計算効率の定式化による
写実的な画像のロバストかつ効率的な生成

平成 31 年 3 月

和歌山大学大学院システム工学研究科

名畑　豪祐

A Robust and Efficient Rendering Method by Formulating Computational Efficiency

March 2019

Graduate School of Systems Engineering

Wakayama University

Kosuke Nabata

概要

近年のコンピュータグラフィックス（CG）技術の進歩に伴い，以前と比べ写実的な画像をはるかに高速に生成できるようになってきている．しかしながら，写実的な画像を生成するには依然として長い時間がかかるため，写実的な画像を効率的に生成する手法の研究が現在も多く行われている．基本的に画像生成アルゴリズムは，多くのパラメータを要し，手法の効率や結果画像の画質はパラメータに大きく依存している．また，画像生成にかかる計算時間と画質はトレードオフの関係にあり，所望の画質を満たしつつ高速に画像生成が可能なパラメータを設定するのは困難である．しかしながら多くの研究では，最適なパラメータが与えられた下での計算効率や画質を重視しており，最適なパラメータを見つけるための労力は無視されている．そのため実際の応用の場面では，高い計算効率や所望の画質の結果を得るために，ユーザはパラメータ調整とレンダリングを繰り返す必要があり，多大な労力と時間を要する．そこで本論文では，計算時間と画質の関係を定式化することで，パラメータ調整の労力の少ないロバストかつ効率的な 3 つの画像生成手法を提案する．1 つ目の方法では，様々な大域照明アルゴリズムの基盤的技術であるレイトレーシングにおいて，高品質な高速化データ構造を構築することで，効率的に写実的な画像生成を行う．2 つ目の方法では，写実的な画像を効率的に生成する手法の一つである多光源レンダリング法において，計算時間のボトルネックとなっている可視関数の評価回数を確率的アプローチにより削減することで効率的に写実的な画像の生成を行う．最後に 3 つ目の提案法では，多光源レンダリング法において，輝度推定値の誤差を制御することで，一度のレンダリングで所望の画質の結果を生成する．本研究により，ユーザは少ないパラメータ調整の労力で高効率に写実的な画像を生成でき，CG による写実的な画像を利用する様々な分野において，作業効率の大幅な向上をもたらすことができる．

Abstract

Recent advances in computer graphics make it possible to render photorealistic images much more efficiently than ever before. However, since the computational cost of photorealistic rendering is still high, efficient rendering methods are widely studied. Most global illumination algorithms require a lot of parameters, which significantly affect the rendering efficiency and the image quality. Due to the trade-off relation between the rendering efficiency and the image quality, it is difficult to set parameters that satisfy both efficient renderings and desired image qualities. Previous rendering methods, however, mainly focus on the rendering efficiency under the condition that the optimal parameters have been already obtained, and an effort to find the optimal parameters are usually ignored. Therefore, in practical use of the previous rendering methods, users need to repeat parameter adjustments and rendering until desired images are rendered. To solve this problem, we propose three efficient and robust rendering methods that can reduce the trial and error processes of setting appropriate parameters by formulating the relation between the computational efficiency and the image quality. Our first method accelerates a ray tracing algorithm that is the basis of most global illumination algorithms. Our second method accelerates many-light rendering, which is one of the most efficient global illumination algorithms, by reducing the number of visibility evaluations which is a bottleneck of many-light rendering. Finally, our third method accelerates the many-light rendering by stochastic control of errors due to radiance estimation. Since our proposed three methods have fewer parameters to tweak than previous methods, users can create photorealistic images with little effort of parameter adjustment. Our research brings a significant improvement in work efficiency to many applications of realistic image synthesis.

目次

第１章　序論

本章では，コンピュータグラフィックス（CG）による写実的な画像生成に関する研究の概説を行う．また，本論文において提案する 3 つの手法の概要と論文構成について説明する．

1.1　研究背景

近年，コンピュータの普及や性能の向上により，様々な分野で CG の技術が応用されてきている．従来は，映画やコンピュータゲームなどの映像製作が主な応用範囲であったが，現在では，建築や製造業における設計支援や，災害，自然現象のシミュレーション，医療における可視化システムなどにも応用されており，業務の効率化に大きく貢献している．建築業界を例に挙げて CG の活用例を説明すると，建築物を造る際，依頼主と建築業者は建築物の完成イメージを共有しながら設計を行っていく．ここで従来は，様々な図面を用いて完成イメージの共有を試みていたが，図面からでは詳細な完成イメージを想像するのは難しく，効率的に設計を行うことができていなかった．そのため現在では，図面ではなく完成物の CG 画像を用いて完成イメージの共有を行っている．CG 画像を用いれば瞬時に完成イメージを共有することができ効率的に設計を行うことができる．このように CG の技術を用いることで作業効率の向上が可能であるが，一つ問題がある．それは CG による写実的な画像を生成するには長い計算時間が必要という点である．複雑なシーンでは 1 枚の画像を生成するだけでも数十時間かかる場合があり，映像製作など大量の画像が必要な場合は膨大な時間を要する．そのため，CG を用いた応用分野において，さらなる作業効率の効率化のためにも，写実的な画像を効率的に生成する手法の研究が重要である．

CG による写実的な画像を生成するには，光源から発せられる光のあらゆる輸送経路を考慮した上で，視点に到達する光のエネルギーを計算する光輸送問題を解く必要がある．この

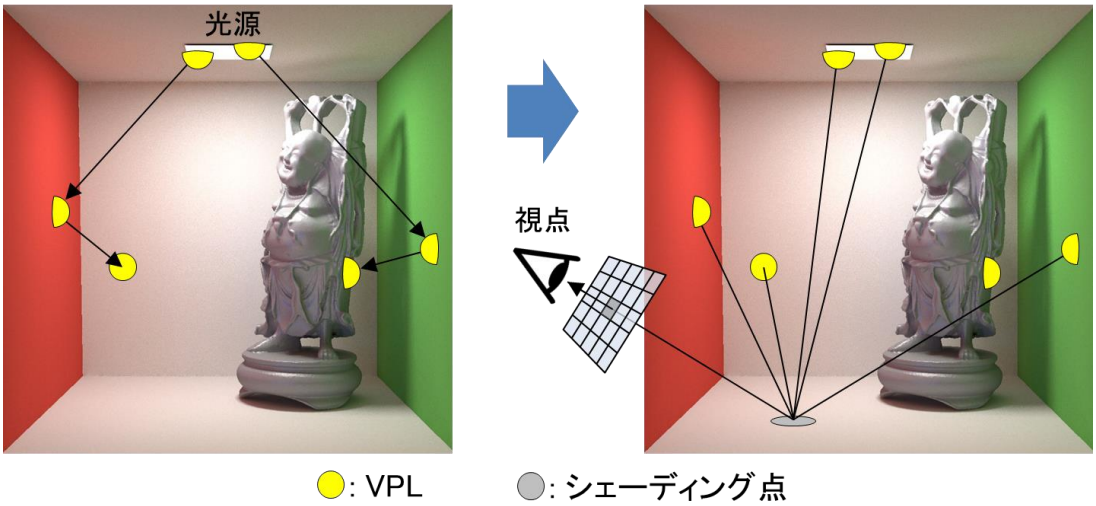


図 1.1　多光源レンダリング法．あらかじめ大量の VPL を生成しておき，各ピクセルではシェーディング点に入射する VPL から光が視点方向に反射する量を基にピクセルの値を計算する．

計算はコストが高く，近年のコンピュータ性能の向上をもってしても画像生成に多くの時間を要する．CG による写実的な画像生成に関する基本理論[26]が完成した 1980 年代から現在に至るまで，様々な光輸送問題の効率的な解法が提案されてきたが[18, 21, 25, 50, 52]，未だユーザが要求するレベルの写実性と効率性を両立した手法は存在せず，現在も広く研究されている．このような背景の下，写実的な画像を効率的に生成する手法として，近年，多光源レンダリング法がよく研究されている[9]．この手法は二段階のアルゴリズムで，まず光源から光の輸送経路を大量に生成する．この手法では光の輸送経路において光の放射点やシーンとの交点のことを仮想的な点光源（Virtual Point Light，VPL）と呼ぶ．VPL の生成後は，各ピクセルでピクセルに対応するシーン上の点（以降シェーディング点と呼ぶ）を生成し，VPL からの直接光を計算することで，視点方向に出射する輝度を近似的に計算する（図 1.1 参照）．この手法の特徴として，使用する VPL の数によって計算時間と近似精度を制御できる点がある．使用する VPL の数が少ない場合は，光輸送問題の粗い近似ではあるが，光源からの直接光と間接光を考慮した画像を数秒以下で生成することができる．一

(a) VPL 数：100　　(b) VPL 数：100 万　　(c) 参照画像[26]

図 1.2　VPL 数と近似精度の関係．多光源レンダリング法では VPL 数を増やすほど光輸送問題の正確な近似となる．ただし，すべての VPL を使用して輝度計算を行う場合，計算時間は VPL 数に比例して増加し，(a)は計算時間 4.3 秒，(b)は計算時間 33,184 秒（約 9 時間）である．なお，画像解像度は$1{,}024^2$であり，Intel Core i7-6950X CPU を搭載した PC での計測時間である．(b)の画像生成時の詳細な計算時間の内訳は図 1.3 に示す通りである．

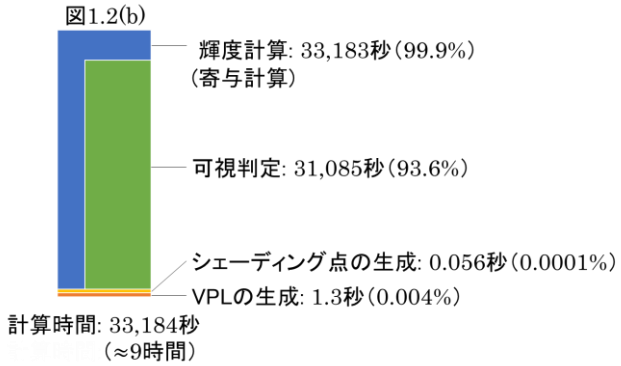


図 1.3　図 1.2(b)の画像生成における計算時間の内訳．括弧内の数字は全体の計算時間内で占める割合である．多光源レンダリング法では，VPL は全シェーディングで共有可能であり生成は一度だけでよいため，大量に VPL を生成しても計算コストは低い．大量に VPL を使用する場合，輝度計算が全体の計算時間の大部分を占め，VPL とシェーディング点間の可視判定は輝度計算の計算時間の大部分を占める．

方，使用する VPL の数を増やせば，光輸送問題の高精度な近似となり，写実的な画像を生成することができる（図 1.2 参照）．このような特徴から，プレビュー用として少ない VPL で高速に画像生成を行いシーンを調整してから，大量の VPL で写実的な画像を生成すると

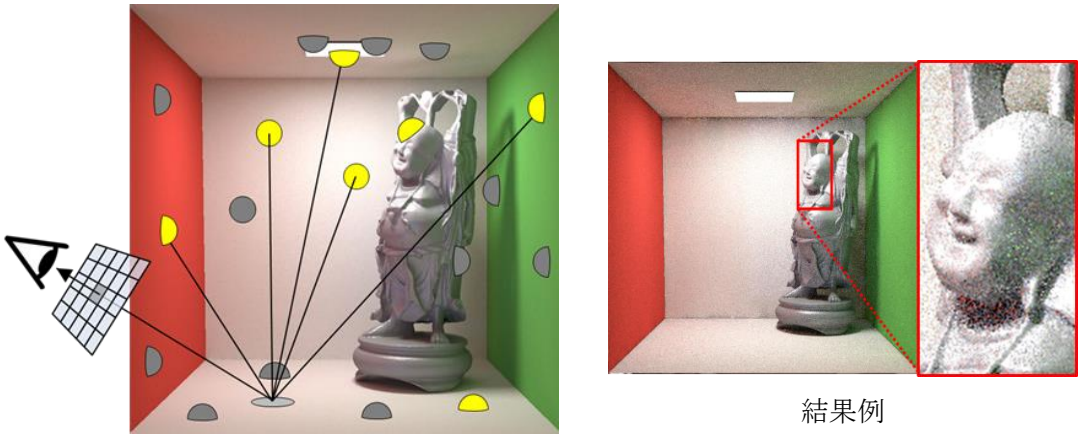


図 1.4　VPL サンプリングによる輝度推定．大量に生成した VPL から少量の VPL をサンプリングし，全 VPL を使用した場合の輝度を推定する．推定値には誤差が含まれるが，サンプル数を増やすか，結果に与える影響の大きい VPL を重点的にサンプリングすることで誤差を低減できる．推定精度が低い場合，各ピクセルで誤差の大きさがバラつくことにより結果画像にノイズが発生する（右図）．

いう利用も可能であり，様々なアプリケーションで多光源レンダリング法が採用され始めている．

多光源レンダリング法では，大量の VPL を用いることで光輸送問題の高精度な近似解を得ることができる．しかしながら，計算時間は VPL 数に比例して増加するため，大量の VPL を用いると画像生成に膨大な計算時間を要するという問題がある．そのため，この問題を解決するための方法が多く提案されている[17, 58, 61]．これらの方法では，大量の VPL を生成しつつ，その中から少量の VPL をサンプリングし，大量の VPL すべてを用いた場合の輝度を推定する（図 1.4 参照）．推定結果には誤差が含まれるが，サンプル数を増やすことで誤差を低減できる．また，生成される VPL のほとんどは結果に与える影響が小さいため，結果に与える影響が大きい重要な VPL を重点的にサンプリングできれば，少量の VPL でも高精度に輝度の推定が可能である．図 1.3 に示すように，多光源レンダリング法では VPL の生成自体は計算コストが低く，輝度計算が画像生成の計算時間の大部分を占めるため，この方法により大量の VPL を生成しても高速に画像を生成できる．

多光源レンダリング法を効率化する別のアプローチとしては，VPL の寄与（VPL からの光が視点に到達する量）の計算コストを削減する方法がある．VPL の寄与計算では，VPL からの光が輝度計算点（以降，シェーディング点と呼ぶ）に到達するかの判定，つまり，VPL とシェーディング点間の可視判定が必要となる．可視判定を行うには，二点間を結ぶ線分とシーン中のすべての物体で交差判定を行う必要があるため，計算コストが非常に高く，VPL の寄与計算時間の大部分を占める（図 1.3 参照）．そのため，可視判定の高速化は多光源レンダリング法の高速化に直結する．可視判定を高速化する方法としては，線分とシーン中の物体の交差判定において，明らかに交差しない物体との交差判定を省略する方法[53]や，寄与の小さい VPL に対して，可視判定自体を省略する方法[51]などがある．

近年の多光源レンダリング法の研究により，大量の VPL を使用したとき計算時間がかかりすぎてしまうという問題は大幅に改善されてきている．しかしながら，多くの手法は経験則に基づいてるため，経験則が当てはまらない状況では大幅な高速化は望めない（例えば，文献[36]では近傍のシェーディング点で VPL の寄与が相関しているという経験則に基づき高速化を行っているが，鋭い反射特性を持つ材質ではこの経験則は当てはまらない）．また，手法の効率はアルゴリズムのパラメータに大きく依存するにも拘らず，適切なパラメータの設定方法は提案されていないことが多く，ユーザが手作業で適切なパラメータを設定する必要がある．パラメータによっては値を変更するとレンダリングを最初からやり直さなければならない場合もあり，ユーザはパラメータ調整に多大な労力と時間を強いられる．さらに，高い効率で画像生成ができるパラメータを発見できたとしても，最適なパラメータはシーンだけでなく，材質や視点が異なるだけでも変化するため，映像製作などこれらの要素が頻繁に変化する応用分野では，パラメータ調整の労力削減は極めて重要である．次節では，これらの問題点を踏まえた上で，多光源レンダリング法を高速化する 3 つの提案法の概要について説明する．

1.2 研究概要

本論文では，写実的な画像生成手法の一つである多光源レンダリング法において，計算効率を定式化することで，ヒューリスティックスに依らずパラメータ調整の少ないロバストかつ効率的な画像生成法を提案する．なお CG の分野においてロバスト性とは，シーンに依らず安定的な効率を出せる性質を指す．多光源レンダリング法は，仮想的な点光源である VPL を用いて画像生成を行う手法で，使用する VPL の数により光輸送問題に対する近似精度と計算時間を制御できる．この手法で写実的な画像を生成するには大量の VPL が必要であり，単純にすべての VPL の寄与を計算する場合，計算時間がかかりすぎてしまう．大量の VPL を効率的に扱う手法が多く提案されているが，依然として長い計算時間を要する．そこで，本論文では以下の 3 つのアプローチにより，実的な画像を効率的に写生成する．

- レイトレーシングの高速化
- 可視判定省略による高速化
- 誤差制御による高速化

多光源レンダリング法の基本的な処理の流れは図 1.5 に示す通りである．1 つ目のアプローチでは，VPL・シェーディング点の生成と VPL の寄与計算における VPL・シェーディング点間の可視判定を高速化し，2 つ目のアプローチでは，VPL の寄与計算における VPL・シェーディング点間の可視判定を省略することで高速化を行う．3 つ目のアプローチでは，ユーザが指定する許容誤差パラメータを基に自動的に VPL サンプル数を決定することで，ユーザが所望する画質の画像を一度のレンダリングで生成する．以下では，各アプローチの概要について述べる．

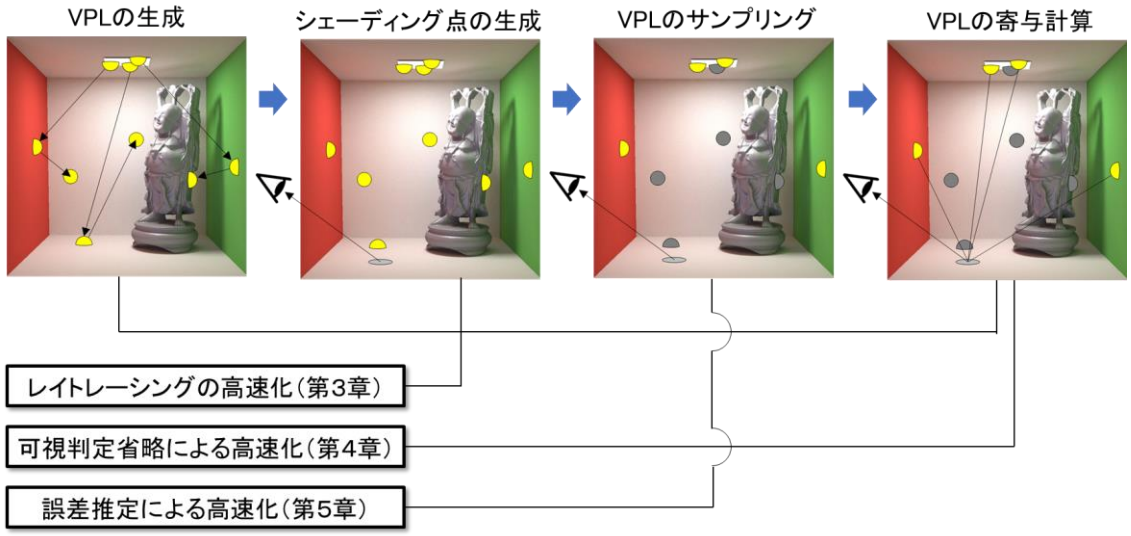
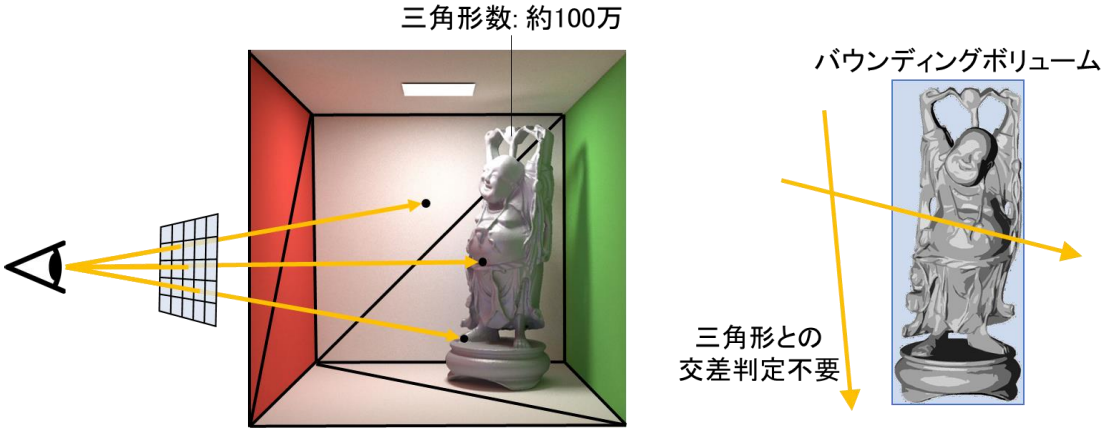


図 1.5　多光源レンダリング法の処理の流れと 3 つの提案法の関係.

1.2.1　レイトレーシングの高速化

レイトレーシングは，視点から光の経路を追跡することで画像を生成する手法である[8, 59]．レイトレーシングは，多光源レンダリング法を含む様々な大域照明アルゴリズムの基礎的技術であり，レイトレーシングの高速化は，あらゆる大域照明アルゴリズムの高速化につながる．レイトレーシングにおける基本的な計算は，ある点からある方向に進む光線（レイ）とシーン中の物体表面との交点計算である（図 1.6(a)参照）．多光源レンダリング法においては，VPL やシェーディング点の生成，VPL とシェーディング点間の可視性の判定に用いられる．レイとシーンの交点計算を行うには，シーン中の物体とレイで交差判定を行う必要がある．CG の分野では，物体表面を三角形ポリゴン（以降，三角形と呼ぶ）の集合で表すのが一般的であり，複雑なシーンでは三角形の数が膨大となるため，すべての三角形に対してレイとの交差判定を行うのは計算コストが非常に高い．そのため，レイトレーシング



(a) レイトレーシング　　(b) 高速化データ構造の例

図 1.6 (a) レイトレーシングでは，始点と方向で定義されるレイとシーンの交点計算を行うことで光の輸送経路を構築する．一般的に CG では物体を三角形ポリゴンの集合で表すため，レイとシーンの交点を計算するにはすべての三角形ポリゴンとレイで交差判定を行う必要がある．(b) 高速化データ構造の一つであるバウンディングボリュームは，三角形集合を包含する立体のことで，バウンディングボリュームとレイで交差判定を行い，交差しない場合はバウンディングボリューム内の三角形とレイも必ず交差しないため三角形との交差判定を省略できる．

を行う前にあらかじめ交差判定を高速化するためのデータ構造[2, 15, 44]を構築しておき，レイトレーシング時には，構築した高速化データ構造を参照し，交差判定を効率的に行うのが一般的である（図 1.6(b)参照）．ただし高速化データ構造を使用する場合，高速化データ構造を保持するための大量のメモリが必要となる問題がある，また，高品質な高速化データ構造を構築するには，レイの分布に関する情報が必要となるが，高速化データ構造の構築の段階ではレイの分布は未知であるため，レイの分布が一様と仮定して高速化データ構造を構築しているという問題がある．

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[1, 30, 34, 41]．この手法では，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ため，高速化データ構造の構築時にレイの分布の情報を持つ．しかしながら DACRT では，依然として高速化データ構造の構築にレイの分布の情報を利用せず，従来の高速化データ構造の構築方法と同様，レイの分布は一様と仮定しており，高速化データ構造の構築方法に改善の余地がある．そこで本研究では，DACRT においてレイの分布の情報を用いて高速化データ構造の構築を行うことで，追跡しているレイに最適化された高速化データ構造を構築する．また，高速化データ構造を用いたレイトレーシングの計算効率を定式化することで，効率的な高速化データ構造の利用方法を提案する．これにより，効率的にレイと三角形の交差判定を行うことができる．

1.2.2 可視判定省略による高速化

レイトレーシングの高速化により，多光源レンダリング法の計算時間のボトルネックである VPL とシェーディング点間の可視判定を効率的に行うことができる．しかしながら，可視判定の回数は膨大であるため，依然として画像生成には時間を要する．そこで本研究では，可視判定を確率的に行うことで，計算コストの高い可視判定の回数自体を削減する．ここで，可視判定を確率的に行うとは，VPL とシェーディング点が与えられたとき，可視判定を行うか省略するかをランダムに決定するという意味である．与えられた VPL とシェーディング点が，可視か不可視である確率が高い場合，正確に可視判定を行うのは無駄であるため，可視判定を省略する確率を高くするべきであり，可視か不可視かが曖昧な場合は，正確に可視判定を行うべきである．

可視判定を確率的に行う場合，すべての VPL から少量の VPL をサンプリングすることによるランダム性だけでなく，可視判定結果にもがランダム性が発生する．そのため，同じ VPL サンプル数では輝度の推定精度は必ず悪化する．しかしながら，計算コストの高い可

視判定をある確率で省略するため，与えられた時間内において，より多くの VPL サンプルを使用できるため，同じ VPL サンプル数では出射輝度の推定精度は悪化しても，同時間で比較したとき，サンプル数の増加により，より高精度な推定が可能となる．1 サンプル当たりの平均計算時間と推定精度はトレードオフの関係にあり，可視判定の省略確率を大きくすれば，1 サンプル当たりの平均計算時間は減少するが，推定精度も減少してしまう．本研究では，計算時間と推定精度の関係を定式化することで，画像生成の効率を最大化する可視判定の省略確率を導出する．

1.2.3 誤差推定による高速化

レイトレーシングの高速化，可視判定省略による高速化により，高速な画像生成が可能となったが，これらのアプローチではシェーディング点における出射輝度の推定値の誤差を制御していないという問題がある．つまり，これらのアプローチではユーザによって VPL サンプル数などのパラメータが指定されたときに，より短い時間で計算を終えることができるが，推定値に含まれる誤差の大きさは知ることができない．そのため，過度に大きい VPL サンプル数を指定すれば，誤差は小さいが計算時間がかかりすぎてしまい，過度に小さい VPL サンプル数を指定すれば，計算時間は短いが誤差の大きい推定値が得られ，結果画像にノイズが発生する．ノイズの大きい画像が得られた場合，ユーザはパラメータを調整し再度レンダリングを行う必要があり，ノイズを視認できないレベルの画像を生成するのに，結局長い計算時間がかかってしまう．そのため，推定値の誤差を制御しつつ高速に画像を生成する方法が必要である．しかしながら，これまでの多光源レンダリング法の研究では，適切なパラメータ設定が行われている下で，いかに高速に画像を生成できるかに主眼が置かれており，ノイズの大きい画像が得られたときの再レンダリングの必要性を無視していた．誤差制御を試みる手法[7, 55, 56, 57]も存在するが，ヒューリスティックスに基づく手

法であり，推定誤差を過少評価する傾向があるため，依然としてユーザはパラメータ調整とレンダリングを繰り返す必要があった．

そこで本研究では，推定値の誤差を正確に制御しつつ効率的に写実的な画像を生成する手法を提案する．本手法では，刺激の弁別閾（差を知覚できる最小の差）は基礎刺激の強度に比例するというウェーバーの法則[6]に基づき，推定値の相対誤差を制御する．誤差の計算には真値が必要であるが，真値は推定対象そのものであるため，提案法では統計的なアプローチにより誤差の大きさを推定する．本手法では，ユーザは許容相対誤差と，相対誤差が許容範囲に収まる確率である信頼度パラメータを指定するだけでよく，VPL サンプル数は，これらのパラメータに基づき自動で設定される．提案法により，一度のレンダリングでユーザの所望する画質の画像を生成できる．

1.3 論文構成

本論文の構成は以下の通りである．第 2 章では，提案法に関連するレンダリングの基礎について説明する．第 3 章から第 5 章では，提案法であるレイトレーシングの高速化，可視判定省略による高速化，誤差推定による高速化について，詳細な研究背景と関連研究を交えて説明し，先行研究との比較実験の結果を示す．最後に第 6 章で，本論文のまとめを行う．

第２章　レンダリングの基礎

本章では，CG による写実的な画像の生成方法の基礎であるレンダリング方程式，およびレンダリング方程式を解く手法として広く用いられているモンテカルロ法について説明する．レンダリング方程式は，物体表面上の点における光の振る舞いを記述する積分方程式であり，モンテカルロ法は，乱数を用いた積分の数値計算法である．また，レンダリング方程式とモンテカルロ法を基に写実的な画像を効率的に生成する多光源レンダリング法，多光源レンダリング法を含む様々な大域照明アルゴリズムの基盤技術であるレイトレーシングについて説明する．

2.1　モンテカルロ法

CG による画像生成では，ピクセル値の計算など様々な場面で積分計算が必要となる．しかしながら，ほとんどの場合で解析的な積分計算が困難であるため近似計算が必要となる．低次元の関数に対しては区分求積法を用いれば効率的に高精度な近似が可能であるが，CG で扱う関数は高次元であることが多いため区分求積法を用いるのは非効率である．そのため，CG では乱数を用いた積分の近似法であるモンテカルロ法を用いる．

モンテカルロ法は次元数に関係なく適用できる手法であるが，説明簡略化のため以下の式で定義される一次元の積分を考える．なお，本節で説明するモンテカルロ法の特性は，多次元の場合にも当てはまる．

$$I = \int_D f(x)dx \tag{2.1}$$

ここで，fは定義域D上で定義された実数値関数である．モンテカルロ法では，確率密度関数pに従い独立に生成したN個のサンプル$X_1, \dots, X_N$を用いて，以下の式で積分値を推定する．

$$I \approx F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \tag{2.2}$$

確率密度関数pに従うサンプルは逆関数法を用いることで生成できる[38]．サンプル$X_1, \ldots, X_N$は確率変数であることから，それらを用いた推定値F_Nもまた確率変数であり，推定値F_Nの期待値$E[F_N]$は以下の式で表される．

$$\begin{aligned} E[F_N] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}E\left[\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_D \frac{f(x)}{p(x)}p(x)dx \\ &= I \end{aligned} \tag{2.3}$$

上式より，推定値F_Nの期待値は真値と一致することが分かる．そして，F_Nの分散$V[F_N]$は以下の式で表される．

$$\begin{aligned} V[F_N] &= V\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N^2}\sum_{i=1}^{N}V\left[\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N}V\left[\frac{f(X)}{p(X)}\right] \end{aligned} \tag{2.4}$$

ここで，Xは確率密度関数pに従う確率変数である．上式より，モンテカルロ法では，サンプル数Nを大きくするほど推定値の分散が減少し，サンプル数Nが無限のとき分散が 0 となることが分かる．ただし，モンテカルロ法の計算時間はサンプル数に比例して増加するため，単純にサンプル数を増やして分散を減少させる方法は非効率である．以降では，効率的に推定を行う手法である重点的サンプリングとロシアンルーレット法について説明する．

重点的サンプリング

積分値の計算において被積分関数fの値が小さい領域よりも，値が大きい領域の方が積分値に与える影響が大きい．重点的サンプリングは，被積分関数fの値が大きい領域を重点的にサンプリングすることで分散を低減する方法である．式(2.4)において分散$V[f(X)/p(X)]$に注目すると，この分散は以下の式で表される．

$$\begin{aligned} V\left[\frac{f(X)}{p(X)}\right] &= E\left[\frac{f(X)^2}{p(X)^2}\right] - E\left[\frac{f(X)}{p(X)}\right]^2 \\ &= \int_D \frac{f(x)^2}{p(x)} dx - \left\{\int_D f(x)dx\right\}^2 \end{aligned} \tag{2.5}$$

上式より，確率密度関数pが被積分関数fに比例するほど分散が減少し，$p(x) = f(x)/\int_D f(x)dx$のとき分散は 0 となることが分かる．ただし，分散が 0 となる理想的な確率密度関数には，推定対象であり未知の値である$\int_D f(x)dx$が必要であるため使用することはできない．実際の応用では，積分計算が可能な関数$g(x) \approx f(x)$を用意し，$p(x) = g(x)/\int_D g(x)dx$としてサンプリングを行う．または，被積分関数$f$が 2 つの関数$f_1$と$f_2$の積$f_1 f_2$で表され，$f_1$か$f_2$が積分可能のとき，$p(x) = f_1(x)/\int_D f_1(x)dx$や$p(x) = f_2(x)/\int_D f_2(x)dx$としてサンプリングを行う．被積分関数の形状にもよるがほとんどの場合で，定義域上を一様にサンプリングするよりも，被積分関数の一部に比例した確率密度関数を使用した方が分散を低減できる．

なお，被積分関数に比例した確率密度関数に従うサンプル列を生成する方法としてマルコフ連鎖モンテカルロ法があるが，この方法では理想的な確率密度関数に従うサンプル列を生成できるが，確率密度関数の値は計算できないという問題があり，積分値の推定に使用することはできない．重点的サンプリングとマルコフ連鎖モンテカルロ法を組み合わせ，レンダリングに応用した方法もいくつか提案されているが，本研究との関連は小さいため詳細は[51, 52]を参照されたい．

ロシアンルーレット法

モンテカルロ法では，サンプル数を増やすほど分散が減少し，推定精度を向上させることができる．ロシアンルーレット法は，同時間において，より多くのサンプルを用いることで分散を低減する手法である．この手法では，従来通りの方法でサンプルを生成したのち，被積分関数fを以下の式で置き換えて積分値を推定する．

$$\hat{f}(x) = \begin{cases} \dfrac{f(x)}{q(x)} & \text{確率} q(x) \text{で選択} \\ 0 & \text{それ以外} \end{cases} \tag{2.6}$$

この方法では，確率qで実際に被積分関数fを評価し，それ以外（確率$1 - q(x)$）では被積分関数fの評価を省略する．そのため，1 サンプル当たりの平均計算時間が減少し，同じ計算時間において，より多くのサンプルを用いることができる．なお，以下に示すように$\hat{f}$の期待値は真値fと一致するため，ロシアンルーレット法を用いても積分の推定値は真値と一致する．

$$\begin{aligned} E\left[\hat{f}(x)\right] &= \frac{f(x)}{q(x)} q(x) + 0\big(1 - q(x)\big) \\ &= f(x) \end{aligned} \tag{2.7}$$

この方法では積分値の推定において，サンプル生成のランダム性だけでなく，ロシアンルーレット法においてもランダム性が発生する．そのため，同じサンプル数の下では，推定値の分散は必ず増加する．分散の増加量と 1 サンプルに対する平均計算時間の減少量は，確率qに依存するため，確率qの適切な設定が重要となる．

これまでモンテカルロ法を用いて積分値を推定する方法を説明してきたが，モンテカルロ法は総和の推定にも用いることができ，離散点$x_1, \ldots, x_M$におけるfの総和は以下の式で推定される．

$$\begin{aligned} I &= \sum_{i=1}^{M} f(x_i) \\ &\approx \frac{1}{N} \sum_{j=1}^{N} \frac{f(X_j)}{p(X_j)} \end{aligned} \tag{2.8}$$

ここで，$X_i \in \{x_1, \ldots, x_M\}$は$i$回目の試行におけるサンプルであり，確率質量関数$p$に従い生成される．総和の推定値に対する期待値や分散の特性などは，積分の推定値に対するものと同様である．

2.2　レンダリング方程式

写実的な画像を生成するには光の振る舞いを物理則に基づき計算する必要がある．レンダリング方程式は，物体表面における光の振る舞いを記述する方程式であり，物体表面上の点xから方向ω_oに出射する輝度$L(x, \omega_o)$は以下の式で計算される（図 2.1 参照）[26]．

$$L(x, \omega_o) = L_e(x, \omega_o) + L_r(x, \omega_o) \tag{2.9}$$

ここで，$L_e(x, \omega_o)$は点xが光源上である場合の発光により方向ω_oに出射する輝度であり，$L_r(x, \omega_o)$は点xに入射した光が方向ω_oに反射する輝度である．L_rは以下の式で計算される．

$$L_r(x, \omega_o) = \int_{\Omega} L(x, \omega_i) f(x, \omega_i, \omega_o) |\cos\theta| d\omega_i \tag{2.10}$$

ここで，Ωは物体表面の上半球，ω_iは入射方向，fは BRDF（Bidirectional Reflectance

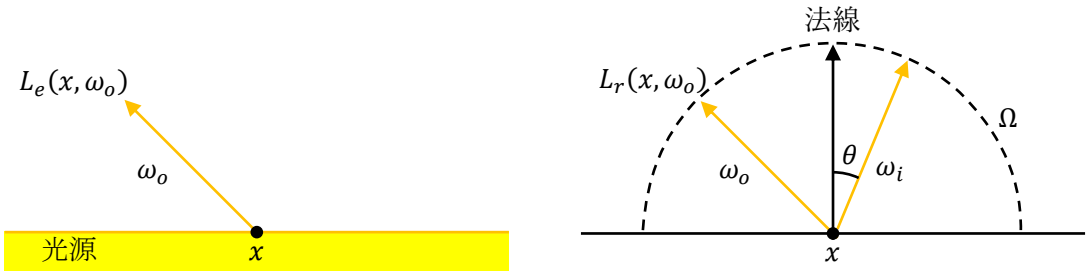


図 2.1　レンダリング方程式．点xから方向ω_oに出射する輝度$L(x,\omega_o)$は，自己発光による出射輝度$L_e(x,\omega_o)$と，入射光の反射による出射輝度$L_r(x,\omega_o)$の和で計算される．反射による輝度の計算は，あらゆる方向から入射する光を考慮する必要がある．

Distribution Function)，θは入射方向と物体表面法線のなす角，$d\omega_i$は微小立体角である．BRDF $f(x,\omega_i,\omega_o)$は，点xに方向ω_iから入射した光が方向ω_oに反射する割合を表す関数であり，物体の材質により異なる．立体角は，2 次元における平面角を 3 次元に拡張したもので，単位球上における面積である．

式(2.10)では，光の振る舞いの直感的な解釈のため微小立体角に関する積分で反射輝度を表していたが，シーン中の 3 点x，x'，x''を用いて，微小面積に関する積分に書き直すことができる．立体角は単位球上における面積であることから，シーン中の微小面積と微小立体角の関係は以下の式で表される（図 2.2 参照）．

$$d\omega_i = \frac{|\cos\theta'|}{\|x-x'\|^2} V(x,x')dA(x') \tag{2.11}$$

ここで，θ'は点x'から点xの方向と点x'における法線のなす角，Vは可視関数，dAは微小面積である．可視関数Vは，2 点間の可視性を表す関数であり以下の式で計算される．

$$V(x,x') = \begin{cases} 1 & x\text{と}x'\text{が互いに可視} \\ 0 & \text{それ以外} \end{cases} \tag{2.12}$$

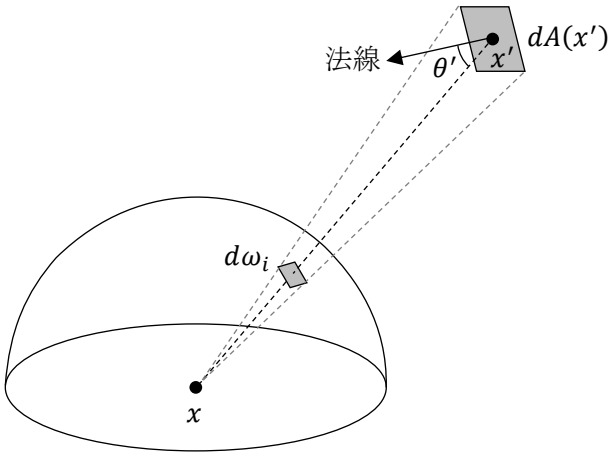


図 2.2　微小面積と微小立体角の関係. 微小面を単位球に投影したときの面積が微小立体角である.

式(2.9)，式(2.10)，式(2.11)より，点xから点x''の方向に出射する輝度は，以下の式で計算される.

$$\begin{aligned} L(x,x'') &= L_e(x,x'') + \int_{\mathcal{M}} L(x',x) f(x',x,x'') \frac{|\cos\theta||\cos\theta'|}{\|x-x'\|^2} V(x,x') dA(x') \\ &= L_e(x,x'') + \int_{\mathcal{M}} L(x',x) f(x',x,x'') G(x,x') V(x,x') dA(x') \end{aligned} \tag{2.13}$$

ここで，$\mathcal{M}$はシーン中の物体表面を表し，$G(x,x') = \frac{|\cos\theta||\cos\theta'|}{\|x-x'\|^2}$は幾何項と呼ばれる．次節では 3 点形式のレンダリング方程式を用いて，多光源レンダリング法の定式化を行う.

2.3　多光源レンダリング法

CG による画像生成は各ピクセルの値を計算することで行われる．各ピクセルの値は，そのピクセルを通り視点に到達する放射輝度から計算される．視点をx_v，ピクセルに対応するシェーディング点をxとすると，視点x_vに到達する放射輝度は以下の式で表される.

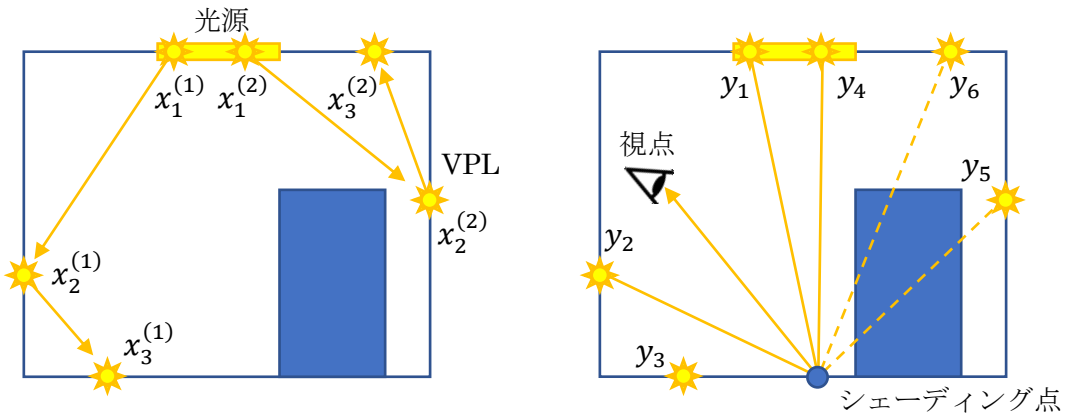
$$L(x, x_v) = L_e(x, x_v) + \int_{\mathcal{M}} L(x', x) f(x', x, x_v) G(x', x) V(x', x) dA(x') \tag{2.14}$$

上式は再帰式になっており，点x'から点xに入射する輝度$L(x', x)$を計算するには，点x'においてもレンダリング方程式を解く必要がある．入射輝度$L(x', x)$にレンダリング方程式の代入を繰り返すことで以下の式が得られる．

$$\begin{aligned}
&L(x, x_v) \\
&= L_e(x, x_v) \\
&+ \int_{\mathcal{M}} L_e(x_1, x) f(x) G(x_1, x) V(x_1, x) dA(x_1) \\
&+ \int_{\mathcal{M}} \int_{\mathcal{M}} L_e(x_1, x_2) f(x_2) G(x_1, x_2) V(x_1, x_2) f(x) G(x_2, x) V(x_2, x) dA(x_1) dA(x_2) \\
&+ \sum_{n=3}^{\infty} \int_{\mathcal{M}} \dots \int_{\mathcal{M}} L_e(x_1, x_2) T(x_1, \dots, x_n) f(x) G(x_n, x) V(x_n, x) dA(x_1) \dots dA(x_n)
\end{aligned} \tag{2.15}$$

$$T(x_1, \dots, x_n) = \prod_{i=1}^{n-1} f(x_{i+1}) G(x_i, x_{i+1}) V(x_i, x_{i+1}) \tag{2.16}$$

なお表記簡略化のため，$f(x', x, x'')$を$f(x)$と置いた．上式において，n重積分の項は，光源から出射した光がn回反射して視点に到達する輝度を表しており，視点に到達する輝度は，光源からの光が直接視点に到達する輝度と$n(\geq 1)$回反射して視点に到達する輝度の和で計算される．ただし，各積分式は解析的に解けないため，モンテカルロ法で各積分値を推定する．光源からの光がn回反射して視点に到達する輝度を推定する場合，シーン中のn個の点$x_1, \dots, x_n$が 1 サンプルに相当し，多光源レンダリング法では光源側の点x_1から順にサンプリングしていくことで生成する．x_1は光源上の物体表面をサンプリングすることで生成でき，x_2以降は光の放射，反射方向をサンプリングし，シーンとの交点を計算することで生成できる．生成された点$x_1, \dots, x_n$は，他の積分値の推定に再利用が可能で，$x_1, \dots, x_m$を$m(< n)$回反射して視点に到達する輝度の推定のためのサンプルとして使用できる．そのため，最大反射



(a) VPL の生成　　(b) 輝度計算

図 2.3　多光源レンダリング法．まず，光源側からシーン中の点をサンプリングしていくことで VPL を生成する．次にシェーディング点では，あらゆる方向から入射する光を VPL からの光で近似し，視点に到達するエネルギーを計算する．なお左図において$x_i^{(j)}$はj回目の試行における光源からi番目の点である．

回数をn_{max}とすると，$x_1, \ldots, x_{n_{max}}$をN回サンプリングするだけで，反射回数がn_{max}以下で視点に到達する輝度の推定が可能である．また，これらのサンプルは，シェーディング点に依存しないため，シェーディング点毎に生成する必要はなく，一度だけ生成するだけでよい．

多光源レンダリング法では，生成したシーン中の点を VPL（Virtual Point Light）と呼び，式(2.14)は VPL yを用いて以下の式で推定される（図 2.3 参照）．

$$L(x, x_v) \approx L_e(x, x_v) + \sum_{i=1}^{N_{vpl}} I(y_i, x) f_r(y_i, x, x_v) G(y_i, x) V(y_i, x) \tag{2.17}$$

ここで，N_{vpl}は VPL 数，Iは放射強度であり，放射強度I，BRDFf_r，幾何項G，可視関数Vの積は VPL の寄与と呼ばれる．VPL yが光源からn番目の点とすると，放射強度Iは以下の式で計算される．

$$I(y,x)=\begin{cases}\dfrac{L_e(x_1,x)}{Np(x_1)} & n=1 \text{ の場合}\\ \dfrac{L_e(x_1,x_2)T(x_1,\dots,x_n)}{Np(x_1,\dots,x_n)} & \text{それ以外}\end{cases} \tag{2.18}$$

上式を用いれば，多光源レンダリング法による推定値の期待値は真値と一致する．しかしながら，$L_e(x_1,x)$または$f_r(x_{n-1},x_n,x)$の指向性が強いとき，VPL が照らす領域は非常に狭まり，結果画像にアーティファクトが発生する場合がある．そのため，多光源レンダリング法の研究では，x_nにおけるL_eまたはf_rの指向性を無視することが多く[7, 17, 55, 56, 58, 61]，本研究でも同様の近似を行い，放射強度を$I(y)$で表す．

本研究では，可視判定回数の削減や誤差推定により多光源レンダリング法を効率化するが，その他のアプローチで多光源レンダリング法を効率化する研究もあり，ここでいくつか紹介する．多光源レンダリング法では使用する VPL の数を増やすことで近似精度が向上し，写実的な画像を生成できる．しかしながら，反射や放射の指向性が強い材質が存在する場合など，単純に VPL 数を増やすだけでは，十分な近似精度を得るのに莫大な VPL が必要になる場合がある．そのため，少ない VPL でも十分な近似精度を得るための研究が多く行われている[22, 35, 45, 48]．これらの手法では，点光源ではなく面積や長さを持った仮想的な光源を生成することで問題を解決している．また，シェーディング点を可視点以外にも生成する方法[13, 31]や，結果に与える影響の少ない VPL の生成を抑える方法[19]，視点側から VPL を生成する方法[12]により，この問題を解決する方法もある．これらの研究により多光源レンダリング法は，あらゆるシーンを効率的にレンダリングできる実用的な手法となってきている．

2.4　レイトレーシング

レイトレーシングは光（レイ）の経路を視点側から追跡することで画像を生成する手法[8, 44]であり，多光源レンダリング法を含む様々な大域照明アルゴリズムの基盤的な技術である．この手法における基本的な処理は，レイとシーンの交点計算であり，多光源レンダリング法においては，VPL やシェーディング点の生成，VPL とシェーディング点間の可視関数の評価に用いられる．レイとシーンの交点計算では，始点と方向で定義されるレイと，シーン中のすべての物体で交差判定を行い，最も始点に近い交点を計算する．CG では物体を三角形の集合で表現するのが一般的であり，複雑なシーンでは三角形の総数は膨大となるため，すべての三角形に対して交差判定を行うのは計算コストが非常に高い．

シーン中の三角形とレイの交差判定を効率化する方法として，高速化データ構造の利用がある．高速化データ構造は画像生成に先立って構築するもので，格子[15]，kd 木[2]，バウンディングボリューム階層（BVH, Bounding Volume Hierarchy）[44]などの種類がある．ここでは，本研究に関係がある BVH について説明する（図 2.4 参照）．バウンディングボリュームは，複数の三角形を包含する立体で，立体の形状には 3 次元座標系の各軸に平行な直方体（AABB, Axis Aligned Bounding Box）がよく用いられる．バウンディングボリュームを用いることで，バウンディングボリュームとレイが交差していないと判定されれば，包含する三角形集合とレイも交差していないと判定することができ，大幅にレイと三角形の交差判定回数を削減できる．バウンディングボリュームによる効率化の度合いは，バウンディングボリュームの大きさに依存する．多くの三角形を包含する大きいバウンディングボュームを用いた場合，レイと交差する確率が高くなるため，包含する三角形集合とレイの交差判定を省略できる確率は小さくなる．一方，包含する三角形が少なく，小さいバウンディングボリュームを用いた場合，レイと交差する確率は小さくなり，高確率で三角形との交差判定を省略できるが，多数のバウンディングボリュームが必要となるため，レイとバウン

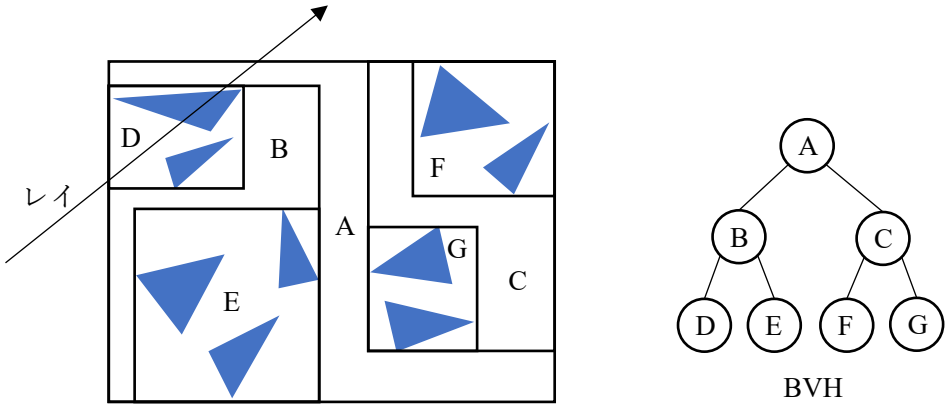


図 2.4　BVH を用いたレイトレーシング．バウンディングボリュームとレイが交差しない場合，バウンディングボリュームが包含する三角形とレイも交差しないため，三角形とレイの交差判定を省略できる．BVH はバウンディングボリュームを木構造により階層化したもので，この例では，ノード A，B，D，E，C の順で探索し，ノード D が包含する三角形とだけレイと交差判定を行う．

ディングボリュームの交差判定コストが問題となる．この問題を解決するため，バウンディングボリュームを木構造で階層化したものが BVH である．BVH の葉ノードはいくつかの三角形を包含するバウンディングボリュームを表し，内部ノードは子ノードを包含するバウンディングボリュームを表す．BVH を用いた交差判定は，根ノードからの深さ優先探索により行われる．各ノードでは対応するバウンディングボリュームとレイで交差判定を行い，交差しない場合は，そのノードの探索を終了する．交差する場合は，そのノードが内部ノードなら子ノードを探索し，葉ノードならバウンディングボリュームが包含する三角形と交差判定を行う．このように，BVH を用いることで効率的にレイとシーン中の三角形の交差判定を行うことができる．

高速化データ構造を用いることで，レイとシーン中の三角形の交差判定を大幅に効率的に行うことができる．しかしながら，高速化データ構造の構築は時間がかかる処理であり，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，効率的に高速

化データ構造を構築する方法が必要である．また，高速化データ構造はメモリを大量に使用する場合が多いため，メモリ使用量に制限がある環境でもレイトレーシングを実行するために，メモリ使用量を抑えた高速化データ構造の構築方法が必要である．

第３章　レイトレーシングの高速化

3.1　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[27]，kd 木[63]，バウンディングボリューム階層（Bounding Volume Hierarchy, BVH）[53]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りつつある．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（物体を構成する幾何形状の最小単位，三角形など）の交差判定を行う．

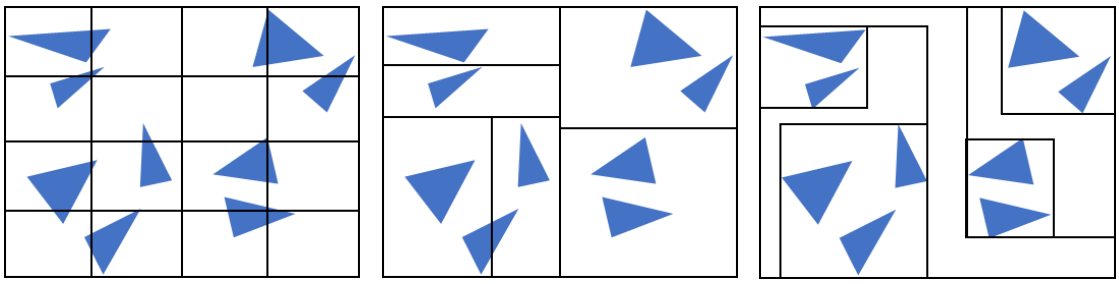
近年，高速化データ構造の構築とレイトレーシングを同時に行うことで，高速化データ構造のための追加のメモリが不要な DACRT（Divide-And-Conquer Ray Tracing）が提案された[1, 30, 34, 41]．この手法では，大量のレイとプリミティブの交差判定問題を分割統治法を用いて解いている．分割統治法とは，そのままでは解けないような問題を小さな問題へ分割し，分割された問題を解くことで元の問題を解く方法である．DACRT における問題は，すべてのレイに対して最初に交差する物体表面との交点を求めることであり，問題サイズはレイの数とプリミティブの数の積に相当する．先行研究[1, 30, 34, 41]では，プリミティブの分布のみを考慮して交差判定問題を分割しており，レイの空間的な分布を考慮していなかった．そのため，問題を分割してもレイの数が大きく削減されないことがあり，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．近年のディスプレイの高解像度化により，高解像度画像の需要は高まっており，レイの分布を考慮して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，レイサンプリングで得られたレイの分布を基に，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．しかしながら，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．そこで本研究では，問題分割の計算効率を定式化し，非効率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）やシャドウレイ（可視判定のためのレイ），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．

3.2　関連研究

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[15]，kd 木[2]，階層バウンディングボリューム（Bounding Volume Hierarchy, BVH）[44]など（図 3.1 参照））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一度だけ構築していれば良いため，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[16, 28, 37, 53, 62]．



(a) 格子　(b) kd 木　(c) BVH

図 3.1　様々な高速化データ構造の例．(a) 格子は各座標軸に垂直な平面を等間隔に配置し空間を分割．(b) kd 木は座標軸の一つに垂直な平面を適応的に配置し空間を分割．(c) BVH は三角形を包含する境界立体をさらに包含する境界立体を考え階層化．

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[1, 30, 34]．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[34]は，円錐状のパケット（複数のレイをまとめたもの）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．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[1]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. Ravichandran ら[41]は，DACRT の GPU による実装を行った．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．

レイの分布を考慮した高速化データ構造の構築と探索を行う研究は多く行われている．Bittner ら[5]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．しかしながら，この手法では，あらゆる種類のレイ（プライマリレイ，センカンダリレイなど）の分布を同時に考えるため，レイのコヒーレンスが失われ，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 ら[14]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．しかしながら，シャドウレイに最適化された BVH のための追加の構築時間，メモリが必要である．

3.3 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，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，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分割することで，交差判定問題を二つの小問題に分割する．分割された各問題でも同様に，分割を繰り返していき，どちらかの集合の要素が十分に小さくなれば, 問題サイズが十分小さいとして, 集合内の全てのレイと三角形で交差判定を行い交差判定問題を解く．すべての小問題を解けたとき，全てのレイで始点に最も近い三角形との交点が計算されたことになる．問題の分割を行う際は，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

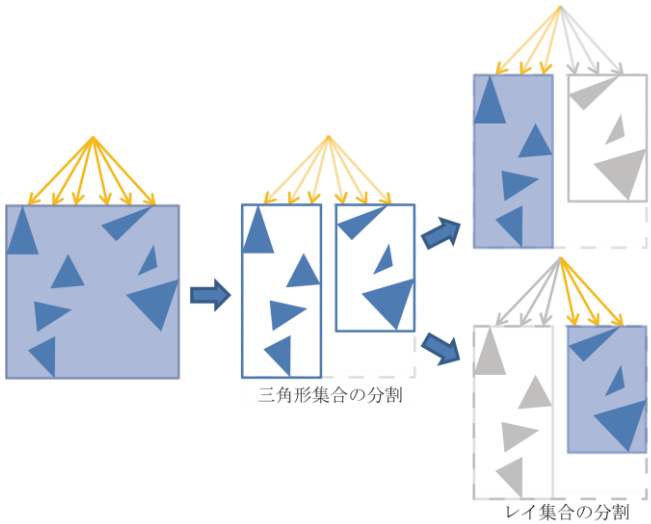


図 3.2　問題分割の流れ．三角形集合を二分し，分割された三角形集合を包含するバウンディングボリュームとレイ集合で交差判定を行い，交差するレイの集合を計算することで問題を分割する．

の流れを図 3.2 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．三角形集合のよい分割を行うには，以下のコスト関数Cに従い分割すればよい（図 3.3 参照）．

$$C(V \to V_L, V_R) = C_T + C_I(p_L N_L + p_R N_R) \tag{3.1}$$

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と交差しているレイがV_L，V_Rと交差する確率，N_L，N_RはV_L，V_R内に含まれる三角形の数である．なお，$p_L + p_R \neq 1$であることに注意されたい．式(3.1)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

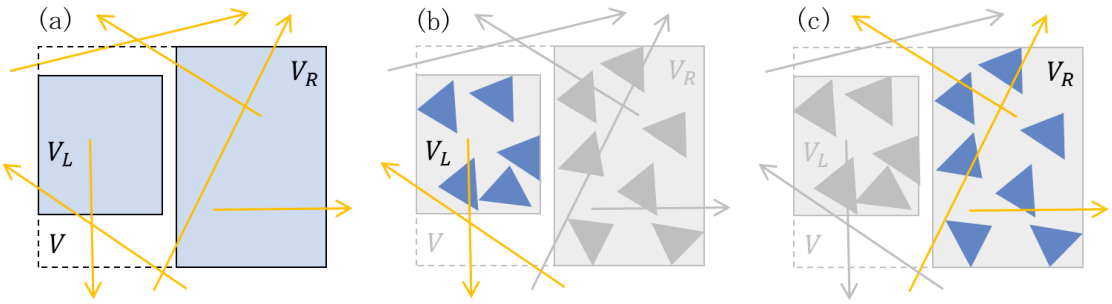
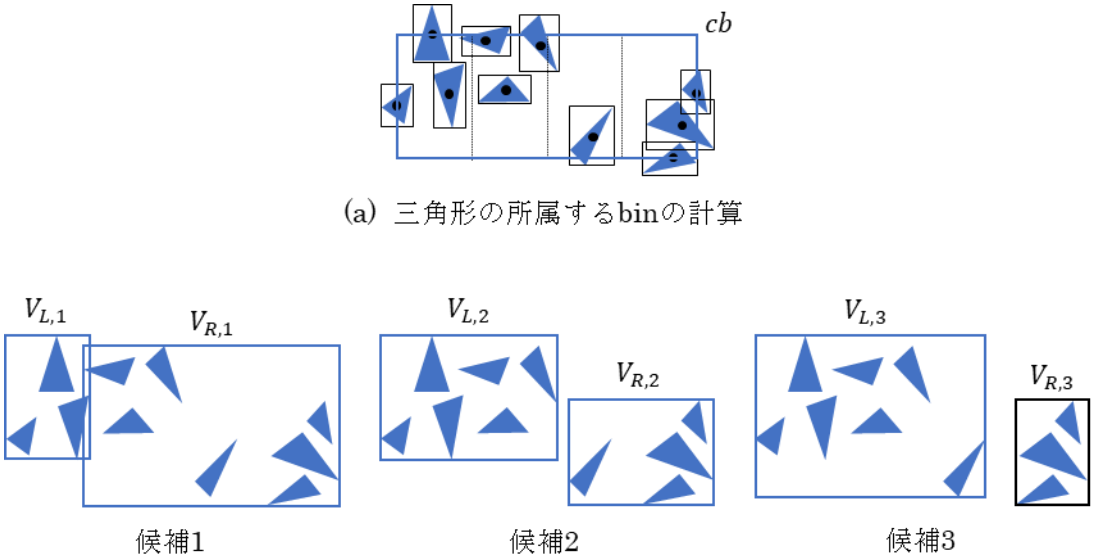


図 3.3　三角形集合の分割におけるコスト関数の説明図．(a) BV Vと交差するレイはVを分割してできる BV V_LとV_Rとの交差判定によりコストC_Tが発生する．(b) V_Lと交差するレイは今後V_Lに含まれるN_L個の三角形と交差判定を行うため，Vと交差するレイがV_Lと交差する確率をp_L，レイと三角形の交差判定コストをC_Iとすると，レイ 1 本当たり平均で$C_I p_L N_L$のコストが発生する．(c) V_Lの場合と同様の議論より，レイ 1 本当たり平均で$C_I p_R N_R$のコストが発生する．

した SAH（Surface Area Heuristic）コスト関数が一般的に使用されている[54]．表面積ではなく体積の割合で交差確率を近似する方法も考えられるが，体積を用いた場合，厚さが 0 の BV では交差確率が 0 となってしまい，交差確率の良い近似を行えない．

3.4　提案法

アルゴリズム 3.1 に，提案法のアルゴリズムの擬似コードを示す．このアルゴリズムでは，入力としてレイ集合R，三角形集合T，三角形集合の各三角形の BV の重心を包含する BV であるcb（図 3.4(a)参照）を受け取り，レイ集合Rと三角形集合T間の交点を計算する．なお，各重心を包含する BV であるcbは，三角形集合の分割に使用するもので，レイとの交差判定には使用しないことに注意されたい．本研究では，実装の容易性やパフォーマンスの観点から，BV として軸平行境界ボックス（Axis Aligned Bounding Box, AABB）を用いるが，その他の形状の BV でも提案法を適用できる．提案法では最初に，問題サイズが十分に



(a) 三角形の所属するbinの計算

(b) 分割候補の生成

図 3.4　binning による三角形集合の分割候補の生成．三角形は AABB の重心が含まれる bin に所属し，三角形集合の分割は bin 間の境界で行われる．

小さいか判定する．提案法では従来法と同様，レイの数が閾値δ_R以下または三角形の数が閾値δ_T以下のとき問題サイズが十分に小さいと見なす．問題サイズが十分に小さい場合は，レイ集合と三角形集合で交差判定を行い，問題サイズが大きい場合は，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 binning ベースの手法[53]を用いる．bin とは，BV をある軸に垂直な面で等分割したときの各領域のことを指す．この手法では，三角形集合の分割は bin 間の境界で行われる．K個の bin を用いる場合，$K-1$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（図 3.4 参照）．ここで，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．

アルゴリズム 3.1 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

アルゴリズム 3.1　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_Rとδ_T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．関数 NaiveRT は与えられたレイ集合と三角形集合のすべての組み合わ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(3.2)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:     (α_0, T_0, V_0, cb_0) ← (α_{L,j_min}, T_{L,j_min}, V_{L,j_min}, cb_{L,j_mn})
20:     (α_1, T_1, V_1, cb_1) ← (α_{R,j_min}, T_{R,j_min}, V_{R,j_min}, cb_{R,j_mn})
21:   else
22:     (α_0, T_0, V_0, cb_0) ← (α_{R,j_min}, T_{R,j_min}, V_{R,j_min}, cb_{R,j_mn})
23:     (α_1, T_1, V_1, cb_1) ← (α_{L,j_min}, T_{L,j_min}, V_{L,j_min}, cb_{L,j_mn})
24:   end if
25:   for i = 0 to 1 do
26:     if α_i > 1 − C_bv / (n_child C_child) then
27:       DACRT(R, T_i, cb_i)                   ▽レイ集合分割を省略
28:     else
29:       DACRT(R ∩ V_i, T_i, cb_i)
30:     end if
31:   end for
32: end procedure
```

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分割候補を選択する．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．最後に，レイサンプリングで得たレイ集合と BV の交差割合αを用いて，レイ集合と BV で交差判定を行うか判定し，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，通常通りレイ集合の分割を行い再帰処理を続ける．

3.4.1　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．サンプリングされたレイ（以降，サンプルレイと呼ぶ）に対する処理の疑似コードはアルゴリズム 3.2 に示す通りである．レイサンプリングでは，レイと BV 間の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，$n_{L,j}$，$n_{R,j}$はサンプルレイが$V_{L,j}$と$V_{R,j}$のどちらと先に交差するかを記録するためのものである．アルゴリズム 3.2 では，まず，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[60]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サンプルレイが$V_{L,j}$と$V_{R,j}$のどちらにより近いかを判定し，$V_{L,j}$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．これらの処理を各分割候補に対して行う．なお，レイサンプリングにはコストがかかるため，レイの数が十分多いときのみ，レイサンプリングを行い，レイの数が少ないときは，Afra [1]の提案した DACRT を行う．

アルゴリズム 3.2　サンプルレイrと BV である$V_{L,j}$，$V_{R,j}$の交差判定アルゴリズム．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[60]，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
```

3.4.2　コスト関数

Afra の手法[1]では，コスト関数として SAH コスト関数を使用し三角形集合を分割している．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，レイと BV が交差する確率は BV の表面積に依存するため，表面積の割合は交差確率のよい近似となるが，分布が偏っている場合は，よい近似とは言えない．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であるが，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．提案法では，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．

$$C(V \to \{V_L, V_R\}) = C_T + C_I\left(\alpha_{L,j} N_{L,j} + \alpha_{R,j} N_{R,j}\right) \tag{3.2}$$

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 の構築を行うことができる．

3.4.3　探索順序

効率的に問題サイズを削減する上で，ノードの探索順序は重要な要素のひとつである．レイの始点から三角形との交点までの距離が，レイの始点から BV との交点までの距離より短い場合，そのノードには，よりレイの始点に近い三角形と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 の手法[1]では探索順序を 1 本のレイを使用して決定していた．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効率の低下を招いてしまう．そこで，レイサンプリング時に計算しておいた$n_{L,j}$，$n_{R,j}$のより値の大きい方のノードを先に探索する．これにより，より安定して効率的な探索順序の決定を行うことができる．

3.4.4　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 3.5 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みなし，次の処理へ移る方が効率的となる．交差判定を省

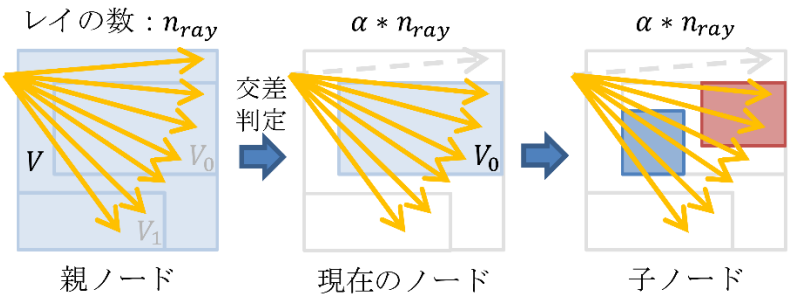


図 3.5　レイ集合の分割が非効率な場合．レイ集合の大部分が BV V_0と交差するため，レイ集合の分割による，問題サイズの大きな削減ができない．レイ集合と現在のノードの BV の交差確率をαとすると，交差判定後のレイ集合の数はαn_{ray}になると予想でき，削減数が小さい場合はレイ集合の分割を省略するべきである．

略した場合の処理の流れを図 3.6 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αn_{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，αn_{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．なお，現在のノードが内部ノードの場合，$C_{child}$はレイと BV の交差判定コストであり，現在のノードが葉ノードの場合，C_{child}はレイと三角形の交差判定コストである．もし様々なプリミティブの種類を同時に扱う場合，葉ノードにおけるC_{child}はプリミティブの種類により異なるが，式変形は自明であり省略する．以上より，レイと BV

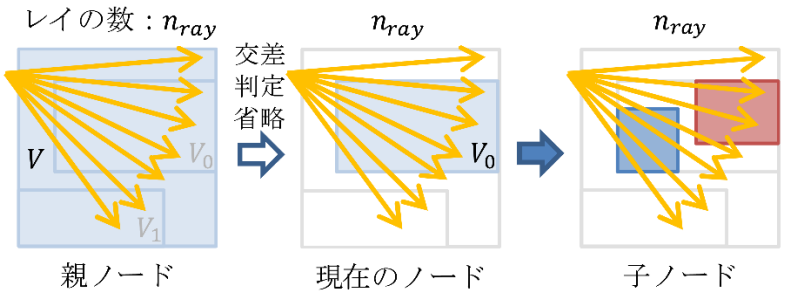


図 3.6　レイ集合の分割を省略した場合．レイ集合の分割を省略するため，レイ集合の数はn_{ray}のまま，さらなる問題の分割が行われる．

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．

$$C_{int} = n_{ray}C_{bv} + \alpha n_{ray}C_{child}n_{child} \tag{3.3}$$

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．

$$C_{skip} = n_{ray}C_{child}n_{child} \tag{3.4}$$

式(3.3)，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わされる．

$$\alpha > 1 - \frac{C_{bv}}{n_{child}C_{child}} \tag{3.5}$$

交差割合αが式(3.5）の条件を満たす場合は，レイと BV の交差判定を行わずに，全てのレイが交差しているとみなす．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，非効率なレイ集合の分割を避けることがで

きる．なお，現在のノードが内部ノードの場合は，子ノードとレイとの交差判定コストC_{child}はC_{bv}となり，2 分木の BVH の場合n_{child}は 2 となるので，式(3.5)は以下の式に簡略化される．

$$\alpha > 0.5. \tag{3.6}$$

3.5 実験結果

提案法の実験結果を図 3.7 から図 3.10 に示す. 実行環境は, CPU が Intel Core i7 2.67GHz, メモリ 6.0GB RAM の PC で，1 スレッドのみで実行している．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．なお，本実験の対象シーンでは，全体の計算時間の内，すべてのレイが最近傍の三角形を計算するまでの時間はおおよそ 8 割を占めている．比較対象は Afra の手法[1]であり，この手法と同様，三角形集合，レイ集合が十分小さいと見なす閾値であるδ_T，δ_Rはそれぞれ 8 に設定し，三角形集合の分割における bin の数Kは 32 に設定している．先述したように，提案法のレイサンプリングは計算コストがかかるため，提案法ではレイの数が 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う．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．後の実験結果で示すように，サンプリング数やサンプリング方法を変えて比較を行ったが，このパターンが最も良好な結果が得られている．三角形集合の分割のコストを抑えるために，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．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，レイと三角形の交差判定アルゴリズムは一般的に用いられている方法[32]を SIMD 化したものを用いている．

図 3.7 は Sibenik シーンにおける解像度別の速度比較の結果である．このシーンでは，モデルの床に鏡面反射材質を設定しており，図 3.7(a)(b)は点光源，図 3.7(c)は面光源を設置している．図 3.7 より提案法は，解像度が大きいほど Afra の手法に対する高速化率が大きくなっており，解像度$4{,}096^2$では平均 1.85 倍，最大 2 倍程度の高速化を達成している．図 3.7(a)の解像度$4{,}096^2$の結果において，本研究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の影響を調べるため，それぞれ 1 つだけを有効にして実験を行ったところ，それぞれの高速化率とは 1.24，1.05，1.28 となり，それぞれの計算時間削減への寄与率は 42%，10%，48%となった．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 3.8(a)は提案法が非効率となった場合の例である．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．これは，視点と点光源がモデルの上空にあり，レイ集合分割の省略が行われる頻度が少なくなるためと思われる．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うことができている．図 3.8(b)に環境遮蔽，図 3.8(c)に San Miguel シーンでの 3 回反射までのパストレーシングの結果を示す．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り，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光の影響を計算する手法である．図 3.9 に Conference シーン，図 3.10 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(3.1)を精度よく近似できてい

るためと思われる．

図 3.11 に解像度別の高速化率，反射回数別の高速化率を示す．図 3.11(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ー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．また，図 3.11(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．

図 3.12 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，レイの数によらず一定間隔で 100 本サンプリング，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，レイの数によらず一定間隔に 100 本サンプリングした場合がわずかによい結果が得られており，提案法では，レイの数によらず一定間隔で 100 本サンプリングする方法を用いている．

図 3.13 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 3.13 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．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 3.7(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．

3.6 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，高速化データ構造の探索における計算効率を定式化することで，レイ集合を効率的に分割できない場合の交差判定を避ける方法を提案した．先行研究との比較実験では，低解像度画像において稀に提案法の方が，効率がわずかに低くなる場合があったが，ほぼすべてのシーンで提案法の方が効率的にレイトレーシングを行うことができた．また，画像解像度が高いほど高速化率が向上し，本実験における最大解像度4.096^2において，最大約 2 倍の高速化を実現した．

今後の課題としては，マルチスレッドによる並列処理に適したアルゴリズムの開発が挙げられる．従来のレイトレーシングでは，ピクセル毎に独立してレイを追跡するため，並列化が容易であったが，DACRT では全ピクセルのレイを同時に追跡するため，並列化が困難である．単純な並列化方法として，レイ集合をスレッドの数だけ分割し，各スレッドで独立に DACRT を実行する方法が考えられるが，この方法では，レイ集合の分割に対する計算コストは削減できるが，三角形集合の分割に対する計算コストの削減は期待できない．近年，CPU はメニーコア化の流れにあり，DACRT を実用化する上で並列処理に適したアルゴリズムの開発は必須である．

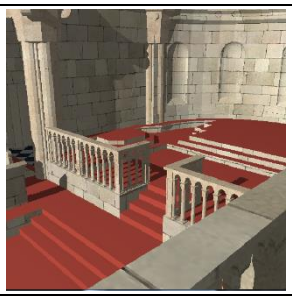
			
解像度	(a) 点光源	(b) 点光源	(c) 面光源
512^2	415ms/295ms (1.41)	281ms/183ms (1.54)	374ms/255ms (1.47)
$1{,}024^2$	1,520ms/1,003ms (1.52)	1,130ms/700ms (1.61)	1,430ms/860ms (1.66)
$2{,}048^2$	5,992ms/3,759ms (1.59)	4,656ms/2,647ms (1.76)	5,730ms/3,164ms (1.81)
$4{,}096^2$	27,300ms/14,700ms (1.86)	19,772ms/11,330ms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

図 3.7　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．

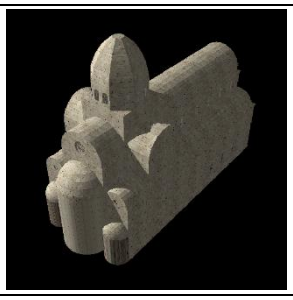
			
解像度	(a) 点光源	(b) 環境遮蔽	(c) パストレーシング
512^2	110ms/110ms (1.00)	446ms/365ms (1.22)	17,319ms/16,876ms (1.03)
$1{,}024^2$	331ms/334ms (0.99)	1,716ms/1,217ms (1.41)	27,725ms/26,435ms (1.05)
$2{,}048^2$	1,236ms/1,130ms (1.09)	6,948ms/4,359ms (1.59)	67,253ms/59,893ms (1.12)
$4{,}096^2$	5,075ms/4,219ms (1.20)	29,439ms/19,300ms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

図 3.8　Sibenik シーンと San Miguel 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．

			
解像度	(a) 点光源	(b) 点光源	(c) パストレーシング
512^2	234ms/189ms (1.24)	273ms/189ms (1.44)	1,208ms/1,071ms (1.13)
$1{,}024^2$	803ms/602ms (1.34)	1,030ms/670ms (1.54)	3,936ms/3,279ms (1.20)
$2{,}048^2$	3,113ms/2,182ms (1.43)	5,992ms/2,636ms (1.60)	15,533ms/12,316ms (1.26)
$4{,}096^2$	12,784ms/8,930ms (1.43)	17,987ms/10,618ms (1.69)	65,117ms/47,761ms (1.36)

図 3.9　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.

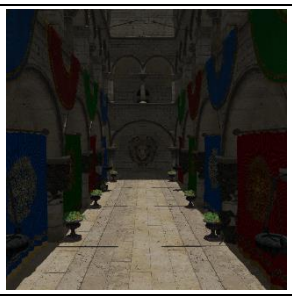
			
解像度	(a) 点光源	(b) 点光源	(c) パストレーシング
512^2	661ms/478ms (1.38)	599ms/448ms (1.52)	2,853ms/2,331ms (1.22)
$1{,}024^2$	2,298ms/1,428ms (1.61)	2,213ms/1,471ms (1.50)	8,288ms/7,066ms (1.17)
$2{,}048^2$	8,801ms/5,562ms (1.74)	8,774ms/5,363ms (1.64)	31,960ms/23,779ms (1.34)
$4{,}096^2$	35,397ms/19,657ms (1.80)	35,643ms/20,548ms (1.73)	136,327ms/98,228ms (1.39)

図 3.10　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.

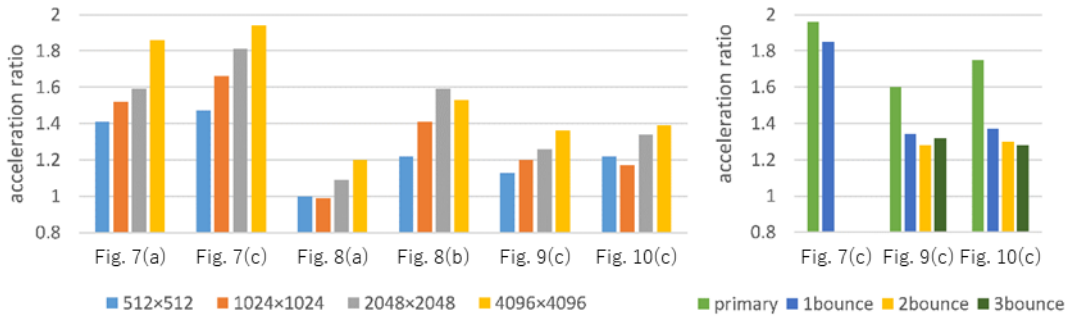


図 3.11　(a) 解像度別の高速化率．(b) 反射回数別の高速化率．

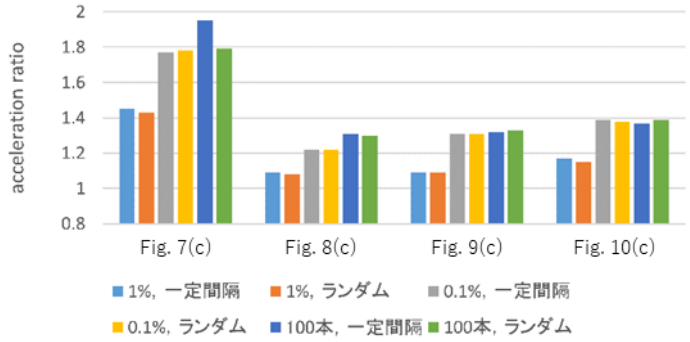


図 3.12　レイのサンプリング方法別の高速化率．

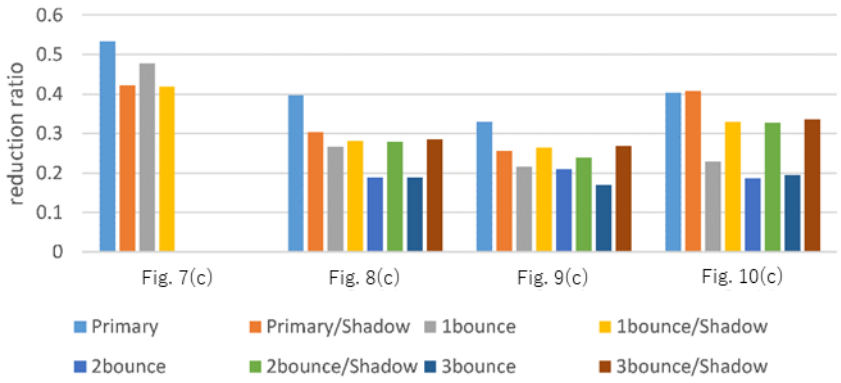


図 3.13　Afra の手法に対する反射回数別のレイとバウンディングボリュームの交差判定回数の削減率．

第４章　可視判定省略による高速化

4.1　はじめに

CG の分野において，写実的な画像の効率的な生成は重要な研究課題の一つである．写実的な画像は，映画，工業デザイン，建築設計，ゲームなど様々な分野で使用されている．写実的な画像を生成するには直接光だけでなく，間接光も考慮する必要があるため，計算コストが高く，画像生成に多くの時間を要する．

近年，効率的に写実的な画像を生成する手法の一つとして，多光源レンダリング法が研究されている[9, 29]．多光源レンダリング法は，あらかじめシーン中に大量の仮想的な点光源（Virtual Point Light, VPL）を設置しておき，出射輝度の計算点であるシェーディング点では，あらゆる方向から入射する光を VPL からの光で近似する．このとき，設置する VPL の数が多いほど近似精度が向上するが，計算時間は VPL 数に比例して増加する．そのため，少量の VPL をサンプリングし，すべての VPL を使用したときの出射輝度を推定する手法が一般的である[17, 58, 61]．推定結果には誤差が含まれるが，寄与の大きい VPL を重点的にサンプリングするか，サンプル数を増やすことにより誤差を低減することができる．

VPL の寄与を計算するには，シェーディング点と VPL 間の可視判定が必要となる．可視判定を行うには，シェーディング点から VPL に向かうレイとシーン中のすべての物体で交差判定を行う必要があり，最先端の高速化データ構造を使用しても非常に計算コストが高い処理である．そのため，VPL の寄与計算において可視判定の計算時間が支配的となっており，同時間でより多く VPL をサンプリングするには，可視判定の高速化が重要である．

本研究では，VPL の寄与計算において可視判定を確率的に行うことで，可視判定回数を削減し効率的に写実的な画像を生成する手法を提案する．提案法はロシアンルーレット法に基づく手法で，ある確率で可視判定を省略し，それ以外では実際に可視判定を行う．ただ

し，従来のロシアンルーレット法と異なり，提案法では関数評価の省略時の推定値を 0 に制限しない．これにより，関数評価を省略する確率が同じとき，従来のロシアンルーレット法よりも分散の増加を抑えることが可能となる．提案法では可視判定を省略する確率（以降，省略確率と呼ぶ）を大きくするほど，可視判定回数を削減でき，同時間においてより多くのサンプルを計算できる．しかしながら，可視判定を確率的に行う場合，省略確率を高くするほど可視判定結果に対する分散が増加し，生成される画像にノイズが生じるという問題がある．そこで，確率的評価による分散の増加量とサンプル数の関係を定式化し，与えられた時間内で分散を最小とする省略確率の導出を行う．本研究により，正確に可視判定を行った場合と比較して，同時間レンダリングにおいて，よりノイズの少ない画像を生成することができる．実験では，同画質の画像を最大 1.55 倍高速に生成することができ，可視判定を確率的に行う先行研究[51]と比較しても最大 1.3 倍の高速化を達成した．

4.2 関連研究

多光源レンダリング法による輝度計算において，少量の VPL をサンプリングし，輝度を推定する手法が多く提案されている．VPL の寄与は，VPL の放射強度，シェーディング点における BRDF，幾何項，可視関数の積で計算される．これら 4 つの項の積に比例して VPL のサンプリングを行うことで，分散の小さい高精度な推定を行うことができる．Wang ら[58]は，BRDF の重点的サンプリングを利用することで，放射強度・BRDF・幾何項の積に比例したサンプリング方法を提案した．Georgiev ら[17]は，VPL の寄与を記録するキャッシュ点を使用することで，単純なシーンにおいて，VPL の寄与に比例したサンプリングを行う方法を提案した．Wu ら[61]は，Wang らの手法を改良し，VPL とシェーディング点をクラスタリングすることで，複雑なシーンにおいても，VPL の寄与に比例したサンプリング方法を提案した．これらの手法により，少ないサンプルでも高精度な輝度推定が可能とな

ったが，可視判定の計算コストが高いシーンでは，依然として画像生成に時間がかかってしまう．特に Georgiev らの手法[17]と Wu らの手法[61]は，可視関数を考慮した VPL のサンプリングを行っているため，サンプリングされた VPL とシェーディング点間は互いに可視である確率が高く，常に可視関数を評価するのは非効率である．

計算コストが高い可視判定の効率化を行う手法も多く提案されている．Popov ら[39]は，経路空間を量子化し，可視判定結果を再利用することで，可視判定回数を削減する手法を提案した．また Ritschel ら[43]は，近似的に可視判定を高速に行うためのデータ構造であるシャドウマップ（光源からの距離を記録した画像）を効率的に生成する方法を提案し，Ulbrich ら[49]は，結果に与える影響が大きいシャドウマップを重点的に生成する方法を提案した．これらの手法では，提案法のアプローチと異なり決定論的に可視関数の近似を行い，常に可視判定を省略するため，非常に高速に画像を生成することができる．しかしながら，これらの手法では生成される画像に，経路空間の量子化やシャドウマップによる誤差が含まれるという問題がある．Billen ら[3,4]は，可視判定の交差判定対象を確率的に選択することで，交差判定回数を削減する手法を提案した．しかしながら，同時間レンダリングにおいて分散が増加する結果となっており実用的ではない．Veach[51]は，画像生成の効率を考慮したロシアンルーレット法により，可視判定回数を削減する手法を提案した．しかしながら，ロシアンルーレット法の関数評価省略時の推定値を 0 とする制限により，寄与が小さい VPL に対してしか可視判定の省略確率が高くできず，大幅な高速化は期待できない．

提案法は，Wu らの手法を基に，可視関数を確率的に評価することで，さらなる効率化を行うものである．可視関数の確率的評価は，Veach の提案した画像生成の効率を考慮したロシアンルーレット法を基にしており，可視関数の値に対する事前情報を用いることで，さらなる効率化を行う．Veach の手法では，寄与が小さい VPL に対して可視判定を省略する確率が高くなるのに対し，提案法では，寄与が小さい VPL だけでなく，高確率で可視または

不可視の VPL に対しても可視判定を省略する確率を高く設定することができる.

4.3 VPL サンプリングによる輝度推定

多光源レンダリング法では，あらゆる方向から入射する光を，VPL からの光で近似した以下の式で，シェーディング点xから視点x_v方向に出射する放射輝度Lを計算する.

$$L(x, x_v) = \sum_{i=1}^{N_{vpl}} I(y_i) f_r(y_i, x, x_v) G(y_i, x) V(y_i, x) \tag{4.1}$$

ここで，N_{vpl}は VPL の数，y_iはi番目の VPL，Iは放射強度，f_rは BRDF，Gは幾何項，Vは可視関数である. 十分な近似精度を得るには, 大量の VPL が必要であるが, 計算時間は VPL の数に比例して増加する. そこで, 少量の VPL をサンプリングし, 式(4.1)の値を推定する.

$$\begin{aligned} L(x, x_v) &\approx \hat{L}(x, x_v) \\ &= \frac{1}{N} \sum_{s=1}^{N} \frac{I(y_s) f_r(y_s, x, x_v) G(y_s, x) V(y_s, x)}{p(y_s|x)} \end{aligned} \tag{4.2}$$

ここで，Nはサンプリング回数，y_sはs回目の試行における VPL サンプルであり，確率質量関数pに従いサンプリングされる．寄与がある VPL に対して$p(y_i|x) > 0$であれば，使用する確率質量関数は任意に選択でき, 式(4.2)の期待値$\mathbb{E}\left[\hat{L}(x, x_v)\right]$は式(4.1)と一致する. ただし, 推定値の分散は確率質量関数に大きく依存し，以下の式で表される.

$$\begin{aligned} var\left[\hat{L}(x, x_v)\right] &= \frac{1}{N}\left\{\mathbb{E}\left[\hat{L}(x, x_v)^2\right] - \mathbb{E}\left[\hat{L}(x, x_v)\right]^2\right\} \\ &= \frac{1}{N}\left\{\sum_{i=1}^{N_{vpl}} \frac{I(y_i)^2 f_r(y_i, x, x_v)^2 G(y_i, x)^2 V(y_i, x)^2}{p(y_i|x)} - L(x, x_v)^2\right\} \end{aligned} \tag{4.3}$$

上式より，確率質量関数pが$I \cdot f_r \cdot G \cdot V / L$と類似するほど，つまり，VPL の寄与に比例するほ

ど，分散を低減することができる．また，サンプル数Nを増やすことによっても分散を低減できることが分かる．そのため，サンプリングを効率的に行うことができれば，同時間のレンダリングにおいて，より多くのサンプルをとることができ，高精度な推定が可能となる．

本研究では，$I(y_i)f_r(y_i,x,x_v)G(y_i,x)$を VPL y_iの暫定的な寄与t_iと定義する．また，表記簡略化のため$V(y_i,x)$，$p(y_i|x)$を，それぞれv_i，p_iと表記する．

4.4 提案法

提案法の処理の流れを図 4.1 に示す．基本的な処理の流れは，Wu らの手法[61]と同様であり，最初に VPL とシェーディング点の生成を行い，それぞれをクラスタリングする（図 4.1(a)）．次に，各シェーディング点クラスタで各 VPL クラスタに対して可視関数の平均値を推定する（図 4.1(b)）．平均値の推定は，各クラスタの要素を数個サンプリングし可視関数を評価することで行う．各シェーディング点で出射輝度を計算する際は，推定した可視関数の平均値を用いて VPL クラスタに対する確率分布を構築する（図 4.1(c)）．VPL のサンプリングは，VPL クラスタをサンプリングし，そのクラスタから VPL を一様にサンプリングすることで行う．このサンプリング方法では，可視の VPL を多く含む VPL クラスタがサンプリングされやすくなるため，そのクラスタからサンプリングされる VPL は可視である確率が高い．そして提案法では，そのような VPL に対して可視関数の評価を高確率で省略するため，同時間でより多くのサンプルを使用することができ，分散の小さい推定が可能である．クラスタリング及び確率分布の構築の詳細は[61]を参照されたい．

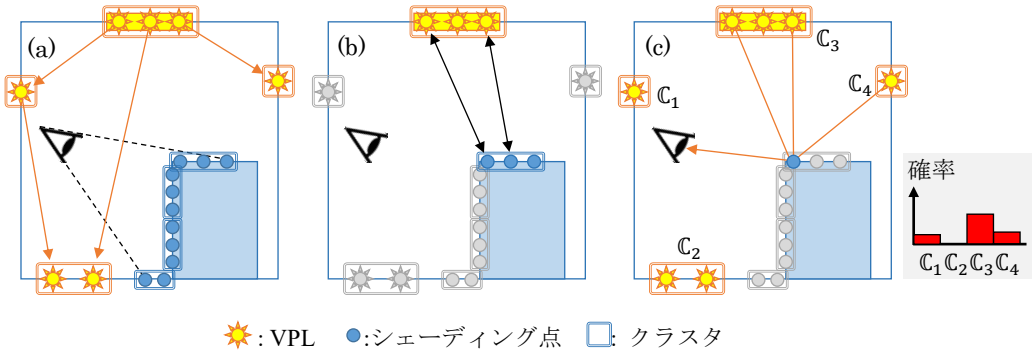


図 4.1　提案法の処理の流れ．(a) VPL とシェーディング点の生成しクラスタリング．(b) 各シェーディング点クラスタと各 VPL クラスタで要素を数個サンプリングしクラスタ間の可視関数の平均値を推定．(c) 推定した可視関数の平均値を基に， VPL クラスタに対する確率分布を構築し，視点に出射する輝度を推定．

4.4.1　可視関数の確率的評価

可視関数は 2 点間の可視性を表す関数で，シェーディング点xと VPL y_i間の可視関数v_iは，xとy_iが互いに可視の場合 1 を返し，それ以外は 0 を返す．可視関数の評価は計算コストが高く，VPL の寄与計算において可視関数評価の計算時間が支配的である．そこで提案法では，以下の式を用いて可視関数の値を推定する．

$$\tilde{V}(y_i, x) = \begin{cases} \dfrac{\alpha_i}{q_i} & \text{確率} q_i \text{で選択} \\ \dfrac{v_i - \alpha_i}{1 - q_i} & \text{それ以外} \end{cases} \tag{4.4}$$

ここで，q_iはシェーディング点xにおける VPL y_iの省略確率，α_iはパラメータで任意の実数をとることができる．以下の式に示すように，提案法の可視関数の推定値の期待値$\mathbb{E}[\tilde{V}(y_i, x)]$は真値$v_i$と一致する．

$$\mathbb{E}[\tilde{V}(y_i, x)] = \frac{\alpha_i}{q_i} q_i + \frac{v_i - \alpha_i}{1 - q_i}(1 - q_i) = v_i \tag{4.5}$$

提案法の可視関数の推定式では，省略確率q_iで可視関数の評価を省略し，α_i / q_iを推定値とす

る．そして，確率$(1-q_i)$で実際に可視関数を評価し，v_iに 0 か 1 の結果を代入した値を推定値とする．そのため，省略確率q_iを大きくするほど，計算コストが高い可視関数の評価を省略でき，同時間においてより多くのサンプルを計算することができる．

提案法では，式(4.2)の可視関数Vを推定式 $\tilde{V}$で置き換え，s回目の試行における VPL サンプルy_sの暫定的な寄与t_sと確率質量関数の値p_sを用いて以下の式で輝度計算を行う．

$$\tilde{L}(x, x_v) = \frac{1}{N}\sum_{s=1}^{N}\frac{t_s\tilde{V}(y_s, x)}{p_s} \tag{4.6}$$

可視関数の推定値の期待値が真値と一致するため，輝度の推定値の期待値も真値と一致する．

パラメータα_iの計算

提案法では，可視関数を確率的に評価するため，可視関数の推定値には誤差が含まれる．そこで，パラメータα_iには誤差が最小となる値を使用する．提案法の可視関数の推定値の分散は以下の式で表される．

$$\begin{aligned} var[\tilde{V}(y_i, x)] &= \mathbb{E}[\tilde{V}(y_i, x)^2] - \mathbb{E}[\tilde{V}(y_i, x)]^2 \\ &= \left\{\frac{\alpha_i^2}{q_i^2}q_i + \frac{(v_i-\alpha_i)^2}{(1-q_i)^2}(1-q_i)\right\} - v_i^2 \\ &= \frac{(v_iq_i-\alpha_i)^2}{q_i(1-q_i)} \end{aligned} \tag{4.7}$$

上式より，$\alpha_i = v_iq_i$のとき，確率的評価による分散は 0 となる．しかしながら，可視関数の値v_iは未知であるため可視関数の近似値が必要となる．提案法では，シェーディング点xが所属するシェーディング点クラスタと，VPL y_iが所属する VPL クラスタ間の可視関数の平均値をv_iの近似値r_iとして用い，$\alpha_i = r_iq_i$とする．式(4.4)，式(4.7)に$\alpha_i = r_iq_i$を代入した式は以下の通りである．

$$\tilde{V}(y_i, x) = \begin{cases} r_i & \text{確率} q_i \text{で選択} \\ \dfrac{v_i - r_i q_i}{1 - q_i} & \text{それ以外} \end{cases} \tag{4.8}$$

$$var[\tilde{V}(y_i, x)] = \frac{q_i (v_i - r_i)^2}{1 - q_i} \tag{4.9}$$

省略確率q_iの計算

可視関数を確率的に評価した場合，確率的評価による分散の増加量と，1 サンプル当たりの計算時間はトレードオフの関係にある．そこで推定値の分散はサンプル数Nに反比例して減少するという関係から，与えられた時間内で分散を最小とする省略確率q_iを導出する．

1 サンプルの場合において，提案法の可視関数の推定式(4.8)を用いたときの輝度の推定値の分散は以下の式で計算される（導出の詳細は付録 A を参照）．

$$var\left[\tilde{L}^{(1)}(x, x_v)\right] = var\left[\hat{L}^{(1)}(x, x_v)\right] + \sum_{j=1}^{N_{vpl}} \frac{t_j^2}{p_j^2} \frac{q_j \left(v_j - r_j\right)^2}{1 - q_j} p_j \tag{4.10}$$

ここで，$var\left[\hat{L}^{(1)}(x, x_v)\right]$はサンプル数が 1 の場合の式(4.3)である．また，第 2 項は可視関数の確率的評価による分散の増加量を表しており，可視関数を確率的に評価する場合，1 サンプルの場合では必ず分散が増加する．

次に，時間Tが与えられた下で計算できるサンプル数Nについて考える．まず，シェーディング点の生成にかかる時間をt_xとすると，VPL サンプリングに割ける時間は$T - t_x$となる．そして，VPL y_jの寄与計算にかかる時間は，可視関数を評価した場合と省略した場合の VPL の寄与計算時間をそれぞれt_{skip}とt_{eval}とすると，平均で$t_{skip} q_j + t_{eval}\left(1 - q_j\right)$となる．また，VPL によって可視関数評価の省略確率は異なるため，最終的な VPL の寄与計算時間は，各 VPL の平均寄与計算時間をさらに平均したものとなる．よって，時間Tが与えられた下で計算できるサンプル数Nは以下の式で表される．

$$N = \frac{T - t_x}{\sum_{j=1}^{N_{vpl}} \{t_{skip} q_j + t_{eval}(1 - q_j)\} p_j} = \frac{T - t_x}{t_{eval} - t_{diff} \sum_{j=1}^{N_{vpl}} q_j p_j} \tag{4.11}$$

ここで，$t_{diff} = t_{eval} - t_{skip}$とおき，$\sum_{j=1}^{N_{vpl}} p_j = 1$を用いた．分散はサンプル数$N$に反比例して減少することから，式(4.10)，式(4.11)より，Nサンプルの場合において，輝度の推定値の分散は以下の式で表される．

$$\left\{ var\left[\hat{L}^{(1)}(x, x_v)\right] + \sum_{j=1}^{N_{vpl}} \frac{t_j^2}{p_j^2} \frac{q_j (v_j - r_j)^2}{1 - q_j} p_j \right\} \left\{ \frac{t_{eval} - t_{diff} \sum_{j=1}^{N_{vpl}} q_j p_j}{T - t_x} \right\} \tag{4.12}$$

なお，分散が最小となる省略確率q_iの計算において定数は影響しないため，以降定数$(T - t_x)^{-1}$は省略する．VPL y_iに対する可視関数評価の省略確率を計算するために，式(4.12)をq_iについて微分し，結果を 0 とおくと以下の式が得られる．

$$\frac{t_i^2}{p_i^2} \frac{(v_i - r_i)^2}{(1 - q_i)^2} \left\{ \frac{t_{eval}}{t_{diff}} - \sum_{j=1}^{N_{vpl}} q_j p_j \right\} - \left\{ var\left[\hat{L}^{(1)}(x, x_v)\right] + \sum_{j=1}^{N_{vpl}} \frac{t_j^2}{p_j^2} \frac{q_j (v_j - r_j)^2}{1 - q_j} p_j \right\} = 0 \tag{4.13}$$

上式を解けば，画像生成の効率を最大化する可視関数の省略確率q_iが得られるが，2 つの中括弧内の値の計算には他の VPL に対する省略確率が含まれるため，この式を解くのは困難である．そこで提案法では，Veach の手法[51]と同様，現在の計算対象であるシェーディング点より以前に計算した近傍のシェーディング点の輝度計算情報を使用し，各中括弧内の値をそれぞれ定数$\hat{t}$，$\hat{\sigma}^2$で近似する（近似方法は 4.4.2 節で述べる）．

$$\frac{t_i^2}{p_i^2} \frac{(v_i - r_i)^2}{(1 - q_i)^2} \hat{t} - \hat{\sigma}^2 = 0 \tag{4.14}$$

上式をq_iについて解くことで，与えられた時間内で推定値の分散を最小とする可視関数評価

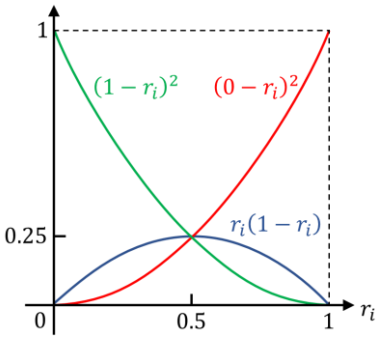


図 4.2　$(v_i - r_i)^2$と$r_i(1-r_i)$のグラフ．緑線は$v_i = 1$，赤線は$v_i = 0$の場合の$(v_i - r_i)^2$のグラフである．r_iとv_iが近いとき，$r_i(1-r_i)$は$(v_i - r_i)^2$のよい近似となる．

の省略確率が得られる．

$$q_i = 1 - \sqrt{\frac{t_i^2 (v_i - r_i)^2}{p_i^2} \frac{\hat{t}}{\hat{\sigma}^2}} \tag{4.15}$$

上式では，パラメータα_iの計算時と同様，未知の可視関数v_iが含まれるため何らかの近似が必要となる．提案法では，$(v_i - r_i)^2$を平均値で近似する（図 4.2 参照）．提案法では可視関数の予測値r_iとしてクラスタ間の可視関数の平均値を用いるため，$(v_i - r_i)^2$の平均値はクラスタ間の可視関数の分散であり，$r_i(1-r_i)$で計算される．この近似では，r_iが 1 か 0 に近いとき，$r_i(1-r_i)$が小さくなるため省略確率が大きくなり，r_iが 0.5 に近いとき，$r_i(1-r_i)$が大きくなるため省略確率が小さくなる．つまりこの近似方法では，可視か不可視である確率が高いとき（r_iが 1 か 0 に近い），高確率で可視判定を省略すべきであり，可視か不可視かが曖昧なとき（r_iが 0.5 に近い），可視判定を正確に行うべきという直感通りの特性を持つ．4.5 節で示す通り，その他の近似方法との比較実験を行ったが，この方法が最も良好な結果が得られた．なお，式(4.15)で可視関数評価の省略確率q_iを計算すると，q_iが負になる場合があるが，その場合は$q_i = 0$とすればよい．

4.4.2 パラメータ$\hat{t}$，$\hat{\sigma}^2$の計算

提案法の可視関数の確率的評価法において，VPL y_iに対する可視関数の省略確率の計算の際，他の VPL に依存する項$\hat{t}$，$\hat{\sigma}^2$の計算が必要となる．そこで提案法では Veach の手法[51]と同様，現在の計算対象のシェーディング点と，以前に計算した近傍のシェーディング点で VPL の寄与が相関していると仮定し，以前に計算した近傍のシェーディング点における輝度計算情報を用いてモンテカルロ法によりこれらの値を推定する．

$$\hat{t} = \frac{t_{eval}}{t_{diff}} - \frac{1}{N}\sum_{k=1}^{N} q_k \tag{4.16}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{t_k \tilde{V}(y_k, x)}{p_k}\right)^2 - \left(\frac{1}{N}\sum_{k=1}^{N}\frac{t_k \tilde{V}(y_k, x)}{p_k}\right)^2 \tag{4.17}$$

ここで，y_kは以前に計算した近傍のシェーディング点におけるk番目の VPL サンプルであり，q_k，t_k，p_kはそれぞれ VPL サンプルy_kに対する可視関数評価の省略確率，暫定的な寄与，確率質量関数の値である．$\hat{\sigma}^2$は，可視関数の確率的評価を用いた際の推定値の分散（式(4.10)）を表すことから，式(4.17)で表される標本分散で近似している．なお，最初に計算するシェーディング点は，可視関数評価の省略確率を 0 とした．

提案法では$\hat{t}$，$\hat{\sigma}^2$を以前に計算した近傍のシェーディング点の情報を用いて近似するため，良い近似を行うには以前に計算した近傍のシェーディング点と現在のシェーディング点で VPL からの寄与が相関している必要があり，シェーディング点の計算順序が重要となる．シェーディング点クラスタ内のシェーディング点は，VPL からの寄与が相関していることが期待されるため，本研究では，シェーディング点クラスタを 1 つのレンダリング単位として，クラスタ内の全シェーディング点で輝度計算が終了してから，次のクラスタを処理する方法を採用した．

4.4.3　バイアスの回避

提案法では，可視関数評価の省略確率の計算式（式(4.15)）中の$(v_i - r_i)^2$の項を平均値$r_i(1-r_i)$で近似する．そのため，r_iが 0 か 1 のとき，可視関数評価の省略確率は 1 となり，実際の可視関数の値v_iと近似値r_iが異なる場合，推定値は真値に収束しない．そのため提案法では，$(v_i - r_i)^2$を$\max\left(\epsilon, r_i(1-r_i)\right)$で置き換える．これにより，バイアスを回避できるだけでなく，高確率で可視，不可視の VPL でも，寄与が非常に大きい場合は可視関数評価を省略する確率が小さくなりロバスト性を向上させることができる．様々な値で実験した結果，下限値ϵは 0.1 程度に設定すれば良好な結果が得られた．

4.4.4　負の可視関数推定値

通常の可視関数の取りうる値は 0 か 1 だけであるため，輝度の推定値は必ず 0 以上となる．一方，提案法では可視関数の値を推定し，可視関数の推定値は 0 か 1 だけではなく，あらゆる値を取りうるため，負の値となることもある．VPL サンプル数が十分であれば，最終的な輝度の推定値は真値に収束するが，サンプル数が十分でない場合，輝度の推定値が負となることがある．そのため，ディスプレイに結果を出力する際は，負のピクセル値を 0 でクランプするといった，ピクセル値が負となる可能性を考慮したトーンマッピングが必要である．

4.5　実験結果

本節では，可視関数の確率的評価の有無による比較，先行研究である Veach の手法[51]との比較により提案法の有効性を検証する．なお，実験は Intel Core i9-7980XE CPU，32GB メモリを搭載した PC を使用し，マルチスレッドによる並列化を行っている．レイとシーン

の交差判定に使用する高速化データ構造には, SIMD 命令に最適化された 4 分木の BVH[10]を使用している．提案法の可視関数の推定式のパラメータであるt_{eval}とt_{skip}は，可視関数評価の省略確率を 0 とした場合，1 とした場合で低解像度の画像を生成し，その計算時間から設定した．また，他手法との比較の際に用いる誤差の指標には RMSE (Root Mean Square Error)を使用する．提案法と Veach の手法の主な違いは，Veach の手法では常に可視関数の近似値r_iを 0 とし，式(4.15)における$(v_i - r_i)^2$を最大値の 1 で近似している点である．本実験におけるレンダリング画像の解像度は$1{,}280 \times 720$である．

同時間レンダリングでの誤差の比較結果を図 4.3 から図 4.5 に示す. 各シーンの情報は表 4.1 に示すとおりである．なお，表 4.1 の Visibility Cluster の欄は，各手法で共通の処理である VPL・シェーディング点の生成とクラスタリング，確率分布の構築にかかった時間であり，サンプル数に依存しない計算時間である．実験結果より，提案法が最も誤差を低減できていることが分かる．特に，図 4.4 は光沢材質を含むシーン，図 4.5 はジオメトリが複雑であり可視関数の相関が小さいシーンであるが，このようなシーンでも提案法が最も誤差を低減できている．

次に図 4.3 から図 4.5 の各シーンにおいて，提案法と Veach の手法で可視関数の確率的評価を行わない場合と同じ誤差の画像を生成するのに必要な時間を計測した結果を表 4.2 に示す．なお各行の上段と下段は，それぞれ Visibility Cluster の時間を含めた場合と含めない場合である．ノイズを視認できないほど小さくするには, より計算時間が必要となるが, Visibility Cluster の時間は表 4.1 に示す時間で固定であり，無視できるほど小さい割合となるため，下段に示す高速化率（確率的評価なしの計算時間/各手法の計算時間）が実質のものである．結果より，Veach の手法では最大でも 1.26 倍の高速化率であるのに対して，提案法では 1.41 倍から 1.55 倍の高速化率が得られており，提案法の優位性が分かる．図 4.5 のシーンは可視判定の計算コストが高いにもかかわらず，高速化率が小さいのは，ジオ

メトリが複雑であり正確な可視関数の近似ができてないためと思われる．

提案法と Veach の手法で，可視関数評価の省略確率の平均値を可視化した画像を図 4.6 に示す. 結果より, 提案法が Veach の手法より高い省略確率が得られていることが分かる．特に遮蔽物の少ない領域であっても，Veach の手法では省略確率が低いままであるのに対し，提案法では可視関数の事前情報を用いることで高い省略確率が得られている．

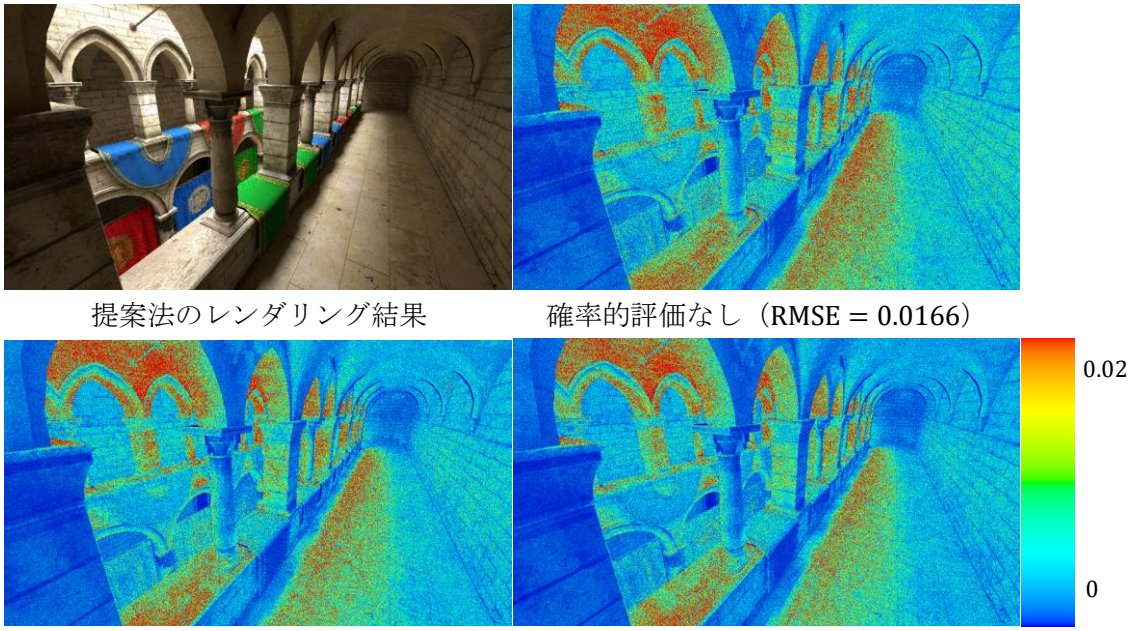
可視関数評価の省略確率の計算式（式(4.15)）において行った近似について検証する．ここでの比較対象は Veach の手法が採ったアプローチである保守的な近似とする．提案法の輝度推定値の分散式（式(4.10)）より，可視関数評価の省略確率が 1 に近づくほど可視関数の確率的評価による分散が急激に増加する．そのため，式(4.15)において$(v_i - r_i)^2$を最大値で置き換え，省略確率が小さくなるようにしたものが保守的な近似である．可視関数の値v_iは 0 か 1 であることから，$(v_i - r_i)^2$の最大値は$\max((1 - r_i)^2, (0 - r_i)^2)$となる．図 4.3 から図 4.5 の各シーンで 60 秒かけてレンダリングしたときの RMSE と可視関数の平均省略確率を表 4.3 に示す．結果より，保守的な近似より提案法の近似の方が，より多くの可視判定を省略しており，RMSE を低減できている．図 4.3 から図 4.6 の Veach の手法の結果と保守的な近似の結果を比較した場合，両者は同程度の RMSE，平均省略確率となった．これは保守的な近似では, 本来可視判定を省略するべきである可視関数の予測値r_iが 0 か 1 に近い VPL ほど$(v_i - r_i)^2$の項が大きい値に置き換えられ省略確率が小さくなってしまうためで, 保守的な近似では可視関数の事前情報使用の恩恵が受けられていないことが分かる．

最後に，可視判定の計算コストが高いシーンほど可視関数の確率的評価が効率的であることを明確に示すために，図 4.3 のシーンにおいて，レイとシーンの交差判定の高速化データ構造に単純な 2 分木の BVH[53]を用いた場合との比較を行う．2 分木の BVH を用いた場合，t_{skip}/t_{eval}は 0.07 である．先ほどの同誤差画像の生成時間の比較と同様に，確率的評価なしで 60 秒かけてレンダリングした結果と同じ誤差の画像を提案法で生成するのにかか

った時間を計測すると，提案法は 40.7 秒であり，高速化率は 1.4 倍から 1.47 倍に向上し，Visibility Cluster の時間を除けば，高速化率は 1.52 倍から 1.63 倍に向上した．このことから，提案法は可視判定の計算コストが高いシーンほど有用である．

4.6 制限事項

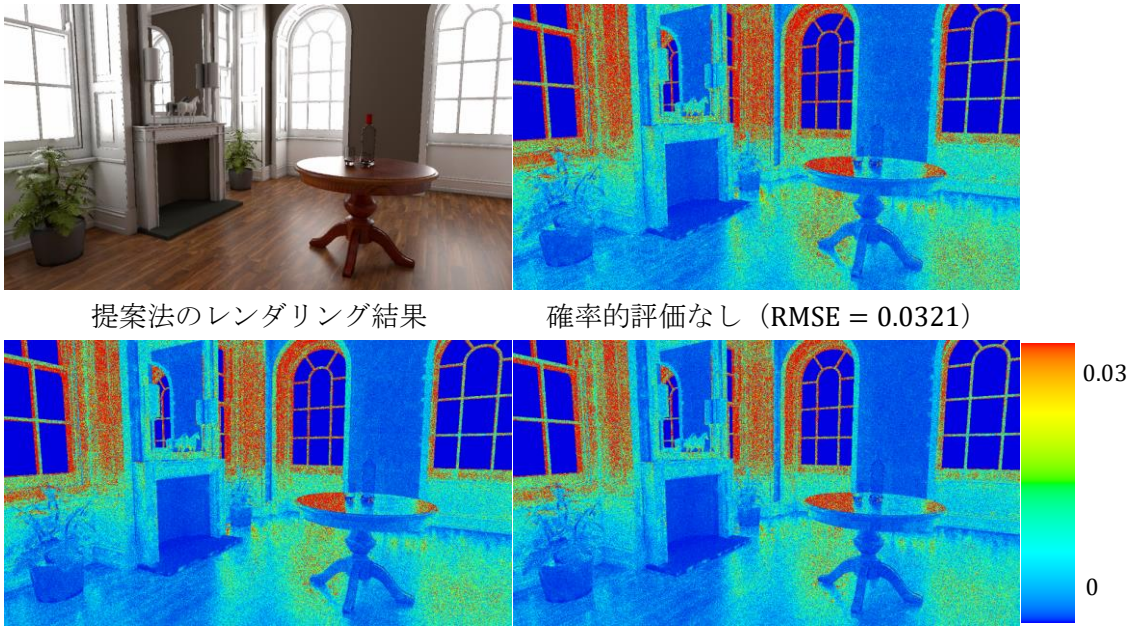
提案法は，可視関数の事前情報として Wu らの手法[61]で用いられる VPL クラスタとシェーディング点クラスタ間の可視関数の平均値を利用することで Veach の手法を改良した手法である．提案法の可視関数評価の省略確率は，事前情報の精度がある程度正確であることを仮定しているため，ジオメトリが複雑な領域など，この仮定を満たさない場合は，同時間において誤差が増加する可能性がある．また，クラスタ内の各シェーディング点は VPL からの寄与が相関していることも仮定しているため（4.4.2 節参照），材質の変化などにより，この仮定を満たさない場合も，同時間において誤差が増加する要因になり得る．本研究では，Wu らの手法[61]を用いてクラスタリングを行っており，この手法では可視関数の相関のみを考慮しているため，提案法に適したクラスタリング方法の考案が必要である．



提案法のレンダリング結果　　確率的評価なし（RMSE = 0.0166）

Veach の手法（RMSE = 0.0147）　　提案法（RMSE = 0.0130）

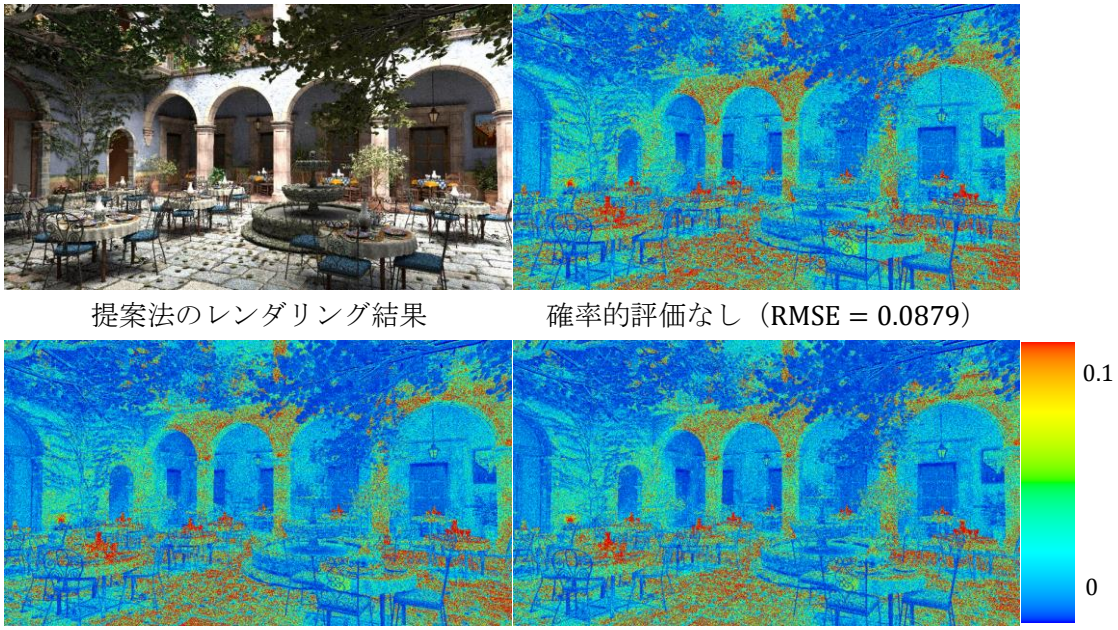
図 4.3　Sponza シーンでの提案法のレンダリング結果と誤差の可視化画像．誤差画像は同じ時間（60 秒）をかけてレンダリングしたときの結果.



提案法のレンダリング結果　　確率的評価なし（RMSE = 0.0321）

Veach の手法（RMSE = 0.0275）　　提案法（RMSE = 0.0239）

図 4.4　Fireplace Room シーンでの提案法のレンダリング結果と誤差の可視化画像．誤差画像は同じ時間（60 秒）をかけてレンダリングしたときの結果.



提案法のレンダリング結果　　確率的評価なし（RMSE = 0.0879）

Veach の手法（RMSE = 0.0787）　　提案法（RMSE = 0.0718）

図 4.5　San Miguel シーンでの提案法のレンダリング結果と誤差の可視化画像．誤差画像は同じ時間（60 秒）をかけてレンダリングしたときの結果．

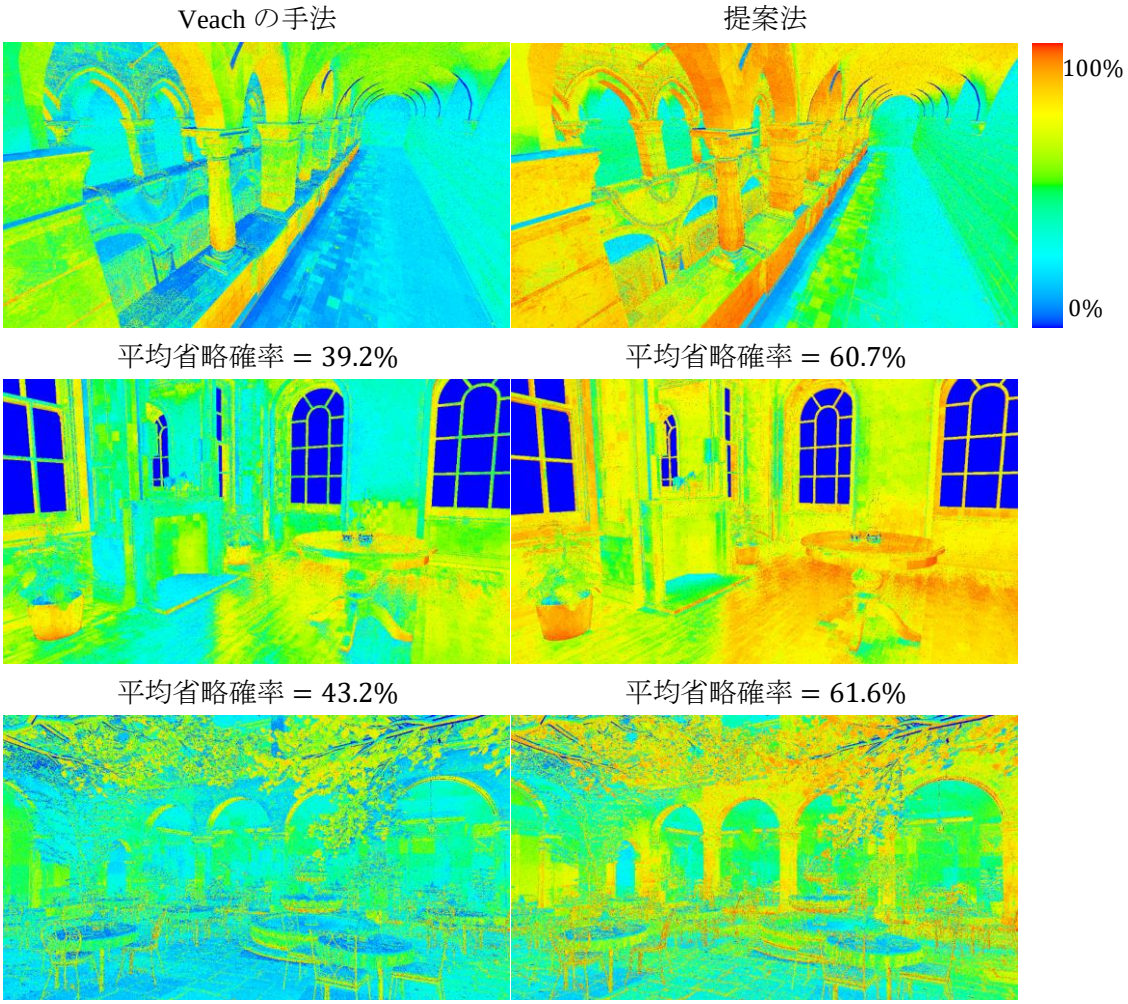


図 4.6　可視関数評価の省略確率の可視化.

表 4.1　シーン情報．t_{eval}とt_{skip}は可視関数を評価した場合と省略した場合の計算時間であり，t_{skip}/t_{eval}が小さいほど可視判定の計算コストが高いことを表す．また Visibility Cluster の欄は，提案法と Veach の手法で共通の処理に対する計算時間である．

	VPL 数	三角形数	Visibility Cluster	$\frac{t_{skip}}{t_{eval}}$
図 4.3	100K	262K	10 秒	0.15
図 4.4	200K	143K	12 秒	0.32
図 4.5	200K	10M	22 秒	0.05

表 4.2　同誤差画像の生成時間の比較．括弧内の数字は高速化率を表し，各行で上段と下段はそれぞれ Visibility Cluster の計算時間を含めた場合と除外した場合である．

	確率的評価なし	Veach の手法	提案法
図 4.3	60秒	53 秒（× 1.13）	43 秒（× 1.40）
	50 秒	43 秒（× 1.16）	33 秒（× 1.52）
図 4.4	60秒	50 秒（× 1.20）	43 秒（× 1.40）
	48 秒	38 秒（× 1.26）	31 秒（× 1.55）
図 4.5	60秒	53 秒（× 1.13）	49 秒（× 1.22）
	38 秒	31 秒（× 1.23）	27 秒（× 1.41）

表 4.3　近似方法による RMSE と可視関数評価の平均省略確率の比較．

	保守的な近似		提案法の近似	
	RMSE	平均省略確率	RMSE	平均省略確率
図 4.3	0.0145	39.6%	0.0130	60.7%
図 4.4	0.0263	45.6%	0.0239	61.6%
図 4.5	0.0770	40.5%	0.0718	55.3%

4.7 まとめ

本研究では，確率的な手法を用いて可視判定回数を削減することで，写実的な画像を効率的に生成する手法を提案した．提案法では，可視関数を確率的に評価するため，可視関数が新たな分散の要因となるが，可視関数評価を省略することによる計算コストの減少量と，可視関数の確率的評価による分散の増加量の関係を考慮した可視関数評価の省略確率を用いることで，与えられた時間内でより誤差の少ない画像を生成することができる．提案法の可視関数の確率的評価法を用いることで，従来の可視関数の確率的評価法と比較し，より効率的にレンダリングを行えることを示した．

今後の課題としては，提案法の適用範囲の拡大などが挙げられる．提案法の可視関数の確率的評価法を用いるには，可視関数に関する正確な事前情報が必要となる．多光源レンダリング法においては，シェーディング点が可視領域に制限されるため，比較的容易に可視関数に関する正確な事前情報を得ることができるが，その他の大域照明アルゴリズムでは，シェーディング点はシーン中のあらゆる領域に生成されるため，容易に可視関数に関する正確な事前情報を得ることはできない．そのため，提案法の適用範囲を拡大するためには，この問題を解決する必要がある．

第５章　誤差推定による高速化

5.1　はじめに

近年，写実的な画像を効率的に生成する手法として多光源レンダリング法がよく研究されている．多光源レンダリング法では，あらかじめ仮想的な点光源（Virtual Point Light, VPL）をシーン中に生成しておき，輝度を計算するシェーディング点では，あらゆる方向から入射する光を VPL からの光で近似して，視点方向に出射する輝度を計算する．十分な近似精度を得るには大量の VPL が必要となるが，輝度計算のコストは VPL 数に比例して増加するため，すべての VPL の寄与を計算するのはコストが高いという問題がある．この問題を解決するため，これまで様々な手法が提案されてきた[23, 24, 36]．これらの手法では，大量の VPL の中から少量の VPL をサンプリングし，すべての VPL を使用した場合の輝度を推定するため，生成する VPL の数に依らず高速に画像を生成することができる．しかしながら，VPL サンプル数などのパラメータを適切に設定しなければ，推定による誤差で結果画像にノイズやアーティファクトが発生してしまうという新たな問題が生じる．そのため，ノイズやアーティファクトのない画像が得られるまで，ユーザはパラメータ調整とレンダリングを繰り返す必要があり，多大な労力と時間を要する．推定値の誤差制御を試みる手法[7, 55, 56, 57]も提案されてはいるが，ヒューリスティックスに基づく手法であり誤差制御の精度は低い．

そこで本研究では，多光源レンダリング法において区間推定法を用いることで，推定値の誤差を制御可能かつ効率的な輝度計算方法を提案する．提案法は，VPL のクラスタリングを行うことで効率的に輝度計算を行う Lightcuts[55]に基づき，Lightcuts と同様，推定値の相対誤差の制御を試みる．これは刺激の弁別閾（差を知覚できる最小の差）は基礎刺激の強度に比例するというウェーバーの法則[6]に基づく．ユーザは許容相対誤差と，誤差制御

の信頼度パラメータを指定するだけでよく，VPL サンプル数などはユーザが指定したパラメータを基に自動で決定される．また本研究では，使用可能な材質モデルが単純なモデルに制限されるという Lightcuts の問題を解決する方法を提案する．さらに，可視関数値のキャッシングを行うことで，誤差制御の精度を低下させることなく，輝度計算を効率化する方法を提案する．提案法は正確に誤差制御を行えるため，一度レンダリングするだけで，ノイズやアーティファクトのない画像を生成でき，Lightcuts と同程度の高い計算効率で画像を生成できる．

5.2 関連研究

これまで，多光源レンダリング法による輝度計算を効率的に行う手法が多く提案されてきた[23, 24, 36]．これらの手法では，適切なパラメータ設定の下では非常に効率的に画像を生成できるが，不適切なパラメータ設定の下ではノイズやアーティファクトのある画像が生成されてしまうため，ユーザが所望する画質の結果を得るにはパラメータ調整とレンダリングを繰り返す必要があり，結局長い時間がかかってしまう．

Walter ら[56]は，誤差制御を試みる手法として Lightcuts を提案した．この手法では，ユーザが指定する許容相対誤差パラメータを基に VPL をクラスタリングし，各クラスタで 1 つの VPL をサンプリングし輝度推定を行う．この手法では，従来の手法と比較してパラメータ調整が容易であり，その後，シェーディング点のクラスタリングによるさらなる効率化や，被写界深度などの複雑な光学現象，多光源レンダリング法が苦手とする鋭い光沢反射材質を扱えるように拡張された[7, 55, 57]．しかしながら，先述したように Lightcuts では正確に誤差を制御することはできないため，Lightcuts を基にしたこれらの手法も，依然としてユーザが所望する画像を得るには試行錯誤が必要となる．

本研究では，Lightcuts を基に，区間推定法を用いて推定値の誤差を制御する．区間推定法を用いて誤差制御を行う手法は多く提案されているが[11, 20, 33, 40, 42, 46]，VPL のクラスタリングを用いた誤差推定に用いることはできない．

5.3 Lightcuts

提案法は，Walter らの提案した Lightcuts に基づくため，Lightcuts における輝度計算方法について簡単に説明する．多光源レンダリング法では，シェーディング点xから視点x_vに出射する輝度$L(x, x_v)$は以下の式で計算される．

$$L(x, x_v) = \sum_{i=1}^{N_{vpl}} I(y_i) f(y_i, x, x_v) G(y_i, x) V(y_i, x) \tag{5.1}$$

ここで，N_{vpl}は VPL 数，y_iはi番目の VPL，Iは放射強度，fは BRDF，Gは幾何項，Vは可視関数である．上式を用いて，写実的な画像を生成するには，大量の VPL が必要となり，すべての VPL の寄与$I \cdot f \cdot G \cdot V$を計算するのはコストが高い．そこで Lightcuts では，まず，VPL をN個のクラスタに分類し，出射輝度の計算式を以下のように変形する．

$$\begin{aligned} L(x, x_v) &= \sum_{i=1}^{N} L_{\mathbb{C}_i}(x, x_v) \\ L_{\mathbb{C}_i}(x, x_v) &= \sum_{y \in \mathbb{C}_i} I(y) f(y, x, x_v) G(y, x) V(y, x) \end{aligned} \tag{5.2}$$

ここで, すべての VPL を含む集合を$\mathbb{S}$とすると, クラスタ$\mathbb{C}_i$は$\mathbb{S}$の部分集合であり, $\bigcup_{i=1}^{N} \mathbb{C}_i = \mathbb{S}$と，$i \neq j$のとき$\mathbb{C}_i \cap \mathbb{C}_j = \emptyset$を満たす．上式において，クラスタ内の VPL の寄与の総和である$L_{\mathbb{C}_i}$をクラスタの寄与と呼び，Lightcuts では，クラスタ毎にN_s個の VPL サンプルを用いてクラスタ寄与を推定し，出射輝度の推定値を計算する．

$$\hat{L}(x, x_v) = \sum_{i=1}^{N} \hat{L}_{\mathbb{C}_i}(x, x_v)$$
$$\hat{L}_{\mathbb{C}_i}(x, x_v) = \frac{1}{N_s} \sum_{s=1}^{N_s} \frac{I(y_s) f(y_s, x, x_v) G(y_s, x) V(y_s, x)}{p(y_s)} \tag{5.3}$$

ここで，pは確率質量関数，y_sはs回目の試行における VPL サンプルであり確率質量関数pに従う．クラスタ寄与を高精度に推定するためには，VPL の寄与$I \cdot f \cdot G \cdot V$に比例した確率質量関数を使用するべきであるが，放射強度I以外はシェーディング点に依存するため，シェーディング点毎に確率分布を構築する必要があり計算コストが高い. そこで Lightcuts では，シェーディング点に依存しない放射強度Iに比例した確率質量関数$p(y) = I(y) / \sum_{y' \in \mathbb{C}_i} I(y')$を用い，以下の式でクラスタの寄与を推定する．

$$\hat{L}_{\mathbb{C}_i}(x, x_v) = \frac{1}{N_s} \sum_{s=1}^{N_s} \frac{I(y_s) f(y_s, x, x_v) G(y_s, x) V(y_s, x)}{\frac{I(y_s)}{\sum_{y \in \mathbb{C}_i} I(y)}}$$
$$= \frac{I_i}{N_s} \sum_{s=1}^{N_s} f(y_s, x, x_v) G(y_s, x) V(y_s, x) \tag{5.4}$$

ここで，I_iはクラスタ内の VPL の放射強度の総和（$\sum_{y \in \mathbb{C}_i} I(y)$）である．

Lightcuts では，各シェーディング点で VPL のクラスタリングを行う．そのため，効率的にレンダリングを行うためには，高速な VPL のクラスタリング方法が必要である．Lightcuts では，VPL の効率的なクラスタリングのために，Light Tree と呼ばれる VPL の 2 分木構造を用いる．Light Tree の葉ノードは個々の VPL に対応し，内部ノードは子孫ノードの VPL を包含するクラスタに対応する．Light Tree の構築は，VPL の生成後に一度だけ行われる．Light Tree を用いた VPL のクラスタリング方法は次の通りである．まず，クラスタの集合を$\mathbb{C}$とし，$\mathbb{C}$の要素を，すべての VPL を包含するクラスタである Light Tree の根ノードで初期化する．そして以降は，出射輝度の推定値の誤差を効率的に低減するために，

クラスタ集合$\mathbb{C}$からクラスタ寄与推定値の誤差の上限値$E_{\mathbb{C}_i}$が最大のクラスタを取り出し，クラスタを分割，分割されたクラスタをクラスタ集合$\mathbb{C}$に加える処理を，クラスタ集合$\mathbb{C}$内のすべてのクラスタが条件$E_{\mathbb{C}_i} < \varepsilon\hat{L}(x, x_v)$を満たすまで繰り返す．ここで，$\varepsilon$はユーザ指定パラメータで，許容相対誤差を表す．クラスタに対応するノードの 2 つの子ノードは，分割されたクラスタを表すため，クラスタの分割にかかるコストは無視できる．クラスタ寄与推定値の誤差の上限値$E_{\mathbb{C}_i}$は以下の式で計算される．

$$E_{\mathbb{C}_i} = I_i f_{ub}(\mathbb{C}_i, x, x_v) G_{ub}(\mathbb{C}_i, x) V_{ub}(\mathbb{C}_i x) \tag{5.5}$$

ここで，f_{ub}，G_{ub}，V_{ub}はそれぞれクラスタ内の VPL に対する BRDF，幾何項，可視関数の上限値である．可視関数の取りうる値は 0 か 1 であるため，Lightcuts では$V_{ub} = 1$と設定している．

Lightcuts により効率的に写実的な画像を生成することができるが，いくつかの問題がある．1 つ目の問題として Lightcuts では，刺激の弁別閾（差を知覚できる最小の差）は基礎刺激の強度に比例するというウェーバーの法則[6]に基づき推定値の相対誤差$\left|L(x, x_v) - \hat{L}(x, x_v)\right| / \hat{L}(x, x_v)$を制御するのが本来の目的であったが，実際のクラスタリング方法では，相対誤差を制御できていない．クラスタ寄与推定値の誤差の上限値$E_{\mathbb{C}_i}$を用いて，$\left|L(x, x_v) - \hat{L}(x, x_v)\right| \approx \sum_{i=1}^{N} E_{\mathbb{C}_i}$として，クラスタ分割停止条件を$\sum_{i=1}^{N} E_{\mathbb{C}_i} < \varepsilon\hat{L}(x, x_v)$とすることも可能であるが，誤差を過度に大きく評価してしまい，計算時間がかかりすぎてしまう．2 つ目の問題として，使用できる材質モデルが，拡散反射，Blinn-Phong，等方 Ward のような単純な材質モデルに制限される点がある．これは，クラスタ寄与推定値の誤差の上限値の計算において，クラスタ内の VPL に対する BRDF の上限値の計算が必要となるためであり，定義式が複雑な BRDF では，上限値を計算することができない．

5.4　誤差推定フレームワーク

5.4.1　概要

図 5.1 に提案法の処理の流れ，アルゴリズム 5.1 に提案法の疑似コードを示す．提案法の処理の流れは基本的に Lightcuts と同様であり，まず VPL を生成したのち，Light Tree を構築する[56]．そして，各シェーディング点では，Light Tree を用いて VPL をクラスタリングし，視点方向に出射する輝度を推定する．出射輝度の推定精度は同じクラスタ数でもシーンや材質，視点によって大きく変化するため，クラスタ数をユーザ指定パラメータとしてしまうと，ユーザは所望する画質の画像を得るのが極めて困難となる．そこで提案法では，以下の分割停止条件を用いてクラスタリングを行うことで，出射輝度の推定値の相対誤差がユーザの許容できる相対誤差ε以下となるクラスタ数を自動で計算する．

$$\left|\hat{L}(x,x_v)-L(x,x_v)\right|<\varepsilon\hat{L}(x,x_v) \tag{5.6}$$

ここで，Lはすべての VPL を使用したときの出射輝度であり，$\hat{L}$はその推定値である．しかしながら上式を用いるには，推定対象である未知の値$L(x,x_v)$が必要となる．そこで提案法では，確率αで真値$L(x,x_v)$が存在する区間$\left[\hat{L}-\Delta L,\hat{L}+\Delta L\right]$，すなわち，以下の式を満たす$\Delta L$を計算する．

$$P_r\left(\left|\hat{L}-L\right|\leq\Delta L\right)=\alpha \tag{5.7}$$

なお，確率αはユーザが指定する値である．上式を満たすΔLはt分布[47]を用いることで計算でき，クラスタ分割停止条件を$\Delta L<\varepsilon\hat{L}$とすることで，確率$\alpha$で相対誤差が$\varepsilon$以下である出射輝度の推定値を計算できる．

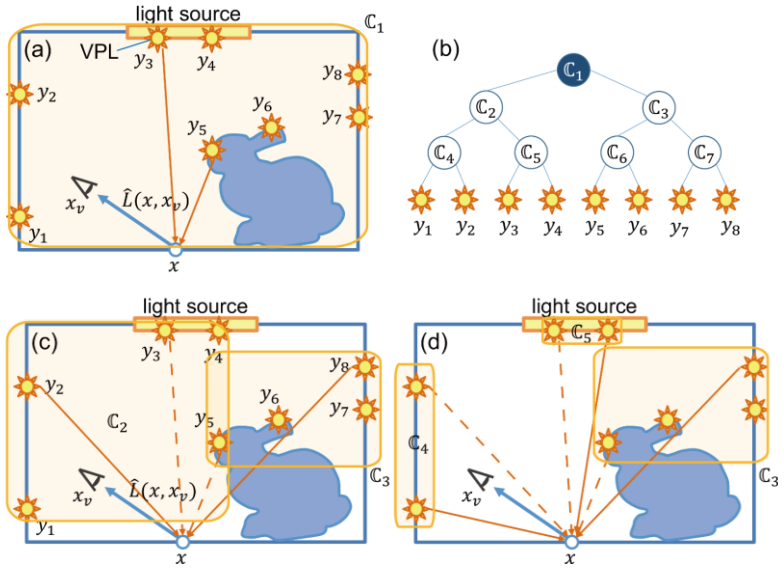


図 5.1　提案法の概要．提案法では，すべての VPL を包含するクラスタ（図(a)）から始めて，条件$\Delta L < \varepsilon \hat{L}$を満たすまでクラスタの分割を繰り返す．クラスタの分割は Light Tree（図(b)）を用いて行われる．Light Tree は葉ノードが個々の VPL，内部ノードが子孫ノードの VPL を包含するクラスタである．そのため，すべての VPL を包含するクラスタ$\mathbb{C}_1$は Light Tree の根ノードに対応し，$\mathbb{C}_1$を分割したクラスタ$\mathbb{C}_2$と$\mathbb{C}_3$（図(c)）は根ノードの子ノードに対応する．$\mathbb{C}_2$と$\mathbb{C}_3$においてクラスタ寄与の推定値$\hat{L}_{\mathbb{C}_2}$と$\hat{L}_{\mathbb{C}_3}$を計算する際，$\hat{L}_{\mathbb{C}_1}$の計算（図(a)）で VPL y_3とy_5の寄与は既に明らかになっているため，再利用が可能であり，追加で計算するのは VPL y_2とy_8の寄与のみでよい．分割候補のクラスタが複数存在する際は，クラスタ寄与の推定値の標準偏差σ_2とσ_3の大きいクラスタを選択し分割する（図(d)においては$\mathbb{C}_2$が選択）．

アルゴリズム 5.1　提案法の輝度推定アルゴリズム．許容相対誤差εと誤差制御の信頼度αを入力とし輝度推定値を返す．

1: クラスタ集合$\mathbb{C}$を$\{\mathbb{C}_1\}$で初期化
2: $\hat{L}$，ΔL，σ_1を推定
3: **while** $\Delta L > \varepsilon \hat{L} \vee \sqrt{2}\sigma_i > \varepsilon \hat{L}$ $(\forall \mathbb{C}_i \in \mathbb{C})$ **do**
4: 　クラスタ集合$\mathbb{C}$から標準偏差が最大のクラスタ$\mathbb{C}_k$を取り出し
5: 　クラスタ$\mathbb{C}_k$を分割しクラスタ$\mathbb{C}_L$，$\mathbb{C}_R$を生成
6: 　クラスタ$\mathbb{C}_L$，$\mathbb{C}_R$を$\mathbb{C}$に追加
7: 　$\hat{L}_{\mathbb{C}_L}$，$\hat{L}_{\mathbb{C}_R}$，σ_L，σ_Rを推定
8: 　$\hat{L}$，ΔLを更新
9: **end while**

5.4.2 VPL のクラスタリング

提案法では Lightcuts と同様，大量の VPL を効率的に扱うため，VPL をクラスタリングし，各クラスタで VPL をサンプリングすることでクラスタ寄与$L_{\mathbb{C}_i}$を推定する．VPL のクラスタリングを定義域の分割と考えると，提案法による輝度の推定方法は層化サンプリングであると解釈できる．層化サンプリングにおいて，各層の推定値が正規分布に従い，各層に割り当てるサンプル数がネイマン配分（推定値の標準偏差に比例した割合でサンプル数を分配）[47]であると仮定すると，以下の式で計算される統計量Tは自由度$n-N$のt分布に従う（導出の詳細は付録 B を参照）．

$$T = \sqrt{\frac{n-N}{n}} \frac{\hat{L} - L}{\sqrt{\sum_{i=1}^{N} s_i^2 / n_i}} \tag{5.8}$$

ここで，Nは層（クラスタ）数，n_iとs_i^2はi番目の層に割り当てられたサンプル数と標本分散，nは合計サンプル数（$n = \sum_{i=1}^{N} n_i$）である．統計量Tがt分布に従うことから，以下のように式変形を行うことで$P_r\left(\left|\hat{L} - L\right| \le \Delta L\right) = \alpha$を満たす$\Delta L$を求めることができる．

$$P_r(|T| \le t_\alpha) = \alpha$$

$$P_r\left(\left|\sqrt{\frac{n-N}{n}} \frac{\hat{L} - L}{\sqrt{\sum_{i=1}^{N} s_i^2 / n_i}}\right| \le t_\alpha\right) = \alpha$$

$$P_r\left(\left|\hat{L} - L\right| \le t_\alpha \sqrt{\frac{n}{n-N} \sum_{i=1}^{N} \frac{s_i^2}{n_i}}\right) = \alpha \tag{5.9}$$

$$\therefore \Delta L = t_\alpha \sqrt{\frac{n}{n-N} \sum_{i=1}^{N} \frac{s_i^2}{n_i}}$$

ここで，$\int_{-t_\alpha}^{t_\alpha} t(x)dx = \alpha$を満たす値$t_\alpha$は$\alpha$分位点と呼ばれ，$t$分布表から算出できる．

提案法による推定値$\hat{L}$の分散は，VPL をどのようにクラスタリングするかや，合計サンプル数nの下で，どのように各クラスタにサンプル数n_iを割り当てるかに依存する．そのため提案法では，効率的なレンダリングのため，分散の低減を目標とした VPL のクラスタリング，サンプル数の割り当てを行う．層化サンプリングでは，推定値の分散は各層の推定値の分散の和で計算されるため，VPL のクラスタリングにおいて分割するクラスタを選択する際は，最も分散の大きいクラスタを選択し分割する．これにより，クラスタ分割により分散を効率的に低減できる．また層化サンプリングにおいて，最も分散を低減できるサンプルの割り当て方法は，各層の推定値の標準偏差に比例して割り当てるネイマン配分であり，各層に割り当てるサンプル数n_iは以下の式で計算される．

$$n_i = \frac{\sigma_i}{\sum_{k=1}^{N} \sigma_k} n \tag{5.10}$$

ここで，σ_iは各層の推定値の標準偏差である．

提案法では，VPL のクラスタリングにおいて分散が大きいクラスタを優先的に分割していく．そのためクラスタの分割が進むほど，各クラスタ寄与の推定値の標準偏差は一様に近づいていき，各層に割り当てるサンプル数も一様に近づく．また層化サンプリングでは，定義域の分割数を減らして各層に多くのサンプルを割り当てるより，定義域の分割数を増やして各層に少しのサンプルを割り当てた方が全体の分散を低減できるという特性がある[38]．よって提案法では，合計サンプル数nでより分散の小さい推定値を計算するために，式(5.9)において必要な標本分散s_i^2の計算に必要な最低数である 2 サンプルを各層に割り当てる．式(5.9)に，$n_i = 2$と$n = 2N$を代入した式は以下の通りである．

$$\Delta L = t_\alpha \sqrt{\sum_{i=1}^{N} s_i^2} \tag{5.11}$$

標準偏差の推定

VPL のクラスタリングにおいて，分割するクラスタを選択する際に，クラスタ寄与の推定値の標準偏差σ_iが必要となる．標本分散s_i^2から推定することも可能であるが，各層では 2 サンプルしか使用しないため，高精度な推定は期待できない．そこで提案法では，以下の式で標準偏差を近似する．

$$\begin{aligned} \sigma_i &= \sqrt{var[I \cdot f \cdot G/p]} \\ &= I_i \sqrt{var[f \cdot G \cdot V]} \\ &\approx I_i \hat{f}_{ub}(\mathbb{C}_i, x, x_v) G_{ub}(\mathbb{C}_i, x) \sqrt{var[V]} \end{aligned} \tag{5.12}$$

なお，表記簡略化のため一部関数の引数を省略した．ここで，$\hat{f}_{ub}$とG_{ub}はクラスタ内の VPL に対する BRDF の上限値の推定値と幾何項の上限値であり，クラスタ$\mathbb{C}_i$内の VPL に対する BRDF と幾何項の値は上限値で一定であると近似した．Lightcuts と異なり提案法では，任意の材質モデルを扱うため BRDF の上限値には推定値を用いる．BRDF の上限値の推定方法は後述する．可視関数の分散$var[V]$は，可視関数の期待値$\bar{V}_i$を用いて以下の式で計算される．

$$\begin{aligned} \bar{V}_i &= \sum_{y \in \mathbb{C}_i} V(y,x) p(y) \\ var[V] &= \mathbb{E}[V^2] - \mathbb{E}[V]^2 \\ &= \mathbb{E}[V] - \mathbb{E}[V]^2 \\ &= \bar{V}_i - \bar{V}_i^2 \\ &= -(\bar{V}_i - 0.5)^2 + 0.25 \end{aligned} \tag{5.13}$$

ここで，pは確率質量関数であり，$p(y) = I(y)/I_i$である．上式では，可視関数の取りうる値は 0 か 1 であるため，$V^2 = V$であることを利用した．提案法では可視関数の分散$var[V]$を上限値である 0.25 で近似して標準偏差を計算する．また 5.4 節では，可視関数の値をキャッシングすることで分散$var[V]$を正確に近似する方法について説明する．

BRDF 上限値の推定

標準偏差σ_iの近似値を計算するため，クラスタ$\mathbb{C}_i$内の VPL に対する BRDF の上限値が必要である．Lambertian, Blinn-Phong, Ward BRDF のような単純なモデルの場合は上限値を解析的に計算できるが[56]，複雑なモデルでは解析的に計算することができない．そこで提案法では，図 5.2 に示すように，3 つの方法を組み合わせて BRDF の上限値を推定する．1 つ目は輝度推定における情報を利用する方法で，クラスタの寄与を推定する際に用いる 2 サンプルy_1，y_2に対する BRDF の値f_1，f_2を使用する．2 つ目は BRDF の重点的サンプリングを利用した方法で，まず各シェーディング点で BRDF の重点的サンプリングによりN_f本のレイを生成する．そして，生成したレイとクラスタのバウンディングボックスで交差判定を行い，交差したレイの内で BRDF の最大値f_{max}を使用する．最後に 3 つ目の方法では，シェーディング点の法線との角度が最小となるクラスタ方向$\omega_{\mathbb{C}_i}$に対する BRDF の値$f_{\omega_{\mathbb{C}_i}}$を使用する．なお，方向$\omega_{\mathbb{C}_i}$は幾何項の上限値の計算において計算される．最終的な BRDF 上限値の推定値は，f_1，f_2，f_{max}，$f_{\omega_{\mathbb{C}_i}}$の最大値である．

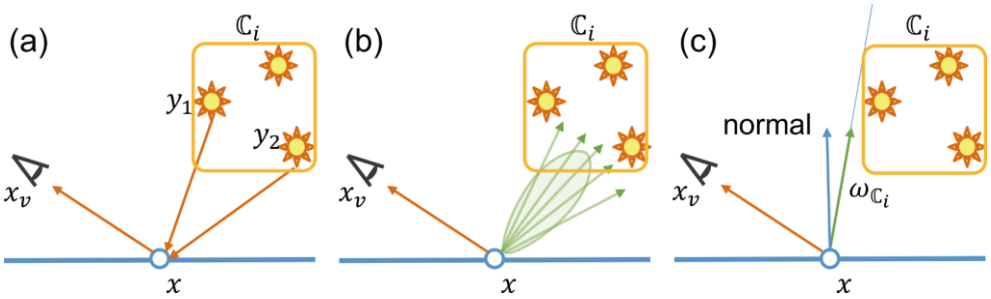


図 5.2　BRDF 上限値の推定．提案法では BRDF 上限値を 4 つの BRDF の値f_1，f_2，f_{max}，$f_{\omega_{\mathbb{C}_i}}$の最大値で近似する．$f_1$と$f_2$はクラスタ寄与の推定値$\hat{L}_{\mathbb{C}_i}$の推定に用いる 2 つの VPL y_1とy_2に対する BRDF の値である（図(a)）．f_{max}は BRDF の重点的サンプリングにより生成したレイで，クラスタ$\mathbb{C}_i$のバウンディングボリュームに交差したレイの方向に対する BRDF の最大値である（図(b)）．$f_{\omega_{\mathbb{C}_i}}$はシェーディング点の法線とクラスタ方向のなす角が最小となる方向に対する BRDF の値である（図(c)）．

追加の分割停止条件

先述した通り，ΔLは各層の推定値が正規分布に従い，各層に割り当てるサンプル数はネイマン配分で行われることを仮定していた．提案法のクラスタリング方法では，クラスタの分割が進めば各クラスタの分散が一様に近づくため，ネイマン配分の仮定は満たされる．しかしながら，各層の推定値が正規分布に従うという仮定は，クラスタの分割が進んだとしても満たすのは困難である．ただし幸いなことに，t分布はロバストな分布であり，合計サンプル数nが十分大きい場合，完全に仮定を満たしていなくても，式(5.8)で計算される統計量Tはt分布に従う．そこで提案法では，合計サンプル数nが十分大きいかを判定するために，分割停止条件に以下の条件を加える．

$$\sqrt{2}\sigma_i < \varepsilon\hat{L} \ (\forall \mathbb{C}_i \in \mathbb{C}) \tag{5.14}$$

上式は，Lightcuts の分割停止条件$E_{\mathbb{C}_i} < \varepsilon\hat{L}$を基にしたもので，$\sqrt{2}$はサンプル数$n_i$の違いや，$\sigma_i$における$\sqrt{var[V]}$と$E_{\mathbb{C}_i}$における$V_{ub}$の違いを考慮したスケーリング係数である．図 5.3 に

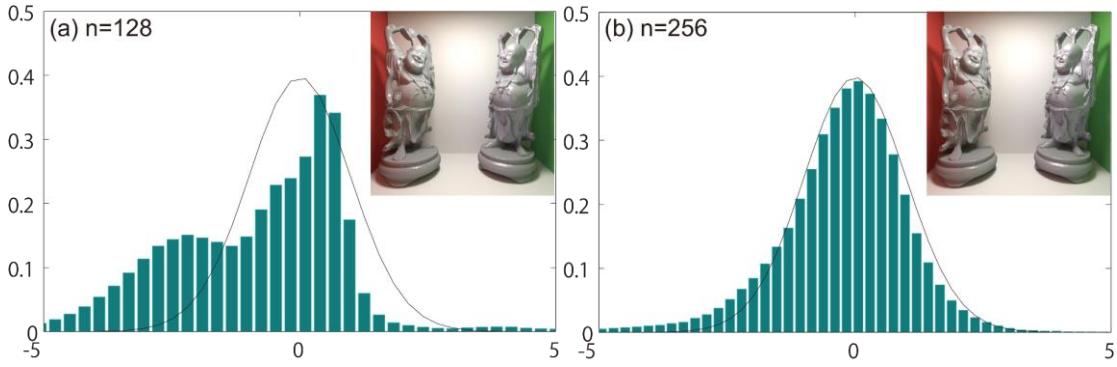


図 5.3　合計サンプル数$n = 128$と256のときの統計量Tのヒストグラムによる可視化．横軸は統計量T，縦軸は確率，実線はt分布を表す．ヒストグラムは挿絵画像の中心ピクセルにおいて計測したものである．合計サンプル数が増えるほど統計量Tがt分布に従うことが分かる．

合計サンプル数nを変えたときの統計量Tの分布を可視化した結果を示す．図 5.3 に示すように，サンプル数nが増えることで，統計量Tがt分布に近づいていることが分かる．

提案法の VPL クラスタリング処理を要約すると，すべての VPL を含むクラスタから始めて，条件$\Delta L < \varepsilon \hat{L}$と式(5.14)を満たすまでクラスタの分割を繰り返す．条件を満たさない場合は，標準偏差σ_kが最大のクラスタ$\mathbb{C}_k$をクラスタ集合$\mathbb{C}$から選択し，分割したクラスタ$\mathbb{C}_L$，$\mathbb{C}_R$で置き換える．そして，クラスタ寄与の推定値$\hat{L}_{\mathbb{C}_L}$と$\hat{L}_{\mathbb{C}_R}$，標準偏差σ_Lとσ_Rを計算し，推定値$\hat{L}$とΔLを更新する．

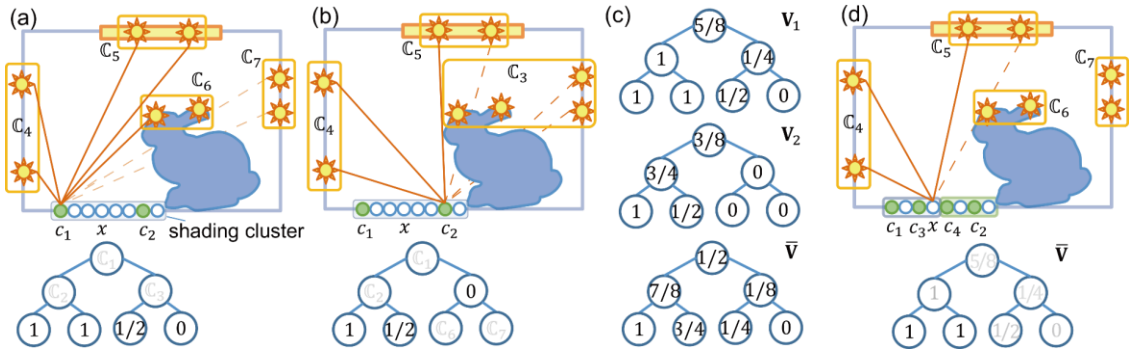


図 5.4　Visibility Caching の概要．シェーディング点クラスタからランダムに選んだ点c_1，c_2をキャッシュ点とし，各キャッシュ点では 5.3 節の方法で生成した VPL クラスタに対して可視関数の平均値を推定する（図(a)，図(b)）．次に各キャッシュ点で，先祖ノードに対する可視関数の平均値$\mathbf{V}_1$，$\mathbf{V}_2$を推定し，それらの平均値$\bar{\mathbf{V}}$を計算する（図(c)）．シェーディング点クラスタ内で可視関数の相関が小さい場合は，シェーディング点クラスタを分割し，同様の処理を行う（図(d)）．

5.5　Visibility Caching

5.5.1　概要

提案法の輝度推定の精度は，VPL のクラスタリング方法に依存し，標準偏差が最大のクラスタを優先的に分割することでクラスタリングを行っていた．5.3 節では，式(5.12)の標準偏差の計算において，可視関数の分散$var[V]$は上限値で近似していた．本節では，近傍シェーディング点間における可視関数の相関を利用し，可視関数の情報をキャッシングすることで，より正確に可視関数の分散$var[V]$を計算する．これにより不可視である VPL クラスタが分割対象に選択され，無駄なサンプルを計算するのを防ぐことができ，計算コストを抑えることができる．

図 5.4 に Visibility Caching の流れを示す．まず，全シェーディング点を生成し，シェーディング点を位置と法線に基づきクラスタリングする[61]．次に，各シェーディング点クラスタで，ランダムにシェーディング点を数個選択し，可視関数を記録するキャッシュ点とする．キャッシュ点では，5.4 節で説明した方法で輝度推定を行い，輝度推定で用いたサンプ

ルの情報から，VPL クラスタに対する可視関数の平均値を計算し記録しておく．そして，クラスタ内のシェーディング点では，同クラスタの各キャッシュ点で記録されている VPL クラスタに対する可視関数の平均値を，さらに平均した値を用いて可視関数の分散$var[V]$を計算する．なお，各キャッシュ点で可視関数の平均が大きく異なる場合は，シェーディング点クラスタを分割する．

5.5.2　可視関数平均値の計算とシェーディング点クラスタの精緻化

各キャッシュ点では，VPL クラスタに対する可視関数の平均値を記録する．しかしながら，クラスタ内のすべての VPL に対して可視関数を計算し平均をとるのはコストが高い．そこで提案法では，各キャッシュ点c_jにおいて 5.4 節で説明したクラスタリング方法で出射輝度の推定値$\hat{L}$を計算する．そして，輝度推定で使用したクラスタ$\mathbb{C}_i$とキャッシュ点c_j間の平均可視関数値$\hat{V}_i$を$\{V(y_{i,1}, c_j) + V(y_{i,2}, c_j)\}/2$で推定する．ここで，$y_{i,1}$と$y_{i,2}$はクラスタ$\mathbb{C}_i$の寄与推定に用いた VPL サンプルである．先祖ノードの VPL クラスタ$\mathbb{C}_a$に対する平均可視関数値は以下の式で推定する．

$$\hat{V}_a = \left(\sum_{y \in \mathbb{C}_L} p(y) \right) \hat{V}_L + \left(\sum_{y \in \mathbb{C}_R} p(y) \right) \hat{V}_R \tag{5.15}$$

ここで，$\mathbb{C}_L$と$\mathbb{C}_R$は VPL クラスタ$\mathbb{C}_a$の分割後のクラスタであり，$\hat{V}_L$と$\hat{V}_R$はクラスタ$\mathbb{C}_L$と$\mathbb{C}_R$に対する平均可視関数の推定値，pはクラスタ$\mathbb{C}_a$における確率質量関数である．上式は下層ノードのクラスタから計算することで効率的に評価できる．なお，クラスタ$\mathbb{C}_j$を分割してできるクラスタに対する平均可視関数値は，クラスタ$\mathbb{C}_j$に対する平均可視関数値$\hat{V}_j$を用いる．

シェーディング点クラスタ内の各キャッシュ点c_jで VPL クラスタに対する平均可視関数値$\mathbf{V}_j$を計算できれば，さらに各キャッシュ点間で平均可視関数値の平均$\bar{\mathbf{V}}$を計算する．そし

て，シェーディング点クラスタ内の可視関数の類似度を測るための指標として，各キャッシュ点c_jで以下の値を計算する．

$$\delta_j = \frac{1}{N_j}\sum_{i=1}^{N_j}\left(\tilde{V}_i - \frac{V(y_{i,1}, c_j) + V(y_{i,2}, c_j)}{2}\right)^2 \tag{5.16}$$

上式は，キャッシュ点c_jにおけるクラスタ集合$\mathbb{C}$内のN_j個のクラスタに対する平均であり，$\tilde{V}_i$は$\bar{\mathbf{V}}$に記録されたi番目のクラスタに対する平均可視関数値である．δ_jが閾値δ_{VC}より大きい場合，シェーディング点クラスタ内で可視関数の類似度が小さいと見なし，シェーディング点クラスタを分割し，各シェーディング点クラスタで再度キャッシュ点を生成する．

各シェーディング点では VPL のクラスタリングを行う際は，式(5.12)で計算される標準偏差が最大のクラスタを選択し分割する．このとき，$\bar{V}_i$を$\bar{\mathbf{V}}$に記録されている値$\tilde{V}_i$で近似して$var[V]$を計算する．ただし，$\tilde{V}_i = 0$または$\tilde{V}_i = 1$のとき，$var[V] = 0$より$\sigma_i = 0$となり，実際は分散が大きいクラスタでも，クラスタが分割されなくなってしまう．そのため提案法では，$\tilde{V}_i$を0.01と0.9の間にクランプした値で$var[V]$を用い$\sigma_i = 0$となるのを防いだ．

5.6　実験結果

本節では，提案法の実験結果と先行研究である Lightcuts との比較を行う．実験は Intel Xeon E5-2697 v2 2.70GHz CPU を搭載した PC で行い，マルチスレッドによる並列化を行っている．パラメータ設定に関しては明示しない限り，許容相対誤差εに2%，信頼度αに95%，BRDF 上限値の推定に用いるレイの数N_fに256，シェーディング点クラスタ分割の閾値δ_{VC}に 0.1 を設定した．また，シェーディング点クラスタ内のキャッシュ点数は10とした．なお許容相対誤差$\varepsilon = 2\%$の設定は先行研究[56]に基づいており，推定値の相対誤差が2%以下の場合，誤差をほぼ知覚できない．提案法では，推定値の誤差制御が目標であることから，実

際に相対誤差がε以下となっているピクセルの割合R_εを計測した．R_εがαに近いほど，正確に相対誤差を制御できていることを表す．VPL クラスタから確率質量関数pに従い VPL をサンプリングする際は，Lightcuts[55]と同様，サンプリングのコストを抑えるために，事前にpに従いサンプリングしたものを Light Tree のノードに記録しておき，一様サンプリングで選びなおす方法を用いる．

図 5.5 に，Sponza シーンにおいて提案法のレンダリング結果と，提案法と Lightcuts での相対誤差を可視化した画像を示す．なお，Lightcuts の許容相対誤差εも2%に設定しており，提案法の計算時間は 282 秒，Lightcuts の計算時間は 60 秒である．提案法では実際に相対誤差が2%以下のピクセルの割合R_εは92.96%であり，ほぼ指定した$\alpha = 95\%$通りに誤差を制御できている．一方 Lightcuts では，計算時間は提案法より短いが，実際に相対誤差が2%以下のピクセルの割合は43.67%しかなく, Lightcuts では相対誤差が制御できていない．図 5.6 は，図 5.5 において提案法と Lightcuts の結果の拡大画像である．Lightcuts では，相対誤差の制御ができていないため，結果画像にノイズが含まれているのが分かるが，提案法の結果画像にノイズは視認できない．先述したように Lightcuts において，$\sum_{i=1}^{N} E_{\mathbb{C}_i}$を推定値の誤差として条件$\sum_{i=1}^{N} E_{\mathbb{C}_i} < \varepsilon\hat{L}$で VPL のクラスタリングを行うこともできる．この条件の下，$\varepsilon = 2\%$で実験したところ，計算時間は 16 時間（58,078 秒）であり，平均相対誤差は 5.69214e-06 となり，$\sum_{i=1}^{N} E_{\mathbb{C}_i}$を推定値の誤差として使用するのは適切ではないことが分かる．

画像生成の効率の比較のため，図 5.5 のシーンにおいて，Lightcuts の相対誤差が2%以下のピクセルの割合が提案法と同程度なるパラメータεで計算時間の比較を行う．なお，Lightcuts では相対誤差を制御できないため，パラメータは試行錯誤で発見し，$\varepsilon = 0.3\%$のとき，提案法と同程度の割合となり，このときの Lightcuts の計算時間は 206 秒となった．Lightcuts では BRDF の上限値計算において解析解を用いているのに対し，提案法では推

定値を用いており，提案法の方が BRDF 上限値の計算コストが高い．計算時間の比較を公平にするため，提案法でも BRDF の上限値計算において解析解を用いた場合，計算時間は 209 秒となり，提案法と Lightcuts の計算時間はほぼ同じとなった．つまり提案法は，相対誤差を制御可能でありながら，Lightcuts と同程度の効率で画像生成を行うことが可能である．

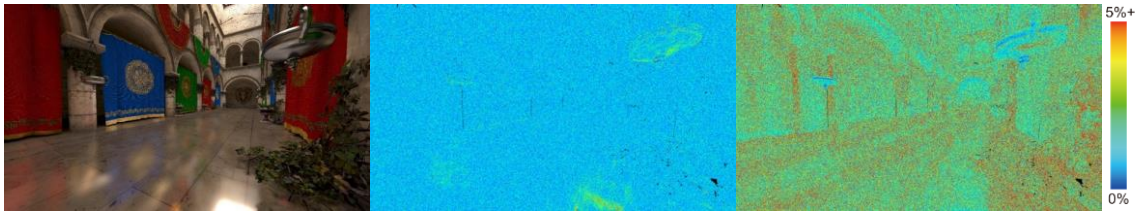
図 5.7 と図 5.8 に，Kitchen シーンと San Miguel シーンにおいて，許容相対誤差εを変えたときの実験結果を示す. これらのシーンでは, 複雑な BRDF である Cook-Torrance BRDF と Ashkhmin-Shrely BRDF を用いており，解析的に BRDF の上限値を計算できない．結果から分かるように，提案法では，許容相対誤差εを変えても，指定通り相対誤差がε以下のピクセルの割合を 95%近くに制御できている．

BRDF 上限値推定のための BRDF 重点的サンプリングのサンプル数N_fと誤差制御の精度R_εと計算時間T_uの関係を調べるため, 図 5.9 に, 複雑な BRDF モデルである Cook-Torrance BRDF を適用した Buddha シーンでのレンダリング結果と相対誤差を可視化した画像，N_fによるR_εとT_uの推移グラフを示す．図 5.9(c)より，N_fを増やすほど BRDF 上限値の推定が正確になるためR_εが増加していき，N_fが 256 を超えてからは，正確な推定を行う上で十分なサンプル数に達したため，R_εの増加傾向が緩やかになっている．そして図 5.9(d)より，N_fを増やすほど計算時間が増加し，N_fが 256 を超えたあたりから，計算時間の増加傾向が激しくなっている．そのため提案法では，十分な推定精度が得らえる最低数である 256 をN_fの値として用いる．

図 5.10 に，Blinn-Phong BRDF を適用した Buddha シーンでの，相対誤差の分布をヒストグラムで可視化した結果を示す. 提案法は統計的に誤差を制御するため, 相対誤差が大きい推定結果が得られる場合もある．しかしながら図 5.10 より，相対誤差が 2%より大きい場合でも，大部分は2%に近い値に分布しており，結果画像からも統計的な誤差制御による

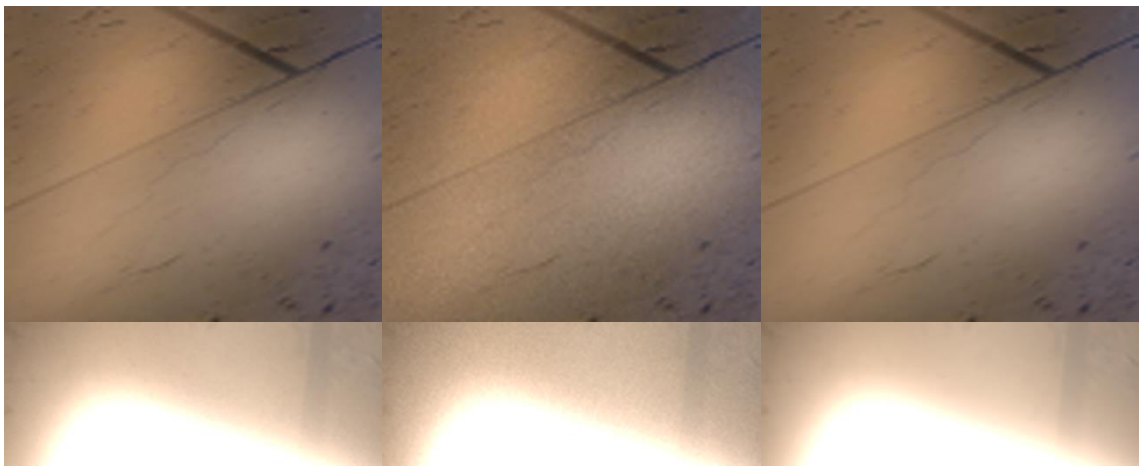
不自然なアーティファクトやノイズは見られなかった.

最後に，これまでの各シーンの情報と計測データをまとめたものを表 5.1，誤差推定の信頼度αを 99%に設定したときの計測データをまとめたものを表 5.2 に示す．ここで，T_cは可視関数のキャッシングありの場合の計算時間，T_uはキャッシングなしの場合の計算時間T_uであり，T_u/T_cは可視関数のキャッシングによる高速化率を表す．表 5.1 より，可視関数のキャッシングを行うことで，誤差制御精度R_εはキャッシングなしの場合と同程度でありながら，18%から 42%高速に画像を生成できている．また表 5.2 より，信頼度αに99%を設定した場合でも，ほぼ指定通りの値に誤差制御の精度R_εを制御できている．



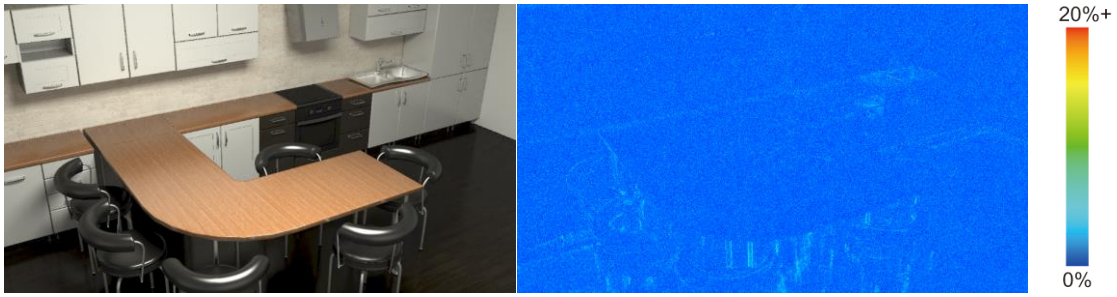
提案法（$R_\varepsilon = 92.96\%$）　　Lightcuts（$R_\varepsilon = 43.67\%$）

図 5.5　Sponza シーンにおける誤差制御の精度の比較．左から順に提案法の結果画像，提案法の相対誤差の可視化画像，Lightcuts の相対誤差の可視化画像である．色と相対誤差の関係はカラーバーに示すとおりである．両手法において許容相対誤差εは 2%に設定し，提案法の信頼度パラメータαは 95%に設定している．

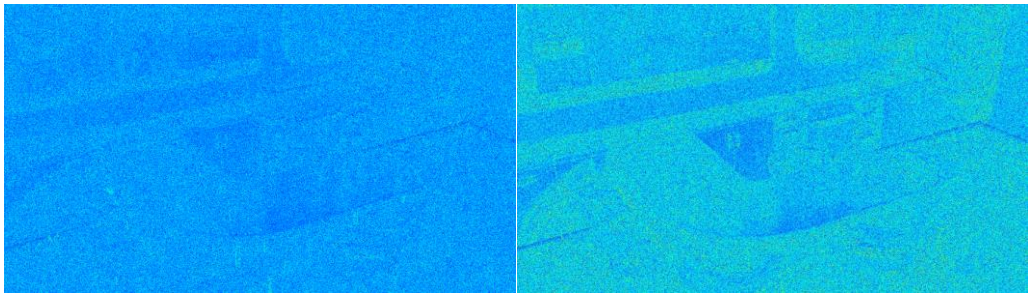


提案法　　Lightcuts　　参照画像

図 5.6　図 5.5 における各手法のレンダリング結果の拡大画像．Lightcuts は誤差制御の精度が低いため結果画像にノイズが現れているのに対し，提案法は統計的に正確な誤差制御を行うため結果画像にノイズはほぼ視認できない．



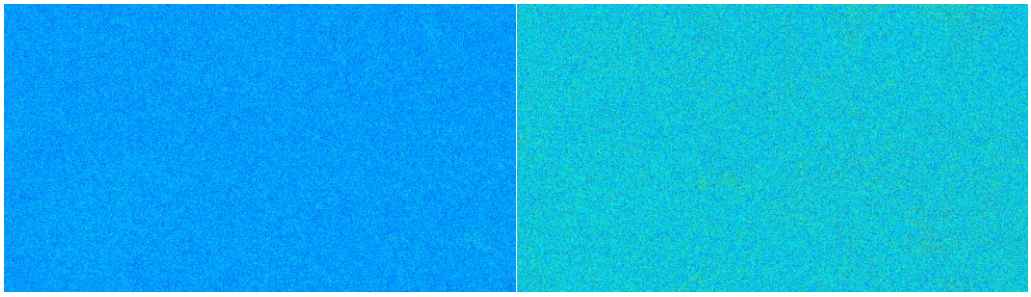
$\varepsilon = 2\%$, $R_\varepsilon = 91.86\%$, $T_c = 70$秒



$\varepsilon = 5\%$, $R_\varepsilon = 93.65\%$, $T_c = 40$秒　　$\varepsilon = 10\%$, $R_\varepsilon = 95.30\%$, $T_c = 31$秒

図 5.7　Kitchen シーンにおける許容相対誤差εごとの誤差制御精度R_εと計算時間T_cの比較．誤差制御の信頼度パラメータαは 95%に設定している．

$\varepsilon = 2\%$, $R_\varepsilon = 92.66\%$, $T_c = 2{,}230$秒



$\varepsilon = 5\%$, $R_\varepsilon = 92.79\%$, $T_c = 757$秒　　$\varepsilon = 10\%$, $R_\varepsilon = 92.07\%$, $T_c = 327$秒

図 5.8　San Miguel シーンにおける許容相対誤差εごとの誤差制御精度R_εと計算時間T_cの比較．誤差制御の信頼度パラメータαは 95%に設定している．

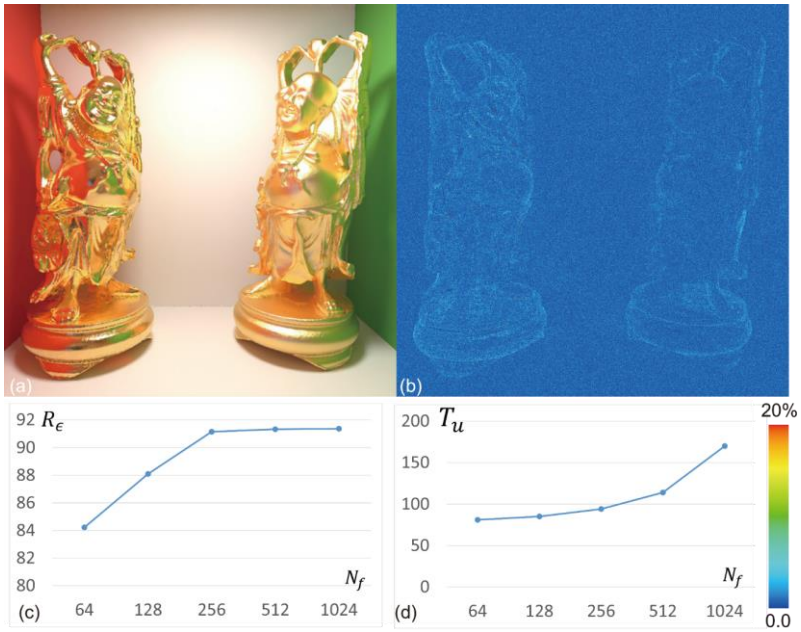


図 5.9　Buddha シーンにおける提案法のレンダリング結果（図(a)）と相対誤差の可視化画像（図(b)），BRDF 上限値推定のための BRDF 重点的サンプリングの回数N_fによる誤差制御精度のグラフ（図(c)）と画像生成時間のグラフ（図(d)）．$N_f = 256$を超えてからは誤差制御精度の変化は小さく，画像生成時間は増加している．

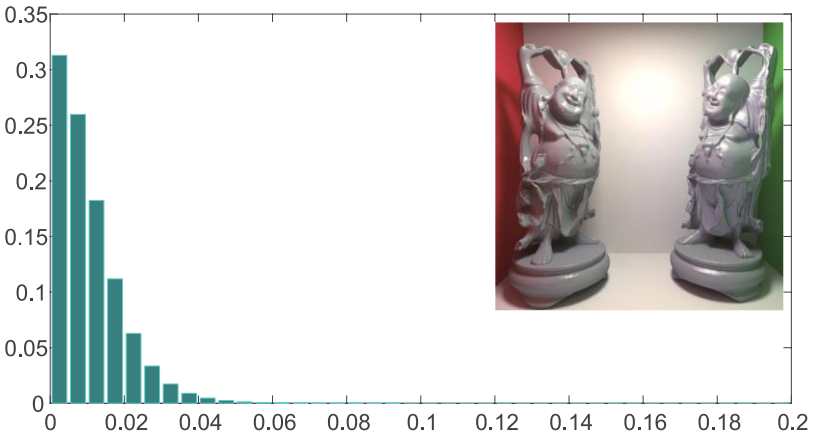


図 5.10　Buddha シーンにおける相対誤差分布のヒストグラムによる可視化．横軸は相対誤差，縦軸は確率を表す．許容相対誤差εは 2%，信頼度αは 95%を設定し，R_εは 93.13%で計算時間は 86 秒である．

表 5.1　シーン情報と計測データのまとめ．計測データは許容相対誤差εに 2%，信頼度αに 95%を指定時のものである．N_{vpl}は VPL 数，N_{tri}は三角形数であり，T_cとT_uはそれぞれ Visibility Caching ありとなしの場合の計算時間，R_εは相対誤差が許容相対誤差ε以下のピクセルの割合，MRE は平均相対誤差である．また，T_u/T_cは Visibility Caching による高速化率を表す．

シーン	Sponza （図 5.5）	Kitchen （図 5.7）	San Miguel （図 5.8）	Buddha （図 5.9）
N_{vpl}	2,061k	2,082k	1,936k	2,075k
N_{tri}	262k	528k	10,464k	1,086k
T_c	282	70	2,230	80
R_ϵ	92.86%	91.86%	92.66%	91.15%
MRE	0.0088	0.0091	0.0089	0.0094
T_u	401	86	2,860	94
R_ϵ	92.90%	92.50%	92.84%	91.19%
MRE	0.0088	0.0089	0.0088	0.0094
T_u/T_c	1.42	1.23	1.29	1.18

表 5.2　許容相対誤差εに 2%，信頼度αに 99%を指定時の計測データのまとめ．

シーン	Sponza （図 5.5）	Kitchen （図 5.7）	San Miguel （図 5.8）	Buddha （図 5.9）
T_c	388	89	3,151	103
R_ϵ	97.67%	96.67%	97.94%	95.93%
MRE	0.0068	0.0072	0.0068	0.0075
T_u	549	114	3,872	126
R_ϵ	97.68%	96.69%	98.06%	95.92%
MRE	0.0068	0.0072	0.0068	0.0075
T_u/T_c	1.41	1.28	1.23	1.22

5.7 まとめ

本研究では，多光源レンダリング法において，区間推定法を用いることで統計的に誤差を制御する手法を提案した. 従来の多光源レンダリング法では, VPL のサンプリングにより，高速に画像生成を行うことができるが，推定値の誤差を制御できないという問題があり，ユーザは，所望の画質の結果を得るために，パラメータ調整とレンダリングの繰り返しが必要であり，結局，長い時間がかかってしまっていた．提案法では，ユーザは許容相対誤差と誤差推定の信頼度を指定するだけでよく，一度のレンダリングで所望する画質の結果を得ることができる．また本研究では，可視関数のキャッシングにより，誤差制御の精度を維持したまま，レンダリングの効率を向上する手法を提案し，実験では 18%から 42%の高速化を実現した.

提案法は Lightcuts に基づく手法であるため，Lightcuts の拡張手法[7, 55, 57]にも提案法の誤差推定フレームワークを適用できる．そのため，今後の課題としては，Lightcuts と同様の拡張により，提案法のさらなる効率化や，被写界深度など様々な光学現象を扱えるようにすることが挙げられる.

第６章　結論

本論文では，写実的な画像の生成における計算効率や計算精度の定式化を行うことで，ユーザのパラメータ調整の労力やヒューリスティックなアルゴリズムを減らしつつ，ロバストかつ効率的に写実的な画像生成を行う 3 つの手法を提案した．まず 1 つ目は，レイの分布を考慮した高速化データ構造の構築と，高速化データ構造使用時の計算効率の定式化により非効率な高速化データ構造の探索を避けることで，様々な大域照明アルゴリズムの基盤的な技術であるレイトレーシングを高速化する手法を提案した．2 つ目は，写実的な画像の効率的な生成法である多光源レンダリング法において，ボトルネックとなっている可視関数の評価を，ある確率で省略することで効率化する手法を提案した．この手法では可視関数評価の省略確率が計算効率に大きく依存するが，提案法では計算効率を定式化することで，最適な省略確率を導出した．最後に 3 つ目は，多光源レンダリング法の VPL クラスタリングによる輝度推定において，推定値が従う分布を定式化し，推定値の誤差を制御することで，一度のレンダリングで，ユーザの所望する画質の結果画像を生成する手法を提案した．本研究により，ユーザは少ないパラメータ調整の労力で効率的に写実的な画像を生成でき，CG による写実的な画像を用いる様々な分野において大幅な作業効率の向上をもたらすことができる．

本論文で提案した 3 つの手法は，手法の性質上，互いに組み合わせることは困難であるが，各手法はそれぞれ異なる用途がある．まず 1 つ目のレイトレーシングの高速化手法について，この手法では大量のレイの交点計算を同時に行うという特殊性から，他の手法との組み合わせには適さない．そのためこの手法は，光源からの直接光だけを考慮したプレビュー用として使用するのが実用的な用途であり，カメラ位置の設定などに使用するとよい．2 つ目の可視判定省略による高速化手法と，3 つ目の誤差推定による高速化手法の組み合わせについては，輝度の推定方法が異なるため組み合わせることはできない．しかしながら，こ

れらの手法は異なる特徴を持ち，2 つ目の手法では，サンプル数が増えるにつれて推定精度が安定的に向上していくのに対し，3 つ目の手法では，サンプル数（クラスタ数）が少ないときは推定精度が非常に低く，サンプル数が多くなるにつれて推定精度が大きく向上するという特徴がある．つまり，計算時間が短くサンプル数が十分でないときは 2 つ目の手法を用いる方がノイズの小さい画像を生成できる．よって，可視判定省略による高速化手法は，光源からの間接光を考慮したプレビュー用として使用し，誤差推定による高速化手法は，推定値の誤差制御が重要となる最終段階で使用するべきである．

最後に各提案法の今後の課題について，以下に簡単にまとめる．

(1) レイトレーシングの高速化（第 3 章）：本手法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 DACRT の効率化を行った．しかしながら，DACRT では単純なマルチスレッドによる並列化は効率的ではなく，近年の CPU のメニーコア化の流れを考えると，DACRT の実用化において，効率的な並列化方法の考案が必要である．

(2) 可視判定省略による高速化（第 4 章）：本手法では多光源レンダリング法において可視判定回数を削減することで効率化を行った．本手法では，可視関数に関する事前情報が必要であり，多光源レンダリング法では，容易に可視関数に関する事前情報を得ることができた．しかしながら，現在の可視関数に関する事前情報の取得方法では，その他の大域照明アルゴリズムに本手法を応用するのが難しい．そのため，本手法の応用範囲を拡大する上で，様々な大域照明アルゴリズムに適用可能な，可視関数に関する事前情報の取得方法の考案が必要である．

(3) 誤差推定による高速化（第 5 章）：本手法は Lightcuts に基づく手法で，誤差制御を正確に行えるように改良した．Lightcuts は様々な光学現象の再現や効率化の拡張がなされており，Lightcuts と同様の拡張を本手法に適用し，さらなる効率化や応用範囲の拡大が今後の課題である．

参考文献

[1] A. T. Afra. Incoherent Ray Tracing without Acceleration Structures. In Proceedings of Eurographics Short Paper, pp. 97-100, 2012.

[2] J. L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM, Vol. 18, No. 9, pp. 509-517, 1975.

[3] N. Billen, B. Engelen, A. Lagae, and P. Dutre. Probabilistic Visibility Evaluation for Direct Illumination. Computer Graphics Forum, Vol. 32, No. 4, pp. 39-47, 2013.

[4] N. Billen, A. Lagae, and P. Dutre. Probabilistic Visibility Evaluation using Geometry Proxies. Computer Graphics Forum, Vol. 33, No. 4, pp. 143-152, 2014.

[5] J. Bittner and V. Havran. RDH: Ray Distribution Heuristics for Construction of Spatial Data Structures. In Proceedings of the 25th Spring Conference on Computer Graphics, pp. 51-58, 2009.

[6] H. R. Blackwell. Luminance Difference Thresholds. In Handbook of Sensory Physiology, pp. 78-101, 1972.

[7] N. Bus, N. H. Mustafa, and V. Biri. Illuminationcut. Computer Graphics Forum, Vol. 34, No. 2, pp. 5651-573, 2015.

[8] R. L. Cook, T. Porter, and L. Carpenter. Distributed Ray Tracing. ACM SIGGRAPH Computer Graphics, Vol. 18, No. 3, pp. 137-145, 1984.

[9] C. Dachsbacher, J. Krivanek, M. Hasan, A. Arbree, B. Walter, and J. Novak. Scalable Realistic Rendering with Many-light Methods. Computer Graphics Forum, Vol. 33, No. 1, pp. 88-104, 2014.

[10] H. Dammertz, J. Hanika, and A. Keller. Shallow Bounding Volume Hierarchies for Fast SIMD Ray Tracing of Incoherent Rays. Computer Graphics Forum, Vol. 27, No. 4, pp. 1225-1233, 2008.

[11] H. Dammertz, J. Hanika, A. Keller, and H. Lensch. A Hierarchical Automatic Stopping Condition for Monte Carlo Global Illumination. In Proceedings of the Winter School of Computer Graphics, pp. 159-164, 2010.

[12] T. Davidovic, J. Krivanek, M. Hasan, P. Slusallek, and K. Bala. Combining Global and Local Virtual Lights for Detailed Glossy Illumination, ACM Transactions on Graphics. Vol. 26, No. 6, pp. 143:1-143:8, 2010.

[13] T. Engelhardt, J. Novak, T.-W. Schmidt, and C. Dachsbacher. Approximate Bias Compensation for Rendering Scenes with Heterogeneous Participating Media. Computer Graphics Forum, Vol. 31, No. 7, pp. 2145-2154, 2012.

[14] N. Feltman, M. Lee, and K. Fatahalian. SRDH: Specializing BVH Construction and Traversal Order Using Representative Shadow Ray Sets. In Proceedings of the Forth ACM SIGGRAPH/Eurographics Conference on High Performance Graphics, pp.49-55, 2012.

[15] A. Fujimoto, T. Tanaka, and K. Iwata. ARTS: Accelerated Ray Tracing System. IEEE Computer Graphics and Applications, Vol. 6, No. 4, pp. 16-26, 1986.

[16] K. Garanzha, J. Pantaleoni, and D. McAlister. Simpler and Faster HLBVH with Work Queue. In Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, pp. 59-64, 2011.

[17] I. Georgiev, J. Krivanek, S. Popov, and P. Slusallek. Importance Caching for Complex

Illumination. Computer Graphics Forum, Vol. 31, No. 2, pp. 701-710, 2012.

[18] I. Georgiev, J. Krivanek, T. Davidovic, and P. Slusallek. Light Transport Simulation with Vertex Connection and Merging. ACM Transactions on Graphics, Vol. 31, No. 6, pp. 192:1-192:10, 2012.

[19] I. Georgiev and P. Slusallek. Simple and Robust Iterative Importance Sampling of Virtual Point Lights. In Proceedings of Eurographics (short papers), 2010.

[20] T. Hachisuka, W. Jarosz, and H. W. Jensen. A Progressive Error Estimation Framework for Photon Density Estimation. ACM Transactions on Graphics, Vol. 29, No. 6, pp. 144:1-144:12, 2010.

[21] T. Hachisuka, J. Pantaleoni, and H. W. Jensen. A Path Space Extension for Robust Light Transport Simulation. ACM Transactions on Graphics, Vol. 31, No. 6, pp. 191:1-191:10, 2012.

[22] M. Hasan, J. Krivanek, B. Walter, and K. Bala. Virtual Spherical Lights for Many-Light Rendering of Glossy Scenes. ACM Transactions on Graphics, Vol. 28, No. 5, pp. 143:1-143:6, 2009.

[23] M. Hasan, F. Pellacini, and K. Bala. Matrix Row-Column Sampling for the Many-Light Problem. ACM Transactions on Graphics, Vol. 26, No. 3, pp. 26:1-26:10, 2007.

[24] Y. Huo, R. Wang, S. Jin, X. Liu, and H. Bao. A Matrix Sampling-and-Recovery Approach for Many-Lights Rendering. ACM Transactions on Graphics, Vol. 34, No. 6, pp. 210:1-210:12, 2015.

[25] H. W. Jensen. Realistic Image Synthesis Using Photon Mapping. A. K. Peters, Ltd., 2001.

[26] J. T. Kajiya. The Rendering Equation. SIGGRAPH Computer Graphics, Vol. 20, No. 4, pp. 143-150, 1986.

[27] J. Kalojanov, M. Billeter, and P. Slusallek. Two-Level Grids for Ray Tracing on GPUs. Computer Graphics Forum, Vol. 30, No. 2, pp. 307-314, 2011.

[28] J. Kalojanov and P. Slusallek. A Parallel Algorithm for Construction of Uniform Grids. In Proceedings of the Conference on High Performance Graphics 2009, pp. 23-28, 2009.

[29] A. Keller. Instant Radiosity. In Proceedings of SIGGRAPH, pp. 49-56, 1997.

[30] A. Keller and C. Wachter. Efficient Ray Tracing without Auxiliary Acceleration Data Structure. High Performance Graphics 2011 (Poster), 2011.

[31] T. Kollig and A. Keller. Illumination in the Presence of Weak Singularities. Monte Carlo and Quasi-Monte Carlo methods, pp. 245-257, 2004.

[32] T. Moller and B. Trumbore. Fast, Minimum Storage Ray-Triangle Intersection. Journal of Graphics Tools, Vol. 2, No. 1, pp. 21-28, 1997.

[33] B. Moon, J. Y. Jun, J. Lee, K. Kim, T. Hachisuka, and S.-E. Yoon. Robust Image Denoising Using a Virtual Flash Image for Monte Carlo Ray Tracing. Computer Graphics Forum, Vol. 32, No. 1, pp. 139-151, 2013.

[34] B. Mora. Naïve Ray-Tracing: A Divide-And-Conquer Approach. ACM Transactions on Graphics, Vol. 30, No. 5, pp. 117:1-117:12, 2011.

[35] J. Novak, D. Nowrouzezahrai, C. Dachsbacher, and W. Jarosz. Progressive Virtual Beam Lights. Computer Graphics Forum, Vol. 31, No. 4, pp. 1407-1413, 2012.

[36] J. Ou and F. Pellacini. Lightslice: Matrix Slice Sampling for the Many-Lights

Problem. ACM Transactions on Graphics, Vol. 30, No. 6, pp. 179:1-179:8, 2011.

[37] J. Pantaleoni and D. Luebke. HLBVH: Hierarchical LBVH Construction for Real-Time Ray Tracing of Dynamic Geometry. In Proceedings of the Conference on High Performance Graphics, pp. 87-95.

[38] M. Pharr and G. Humphreys. Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann Publishers, 2010.

[39] S. Popov, I. Georgiev, P. Slusallek, and C. Dachsbacher. Adaptive Quantization Visibility Caching. Computer Graphics Forum, Vol. 32, No. 2, pp. 399-408, 2013.

[40] W. Purgathofer. A Statistical Method for Adaptive Stochastic Sampling. Computer & Graphics, Vol. 11, No. 2, pp. 157-162, 1987.

[41] S. Ravichandran and P. J. Narayanan. Parallel Divide and Conquer Ray Tracing. In Proceedings of SIGGRAPH Asia 2013 Technical Briefs, pp. 30:1-30:4, 2013.

[42] J. Rigau, M. Feixas, and M. Sbert. Refinement Criteria Based on f-Divergences. In Eurographics Workshop on Rendering, pp. 260-269, 2003.

[43] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. ACM Transactions on Graphics, Vol. 27, No. 5, pp. 129:1-129:8, 2008.

[44] S. M. Rubin and T. Whitted. A 3-Dimensional Representation for Fast Rendering of Complex Scenes. SIGGRAPH Computer Graphics, Vol. 14, No. 3, pp. 110-116, 1980.

[45] F. Simon, J. Hanika, and C. Dachsbacher. Rich-VPLs for Improving the Versatility of Many-Light Methods. Computer Graphics Forum, Vol. 34, No. 2, pp. 575-584, 2015.

[46] R. Tamstorf and H. W. Jensen. Adaptive Sampling and Bias Estimation in Path

Tracing. In Proceedings of the Eurographics Workshop on Rendering Techniques, pp. 285-296, 1997.

[47] S. K. Thompson. Sampling, 3rd edition. Wiley, 2012.

[48] Y. Tokuyoshi. Virtual Spherical Gaussian Lights for Real-time Glossy Indirect Illumination. Computer Graphics Forum, Vol. 34, No. 7, pp. 89-98, 2015.

[49] J. Ulbrich, J. Novak, H. Rehfeld, and C. Dachsbacher. Progressive Visibility Caching for Fast Indirect Illumination. In Proceedings of International Workshop on Vision, Modeling, and Visualization, pp.203-210, 2013.

[50] E. Veach. Bidirectional Estimators for Light Transport. In Proceedings of Eurographics Rendering Workshop, pp. 147-162, 1994.

[51] E. Veach. Robust Monte Carlo Methods for Light Transport Simulation. PhD thesis, Stanford University, 1998.

[52] E. Veach and L. J. Guibas. Metropolis Light Transport. In Proceedings of ACM SIGGRAPH, pp. 65-76, 1997.

[53] I. Wald. On Fast Construction of SAH-based Bounding Volume Hierarches. In Proceedings of the Eurographics/IEEE Symposium on Interactive Ray Tracing, pp. 33-40, 2007.

[54] I. Wald, S. Boulos, and P. Shirley. Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies. ACM Transactions on Graphics, Vol. 26, No. 1, pp. 6:1-6:18, 2007.

[55] B. Walter, A. Arbree, K. Bala, and D. P. Greenberg. Multidimensional Lightcuts. ACM Transactions on Graphics, Vol. 25, No. 3, pp. 1081-1088, 2006.

[56] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: A Scalable Approach to Illumination. ACM Transactions on Graphics, Vol. 24, No. 3, pp. 1098-1107, 2005.

[57] B. Walter, P. Khungurn, and K. Bala. Bidirectional Lightcuts. ACM Transactions on Graphics, Vol. 31, No. 4, pp. 59:1-59:11, 2012.

[58] R. Wang and O. Akerlund. Bidirectional Importance Sampling for Unstructured Direct Illumination. Computer Graphics Forum, Vol. 28, No. 2, pp. 269-278, 2009.

[59] T. Whitted. An Improved Illumination Model for Shaded Display. Communications of the ACM, Vol. 23, No. 6, pp. 343-349, 1980.

[60] A. Williams, S. Barrus, R. K. Morley, and P. Shirley. An Efficient and Robust Ray-Box Intersection Algorithm. Journal of Graphics Tools, Vol. 10, No. 1, pp. 49-54, 2005.

[61] Y. T. Wu and Y. Y. Chuang. VisibilityCluster: Average Directional Visibility for Many-Light Rendering. IEEE Transactions on Visualization and Computer Graphics, Vol. 19, No. 9, pp. 1566-1578, 2013.

[62] Z. Wu, F. Zhao, and X. Liu. SAH-KD-Tree Construction on GPU. In Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, pp. 71-78, 2011.

[63] K. Zhou, Q. Hou, R. Wang, and B. Guo. Real-Time KD-Tree Construction on Graphics Hardware. ACM Transactions on Graphics, Vol. 27, No. 5, pp. 126:1-126:11, 2008.

付録

A　可視関数の確率的評価を用いた輝度推定値の分散の導出

可視関数の確率的評価を行う場合の輝度推定値の分散（式(4.10)）の導出について説明する．分散の定義より，確率変数X，Yにおいて，YがXに依存するとき，XYの分散$var[XY]$は

$$var[XY] = \mathbb{E}[var[Y|X]X^2] + var[\mathbb{E}[Y|X]X]$$

で計算される．提案法の場合，Xはt_s/p_sに相当し，Yは$\tilde{V}$に相当する．このとき$\mathbb{E}[Y|X] = v_s$であることから，$var[\mathbb{E}[Y|X]X] = var\left[\hat{L}^{(1)}(x, x_v)\right]$であり，式(4.10)の第 1 項である．一方，$var[Y|X]$は可視関数の推定値の分散であり式(4.9)で表される．これを$\mathbb{E}[var[Y|X]X^2]$に代入すれば式(4.10)の総和計算部分が得られる．

B　統計量Tの導出

式(5.8)で計算される統計量Tの導出を行う．まず，層化サンプリングにおいて，各層のサンプル値は正規分布$\mathcal{N}(\mu_i, \sigma_i^2)$に従うと仮定する．ここで，$\mu_i$は平均，$\sigma_i$は標準偏差である．このとき，$i$番目の層における$n_i$個のサンプル$X_{i1}, \ldots, X_{in_i}$の標本平均$\bar{X}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} X_{ij}$は正規分布$\mathcal{N}(\mu_i, \sigma_i^2/n_i)$に従い，正規分布の再生性より，$N$個の層の標本平均の和$\bar{X} = \sum_{i=1}^{N} \bar{X}_i$も正規分布$\mathcal{N}(\sum_{i=1}^{N}\mu_i, \sum_{i=1}^{N}\sigma_i^2/n_i)$に従う．そうすると，$\bar{X}$を変換した以下の確率変数$Y$は標準正規分布$\mathcal{N}(0,1)$に従う．

$$Y = \frac{(\bar{X} - \mu)}{\sqrt{\sum_{i=1}^{N}\sigma_i^2/n_i}}$$

ここで，$\mu = \sum_{i=1}^{N}\mu_i$である．次に，以下の式で計算される$i$番目の層の不偏分散$u_i^2$を考える．

$$u_i^2 = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}\left(X_{ij} - \bar{X}_i\right)^2$$

このとき，$\frac{1}{\sigma_i^2}\sum_{j=1}^{n_i}\left(X_{ij} - \bar{X}_i\right)^2 = (n_i - 1)u_i^2/\sigma_i^2$は自由度$n_i - 1$の$\chi^2$分布に従い，$\chi^2$分布の再生性より，以下の式で定義される確率変数$Z$は自由度$\sum_{i=1}^{N}(n_i - 1) = n - N$の$\chi^2$分布に従う．

$$Z = \sum_{i=1}^{N}\frac{(n_i - 1)u_i^2}{\sigma_i^2}$$

そうすると，t分布の定義より，$T = Y/\sqrt{Z/(n-N)}$は自由度$n - N$のt分布に従い，YとZを展開すると以下の式が得られる．

$$\begin{aligned} T &= \frac{Y}{\sqrt{\frac{Z}{n-N}}} \\ &= \frac{\frac{\bar{X} - \mu}{\sqrt{\sum_{i=1}^{N}\sigma_i^2/n_i}}}{\sqrt{\frac{\sum_{i=1}^{N}(n_i - 1)u_i^2/\sigma_i^2}{n-N}}} \\ &= \frac{\sqrt{n-N}(\bar{X} - \mu)}{\sqrt{\sum_{i=1}^{N} n_i s_i^2/\sigma_i^2}\sqrt{\sum_{i=1}^{N}\sigma_i^2/n_i}} \end{aligned}$$

ここで，s_i^2は標本分散であり，上式では，$n_i s_i^2 = (n_i - 1)u_i^2$の関係を用いた．最後に，合計サンプル数$n = \sum_{i=1}^{N} n_i$として，各層へのサンプル数の配分が，分散を最小とする理想的な配分であるネイマン配分（標準偏差に比例して配分）であると仮定し，$n_i = \frac{\sigma_i}{\sum_{j=1}^{N}\sigma_j}n$と$\sigma_i = \frac{n_i}{n}\sum_{j=1}^{N}\sigma_j$の関係を用いると，統計量$T$は以下の式に変形される．

$$T = \frac{\sqrt{n-N}(\bar{X}-\mu)}{\sqrt{\sum_{i=1}^{N}\frac{n_i s_i^2}{\sigma_i^2}}\sqrt{\sum_{i=1}^{N}\frac{\sigma_i^2}{n_i}}}$$

$$= \frac{\sqrt{n-N}(\bar{X}-\mu)}{\sqrt{\frac{n^2\sum_{i=1}^{N}\frac{s_i^2}{n_i}}{\left(\sum_{j=1}^{N}\sigma_j\right)^2}}\sqrt{\frac{1}{n}(\sum_{i=1}^{N}\sigma_i)^2}}$$

$$= \sqrt{\frac{n-N}{n}}\frac{\bar{X}-\mu}{\sqrt{\sum_{i=1}^{N}\frac{s_i^2}{n_i}}}$$

提案法の輝度推定の文脈においては, μは出射輝度の真値L, $\bar{X}$は出射輝度の推定値$\hat{L}$であり, 変数を置き換えることで式(5.8)が得られる.

謝辞

本研究の過程において，指導教官として長きに渡る懇切なる御指導，御鞭撻を賜りました和歌山大学システム工学部岩崎慶准教授に心より感謝いたします.

本研究に対し貴重な御意見，御討論を頂きました広島修道大学経済科学部西田友是教授，北海道大学大学院情報科学研究科土橋宜典准教授に心より感謝いたします.

本論文をまとめるにあたり御助言，御指導を頂いた和歌山大学システム工学部和田俊和教授，天野敏之教授に心より感謝いたします.

研究業績

学会誌掲載論文・ジャーナル論文

[1] 名畑豪祐, 岩崎慶, 土橋宜典, 西田友是, 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, 情報処理学会論文誌, Vol. 55, No. 12, pp. 2559-2568, 2014.

[2] K. Nabata, K. Iwasaki, Y. Dobashi, T. Nishita, An Error Estimation Framework for Many-light Rendering, Computer Graphics Forum, Vol. 35, No. 7, pp. 431-439, 2016.

[3] 名畑豪祐, 岩崎慶, 多光源レンダリング法のための画像生成の効率を考慮した可視関数の確率的評価, 情報処理学会論文誌.

[4] H. Yoshida, K. Nabata, K. Iwasaki, Y. Dobashi, T. Nishita, Adaptive Importance Caching for Many-light Rendering, Journal of WSCG, Vol. 23, No. 1, pp. 65-72, 2015.

査読付き国際会議

[1] K. Nabata, K. Iwasaki, Y. Dobashi, T. Nishita, Efficient Divide-And-Conquer Ray Tracing using Ray Sampling, In Proceedings of High Performance Graphics 2013, pp. 129-135, 2013.

[2] H. Sakai, K. Nabata, Y. Yasuaki, K. Iwasaki, Error Estimation for Many-light Rendering with Supersampling, In Proceedings of SIGGRAPH Asia 2018 Technical Briefs, pp. 17:1-17:4, 2018.

査読付き国内会議

[1] 名畑豪祐，岩崎慶，土橋宜典，西田友是，多光源レンダリング法のための効率的な可視関数の確率的評価, Visual Computing/グラフィックスと CAD 合同シンポジウム, 2014.

[2] 名畑豪祐，岩崎慶，土橋宜典，西田友是，VPL サンプリングのための効率的な可視関数の確率的評価，Visual Computing/グラフィックスと CAD 合同シンポジウム，2015.

[3] 酒井広和，名畑豪祐，安明真哉，岩崎慶，スーパーサンプリングを用いた多光源レンダリングのための誤差推定法，Visual Computing/グラフィックスと CAD 合同シンポジウム，2018.

国内会議

[1] 溝口智博，名畑豪祐，岩崎慶，高解像度法線マップを用いたハイライトの効率的レンダリング，画像関連学会連合会秋季大会，2015.

[2] 安明真哉，名畑豪祐，岩崎慶，関与媒質の多光源レンダリングのための誤差推定法，情報処理学会第 79 会全国大会 6X-01，2017.

[3] 岩崎慶，名畑豪祐，関与媒質のレンダリングのための誤差推定法，コンピュータグラフィックスとビジュアル情報学研究会，Vol. 2017-CG-168, No. 8, pp.1-6, 2017.