対する高速化率を計測した．実験結果より，いずれの場合も Afra の手法より高速に交差判定問題を解くことができていることが分かる．また，一定間隔・ランダムどちらのサンプリング方法を用いた場合でも，ほぼ同じ高速化率が得られていることが分かる．レイのサンプル数に関しては，レイ集合の 1%を使用した場合は，レイサンプリングのための計算時間が大きくなるため，高速化率が小さくなっているが，レイ集合内のレイの数によらず 100 本使用した場合と，レイ集合の 0.1%を使用した場合では，ほぼ同じ高速化率となった．レイのサンプル数に 100 本使用した場合と，レイ集合の 0.1%を使用した場合では，一定間隔・ランダムのどちらを用いてもほぼ同じ結果となったが，レイの数によらず 100 本一定間隔にサンプリングした場合がわずかに良い結果が得られており，提案法では，レイの数によらず 100 本一定間隔でサンプリングする方法を用いている．

図 10 に解像度 $4{,}096^2$ の画像生成時の，Afra の手法に対する反射回数別のレイとバウンディングボリュームの交差判定回数の削減率を示す．図 10 より，提案法ではすべての場合で Afra の手法より交差判定回数を削減することができていることが分かる．提案法では，レイの分布の偏りを主に利用しており，レイの分布の偏りが特に大きいプライマリレイ，シャドウレイでそれぞれ交差判定回数を 30%から 50%程度，30%から 40%程度削減できている．鏡面反射したレイ（図 4 (c) 1bounce）についても，レイの分布の偏りが生じるため，交差判定回数を 50%程度に削減できている．また，レイの分布の偏りがほとんど存在しない

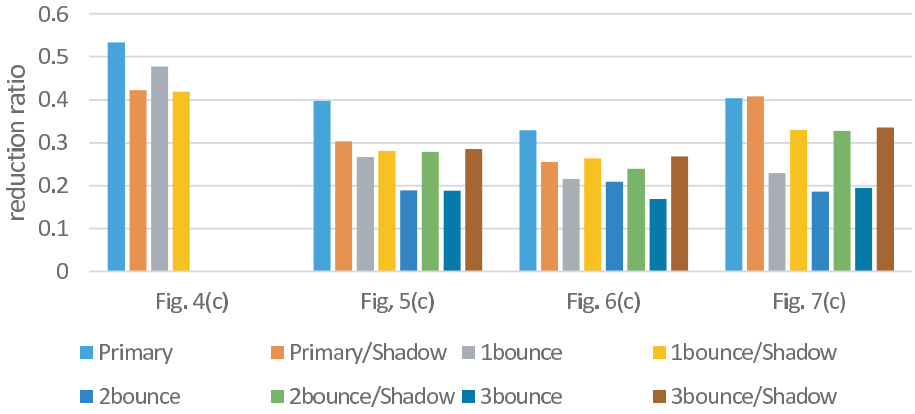


図 10 Afra の手法に対する反射回数別のレイとバウンディングボリュームの交差判定回数の削減率

Fig. 10 Reduction ratios of the number of ray and bounding volume intersection tests for different bounce numbers of rays, compared to Afra's method.

ランダムレイの場合でも，レイについての問題分割の省略などにより，交差判定回数を 20%程度削減することができている．

6. まとめ

本研究では，DACRT にレイのサンプリング処理を追加し，追跡しているレイの分布に適した高速化データ構造の構築，探索順序の決定方法を提案した．また，レイ集合を効率的に分割できないときの交差判定を避ける方法を提案した．本手法では，レイの数が少ない場合など稀に追跡効率がわずかに落ちることがあったが，ほぼすべてのシーンで高速化を達成し，最大約 2 倍の高速化を実現した．

今後の課題としては，マルチスレッド化，GPU 実装などがあげられる．

参考文献

[1] Kalojanov, J., Billeter, M. and Slusallek, P.: Two-Level Grids for Ray Tracing on GPUs, *Computer Graphics Forum*, Vol.30, No.2, pp.307–314 (2011).
[2] Zhou, K., Hou, Q., Wang, R. and Guo, B.: Real-time KD-tree construction on graphics hardware, *ACM Trans. Graph.*, Vol.27, No.5, pp.126:1–126:11 (2008).
[3] Wald, I.: On fast Construction of SAH-based Bounding Volume Hierarchies, *Proc. 2007 IEEE Symposium on Interactive Ray Tracing*, pp.33–40 (2007).
[4] Mora, B.: Naive ray-tracing: A divide-and-conquer approach, *ACM Trans. Graph.*, Vol.30, No.5, pp.117:1–117:12 (2011).
[5] Keller, A. and Wachter, C.: Efficient Ray Tracing without Auxiliary Acceleration Data Structure, *Proc. High Performance Graphics 2011 (Poster)* (2011).
[6] Afra, A.T.: Incoherent Ray Tracing without Acceleration Structures, *Proc. Eurographics Short Paper*, pp.97–100 (2012).
[7] Ravichandran, S. and Narayanan, P.J.: Parallel Divide and Conquer Ray Tracing, *Proc. SIGGRAPH Asia 2013 Technical Briefs*, No.30 (2013).
[8] Nabata, K., Iwasaki, K., Dobashi, Y. and Nishita, T.: Efficient Divide-and-conquer Ray Tracing Using Ray Sampling, *Proc. 5th High-Performance Graphics Conference*, pp.129–135 (2013).
[9] Fujimoto, A., Tanaka, T. and Iwata, K.: ARTS: Accelerated Ray Tracing System, *IEEE Computer Graphics and Applications*, Vol.6, No.4, pp.16–26 (1986).
[10] Bentley, J.L.: Multidimensional binary search trees used for associative searching, *Comm. ACM*, Vol.18, No.9, pp.509–517 (1975).
[11] Rubin, S.M. and Whitted, T.: A 3-dimensional representation for fast rendering of complex scenes, *SIGGRAPH Comput. Graph.*, Vol.14, No.3, pp.110–116 (1980).
[12] Kalojanov, J. and Slusallek, P.: A parallel algorithm for construction of uniform grids, *Proc. High Performance Graphics 2009*, pp.23–28 (2009).
[13] Wu, Z., Zhao, F. and Liu, X.: SAH KD-tree construction on GPU, *Proc. High Performance Graphics*, pp.71–78 (2011).
[14] Garanzha, K., Pantaleoni, J. and McAllister, D.: Simpler and faster HLBVH with work queues, *Proc. High Performance Graphics 2011*, pp.59–64 (2011).
[15] Bittner, J. and Havran, V.: RDH: Ray distribution heuristics for construction of spatial data structures, *Proc. 25th Spring Conference on Computer Graphics*, pp.51–58 (2009).
[16] Feltman, N., Lee, M. and Fatahalian, K.: SRDH: Specializing BVH Construction and Traversal Order Using Representative Shadow Ray Sets, *Proc. High Performance Graphics 2012*, pp.49–55 (2012).
[17] Wald, I., Boulos, S. and Shirley, P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies, *ACM Trans. Graph.*, Vol.26, No.1, pp.6:1–6:18 (2007).
[18] Pharr, M. and Humphreys, G.: *Physically Based Rendering: From Theory to Implementation*, Morgan Kaufmann Publishers (2010).

名畑 豪祐 （学生会員）

2013 年和歌山大学システム工学部情報通信システム学科卒業．現在，和歌山大学大学院システム工学研究科に在籍．

岩崎 慶 （正会員）

1999 年東京大学理学部情報科学科卒業．2001 年同大学大学院新領域創成科学研究科複雑理工学専攻博士前期課程修了．2004 年同大学院新領域創成科学研究科複雑理工学専攻博士後期課程修了．同年和歌山大学システム工学部情報通信システム学科助手．2007 年同講師．2009 年同准教授．主としてコンピュータグラフィクスに関する研究に従事．科学博士．

土橋 宜典 （正会員）

1992 年広島大学工学部第二類（電気系）卒業．1994 年同大学大学院工学部研究科システム工学専攻博士課程前期修了．1997 年同専攻博士課程後期修了．同年広島市立大学情報科学部助手．2000 年北海道大学工学研究科助教授．主としてコンピュータグラフィクスに関して，照明シミュレーション，景観予測等の研究に従事．工学博士．

西田 友是 （正会員）

1971 年広島大学工学部卒業．1973 年同大学大学院工学研究科修了．同年マツダ（株）に入社．1979 年福山大学工学部講師．1984 年同助教授．1990 年同教授．1998 年東京大学理学部教授．1999 年同大学大学院新領域創成科学研究科教授．2014 年から広島修道大学教授および UEI リサーチ所長（兼）になり，現在に至る．コンピュータグラフィックスの研究に従事．工学博士．