2009－CG－134　(19)
2009／2／17

# Visibility Cuts を用いた動的シーンの高速レンダリング

宮前雄生　岩崎慶　吉本富士市
和歌山大学システム工学部

写実的な画像を効率的に生成するため，環境マップで表わされる照明を使ったライティング手法の研究が数多くなされている．本研究では環境照明下の動的なシーンにおいて，物体の材質の動的な編集に対応した高速レンダリング手法を提案する．物体の遮蔽情報を自己遮蔽と他の物体による遮蔽に分け，これらをレンダリング時に統合することで動的なシーンに対応する．遮蔽情報には Visibility Cuts と呼ばれる表現を用い，材質の動的な編集に対応する．Visibility Cut を効率的に表現することで前計算データ量を削減し，効率的なレンダリングが可能となった．

# Fast Rendering of Dynamic Scenes using Visibility Cuts

Yuki Miyamae　Kei Iwasaki　Fujiichi Yoshimoto
Faculty of Systems Engineering, Wakayama University

To render photo-realistic images efficiently, many rendering methods using environment map lighting have been proposed. We propose a fast rendering method of dynamic scenes under all-frequency environment lighting with dynamic BRDFs. Our method precomputes the self-occlusion and occlusions due to other objects. By integrating these occlusions in the rendering process, our method can render dynamic scenes efficiently. The occlusion information is represented by using visibility cuts. This makes it possible to change BRDFs in the rendering process. We develop an efficient representation of visibility cuts, and achieved to reduce the precomputed data and efficient rendering.

## 1. はじめに

写実的な画像の生成はコンピュータグラフィックスの分野において重要なテーマの一つである．現実世界の照明をシミュレートするために，これまではシーンに点光源や平行光線，面光源などを配置して照明計算を行っていた．近年，より現実的な照明環境を再現するため，シーンに遠方から入射する照明を画像（環境マップ）として表現する環境照明を用いたレンダリング手法の研究がなされてきた．このような環境照明下のシーンの照明計算は，全方向から入射する光を考慮する必要があるため，リアルタイムにレンダリングすることは難しい問題であった．Sloan らが提案した Precomputed Radiance Transfer(PRT)[4] は環境照明下のシーンのリアルタイムなレンダリングを実現した．PRT 法は後の研究によって改良され，より汎用性の高い多くの手法が提案された．しかし，製品デザインソフトウェアなどにおいて重要な機能である，物体の移動と材質の編集を同時に行うことができる手法はまだ提案されていない．そこで本研究では，環境照明下のシーンにおいて，物体の平行移動に対応し，かつ物体の反射特性を表わす関数である BRDF の動的な編集が可能な高速レンダリング手法を提案する．

## 2. 関連研究

PRT 法[4]は光の伝播を前計算することで，環境照明下のシーンのリアルタイムなレンダリングを実現した．しかし，シーンの物体を動かすことはできず，照明は輝度変化のなだらかな低周波照明に限られている．Ng ら[3]は照明の近似に Wavelet 基底関数を用いて高周波の照明に対応した．Zhou ら[6]は物体の周囲の遮蔽情報をサンプル点で前計算し，それらを統合することで，物体が移動・回転する動的シーンに対応した．しかしこれらの手法では BRDF の前処理をする必要があるため，レンダリング時に BRDF を変更することは難しい．

Akerlund ら[1]が提案した Precomputed Visibility Cuts は，遮蔽情報にカットと呼ばれる表現を用いて BRDF の動的な変更に対応した．Cheslack-Postava ら[2]は複数のカットを効率的に統合するアルゴリズムを提案し，Visibility Cuts のアプローチをピクセル単位のレンダリングに拡張した．しかしこれらの手法は動的なシーンに対応していない．

提案手法では，遮蔽情報の表現に，BRDF の動的な編集が可能な Visibility Cuts を用いる．そして Zhou らの手法のように，物体の周囲の遮蔽情報をサンプル点で前計算することにより，物体の移動に対応する．また，カットのデータの表現を変更することで，従来手法[2]よりもデータ量を削減し，より効率的にカットを統合する．

## 3. 提案手法の概要

本節では提案手法の概要について説明する．本手法で扱う物体は変形しない剛体であり，回転はせず，平行移動のみを行うものとする．また，相互反射は考慮せず，環境マップからの直接光のみを扱う．

視点方向 $\omega_o$ から見た環境照明下の頂点 $x$ の輝度 $B$ は次式で計算される．

$$B(x,\omega_o)=\int_\Omega L(x,\omega)\,f_r(\omega,\omega_o)V(x,\omega)\cos\theta\,d\omega$$
$$=\int_\Omega L(x,\omega)\,f_r(\omega,\omega_o)\tilde{V}(x,\omega)d\omega \qquad (1)$$

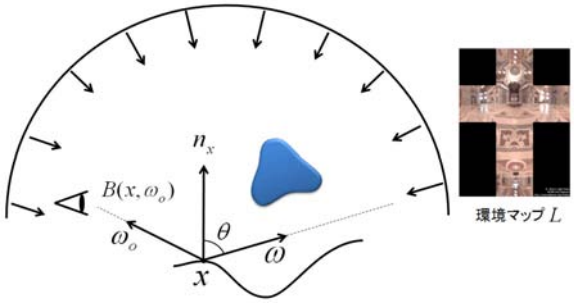


図 1　頂点の輝度計算

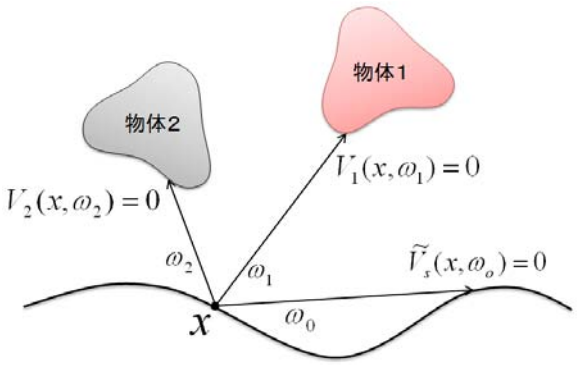


図 2　自己遮蔽 $\tilde{V}_s$ と物体 $n$ による遮蔽 $V_n$

$L$ は環境照明，$f_r$ は BRDF，$\cos\theta$ は頂点 $x$ の法線と方向 $\omega$ のなす角の余弦（負の場合は 0）である(図 1)．$V$ は頂点から方向 $\omega$ にレイを射出したとき，物体によって遮られる場合に 0，それ以外に 1 を返す可視関数である．$\tilde{V}$ は $V$ と $\cos\theta$ の積である．

各物体が平行移動する動的シーンでは，可視関数 $\tilde{V}$ は変化するため，前計算することはできない．そこで可視関数を自己遮蔽と他の物体による遮蔽の積によって以下の式で計算する．

$$\tilde{V}(x,\omega)=\tilde{V}_s(x,\omega)V_1(x,\omega)V_2(x,\omega)\cdots V_n(x,\omega) \qquad (2)$$

$\tilde{V}_s$ は頂点が属する物体の自己遮蔽関数と $\cos\theta$ の積，$V_n$ は物体 $n$ による遮蔽関数である(図 2)．形が変化しない剛体においては，物体の自己遮蔽 $\tilde{V}_s$ と物体が周囲に及ぼす遮蔽 $V_n$ は変化しないため，前計算することができる．全周波の照明に対応するためには，遮蔽関数を数万方向でサンプリングする必要がある．しかし，そのままでは計算量が多くなるため，[1]の手法により，二分木

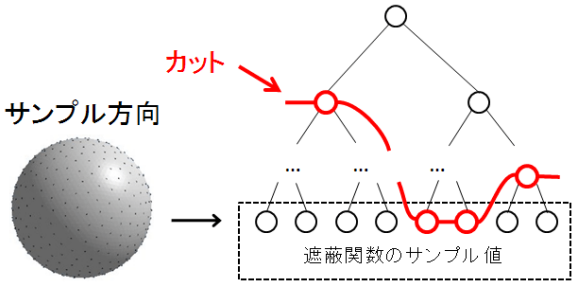


図 3　遮蔽関数のクラスタリング

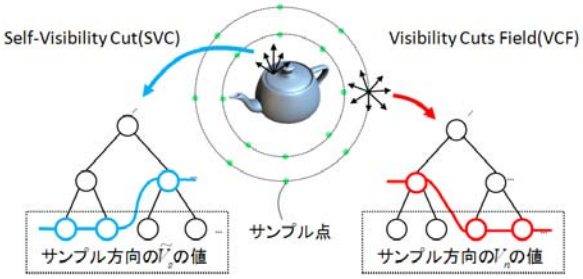


図 4　SVC と VCF

構造を用いて遮蔽関数をクラスタリングする. この時選択されたクラスタ群を, 従来法と同様にカットと呼ぶ(図 3). $\tilde{V}_s$ は物体の各頂点で計算し, クラスタリングする. これを Self-Visibility Cut(SVC)と呼ぶ. $V_n$ は物体の周囲のサンプル点で計算し, クラスタリングする. これを Visibility Cuts Field(VCF)と呼ぶ(図 4). レンダリング時には頂点の SVC と影響範囲内の物体の VCF のカットを統合して, 次式で輝度を計算する.

$$B(x,\omega_o) \approx \sum_k v_k f_r(\omega_k,\omega_o)\int_{\Omega_k} L(x,\omega)d\omega = \sum_k v_k f_r(\omega_k,\omega_o) l_k \qquad (3)$$

$\Omega_k$ は統合されたクラスタ $k$ に対応する立体角, $v_k$ はクラスタ $k$ の $\tilde{V}$ の値, $l_k$ は $\Omega_k$ 内の照明の積分である. $\omega_k$ は $\Omega_k$ 内の平均方向であり, $f_r$ は $\omega_k$ でサンプリングした BRDF である. 本研究では, レンダリング時に BRDF を計算するため動的に変更することが可能である.

## 4. アルゴリズムの詳細

本節ではアルゴリズムの詳細を述べる. まず前計算処理について説明し, 次に新しいカットの表現方法とカット統合アルゴリズムについて述べる. 最後にレンダリング時の処理について説明する.

### 4.1. 前計算

前計算では[1]の手法を用いて遮蔽関数のクラスタリングを行う. クラスタリングに用いる二分木構造では, 葉ノードが個々のサンプル方向を表わし, その他のノードは子ノードの平均方向を表わすクラスタである. 個々のサンプル方向は, 多数の点（32,768 点または 65,536 点）をポイント反発アルゴリズム[5]を用いて単位球上に均一に生成して決定する.

まず物体の各頂点において SVC を計算する. 頂点から全てのサンプル方向へレイトレーシングを行い, $\tilde{V}_s$ の値を二分木の葉ノードに格納する. そして, 親ノードに子ノードの値の平均値 $v_k$, クラスタリングによる遮蔽関数の誤差 $E_k$, まとめられた立体角 $|\Omega_k|$ を順次格納していく. 誤差 $E_k$ は以下の式で計算する.

$$E_k = \frac{1}{|\Omega_k|}\int_{\Omega_k} (v_k - \tilde{V}_s(\omega))^2 d\omega \qquad (4)$$

木を完成したら, 次にクラスタを選択する. まずクラスタの誤差と立体角に対する閾値を設定する. 根ノードから始めて, 全てのノードの誤差と立体角がそれぞれの閾値以下になるまで, ノードを子ノードに分割していく. 最終的に選択されたノード(クラスタ)群がカットである. カットは, 二分木におけるノードの位置と値 $v_k$ のベクトルとして保存する. 以上の処理を全ての頂点で行う.

次に VCF の計算を行う. VCF のカットの計算は, 物体の重心を中心とする同心球上のサンプル点で行う(図 4). 各球上の点の位置はキューブマップ上に格子状に配置した点を球に投影することで求める. これらのサンプル点において $V_n$ の値を計算し, SVC と同様にクラスタリングする. ただし, クラスタ選択時には, 立体角の制約をなくし, 誤差の閾値を 0.25 より小さくとることで, $V_n$ の値が同じ領域のみを可能な限りクラスタリ

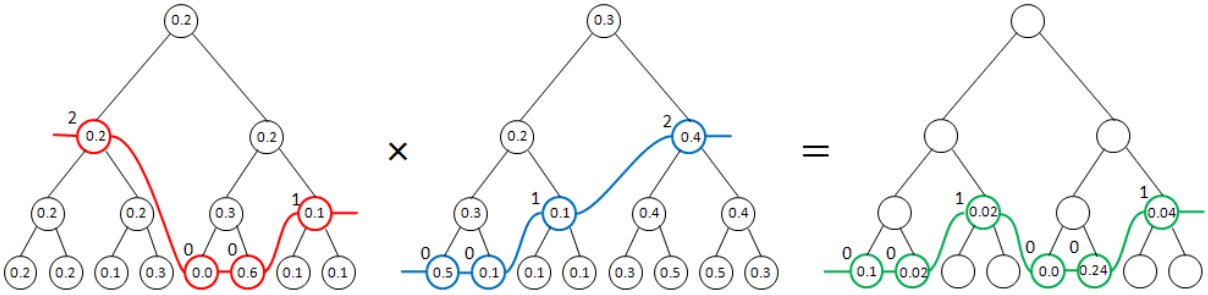

図 5　カットの積

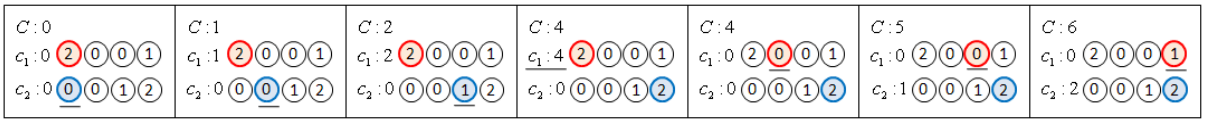

図 6　カットの走査

ングする. これにより遮蔽情報の精度を落とさずにノードの数（カットサイズ）を減らし，データ量を抑えることができる. 全てのサンプル点で同様の計算を行い，カットを保存しておく.

### 4.2. カットの表現

従来手法[1,2]では, 二分木におけるカットノードの位置を表わすために, ノードに一意に割り当てられたインデックスを用いている. しかし，カットノードは, 根ノードからそれぞれの葉ノードまでの経路上に唯一存在するという性質を持ち, カットを先頭のノードから走査すれば, ノードの高さ（葉ノードからそのノードまでの枝の数）のみでその位置を特定することができる. そこで提案手法では, インデックスの代わりに二分木におけるノードの高さを保存し, データ量を削減している. 例えばサンプル点数が 32,768 の場合，二分木のインデックスには 2 バイトを要するが, 高さは 4 ビットで表わすことができる.

### 4.3. カットの統合

ここではカットの積や補間を計算するカット統合アルゴリズムについて説明する. 従来法[2]の統合アルゴリズムでは, カットノードの位置をインデックスで表わしていたが, 本手法ではノードの高さを使用する.

カットの積を例に, 求める統合結果を図 5 に示す. ノードの左肩の数字は，ノードの高さを表わす. 統合後のカットは，統合されるカットのうちで最も低い位置を通るカットになる. アルゴリズムの基本的なアイデアは, 全てのカットのノードが重なった（先祖と子孫の関係にある）状態を維持するように, 最も高さが低いノードを進めながら演算を行うことである. 処理の手順は次のようになる.

1. 全体のカウンタ $C$ と, それぞれのカットのカウンタ $c_k$ を用意し，0 に初期化する.
2. 各々のカットの先頭のノードに着目する.
3. 演算を行い結果を格納する. 結果のノードの高さは着目したノードの内で最も低い高さ $h_{\min}$ になる. インデックスは根ノードの高さを $H$ として $2^{(H-h_{\min})}+C/2^{h_{\min}}-1$ となる.
4. 最も高さの低いノード $i$ を進め, $c_i=0$ とする. その他のカットのカウンタ $c_k$ と全体のカウンタ $C$ に $2^{h_{\min}}$ を足す.
5. 各々のノードの高さを $h_k$ として，$c_k \geq 2^{h_k}$ となったノードを進め，$c_k=0$ とする.
6. 全てのカットが最後のノードに達するまで 3~5 を繰り返す.

図 5 のカットを統合する場合のカットの走査の様子を図 6 に示す. ノードの中の数字はノードの高さを表わし, 下線は最も高さの低いノードを示す. 従来法ではノードを重ねるために, まず着目した全てのノードが重なっているか

を調べ, 重なっていない場合には適切なノードを進めるという処理を行っている. しかし一度でノードが重なるという保証はなく, ノードが重なるまでこの処理を繰り返す必要がある. 一方提案手法では, 常にノードが重なった状態を維持することができるため, より効率的にカットを走査することができる.

SVC と VCF のカットを統合して頂点色計算を行う場合は, 次節で説明するライトツリーを効率的に参照するために, 統合後のカットノードのインデックスが必要になる. その場合には, 手順 3 に示す計算でレベル順のインデックスを得ることができる. また, SVC のノードの値が 0 であれば, その子孫の部分は照明計算に影響しないため, すぐにそのノードを進めて処理を省略することができる. この場合にもカウンタの値を参照することで, 他のノードを適切に進めることができる.

### 4.4. レンダリング

レンダリング時にはまず照明をサンプリングし, 前計算で使用した二分木構造を使って照明の木を構築する. まず環境マップをサンプル点に投影し, その色を葉ノードに格納する. そして親ノードに二つの子ノードの色の和を順次格納していき, 木を完成する. これをライトツリーと呼ぶ.

次に頂点色の計算を行う. まず頂点の可視関数を計算するため, SVC と VCF のカットを統合する. 頂点に影響する物体の VCF から, 頂点の近傍 8 サンプル点を求め, サンプル点におけるカットを補間する. 影響を受ける物体ごとに補間したカットを求め, それらと頂点の SVC の積を計算し, その頂点のカットとする(図 7). 次に統合したカットのノードのインデックスでライトツリーを参照し, 照明の色を得る. そして, クラスタの平均方向と視点方向を利用して BRDF を直接評価する. クラスタの遮蔽関数の値, 照明の色, BRDF の積を全てのカットノードで足し合わせ, 頂点色とする. 最後に求めた頂点色でポリゴンを描画し, シーンをレンダリングする.

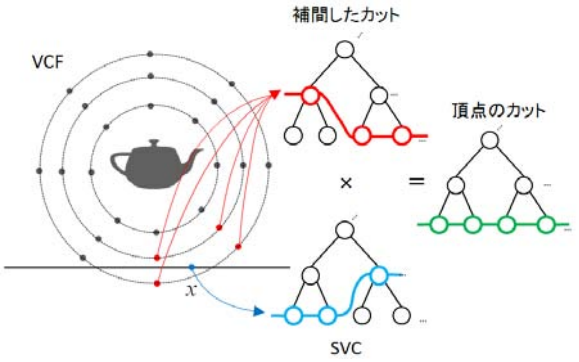

図 7　SVC と VCF の統合

## 5. 結果

本節では, 提案手法により得られた結果を示す. 計算環境は Intel Core2 Quad CPU Q9650 3.00GHz, 3GB RAM, GeForce GTX 280 を搭載した PC である. 全てのシーンにおいて, 照明サンプル点数は 32,768 とした. 各物体の VCF には, 境界球の半径を $r$ として, 物体の中心から $0.2r$〜$10r$ までに 32 個の球を等間隔に配置し, 各球上に 6×32×32 個, 合計 196,608 個のサンプル点を用いた. 図 8 は物体をレンダリング時に平行移動させた結果画像である. 図 9 は bunny の BRDF(Blinn-phong)の指数パラメータ $\alpha$ をレンダリング時に変化させたものである. 各シーンのデータを表 1 に示す. データ量は SVC と VCF のデータ量合計である. 描画時間は, 左が BRDF の編集, 視点位置・照明の変更時の描画時間であり, 右が物体移動時の描画時間である. 物体の移動の計算には数秒を要しているが, BRDF の編集, 視点位置・照明の変更はインタラクティブに行うことができる. これは, 物体の移動時に統合したカットを保存し, BRDF 編集時などには保存したカットを用いて頂点色計算のみを行っているからである. データ量と描画時間は, 従来手法[2]と[6]を組み合わせた方法と, 提案手法の両方を示している. teapot シーン(図 8)では提案手法の方が約 64%データ量が少なく, カットの統合も約 12%高速であることがわかる. bunny シーン(図 9)では約 53%データ量が少なく, カットの統合時間はほぼ同じである.

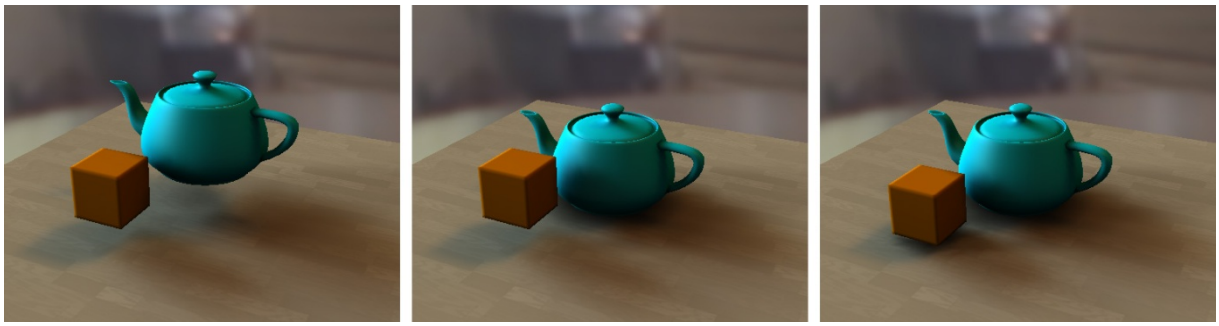

図 8　物体の平行移動

(a)拡散反射のみ　　(b) $\alpha$ =15　　(c) $\alpha$ =70

図 9　Blinn-Phong の指数パラメータの編集

表 1　各シーンの結果データ

| | | | 従来法[2]と[6]の組み合わせ | | 提案法 | |
|---|---|---|---|---|---|---|
| シーン | 頂点数 | 物体数 | データ量 | 描画時間 | データ量 | 描画時間 |
| teapot(図 8) | 11,565 | 3 | 266MB | 0.13/2.03[s] | 97MB | 0.13/1.79[s] |
| bunny(図 9) | 40,172 | 2 | 251MB | 0.50/3.41[s] | 117MB | 0.50/3.36[s] |

## 6. まとめ

本研究では, 全周波環境照明下のシーンにおいて物体の平行移動に対応し，かつ BRDF の動的な編集が可能な高速レンダリング手法を提案した．また，カットノードの表現に二分木における高さを用いることでデータ量を削減し, 効率的なカット統合アルゴリズムを提案した. 現在 BRDF の編集はインタラクティブに行うことができるが, 物体の移動はインタラクティブな速度には達していない．今後，GPU による頂点色計算の高速化により, 物体の移動もインタラクティブに行うことができるようになると考えられる.

## 参考文献


[1]O. Akerlund et al., Precomputed visibility cuts for interactive relighting with dynamic brdfs, Pacific Graphics, pp. 161-170, 2007.

[2]E. Cheslack-Postava et al., Fast, realistic lighting and material design using nonlinear cut approximation, ACM Transactions on Graphics, Vol. 27, No. 5, Article 128, 2008.

[3]R. Ng et al., All-frequency shadows using non-linear wavelet lighting approximation, ACM Transactions on Graphics, Vol. 22, No. 3, pp. 376-381, 2003.

[4]P. Sloan et al., Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments, ACM Transactions on Graphics, Vol. 21, No. 3, pp. 527-536, 2002.

[5]G. Turk, Generating textures on arbitrary surfaces using reaction-diffusion, Computer Graphics, Vol. 25, No. 4, pp. 289-298, 1991.

[6]K. Zhou et al., Precomputed shadow fields for dynamic scenes, ACM Transactions on Graphics, Vol. 24, No. 3, pp. 1196-1201, 2005.