Vol.2019-CG-176 No.8
Vol.2019-DCC-23 No.8
Vol.2019-CVIM-219 No.8
2019/11/7

バイナリ空間分割木を用いた多角形光源による影の計算

奥野弘貴[1]　岩崎慶[1]

概要：本稿では，多角形光源による影を考慮した，物理則に基づいたレンダリング手法を提案する．多角形光源による直接光の計算では，入射輝度，双方向反射率分布関数（BRDF），可視関数の三つを多角形領域について積分する必要があり，この計算には高い計算コストが要求される．多角形光源によるリアルタイムレンダリングを実現するための手法として，多角形光源の辺に沿った境界積分による解析的な手法が存在するが，可視性を考慮していないため影をレンダリングすることができない．そこで，提案法は事前計算された可視関数をクラスタリングによる階層構造で表現することで，境界積分の閉形式を維持しつつ解析的なレンダリングを可能にする．提案法では多角形光源をシェーディング点から可視となる領域に細分化する．実験結果により，GGX BRDF を用いて多角形光源による複雑な影を考慮したレンダリングがインタラクティブな速度でおこなえていることを示す．

キーワード：多角形光源，影，可視関数，GGX BRDF

Interactive Rendering of Shadows for Polygonal Light using a Binary Space Partitioning Visibility Tree

HIROKI OKUNO[†1]　KEI IWASAKI[†1]

1. はじめに

多角形光源によるインタラクティブな物理ベースレンダリングは重要な研究の一つとなっている．特に影はシーン内の物体同士の空間的関係を表現することができ，リアリティのあるレンダリング結果を得るための重要な役割を果たしている．近年，多角形光源を用いた手法[1]や球光源を用いた手法[2]など，面光源によるレンダリング手法が数多く研究されているが，面光源による効率的な影の表現がおこなえる物理ベースレンダリングは未だ大きな課題の一つとなっている．これは，面光源からの入射輝度，双方向反射率分布関数（BRDF），可視関数の三つの積の積分計算が非常に高コストになってしまうためである．

多角形光源を扱っている従来法では，光源が方向によらず一定の光を放出していると仮定し，可視性を無視することで三つの積の積分計算を，余弦で重み付けされた BRDF の多角形領域における積分計算に置き換えることで，多角形光源からの寄与を解析的に計算している．余弦で重み付けされた BRDF は，多角形の辺に沿った境界積分による閉形式が与えられた球面分布で近似され，リアルタイムレンダリングを実現している．しかし，影をレンダリングするためには可視性を考慮する必要があり，多角形光源がシェーディング点から可視となるような領域を正確かつ高速に求めることは非常に難しい．代わりにモンテカルロ積分によるレンダリングをおこなうことが考えられるが，リアルタイムなパフォーマンスを維持するためには少数のシャドウレイで可視性を推定する必要があり，ノイズの発生に繋がってしまう恐れがある．

そこで我々は事前に計算された可視関数をクラスタリングすることで多角形光源による影のレンダリングをおこなう手法を提案する．提案法では境界積分による解析的な計算を維持し，図 1 に示すように GGX BRDF を用いて全ての周波数の影をレンダリングすることが可能である（例えば，光沢度が強い表面ではハードシャドウとなり，粗い表面ではソフトシャドウとなる）．また，ユーザはインタラクティブに多角形光源を平行移動，回転させることが可能で，GGX BRDF のパラメータや視点位置も変更可能である．

2. 関連研究

2.1 多角形光源レンダリング

多角形光源による直接照明の計算には，入射輝度と BRDF の積を多角形領域について積分する必要がある．積分値を確率的に推定する方法としてモンテカルロ積分が挙げられるが，サンプリング数が少ない場合レンダリング結果に多くのノイズが発生してしまう．この問題を解決するために，多角形領域における積分を解析的に計算する方法[3]が提案されている．Heitz ら[1]は多角形光源下で GGX BRDF を用いてリアルタイムシェーディングを行うための線形変換された余弦分布（LTC）を提案した．この手法は GGX BRDF を LTC で近似表現することで多角形光源からの寄与を解析的に計算することを可能にしたが，可視性を考慮していないため影のレンダリングをおこなうことができない．その後 Heitz ら[4]は可視性を確率論的なアプロー

1 和歌山大学
Wakayama University

Vol.2019-CG-176 No.8
Vol.2019-DCC-23 No.8
Vol.2019-CVIM-219 No.8
2019/11/7

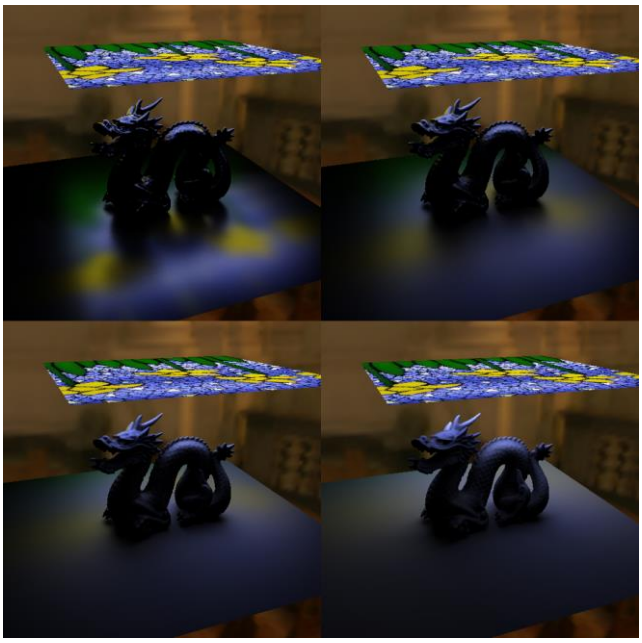

図 1　提案法による dragon のレンダリング結果．
提案法では GGX BRDF を用いたテクスチャライトによるインタラクティブなレンダリングが可能である．左上から右下にかけて，GGX BRDF の粗さパラメータαをそれぞれ 0.1，0.25，0.5，1.0 に変更した結果である．

チで推定することによって影をレンダリングする手法を提案したが，この手法ではレンダリング結果にノイズが発生するためデノイズ処理が必要となる．

近年では球面調和関数を多角形領域について解析的に積分することで，多角形光源による効率的なレンダリングをおこなう手法が提案された[5][6]．しかし，球面調和関数は高周波データを近似表現するために高次の基底関数が必要となるため，インタラクティブな速度でハードシャドウや鋭い光沢反射を伴うシーンをレンダリングすることが難しい．

2.2　直線を用いた可視関数の表現

我々の手法と同様に，符号付き距離関数で可視関数を表現する手法がいくつか提案されている[7][8]が，これらの手法は基本的に面光源に対応していない．Nowrouzezahrai ら[9]は，可視性が変化する場所を表す visibility silhouette と呼ばれるもので可視関数を表現し，環境照明下での放射輝度を半解析的に計算する手法を提案した．Ho ら[10]は vectorized visibility function と呼ばれる可視関数の表現方法を提案した．この手法では三次元座標を繋ぎ合わせることによって得られるベクトルの集合により可視関数を表現している．しかしながらこれらの手法は環境マップライティングのための手法であり，多角形光源によるレンダリングには対応していない．

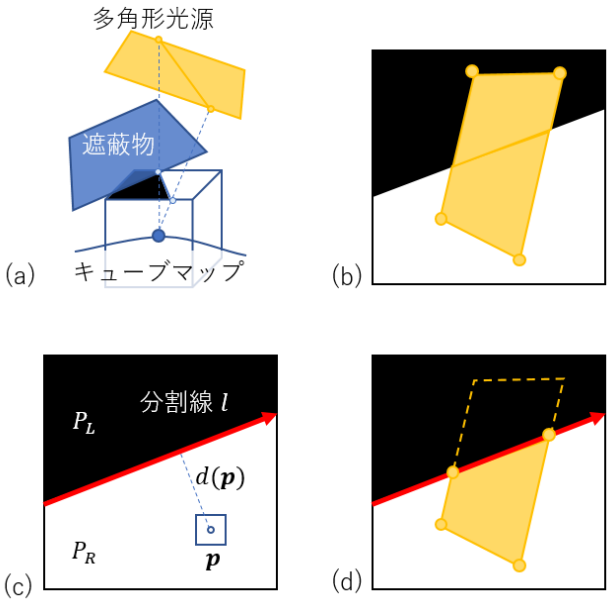

図 2　提案法の基本的なアイデア．
提案法では(c)に示すように，可視関数の可視領域と不可視領域の境界を二次元平面上の直線によって表現する．また，(d)に示すようにキューブマップ上に投影された多角形光源をその直線により分割することで可視の多角形領域を得ることができる．

3.　提案手法

3.1　基本的なアイデア

多角形光源による出射輝度Bは以下の式で計算される．

$$B(\boldsymbol{\omega}_o) = \int_{\Omega} L(\boldsymbol{\omega}_i) V(\boldsymbol{\omega}_i) \tilde{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) d\boldsymbol{\omega}_i \qquad (1)$$

ここで，Ωはシェーディング点から多角形光源へと向かう方向の集合を表し，$\boldsymbol{\omega}_i, \boldsymbol{\omega}_o$はそれぞれ入射方向，出射方向，$L(\boldsymbol{\omega}_i)$は$\boldsymbol{\omega}_i$方向からの入射輝度を表す．計算の簡単化のために，多角形光源が方向によらず一定の光を放つと仮定することでLが定数値となり積分の外に出すことができる（ただし提案法では，図 1 に示されるようにテクスチャライトのような入射輝度が方向に依存する光源を扱うことができる）．Vは可視関数を表し，$\boldsymbol{\omega}_i$方向に遮蔽物が存在する場合 0 を返し，それ以外の場合は 1 を返す．$\tilde{f}_r$は余弦で重み付けされた BRDF である．

提案法では静的なシーンを想定し，シーン中の各頂点$\boldsymbol{x}$で可視関数Vを事前に計算し，キューブマップに格納する．その際，キューブマップ上の各面に格納されている二次元の可視関数を Binary Space Partitioning Visibility Tree（BSPVT）と呼ばれるバイナリ空間分割木で表現する．BSPVT は図 2 に示すような直線を用いて，キューブマップ上の各面に格納されているピクセル値を類似（又は同じ）した値の集合

Vol.2019-CG-176 No.8
Vol.2019-DCC-23 No.8
Vol.2019-CVIM-219 No.8
2019/11/7

となるように分割をおこなうことで得られる二分木である．図 2 に示すように，キューブマップの各面において，直線を用いることで，多角形光源を可視領域，不可視領域に分割することができる．これにより出射輝度Bは分割された各多角形領域の可視関数の平均値と，各多角形領域における余弦で重み付けされた BRDF の積分値との積和で求めることができ，以下の式で表される．

$$B(\boldsymbol{\omega_o}) = L\int_{\Omega} V(\boldsymbol{\omega}_i)\tilde{f_r}(\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)d\boldsymbol{\omega}_i \approx L\sum_i \bar{v}_i \int_{\Omega_i} \tilde{f_r}(\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)d\boldsymbol{\omega}_i \quad (2)$$

式(2)におけるΩ_iは分割された多角形領域の集合を表し，$\bar{v}_i$は各多角形領域の可視関数の平均値である．また，余弦で重み付けされた BRDF の積分計算は LTC[1]を用いることで解析的に計算することができる．

3.2 Binary Space Partitioning Visibility Tree

提案法ではキューブマップに格納されている可視関数を BSPVT で表現する．BSPVT はキューブマップの各面に格納されているピクセル値を似た（又は同じ）値をもつピクセルの集合にクラスタリングすることで得られる二分木である．BSPVT の各内部ノードは図 2 に示すように，ピクセルの集合Pを二つの領域P_L, P_Rに分割するような直線のパラメータを保持している．P_L内のピクセルは直線から負の符号付き距離を持ち，P_R内のピクセルは 0 以上の符号付き距離を持つ．BSPVT の各葉ノードは，クラスタ内のピクセルの可視関数の値が等しいとみなされる領域に対応する．

BSPVT の構築方法をアルゴリズム 1 に示す．提案法では類似したピクセルの集合が得られるように，直線によりピクセルの集合Pを二つの領域P_L, P_Rに分割する．そのような直線を求めるために誤差関数$E(P,l)$を導入する．lは直線を表し，誤差関数$E(P,l)$の値が最小となるようなlを求める．誤差関数$E(P,l)$は，ピクセルの集合Pの各ピクセルpに格納されている可視関数の値$V(p)$と直線lによって以下の式で定義される．

$$E(P,l) = \sum_{p\in P}\left(H_0(d(p)) - V(p)\right)^2 \quad (3)$$

ここで，$H_0(x)$はヘヴィサイド関数を表し，$x \geq 0$のとき 1 を返し，それ以外のときは 0 を返す関数である．また，$d(p)$は直線lからピクセルpの中心座標までの符号付き距離を表す．図 2(b)に示すように，P_L内のピクセルpは$H_0(d(p))$の値が 0 となり，P_R内のピクセルでは 1 となる．直線lは，最適化アルゴリズムである Nelder-Mead 法により誤差関数$E(P,l)$の値を最小化することで得られる．また，誤差関数の値がユーザの指定した閾値ϵ以下になった場合類似したピクセルの集合が得られたと判断し，BSPVT の葉ノードとする．葉ノードにはクラスタ内の可視関数の平均値$\bar{v} = \frac{1}{|P|}\sum_{p\in P} V(p)$を格納する．

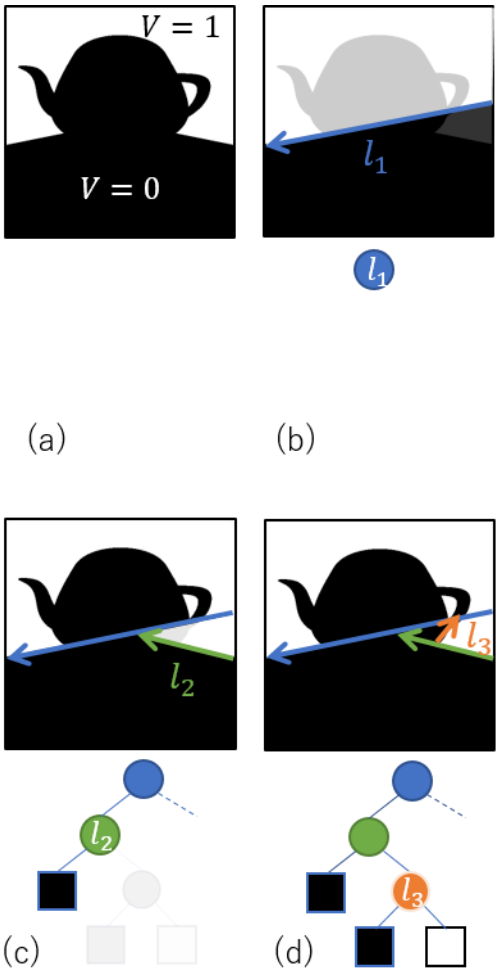

図 3　BSPVT の構築方法．
(a)の上段はキューブマップ上のある面の可視関数を表している．(b)から(d)に示すように可視関数の値は直線での分割により定義される．図の下段に示す色付きの円は BSPVT の内部ノードを表し，白黒で示す四角形は葉ノードを表している．黒い四角は$V = 0$の領域を表し，白い四角は$V = 1$の領域を表している．

アルゴリズム1　BSPVTの構築アルゴリズム．提案法ではキューブマップの各面に存在する全てのピクセルの集合$\mathcal{P}$を直線によりクラスタリングする．各クラスタCにはピクセルの集合Pとクラスタ内誤差eが格納されている．

1: キューの初期化　$Q \leftarrow \{\mathcal{P}, \infty\}$
2: **while** $\forall e > \epsilon$ in Q **do**
3:　Qから最も誤差eの値が大きいクラスタ$C = \{P, e\}$を取り出す
4:　誤差関数$E(P,l)$の値が最小となる直線lを求める
5:　直線lによりPをP_L, P_Rの二つの領域に分割する
6:　クラスタ$C_L = \{P_L, E(P_L,l)\}, C_R = \{P_R, E(P_R,l)\}$を$Q$にプッシュする

図 4 は dragon のモデルを使用して BSPVT を構築した際の元の可視関数との比較結果を表している．提案法では dragon のような曲線が多く見られる複雑な可視関数であっても BSPVT によって高精度に表現できている．

Vol.2019-CG-176 No.8
Vol.2019-DCC-23 No.8
Vol.2019-CVIM-219 No.8
2019/11/7

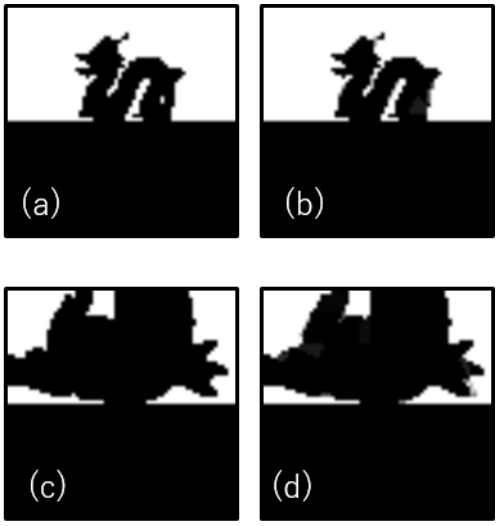

図 4　dragon のモデルによる可視関数の可視化図．
(a)(c)は事前計算時にキューブマップに格納された元の可視関数を表し，(b)(d)は BSPVT により定義された可視関数を表す．提案法では可視関数の複雑な境界（例えば曲線部分など）を精度よく表現することができることを示している．

3.3 BSPVT を用いた多角形光源レンダリング

シーン中の各頂点において多角形光源をキューブマップに投影し，投影された多角形領域を BSPVT の各ノードに格納されている直線により分割する．図 5 は BSPVT の探索の様子を表し，多角形光源の分割処理の流れを示している．図 5(a)の下段は BSPVT を表し，上段は BSPVT により定義された可視関数を表している．提案法では図 5(b)のように多角形光源を各頂点におけるキューブマップの面に投影し，多角形領域を分割する．多角形光源は各内部ノード（図 5(a)から(d)の下段に示す色付けされた円）に格納されている直線により分割される．図 5(c)のように葉ノードにおけるクラスタ内の可視関数の平均値$\bar{v}$が 0 でない場合，分割後の多角形領域P_1について BRDF の積分計算をおこなう．また，図 5(d)に示すように$\bar{v}=0$となるような葉ノードに到達した場合，分割後の多角形領域P_2についての積分計算はおこなわずに破棄する．

4. 実験結果

図 1 はテクスチャライトによる Stanford Dragon のレンダリング結果である．提案法では Heitz ら[1]の手法と同様にテクスチャを事前にフィルタリングしておくことでテクスチャライトによるレンダリングが可能となっている．現在は CPU によるマルチスレッドでの実装となっており，実験に使用した PC は Intel Xeon E5-2650 v4 2.20GHz CPU である．表 1 はモデルの頂点数，レンダリング速度，BSPVT の事前計算時間，BSPVT のデータサイズ，頂点あたりの葉

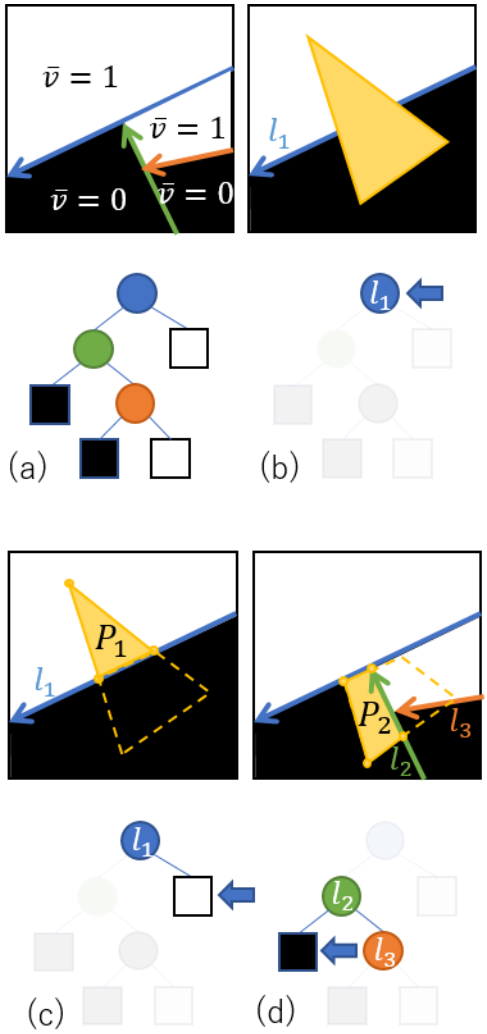

図 5　BSPVT の探索の流れ．
上の行はキューブマップに投影された多角形光源（オレンジ色の三角形）を BSPVT により領域分割をおこなう様子を表しており，下の行は BSPVT の対応するノード（青い矢印で示したノード）を表している．

ノードの平均数を示している．提案法ではインタラクティブなレンダリング速度（1.8fps~43.5fps）で多角形光源による影のレンダリングを実現している．本資料の全てのシーンにおいてレンダリング画像の解像度は 512×512 で，キューブマップに保存された可視関数Vの解像度は 64×64 となっている．

図 6 はテクスチャライトを用いて Teapot のシーンをレンダリングした結果である．左上から右下にかけて，GGX BRDF の粗さパラメータαの値を 0.1，0.25，0.5，1.0 と変化させた結果を示している．提案法ではハードシャドウ（左上）からソフトシャドウ（右下）まで全ての周波数の影を表現することができる．図 7 は Stanford Bunny を用いて多角形光源を移動，回転させてレンダリングをおこなった結果を示している．

図 8 は提案法（左）と，GGX BRDF を用いてレイトレー

Vol.2019-CG-176 No.8
Vol.2019-DCC-23 No.8
Vol.2019-CVIM-219 No.8
2019/11/7

表 1　提案法の詳細データ

	図 1	図 6	図 7
頂点数	506K	17K	138K
レンダリング速度	2.5fps	43.5fps	1.8-10.0fps
事前計算時間	14.4h	2.7min	1.2h
データサイズ	837.8MB	23.1MB	177.0MB
頂点あたりの葉ノードの数	127.9	105.8	99.9

図 6　teapot によるレンダリング結果.
左上から右下にかけて GGX BRDF の粗さパラメータαを
それぞれ 0.1，0.25，0.5，1.0 に変化させている.

シングにより作成された参照画像（中央）との比較結果を示している．一番右の画像は提案法と参照画像との絶対誤差を可視化したものである．図 8 に示すように，提案法によるレンダリング結果は参照画像に匹敵する精度でのレンダリングが可能である．また，提案法と参照画像との誤差は二つの近似によるものである（LTC による BRDF の近似と BSPVT による可視関数の近似）．そこで，BSPVT による近似誤差を取り除き，可視関数を無視して LTC のみでレンダリングされた画像と，GGX BRDF を用いてレイトレーシングによって作成された参照画像を比較する．図 8 の下段の画像は可視性を無視した参照画像と LTC によるレンダリング結果の比較を表している．誤差画像を見てもわかるとおり，絶対誤差 8%以上のピクセル（特に dragon の背後

図 7　bunny によるレンダリング結果.
多角形光源を移動，回転させてレンダリングを行った結果を示している.

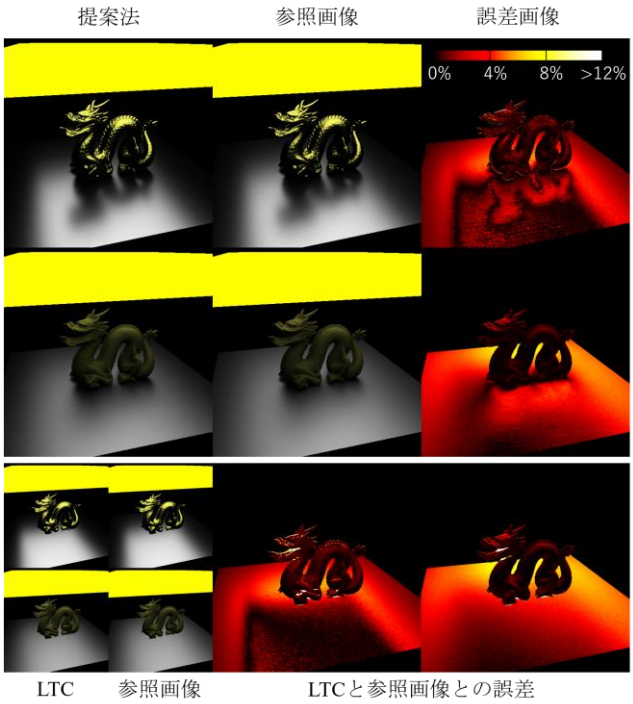

図 8　提案法と参照画像との比較結果.
上段の左が提案法によるレンダリング結果，中央がレイトレーシングにより作成された参照画像，右が提案法と参照画像との誤差を可視化した誤差画像である．また，下段に示しているのは可視性を無視し，LTC のみによるレンダリング結果と，可視性を無視して作成された参照画像との比較結果である.

Vol.2019-CG-176 No.8
Vol.2019-DCC-23 No.8
Vol.2019-CVIM-219 No.8
2019/11/7

の領域）は LTC による BRDF の近似が原因であり，BSPVT による可視関数の近似によって発生する誤差は比較的小さいと考えられる．

5. まとめ

本稿では，多角形光源レンダリングにおける効率的な影の計算の一手法を提案した．可視関数を木構造により階層的に表現する BSPVT を導入し，多角形光源を可視領域と不可視領域に効率的に分割することが可能となった．提案法は，多角形光源によるあらゆる周波数の影をインタラクティブな速度でレンダリング可能であること示した．

提案法は現在頂点単位で影の計算をおこなっており，シーン中の物体が十分に細分化されていない場合はエイリアシングが発生してしまう恐れがある．この問題を解決するために，各頂点で格納された BSPVT を補間することによって，フラグメント単位で影をレンダリングする手法を考える必要がある．また，現在の実装は CPU による実装のため，GPU による実装をおこなうことで更なる高速化が見込まれる．

参考文献

[1] Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. Real-time Polygonal-light Shading with Linearly Transformed Cosines. *ACM Transactions on Graphics* 35, 4(2016), 41:1-41:8.

[2] Jonathan Dupuy, Eric Heitz, and Laurent Belcour. A Spherical Cap Preserving Parameterization for Spherical Distributions. *ACM Transactions on Graphics* 36, 4(2017), 139:1-139:12.

[3] James Arvo. Applications of Irradiance Tensors to the Simulation of non-Lambertian Phenomena. In *Proc. of SIGGRAPH'95.* 335-342.

[4] Eric Heitz, Stephen Hill, and Morgan McGuire. Combining Analytic Direct Illumination and Stochastic Shadows. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games(I3D '18).* 2:1-2:11.

[5] JingwenWang and Ravi Ramamoorthi, Analytic Spherical Harmonic Coefficients for Polygonal Area Lights. ACM Transactions on Graphics 37, 4 (2018), 54:1–54:11.

[6] Laurent Belcour, Guofu Xie, Christophe Hery, Mark Meyer,Wojciech Jarosz, and Derek Nowrouzezahrai. 2018. Integrating Clipped Spherical Harmonics Expansions. ACM Transactions on Graphics 37, 2(2018), 19:1-19:12.

[7] Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. All-Frequency Rendering of Dynamic, Spatially-Varying Reflectance. *ACM Transactions on Graphics* 28, 5(2009), 133:1-133:10.

[8] Kun Xu, Yun-Tao Jia, Hongbo Fu, Shi-Min Hu, and Chiew-Lan Tai. Spherical Piecewise Constant Basis Functions for All-Frequency Precomputed Radiance Transfer. *IEEE Transaction on Visualization and Computer Graphics* 14, 2(2008), 454-467.

[9] Derek Nowrouzezahrai, Ilya Baran, Kenny Mitchell, and Wojciech Jarosz. Visibility Silhouettes for Semi-Analytic Spherical Integration. *Computer Graphics Forum* 33, 1(2014), 105-117.

[10] Tze-Yiu Ho, Yi Xiao, Rui-Bin Feng, Chi-Sing Leung, and Tien-Tsin Wong, All-Frequency Direct Illumination with Vectorized Visibility. *IEEE Transactions on Visualization and Computer Graphics* 21, 8(2015), 945-958.