4Z-3

# 球面ガウス関数を用いた全周波環境照明下の動的シーンの高速レンダリング

古家 亙† 岩崎 慶‡

和歌山大学大学院 システム工学研究科 † 和歌山大学 システム工学部‡

## 1. はじめに

実世界の複雑な照明環境を用いて、写実的でありながら効率的にシーンをレンダリングする手法の研究が数多くなされてきた。Sloan らの提案した Pre-computed Radiance Transfer(PRT)[1]は、大域照明計算における高コストで複雑な計算を事前に算出しておくことにより、写実的なシーンを高速にレンダリングすることに成功した。しかし、その対象は静的なシーンに限定されており、動的なシーンには対応できていなかった。その後、より汎用的なアルゴリズムにするべく PRT の分野で多くの手法[3,4]が研究されてきている。Wang らにより提案された、照明を近似する基底関数として球面ガウス関数を用いる全周波照明環境下でのリアルタイムな照明計算手法[2]は、動的な光源、視点、さらに、物体表面の反射特性である BRDF の変更を可能にした。しかしながらこの方法では、静的なシーンを仮定しており，シーン内の物体の動的な位置の変更には対応していない。 そこで本研究は、球面ガウス関数で表現された環境照明下における影を、デプスマップを用いて高速に計算する手法を提案する。本研究では前計算を必要としないため、全周波照明環境下の動的なシーンにおける高速な照明計算が可能となる。

## 2. 関連研究

PRT[1]は光の伝搬を前計算することにより、直接照明、間接照明、影等の複雑な照明効果を高速にレンダリングする手法である。しかしながら、この方法は当初、静的なシーンに限定されたものであった。その後、Zhou らが導入した Pre-computed Shadow Fields(PSF)[3]により、基底関数に wavelet や spherical harmonic(SH)を使用した PRT 手法において動的なシーンをレンダリングできるようになった。しかしながらこの方法は変形物体には対応していない。また、Ren ら[4]は物体を球によって近似することにより剛体だけでなく変形物体にも PRT を適用可能にしたが、この方法では低周波環境照明しか考慮していない。

本研究では、全周波環境照明下の動的シーンの高速レンダリング手法を提案する。

## 3. 提案手法

シーン内の物体上にある位置xにおける直接照明の輝度$R$は次式で計算される。

$$R(\boldsymbol{x},\boldsymbol{o}) = \int_{\Omega} L(\boldsymbol{i})V(\boldsymbol{i})\rho_{\mathbf{o}}(\boldsymbol{i})\cos\theta\, d\boldsymbol{i} \quad \text{(1)}$$

ここで$L$は照明、$V$は可視関数、$\rho_{\mathbf{o}}$は BRDF、$\cos\theta$は位置$\boldsymbol{x}$の法線と入射ベクトル$\boldsymbol{i}$のなす角の余弦（負の場合は 0)、

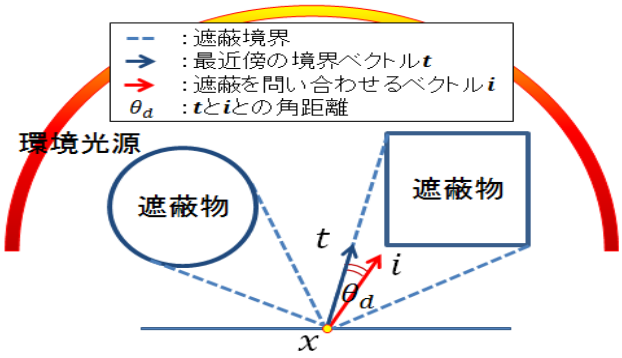


図 1 SSDF で使用する角距離$\theta_d$の決定

$\mathbf{o}$は出射ベクトルである。また、$\Omega$は単位半球上の方向を表している。BRDF$\rho_{\mathbf{o}}$や余弦$\cos\theta$、照明$L$は以下の式で表される球面ガウス関数(SG)で表現される。

$$G(\boldsymbol{v};\boldsymbol{p},\lambda,\mu) = \mu e^{\lambda(\boldsymbol{v}\cdot\boldsymbol{p}-1)} \quad \text{(2)}$$

ここで$\boldsymbol{p}$は球面ガウス関数のローブの中心軸、$\lambda$は鋭さを表すパラメータ，$\mu$は球面ガウス関数の振幅をあらわす。また、方向$\boldsymbol{v}$は球面ガウス関数の球面上のパラメータであり、この関数の定義域に相当する。

可視関数$V$は、入射方向ベクトル$\boldsymbol{i}$が遮蔽されているかどうかを示す球面二値関数である。本研究では、Wang らの方法と同様に可視関数を spherical signed distance function (SSDF) に変換する。SSDF $V^d$は、以下のように与えられる。

$$V^d(\boldsymbol{i}) = \begin{cases} +\min\limits_{V(\boldsymbol{t})=0} \arccos(\boldsymbol{t}\cdot\boldsymbol{i}), if\ V(\boldsymbol{i})=1, \\ -\min\limits_{V(\boldsymbol{t})=1} \arccos(\boldsymbol{t}\cdot\boldsymbol{i}), if\ V(\boldsymbol{i})=0 \end{cases} \quad \text{(3)}$$

ここで$\boldsymbol{t}$は入射ベクトル$\boldsymbol{i}$に最も近い遮蔽境界への方向ベクトルである（図１）。SSDF は、方向$\boldsymbol{i}$が遮蔽されていない場合、最近傍の影境界への方向ベクトルとのなす角を返し、方向$\boldsymbol{i}$が遮蔽されている場合には、最近傍の遮蔽されていない方向ベクトルのなす角にマイナスをかけたものを返す関数である。

式(1)を用いて位置$\boldsymbol{x}$における輝度を計算するためには、照明や余弦項を近似する球面ガウス関数と可視関数との積の積分を計算する必要がある。球面ガウス関数$G(\boldsymbol{i};\boldsymbol{p},\lambda,\mu)$と可視関数の積の積分は、球面ガウス関数の主軸$\boldsymbol{p}$を SSDF に代入して得られる角度$\theta_d = V^d(\boldsymbol{p})$を用いて以下の式で計算される。

$$\int_{\Omega} G(\boldsymbol{i};\boldsymbol{p},\lambda,\mu)\cdot V(\boldsymbol{i})\, d\boldsymbol{i} \approx \mu f_h(\theta_d,\lambda) \quad \text{(4)}$$

ここで$f_h$は以下の式で表される。

---

Fast Rendering of Dynamic Scenes under All-frequency Lighting by using Spherical Gaussians

†Wataru Furuya・Wakayama University Faculty of Systems Engineering

‡Kei Iwasaki・Wakayama University Faculty of Systems Engineering

$$f_h(\theta_d,\lambda)=\left\{\frac{1.05}{1+e^{k_\lambda\theta_d}}+\frac{0.05}{2}\right\}\left\{\frac{2\pi}{\lambda}\left(1-e^{-\lambda}\right)\right\}\quad\text{(5)}$$

$$k_\lambda\approx 0.204\lambda^3-0.892\lambda^2+2.995\lambda+0.067\quad\text{(6)}$$

すなわち、位置$\boldsymbol{x}$における輝度を求めるためには、各球面ガウス関数の主軸$\boldsymbol{p}$における SSDF の値を求める必要がある。Wang らの手法ではこの SSDF は事前計算において計算していたため静的なシーンに限定されていた。

そこで、提案手法ではスクリーン空間内で SSDF をカメラからの深度値、光源からの深度値を元にレンダリング時に計算することで動的なシーンにおいても高速な描画を実現する。

### 3.1. 各パラメータのテクスチャ化

SSDF を計算するために、テクスチャに各種情報をレンダリングする。まず、カメラ位置から見たシーンの情報をテクスチャに保存し、SSDF の計算パスやその後のレンダリングパスへ渡す。テクスチャに保存される情報は以下の通りである。

- 座標値、法線、色（材質、テクスチャ）
- 裏面をカリングした場合の深度値
- 表面をカリングした場合の深度値
- 球面ガウス関数の中心軸ベクトル$\boldsymbol{p}$

なお、中心軸ベクトル$\boldsymbol{p}$は拡散反射成分と鏡面反射成分の二種類が保存される。つぎに、照明$\boldsymbol{L}$ を表現する各球面ガウス関数の主軸方向にカメラの方向をセットし、深度値、座標値（ワールド座標系）をテクスチャにレンダリングする。さらに、SSDF を効率的に計算するため、この深度テクスチャのミップマップピラミッドを作成する。なお、通常のミップマップが各ピクセルの平均値を格納するのに対して、提案手法では 4 ピクセルの最小値を次のレベルのミップマップのピクセル値とする。カメラからの深度値を記録したテクスチャをカメラデプスマップ、光源からの深度値を記録したものをライトデプスマップと呼ぶことにする。

### 3.2. SSDF の計算

SSDF の計算パスでは、位置$\boldsymbol{x}$における SSDF 値を求めるため、さきに算出してテクスチャに記録しておいた球面ガウス関数の中心ベクトル$\boldsymbol{p}$を取得して、スクリーン空間に投影し、$\boldsymbol{x}$から方向$\boldsymbol{p}$へ向かうレイが遮蔽されているかどうかを、カメラデプスマップを用いて判定する。式(3)で示されているように、$\boldsymbol{p}$が遮蔽されているならば、遮蔽境界の内、最も近傍の遮蔽されていないベクトルを、そうでないならば、逆に最も近傍の遮蔽されているベクトルを境界ベクトル$\boldsymbol{t}$として求める。

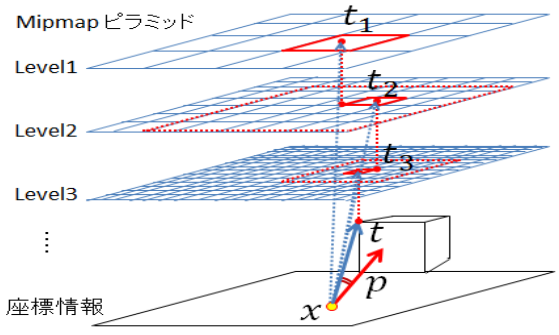


**図 2　境界ベクトル$\boldsymbol{t}$のミップマップによる探索**

境界ベクトル$\boldsymbol{t}$は、図 2 のようにライトデプスマップのミップマップピラミッドを用いて求める。境界ベクトル$\boldsymbol{t}$は候補となる多数のベクトルの中から探索する必要がある。よって、この処理では探索範囲を限定するため、まず、ミップマップ化したライトデプスマップの低解像度レベルのものを使用し画像全体を探索範囲として大まかに境界ベクトルの候補座標$\boldsymbol{t_1}$を決定する。次に、一段解像度を上げたライトデプスマップを使用し、候補座標を中心に 6×6 の近傍ピクセルを探索し次の候補$\boldsymbol{t_2}$を絞り込む。これを、最大解像度まで繰り返し、最終的に求められた座標位置に記録されているワールド座標系の座標値と位置$\boldsymbol{x}$とのベクトルを境界ベクトル$\boldsymbol{t}$として SSDF 値を計算する。求めた SSDF の値は、テクスチャにレンダリングする。

### 3.3. レンダリング

レンダリングパスでは、先ほど求めた SSDF を使用し Wang らの手法[2]を用いてレンダリングする。ここでは説明簡略化のため拡散反射についての計算法について述べる。照明 L を少数の球面ガウス関数の線形和で表現する。余弦項も 1 つの球面ガウス関数で表現すると、照明と余弦項の積は球面ガウス関数の線形和で表現されるため、各球面ガウス関数と可視関数の積の積分を式(4)から求め、足しあわせることによって$\boldsymbol{x}$の輝度を計算する。

## 4. 結果

本手法でレンダリングされた結果画像を図 3 に示す。計算環境は、Intel Core i7 CPU 920 2.67GHz、3GB RAM、NVIDIA GeForce GTX 470 を搭載した PC である。SSDF の算出及び、直接照明の輝度計算は GPU 上で GLSL を用いて行った。描画画面サイズは 256×256 ピクセル、ミップマップサイズは 256、128、64、32 の 4 つを用いた。描画更新レートは 1fps であり、動的なシーンにおける高速な描画を達成できた。今後の課題として更なる高速化や変形物体への対応などが挙げられる。

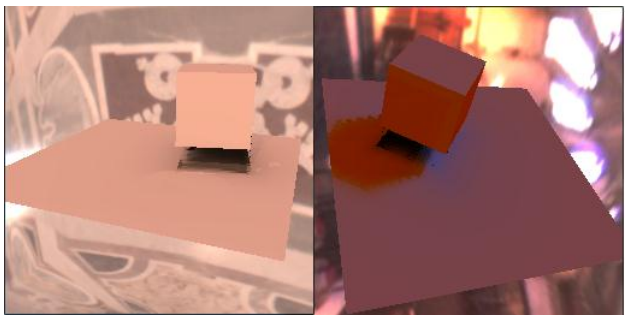

**図 3　提案手法でのレンダリング結果**

## 参考文献

[1]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.
[2]J. Wang et al., All-frequency rendering of dynamic, spatially-varying reflectance, ACM Transactions on Graphics, Vol. 28, No. 5, 2009.
[3]K. Zhou et al., Precomputed shadow fields for dynamic scenes, ACM Transactions on Graphics, Vol. 24, No. 3, 2005.
[4]Z. Ren et al., Real-time soft shadows in dynamic scenes using spherical harmonics exponentiation, ACM Transactions on Graphics, Vol. 25, No. 3, 2006.